Exploitation du logiciel HELP sous Multics : mode d'emploi des commandes principales

Patrick Robert

CENTRE NATIONAL D'ETUDES
DES TELECOMMUNICATIONS

CENTRE NATIONAL DE LA
RECHERCHE SCIENTIFIQUE

CENTRE DE
RECHERCHES
EN PHYSIQUE DE
L'ENVIRONNEMENT
TERRESTRE
ET PLANETAIRE

CRPE

NOTE TECHNIQUE
CRPE / 131

EXPLOITATION DU LOGICIEL HELP SOUS MULTICS MODE D'EMPLOI DES COMMANDES PRINCIPALES

Par

P. ROBERT CRPE / OPN / CNET
92131 Issy - les - Moulineaux

CENTRE NATIONAL D'ETUDES
DES TELECOMMUNICATIONS
Centre PARIS B

CENTRE NATIONAL DE LA
RECHERCHE SCIENTIFIQUE
Département TOAE

CENTRE DE RECHERCHE EN PHYSIQUE DE
L'ENVIRONNEMENT TERRESTRE ET PLANETAIRE

NOTE TECHNIQUE CRPE/131

EXPLOITATION DU LOGICIEL HELP SOUS MULTICS
MODE D'EMPLOI DES COMMANDES PRINCIPALES

par

Patrick ROBERT

RPE/OPN

38-40 rue du général Leclerc
92131 ISSY-LES-MOULINEAUX,FRANCE

Le Directeur
Vu
I. REVAH

Le Directeur Adjoint
Vu
P. BAUER

Février 1984

EXPLOITATION DU LOGICIEL HELP SOUS MULTICS

MODE D'EMPLOI DES COMMANDES PRINCIPALES

RESUME

Dans ce document les segments d'information relatifs aux principales commandes MULTICS, disponibles individuellement par le système de documentation en ligne "help", sont organisés par catégorie, mis en forme et listés.

Les commandes exploitées (200 environ) dont on a ainsi le mode d'emploi complet sont extraites du "Guide de poche MULTICS, Commandes et fonctions actives" (CII-HB,68F2AW17) et du cours de C. Davoust (NT/PAA/ATR/PIT669).

Ce document est complété en dernière partie par une table donnant pour toutes les commandes exploitables par "help" la liste de leur abréviation et/ou synonymes, le nombre de lignes du segment d'information correspondant, et l'information indiquant si la commande est également une fonction active.

Document édité sur l'imprimante a laser XEROX 9700 de PIT

EXPLOITATION DU LOGICIEL HELP SOUS MULTICS

MODE D'EMPLOI DES COMMANDES PRINCIPALES

PLAN

I- INTRODUCTION

Ce document est le résultat de l'exploitation du système de documentation en ligne "help" disponible sous le système MULTICS.

Il a été mis en oeuvre sur le DPS8 de CII HONEYWELL BULL implanté au CNET à Issy les Moulineaux.

L'exploitation de ce système de documentation se fait par l'intermédiaire d'un programme Fortran créant un fichier de commandes spécifiques dont l'exécution en processus absentee effectue les reqûetes successives au logiciel interactif "help".Les résultats sont ensuite mis en forme au moyen d'un autre programme approprié.

Cet ensemble de manoeuvres est lui meme automatisé en un seul fichier de commandes.

L'entrée de ce système est un fichier de données où sont rangées les commandes dont on désire le libellé complet et le mode d'emploi.

La sortie de ce système est un fichier dont le contenu est ce propre document.

Ce document est lui même édité directement par le listage de ce fichier sur l'imprimante à laser du CNET,selon des critères standards d'impression.

La liste des commandes référenciées dans le fichier d'entrée est celle du "guide de poche MULTICS,commandes et fonctions actives" (CII-HB,68F2AW17) complétée d'autres commandes utiles extraites du cours de C.Davoust (NT/PAA/ATR/PIT669).

Elle n'est pas exhaustive,mais regroupe néanmoins les 200 commandes et/ou fonctions actives les plus utilisées parmi les quelques 1000 disponibles.

Elle a volontairement été limitée afin de garder une taille convenable à ce document qui se veut une version plus légère et sensiblement différente du "Multics Programmers' Manual - Commandes and Active Functions" (Doc.CII-HB 68A2AG92).

Ce document est complété en dernière partie par une table donnant la liste alphabétique complète de toutes les commandes MULTICS exploitables par "help", avec,pour chaque commande,la liste de ses abréviations ou synonymes,le nombre de lignes du segment d'information correspondant,et l'information indiquant si cette commande est également une fonction active.

II- LISTE PAR CATEGORIE DES COMMANDES PRINCIPALES

manipulation de segments et de directories

Accès au système,environnement,documentation

compilation et exécution fortran

controle d'accès et règles de recherche

messagerie

système absentee

input et output

controle des ressources

éditeurs et traitement de texte

dates et temps

démons

demandes de restauration

III- LISTE ALPHABETIQUE DES COMMANDES PRINCIPALES

IV- LIBELLE,SYNTAXE,FONCTION ET MODE D'EMPLOI DES COMMANDES PRINCIPALES

abbrev

03/04/76 abbrev, ab

Syntax: ab

Function: expands abbreviations in command lines. Each command line typed to the system is broken up into substrings between break characters. Substrings found in the user's abbreviation profile in the home directory are replaced by their expansions. Substrings within double quotes are not expanded.

List of requests: In the descriptions below, an abbreviation <abbr> can be a maximum of 8 characters and cannot contain any break characters. See "Break characters".

.a <abbr> <rest of line>
   add an abbreviation that is expanded regardless of line position; if already defined, query user.

.ab <abbr> <rest of line>
   add an abbreviation that is expanded only when at beginning of line or following a semicolon.

.af <abbr> <rest of line>
   same as .a without user query.

.abf <abbr> <rest of line>
   same as .ab without user query.

.d <abbr1>...<abbrN>
   delete abbreviations.

.l <abbr1>...<abbrN>
   list definitions of abbreviations; if none specified, list all.

.la <letter1>...<letterN>
   list all abbreviationss beginning with letter(s); each letter argument can only be a single character.

.q
   quit expanding abbreviations.

. <rest of line>
   do not expand abbreviations in this line.

.p
   print out profile segment being used.

.u path
   use the profile segment specified by path; profile suffix must be given.

.r
   enter mode that remembers command line after expansion.

.f
enter mode that forgets command line after expansion.

.s <rest of line>
show <rest of line> expanded but do not execute. If in remember mode and <rest of line> is not specified, the last line expanded is shown.

Break characters:

horizontal tab	semicolon
vertical tab	formfeed
newline	vertical bar
space	parentheses
quote	less than
dollar sign	greater than
apostrophe	brackets
grave accent	braces
period	

Abbreviations cannot contain other abbreviations; see the do command.

absentee

10/01/80 Absentee facility

A facility for requesting absentee processes is available to users. A user can request that a process be created which executes commands from a segment and places its output into a segment.

To request an absentee computation, one first constructs an absentee control segment which is similar in syntax to an exec_com segment. The absentee process (when it is created for the user) will read from this control segment. The suffix of the control segment must be ".absin". Then a command (enter_abs_request) is issued that actually requests that an absentee process be created on behalf of the user. The output of this absentee process goes into an absentee output segment. The name of this output segment can be specified in the enter_abs_request command. If the name is not specified, then the pathname of the control segment is used, except that its suffix is .absout. The user can delay the creation of the absentee process until after a specified time by means of the "-time" control argument to the ear command. If this option is not selected, at an arbitrary time in the future an absentee process is created for the requestor. Type "help enter_abs_request" or "help ear" for further discussion of this command.

The resulting process is identical to an interactive process except that:
1) read operations from user_input are done from the absentee input segment.
2) write operations to user_output are directed to the absentee output segment.
3) special condition handlers are established for record_quota_overflow and cput.
4) any error intercepted by the standard unclaimed signal handler, except for command_error and command_question, logs out the absentee process.

Two other commands are installed as part of the absentee facility:
1) list_abs_requests (lar) - a command that gives the user information on the requests for absentee processes that the user has made. Type "help list_abs_requests" or "help lar" for more information.
2) cancel_abs_request (car) - a command that can be used to delete a request for an absentee process. For further details, type "help cancel_abs_request" or "help car".

Examples:
Suppose that a user wants to request an absentee computation to perform an off-line compilation. The user creates a control segment called absentee_pl1.absin containing:

```
cwd current
pl1 x -table -source -symbols
dp -dl x.list
```

```
logout
```

The command line:

```
enter_abs_request absentee_pl1.absin
```

causes an absentee process to be created (some time in the future) that:
1) sets the working directory to a directory named current inferior to the users's default working directory.
2) compiles a pl1 program named x.pl1 with three control arguments
3) dprints 1 copy of the list segment.
4) logs out.

The output of these tasks appears in the same directory as absentee_pl1.absin in a segment called absentee_pl1.absout.

Notes:
1) The enter_abs_request command checks for the existence of the absentee control segment and refuses the request if it is not present.
2) An absentee process can be requested only for the Person_id and Project_id of the user submitting the request.
3) The facility is designed so that more than one absentee process can run at one time. The user should take care, when submitting several requests that use the same control segment, that the output of each request is directed to a different output segment (see enter_abs_request -output_file).
4) There can be both an interactive and an absentee process for the same user at the same time.
5) The who command denotes absentee users by placing an asterisk directly after person.project, for example "Green.Multics*".

6) The cancel_abs_request command can cancel a request for an absentee process that is already logged in.
7) The user can ask operations to bump or to cancel an absentee process. The difference is as follows. Bumping destroys the absentee process but allows the computation to begin again. Cancelling an absentee process prevents it from ever being restarted. This distinction is relevant only if the absentee computation was declared to be restartable via the "-restart" ("-rt") control argument of the ear command. The user who contacts operations to destroy an absentee process should be sure to specify which function is wanted.
8) The new_proc command is an undefined command in an absentee process. It results in the termination of the absentee process.
9) For an absentee process to end properly, logout should be the last command encountered in the absentee control segment. If this condition is not met, an error message (indicating that the input is exhausted) is printed.

10) The absentee control segment should not be edited or its bit count changed during the course of the absentee process. This action causes unpredictable results.
11) Since the syntax of the absentee control segment is the same as an exec_com segment, the user should be aware of a few deviations. Certain exec_com requests are ignored in an absentee environment. Currently these are:

1) &attach
2) &detach
3) &command_line
4) &ready

The reasons for these differences are:

1 & 2) Input is already attached to the absentee input segment.
3) In an absentee process, command lines cannot be distinguished from input lines.
4) Unlike exec_com, control of the ready message can be achieved only by the ready_on and ready_off commands. All other control requests work normally.

12) The absentee facility provides a number of priority queues. The absentee commands (ear, lar, car) have a "-queue" control argument that allows the user to specify the particular queue desired. There are four queues. Site administrators can control the default queue used to submit requests when the "-queue" control argument isn't given to ear, pl1_abs, etc., the cost of using each queue, scheduling parameters for absentee processes in each queue, and the lowest priority queue serviced on each shift.

13) The answering service enforces a limit stop (defined by the installation) on the cpu time that can be used by an absentee process. A user is able to specify a per-job time limit less than or equal to this maximum. Specification of a time limit causes a cpu timer to be established in the absentee process. Resetting all cpu timers makes the limit ineffective.
14) A user cannot convert his interactive process to an absentee process, nor his absentee process to an interactive one.
15) If a record quota overflow occurs during the execution of an absentee process, in some cases the end of the absentee output segment can be overwritten with a short message.
16) In an absentee process, cu_$set_cl_intermediary is invoked to set the procedure called by the standard unclaimed signal handler after outputting diagnostics. Thus, after getting a signalled error (except "command_error" or "command_question"), the standard unclaimed signal handler passes control to a procedure equivalent to logout.

17) An argument is passed to start_up.ec to indicate which type of process is being created. Type "help start_up" for further details.
18) A resetread on user_input results in the termination of the absentee process. Procedures currently performing a resetread when handling errors include the following:

basic
debug
edm
qedx

accept_messages

09/17/81 accept_messages, am

Syntax: am address -control_args

Function: initializes or reinitializes the user's process for accepting both messages that are sent by the send_message command and notifications of the form "You have mail." that are sent by the send_mail command.

Arguments:

address
is the address of a mailbox. If no address is specified, the user's default mailbox is assumed. The mailbox must be specified in one of the following forms:

STR
is any argument that does not begin with a minus sign (-). If it contains either of the characters > or < it is interpreted as a mailbox pathname (the .mbx suffix is added if not present); otherwise it is interpreted as a User_id.

-pathname PATH, -pn PATH
specifies the pathname of the mailbox. The .mbx suffix is assumed if it is not present.

Control arguments:

-brief, -bf
prevents accept_messages from informing the user that it is creating a mailbox, and prints messages in short format.

-call cmdline
when the message is received, instead of printing it in the default format, accept_messages calls the command processor with a string of the form:

cmdline number sender time message path

where:

cmdline
is any Multics command line; cmdline must be enclosed in quotation marks if it contains blanks or other command language characters.

number
is the sequence number of the message, assigned when the -hold control argument is used; otherwise, number is 0.

sender
is the User_id of the person who sent the message.

time
is the date-time the message was sent.

message
is the actual message sent.

path
is the pathname of the mailbox to which the message was sent. If the message was sent to the default mailbox, path is omitted.

To reverse the effect of a previously specified -call control argument, the user can specify the -call control argument with no cmdline argument.

-flush DT
discards messages sent before the specified date-time, where DT is a string acceptable to the convert_date_to_binary_ subroutine (described in the MPM Subroutines). This control argument is intended to be used by operators and consultants.
-hold, -hd
holds messages until explicitly deleted by the delete_message command. Messages printed when the -hold control argument is in effect are preceded by an identifying number.
-long, -lg
precedes every message printed by the sender's Person_id and Project_id. This is the default.
-no_hold, -nhd
reverts the -hold control argument.

-prefix STR
places STR in front of all messages printed as they are received. STR can be up to 12 characters long and can contain the ioa_ control strings ^/, ^| and ^- if desired.
-print, -pr
prints all messages that were received since the last time the user was accepting messages.
-short, -sh
precedes consecutive messages from the same sender by "=:" instead of the Person_id and Project_id.
-time N, -tm N
prints undeleted messages every N minutes, preceded by a message of the form "You have X messages" where X is the number of undeleted messages. If N equals 0, time mode is reset.

Notes: The user should not give conflicting control arguments in the same invocation of the command (i.e., -long and -short or -long and -brief).

acl_matching

02/27/76 acl_matching

The strategy for matching an access control name argument is defined by three rules:

A literal component, including "*", matches only a component of the same name.

A missing component not delimited by a period is treated the same as a literal "*" (e.g., "*.Multics" is treated as "*.Multics.*").
Missing components on the left must be delimited by periods.

A missing component delimited by a period matches any component.

Examples:

..* matches only the literal ACL entry "*.*.*".
Multics matches only the ACL entry "Multics.*.*".
(The absence of a leading period makes Multics the first component.)
JRSmith.. matches any ACL entry with a first component of JRSmith.
.. matches any ACL entry.
. matches any ACL entry with a last component of *.
"" (null string) matches any ACL entry ending in ".*.*".

acquire_resource

06/17/81 acquire_resource, aqr

Syntax: aqr type STR1 ... STRn -control_args
or: aqr type -number N -control_args

Function: selects a resource of a given type from a free pool of all such resources, and makes the user the accounting owner of the resource. The accounting owner is given full control over the access rights for all users of the resource, as well as control over many parameters of the resource. Ownership of the resource is terminated via the release_resource command.

Arguments:

type
is a resource type defined in the resource type description table (RTDT).

STRi
is the unique identifying name of the particular resource being acquired. If STR looks like a control argument (i.e., if it is preceded by a hyphen), then it must be preceded by -name or -nm. If name is not supplied, a resource is chosen to satisfy the constraints imposed by the control arguments given (if any).

Control arguments:

-access_class accr, -acc accr
sets the initial AIM access class parameters where accr is an access class range. Users at any authorization within the access class range inclusive are allowed to read and write to the resource (provided they also meet other access requirements).

-acs_path path
specifies the pathname of the access control segment (ACS) for this resource. The ACS is not created by this command, but must be created by the owner, and the desired access control list set. If the ACS does not exist or is not specified, the default access is rew to the accounting owner, and null to all others.

-alloc STR
sets the allocation state of the resource to free or allocated, where STR must be either "on" or "off". If this control argument is not given, the allocation state is free. on sets the allocation state to allocated; off sets the allocation state to free.

-attributes STR, -attr STR
specifies that the resource chosen must possess the potential attributes specified in STR. When a satisfactory resource is located, the current attributes are set to a proper combination of these attributes (see "Notes" below).

-comment STR, -com STR
specifies the desired value of the comment string for this resource. STR can be either "on" or "off". on resources may only be released by privileged process. off resources may be released by owner.

-lock STR
   locks or unlocks the resource, preventing or allowing use of that resource, where STR must be either "on" or "off". on prevents any of the resource; off allows use of the resource (off is the default).

-number N, -nb N
   specifies that the number of such resources to be acquired is N. If this control argument is not given, 1 is assumed. This control argument may only be specified if a name is not given.
-owner STR, -ow STR
   specifies that this is an acquisition on behalf of the user specified by STR. If STR is given as "system", then the resource is assigned to the system pool. If STR is given as "free", then the resource is acquired to the free pool (effectively the same as no -owner). If STR is of the form Person_id.Project_id (where neither Person_id nor Project_id may be a star), the user specified has all the rights of ownership to the resource as if he had acquired it personally, except that if -release_lock on is specified, the owner may not release (give up ownership of) the resource voluntarily.

-priv
   specifies that a privileged call is to be made to obtain the status of this resource (see "Access Restrictions" below).
-release_lock STR, -rll STR
   specifies whether this resource may be released by the owner, or may only be released by a privileged process (see "Access Restrictions" below), where STR must be either "on" or "off". If this control argument is not specified, the resource may be released by the owner (does not require special privilege).

Notes: This command acquires a resource for either the user issuing it (requestor) or the user specified by the -owner control argument. If the requestor is registered on more than one project and needs corresponding access, or other users (on any project) need access to acquire a resource, the requestor must create or modify the access control segment (ACS). The requestor must then specify the new/modified ACS by issuing this command using the -acs_path control argument. The User_id, a Person_id.Project_id pair, specifies the user to be added to or deleted from the ACS.

Access Restrictions: The use of the -owner, -release_lock, or -access_class control arguments requires execute access to the rcp_admin_ gate.

add_name

01/15/76 add_name, an

Syntax: an path names

Function: adds an alternate name to an entry.

Arguments:
path
pathname of an entry. The star convention is allowed.
names
additional names to be added. The equals convention is allowed.

add_search_paths

12/03/80 add_search_paths, asp

Syntax:
```
   asp search_list search_path1  -control_args  ...
        search_pathN  -control_args
```

Function: adds one or more search paths to the specified search list.

Arguments:

search_list
   is the name of the search list to which the new search paths are added.

search_pathi
   specifies a new search path, where search_path1 is a relative or absolute pathname or a keyword (see "List of keywords" below).

Control arguments: are used only after the search_path argument. Only one is allowed for each search_path.

-after STR, -af STR
   specifies that the new search path is positioned after the search path denoted by STR.

-before STR, -be STR
   specifies that the new search path is positioned before the search path denoted by STR.

-first, -ft
   specifies that the new search path is positioned as the first search path in the search list.

-last, -lt
   specifies that the new search path is positioned as the last search path in the search list. This is the default.

List of keywords: The following are keywords accepted as search paths in place of absolute or relative pathnames.

-home_dir, -hd
-process_dir, -pd
-referencing_dir, -rd
-working_dir, -wd

Notes: In addition, a pathname can be specified with the Multics active function [user name] or [user project]. A search path enclosed in quotes is not expanded when placed in the search list. It is expanded when referenced in a user's process. This feature allows search paths to be defined that identify the process directory or home directory of any user.

If a link target does not exist, the search facility continues to search for a matching entryname.

List of related search facility commands:

delete_search_paths, dsp
print_search_path, psp
set_search_paths, ssp
where_search_paths, wsp

add_search_rules

05/22/81 add_search_rules, asr

Syntax: asr path1 -control_args ... pathN -control_args

Function: Adds pathnames and keywords to the search rules for object segments.

Arguments:
pathJ
   is the absolute or relative pathname of a directory, or one of the keywords listed below under "List of keywords".

Control arguments:
-after PATH, -af PATH
   appends the previous path argument after the existing search rule named by PATH.
-before PATH, -be PATH
   inserts the previous path argument before the existing search rule named by PATH.
-force, -fc
   deletes any old occurrence of path in the search rules before adding the new rule. The default is to fail and print an error message if the rule to be added already exists in a different position.
-no_force, -nfc
   fails and prints an error message if a rule to be added already exists in a different position. (Default)

List of keywords:
   Both pathJ and PATH arguments can be either pathnames or keywords. The defined keywords are--
initiated_segments
referencing_dir
working_dir

   In addition, PATH in control args can be:
home_dir
process_dir
any site-defined keywords

Notes: No warning is printed if a rule to be added already exists in the same position as that for which it is intended.

adjust_bit_count

10/16/79 adjust_bit_count, abc

Syntax: adjust_bit_count paths -control_args

Function: sets the bit count of segments by examining the contents.

Arguments:
paths
   are the pathnames of segments; star convention is allowed.

Control arguments:
-character, -ch
   set to the last nonzero character (default is last nonzero word).
-chase
   chase links when using the star convention. The default is to chase links only when specified without the star convention.
-long, -lg
   print a message when the bit count of a segment is changed, giving the old and new values.
-no_chase
   do not chase links when using the star convention. (Default)

Notes: This command should not be used on segments in structured files.

answer

07/13/81 answer

Syntax: answer STR -control_args command_line

Function: provides preset answers to questions asked by another command.

Arguments:

STR
is the desired answer to any question. If the answer is more than one word, it must be enclosed in quotes. If STR is -query, the question is passed on to the user. The -query control argument is the only one that can be used in place of STR.

command_line
is any Multics command line. It can contain any number of separate arguments (i.e., have spaces within it) and need not be enclosed in quotes.

Control arguments:

-brief, -bf
suppresses printing (on the user's terminal) of both the question and the answer.

-call STR
evaluates the active function string STR to obtain the next answer in a sequence. STR must be quoted if it contains command language characters. The surrounding brackets must be omitted, as in "segs *.pl1". The return value "true" is translated to "yes", and "false" to "no". All other return values are passed as is.

-match STR
answers only questions whose text matches STR. If STR is surrounded by slashes (/), it is interpreted as a qedx regular expression. Otherwise, answer tests whether STR is literally contained in the text of the question. Multiple occurrences of -match and -exclude are allowed (see Notes below). They apply to the entire command line.

-exclude STR, -ex STR
passes on, to the user or other handler, questions whose text matches STR. If STR is surrounded by slashes (/), it is interpreted as a qedx regular expression. Otherwise, answer tests whether STR is literally contained in the text of the question. Multiple occurrences of -match and -exclude are allowed (see Notes below). They apply to the entire command line.

-query
skips the next answer in a sequence, passing on the question to the user. The answer is read read from the user_io I/O switch.

-then STR
supplies the next answer in a sequence.

-times N
gives the previous answer (STR, -then STR, or -query) N times only

(where N is an integer).

Notes: Answer provides preset responses to questions by establishing an on unit for the condition command_question, and then executing the designated command. If the designated command calls the command_query_ subroutine (described in the MPM Subroutines) to ask a question, the on unit is invoked to supply the answer. The on unit is reverted when the answer command returns to command level. See "List of System Conditions and Default Handlers" in the MPM Reference Guide for a discussion of the command_question condition.

If a question is asked that requires a yes or no answer, and the preset answer is neither "yes" nor "no", the on unit is not invoked.

The last answer specified is issued as many times as necessary, unless followed by the -times N control argument.

The -match and -exclude control arguments are applied in the order specified. Each -match causes a given question to be answered if it matches STR, each -exclude causes it to be passed on if it matches STR. A question that has been excluded by -exclude is reconsidered if it matches a -match later in the command line. For example, the command line:

```
answer yes -match /fortran/ -exclude /fortran_io/ -match /^fortran_io/
```

answers questions containing the string "fortran", except that it does not answer questions containing "fortran_io", except that it DOES answer questions BEGINNING with "fortran_io".

archive

01/12/81 archive, ac

Syntax: ac key archive_path paths

Function: combines an arbitrary number of separate segments into one single segment.

Arguments:

key
is one of the functions listed below under "List of Keywords." The key functions are listed according to their operation.

archive_path
is the pathname of the archive segment to be created or used. The archive suffix is added if the user does not supply it. The star convention can be used with extraction and table of contents operations.

paths
are the components to be operated on by table of contents and delete operations. The star and equal conventions cannot be used.

List of keywords:
key functions are listed below according to their operation.

Table of Contents Operation--

t
print the entire table of contents if no components are named by the path arguments; otherwise print information about the named components only. Title and column headings are printed at the top.

tl
print the table of contents in long form; operates like t, printing more information for each component.

tb
print the table of contents, briefly; operates like t, except that the title and column headings are suppressed.

tlb
print the table of contents in long form, briefly; operates like tl, except that the title and column headings are suppressed.

Append Operation--

a
append named components to the archive segment. If a named component is already in the archive, a diagnostic is issued and the component is not replaced. At least one component must be named by the path arguments.

ad
append and delete; operates like a and then deletes all segments that have been appended to the archive.

adf
append and force deletion; operates like a and then forces deletion of all segments that have been appended to the archive.

ca
copy and append; operates like a, appending components to a copy of the new archive segment created in the user's working directory.

cad
copy, append, and delete; operates like ad, appending components to a copy of the archive segment and deleting the appended segments.

cadf
copy, append, and force deletion; operates like adf, appending components to a copy of the archive segment and forcibly deleting the segments requested for appending.

Replace Operation--

r
replace components in, or add components to the archive segment. When no components are named in the command line, all components of the archive for which segments by the same name are found in the user's working directory are replaced. When a component is named, it is either replaced or added.

rd
replace and delete; operates like r, replacing or adding components, then deletes all segments that have been replaced or added.

rdf
replace and force deletion; operates like r and forces deletion of all replaced or added segments.

cr
copy and replace; operates like r, placing an updated copy of the archive segment in the user's working directory instead of changing the original archive segment.

crd
copy, replace and delete; operates like rd, placing an updated copy of the archive segment in the user's working directory.

crdf
copy, replace, and force deletion; operates like rdf, placing an updated copy of the archive segment in the user's working directory.

Update Operation--

u
update; operates like r except that it replaces only those components for which the corresponding segment has a date-time modified later than that associated with the component in the archive.

ud
update and delete; operates like u and deletes all updated segments after the archive has been updated.

udf
update and force deletion; operates like u and forces deletion of all updated segments.

cu
copy and update; operates like u, placing an updated copy of the archive segment in the user's working directory.

cud
copy, update, and delete; operates like ud, placing an updated copy

of the archive segment in the user's working directory.

cudf
copy, update, and delete force; operates like udf, placing an updated copy of the archive segment in the user's working directory.

Delete Operation--

d
delete from the archive those components named by the path arguments.

cd
copy and delete; operates like d, placing an updated copy of the archive segment in the working directory.

Extract Operation--

x
extract from the archive those components named by the path arguments, placing them in segments in the storage system. The directory where a segment is placed is the directory portion of the path argument. The access mode stored with the archive component is placed on the segment for the user performing extraction. If no component names are given, all components are extracted and placed in segments in the working directory. The archive segment is not modified.

xf
extract and delete force; operates like x, forcing deletion of any duplicate names or segments found where the new segment is to be created.

Notes: The table of contents operation and the extract operation use the existing contents of an archive segment; the other operations change the contents of an archive segment. A new archive segment can be created with either the append or replace operation. In each of the operations that add to or replace components of the archive, the original segment is copied and the copy is written into the archive, leaving the original segment untouched unless deletion is specified as part of the operation.

The star convention can be used in the archive segment pathname during extract and table of contents operations; it cannot be used during append, replace, update, and delete operations.

Each component of an archive segment retains certain attributes of the segment from which it was copied. These consist of one name, the effective mode of the user who placed the component in the archive, the date-time last modified, the bit count, and the date-time placed in the archive.

archive_sort

06/03/76 archive_sort, as

Syntax: as paths

Function: sorts components of an archive segment into ascending order by name using standard ASCII collating sequence.

Arguments:
paths
   are pathnames of archive segments; archive suffix need not be given.

Notes: The original archive segment is overwritten by the sorted archive segment.

archive_table

02/06/81 archive_table, act

Syntax: act archive_path starnames -control_args

Function: returns the names of specified archive components in a specified archive segment. Names are returned separated by single spaces.

Arguments:

archive_path
   is the pathname of an archive segment, with or without the archive suffix. The star convention is NOT allowed.

starnames
   are optional component names to be matched against names of archive components. The star convention IS allowed.

Control arguments:

-absolute_pathname, -absp
   returns full pathnames of archive components, of the form ARCHIVE_DIR>ARCHIVE_NAME::COMPONENT_NAME, rather than just the component names.

Notes: Invoked as an active function, names are returned requoted and separated by single spaces. Invoked as a command, archive_table prints one component name per line.

Syntax as active function:

```
       [act archive_path  starnames   -control_args ]
```

assign_resource

03/13/80 assign_resource, (ar)

Syntax: ar resource_type -control_args

Function: calls the resource control package (RCP) to assign a resource to the user's process.

Arguments:
resource_type
specifies the type of resource to be assigned. Currently, only device types can be specified. The -device control argument is used to name a specific device to assign. Other control arguments are used to specify characteristics of the device to be assigned. The following device type keywords are supported:
tape_drive
disk_drive
console
printer
punch
reader
special

Control arguments:
-device STR, -dv STR
specifies the name of the device to be assigned. If this control argument is specified, other control arguments that specify device characteristics are ignored. (See "Examples" below.) If the -long control argument (see below) is used in conjunction with this control argument, a message containing the name of the assigned device is printed on the user's terminal; otherwise, no message is printed.
-model N
specifies the device model number characteristic. Only a device that has this model number is assigned. In order to find the model numbers that are acceptable, use the print_configuration_deck command described in System Tools, Order No. AZ03.

-track N, -tk N
specifies the track characteristic of a tape drive. The value can be either 9 or 7. If this control argument is not specified and if the -volume control argument is not specified, a track value of 9 is used when assigning a tape device.
-density N, -den N
specifies the density capability characteristic of a tape drive. There can be more than one instance of this argument. A tape drive is assigned that is capable of being set to all of the specified densities. Note that the values permitted depend on the particular hardware on the system. The acceptable values for this argument are:
200

556
800
1600
6250

-train N, -tn N
specifies the print train characteristic of a printer.
-line_length N, -ll N
specifies the line length of a printer. Its value must be one that is found in the "line length" field of a printer PRPH configuration card. If this field is not specified on a printer PRPH configuration card, this device characteristic is ignored for this printer.
-volume STR, -vol STR
specifies the name of a volume. If possible, the device assigned is one on which this volume has already been placed. If this is not possible (e.g., the volume is on a device assigned to a process) any available, appropriate, and accessible device will be assigned.

-number N, -nb N
specifies the number of resources to assign. All of the resources assigned have the device characteristics specified by any other arguments passed to this command. If this control argument is not specified, one resource is assigned.
-comment STR, -com STR
is a comment string that is displayed to the operator when the resource is assigned. If more than one string is required, the entire string must be in quotes. Only printable ASCII characters are allowed. Any unprintable characters (also tabs or new lines) found in this string are converted to blanks.
-long, -lg
specifies that all of the device characteristics of the assigned device should be printed. If this argument is not supplied, only the name of the assigned device is printed.

-system, -sys
specifies that the user wants to be treated as a system process during this assignment. If this argument is not specified OR if the user does not have the appropriate access, then the RCP assumes that this assignment is for a nonsystem process.
-wait N , -wt N
specifies that the user wants to wait if the assignment cannot be made at this time because the resources are assigned to some other process. The value N specifies the maximum number of minutes to wait. If N minutes elapse and a resource is not yet assigned, an error message is printed. If N is not specified, it is assumed that the user wants to wait indefinitely.

-speed N
specifies the speed of a tape drive. The acceptable values depend on the particular hardware on the system and can be the following:
75
125
200

Notes: Currently, only device resources can be assigned. An assigned device still must be attached by a call to some I/O module. If a device is successfully assigned, the name of the device is printed. (If the user requests a specific device that is successfully assigned, the name of the device is not printed unless the user asks for it. See the -device and -long control arguments above.)

Examples: In the example below, the user issues the assign_resource command with the "tape_drive" keyword and the -model control argument. The system responds with the name of the assigned device.

```
! assign_resource tape_drive -model 500

   Device tape_04 assigned
```

In the next example, the user issues the assign_resource command with the "tape_drive" keyword and the -device and -long control arguments. The system responds with the name of the assigned device and the model number, track, density and speed characteristics.

```
! assign_resource tape_drive -device tape_05 -long

   Device tape_05 assigned
   Model     =  500
   Tracks    =  9
   Densities =  200 556 800 1600
   Speed     =  125
```

attach_audit

06/17/81 attach_audit, ata

Syntax: ata old_switch new_switch -control_args

Function: starts auditing. Moves the attachment of the specified switch to another switch. Attaches the first switch via audit_ to the second.

Arguments:
old_switch
   is the switch to be audited. (DEFAULT -- user_i/o)
new_switch
   is the dummy switch to receive old_switch's previous attachment.
   (DEFAULT -- audit_i/o.time)

Control arguments:
-pathname STR, -pn STR
   use STR as the audit file. (DEFAULT -- [homedir]>[date].audit)
-truncate, -tc
   truncate the audit file, if it already exists.
   (DEFAULT -- extend it.)
-modes STR
   set the modes on user_i/o using STR as the mode string.

Notes: If no arguments or control arguments are given, auditing is set up for user_i/o with a default audit file of [date].audit. Multiple uses on the same day are all logged, one after the other, in the same audit file. The attach_audit command sets the safety_switch "on" for the audit file, detach_audit turns the safety_switch off.

For more information on the audit facility, type:
  help audit_
  help audit_.gi
  help detach_audit
  help audit_editor
  help display_audit_file
  help new_audit.gi

*audit_*

03/20/81 - audit_ I/O module

The audit_ I/O module intercepts I/O activity on a given switch, allowing one to log and or edit this data.

Attach description:

```
audit_ switch_name -control_args
```

Arguments:

switch_name
: is the name of an I/O switch to inserted between the existing switch and its I/O module.

Control arguments:

-truncate, -tc
: truncate the old audit file, if it has the same name as the new one.

-pathname, -pn
: use this pathname as the audit file. The default pathname is [homedir]>[date].audit .

Modes operation:

audit_input
: audit input lines. (DEFAULT -- on)

audit_output
: audit output lines. (DEFAULT -- on)

audit_edit
: enable audit editing. Does not put the user in the audit editor, it only makes it possible to enter the editor. (DEFAULT -- on)

audit_meter
: write a metering stamp before each entry in the file. The stamp consists of the actual time of the metering, incremental cpu time since the last stamp, and the incremental page faults since the last stamp.

audit_file_size=n
: set the maximum number of records for the audit file to n. The file is treated as a circular buffer of n records. A file size of "unlimited" allows the audit file to grow indefinitely.

audit_trigger=x
: set the audit request trigger character to x.

audit_trace
trace all control and mode calls to the module. Mode trace entries are idntified by a TM tag, control trace entries are identified by a TC tag.

audit_truncate
truncate the audit file.

audit_transparent
turn off auditing of audit and audit edit requests, as well as their output.

audit_suspend
turn off all modes.

audit_use_editor_prompt
turn on prompting in the audit editor.

audit_editor_prompt_string=STR, audit_epstr=STR
set the audit editor prompt string to STR. The audit editor prompt has the default appearance "audit editor: " ,or if the number of recursive invocations of the editor is greater than 1, "audit editor(level N): ", where N is the depth of the current invocation. This string is used as an ioa_ control string, with the arguments being: a bit which is on if the level is greater than 1; and, the level. The default string is "^/audit editor^[(^d)^]:^2x".

The audit file:

The default audit file pathname is [homedir]>[date].audit . The default file_size is unlimited. If one has sufficient data logged, the audit file may become a multi-segment file. The first 10 bytes of the file contains the header, which is used by both the audit_ I/O module and the audit_editor.

The entry type identifiers are:

EL
edit line, returned from audit editor.

IC
result of a get_chars.

IL
result of a get_line.

M
metering data.

OC
result of a put_chars.

TC
control request trace.

TM
mode request trace.

Notes:

For information about the audit editor see audit_editor.info .

Audit requests:

The audit requests are always recognized when auditing is on. The three character request sequence is the trigger character followed by the desired request followed by a new line. The default trigger character is an exclamation mark ("!"). The requests are:

!.

print "audit" and which of input and output is being audited.

!?

print a brief description of available audit requests.

!e

enter the audit editor.

!E

enter the audit editor, with the input line processed as edit requests.

!a

abbrev expand the input line. See the MPM documentation on abbrev for more information.

!r

replay the input line. That is, display the input line without a new line. Further input up to the next new line is appended to the redisplayed input. This is the input line which is passed through the audit_ I/O module.

!t

instructs the audit_ module not to log the input line, i.e. to make it transparent.

!d

delete the line. It prevents the input line from ever being seen.

!n

no operation. The input line to which this is appended is simply passed through the audit_ module.

audit_.gi

03/20/81 - Auditing and editing I/O:

Auditing allows one to keep a record or log of activity on a particular I/O switch. Also, it allows editing of input. The simplest use of auditing is:

```
! attach_audit
```

This will set up auditing of input and output and enable audit editing. An audit file will be placed in the user's home directory with a two component entry name. The first component of the name is the date and the second component is the suffix ".audit".

Audit requests:

The audit requests are always recognized when auditing is on. The three character request sequence is the trigger character followed by the desired request followed by a new line. The default trigger character is an exclamation mark ("!"). The requests are:

!.
print "audit" and which of input and output is being audited.

!?
print a brief description of available audit requests.

!e
enter the audit editor.

!E
enter the audit editor, with the input line processed as edit requests.

!a
abbrev expand the input line. See the MPM documentation on abbrev for more information.

!r
replay the input line. That is, display the input line without a new line. Further input up to the next new line is appended to the redisplayed input. This is the input line which is passed through the audit_ I/O module.

!t
instructs the audit_ module not to log the input line, i.e. to make
it transparent.

!n
no operation. The input line to which this is appended is simply passed through the audit_ module.

!d
delete the line. It prevents the input line from ever being seen by the system.

Editor requests:

The audit editor requests are presented in two categories, familiar requests and special requests. The editor syntax is basically that of qedx. Any number of requests may be on the same line and spaces are ignored. Addressing, where appropriate, is done the same as in qedx with two notable exceptions. First, the "." is a request for self-identification rather than an indicator for the current address. Second, addresses are in terms of entries in the audit file rather than lines in a buffer. If the default search tag is in use, as is the case unless specifically defeated, the absolute entry number refers to the number of entries with the default search tag from the beginning of the file. Similarly, a relative entry address refers to the number of entries with the default search tag before or after the current address.

Familiar requests:

[[ADR1,]ADR2]p print

print the addressed entries.

s/REGEXP/STRING/ substitute

replace occurrences of REGEXP in the edit buffer with STRING.

ADR location

locate the addressed entry. If ADR is not followed by a request the edit buffer is printed. An ADR can contain an absolute entry reference at its beginning, relative addresses in any portion, and regular expressions in any portion. An absolute address is either a number or the dollar-sign (to indicate the last entry in the audit file.

..STRING execute

pass STRING to command processor and return to the audit editor.

= print current entry number

print the current entry number. This value is dependent on the current default search tag. If the default search tag changes, the current entry may also change.

q quit

quit from the editor without doing anything (i.e., returning any characters).

Special requests:

expand — expand abbrev

abbrev expand the edit buffer.

off — audit off

don't audit input and output in the editor.

on — audit on

audit the editor.

l — last returned line

address the last line returned by the audit editor.

r[STRING] — return line

return the rest of the request line, if non-null. Otherwise, return the edit buffer (without trigger sequence).

n — return new-line

returns a new-line character.

type — print type

print the audit entry type of the current position.

exec — execute

pass the edit buffer to the command processor and return to the audit editor.

d/STRING/ — default search tag

set the default search tag to STRING. If STRING is only one character, then only the first character of the tag is used to determine if an entry is seen (in counting entries and doing searches). If STRING is two characters, the match is done one both characters of the tag.

Request syntax and processing:

The square brackets in the request syntax above are to indicate the contained item is optional. The square brackets are not typed when entering the request. If execution of a request, in the audit editor, should fail for any reason, the processing of that request line is aborted, the user is informed of the failure, and a new request is prompted for. Note that this means the user is left in the editor when a problem is encountered executing a request line associated with a E audit request.

The audit editor may be entered recursively, and each level of the editor has its own memory for the last returned line from its level.

Examples:

```
To set up with default audit file in homedir;
          attach_audit
To set up with an audit file in the process_dir;
          attach_audit -pn [pd]>my_audit_file
To set the audit file to be a circular  file of 5 records;
          io modes user_i/o audit_file_size=5
To re-execute the last use of the pll command;
          </^pll/r!E
To execute the above command line again;
          lr!E
```

audit_editor

11/18/80 The audit_ editor.

The audit_ editor is invoked by typing the edit request when auditing. The edit request comprising a three character sequence;

trigger character || "e" or "E" || new-line

The default trigger character is "!".

Editor request list:

```
[[ADR,]ADR]p                    print
s/REGEXP/STRING/                substitute
ADR                             location
..STRING                        execute
q                               quit
:                               defeat default search tag
? (or .?)                       list editor requests
expand (or .expand)             expand abbrev
off (or .off))  audit off
on (or .on)  audit on
l (or .l)   last returned line
r[STRING] (or .r[STRING]) return line
n (or .n)   return newline
type (or .type)  print type
exec (or .exec)  execute edit line
d/STRING/ (or .d/STRING/) default search tag
=   print current entry number
```

Explanation of editor requests:

The audit editor requests are presented in two categories, familiar requests and special requests. The editor syntax is basically that of qedx. Any number of requests may be on the same line and spaces are ignored.

Addressing, where appropriate, is done the same as in qedx with two notable exceptions. First, the "." is a request for self-identification rather than an indicator for the current address. Second, addresses are in terms of entries in the audit file rather than lines in a buffer.

If the default search tag is in use, as is the case unless specifically defeated, the absolute entry number refers to the number of entries with the default search tag from the beginning of the file. Similarly, a relative entry address refers to the number of entries with the default search tag before or after the current address.

Addressing:

An address can consist of one or more of the following three types of address, the relative address, the absolute address, and the search address.

An absolute address refers to an entry by its entry number. This entry number is determined by counting, from the beginning of the file, the number of entries which match the default search tag. The use of a colon (":") means every entry is counted.

A relative address is a number preceded by either a "+" or a "-". It refers to the entry which is the specified number of entries with the default search tag before , "-", or after, "+", the entry currently in the edit buffer.

A search address is a regular expression which may be preceded by a less-than ("< "). A regular expression is a character string beginning and ending with a slash ("/"). A search address which is a regular expression alone refers to the next entry in the file after the one currently in the edit buffer, which contains a match for the regular expression.

A search address which comprises a regular expression preceded by a less-than, "< ", does a backward search for the first entry previous to the current entry containing a match for the regular expression. N is a positive integer, and /REGEXP/ is a regular expression. The three types of addresses and their variations are: N -N +N /REGEXP/ < /REGEXP/

Familiar Requests:

[[ADR1,]ADR2]p print

print the addressed entries.

s/REGEXP/STRING/ substitute

replace occurrences of REGEXP in the edit buffer with STRING.

ADR location

locate the addressed entry. If ADR is not followed by a request the edit buffer is printed. An ADR can contain an absolute entry reference at its beginning, relative addresses in any portion, and regular expressions in any portion. An absolute address is either a number or the dollar-sign (to indicate the last entry in the audit file.

..STRING execute

pass STRING to command processor and return to the audit editor.

q quit

return the current line with the trigger sequence appended.

= print current entry number

print the entry number associated with the current position in the audit file. The value of the entry number for the current position can change with different default search tags. See the ":" and "d" requests below.

Special requests:

: defeat default search tag

look at every entry, regardless of entry class (or tag). only effective for requests following it and on the same request line.

? (or .?) list editor requests

list the editor requests and a brief description of their function.

expand (or .expand) expand abbrev

abbrev expand the edit buffer.

off (or .off) audit off

don't audit input and output in the editor.

on (or .on) audit on

audit the editor.

l (or .l) last returned line

address the last line returned by the audit editor.

r[STRING] (or .r[STRING]) return line

return the rest of the request line, if non-null. Otherwise, return the edit buffer (without trigger sequence).

n (or .n) return new-line

returns a new-line character.

type (or .type) print type

print the audit entry type of the current position.

exec (or .exec) execute

pass the edit buffer to the command processor and return to the audit editor.

d/STRING/ (or .d/STRING/) default search tag

set the default search tag to STRING. If STRING is only one character, then only the first character of the tag is used to determine if an entry is seen (in counting entries and doing searches). If STRING is two characters, the match is done one both characters of the tag.

Notes:

The audit_ editor may be invoked while in the audit_ editor, if the editor is being audited. For every level of the editor, there is a remembered last returned line distinct from all other remembered last returned lines.

There is also a position in the audit file associated with the last returned line. This position is the location that the last returned line was recorded (this position exists since last returned lines are audited). The "l" request sets the current position to be this associated position. It is important to note that this position (or entry) is distinct from wherever the original copy of the last returned line(the one which was edited to produce the last returned line) was located.

bind

07/25/81 bind, bd

Syntax: bd paths -control_args

Function: produces a single bound object segment from one or more unbound object segments, stored in archive segments, which are called the components of the bound segment.

Arguments:

paths
   are the pathnames of archive segments containing one or more component object segments to be bound. The archive suffix is assumed. Up to 16 input archive segments can be specified. They are logically concatenated in a left-to-right order to produce a single sequence of input component object segments.

Control arguments:

-brief, -bf
   suppresses printing of warning messages.

-force_order, -fco
   is equivalent to including a Force_Order statement in the bindfile. Since the need to use Force_Order is often temporary, and caused by update archives which have had components deleted, this is preferable to using the Force_Order statement, since it need only be used while the temporary condition exists.

-force_update, -fud
   is similar in function to -update, except that the archive specified following -force_update need not exist. Any archive which does exist is treated the same way as for -update, and any which does not is simply ignored. This is useful for constructing abbrevs which bind archives which may or may not have update archives in various locations.

-list, -ls
   produces a listing segment whose name is derived from the name of the bound object segment plus a suffix of list. The listing segment is generated for the purpose of dprinting; it contains the bound segment's bind control segment (see "Notes on bindfile" below), its bind map, and that information from the bound object segment printed by the print_link_info command. This control argument cannot be invoked with -map. In the absence of the -list or -map control arguments, no listing segment is generated.

-map
   produces a listing segment (with the suffixes list and map) that contains only the bind map information. This control argument is incompatible with -list. In the absence of the -list or -map control arguments, no listing segment is generated.

-update paths, -ud paths

indicates that the following list of archive segments (paths) specifies update rather than input object segments. The archive suffix is assumed in paths. Up to a combined total of 16 input and update segments can be specified. The contained update object segments are matched against the input object segments by object segment name. Matching update object segments replace the corresponding input object segments; unmatched ones are appended to the sequence of input object segments. If several update object segments have the same name, only the last one encountered is bound into the bound segment.

Notes: Compilers and the assembler produce unbound object segments. Binding has three benefits: the reduction of storage fragmentation, the prelinking of external references between the components, and the reduction of size of address space necessary to execute the components.

Notes on output: The binder produces as its output two segments: an executable bound object segment and an optional, printable ASCII listing segment. The name of the bound segment is, by default, derived from the entryname of the first input archive segment encountered by stripping the archive suffix from it. The name of the listing segment is derived from the name of the bound segment by adding the list suffix to it. Use of the Objectname master statement in the bindfile (see "List of master keywords" below) allows the name of the bound segment to be stated explicitly. In addition, use of the Addname master statement in the binding instructions causes additional segment names to be added to the bound segment. The primary name of the bound segment must not be the same as the name of any component.

Notes on bindfile: The bindfile is a segment containing symbolic instructions that control the operation of the binder. Its entryname must contain the bind suffix and it must be archived into any one of the input archive segments.

In case two bindfiles are specified, one in an input archive segment and the other in an update archive segment, the latter takes precedence and a warning message is printed to that effect.

The syntax of the bindfile statements consist of a keyword followed by zero or more parameters and then delimited by a statement delimiter. Master statements pertain to the entire bound object segment; normal statements pertain to a single component object within the bound segment. Master statements are identified by master keywords that begin with a capital letter; normal keywords begin with a lowercase letter. A keyword designates a certain action to be undertaken by the binder pertaining to parameters following the keyword.

List of master keywords:

Objectname
   the parameter is the segment name of the new bound object.

Order
   the parameters are a list of objectnames in the desired binding order. In the absence of an order statement, binding is done in the order of the input sequence. The order statement requires that there be a one-to-one correspondence between its list of parameters

and the components of the input sequence.

Force_Order
same as Order, except that the list of parameters can be a subset of the input sequence, allowing the archive segments to contain additional segments that are not to be bound (e.g., source programs).

Global
the parameters can be either retain, delete, or no_link. The parameter selected pertains to all component object segments within the bound segment. A global or explicit statement concerning a single component object or a single external symbol of a component object overrides the Global statement for that component object or symbol.

Addname
the parameters are the symbolic names to be added to the bound segment. If Addname has no parameters, it causes the segment names and synonyms of those component objects for which at least a single entrypoint was retained to be added to the bound segment.

No_Table
does not require parameters. It causes the symbol tables from all the component symbol sections containing symbol tables to be omitted from the bound segment. If this keyword is not given, all symbol tables are kept.

Perprocess_Static
does not require parameters. It causes the perprocess_static flag of the bound segment to be turned on, which prevents the internal static storage from being reset during a run unit.

If no bindfile is specified, the binder assumes the following default parameters:

```
Objectname:  segment name of the first input archive file.
Global:      retain;  /*regenerate all definitions*/
```

List of normal keywords:

objectname
the single parameter is the name of a component object as it appears in the archive segment. The objectname statement indicates that all following normal statements (up to but not including the next objectname statement) pertain to the component object whose name is the parameter of the objectname statement.

synonym
the parameters are symbolic segment names declared to be synonymous to the component object's objectname.

retain
the parameters are the names of entrypoints defined within the component object segment that the user wishes to retain as entrypoints of the bound object segment.

delete
the parameters are the names of entrypoints defined within the component object segment that the user does not wish to be retained as entrypoints of the new bound segment.

no_link
the parameters are the names of entrypoints that are NOT to be

prelinked during binding. The no_link statement implies a retain statement for the specified names.

global
the parameter can be either retain, delete, or no_link. The parameter selected becomes effective for all entrypoints of the component object. An explicit retain, delete, or no_link statement concerning a given entrypoint of the component object overrides the global statement for that specific entrypoint. A global no_link causes all external references to the component object to be regenerated as links to entrypoints; this allows execution-time substitution of such a component by a free standing version of it, for example for debugging purposes.

table
does not require parameters. It causes the symbol table for the component to be retained and is needed to override the No_Table master keyword, described above.

List of bindfile delimiters:

:
keyword delimiter used to identify a keyword followed by one or more parameters. A keyword that is followed by no parameters is delimited by a statement delimiter.

;
statement delimiter.

,
parameter delimiter. The last parameter is delimited by a statement delimiter.

/*
begin comment.

*/
end comment.

Notes on error messages: The binder produces three types of error messages. Messages beginning with the word "Warning" do not necessarily represent errors, but warn the user of possible inconsistencies in the input components or bindfile. Messages beginning with the word "binder_" normally represent errors in the input components. Errors detected during the parsing of the bindfile have the format--

Bindfile Error Line #N ....

where N is the line number of the erroneous statement. If an error is detected during parsing, the binder aborts because it cannot bind according to the user's specifications.

calendar

01/05/81 calendar

Syntax: calendar paths -control_args

Function: prints a calendar page for one month. The preceding and following months are also shown.

Arguments:
paths
   are the pathnames of segments that contain a list of events in the form of text to be inserted into the calendar.

Control arguments:
-date DATE, -dt DATE
   identifies which month is printed. This argument must be a date acceptable to the convert_date_to_binary_ subroutine (described in the MPM Subroutines). If the -date control argument is not given, the current month is printed.
-fiscal_week, -fw
   labels boxes with fiscal week numbers.
-wait, -wt
   causes the command to wait for a single newline character from the user before printing the calendar.
-stop, -sp
   causes the command to wait for a single newline character from the user before printing the calendar and after printing it.

-force, -fc
   causes a calendar to be printed regardless of errors in the input files.
-box_height HEIGHT, -bht HEIGHT
   changes the height of each calendar box from 7 lines to HEIGHT lines. If HEIGHT < 7, calendars for previous and following months do not appear in margin.
-julian, -jul
   prints "julian dates" in bottom line of each box -- the number of days from the beginning of the year and the number of days remaining in the year.

Notes on Output: Each box for a calendar day is 16 characters wide. and 7 lines high unless otherwise determined by the -box_height control argument. Each box in the calendar contains the number of the day of the month; other information can also appear in the box, at the user's option. The month preceding the specified month and the month following it are also printed.

Notes on Input: Each segment contains lines that set up a string to be inserted into the appropriate box of the calendar. The fields in

these lines are separated by commas and have the form--

opcode,dtfield,...,dtfield,text

The first field is the operation code (either date, rel, repeat, easter or rename). The second and succeeding fields depend on which operation code is used. Lines that produce a date not in the current month are ignored.

List of operation codes:

date

has the following syntax-- date,DT,TEXT
DT is the date for which the following text is to be inserted.
TEXT is arbitrary text up to 16 characters long.

rel

allows a note to be inserted for a day which is calculated relative to the beginning of a month. Its syntax is as follows--
rel,MONTHNO,RELDT1,RELDT2,TEXT
MONTHNO is a one or two digit number from 1 to 12 indicating the month from which the event is to be calculated, or can be -1, 0 or +1. -1 indicates the month previous to the printed month, 0 refers to the month being printed, and +1 indicates the month after the printed month.
RELDT1 is a date converted relative to the day before the beginning of the specified month.
RELDT2 is a date which is converted relative to the date indicated by the RELDT1 of the third field. It specifies the date selected for the insertion of the TEXT.
TEXT is arbitrary text.

repeat

inserts a note into the boxes for several days which are separated by a constant interval of time. The syntax is as follows--
repeat,STARTDT,END_OR_COUNT,INTERVAL,TEXT
STARTDT is the date on which the series of events starts. 0 indicates that the series starts on the first day of the printed month.
END_OR_COUNT is the end date or 0, or a count of the number of events in the series. 0 indicates that the series continues throughout the entire month being printed. An integer number gives the number of events in the series.
INTERVAL is any offset acceptable to the convert_date_to_binary_ subroutine, or 0. An offset is truncated to an integral number of days; but if it is less than one day, it is treated as if it were 1 day. 0 indicates an interval of 1 day.
TEXT is arbritrary text to be placed in the box of each day in the series.

easter

calculates the date for Easter and inserts its text in that date if it falls in the printed month. The syntax is--
easter,TEXT

rename

allows the user to change the names of days or months.

Its syntax is--
rename,OLDNAME,NEWNAME
OLDNAME gives the name of a day or month to be changed. If the name of that day or month was previously changed in the current invocation of the command, OLDNAME must be the current name.
NEWNAME gives the name to replace the OLDNAME.

Notes: All dates must be acceptable to the convert_date_to_binary subroutine. See date_time_strings.gi.info for acceptable forms.

If the command finds errors in its arguments it reports the errors and does not print a calendar. If it finds errors in an input file, it stops after all errors have been reported, unless the user gives the -force control argument to indicate that the calendar should be printed in spite of errors.

For more information, refer to the MPM Commands and Active Funtions, Order no. AG92.

calendrier

05/13/83 calendrier

Syntaxe: calendar noms_de_chemin -arguments_de_controle

Fonction: Imprime une page de calendrier pour un mois donn' e.

Argument:
noms_de_chemin
sont des noms de segment contenant des donnees a inserer dans le calendrier. Cet argument est facultatif. S'il n'est pas donn' e, le programme calendar imprime un calendrier contenant les titres (noms des jours et des mois) en anglais.

On a cree un segment permettant d'obtenir :
- les noms des jours (Lundi, etc...) et des mois (Janvier, etc...) en francais ;
- quelques dates importantes indiquees dans les cases correspondantes (a savoir : les jours feries francais : Nouvel An, Paques, etc..., ainsi que les jours de passage a l'heure d'ete et a l'heure d'hiver).

Ce segment est : >am>don>cal_fr .

Chaque utilisateur peut creer d'autres segments contenant les informations qu'il desire inserer dans ses calendriers. On peut suivre les indications du "help calendar", et prendre pour modele le segment indique ci-dessus (on peut imprimer ce segment par un "print" ordinaire).

Arguments de controle:
Les principaux sont : -dt DATE -fw -sp
(Le "help calendar" donne d'autres arguments, qui semblent moins utiles)

Argument de date:
-dt DATE
indique un jour quelconque du mois desire, par exemple :

-dt 05/01/83	pour Mai 1983
-dt 09/01/83	pour Septembre 1983
-dt 12/01/84	pour Decembre 1984

Argument -fw:
-fw
L'option -fw permet d'avoir le numero de semaine indique dans chaque case de lundi.

Argument -sp:
-sp
L'option -sp permet d'inserer une attente avant le debut de l'impression, et apres la fin de l'impression ; ceci permet de degager le papier afin de separer le calendrier proprement dit de ce qui precede (le texte de

la commande calendar), et de ce qui suit (le message "ready") ;
lorsqu'on est pret, on tape le caractere "return".

Exemple:

```
calendar >am>don>cal_fr -dt 12/01/83
```

donnera le calendrier du mois de decembre 1983.

On peut se creer une abreviation, par exemple :

```
.ab CAL calendar >am>don>cal_fr
```

et commander :

```
CAL -dt 12/01/83 -fw
```

Longueur de ligne:

Il faut une longueur de ligne au moins egale a 120 ;
donc, si necesaire, commander d'abord :

```
ll 120
```

(ou une valeur superieure a 120).

Conseil:

La derniere ligne imprim' ee comporte quelques grigri indesirables (" 014"
et le message "ready"). Pour les eviter, on peut commander d'abord :

```
stty -modes edited
rdf
```

avant d'envoyer la commande calendar. Mais n'oubliez pas, ensuite,
de revenir a :

```
rdn
stty -modes ^edited
```

si c'est votre mode habituel.

On peut, dans ce cas, utiliser des abreviations telles que :

```
.ab CALE stty -modes edited;rdf;calendar >am>don>cal_fr
.ab CALFIN rdn;stty -modes ^edited
```

(En fait, il est inutile de commander rdf ... rdn dans le cas ou l'on
emploie l'option -sp).

cancel_abs_request

10/01/80 cancel_abs_request, car

Syntax: car request_identifiers -control_args

Function: allows a user to delete a request for an absentee computation that is no longer needed.

Arguments:
request_identifiers can be chosen from the following:
path
is the full or relative pathname for the absentee input segment of requests to be cancelled. The star convention is allowed.
-entry STR, -et STR
identifies requests to be cancelled by STR, the entryname portion of the absentee input segment pathname. The star convention is allowed.
-id ID
identifies one or more requests to be cancelled by request identifier. This identifier may be used to further define any path or -entry identifier (see "Notes").

Control arguments:
-all, -a
indicates that all priority queues and the foreground queue are to be searched starting with the foreground queue followed by the highest priority queue and ending with the lowest priority queue.
-brief, -bf
suppresses messages telling that a particular request identifier was not found or that requests were cancelled when using star names or the -all control argument.
-foreground, -fg
specifies that the foreground absentee queue contains the request(s) to be cancelled.

-queue N, -q N
specifies that absentee queue N contains the request to be cancelled, where N is an integer specifying the number of the queue. The default queue is defined by the system administrator. For convenience in writing exec_coms and abbreviations, the word foreground or fg following the -queue control argument performs the same function as the -foreground control argument. If the -queue, -fg, and -all control arguments are omitted, only the default priority queue is searched.
-sender STR
specifies that only requests from sender STR should be cancelled. One or more request identifiers must also be specified. In most cases, the sender is an RJE station identifier.

-user User_id

specifies the name of the submitter of the request to be cancelled, if it is not the same as the group identifier of the process. The User_id can be specified as Person_id.Project_id, Person_id, or *.Project_id. This control argument is primarily for operators and administrators

Access required: The user must have o extended access to the queue to cancel their own requests. The user must have r and d extended access to cancel a request entered by another user.

Notes: If the absentee process has already logged in, the user is given the choice of bumping the job and cancelling the request from the queue, or allowing the job to continue running and remain in the queue. This allows the user to cancel a running absentee process.

When star names are not used and a single request identifier matches more than one request in the queue(s) searched, none of the requests are cancelled. However, a message is printed telling how many matching requests were found.

If any path or -entry STR request identifiers are given, only one -id ID request identifier will be accepted and it must match any requests selected by path or entryname.

Multiple -id ID identifiers can be specified in a single command invocation only if NO path or entry request identifiers are given.

The -queue, -foreground, and -all control arguments are mutually exclusive.

Normally, deletion can be made only by the user who originated the request.

cancel_daemon_request

10/08/80 cancel_daemon_request, cdr

Syntax: cdr request_identifiers -control_args

Function: deletes an I/O daemon request that is no longer needed.

Arguments:
request_identifiers can be chosen from the following:

path
identifies a request to be cancelled by the full or relative pathname of the input data segment. The star convention is allowed.

-entry STR, -et STR
identifies a request to be cancelled by STR, the entryname portion of the input data segment pathname. The star convention is allowed.

-id ID
identifies one or more requests to be cancelled by request identifier. This identifier may be used to further define any path or -entry identifier (see "Notes").

Control arguments:

-all, -a
searches all priority queues for the specified request type starting with the highest priority queue and ending with the lowest priority queue. This control argument is incompatible with the -queue control argument.

-brief, -bf
suppresses messages telling that a particular request identifier was not found or that requests were cancelled when using star names or the -all control argument.

-queue N, -q N
specifies that queue N of the request type contains the request to be cancelled, where N is a decimal integer specifying the number of the queue. If this control argument is omitted, only the default queue for the request type is searched. This control argument is incompatible with the -all control argument.

-request_type STR, -rqt STR
indicates that the request to be cancelled is to be found in the queue for the request type identified by the string STR. If this control argument is not given, the default request type is "printer". Request types can be listed by the print_request_types command.

-user User_id
specifies the name of the submitter of the request to be cancelled, if not the group identifier of the process. The User_id can be equal to Person_id.Project_id, Person_id, or .Project_id. Both r and d extended access to the queue are required. This control argument is primarily for operators and administrators.

Access required: The user must have o extended access to the queue to cancel their own requests. The user must have r and d extended access to cancel a request entered by another user.

Notes: If the request is already running, the entry is still removed from the queue but the running request is not stopped. However, the user is given a message stating that the request is running.

When a request has been removed from the queue after it has started running and before it has finished, any user requested deletion of the segment (done with the -delete control argument to the dprint command) will be ignored by the system.

Multiple -id ID identifiers can be specified in a single command invocation only if NO path or entry request identifiers are given.

If any path or -entry STR request identifiers are given, only one -id ID request identifier will be accepted and it must match any requests selected by path or entryname.

When star names are not used and a single request identifier matches more than one request in the queue(s) searched, none of the requests are cancelled. However, a message is printed telling how many matching requests there are.

Normally, deletion can be made only by the user who originated the request.

See also the descriptions of the dprint and dpunch commands.

cancel_resource

01/18/79 cancel_resource, cnr

Syntax: cnr -id reservation_id -control_arg

Function: cancels reservations made with the reserve_resource command using the reservation identifier obtainable from the list_resources command.

Arguments:
reservation_id
must be present and is the reservation identifier of the reservation to be cancelled. It must be preceded by the -id control argument.

Control arguments:
-priv
allows a privileged cancellation to be done, such as, a cancellation of a reservation belonging to another user.

Access required: Use of the -priv control argument requires access to rcp_sys_.

Notes: Reservation identifiers may be obtained by using the list_resources command.

cancel_retrieval_request

10/08/80 cancel_retrieval_request, crr

Syntax: crr request_identifiers -control_args

Function: allows a user to delete a request for a volume retrieval that is no longer needed.

Arguments:
request_identifiers can be chosen from the following:

path
is the full or relative pathname of the segment or subtree of the retrieval request to be cancelled. The star convention is allowed.

-entry STR, -et STR
identifies the request to be cancelled by STR, the entryname portion of the segment or subtree pathname. The star convention is allowed.

-id ID
identifies the request to be cancelled specified by its request ID number. This identifier may be used to further define any path or -entry identifier (see "Notes").

Control arguments:

-all, -a
indicates that all retrieval queues are to be searched starting with the highest priority queue and ending with the lowest priority queue. This control argument is incompatible with the -queue control argument.

-brief, -bf
suppresses messages telling the user that a particular request identifier was not found or that requests were cancelled when using star names or the -all control argument.

-queue N, -q N
specifies that retrieval queue N contains the request to be cancelled, where N is a decimal integer specifying the number of the queue. If this control argument is omitted, only the default priority queue is searched. This control argument is incompatible with the -all control argument.

-user User_id
specifies the name of the submitter of the requests to be cancelled, if not equal to the group identifier of the process. The User_id can be Person_id.Project_id, Person_id, or .Project_id. Both r and d extended access to the queue are required. This control argument is primarily for operators and administrators.

Access required: The user must have o extended access to the queue to cancel their own requests. The user must have r and d extended access to cancel a request entered by another user.

Notes: If any path or -entry STR request identifiers are given, only one -id ID request identifier will be accepted and it must match any requests selected by path or entryname.

Multiple -id ID identifiers can be specified in a single command invocation only if NO path or entry request identifiers are given.

Normally, deletion can be made only by the user who originated the request.

When star names are not used and a single request identifier matches more than one request in the queue(s) searched, none of the requests are cancelled. However, a message is printed telling how many matching requests there are.

change_default_wdir

02/12/76 change_default_wdir, cdwd

Syntax: cdwd path

Function: sets a directory as the user's default working directory.

Arguments:
path
is the pathname of the directory; if omitted, the current working directory becomes the default working directory.

Notes: The original default working directory is the user's home directory upon logging in.

change_wdir

02/12/76 change_wdir, cwd

Syntax: cwd path

Function: changes the user's working directory.

Arguments:
path
is the pathname of a directory; if omitted, the default working directory is assumed. (see change_default_wdir)

check_iacl

01/09/80 check_iacl

Syntax: check_iacl dirpath -control_args

Function: lists ACL terms that disagree with initial ACL.

Arguments:
dirpath
pathname of a directory to check. The star convention is allowed.
-working_directory or -wd specifies the working directory.
(Default if omitted: working directory)

Control arguments:
-all
lists changed and deleted ACL entries as well. (Default: added entries)
-exclude Pers.Proj, -ex Pers.Proj
excludes ACL entries with names matching Pers.Proj.

Notes:
Unless -all is specified, only ACL terms that are ADDITIONS to the initial ACL are listed.

Up to 10 -exclude arguments are allowed.

check_info_segs

09/26/80 check_info_segs, cis

Syntax: cis -control_args

Function: prints a list of info segments modified since a given time.

Control arguments:
-absolute_pathname, -absp
   prints or returns absolute pathnames of segments rather than entrynames.
-brief, -bf
   does not print names of changed info segs and "No change" message. For use with -call. -bf cannot be used with the cis active function.
-call cmdline
   calls the command processor with "cmdline path" for each changed segment; path is the absolute pathname of a changed segment. If cmdline contains blanks, it must be enclosed in quotes. This control argument cannot be used with the cis active function.
-date DT, -dt DT
   uses the date DT instead of the date in the user's profile. The date in the profile is not updated.

-long, -lg
   prints the date-time-entry-modified as well as the segment name. -lg cannot be used with the cis active function.
-no_update, -nud
   does not update the date in the user's profile.
-pathname star_path, -pn star_path
   star_path is a pathname with a star name in the entryname portion. All segments that match star_path are checked. More than one -pathname control argument can be given. If none are given, the directories in the "info_segments" search list, which has synonyms "info_segs" and "info" are used.
-time_checked, -tmck
   prints the date_time that is stored in the user's profile indicating from when checking of modified info segments would occur if the -date control argument were not used. This control argument is incompatible with all others when used with the cis active function. It does not update the time in the user's profile when used as the only control argument.

Notes: The first time cis is invoked by a user, it just sets the date in the user's profile. A profile is created if one doesn't exist. The date-time-entry-modified for link targets are checked, not the dtem of the link.

Syntax as active function: [cis -control_args]

Notes on active function:
The cis active function returns entrynames of selected info segments separated by spaces. If -absp is specified, it returns full pathnames of info segments separated by spaces.

Warning:
Since cis active function also sets the date in the user profile, a command line using [cis] sets this date before processing any of the returned info seg names. As a result, segments can be unintentionally skipped and not seen a second time if a command line containing [cis] is interrupted.

clear_resource

01/29/79 clear_resource, clr

Syntax: clr type STRs

Function: Confirms that the contents of a released volume have been destroyed so that it may be returned to the free pool of resources kept by RCP resource management.

Arguments:
type
is a resource type defined in the RTDT. STR is the name of a resource to be cleared. If it begins with a hyphen, it must be preceded by -name (-nm.)

Access required: Use of this command requires access to the rcp_sys_ gate.

Notes: If multiple resource names are given and one of the named resources cannot be cleared, none of the resources will be cleared.

close_file

02/26/76 close_file, cf

Syntax: close_file -control_arg filenames

Function: closes FORTRAN and PL/I files.

Arguments:
filenames
   are the names of open files.

Control arguments:
-all
   closes all open files. In this case, no filename appears.

Notes: The format of a FORTRAN file name is fileNN where NN is a two-digit number other than 00; e.g., file05. PL/I file names are selected by the user and can have any format.

For each filename argument, all PL/I files of that name and, if applicable, the FORTRAN file of that name are closed.

The command "close_file -all" does not affect I/O switches that are not associated with FORTRAN or PL/I files.

collate

09/23/80 collate

Syntax: collate

Function: returns the 128 characters of the ASCII character set in collating sequence.

Syntax as active function: [collate]

compare

03/12/80 compare

Syntax: compare path1 |offset1 path2 |offset2 -control_args

Function: compares two segments and lists their differences.

Arguments:
path1, path2
   pathnames of segments to be compared. path2 can be an equal name.
offset1, offset2
   octal offsets within the segments; if omitted, the entire contents are compared.

Control arguments:
-brief, -bf
   prints only the first and last words of each discrepancy.
-length N, -ln N
   the comparison continues for no more than N (octal) words.
-long, -lg
   prints all discrepancy words. (default)
-mask N
   the octal mask N is used in the comparison. If N is less than 12 octal digits, it is padded on the left with zeros.

Notes: The maximum number for words to be compared is the word count for the first segment minus its offset or the word count of the second segment minus its offset, whichever is greater. If the segments are of unequal length, the remaining words for the longer segment are printed as discrepancies.

Syntax as an active function:
[compare path1|offset1 path2|offset2 -control_args]
returns true if the compared portions are identical, false otherwise.

compare_ascii

05/12/81 compare_ascii, cpa

Syntax: cpa paths -control_args

Function: compares ASCII segments and prints any differences.

Arguments:
paths
are the pathnames of the segments to be compared. The equal and :: conventions are allowed. Up to six segments can be compared, in addition to the original if one is supplied. The equal convention can be used in any pathname except the first one on the command line, which is assumed to be the original unless otherwise specified.

Control arguments:
-header, -he
prints a heading, giving the full pathname and identifying letter of each segment. This heading is not printed by default.
-minchars NN
specifies the minimum number of characters that must be identical for compare_ascii to assume that it has found the end of a difference. The default is 20 characters. See "Notes" below.
-minlines NN
specifies the minimum number of lines that must be identical for compare_ascii to assume that it has found the end of a difference. The default is two lines. See "Notes" below.
-no_original, -no_orig
indicates that no original segment is supplied. If neither -no_original nor -original is given, the first pathname on the command line is assumed to be the original.

-no_numbers, -nnb
does not print identifying letter and line numbers preceding the lines from the segments being compared. The default is to print them.
-no_totals, -ntt
does not print the totals line.
-original pathA, -orig pathA
specifies the pathname pathA of the original segment of which the others are modified versions.
-print_new_lines, -pnl
prints only new lines. New lines are lines found in one or more of the modified versions but not in the original. An original must be supplied if this argument is used.
-totals, -tt
prints only the totals line, giving the number of differences and the number of changed lines. The default is to print discrepancies and totals line.

Notes: The output is organized with the assumption that the pathA segment was edited to produce pathB. This command prints lines that were added, replaced, or deleted; it identifies each line by line number within the respective segment and also by the letter A or B to indicate which segment the line is from (A for pathA and B for pathB).

Values for minchars and minlines can be specified without being preceded by control arguments. The order is: minchars minlines.

The values of minchars and minlines control the size of displayed differences. Large values for these parameters cause small, closely-spaced differences to be displayed as one large difference, while very small values (such as -minlines 1 -minchars 2) will cause small changes to be displayed individually but might also cause large differences to be broken down into small parts, thereby giving a misleading picture of what was actually done to produce the modified versions. The user should adjust these parameters to produce the most useful results.

compare_object

06/30/75 compare_object, cob

Syntax: cob old_path new_path -control_args

Function: Compares two object segments and, optionally, prints out the changes made to the segment specified by old_path to yield the segment specified by new_path. The old_path segment is assumed to be older than the new_path segment, and they are assumed to have been produced from the same source segment, by different versions of a language processor.

Arguments:
old_path
   is the pathname of the old object segment. The star convention is allowed.
new_path
   is the pathname of the new object segment. The equal convention is allowed.

Control arguments:
-brief, -bf
   suppresses detailed description of the discrepancies, instead printing a summary.
-text
   compares the text sections.
-defs
   compares the definitions sections.
-link, -lk
   compares the linkage sections.
-static
   compares the static storage if they have separate static, otherwise compares the linkage sections.
-all, -a
   compares the text, definitions, static (if any), and linkage sections. This is the default.

Notes: Control arguments must follow the two pathnames.

In comparing the lengths of the symbol sections, compare_object uses a heuristic to determine whether a discrepancy is serious or trivial. This heuristic is inaccurate for ALM, bound, or large pl1 segments.

compose

07/08/81 compose, comp (Vers 8.0)

Syntax: comp paths -control_args

Function: formats documents for production on various devices including terminals and line printers.

Arguments:
paths
pathnames of input files to be formatted (up to 200 input files-- SSFs or MSFs-- can be specified) named <name>.compin. The suffix need not be supplied in the command line. The star convention is not supported.

Control arguments:
-arguments arg1 arg2 ...
-ag arg1 arg2 ...
all given arg's are made available as user variables; any that contain blanks must be given as quoted strings. This control argument must be the last one in the command line.
-brief
-bf
Show only the error list header (giving the count of errors) at normal termination or in response to the program_interrupt command. This control is effective only when errors are being accumulated for later display, that is, when output is being sent to the user's terminal. The default is to print the entire error list.

-change_bars x,p,l,r,d
-cb x,p,l,r,d
generates text change symbols (change bars) in the output according to the parameters given.
- x change level character (default is NUL)
- p symbol placement key character (default is outside margin)
- l text change symbol to be placed at the left margin. Must be of the form "n<string>", where <string> is any character string (the change symbol) and n is the number of spaces between the text and <string> (Default is 2). n may be given by itself to change the space for the default symbol and must be given if <string> is given.
- r text change symbol to be placed to the right of text (same form as l above)
- d text deletion symbol (default same form as l above except default <string> is asterisk)

-change_bars_art x,p,l,r,d
-cba x,p,l,r,d
as for change_bars above except that "l", "r", and "d" fields (and their defaults) are artwork constructs.

-check
-ck
   syntax check mode; no output is produced.
-device name
-dev name
-dv name
   prepare output for device "name". "name".comp_dsm must exist and be locatable with compose search list. Default is "ascii" for terminal output and "printer" for -output_file.

-from n
-fm n
   begin output at page "n". n must be an EXACT match for an existing page number. Default is the first page of the file, regardless of its page number.
-galley n1 ,n2
-gl n1 ,n2
   prepare galley mode (single column, no running headers or footers) text for input lines "n1,n2". The default n1,n2 is the entire input file.
-hyphenate n
-hyph n
-hph n
   change default hyphenation mode to ON with a least word part of "n". The default value for "n" is 3.

-indent n
-ind n
   add "n" columns of white space at the left of each output line. The default value is 0.
-input_file path
-if path
   "path" is an input file pathname although it looks like a control argument or numeric parameter. "path" is required; there is no default.
-linespace n
-ls n
   set the minimum linespace value to "n" (1 = single, 2 = double, etc.). The default value is 1.

-noart
-noa
   disable artwork conversion replacing artwork constructs with blanks.
-nobell
-nob
   suppress the BEL signal for pause events (see -stop and -wait below).
-nofill
-nof
   change the default fill mode to OFF.
-number
-nb
   show input line numbers in the output; also show list of inserted files for cross-reference.

-number_brief

-nbb
   show input line numbers in the output without the list of inserted files.
-output_file path
-of path
   direct all output to the bulk collector file given by "path". The default path is [wd]>"name".compout where "name".compin is the input file.
-page n n,n ...
-pg n n,n ...
   print only the individual pages given. At least one page must be given; there is no default.
-pages_changed
-pgc
   print only Addendum pages and pages with change-bars.

-parameter string
-pm string
   assign "string" to the builtin variable Parameter. "string" is required; there is no default.
-passes n
-pass n
   make "n" processing passes over the input file(s). The default value is 1.
-stop
-sp
   pause before first output page and after every page, giving a visual and audible signal to the user.

-to n
   end output with page "n". "n" must be an EXACT match for an existing page number. The default is the last page of the input file regardless of its page number.
-wait
-wt
   pause before first output page, giving a visual and audible signal to the user.

Notes:
   Type "help compose.controls" for a summary of compose controls and "help compose.builtins" for a list of builtin variables.
   Type "help convert_runoff" for information on converting runoff input files.
   Type "help compose.artwork.gi" for information on constructing artwork.gi within compose.

   The following terminals are supported for artwork:

Terminal	name for -device option
ASCII (default)	ascii
DTC300S	dtc300s
Diablo 1620 (HYTERM)	hyterm

(END)

contents

09/12/80 contents

Syntax: contents path

Function: returns the contents of a segment as a character string. Newline characters in the segment are changed to blanks in the string.

Syntax as active function: [contents path]

copy

07/17/81 copy, cp

Syntax: cp path1 path2 ... path1N path2N -control_args

Function: causes copies of specified segments and multisegment files to be created in the specified directories with the specified names. Access control lists (ACLs) and multiple names are optionally copied.

Arguments:

path1
is the pathname of a segment or multisegment file to be copied. The star convention is allowed. The default action of this command when path1 specifies a link is given below in the "Notes" section.

path2
is the pathname of a copy to be created from path1. If the last path2 argument is not given, the copy is placed in the working directory with the entryname of path1. The equal convention is allowed.

Control arguments:

-acl
copies the ACL.

-all, -a
copies multiple names and ACLs.

-brief, -bf
suppresses warning messages.

-chase
copies the targets of links that match path1. See "Notes" for the default action.

-long, -lg
prints warning messages as necessary. This is the default.

-name, -nm
copies multiple names.

-no_acl
does not copy the ACL. This is the default.

-no_chase
does not copy the targets of links that match path1. See "Notes" for the default action.

-no_name, -nnm
does not copy multiple names. This is the default.

Access required: Read access is required for path1. Status permission is required for the directory containing path1 if the -name, -acl or -all control argument is specified. Append permission is required for the directory containing path2. Modify permission is required for the directory containing path2 if the -name, -acl, or -all control argument is specified.

Notes: The control arguments can appear once anywhere in the copy command line after the command name and apply to the entire copy command line.

The default for chasing links depends on path1. If path1 is not a starname, links are chased by default. If path1 is a starname, links are not chased.

If the ACL of a segment or multisegment file is being copied, the initial ACL of the target directory has no effect on the ACL of the segment or multisegment file after it has been copied into that directory. The ACL remains exactly as it was in the original directory. The AIM access class of a segment is not copied by -acl.

Since two entries in a directory cannot have the same entryname, special action is taken by this command if the name of the segment or multisegment file being copied (specified by path1) already exists in the directory specified by path2. If the entry being copied has an alternate name, the entryname that would have resulted in a duplicate name is removed and the user is informed of this action; the copying operation then takes place. If the entry being copied has only one entryname, the entry that already exists in the directory must be deleted to remove the name. The user is asked if the deletion should be done; if the user answers "no", the copying operation does not take place.

The copy command prints a warning message if the bit count of path1 is less than its current length or if the current length is greater than the number of records used. These warnings are suppressed by the use of the -brief control argument.

copy_acl

08/04/80 copy_acl

Syntax: copy_acl path1A path2A ... path1N path2N

Function: copies the access control list (ACL) from one file or directory to another, replacing the current ACL if necessary.

Arguments:

path1A
is the pathname of a file or directory whose ACL is to be copied. The star convention is allowed. Either -working_dir or -wd specifies the working directory.

path2A
is the pathname of a file or directory onto which the initial ACL is to be copied. The equal convention is allowed. Either -working_dir or -wd specifies the working directory.

Access required: Status permission required for containing dir of path1i. Modify permission required for containing dir of path2i.

Notes: The star and equal convention can be used.

copy_cards

03/01/76 copy_cards, ccd

Syntax: ccd deck_name new_deck_name

Function: copies specified card image segments from system pool storage into a user's directory.

Arguments:
deck_name
   name entered on the deck_id card.
new_deck_name
   pathname of segment in which matching card image segment is placed.
   If omitted, the working directory and deck_name are assumed.

Notes: The segments to be copied must have been created using the Multics card input facility. When there are multiple copies of the same deck in pool storage, all are copied.

The deck_name may use the star convention; path may use the equal convention.

When an attempt is made to read a card deck having the same name as some previously read deck still in pool storage, a numeric suffix is added to the name of the new deck, e.g., "deck_name.1"

Only those card decks having an access class equal to the user's current authorization can be copied. Other decks will not be found.

copy_dir

06/05/81 copy_dir, cpd

Syntax: cpd source_dir target_dir entry_type_keys -control_args

Function: copies a directory and its subtree to another point in the hierarchy. The user can also specify that portions of the subtree be copied and can control the processing of links.

Arguments:

source_dir
is the pathname of a directory to be copied. The star convention is allowed to match directory names. Matching names associated with other storage types are ignored. The source_dir can not be contained in target_dir.

target_dir
is the pathname of the copy of the source_dir. The equal convention is allowed. If target_dir is not specified, the copy is placed in the working directory with the entryname of source_dir. If the target_dir does not exist, it is created.

Control arguments:

-brief, -bf
suppresses the printing of warning messages.

-force
executes the command, when target_dir already exists, without asking the user. If the -force control argument is not specified, the user is queried.

-replace, -rp
deletes the existing contents of target_dir before the copying begins. If target_dir is non-existent or empty, this control argument has no effect. The default is to append the contents of source_dir to the existing contents of target_dir.

-acl
gives the ACL on the source_dir entry to its copy in target_dir. Although initial ACLs are still copied, they are not used in setting the ACL of the new entries when this control argument is specified.

-primary, -pri
copies only primary names. The default is to copy all names.

-no_link_translation, -nlt
copies links with no change. The default is to translate links being copied. If there are references to the source directory in the link pathname of a link being copied, the link pathname is changed to refer to the target directory.

-chase
copies the target of a link. The default is not to chase links. Chasing the links eliminates link translation.

List of entry_type_keys: control what type of storage system entries

in the subtree are copied. If no entry_type_key is specified, all entries are copied. The keys are--

-branch, -br
-directory, -dr
-file, -f
-link, -lk
-multisegment_file, -msf
-non_null_link, -nnlk
-segment, -sm

If one or more entry_type_keys are specified, but not the -directory key, the subtree of source_dir is not walked.

Access required: Status permission is required for source_dir and all of the directories in its tree. Status permission is required for the directory containing source_dir. Read access is required on all files under source_dir. Append and modify permission are required for the directory containing target_dir if target_dir does not exist prior to the invocation of the copy_dir command. Modify and append permission are required on target_dir if it already exists. This command does not force access.

If the -acl control argument is not specified, the system default ACLs are added, then the initial ACL for the containing directory is applied (which may change the system supplied ACL). Initial ACLs are always copied for the current ring of execution.

Notes: If target_dir already exists and -force is not specified, the user is so informed and asked if processing should continue. If target_dir is contained in source_dir, an appropriate error message is printed and control is returned to command level.

If name duplication occurs while appending the source_dir to the target_dir and the name duplication is between directories, the user is queried whether processing should continue. If the user answers yes, the contents of the directory are copied (appended) but none of the attributes of that directory are copied. If the answer is no, the directory and its subtree is skipped. If name duplication should occur between segments, the user is asked whether to delete the existing one in target_dir. (See the copy command)

If the -replace control argument is specified or target_dir does not exist, name duplication does not occur.

If part of the tree is not copied (by specifying a storage system entry key), problems with link translation may occur. If the link target in the source_dir tree was in the part of the tree not copied, there may be no corresponding entry in the target_dir tree. Hence, translation of the link causes the link to become null.

See also the copy, move and move_dir commands.

copy_file

04/24/81 copy_file, cpf

Syntax: cpf in_control_arg out_control_arg -control_args

Function: copies records from an input file to an output file that has been restructured for maximum compactness. The input and output file records must be structured. (See "Notes on unstructured files" below). The input file can be copied either partially or in its entirety.

Arguments:

in_control_arg

the input file from which records are read can be specified by either of the following:

-input_switch STR, -isw STR

specifies the input file by means of an already attached I/O switch name, where STR is the switch name.

-input_description STR, -ids STR

specifies the input file by means of an attach description STR. STR must be enclosed in quotes if it contains spaces or other command language characters.

out_control_arg

the output file to which the records are written can be specified by either of the following:

-output_switch STR, -osw STR

specifies the output file by means of an already attached I/O switch name, where STR is the switch name.

-output_description STR, -ods STR

specifies the output file by means of an attach description STR. STR must be enclosed in quotes if it contains spaces or other command language characters.

Control arguments:

-all, -a

copies until the input file is exhausted. This is the default.

-brief, -bf

suppresses an informative message indicating the number of records or lines actually copied.

-count N, -ct N

copies until N records have been copied or the input file is exhausted, whichever occurs first, where N is a positive integer. The default is to perform copying until the input file is exhausted.

-from N, -fm N

copies records beginning with the Nth record of the input file, where N is a positive integer. The default is to begin copying with the "next record." (See "Notes" below.)

-keyed

copies both records and keys from a keyed sequential input file to a

keyed sequential output file. The default is to copy records from an input file (either keyed or not) to a sequential output file. (See "Notes on Keyed Files" below.)

-long, -lg
prints an informative message indicating the number of records or lines actually copied. This is the default.

-start STR, -sr STR
copies records beginning with the record whose key is STR, where STR is 256 or fewer ASCII characters. The default is to begin copying with the "next record."

-stop STR, -sp STR
copies until the record whose key is STR has been copied or the input file is exhausted, whichever occurs first, where STR is 256 or fewer ASCII characters.

-to N
copies until the Nth record has been copied or the input file is exhausted, whichever occurs first, where N is a positive integer greater than or equal to the N given with the -from control argument. This control argument can only be specified if -from is also specified.

Notes on unstructured files: The copy_file command operates by performing record I/O on structured files. If it is desired to copy from/to an unstructured file, the record_stream_ I/O module can be used, e.g., by typing the command line:

```
cpf -ids "record_stream_ -target vfile_ pathname" -osw OUT
```

The effect is to take lines from the file specified by pathname via the vfile_ I/O module, transform them into records via the record_stream_ I/O module, and then copy them to the I/O switch named OUT.

Notes on keyed files: The copy_file command can copy a keyed sequential file to produce an output file that has been restructured for maximum compactness as a keyed file or as though it were purely sequential. By default, the command copies only records and does not place keys in the output file. To copy the keys, the -keyed control argument must be used. When -keyed is used, the input file must be a keyed sequential file. Whether keys are copied or not, control arguments can be used to delimit the range of records to be copied (i.e., -start, -stop, -from, -to, -count). Copying is always performed in key order.

Notes: If either the input or output specification is an attach description, it is used to attach a uniquely named I/O switch to the file. The switch is opened, the copy performed, and then the switch is closed and detached. Alternately, the input or output file can be specified by an I/O switch name. Either the io_call command or iox_ subroutine can be used to attach the file prior to the invocation of the copy_file command. (See the descriptions of the io_call command and the iox_ subroutine.)

copy_iacl_dir

09/17/81 copy_iacl_dir

Syntax: copy_iacl_dir path1A path2A ... path1N path2N

Function: copies the initial access control list for directories (directory initial ACL) of one directory to another, replacing the current directory initial ACL if necessary.

Arguments:

path1i
is the pathname of a directory. The star convention is allowed. Either -working_directory or -wd specifies the working directory.

path2i
is the pathname of the target directory. The equal convention is allowed. Either -working_directory or -wd specifies the working directory.

Access required:: Status permission is required on path1i. Modify permission is required on path2i.

Notes: See the MPM Reference Guide for a description of initial ACL's.

copy_iacl_seg

09/17/81 copy_iacl_seg

Syntax: copy_iacl_seg path1A path2A ... path1N path2N

Function: copies a segment initial access control list (initial ACL) from one directory to another, replacing the current initial ACL if necessary.

Arguments:

path1i
: is the directory from which the initial ACL is to be copied. The star convention is allowed. Either -working_directory or -wd specifies the working directory.

path2i
: is the directory into which the initial ACL is to be copied. The equal convention is allowed. Either -working_directory or -wd specifies the working directory.

Access required: Status permission is required on path1i. Modify permission is required on path2i.

create

02/12/76 create, cr

Syntax: cr paths

Function: creates segments.

Arguments:
paths
   are the pathnames of segments to be created.

Access required: append on the parent directory.

create_data_segment

10/24/77 create_data_segment, cds

Syntax: cds path -control_arg

Function: translates a create_data_segment source program (CDS program) into an object segment.

Arguments:
path
is the pathname of a CDS segment; the cds suffix need not be given.

Control arguments:
-list, -ls
produces a source listing of the CDS program followed by object segment information.

Notes: Because of the invocation of the PL/I compiler, the CDS run is aborted if a severity error greater than 2 occurs.

create_dir

09/17/81 create_dir, cd

Syntax: cd paths -control_args

Function: causes a specified directory branch to be created in a specified directory, or in the working directory. That is, it creates a storage system entry for an empty subdirectory. See the description of the create command for information on the creation of segments.

Arguments:
paths
   are pathnames of directories to be created.

Control arguments:
-access_class STR, -acc STR
   applies to each pathi and causes each directory created to be upgraded to the specified access class. The access class can be specified with either long or short names.
-logical_volume VOL, -lv VOL
   specifies that each directory created is to be a master directory whose segments are to reside on the logical volume named VOL.
-name STR, -nm STR
   specifies an entryname STR that begins with a minus sign, to distinguish it from a control argument.
-quota N
   specifies the quota to be given to the directory when it is created. This argument must be specified if either the -access_class or -logical_volume control argument is specified. If omitted, the directory is given zero quota. The value of N must be a positive integer, and applies to each pathi.

Access required: The user must have append permission to a directory in order to create a subdirectory in that directory.

Notes: If a quota is specified and the directory being created is not a master directory, the containing directory must have sufficient quota to move quota to the directory being created. (See the move_quota command for additional information.)

create_wordlist

02/10/79 create_wordlist, cwl

Syntax: cwl path -control_args

Function: creates an alphabetized list of all distinct words found in a specified text segment. The list is saved in a segment created in the working directory and given the name of the text segment with the .wl suffix appended.

Arguments:
path
   is the pathname of the text segment.

Control arguments:
-brief, -bf
   do not print the number of words.
-from N, -fm N
   begin processing words starting from line number N.
-header, -he
   print the pathname of the text segment.
-no_control_lines, -ncl
   skip control lines, i.e., lines that begin with a period.
-no_exclude, -ne
   do not exclude words containing no letters (e.g., words with only special characters or punctuation).
-no_sort, -ns
   omit alphabetical sorting of the word list; words are ordered as they appeared in the text segment and duplications are not omitted.
-to N
   stop processing words after line N.

Notes: Words are delimited by white space, i.e., space, horizontal tab, vertical tab, newline, and new page characters. Surrounding punctuation is removed. Completely underlined words are de-underlined. Words containing no letters are ignored unless -no_exclude is specified.

date

12/30/80 date

Syntax: date dt

Function: returns the date abbreviation for a specified date or the current date.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current date is returned. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [date dt]

date_time

03/19/81 date_time

Syntax: date_time dt

Function: returns a date and time value for a specified date-time or the current date-time consisting of: a date, a time from 0000.0 to 2359.9, a time zone, and a day of the week. The date and time value is returned as a single, quoted string of the form "mm/dd/yy hhmm.m zzz www" (e.g., "08/17/76 0945.7 est Tue").

Arguments:

dt

is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current date-time is returned. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [date_time dt]

day

12/30/80 day

Syntax: day dt

Function: returns a one- or two-digit number of a day of the month, from 1 to 31.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current day of the month is returned. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [day dt]

day_name

12/30/80 day_name

Syntax: day_name dt

Function: returns the full name of a day of the week for a specified date or the current date.

Arguments:
dt
is a date_time in a form acceptable to convert_date_to_binary_. If no argument is specified, the name of the current day is returned. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [day_name dt]

debug

02/02/79 debug, db

Syntax: db

Function: Interactive debugging aid. Type "help probe" for another.

Data requests: three fields with the following format--
<generalized address> <operator> <operands>

Generalized address--
[/segment name/] [offset] [segment ID] [relative offset]

Operators--

,	print	>	alter program control
=	assign	:=	call a procedure
<	set a break		

Segment IDs--

&a	argument list	&i	internal static section
&l	linkage section	&p	parameters
&t	text section	&s	stack frame

Operands:
, operand operand first operand is output mode; second operand is how much to print. (See Output modes) (See Output modes)

= operand new value to use; can be octal number, decimal number, character string, register value, instruction format input, floating point number, pointer, bit string, or variable.

:= operand procedure_name (argument list).

Registers:

$a	$exp
$q	$tr
$aq	$ralr
$eaq	$ppr
$x0	$tpr
.	$even
.	$odd
.	$ind
$x7	$prs
$pr0	$regs
.	$scu
.	$all
.	
$pr7	

```
Output modes:
o  octal                      e  floating point with exponent
h  half-carriage octal        f  floating point
d  decimal                    b  bit string
a  ASCII                      g  graphic
i  instruction                comp-5  COBOL
p  pointer                    comp-6  COBOL
s  source statement           comp-7  COBOL
l  code for line number       comp-8  COBOL
n  no output


Control requests:
.ti,j   trace stack from frame i for j frames
.+i or .-i  pop or push stack by i frames
.i   set stack to i'th frame
..   Multics command
.d or .D  print default values
.c,i    continue after break fault (ignore next i break fault)
.ct,i   continue, in temporary break mode
.cr,i   continue, in normal mode
.q   return from debug to caller
.bri  reset break i


.br  reset the breaks of the default object segment
.bgr  reset all breaks
.bli  list break i
.bl  list the breaks of the default object segment
.bgl  list all breaks
.bei <line>  execution line for break i
.be <line>   execution line for all breaks of the default
 object segment
.bge <line>  execution line for all breaks
.boi  disable break i
.bo  disable the break of the default object segment
.bgo  disable all breaks
.bni  enable break i


.bn   enable the breaks of the default object segment
.bgn  enable all breaks
.bgt <line>  establish a temporary global command
.bci al -rel- a2  make conditional break i
.bc  al -rel- a2  make conditional all breaks of default object
segment
.bsi n  set skips of break i to n
.bd name/no. set (or print) default object segment
.bp   print names of all segments with breaks
.ai,m  print argument i in modes m (modes--o, p, d, a, b, l, e, f, ?)
.f  use registers from last fault
.C  use crawlout registers
.mb  change to brief output mode
.ml  change to long output mode
.si  identifies switch_name used for input
.so  identifies switch_name used for output
```

default_wdir

07/23/80 default_wdir, dwd

Syntax: dwd

Function: returns the pathname of the default working directory of the process in which it is invoked, as set by the change_default_wdir (cdwd) command.

Syntax as active function: [dwd]

defer_messages

06/17/81 defer_messages, dm

Syntax: dm destination -control_arg

Function: suspends printing of messages.

Arguments:
destination
is of the form Person_id.Project_id to specify a mailbox. Default is the user's default mailbox.

Control arguments:
-pathname path, -pn path
specifies a mailbox by pathname. The .mbx suffix is assumed.

Notes: Deferred messages stay in the user's mailbox until the user issues the print_messages (pm) command. The immediate_messages (im) command restores printing of messages as they are received.

For a description of the mailbox, see the accept_messages and print_mail commands.

delete

12/22/80 delete, dl

Syntax: dl paths -control_args

Function: causes the specified segments and/or multisegment files to be deleted. See also the delete_dir and unlink commands.

Arguments:
paths
   are the pathnames of segments or multisegment files. The star convention is allowed.

Control arguments:
-brief, -bf
   inhibits the printing of an error message if a segment or multisegment file to be deleted is not found.
-chase
   deletes targets of links specified by paths, as well as segments.
-force, -fc
   deletes the specified entries whether or not they are protected, without issuing a query.
-long, -lg
   prints a message of the form "Deleted file <path>" for each entry deleted.
-name STR, -nm STR
   specifies a nonstandard entry name STR (e.g., invalid starname such as **.**.compout or name containing <.)

-no_chase
   does not delete targets of links. (Default)
-query_each, -qye
   issues a query for every entry to be deleted, whether of not it is protected. Protected segments will be noted in the query.
-query_all, -qya
   lists all segments to be deleted, and issues a query as to whether they should all be deleted or not. Unless -force is given, an individual query will be given for protected segments.

Access required: The user must have modify permission on the containing directory.

Notes: At least one path, or -name STR, must be specified.

In order to delete a segment or multisegment file with the delete command, the entry must have both its safety switch and its copy switch off. If either is on, the user is interrogated whether to delete the entry.

Use delete_dir to delete directories. Use unlink to delete links.

delete_acl

02/06/80 delete_acl, da

Syntax: da path User_ids -control_args

Function: removes entries from the ACLs of segments, multisegment files, and directories.

Arguments:
path
pathname of a segment, multisegment file, or directory. If it is -wd or -working_dir, the working directory is assumed. The star convention is allowed.
User_ids
are access control names that must be of the form Person_id.Project_id.tag. All ACL entries with matching names are deleted. If User_id is omitted, the user's Person_id and current Project_id are assumed.

Control arguments:
-all, -a
deletes all ACL entries except for *.SysDaemon.*.
-directory, -dr
specifies that only directories are affected. The default is segments, multisegment files, and directories.
-segment, -sm
specifies that only segments and multisegment files are affected.
-brief, -bf
suppresses the messages "User name not on ACL." and "Empty ACL."

Access required: modify on the containing directory.

Notes:
Type "help acl_matching" for an explanation of matching strategy used on User_ids.

delete_dir

10/08/80 delete_dir, dd

Syntax: dd paths -control_args

Function: causes the specified directories and any segments, links, and multisegment files they contain, to be deleted. All inferior directories and their contents are also deleted.

Arguments:
paths
are pathnames of directories. The star convention is allowed.

Control arguments:
-brief, -bf
inhibits the printing of an error message if the directory to be deleted is not found.
-force
deletes the specified directories without issuing a query.
-long, -lg
prints a message of the form "Deleted directory <path>" for each directory deleted.
-name STR, -nm STR
specifies a nonstandard entry name STR (e.g., invalid starname such as **.**.compout or name which contains <.)

-query_each, -qye
issues a query for each directory being deleted. This is the default.
-query_all, -qya
lists all directories to be deleted, and issues one query for all of them.

Access required: The user must have modify permission on both the directory and its superior directory.

Notes: At least one path or -name must be specified.

If the -force control argument is not specified, delete_dir asks the user whether to delete the specified directory. It is then deleted only if the user types "yes".

When deleting a nonempty master directory, or a directory containing inferior nonempty master directories, the user must have previously mounted the logical volume(s). If a nonempty master directory for an unmounted volume is encountered, no subtrees of that master directory are deleted, even if they are mounted.

Use delete to delete segments. Use unlink to delete link entries.

WARNING: Protected segments in pathi or any of its subdirectories are not deleted. Segments whose write bracket is less than the current ring (except for mailboxes and message segments) are also not deleted. Consequently, the subtree is not completely deleted if it contains any such segments. For a discussion of protected segments, see the safety switch attribute in the MPM Reference Guide. For a discussion of ring brackets, see "Intraprocess Access Control" in the MPM Reference Guide.

delete_iacl_dir

10/08/80 delete_iacl_dir, did

Syntax: did path User_ids -control_args

Function: deletes entries from a directory's initial access control list (initial ACL) in a specified directory. A directory initial ACL contains the ACL entries to be placed on directories created in the specified directory.

Arguments:

path
specifies a pathname of the directory whose directory initial ACL should be changed. If path is -wd, -working_dir, or omitted, the working directory is assumed. The star convention is allowed.

User_ids
are access control names that must be of the form Person_id.Project_id.tag. All entries in the directory initial ACL that match User_id are deleted. (For a description of the matching strategy, refer to the set_acl command.) If no User_ids are specified, the user's Person_id and current Project_id are assumed.

Control arguments:

-all, -a
deletes the entire directory initial ACL with the exception of an entry for *.SysDaemon.*.

-brief, -bf
causes the messages "User name not on ACL of path" and "Empty initial ACL" to be suppressed.

-ring N, -rg N
identifies the ring number whose directory initial ACL should be deleted.If it is present, it must be followed by N (where user's ring < or = N < or = 7). It can appear anywhere on the line and affects the whole line. If this argument is not given, then the user's ring is assumed.

Access required: The user must have modify (m) permission on the directory.

delete_iacl_seg

10/08/80 delete_iacl_seg, dis

Syntax: dis path User_ids -control_args

Function: deletes entries from a segment initial access control list (initial ACL) in a specified directory. A segment initial ACL contains the ACL entries to be placed on segments created in the specified directory.

Arguments:

path
specifies the pathname of a directory whose segment initial ACL should be changed. If path is -wd, -working_dir, or omitted, the working directory is assumed. The star convention is allowed.

User_ids
are access control names that must be of the form Person_id.Project_id.tag. All entries in the directory initial ACL that match the given User_ids are deleted. (For a description of the matching strategy, refer to the set_acl command. If no User_ids are specified, the user's Person_id and current Project_id are assumed.

Control arguments:

-all, -a
deletes the entire initial ACL with the exception of an entry for *.SysDaemon.*.

-brief, -bf
causes the messages "User name not on ACL of path" and "Empty initial ACL" to be suppressed.

-ring N, -rg N
identifies the ring number whose segment initial ACL should be deleted. If it is present, it must be followed by N (where user's ring < or = N < or = 7). It can appear anywhere on the line and affects the whole line. If this control argument is not specified, the user's ring is assumed.

Access required: The user must have modify (m) permission on the directory.

delete_message

06/17/81 delete_message, dlm

Syntax: dlm destination numbers -control_args

Function: deletes a message sent by the send_message command and saved in a mailbox with the -hold control argument to the accept_messages command. (See the accept_messages command for more details.)

Arguments:
destination
   can be of the form Person_id.Project_id to specify a mailbox. If destination contains either < or >, it is assumed to be the pathname of a mailbox. This argument and the -pathname control argument are mutually exclusive.
numbers
   are message numbers as printed by the print_message command when accept_messages -hold is in effect.

Control arguments:
-all, -a
   deletes all messages from the mailbox.
-pathname path, -pn path
   specifies a mailbox by pathname. The mbx suffix is assumed. This control argument and the destination argument are mutually exclusive.

Notes: If no mailbox is specified, the user's default mailbox is assumed. For a description of the mailbox, refer to the accept_messages and print_mail commands.

delete_name

09/17/81 delete_name, dn

Syntax: dn paths -control_arg

Function: deletes specified names from segments, multisegment files, links, or directories that have multiple names.

Arguments:
paths
   are pathnames to be deleted. The star convention is allowed.

Control arguments:
-brief, -bf
   suppresses error messages when entries are not found with specified pathnames. The default is -long (-lg).
-long, -lg
   prints error messages when entries are not found. This is the default.
-name STR, -nm STR
   specifies a nonstandard entry name STR (e.g., a name which looks like a starname such as *.compout or name containing <).

Access required: The user must have modify permission on the parent directory.

Notes: At least one path or -name STR must be specified.

delete_search_paths

09/17/81 delete_search_paths, dsp

Syntax: dsp search_list search_paths -control_arg

Function: allows a user to delete one or more search paths from the specified search list.

Arguments:

search_list
   is the name of the search list from which the specified search paths are deleted. It must be quoted if it contains spaces or other command language characters.

search_pathi
   specifies a search path to be deleted. The search path can be an absolute or relative pathname or a keyword. It is necessary to use the same name that appears when the print_search_paths command is invoked.

Control arguments:

-all, -a
   specifies that the search list itself is to be deleted. Any search paths specified are ignored. This control argument must be used to delete all the search paths in a search list.

Notes: For a complete list of the search facility commands, see the the add_search_paths command.

delete_search_rules

01/22/76 delete_search_rules, dsr

Syntax: dsr paths

Function: deletes current search rules.

Arguments:
paths
   are pathnames to be deleted from the current search rules.
   One of the paths can be the keyword working_dir.

Notes: Site-defined keywords and home_dir and process_dir are not accepted by delete_search_rules although they are accepted by the add_search_rules command. Deletion of the keywords initiated_segments and referencing_dir is discouraged and can lead to unpredictable results.

detach_audit

12/13/78 detach_audit, dta

Syntax: dta switchname

Function: stops auditing. This removes audit_ from the specified switch and puts the switch back the way it was before audit_ was used.

Arguments:
switchname
is the switch from which audit_ is to be removed. (DEFAULT -- user_i/o).

Notes: For further information about the audit facility, type:
help audit_
help audit_.gi
help attach_audit
help audit_editor
help display_audit_file

discard_output

09/17/81 discard_output, dco

Syntax: dco -control_arg command_line

Function: executes a command line while temporarily suppressing output on specified I/O switches.

Control arguments:
-output_switch STR, -osw STR
where STR is the name of an I/O switch. If no control arguments are specified, output on the user_output I/O switch is suppressed. If the control argument is specified, it must appear before command_line.

Arguments:
command_line
is a command line. It need not be quoted.

Notes: If the command specified in command_line cannot be executed, an error message is printed.

display_audit_file

03/20/81 - display_audit_file, daf

Syntax: daf path -control_args

Function: displays an audit file. What portion of which audit file, and in what format, is determined from the argument (if any) and control arguments.

Arguments:

path
is the pathname of an audit file to be displayed. The path argument and the -switch control argument are incompatible. (DEFAULT -- The audit file in use by user_i/o, if user_i/o is being audited. If not, another switch being audited is looked for.)

Control arguments:

-switch STR, -sw STR
use the audit file associated with switch STR. (DEFAULT -- user_i/o)

-from STRX, -fm STRX
Begin displaying the audit file with the first entry satisfying STRX. If STRX is an entry number, then it indicates the entry of that number. If STRX is a time, then it indicates the first entry having a time stamp greater than or equal to STRX. If STRX is a string, then it indicates the first entry containing the string. (DEFAULT -- beginning of the file.)

-to STRX
Stop displaying the audit file with the first entry indicated by STRX. STRX has the same interpretation as for -from above.

-next STRX
Stop displaying the audit file with the entry indicated by STRX. If STRX is an entry number, then it indicates the entry this number of entries after the first entry displayed. If STRX is a time, then it indicates the entry this period of time after the first entry displayed. If STRX is a string, then it indicates the first entry containing this string after the first entry displayed.

-last STRX
Begin displaying the audit file with the entry indicated by STRX. If STRX is an entry number, then it indicates the entry this number before the last entry in the file. If STRX is a time, then it indicates the entry this period of time before the last entry in the file. If STRX is a string, then it indicates the first entry from the bottom of the file which contains this string.

-match STRXs
display only those entries which contain one or more of the STRXs.

-exclude STRXs, -ex STRXs
do not display any entry containing one or more of the STRXs.

-class STRXs
display only those entries which have a class identifier which matches one or more of the STRXs. The STRXs can be any mixture of upper and lower case. They are translated into upper case internally by the command.

-reverse, -rv
   display the entries in reverse chronological order.
-line_length N, -ll N
   Set the output line length to N. Lines will be wrapped, if they are too long, in such a way as to keep the header margin intact. Any line which follows a newline not originally in the audit file entry is preceded by an asterisk.
-entry_numbers, -etn
   display the entries with their associated entry numbers in the left margin.
-class_identifiers, -cli
   display the entries with their associated class identifiers in the left margin.
-metering, -mt
   display the entries with their associated metering data, time of entry, virtual cpu time, and page faults.
-append_nl, -anl
   append new lines to the end of entries which don't end in a new line. Overrides a -no_append_nl to the left in the command line. (DEFAULT -- newlines are appended if a "leader generating" control argument is present, i.e. -etn -cli and -mt. Otherwise, newlines aren't appended.)
-no_append_nl, -nanl
   prevents the addition of new lines to entries which don't end in a new line. This control argument overrides the appending of newlines because of "leader generating" control arguments, or an occurence (to the left in the command line) of the -append_nl control argument. (DEFAULT -- newlines are not appended if a "leader generating" control argument is not present.)
-insert_nl, -inl
   insert new lines whenever an entry is over length (as determined by the -ll control argument or the current line length for the switch). (DEFAULT -- newlines are inserted.)
-no_insert_nl, -ninl
   prevent new lines from being inserted. This is most useful for overriding the default. (DEFAULT -- newlines are inserted when entries are over length.)
-output_file PATH, -of PATH
   display the audit file into the segment at PATH.

Access required:
   Read access on the audit file.

Notes:

The STRX referred to above is either an argument with no leading "-", or -string followed by any argument. In the former case, STRX is interpreted as an entry number if it is a positive integer (with no decimal point). It's interpreted as a time if it is a positive real number (with a decimal point). If it's neither a positive integer nor a positive real number, then it's interpreted as a string. In the second case, where STRX contains -string, the following argument is taken as the value of STRX and is always interpreted as a string argument.

For an entry to be displayed, it must satisfy all of the constraints specified in the control arguments. Hence, it must be within the range of entries given by any of -from, -to, -next, and

-last; and it must satisfy all of -match, -exclude, and -class.

Examples:

```
    daf -match -string -last
  displays all entries containing the string "-last".
    daf -from -string 10
  displays all entries from the first entry containing the string "10".
```

do

09/17/81 do

Syntax: do command_string args
or: do -control_args

Function: substitutes arguments into a command string. The expanded command line is then passed to the current Multics command processor for execution. As an active function, evaluates to the expanded command line, without executing it.

Arguments:
command_string
is a command line enclosed in quotes.
args
are character string arguments that replace parameters in command_string.

Control arguments:
set the mode of operation of the do command, and can only be specified if neither a command_string nor args are given.
-absentee
establishes an on unit for the any_other condition during the execution of the expanded command line.
-brief, -bf
suppresses printing of the expanded command line (default).
-go
passes the expanded command line to the command processor (default).
-interactive
does not catch any signals. This is the default.
-long, -lg
prints the expanded command line on error_output before it is executed or passed back.
-nogo
does not pass the expanded command line to the command processor.

Notes: Each &i (where i is from 1 to 9) found in command_string is replaced by the corresponding argi. &(nn) can be used for argument numbers greater than 9 (nn are digits). If any argi is not supplied, each instance of &i is replaced by the null string. Each &! in command_string is replaced by a 15-character ID unique to the particular invocation of the do command. Each instance of && is replaced by an ampersand.

Syntax as an active function: [do command_string args]

List of parameters:
&qi
(where i is from 1 to 9) requests quote-doubling in the argument as it is substituted into the expanded command line.

&ri
(where i is from 1 to 9) requests that the argument be requoted and have its quotes doubled during substitution.

&n
is replaced by the actual number of args supplied.

&fi
(where i is from 1 to 9) is replaced by the actual arguments argi through argN.

&f&n
is replaced by the last argument supplied.

&qfi
is replaced by arguments argi through argN with quotes doubled.

&rfi
is replaced by arguments argi through argN, requoted.

dpl

11 jan 84 dprint_laser, dpl

Syntaxe: dpl -control_args path

Fonction: creation d'une requete d'impression d'un segment ou d'un fichier multisegment. Cette commande ne travaille pas directement sur les segments objets standards.

Argument:
path
   est le pathname du segment ou du fichier multisegment; la convention star n'est pas permise.

Arguments de controle:
   Sont acceptes tous les arguments de controle de la commande 'dprint' sauf ceux indiques ci-dessous :
 -header, -he
 -request_type, rqt

Arguments de controle changes:
-copy N, -cp N
   Le segment ou multisegment-file sera imprime en N exemplaires. N vaut 1 par defaut et aura une valeur maximum de 9
-forms FOND
   l'impression sera faite sur le fond de page designe par FOND. Il faut que ce fond de page soit enregistre dans la Xerox, sinon un message d'erreur est edite par la commande 'dpl'. En l'absence d'argument -forms, le fond de page est blanc. On peut donner le nom du fond de page soit en majuscules, soit en minuscules.
-line_length N, -ll N
   cet argument a meme role que pour dprint a la seule difference que la requete est rejetee tout de suite des que N > 136 (auparavant, elle restait bloquee en queue d'attente). En outre, si N depasse le maximum correspondant au format utilise (cf. commande laser), ce fait est signale a l'utilisateur et il lui est demande s'il veut continuer.
-page_length N, -pl N
   meme role que dans dprint. La seule difference reside dans le fait que tout depassement de N du maximum correspondant au format utilise est signale: il est alors demande a l'utilisateur s'il veut continuer.

Arguments nouveaux:
-format XXX, -fmt XXX
   l'impression sera faite suivant le format XXX, XXX etant une chaine de 1 a 6 caracteres. Un format designe un ensemble de parametres tels que police de caracteres, cadrage vertical ou horizontal (c'est-a-dire orientation portrait ou paysage), longueur de la ligne en caracteres et hauteur de la page en lignes. XXX doit correspondre a un format connu de l'imprimante Xerox. Un format peut etre donne soit en majuscules, soit en

minuscules. Le format par defaut est PAYS6.

-lab_auto, -lba
cet argument est equivalent a la sequence d'arguments suivante, "dple -blbl xxx -nb 1,x,y" ou xxx est le nom du segment a lister, 'x' vaut le page_length correspondant au format plus 3 et 'y' vaut la moitie du line_length de ce meme format. Cet argument est incompatible avec les arguments '-nb' et '-blbl'.

-number PG,LG,COL , -nb PG,LG,COL
cet argument sert a demander a l'imprimante a laser une pagination automatique. L'impression du numero de page est effectuee sur la ligne LG a partir de la colonne COL. Seuls, les numeros positifs sont edites, et la numerotation des pages commence a PG. Ainsi, si on desire n pages non numerotees en debut d'impression il faut donner a PG la valeur 1-n (ex: pour 10 pages non numerotees PG sera egal a '-9').

-recto, -rc
Par defaut, l'impression se fait recto-verso avec les papiers standards. Le present argument permet de demander une impression recto seulement.

-recto_verso, -rv
Dans le cas de sorties particulieres, l'impression peut se faire recto seulement. Cet argument permet de forcer la sortie a se faire en recto-verso.

Arguments de traitement apres impression:

-addr pathname
Cet argument permet de demander le routage par courrier du resultat de l'impression. Quand un pathname est indique, il s'agit de celui du segment qui contient l'adresse normalisee qui servira a l'envoi. Le segment en question doit avoir le suffixe ".addr" qu'il n'est pas necessaire d'indiquer dans la commande. Il doit se trouver dans le meme directory que le segment a editer ou, a defaut, dans le home directory du demandeur. Si aucun pathname n'est donne apres -addr, c'est le segment Person_id.addr qui sera recherche. Si le segment recherche n'existe pas, il y a rejet avec un message d'erreur.
La longueur de l'entryname (suffixe non compris) ne doit pas exceder 22 caracteres.

-col XC
Cet argument permet de demander le collage soit sur le grand cote (XC = GC ou gc), soit sur le petit cote (XC = PC ou pc). Par defaut, il n'y a pas collage s'il y a emballage, mais il y a collage sur le grand cote s'il n'y a pas d'emballage demande.

-emb
Le resultat d'impression sera mis sous enveloppe plastique des qu'un routage par courrier est demande. Sinon, il faut le demander explicitement par cet argument.

-papier PPPP, -pap PPPP
On peut demander que l'impression soit faite sur un papier special designe par PPPP. Actuellement PPPP peut etre choisi parmi: perfo, etiq, rose, bleu, vert et jaune.

Acces demande: il est exige d'avoir au moins l'acces "r" sur le segment ou multisegment dont on veut l'impression.

Nota sur l'adresse: l'adresse contenue dans un segment .addr doit etre

normalisee, ce qui donne les regles suivantes :

- il y a au maximum 7 lignes utiles,
- chaque ligne utile comporte dans les caracteres:

1 a 9 un identificateur,
10 le numero de la ligne,
11 a xx la partie a editer (maximum 32 car.),
- la ligne '1' est obligatoire,
- les lignes doivent etre disposees dans l'ordre de leur numero.

Pour plus de details, consulter les normes du logiciel 'adress'.

Liste des formats disponibles :

Nom	ll.max	pl.max	polices	observations
pays4	136	80	R4B0BL	
pays6	136	60	M006BL	Format 'Standard' par defaut.
pays7	132	42		
pays8	110	42	R5TIBL	
port1	132	60	P1012B	Format 'Archivage'
				2 fois 132 car. x 60 lig.
port2	136	70	P1012B	idem port1 sur 130 lignes.
				(attention: ne marche pas encore)
port6	100	80		Idem 'Standard' en portrait
			P0612C	1 Normaux (pas d'accent.)
			P0612C	2 (pas de scient. idem que 1)
			R6B0BP	3 Gras (accent.)
			R612BP	4 type 'baton' (accent.)
			P06ITB	5 Italique
port6a	100	80	R612BP	Caracteres accentues
port6g	100	80	R6B0BP	Caracteres gras
port7	80	60		Format 'Courrier'
			R7TIBP	1 Caracteres accentues
			R7TICP	2 Caracteres scientifiques
			R7TIBP	3 (pas de gras idem que 1)
			P07TDC	4 type 'baton'
			P07ITA	5 Italique
port7b	80	60	P07TDC	car. batons (ascii, pas d'accent.)
port7e	99	66	P07TDC	Format pour papier 'etiquette'
port7s	80	60	R7TICP	Caracteres scientifiques
port8	66	60		Format 'Documentation'
			R8TIBP	1 Caracteres accentues
			R8TICP	2 Caracteres scientifiques
			P08TBC	3 Gras
			R812BP	4 type 'baton' (accentues)
			P08ITA	5 Italique
port8s	66	60	R8TICP	Caracteres scientifiques
port8b	66	60	R812BP	Caracteres 'baton' (accentues)
mart01	60	40	RK2C3P	Orientation portrait
				Gros caracteres (accent. possible)
				Espacement proportionnel

dprint

12/07/81 dprint, dp

Syntax: dp -control_args paths

Function: requests offline printing of segments and multisegment files. This command does not work on standard object segments.

Arguments:
paths
are pathnames of segments and multisegment files; the star convention is not allowed.

Control arguments: affect only pathnames that follow them.
-brief, -bf
suppresses the message "j requests signalled...."
This control argument cannot be overruled later in the command line.
-copy N, -cp N
prints N copies, where N <= 4. (DEFAULT -- 1 copy)
-queue N, -q N
prints paths in priority queue N, where N <= 4. (DEFAULT --depends on request type specified.)
-delete, -dl
deletes paths after printing. This control argument cannot be overruled later in the command line.

-header STR, -he STR
identifies subsequent output by the string STR. (DEFAULT -- the requestor's Person_id)
-destination STR, -ds STR
uses the string STR to determine where to deliver the printed output. (DEFAULT -- the requestor's Project_id)
-notify, -nt
sends confirmation of completed output.
-request_type STR, -rqt STR
places paths in the queue identified by STR. (DEFAULT -- printer)
-forms STR
specifies the type of forms to be used when processing the print file. Standard I/O daemon drivers ignore the forms specification when processing print files.

-indent N, -ind N
indents left margin N columns. (DEFAULT -- 0)
-line_length N, -ll N
continues lines longer than N characters on the next line. (DEFAULT -- depends on the request type specified)
-page_length N, -pl N
prints no more than N lines on a page. (DEFAULT -- depends on the request type specified)
-no_endpage, -nep

skips to the top of a page only when a formfeed character is encountered in the input path.

-single, -sg
prints any form-feed or vertical-tab character in input as a newline character.

-truncate, -tc
truncates any line exceeding the line length (rather than "folding" onto subsequent lines).

-label STR, -lbl STR
puts STR at the top and bottom of every page.

-top_label STR, -tlbl STR
puts STR at the top of every page.

-bottom_label STR, -blbl STR
puts STR at the bottom of every page.

-access_label, -albl
puts access class of path at the top and bottom of every page.

-no_label, -nlbl
does not place any labels on the printed output.

-non_edited, -ned
prints nonprintable control characters as octal escapes.

Access required: At least "r" access to the segment or multisegment file.

The process which performs the printing (as obtained by the "print_request_types" command) must have at least "r" access to the segment or multisegment file and at least "s" access to the containing directory to verify that the user also has at least "r" access to the segment or multisegment file.

If -delete is specified, the I/O coordinator (normally IO.SysDaemon.z) must have at least "m" access to the containing directory and at least "s" access to the parent directory of the containing directory to verify that the user also has at least "m" access to the containing directory.

Notes: If invoked without any arguments, dprint gives the status of the default printer queue.

dpunch

11/13/81 dpunch, dpn

Syntax: dpn -control_args paths

Function: queues specified segments and/or multisegment files for punching by the Multics card punch. It is similar to the dprint command.

Arguments:
paths
   are pathnames of segments and/or multisegment files; the star convention is NOT allowed.

Control arguments: affect only pathnames that follow them.
-brief, -bf
   suppresses the message "j requests signalled, ...".
   This control argument cannot be overruled later in the command line.
-copy N, -cp N
   punches N copies, where N <= 4. (DEFAULT -- 1 copy)
-delete, -dl
   deletes paths after punching. This control argument cannot be overruled later in the command line.
-destination STR, -ds STR
   uses the string STR to determine where to deliver the deck. (DEFAULT -- the requestor's Project_id)

-header STR, -he STR
   identifies subsequent output by the string STR. (DEFAULT -- the requestor's Person_id)
-mcc
   punches the specified paths using character conversion. (DEFAULT)
-notify, -nt
   sends confirmation of completed output.
-queue N, -q N
   punches specified paths in priority queue N (N <= 4). (DEFAULT -- depends on the request type specified.)

-raw
   punches the specified paths using no conversion.
-request_type STR, -rqt STR
   places paths in the queue identified by STR. (DEFAULT -- punch)
-7punch, -7p
   punches the specified paths using 7-punch conversion.

Access required: At least "r" access to the segment or multisegment file.

The process which performs the punching (as obtained by the

"print_request_types" command) must have at least "r" access to the segment or multisegment file and at least "s" access to the containing directory to verify that the user also has at least "r" access to the segment or multisegment file.

If -delete is specified, the I/O coordinator (normally IO.SysDaemon.z) must have at least "m" access to the containing directory and at least "s" access to the parent directory of the containing directory to verify that the user also has at least "m" access to the containing directory.

Notes: If invoked without any arguments, dpunch gives the status of the default punch queue.

dump_segment

12/07/81 dump_segment, ds

Syntax: ds path offset length -control_args
or: ds seg_no offset length -control_args

Syntax as active function: [ds path offset -control_args]
or: [ds seg_no offset -control_args]

Function: prints, in octal or hexadecimal format, selected portions of a segment. It prints out either four or eight words per line and can optionally be instructed to print out an edited version of the ASCII, BCD, EBCDIC (in 8 or 9 bits), or 4-bit byte representation.

The active function returns a single word in octal or hexadecimal representation.

Arguments:

path
is the pathname or (octal) segment number of the segment to be dumped. If path is a pathname, but looks like a number, the preceding argument should be the -name (or -nm) control argument (see below). The star convention is allowed for the command only.

offset
is the (octal) offset of the first word to be dumped. If both offset and length are omitted, the entire segment is dumped.

length
is the (octal) number of words to be dumped. If offset is supplied and length is omitted, 1 word is dumped.

seg_no
is the octal segment number of a segment to be dumped.

Control arguments:

-4bit
prints out, or returns, a translation of the octal or hexadecimal dump based on the Multics unstructured 4-bit byte. The translation ignores the first bit of each 9-bit byte and uses each of the two groups of four bits remaining to generate a digit or a sign.

-address, -addr
prints the address (relative to the base of the segment) with the data. This is the default.

-bcd
prints the BCD representation of the words in addition to the octal or hexadecimal dump. There are no nonprintable BCD characters, so periods can be taken literally. This control argument causes the active function to return BCD.

-block N, -bk N
dumps words in blocks of N words separated by a blank line. The offset, if being printed, is reset to initial value at the beginning

of each block.

-character, -ch, -ascii
prints the ASCII representation of the words in addition to the octal or hexadecimal dump. Characters that cannot be printed are represented as periods. This control argument causes the active function to return ASCII.

-ebcdic9
prints the EBCDIC representation of each 9-bit byte in addition to the octal or hexadecimal dump. Characters that cannot be printed are represented by periods. This control argument causes the active function to return 9-bit EBCDIC.

-ebcdic8
prints the EBCDIC representation of each eight bits in addition to the octal or hexadecimal dump. Characters that cannot be printed are represented by periods. If an odd number of words is requested to dump, the last four bits of the last word do not appear in the translation. This control argument causes the active function to return 8-bit EBCDIC.

-entry_point name, -ep name
specifies that the offset of the first word to be dumped is relative to the location defined by the externally available symbol "name". This control argument can only be used for object segments (created by a compiler or by the create_data_segment program).

-header, -he
prints a header line containing the pathname (or segment number) of the segment being dumped as well as the date-time printed. The default is to print a header only if the entire segment is being dumped.

-hex8
prints the dumped words in hexadecimal with nine hexadecimal digits per word rather than octal with 12 octal digits per word.

-hex9
prints the dumped words in hexadecimal with eight hexadecimal digits per word rather than 12 octal digits per word. Each pair of hexadecimal digits corresponds to the low-order eight bits of each 9-bit byte.

-long, -lg
prints eight words on a line. Four is the default. This control argument cannot be used with -character, -bcd, -4bit, -ebcdic8, -ebcdic9, or -short.

-name PATH, -nm PATH
indicates that PATH is a pathname even though it may look like an octal segment number.

-no_address, -nad
does not print the address.

-no_header, -nhe
suppresses printing of the header line even though the entire segment is being dumped.

-no_offset, -nofs
does not print the offset. This is the default.

-offset N, -ofs N
prints the offset (relative to N words before the start of data being dumped) along with the data. If N is not given, 0 is assumed.

-short, -sh

compacts lines to fit on a terminal with a short line length. Single spaces are placed between fields, and only the two low-order digits of the address are printed, except when the high-order digits change. This shortens output lines to less than 80 characters.

Notes:
Only one of the control arguments -ebcdic8, -ebcdic9, -character, -bcd, or -4bit can be specified.

When invoked as an active function, dump_segment returns only one word of information, which is located at offset within the segment. If the -4bit, -bcd, -character, -ebcdic9, -ebcdic8, -hex8, or -hex9 control arguments are invoked, the information is returned in the specified format only. All other arguments are ignored in active function invocation.

---

ed

30/12/83 ed

Syntax: ed

Fonction: Appelle l'editeur de texte ed.

Note:
Pour plus de precisions on pourra consulter le segment info ed.gi par la commande: help ed.gi. Ou s'adresser a D. Arditti CNET/PAA/TIM/MTI (Tel: 638 55 05).

ed.gi

30/12/83 ed

Generalites:

ed est un editeur de texte fonctionnant avec des numeros de ligne physiquement presents dans le segment edite. Ceci est un gros avantage lorsqu'on veut ecrire, modifier ou mettre au point des programmes ecrits en fortran ou en basic (et meme en pl1). En effet ces numeros de lignes ne sont pas modifies lorsqu'on effectue des insertions ou des suppressions (ceci contrairement a ce qui se passe avec des numeros logiques). D'autre part ces numeros de ligne permettent d'utiliser de maniere particulierement commode les possibilites d'edition locales offertes par certains terminaux (mode message ou ligne, mode page ou bloc). Enfin ces numeros sont reconnus par probe.

ed a ete ecrit en utilisant des modules empruntes a l'editeur de texte Multics FAST. Pour cette raison ed ressemble beaucoup a FAST cependant on a essaye d'apporter quelques ameliorations et commodites supplementaires et surtout on s'est attache a supprimer toutes les restrictions d'utilisation de Multics introduites (volontairement) dans FAST. Comme FAST ressemble a l'ancien editeur de GCOS les nostalgiques de GCOS seront probablement combles.

ed est a priori pense pour etre utilise avec un terminal ayant des possibilites d'edition locales: mode ligne (ou message), mode page (ou bloc). Ceci explique le petit nombre de commandes disponibles. Cependant ed peut fonctionner avec n'importe quel type de terminal.

Fonctionnement:

A l'entree dans ed (apres avoir execute la commande: ed) l'utilisateur dispose d'un buffer de travail vide.

L'utilisateur peut alors envoyer.

- Des "lignes de texte".
- Des commandes de ed.
- Des commandes Multics.

Les "lignes de texte":

Les "lignes de texte" sont reconnues par ed parce qu'elles commencent par un nombre (precisement le premier caractere non blanc envoye est un chiffre). Ce nombre est le numero de ligne. La ligne de texte (numero de ligne compris) vient s'inserer dans le buffer de travail a la place indiquee par ce numero. Si une ligne existe deja avec ce meme numero elle est detruite et remplacee par la nouvelle. Pour supprimer une ligne frapper simplement son numero suivi d'un CR.

Les commandes de ed:

Les commandes de ed sont les suivantes (nom suivit de son abreviation):

- print_text p
- read_text r
- write_text w
- locate l
- substitute s
- delete_text d
- new n
- resequence rsq
- input input
- num num
- info ?
- move_text mt
- merge_text mgt
- quit q
- quit_force Q

Pour obtenir une description precise de chacune de ces commandes faire (apres etre entre dans ed) : help <nom de commande>.

Les commandes Multics:

Lorsqu'il ne s'agit ni d'une ligne de texte ni d'une commande de ed la commande frappee par l'utilisateur est consideree comme une commande Multics et executee normalement. Ce point permet de realiser n'importe quelle operation sans quitter ed : comme FAST, ed est un sous systeme mais contrairement a FAST, ed n'apporte aucune limitation a Multics.
Si l'on veut executer une commande Multics commencant par un chiffre ou portant le meme nom qu'une commande de ed il suffit de faire preceder cette commande par un ";".

Le pathname par defaut:

Le buffer de travail possede un pathname par defaut. C'est ce pathname qui sera utilise par la commande write_text (w) pour sauver ce buffer lorsque la commande w est utilisee sans pathname. Ce pathname se decompose en directory par defaut et nom par defaut. Le pathname par defaut est initialise par les commandes read-text (r) et new (n) avec pathname (s'il n'a pas ete initialise il est vide). La commande w avec pathname modifie le pathname par defaut. la commande info (?) permet entre autre informations de connaitre la directory par defaut et le pathname par defaut.

"L'abbrev PN":

Lorsqu'on travaille en mode "abbrev" ed initialise et met constament a jour une "abbrev" nommee PN (PathName) dont le contenu est le pathname par defaut prive (lorsqu'il existe) de son suffixe .fortran .basic ou .pl1.
Cette "abbrev" peut etre utilisee de diverses manieres:
- Pour compiler (en fortran): ft PN -ln
- pour executer le code objet : PN

On pourra de plus creer le jeu d'"abbrev" suivant qui est tres commode:
- FT do "ft PN -ln -ot"
- RUN do "ft PN -ln -tb;PN"

La premiere compile (en fortran) avec l'option -ot (optimiseur). La seconde compile (en fortran) avec l'option -tb (pour utiliser avec probe) et execute.

Commande Multics "piegees".

Les commandes Multics suivantes (nom suivi de l'abreviation) sont "piegees" par ed.

- ready_off rdf
- ready_on rdn
- abbrev ab
- .quit .q
- logout L

Dans ce contexte "piegee" signifie executee sous le controle de ed. Comme explique plus haut le ";" permet d'echapper a ce controle. Il est alors extremement dangereux d'utiliser le ";" dans ce but.

Message ready:

Apres chaque commande Multic on obtient dans les conditions habituelles le message ready standard. Apres certaines commandes de ed on obtient un message ready precede de "ed" ou "ed ". Le " " signifie que le buffer de travail n'a pas ete sauve.

edm

07/23/75 edm

Syntax: edm path

Function: creates or edits ASCII segments.

Arguments:
path
is the pathname of the segment to be edited.

Notes:
For the "s" and "c" requests, the delimiter may be any character not in the strings s1 and s2; c/a/b/ and cxaxbx work the same. If the first string is empty, characters go in at the front.

For the "q" request, if a "w" has not been done since the last change to the text then edm warns the user that changes made may be lost and asks whether the user still wishes to exit. If no changes have been made since the last "w" then the user exits directly. The "qf" request bypasses this check.

Modes: edm has three modes -- input, edit, and comment. If the path argument is specified and the segment is found, edm begins in edit mode; otherwise, it begins in input mode.

In edit mode, edm accepts and performs edit requests.

In input mode, all lines typed are appended to the file until a line consisting of a period (".") is typed, causing it to return to edit mode.

In comment mode, one line at a time of the file is printed without carriage return, and the user can append to the end of the line by typing a continuation, or can type "." to cause a return to edit mode.

Requests: in edit mode the following are valid.
. enter input mode; exit when a line with only "." is typed
- N back up N lines
, enter "comment" mode; exit when a line with only "." is typed
= print current line number
b go to bottom of file, enter input mode
c N /s1/s2/ change all occurrences of string "s1" to "s2" for N lines
d N delete N lines
updelete delete all lines above current line
E line execute "line" as a Multics command line
f string find a line beginning with "string"
i line insert "line" after current line

merge path insert segment "path" after current line
move M N beginning with line M, remove N lines and insert them after the current line.

```
k  enter brief mode (no response after f, n, l, c, s)
l string  locate a line containing "string"
n N  move down N lines
p N  print N lines
q  exit from edm (See Notes)
qf exit directly from edm with no question
r line  replace current line with "line"
s N /s1/s2/  same as "c"
t  go to top of file
v  enter verbose mode (opposite of "k")
w path  write edited copy of file into "path" (See Notes)
upwrite path  write and delete all lines above current line into "path"
```

emacs

02/24/82 emacs

Syntax: emacs Pathnames -control_args

Function: invokes the emacs editor.

Arguments:
Pathnames
One or more pathnames to be read into emacs buffers. They may be starnames or archive component names. If the -mc or -ap control argument is used, they may be interpreted differently or not at all.

Control arguments:
-no_starup, -ns
Inhibits execution of the user's start_up.emacs file. The start_up.emacs is executed by default.
-query
Queries the user for the terminal type to be used by emacs. The default is to try to determine the terminal type from the communications terminal type set with set_tty or the ttp preaccess request, and to query if no terminal type can be found that way.
-ttp
CTLname specifies that emacs should use CTLname as a terminal type. CTLname may be the absolute or relative pathname of an emacs terminal ctl program, or a reference name of same. See >doc>ss>emacs>ctl-writing.info for more details. The default is explained under the -query control argument.

-mc
PROGRAM loads the lisp program located at the pathname PROGRAM into the lisp environment when emacs is initilaized.
-apply
FUNCTION Args, -ap FUNCTION Args Runs the lisp function FUNCTION as the first function in the emacs environment, in place of the usual start_up mechanism. The arguments Args are given to FUNCTION as arguments. If this argument is used, the PATHNAME arguments are interpreted by this function, if at all.

enter_abs_request

10/01/80 enter_abs_request, ear

Syntax: ear path -control_args

Function: requests that an absentee process be created. This process executes commands from a control segment. The control segment is a list of input lines to the process (type "help exec_com").

Arguments:
path
is the pathname of the absentee control segment; the absin suffix need not be given.

Control arguments:
-arguments STRs, -ag STRs
passes the arguments STRs to the absentee process. This control argument must come last, since everything after it on the command line is taken as arguments to the absentee process.
-brief, -bf
suppresses the message "N already requested...".
-comment STR, -com STR
associates a comment with the request; it can be printed by the lar command. STR must be enclosed in quotes if it contains spaces or other command language special characters.
-deferred_indefinitely, -dfi
delays the creation of the absentee process indefinitely. It is created when the operator releases the request.

-foreground, -fg
enters the request into the foreground queue; the default is one of the priority queues (see the "-queue" control argument).
-limit N, -li N
places a limit of N seconds on the CPU time the absentee process uses.
-long_id, -lgid
prints the long request_id. Default is to print the short request_id. Type "help request_ids".
-notify, -nt
notifies the user of whether a job is logged in, logged out, or deferred.
-output_file path, -of path
specifies the output segment.

-proxy User_id
causes the absentee process to be created with the specified User_id. This feature is primarily for the RJE facility; its use is controlled by the system administrator.
-queue N, -q N
indicates the priority queue; the default queue is defined by the system administrator. For convenience in writing exec_coms and abbreviations, the word foreground or fg following the -queue

control argument performs the same function as the -foreground control argument.

-resource STR, -rsc STR
specifies the resources (e.g., tape drives) that are needed by the absentee process. The process is not created until the resources are available. STR must be enclosed in quotes if it contains blanks or other delimiters. Type "help reserve_resource" for a description of the syntax of STR.
-restart, -rt
starts the computation over again from the beginning if interrupted (e.g., by a system crash). The default is not to restart.
-secondary, -sec
indicates that a foreground request should be logged in as a secondary process (subject to preemption) if no primary slots are available.
-sender STR
enters only requests from sender STR; usually an RJE station identifier.
-time dtime, -tm dtime
delays creation of the absentee process until a specified time.

Notes: Unless it says otherwise, an error message means that the request was not submitted.

enter_retrieval_request

09/17/81 enter_retrieval_request, err

Syntax: err path -control_args

Function: queues volume retrieval requests for specific segments, directories, multisegment files, and subtrees.

Arguments:
path
is the pathname of a segment, directory, or node of a subtree. The star convention is not allowed.

Control arguments:
-brief, -bf
supresses printing of the ID and number of requests in queue.
-from DT, -fm DT
specifies that the search for path and all inferior branches, if specified, stops at time DT. Thus, objects dumped before time DT are not recovered. Time DT must be acceptable to the convert_date_to_binary_ subroutine. If the control argument is not specified, all valid dump volumes are searched.
-long, -lg
prints the long ID of the request. The default is to print the short ID.
-multisegment_file, -msf
specifies that the object named in path is a multisegment file and that all of its components are to be recovered.

-new_path newpath
specifies that if the requestor has the correct access to retrieve the segment specified in path above (which must already exist) and the correct access to create a segment with the pathname newpath, then the object described/identified by path is retrieved into newpath.
-notify, -nt
specifies that the user is to be notified by online mail of the success or failure of the request. The default is to not notify the user.
-previous, -prev
specifies that the object to be retrieved is the one dumped prior to the object presently online. The default is to always retrieve the most recent copy. By specifying this control argument, the requestor can retrieve successively earlier copies of an object.

-queue N, -q N
queues requests in priority queue N. The default is queue 3.
-subtree, -subt
specifies that the subtree inferior to the directory specified in path as well as the directory is to be retrieved. If a subtree is

found intact after a directory is recovered, then no further action is taken, unless a time interval has been specified. See "Notes" for more information. The default is not to retrieve subtrees.

-to DT

specifies that the search for path and all inferior branches, if specified, proceeds from time DT backwards. Thus, objects dumped later than time DT are not recovered. DT must be acceptable to the convert_date_to_binary_ subroutine. If this control argument is not specified, time DT is assumed to be the start of the retrieval operation.

Access required:
The user must have write access or modify permission to an object in order to retrieve it. If an object has been deleted, then append permission on the containing directory is also required.

Notes: In certain cases where a directory is damaged, the inferior subtree may be unavailable until the directory is recovered. When a directory is recovered, and the subtree control argument is specified, a check is made to see if the subtree is available, and if so, retrieval is assumed complete.

Retrieval requests of objects for which the online copy is more recent or the same as the dump copy are refused, unless the -previous, -from, or -to control arguments are used.

The pathnames of the segments and directories to be retrieved need not be specified as a set of primary names. Any set of valid entrynames is acceptable.

exec_com

01/11/82 exec_com, ec

Syntax: ec path ec_args

Function: executes programs written in the exec_com language, used to pass command lines to the Multics command processor and pass input lines to commands reading input. The syntax described here is known as Version 2, for which the first line of the exec_com program must be the line consisting of "&version 2". For a description of Version 1 syntax, type "help vlec".

Arguments:

path
: is the pathname of an exec_com program, written using the constructs described in this info segment. The ec suffix is assumed if not specified. The star convention is NOT allowed.

ec_args
: are optional arguments to the exec_com program, and are substituted for parameter references such as &1. See "List of parameters".

Syntax as an active function: [ec path ec_args]

List of parameters:

&1 - &9
: expand to the 1st through 9th ec_args, or to defaults defined by a &default statement or to null string if there is no corresponding ec_arg. The string &0 is invalid.

&(1) - &(9)
: are synonyms for &1 - &9.

&(11), &(12), etc.
: expands to the corresponding ec_arg, or to a default defined by &default or to null string if there is no corresponding ec_arg. The parentheses are required when there are two or more digits.

&q1 - &q9

&q(1), &q(11), etc.
: expands to the corresponding argument with quotes doubled according to the quote depth of the surrounding context. See "Notes on quoting". This parameter ensures that quotes in the argument to exec_com are handled correctly under the quote-stripping action of the command processor.

&r1 - &r9

&r(1), &r(11), etc.
: expands to the corresponding argument enclosed in an added layer of quotes, and internal quotes doubled accordingly. See "Notes on quoting". This parameter keeps the value of the argument as a single unit after one layer of quote-stripping by the command processor.

&n
   expands to the number of ec_args specified to exec_com.
&f1 - &f9
&f(1), &f(11), etc.
   expands to a list of the Nth through last ec_args separated by
   spaces. If N is greater than the value of &n, expands to null string.
&qf1 - &qf9
&qf(1), &qf(11), etc.
   expands to a list of the Nth through last ec_args, with quotes
   doubled, separated by spaces. If N is greater than the value of &n,
   expands to null string. This parameter is equivalent to:
   &qN &qN+1 &qN+2 ....

&rf1 - &rf9
&rf(1), &rf(11), etc.
   expands to a list of the Nth through last ec_args, individually
   requoted, separated by spaces. If N is greater than the value of
   &n, expands to null string. This parameter is equivalent to:
   &rN &rN+1 &rN+2 ....
&f&n, &qf&n, &rf&n
   expands to the last ec_arg specified to exec_com, either as is,
   with quotes doubled, or requoted.

&ec_dir
   expands to the pathname of the directory containing the exec_com
   currently running. It can be used to call other exec_com's in the
   same directory.
&ec_name
   expands to the entryname of the exec_com currently running, with
   any ec or absin suffix removed (the absin suffix is for an exec_com
   invoked by the absentee facility; type "help ear").
   This parameter can be used to simulate entrypoints in an exec_com
   segment, by adding multiple names to the segment and transferring
   to a different &label depending on the name invoked.
&ec_path
   expands to the expanded, suffixed pathname of the current exec_com.
&ec_switch
   expands to the name of the I/O switch over which the exec_com
   interpreter is reading the exec_com.

List of value expressions:
   (All of these constructs can be nested arbitrarily inside each
   other.)
&(NAME)
   expands to the value assigned to the variable NAME by a previous
   &set statement in the same exec_com. If NAME contains &'s, it is
   first expanded. Therefore, &() constructs can be nested. However,
   &'s in the expansion are not re-expanded. A second level of
   expansion must be specified, therefore, by &(&()).
   If NAME has not been assigned a value by &set, an error occurs.
   Variable names are allowed to contain any characters except &
   and cannot consist solely of digits.
&(N)
   where N is a positive integer, expands to the value of the Nth
   ec_arg to exec_com, or if there is no Nth ec_arg, to the last
   default value assigned to argument N by a &default statement, or if

no default value was assigned, to null string.

&q(NAME), &q(N)
expands to the same thing as &(NAME) or &(N), but with quotes inside the value doubled according to the quote depoth of the surrounding context.
&r(NAME), &r(n)
expands to the same thing as &(NAME) or &(N), but requoted and with internal quotes doubled.
&[ACTIVE STRING], &||[ACTIVE STRING]
expands to the return value of an active string by calling the command processor. This construct ends with the matching right bracket. The &||[...] construct is used in &set statements to treat the expansion as a single argument to &set. It is important to note that &[...] active strings are expanded by exec_com, whereas [...] strings are expanded at command line execution time. Therefore, ||[...] and not &||[...] must be used in a command line to treat the expansion as a single command argument.

List of literals:
Also see "Notes on white space".
&"..."
encloses an arbitrary character string to be taken literally. Quotes inside the string must be doubled, and the closing undoubled quote ends the literal string.
&&
expands to a single & character, not further expanded.
&AMP, &AMP(N)
expands to a single ampersand character (ASCII 046), in which case it is identical to &&, or to N ampersands where N is a positive integer.
&SP, &SP(N)
expands to a single space character (ASCII 040) or to N spaces.
&BS, &BS(N)
expands to a single backspace character (ASCII 010) or to N backspaces.

&HT, &HT(N)
expands to a single horizontal tab character (ASCII 011) or to N horizontal tabs.
&VT, &VT(N)
expands to a single vertical tab character (ASCII 013) or to N vertical tabs.
&FF, &FF(N), &NP, &NP(N)
expands to a single form-feed character (ASCII 014) or to N form-feeds.
&NL, &NL(N), &LF, &LF(N)
expands to a single newline character (ASCII 012) or to N newlines.
&QT, &QT(N)
expands to a single double-quote character (") or to N of them.
&!
expands to a Multics 15-character unique name, for example "!BBBhjBnWQpGbbc". Multiple occurrences of &! within the same exec_com expand to the same string.

List of predicates:

&is_defined(NAME)
expands to "true" if the variable named NAME has been assigned a value by an &set statement in the current exec_com, "false" otherwise (See "Notes on variables"). This construct expands to "true" if &(NAME) can be expanded, "false" if &(NAME) is an error.
&is_defined(N)
where N is a positive integer, expands to "true" if an Nth ec_arg was specified to exec_com or an Nth default was defined via the &default statement (see "List of assignment statements"), "false" otherwise.
&is_absin
expands to "true" if the exec_com is being executed by the absentee facility, "false" if it is being executed by the exec_com command or active function.
&is_active_function, &is_af
expands to "true" if the exec_com is being executed by the exec_com active function, "false" otherwise.
&is_attached
expands to "true" if input is currently attached via an &attach statement, "false" otherwise. See "Notes on input attachment". Input is always attached when running as an absentee.
&is_input_line
expands to "true" if the line in which it appears is being read as an input line by some command, "false" otherwise.

List of control statements:
&attach
causes any commands subsequently invoked in command lines to read their input from the exec_com rather than from the terminal. See "Notes on input attachment".
&detach
causes any commands subsequently invoked in command lines to read their input from the terminal. This is the default. See "Notes on input attachment".

&if EXPRESSION
expands EXPRESSION to get a true or false value. EXPRESSION can contain any exec_com-expandable constructs, such as &[...] (See "List of value expressions"). If the expanded value of EXPRESSION is "true", the following &then statement (if any) is executed next. If the value is "false", the following &else statement (if any) is executed next. If the value is neither "true" nor "false", an error occurs. See "Examples of if statements".
&then LINE
&then &do LINES &end
&else LINE
&else &do LINES &end
where LINE is any exec_com line, including another &if statement. LINE is executed or not depending on the value of the preceding &if clause. The &then and &else statements, unlike other exec_com statements, are allowed to appear on the same line with one another and with &if. See "Examples of if statements".
The contents of a &do-&end block reference the same variables as the containing exec_com. No &goto's are allowed into a &do-&end block from outside it.

&goto LABEL

causes the next statement to be executed to be the statement following the first occurrence of "&label LABEL" in the exec_com.

&label LABEL
specifies a target for "&goto LABEL" and is otherwise ignored. The string LABEL can contain any characters.

&quit
terminates execution of the exec_com. If the program was invoked by the exec_com active function, the active function return value is null string.

&return LINE
terminates execution of the exec_com. If the program was invoked by the exec_com active function, the active function value is the (expanded) value of LINE, the rest of the line. If the program was invoked by the exec_com command, the expanded value of LINE is printed on the terminal.

List of assignment statements:

&set NAME1 VALUE1 ... NAMEn VALUEn
assigns values to the variables NAME1 through NAMEn, which are created if no assignments for them already exist. All NAMEj and VALUEj arguments are fully expanded before any values are set. Therefore, the statement:

&set a &(b) b &(a)
exchanges the values of the variables a and b. Arguments to &set are delimited by white space. White space and literals inside them must be enclosed in "...", for example:

&set answer "&[response Answer?]"
Alternatively, the &||[...] construct can be used, causing the entire return value to be taken as a single argument:

&set answer &||[response Answer?]
There is no restriction on the lengths of NAMEj or VALUEj; NAMEj cannot be all digits. If VALUEj is the unquoted keyword &undefined, any existing value for NAMEj is deleted, and the &is_defined(NAMEj) construct will expand to "false".

&default VALUE1 ... VALUEn
assigns default values for the exec_com parameters &(1) through &(n). The default value of &(j) only matters if no jth ec_arg was specified to exec_com. The &(j) parameter reference expands to the value of the jth ec_arg, or if there is none, to the jth default value set by &default, or if there is none, to null string. VALUEj arguments are separated by white space, and each is fully expanded before default values are set. White space and literal 's in them must be enclosed in &"...". If VALUEj is the keyword &undefined, no jth default value is set. This keyword is used as a place-holder to skip the jth position.

List of printing statements:

&print LINE
prints the expanded remainder of the line, followed by a newline character. If &print appears on a line by itself, a single newline character is printed.

&print_nnl LINE
prints the expanded remainder of the line, without appending a

newline character.

List of tracing statements:
&ready on
&ready off
turns ready messages on or off. Turning them on causes the system ready procedure to print a ready message when it is called. The default is off. This statement does not affect whether the ready procedure is called. The ready procedure is normally called after the execution of a command line (type "help ready_on"). The &ready statement is ignored in the absentee environment.
&ready_proc on
&ready_proc off
determines whether or not the system ready procedure is called after each command line is executed. The default is on for the exec_com command, off for the active function. This statement is ignored in the absentee environment.

&trace TYPES STATE &prefix PREFIX &osw SWITCHNAME
sets tracing for one or more kinds of lines specified by TYPES. TYPES can be any combination of the following:
&command command lines.
&comment comments, including those sharing other lines.
&control control lines, for example &print....
&input lines being read as input to some command.
The default if TYPE is omitted is all four types.

STATE can be one of the following:
off, false disables tracing entirely.
on, true enables tracing, in whichever of the following modes was last specified. The default mode is "&expanded" for command and input lines, "&both" for control lines.
(continued)

&unexpanded prints lines as they appear in the exec_com segment. Implies "on".
&expanded prints lines after all expansion has been done. Implies "on".
&all prints at each stage of expansion. Implies "on".
&both prints each line as it appears in the exec_com, and again after all expansion. Implies "on".

Defaults for ec's invoked by the exec_com command/active function are "&expanded" for command and input lines, "&unexpanded" for control lines, and "off" for comments.
Defaults in the absentee environment are "&expanded" for command and control lines, "off" for control lines and comments.

PREFIX specifies a string to be printed at the start of each line. Default prefixes are all null string.

SWITCHNAME specifies an I/O switch on which to write the trace. The default for all types of lines in ec's invoked by the exec_com command or active function is user_output. The default in the absentee environment is user_io.

Notes on absentee environment:
An exec_com/absin runs in the absentee environment only when it has been invoked directly by the absentee facility, ie. is running an absentee process. Exec_com's called within an absentee process are said to run in the normal exec_com environment.

Input lines in an absentee process come from the absin segment running the process. These, along with output lines, are directed to an absout file. Since both input and output lines are written to the same switch, the default switch is chosen to be user_io for the absentee environment rather than user_output as for exec_com's. This default applies to all tracing, and ensures that even if user_output is redirected somewhere, the input lines driving the process still appear in the absout.

The &attach and &detach statements have no effect in the absentee environment, since input to the absentee process always comes from the absout file. The &is_attached predicate always returns true.

The &ready and &ready_proc statements also have no effect in the absentee environment. Instead, the ready_on and ready_off commands should be used.

Notes on version:
The current version of exec_com is known as Version 2. In many ways similar to the old Version 1, it adds automatic variables, parameter defaults, literal character escapes, indentation, comments on lines, line continuation, expansion of active strings in control lines, and tracing of comments and control lines.

In addition, there are two incompatible changes between the versions. Whereas V1 leaves unrecognized &strings alone, V2 rejects them as syntax errors. This change makes V2 an extensible language. Secondly, V2 parses lines into control keywords and tokens (separated by whitespace) before expansion, so that expansion can only change the values of tokens but not the syntax of a line.

A Version 2 exec_com has "&version 2" as its first line. If this first line is not present, the exec_com is interpreted as Version 1. Version 1 exec_com's can optionally begin with "&version 1"; at some future time, Version 2 will be the default and "&version 1" will be required.

A conversion command is available to translate Version 1 exec_com's to Version 2. Type "help cvec".

Notes on white space:
White space (SPACE, HORIZONTAL TAB, VERTICAL TAB, and FORM-FEED) is ignored at the beginning and end of each line. As a result, exec_com lines can be indented freely. Intentional white space at the beginning or end of a line (for example, an editor input line) must be specified by literal escapes such as &SP. See "List of literals".

Notes on comments:

Comments are specified by the character sequence &- anywhere in a line. Where this sequence appears (outside of &"..."), the remainder of the line is a comment and can contain any characters. White space preceding the comment, if any, is ignored. Therefore, comments can be aligned at a particular column without affecting the executable text.
White space before a comment can be specified by the literal escapes described in "List of literals".

Notes on continuation:
Long command lines and other portions of text that must not be broken can be continued on successive lines by means of the character sequence &+ at the beginning of each continuation line. White space preceding the &+ is ignored and whitespace following the &+ is part of the executable line.

Continuation is not affected by intervening comments, whether at the end of executable text lines or on lines by themselves. This feature can be used to comment parts of statements.

Notes on quoting:
The exec_com interpreter strips one layer of exec_com quotes (&"...") from the text. It does not perform command processor-type stripping of regular quotes ("...").

To defeat one or more levels of command processor quote-stripping, the values of variable and parameter expansions can be quote-doubled or requoted using the "q" and "r" prefixes. Quote-doubling doubles existing quote characters in a string according to the depth of quotes inside which the string is currently nested, so that one level of quote-stripping by the command processor will result in the internal quotes looking the same as they do inside the original string. Requoting goes a step further by first quote-doubling, then surrounding string with an additional layer of quotes, thus causing the entire string to remain a single argument after one level of quote stripping by the command processor. In the examples below, "Level" refers to the number of levels deep in quotes that the parameter reference appears in the exec_com text. Assume that the value of the first ec_arg to exec_com is the string a"b containing a single quote character:

```
 &1 &ql      &rl
Level 0 a"b a"b      "a""b"
Level 1 "a"b" "a""b"      """a""""b"""
Level 2 """a"b""" """a""""a"""   """""""a""""""""b"""""""
```

The exact number of quote characters is unimportant; the important thing is that &q protects internal quotes from one level of quote stripping by the command processor, and &r ensures that the value remains a single argument to the comand processor. These prefixes are very useful since, if the value of the first ec_arg (for example) contains a space, the value of &1 substituted into a command line will be parsed into more than one command line argument.

If a value is null, the &q prefix does not affect it, and the &r prefix results in a pair of quotes, doubled according to the quote depth of the context.

The "q" and "r" prefixes can be used in the following constructs:

```
 &ql, &q(l) &rl, &r(l)
 &qfl, &qf(l) &rfl, &rf(l)
 &q&n, &qf&n &r&n, &rf&n
 &q(VAR NAME) &r(VAR NAME)
```

Notes on input attachment:
By default, commands invoked by command lines within an exec_com read their input from the terminal. By preceding a command line with an &attach statement, the command can be caused to read input lines from the text of the exec_com instead. Note that "&attach" must precede the line on which the input-reading command is invoked. The &detach statement causes any later input-reading command to get its input from the terminal.

While &attach is in effect, the &is_attached predicate expands to "true"; after &detach, it expands to "false".

executive_mail

10/21/81 executive_mail, xmail

Syntax: executive_mail arg

Function: invokes the Executive Mail facility

Argument:
-multics_mode, -mm
activates function key 8 for use in Executive Mail. This function key invokes Multics command level from within the Executive Mail facility. If the terminal you are using does not have function keys, use the two character sequence "esc e" to invoke Multics command level.

Notes: To start the Executive Mail facility, type either "executive_mail" or "xmail" when a ready message (usually of the form "r 14:03 0.285 5") appears on the terminal screen just above the cursor. Users operate Executive Mail by selecting operations from lists called menus. All help needed to operate the system is available within the Executive Mail facility itself.

file_output

07/18/78 file_output, fo

Syntax: fo path -control_args

Function: Directs specified output I/O switches to a file. Attachments are reverted by the revert_output command.

Arguments:
path
   is the pathname of a segment or MSF.

Control arguments:
-ssw switchname
   to specify an I/O switch.
-truncate, -tc
   to truncate the output file.
-extend
   to extend the output file (DEFAULT).

Notes: If the specified file does not exist, it is created.
If it does exist, it is appended to unless -truncate is specified.
If path is not specified, the default is a segment named output_file in the working dir.
  If no switchnames are specified, the default is user_output.
  Any number of switchnames can be specified. Output directed to any of the switches is sent to the output file.
  To avoid getting ready messages in the output file, the fo and ro commands should appear on the same line.

Examples:
    fo user_file -ssw user_output -ssw error_output
  directs both switches to user_file.

fortran

08/29/80 new_fortran, fortran, ft

Syntax: new_fortran path -control_args

Function: invokes the new FORTRAN compiler.

Arguments:
path
is the pathname of a FORTRAN source segment; the fortran suffix need not be given.

Control arguments:
-ansi66
interprets the program according to the 1966 standard for FORTRAN, with Multics FORTRAN extensions. (Default)
-ansi77
interprets the program according to the 1977 standard for FORTRAN, with Multics FORTRAN extensions.
-brief, -bf
writes error messages in short form.
-brief_table, -bftb
generates partial symbol table giving correspondence between source line numbers and object locations.
-card
specifies source segment is in card-image format.

-check, -ck
checks source segment for syntactic and semantic errors. No object segment is produced.
-fold
maps uppercase letters to lowercase form.
-line_numbers, -ln
source segment has line numbers.
-list, -ls
produces complete source program listing plus an assembly-like listing.
-map
produces complete source program listing.
-non_relocatable, -nrlc
inhibits generation of relocation information by the compiler. The resulting object segment cannot be bound.

-optimize, -ot
invokes an extra compiler phase just before code generation to perform certain optimizations.
-profile, -pf
generates extra code to meter execution of individual statements.
-relocatable, -rlc
generates relocation information (Default).

-round
   rounds intermediate results of real and double precision
   calculations before storing. (Default)
-safe_optimize, -safe_ot
   like -optimize, but inhibits some code movement.
-severityN, -svN
   prints only error messages whose severity is N or greater (where N
   is 1, 2, 3, or 4). (Default is 1.)

-stringrange, -strg
   produces rnage checking code for all substring references.
-subscriptrange, -subrg
   produces range checking code for all subscripted array references.
-table, -tb
   generates full symbol table.
-time, -tm
   prints table giving time (in seconds), number of page faults, and
   size of temporary area for each phase of the compiler.
-time_ot
   prints out timing information on the sub-phases of the optimizer.
-truncate, -tc
   truncates intermediate results of real and double precision
   computations before storing.

Notes:
The control arguments -optimize and -safe_optimize are mutually exclusive with -stringrange and -subscriptrange. The control arguments -ansi66 and -ansi77 are mutually exclusive. The control arguments -round and -truncate are mutually exclusive.

Type "help fort_options.gi" for a description of the %options and %global statements. Type "help fortran_77.gi" for general information about FORTRAN 77 on Multics.

For more information on the FORTRAN compiler, refer to the Multics FORTRAN Reference manual, Order No. AT58. For more information on using FORTRAN on Multics, refer to the Multics FORTRAN User's Guide, Order No. CC70.

fortran_abs

11/13/81 fortran_abs, fa

Syntax: fa paths -ft_args -dp_args -control_args

Function: submits an absentee request to perform FORTRAN compilations.

Arguments:
paths
   are the pathnames of segments to be compiled.
ft_args
   are control arguments accepted by the fortran command.
dp_args
   are control arguments (except -delete) accepted by the dprint
   command.

Control arguments:
-queue N, -q N
   is the priority queue of the request. The default queue is defined
   by the system administrator. See the Notes for a description of the
   interaction with the dprinting of listing files.
-hold
   do not dprint or delete any listing files.
-output_file path, -of path
   put absentee output in segment path.
-limit N, -li N
   specifies time limit in seconds for the absentee job.

Notes:
Control arguments and paths can be mixed freely and can appear anywhere on the command line after the command.

Unpredictable results can occur if two absentee requests are submitted that simultaneously attempt to compile the same segment or write into the same absout segment.

If the -queue control argument is not specified, the request is submitted into the default absentee priority queue defined by the site and, if requested, the listing files will be dprinted in the default queue of the request type specified on the command line. (If no request type is specified, the "printer" request type is used.)

If the -queue control argument is specified, and, if requested, the listing files will be dprinted in the same queue as is used for the absentee request. If the request type specified for dprinting does not have that queue, the highest numbered queue available for the request type is used and a warning is issued.

general_ready

10/10/80 general_ready, gr

Syntax: gr -control_args

Function: prints a ready message containing specified values in a specified format.

Control arguments:

The following prefix control arguments must occur prior to any of the format control arguments described below. They allow the user to override the default formats for the contents of the ready message.

-string
allows the user to specify the character string at the beginning of the ready message to replace "r".

-control
allows the user to specify the entire ioa_ control string used to format the ready message. This control argument overrides any format arguments that would normally affect the format of the ready message.

The format and content of the ready message are controlled by the following format control arguments.

-inc_vcpu N
incremental virtual CPU value. N can be from 1 to 9, the default is 3.

-total_vcpu N
total virtual CPU value. N can be from 1 to 9, the default is 3.

-inc_mem_units N
incremental units. N can be from 1 to 9, the default is 3.

-total_mem_units N
total memory units. N can be from 1 to 9, the default is 3.

-inc_cost N
incremental cost charges. N can be from 1 to 9, the default is 2.

-total_cost N
cost charges. N can be from 1 to 9, the default is 2.

-inc_pf
incremental paging values.

-total_pf
paging values.

-level
command processor level numbers.

-date
eight-character date (mm/dd/yy).

-date_time
date and time (mm/dd/yy hhmm.m zzz www).

-day
two-digit day (dd).

-day_name

three-character day of the week (www).

-hour
two-digit hour (hh).
-minute
two-digit minute (mm)
-month
two-digit month (mm).
-time, -tm
six-character time of day (hhmm.m).
-year
two-digit year (yy)
-zone
three-character time zone (zzz).

The following control arguments affect the operation of general_ready, but do not change the format of ready messages.
-set
establishes general_ready as the current ready message procedure. The command processor then calls general_ready to print a ready message after each command line is complete. This control argument also causes general_ready to set an alarm timer to catch shift changes.
-revert
makes the system ready procedure the current ready message procedure and resets any timer alarms established by general_ready to catch shift changes.
-reset
resets incremental usage values to zero without printing a ready message.

-call cmdline
when used with the -set control argument, causes general_ready to call the command processor to execute cmdline after the completion of every command line. cmdline is a single argument to general_ready and therefore, must be enclosed in quotes if it contains any blanks. cmdline is executed even if the printing of ready messages has been inhibited by executing the ready_off command.

Syntax as active function: [gr -control_args]

get_quota

01/27/76 get_quota, gq

Syntax: gq paths -control_arg

Function: prints information about secondary storage quota and pages used.

Arguments:
paths
   are directory names; star convention can be used.

Control arguments:
-long, -lg
   specifies the long form of output.

Notes: The default output is the number of pages of quota assigned to the directory and the number of pages used by the segments in that directory and any inferior directories that are charging against that quota.
  The long form of output gives the quota and pages-used information provided in the short output, and also prints the cumulative time-page-product for the directory (for the current accounting period).

have_mail

02/02/79 have_mail

Syntax: [have_mail path]

Function: returns "true" if there is mail in the user's current default mailbox or in a specified mailbox.

Arguments:
path
is the optional pathname of a mailbox. If path is not specified, the user's default mailbox is assumed.

Command syntax: have_mail path

help

06/17/81 help

Syntax: help info_names -control_args

Function: selectively prints blocks of information (infos) from system info segments.

Arguments:
info_names
entrynames or pathnames of infos to be printed. Star convention allowed. Suffix of info assumed. For subroutines, final entryname is of the form:
subroutine_$entry_point_name
or -entry_point control argument can be used. Star convention not allowed for subroutine info names. See "Info names" below. If no info_names given, info for the help command is printed.

Basic Use:
When help is invoked without arguments, it prints a description of the help command. An info_name may be given to print information about a specific topic. For example, to print information about the list command, type:

```
help list
```

help begins by printing a heading line identifying the information being printed. It prints the first paragraph of information, then asks if the user wants more help. The user may give any answer listed under "List of responses" below. Possible responses include:

yes	print next paragraph
skip	skip next paragraph
rest	print remaining paragraphs
brief	print summary of command, active function, subroutine
no	stop printing information, go on to next selected info
quit	exit help command

Contents of Info Segments:
Information printed by help is stored in formatted segments called info segments. Each segment contains one or more information blocks (infos) which describe a particular command, active function, subroutine entry point, or other feature of the system.

Each info begins with heading line giving date info last modified, and title of info. Command and active function infos use command name (including short name) as title. Subroutine infos use subroutine entry point name.

Information following heading is divided into paragraphs from 1 to 15 lines in length. A paragraph is a logically complete unit containing a small amount of the total information.

Paragraphs are grouped together into sections describing a complete topic. Sections begin with a topic title. Syntax, Function, Arguments, Control arguments, Examples and Notes are typical section titles found in command and active function infos.

help remembers which paragraphs user has seen and which have been skipped or not yet reached. User asked to "Review" paragraphs seen before, or asked if "More help" is needed for unseen paragraphs. help stops printing if all paragraphs seen when the end of info reached. If any paragraphs skipped, help asks if user wants to see them. If user answers "yes", first unseen paragraph printed. User can answer "skip -seen" to view subsequent unseen paragraphs.

Control arguments: Control arguments are listed in four groups-- info selection, information selection, starting paragraph selection, and paragraph grouping.

INFO SELECTION

-pathname path, -pn path
   path identifies info segment to be printed. Star convention allowed. Suffix of info assumed. path may end with $entry_point_name when star convention not used.

-entry_point, -ep
   when info_name argument is given as subroutine_, prints description of subroutine_$subroutine_ entry point, rather than info describing general properties of all subroutine_ entry points. See "Info names" below.

INFORMATION SELECTION

-header, -he
   prints only long heading line, including pathname of info, info heading, info line count. -header conflicts with all other INFORMATION SELECTION control args.

-brief_header, -bfhe
   shortens the long heading printed by default. Instead, help prints brief heading followed by first paragraph, then asks if user wants to see more information. Brief heading includes info heading and line count.

-title
   prints section titles and section line counts, then asks if users wants to see first paragraph of info.

-brief, -bf
   prints only summary of command, active function or subroutine info, including: Syntax section, list of control arguments, etc.

-control_arg STRs, -ca STRs
   prints only descriptions of control (or other) arguments whose names contain one of the strings STRs. STRs must not include a leading minus sign (-).

-all, -a
   prints entire info or subroutine description without questions.

STARTING PARAGRAPH SELECTION

-section STRs, -scn STRs

begins printing at section whose title contains all strings STRs.
By default, printing begins at top.
-search STRs, -srh STRs
begins printing all in paragraph containing strings STRs.
By default, printing begins at top.

PARAGRAPH GROUPING
-minlines I
sets minimum paragraph size to I lines. Default is 4.
See "Grouping of paragraphs" below.
-maxlines J
sets maximum paragraph grouping size to J lines. Default is 15.
See "Grouping of paragraphs" below.

List of responses:
Following responses can be given to questions asked by help.
yes, y
prints next paragraph.
no, n
exits current info. Same as quit if this is last info.
quit, q

rest -scn , r -scn
prints rest of info, without intervening questions. If -section or -scn are given, prints only rest of current section without questions and then asks if user wants to see next section.
top, t
skips to beginning of info, ehtn asks if user wants to see first paragraph.
title -top
lists titles and line counts of sections which follow. If -top or -t given, lists all section titles. help repeats previous question after titles are printed.

section STRs -top ,
scn STRs -top
skips to next section whose title contains all strings STRs. If -top or -t given, title searching starts at top of info. If STRs omitted, uses STRs from previous section response or -section control argument.
search STRs -top ,
srh STRs -top
skips to next paragraph containing all strings STRs. If -top or -t given, searching starts at top of info. If STRs omitted, uses STRs from previous search response or -search control argument.

skip -scn -rest -seen -ep ,
s -scn -rest -seen -ep
skips next paragraph. If -section or -scn are given, skips all paragraphs of current section. If -rest or -r, -entry_point or -ep are given, skips rest of this info or subroutine entry point description, continuing with the next. If -seen is given, skips to next paragraph which user has not seen. Only one control argument allowed in each skip response.

```
brief, bf
   prints summary of command or subroutine info, including
   Syntax section, list of control arguments, etc.  help repeats
   previous question after summary is printed.
control_arg STRs, ca STRs
   prints descriptions of control (or other) arguments
   whose names contain one of the strings STRs.  help repeats
   previous question after descriptions are printed.


entry_point  EP_NAME , ep  EP_NAME
   skips to description of subroutine entry point EP_NAME.
   Form of EP_NAME is:
            entry_point_name
   or       subroutine_$entry_point_name

   If EP_NAME omitted, skips to description of subroutine_$subroutine_
   entry point.


header, he
   prints long heading line, including pathname of info, info heading,
   info line count.
?
   prints list of responses allowed to help queries.
.
   prints "help" to identify the current interactive environment.
.. command_line
   treats remainder of response as command line passed to Multics
   command processor.
```

Info names: are arguments for the help command which identify the info(s) to be printed. Info names may be pathnames or entrynames. Info names may end with $entry_point_name to identify a particular subroutine entry point. The next paragraph illustrates various types of info names.

```
FORM                          EXAMPLES
   entryname                     pl1, pl1.status.info, fortran*.**
   entryname$entry_point_name    hcs_$initiate, delete_$path
   entryname -ep                 com_err_ -ep, ioa_ -ep
   pathname                      >udd>Pubs>new>rename, <info_dir>a*.**
   -pn pathname                  -pn new_info
   pathname$entry_point_name     info>my_util$start_prog
   pathname -ep                  >exl>info>display_fort_ -ep
```

Pathnames contain < or > characters, or follow -pathname control argument. The path identifies segment containing the info to be printed.

Entrynames do not contain < or >. help searches for info segments in directories given in the info_segments (info_segs or info) search list. Type:

```
          print_search_paths info_segments
```

for a list of current help search paths. By default, the search

paths are:
```
>doc>iml_info
>doc>info
```

For pathnames or entrynames, star convention can be used to identify several infos with a matching name. For example, fortran*.** identifies all infos describing the FORTRAN compiler. All info names assumed to end with suffix of info if suffix omitted.

When segment is found, help looks inside for particular info which matches the entryname (or final entryname of path), and prints this info.

For subroutine infos selected by subroutine_$entry, printing begins with description of the $entry entry point; when -entry_point is given, printing begins with subroutine_$subroutine_ entry point description; when neither is given, printing begins with general description of the subroutine. Star convention not allowed when subroutine_$entry or -entry_point given.

Grouping of paragraphs:
The -minlines and -maxlines control arguments allow the user to control how much information help prints before asking the user if he wants to see more. help prints units of information called paragraphs. A paragraph is a group of lines preceded and followed by two blank lines.

The -minlines I control gives the length in lines of the smallest paragraph which help will treat as a distinct unit. Paragraphs shorter than I lines are often printed as part of the preceding paragraph.

The -maxlines J control limits the grouping of short paragraphs (those shorter than I lines long) so that no more than J lines of information are printed before asking if the user wants more help.

For example, consider info divided into paragraphs as follows--
```
  Paragraph 1 (8 lines)
   (2 blank lines)
  Paragraph 2 (3 lines)
   (2 blank lines)
  Paragraph 3 (4 lines)
```

With -minlines 4 and -maxlines 15, help would treat paragraph 2 as a short paragraph which is printed with paragraph 1 (total lines = 13). However, paragraph 3 is 4 lines long, and would be treated as a distinct paragraph.

With -minlines 5 and -maxlines 10, help prints paragraph 1 separately, since grouping short paragraph 2 with paragraph 1 would print 13 lines, exceeding -maxlines. Paragraphs 2 and 3 would be grouped together (total lines = 9) because paragraph 3 is shorter than 5

lines.

Paragraphs which have been seen are not grouped with unseen paragraphs. Similarly, paragraphs at the end of one section of info are not grouped with those beginning another section. Paragraphs are not grouped when the -section or -search control arguments are used to find a particular starting paragraph. If the wrong paragraph were found by the search, grouping might compound the error by printing more of the wrong information. Grouping is suppressed when the section and search requests are used for similar reasons.

Info naming conventions:
Infos for Multics commands, active functions and subroutines are given the name of the program with a suffix of info. For example, the info describing the pl1 compiler command is called pl1.info.

Information about changes made to a command or active function from one release to the next are given the name of the particular system module with a suffix of changes.info. For example, changes to the fortran compiler are described in fortran.changes.info.

General information describing use or features of the system is included in infos whose names end with a suffix of gi.info (gi for general info). For example, acl_matching.gi.info describes how Access Control List entries are matched with User_ids in access control commands such as set_acl.

Finding information:
More than 500 infos are available. To find information about a particular area of the system, use the list_help command with a topic name identifying the area of the system. For example, to list info_names related to the FORTRAN compiler, you could type:

```
list_help fortran
```

The -header control argument of the help command can also be used to find particular information. For example, to get a list of all general information segments, type:

```
help *.gi -he
```

Info segment format:

Users can create info segments describing their own commands, exec_coms and application programs. Info segments must be formatted in a special way so that the help command can parse them into paragraphs. For information about this format, type:

```
help info_seg.gi
```

home_dir

07/23/80 home_dir, hd

Syntax: hd

Function: returns the pathname of the user's home directory (usually of the form >user_dir_dir>Project_id>Person_id).

Syntax as active function: [hd]

hour

12/30/80 hour

Syntax: hour dt

Function: returns the one- or two-digit number of an hour of the day, from 0 to 23.

Arguments:
dt
is a date_time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current hour is returned. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [hour dt]

how_many_users

02/27/76 how_many_users, hmu

Syntax: hmu args -control_args

Function: prints how many users are currently logged in.

Arguments:
args
can be:
Person_id prints a count of users with the name Person_id.
.Project_id prints a count of users with the project name Project_id.
Person_id.Project_id prints a count of users with the specified name and project.

Control arguments:
-long, -lg
prints additional items, including shutdown information and load information on absentee users.
-absentee, -as
prints load information on absentee users only.
-brief, -bf
suppresses the printing of the headers. Only used in conjunction with one of the optional_args.

Notes: If this command is invoked without any arguments, basic summary information is printed. When necessary, absentee counts are denoted by an asterisk (*).

hunt

08/01/80 hunt

Syntax: hunt name path -control_args

Function: searches a specified subtree of the hierarchy for all occurrences of a named segment that is either free standing or included in an archive file.

Arguments:

name
   is the name of a segment for which hunt is to search. The star convention is allowed.

path
   is the pathname of a directory to be interpreted as the root of the subtree in which to search for the specified segment(s). If path is not specified, the subtree rooted at the current working directory is searched.

Control arguments:

-all, -a
   reports on finding links and directories as well as segments.

-archive, -ac
   looks inside archives for components whose names match the name argument. This is the default.

-first
   stops searching as soon as the first occurrence of the specified segment is found.

-no_archive, -nac
   does not look inside archives and is therefore faster.

Notes: The hunt command displays the type of entry found (seg, dir, or link) followed by the entry itself. A total of the number of occurrences found is displayed at the end.

Syntax as active function: [hunt name path -control_args]

Notes as an active function: When invoked as an active function, hunt returns a string of pathnames separated by spaces. Archive components are returned as archive_path::component_name.

All arguments accepted by the hunt command are accepted by the active function.

immediate_messages

06/17/81 immediate_messages, im

Syntax: im destination -control_arg

Function: restores the immediate printing of messages sent to the user by the send_message command and the "You have mail." notification sent by the send_mail command.

Arguments:
destination
is of the form Person_id.Project_id to specify a mailbox. The default is the user's default mailbox. If destination contains < or >, it is assumed to be the pathname of a mailbox.

Control arguments:
-pathname path, -pn path
specifies a mailbox by pathname. The mbx suffix is assumed. This control argument and the destination argument are mutually exclusive.

Notes: This command "cancels" the defer_messages command.

For a description of the mailbox, refer to the accept_messages and print_mail commands.

initiate

03/18/80 initiate, in

Syntax: in path ref_names -control_args

Function: initiates segments.

Arguments:
path
   is the pathname of a segment or a link to a segment.
   The star convention is NOT allowed.
ref_names
   are optional reference names by which to initiate the segment.
   If no ref_names are specified, the segment is initiated by the entryname
   portion of path.

Control arguments:
-all, -a
   initiates the segment by all its names.
-brief, -bf
   does not print a message giving the segment number. (Default)
-chase
   used with -all on a link pathname, initiates the target segment by all the
   names on the target segment. (Default)
-force, -fc
   terminates each reference name first if it is already known.
-long, -lg
   prints a message giving the segment number assigned.
-no_chase
   used with -all on a link pathname, initiates the target segment by all the
   names on the link.
-no_force, -nfc
   prints an error message if a ref_name is already known. (Default)

Access required: nonnull.

io_call

05/22/81 io_call, io

Syntax: io opname switchname args

Function: performs operations on I/O switches and prints or returns the results.

Arguments:
opname
   is a name given below under "List of operations".
switchname
   names I/O switch through which operation is performed (throughout the rest of this discussion, switchname is represented by SW).
args
   are additional arguments that some operations require or accept. Argument N is a buffer size, in characters. When invoked as active function, N can be omitted, and defaults to the size of the active function return string.

List of operations:
attach SW attach_description
   uses attach_description to attach SW.
detach_iocb SW, detach SW
   detachs SW.
open SW mode
   opens SW with given mode. (See "Modes" below.)
close SW
   closes SW.

get_line SW N -control_args
   reads and prints next line from SW; control_args can be: -segment, -nnl, -nl, -lines.
get_chars SW N -control_args
   reads and prints next N characters from SW; control_args can be: -segment, -nnl, -nl, -lines.
put_chars SW STR -control_args
   outputs STR to SW; control_args can be: -segment, -nnl, -nl, -lines. If STR is omitted, -segment must be given.

read_record SW N -control_args ,
read SW N -control_args
   reads and prints next record from SW; control_args can be: -segment, -nnl, -nl, -lines.
write_record SW STR -control_args ,
write SW STR -control_args
   writes STR to SW; control_args can be: -segment, -nnl, -nl, -lines. If STR is omitted, -segment must be given.
rewrite_record SW STR -control_args ,
rewrite SW STR -control_args

replaces current record in file to which SW is attached with STR; control_args can be: -segment, -nnl, -nl, -lines. If STR is omitted, -segment must be given.

delete_record SW, delete SW
deletes current record in file to which SW is attached.

position SW type
positions file to which SW is attached. type can be: bof; eof; forward J, fwd J, f J; reverse J, rev J, r J; I J . I and J are integers.

seek_key SW key
positions indexed file to which SW is attached to record with given key. If record not found, key becomes key for insertion of new record.

read_key SW
reads and prints key and record length of next record in indexed file to which SW is attached.

read_length SW
reads and prints length of next record in structured file to which SW is attached.

control SW order args
performs named order operation on SW; args depend upon the particular order and I/O module through with SW is attached.

modes SW STR -control_args
prints old modes associates with SW, and sets new modes given in STR; control_args can be: -brief.

move_attach SW SW2
moves attachment from SW to SW2. SW is left in detached state.

find_iocb SW
prints location of SW. Switch is created if not already existing.

look_iocb SW
prints location of SW. An error occurs if SW does not exist.

destroy_iocb SW
destroys SW.

print_iocb SW
prints all data from control block for SW.

attached SW
prints true if SW is attached.

opened SW
prints true if SW is opened.

closed SW
prints true if SW is closed.

detached SW
prints true if SW is detached.

attach_desc SW
prints attach description for SW.

open_desc SW
prints current opening mode for SW.

io_module SW
prints name of I/O module through which SW is attached.

valid_op SW operation
   prints true if operation is valid for SW, given its current attachment and opening mode.
test_mode SW mode
   prints true if mode appears in modes string of SW; prints false if ^mode appears. Prints error if mode does not appear or is not an on/off mode.
valid_mode SW mode
   prints true if mode or ^mode appears in modes string of SW; prints false if either does not appear.

Control arguments:
-segment path O L , -sm path O L
   gives pathname of segment into which data from input operations (get_line, get_chars, read_record) is stored, and from which data for output operations (put_chars, write_record, rewrite_record) is obtained. O is an offset within the segment, measured in characters unless -lines is also given. L is a length given for output operations, measured in characters unless -lines is also given.

-nnl
   deletes newline character from end of input data, and suppresses appending newline to end of output data. This is the default.
-nl
   adds newline character to end of input data before printing it, and appends newline character to end of output data if one not already present.
-lines
   causes O and L to be measured in lines, rather than in characters.

Modes:

stream_input, si	keyed_sequential_input, ksqi
stream_output, so	keyed_sequential_output, ksqo
stream_input_output, sio	keyed_sequential_update, ksqu
sequential_input, sqi	direct_input, di
sequential_output, sqo	direct_output, do
sequential_update, squ	direct_update, du
sequential_input_output, sqio	

Syntax as active function: [io_call operation SW args]

Arguments as active function:
get_line SW N -control_args
   returns data read as a quoted string; control_args can be: -no_quote, -nnl, -nl.
get_chars SW N -control_args
   returns data read as a quoted string; control_args can be: -no_quote, -nnl, -nl.
read_record SW N -control_args ,
read SW N -control_args
   returns data read as a quoted string; control_args can be: -no_quote, -nnl, -nl.

position SW type
   returns true if indicated position operation succeeds.
   type can be: bof; eof; forward J, fwd J, f J; reverse J,
   rev J, r J; I J . I and J are integers.
seek_key SW key
   returns true if key exists.

read_key SW -control_args
   returns key of next record as a quoted string; control_args can be:
   -no_quote.
read_length SW
   returns length of next record in a structured file.

control SW order args
   performs named order operation on SW, and returns the result.
   Result and args depend upon particular order given and the I/O
   module in use.
modes SW new_modes
   returns old modes, optionally sets new modes.

look_iocb SW
   returns true if SW exists.
attached SW
   returns true if SW is attached.
opened SW
   returns true if SW is opened.
closed SW
   returns true if SW is closed.
detached SW
   returns true if SW is detached.

attach_desc SW -control_args
   returns attach description for SW as a quoted string; control_args
   can be: -no_quote.
open_desc SW
   returns current opening mode for SW.
io_module SW
   returns name of I/O module through which SW is attached.

valid_op SW operation
   returns true if operation is valid for SW, given its current
   attachment and opening mode.
test_mode SW mode
   returns true if mode appears in modes string of SW; returns false
   if ^mode appears. An error occurs if mode does not appear or is
   not an on/off mode.
valid_mode SW mode
   returns true if mode or ^mode appears in modes string of SW;
   returns false if either does not appear.

Control arguments as active function:
-no_quote, -nq
   do not enclose the returned data in quotes. Data containing

spaces is quoted by default.

laser

08/29/83 laser

Syntaxe: laser -control_args

Fonction: cette commande permet d'obtenir des renseignements pratiques tels que la liste des formats disponibles, la liste des fichiers ecrits sur une bande laser donnee pour un usager donne, etc ...

Arguments de controle:

-format FMT, -fmt FMT
   Cet argument permet d'obtenir des informations sur le format dont le nom est donne en parametre. Si on n'en precise pas, ces memes informations seront rendues pour tous les formats. Dans tous les cas, cette restitution n'est faite que pour les formats auxquels l'utilisateur a acces.

-forms
   permet d'avoir la liste des fonds de page (forms) auxquels l'utilisateur a acces.

-long, -lg
   permet d'avoir davantage de renseignements sur un format. Par defaut, la commande ne rend que le minimum essentiel: les page_length et line_length du format. Avec cet argument, on obtiendra en outre le nom des polices de caracteres utilisees et des observations particulieres. ATTENTION, NON EN SERVICE POUR L'INSTANT.

-no_header, -nhe
   evite l'edition de l'en-tete pour chaque rubrique a lister (format, forms ou programme).

-nbtape N, -nbt N
   specifie a la commande laser qu'on desire le listage du contenu de N bandes laser,

-papier,
   permet d'obtenir la liste des papiers disponibles sur l'imprimante,

-tape xxxxxx, -tp xxxxxx
   specifie a la commande laser qu'on desire commencer le listage du contenu de bande a partir de la bande 'xxxxxx'. Si on ne precise pas de nom, c'est la bande laser en cours d'ecriture qui est prise en compte. Une bande laser (ou spooler) a toujours un nom compose des trois lettres 'las' suivi de son numero d'ordre compris entre 001 et 040.

-user XXX
   specifie a la commande laser qu'on ne veut dans le listage des bandes, que les fichiers appartenant a la personne XXX. Si aucun nom n'est precise, le listage n'est effectue que pour les fichiers du demandeur. La convention "*" est acceptee.

NOTA: cette commande est experimentale et, de ce fait, peut etre amelioree dans la mesure ou une suggestion interessant une majorite d'utilisateur sera realisable.

last_message

06/17/81 last_message, lm

Syntax: lm address

Function: returns the text of the last message received from the send_message command.

Arguments:
address can be any of the following to specify a mailbox:
-pathname path, -pn path
where path is the pathname of a mailbox. The mbx suffix is assumed.
STR
specifies a mailbox pathname of STR that contains a > or <.
Person.Project
specifies the Person_id and Project_id of a user whose mailbox is indicated.

Notes: See also the descriptions of send_message, accept_message, last_message_sender, and last_message_time.

Syntax as active function: [lm address]

last_message_sender

06/17/81 last_message_sender, lms

Syntax: lms address

Function: returns the sender of the last message received (from the send_message command) in the form "Person_id.Project_id" (e.g., RSJones.Demo).

Arguments:
address can be any of the following to specify a mailbox:
-pathname path, -pn path
where path is the pathname of a mailbox. The mbx suffix is assumed.
STR
specifies a mailbox pathname of STR that contains a > or <.
Person.Project
specifies the Person_id and Project_id of a user whose mailbox is indicated.

Notes: The user is cautioned against using this active function when in polite mode. In polite mode, the system holds all messages until the user finishes typing a line (i.e., until the carriage is at the left margin). Therefore, it is possible that while the user is sending a message, the user's process can receive another message from a different user -- a message not yet seen. By using the last_message_sender active function in such a situation, the user can inadvertently attribute a message to the "wrong" person. See also the descriptions of send_message, accept_message, last_message, and last_message_time.

Syntax as active function: [lms address]

last_message_time

06/17/81 last_message_time, lmt

Syntax: lmt address

Function: returns the time that the last message (from the send_message command) was received.

Arguments:
address can be any of the following to specify a mailbox:
-pathname path, -pn path
where path is the pathname of a mailbox. The mbx suffix is assumed.
STR
specifies a mailbox pathname of STR that contains a > or <.
Person.Project
specifies the Person_id and Project_id of a user whose mailbox is indicated.

Notes: See also the descriptions of send_message, accept_message, last_message, and last_message_sender in this manual.

Syntax as active function: [lmt address]

line_length

02/02/79 line_length, ll

Syntax: ll maxlength

Function: sets the maximum terminal line length for output.

Arguments:
maxlength
must be greater than 4. Output lines longer than maxlength are folded.

link

02/25/76 link, lk

Syntax: lk path1A path2A ... path1N path2N

Function: creates a link pointing to a specified segment or directory. The word "link" also refers to interprocedure linkage. Type "help linking".

Arguments:

path1A
: pathname of the segment to which path2i is to point. The pathnames must be specified in pairs.

path2A
: specifies the pathname of the link to be created. If not given (in the final argument position of a command line only), a link to path1A is created in the working directory with the entryname portion of path1i as its entryname.

Access required: append, modify if name duplication occurs.

Notes: The star and equal conventions can be used.

list

09/17/80 list, ls

Syntax: ls entrynames -control_args

Function: prints information about the entries in a single directory.

Arguments:
entrynames
   are the names of entries to be listed. The star convention can be used. If no entrynames are given, all entries in the directory (of the default types or the types specified by control arguments) are listed. A pathname can be given instead of an entryname, causing the entries specified by its entryname portion to be listed, in the directory specified by its directory portion. It is an error to specify more than one directory to be listed in a single invocation of the list command.

Control arguments for directory:
-pathname path, -pn path
   list entries in the directory named path. Note the restriction described above under "Arguments".

Control arguments for entry type:
-segment, -sm
   list segments.
-multisegment_file, -msf
   list multisegment files.
-file, -f
   list files (segments and multisegment files).
-directory, -dr
   list directories.
-branch, -br
   list branches (segments, multisegment files, and directories).
-link, -lk
   list links.
-all, -a
   list all four entry types.

Control arguments for column: (see also "Notes on columns" below)
-date_time_entry_modified, -dtem
   print date-time-entry-modified in the modification-date column.
-date_time_contents_modified, -dtcm
   print date-time-contents-modified in the modification-date column.
-date_time_used, -dtu
   print date-time-used column.
-mode, -md
   print mode column.
-record, -rec

```
   print records used in size column.
-length, -ln
   print length computed from bit count in size column.
-name, -nm
   print names column.
-count, -ct
   print name-count column, giving number of names.
-link_path, -lp
   print link-path column.


Control arguments for totals/header line:
-total, -tt
   print only number of entries and sum of their sizes.
-no_header, -nhe
   omit all heading lines and blank lines.


Control arguments for multiple-name entry:
-primary, -pri
   print only primary names in names column.
-match
   print only names that match one of entryname arguments.


Control arguments for entry order:
-sort KEY, -sr KEY
   sort entries by specified key column (see "Notes on Sorting").
-reverse, -rv
   reverse order of listing (reverses either directory order, or
   order of sorting if sorting was specified).


Control arguments for entry exclusion:
-exclude entryname, -ex entryname
   excludes entries that match entryname; more than one instance of
   this argument can be given.
-first N, -ft N
   list only first N entries (after sorting, if it is specified) of
   each entry type being listed.
-from DATE, -fm DATE
   excludes entries having date/time (dtem, dtcm, dtu) before DATE
   (see "Notes on dates" below).
-to DATE
   excludes entries having date/time (dtem, dtcm, dtu) after DATE
   (see "Notes on dates" below).


Control arguments for output format:
-brief, -bf
   either overrides default columns (see "Notes on Defaults" below)
   or, if -tt given, prints totals information for all selected entry
   types on single line.
-short, -sh
   print link paths starting two spaces after their names.


Notes on columns:
The column printing order is -- modification date, dtu, mode, size,
names, name count, and (for links only) link pathname.  Modification
```

date can be either date-time-contents-modified or
date-time-entry-modified (dtm is accepted as dtem). Size can be
either records used or length computed from the bit count (default).

List of Sorting Keys: The KEY field in "-sort KEY" can be--
name, nm
   sort by primary name, in ASCII collating sequence.
record, rec
   sort by records, largest first.
length, ln
   sort by bit count length, largest first.
mode, md
   sort by mode; order: null, r or s, rw or sm, re, rew or sma.
date_time_entry_modified, dtem
   sort by date-time-entry-modified, most recent first.
date_time_contents_modified, dtcm
   sort by date-time-contents-modified, most recent first.
count, ct
   sort by name count, highest first.

Links can only be sorted by: dtem, dtcm, nm, or ct. When
   sorting by other columns, links are listed in the order in which
   they are found in the directory. See also Defaults.

Notes on Dates:
The -from and -to control arguments compare DATE and date. The DATE
string must be acceptable to the convert_date_to_binary_ subroutine.
The date value is date-time-entry-modified (or
date-time-contents-modified, if it is being printed or sorted on) in
all cases except when date-time-used is the only date being printed
or sorted on.

Defaults: Invoking list without any arguments is the same as typing--
  list -pn [wd] -file -mode -length -name

If the sort column, COL, is omitted after -sort, the default sorting
column is: modification-date, if it is being printed; otherwise
date-time-used, if it is being printed; otherwise names.

Notes: Use of the -name, -mode, -record, -length, or -brief control
arguments overrides the default columns so that only the names column
and explicitly selected columns are printed.

Only one of the two modification dates, and only one of the two size
figures can be used at any one time. Any combination of arguments
that specifies both items from either pair (e.g., printing dtcm but
sorting on dtem) is an error.

list_abs_requests

10/01/80 list_abs_requests, lar

Syntax: lar path -control_args

Function: lists requests in the absentee queues.

Arguments:
path
is the pathname of a request to be listed. The star convention is allowed. Only requests matching this pathname are selected. If the path argument is not specified, all pathnames are selected. Also see the -entry control argument below.

Control arguments:
-absolute_pathname, -absp
prints the full pathname of each selected request, rather than just the entryname.
-admin User_id , -am User_id
selects the requests of all users, or of the user specified by User_id. If the -admin control argument is not specified, only the user's own requests are selected. See "Notes" below.
-all, -a
searches the foreground and all priority queues and prints the totals for each non-empty queue whether or not any requests are selected from it. If the -all control argument is not specified, nothing is printed for queues from which no requests are selected. This control argument is incompatible with the -queue control argument.

-brief, -bf
prevents the printing of the state and the comment of the request. If the -brief control argument is not specified, these items are printed. This control argument is incompatible with the -long and -total control arguments.
-deferred_indefinitely, -dfi
selects only requests that are deferred indefinitely. Such requests are not run until the operator releases them.
-entry STR, -et STR
selects only requests whose entrynames match STR. The star convention is allowed. Directory portions of request pathnames are ignored when selecting requests. This control argument is incompatible with the path argument.

-foreground, -fg
searches only the foreground queue, and prints the totals for this queue, whether or not any requests are selected from it. Also, see the -queue control argument.
-id ID
selects only requests whose identifier matches the specified ID.

-immediate, -im
selects only requests that can be run immediately upon reaching the heads of their respective queues. This does not include requests deferred indefinitely, requests deferred until a specific time, or requests that have reached the head of the queue and have been deferred by the system because their CPU time limits are higher than the maximum for the current shift. It does include requests deferred because of load control or resource unavailability, because those conditions could change at any time. Also, see the -position control argument.

-long, -lg
prints all of the information pertaining to an absentee request including the long request identifier and the full pathname. If this control argument is omitted, only the short request identifier, entryname, state and comment, if present, are printed. The -long, -brief, and -total control arguments are incompatible.
-long_id, -lgid
prints the long form of the request identifier. If this or the -long control argument is not specified, the short form of the request identifier is printed.
-pathname, -pn
prints the full pathname of each selected request, rather than just the entryname, just as -absolute_pathname does.

-position, -psn
prints the position within its queue of each selected request. When used with the -total control argument, it prints a list of all the positions of the selected requests. When used with the -immediate control argument, it considers only immediate requests when computing positions. See "Notes" below.
-queue N, -q N
searches only queue N, and prints the totals for that queue, whether or not any requests are selected from it. If the -queue control argument is not specified, all queues are searched but nothing is printed for queues from which no requests are selected. For convenience in writing exec_coms and abbreviations, the word "foreground" or "fg" following the -queue control argument performs the same function as the -foreground control argument. This control argument is incompatible with the -all control argument.

-resource STR , -rsc STR
selects only requests having a resource requirement. If STR is specified, only requests whose resource descriptions contain that string are selected. This control argument also causes the resource descriptions of the selected requests to be printed, even when the -long control argument is not specified. Type "help reserve_resource" for a description of the syntax of STR.
-sender STR
specifies that only requests from sender STR should be listed. One or more request identifiers must also be specified. In most cases, the sender is an RJE station identifier.

-total, -tt
prints only the total number of selected requests and the total number of requests in the queue plus a list of positions if the -position control argument is also specified. If the queue is

empty, it is not listed. This control argument is incompatible with the -long and -brief control arguments.

-user User_id
selects only requests entered by the specified user. See "Notes" below.

Access required: The user must have o access to the queue(s) to invoke lar. The user must have r extended access to the queue(s), in order to use the -admin, -position, or -user control arguments, since it is necessary to read all requests in the queue(s) in order to select those entered by a specified user or to compute the positions of the selected requests.

Notes: All queues are searched for the user's requests; the request identification, entryname, state, and comment, if present, of each request is printed. If no arguments are specified, only the user's own requests are selected for listing. Nothing is printed for queues from which no requests are selected.

When a user name is specified, with either the -admin or -user control arguments, then proxy requests are selected if either the user who entered the request, or the proxy user on whose behalf it was entered, matches the specified user name.

The entry name specified after the -entry control argument, the entry portion of the pathname argument, and the RJE station name specified after the -sender control argument, may each be starnames.

The User_id arguments specified after the -admin or -user may have any of the following forms:

```
Person_id.Project_id   matches that user only
Person_id.*        matches that person on any project
Person_id        same as Person_id.*
*.Project_id        matches any user on that project
.Project_id        same as *.Project_id
*.*         same as -admin with no User_id following it
```

list_accessible

07/17/81 list_accessible, lac

Syntax: lac path User_id -control_args

Function: scans a directory and lists segments, multisegments, files, and directories with a specified access for a specified User_id.

Arguments:

path
is the pathname of the directory to be scanned. If path is omitted or -wd is specified, the working directory is scanned.

User_id
is an access name. It can have null components. The star convention for access names is allowed. See the description of set_acl in this manual. If User_id is omitted, the User_id of the calling process with a star tag is assumed.

Control arguments: If no control arguments are specified, all the segments and directories to which the named user(s) has nonnull access are listed.

-dir_mode STR
lists directories to which the named user(s) has any of the modes specified in STR, where STR can be any or all of the letters sma.

-seg_mode STR
lists segments to which the named user(s) has any of the modes specified in STR, where STR can be any or all of the letters rew.

Access required: The user must have status (s) permission on the directory.

Notes: If there can be more than one User_id (i.e., the specified User_id has null components), the modes for each matched User_id and the matched User_id are listed on a per entry basis.

list_acl

08/30/79 list_acl, la

Syntax: la path User_ids -control_args

Function: lists the access control lists (ACLs) of segments, multisegment files, and directories.

Arguments:
path
   is the pathname of a segment, multisegment file, or directory. The default is -wd, or -working_dir. If omitted, no User_ids can be specified. The star convention can be used.
User_ids
   are access control names that must be of the form Person_id.Project_id.tag. If User_id is omitted, the entire ACL is listed.

Control arguments:
-ring_brackets, -rb
   lists the ring brackets.
-brief, -bf
   suppresses the message "User name not on ACL of path."
-directory, -dr
   lists the entire ACL of directories only. The default is segments, multisegment files, and directories. (See Notes.)
-segment, -sm
   lists the ACL of segments and multisegment files only.

Notes: If the list_acl command is invoked with no arguments, it lists the entire ACL of the working directory.
  The -directory and -segment control arguments are used to resolve an ambiguous choice that may occur when path is a star name.
  Type "help acl_matching" for an explanation of the User_id matching strategy.

Syntax as an active function:
 [la path User_ids]

Notes on use as an active function:
returns the matching modes and access names separated by spaces, for example: "r One.B.* rw Two.B.a". The -brief control argument is assumed.

list_daemon_requests

02/02/79 list_daemon_requests, ldr

Syntax: ldr path -control_args

Function: lists requests in the I/O daemon queue.

Arguments:
path
is the relative pathname of one or more requests to be listed. The star convention is allowed. If this argument is not specified, all requests are listed. See also the -entry control argument.

Control arguments:
-absolute_pathname, -absp
prints the full pathname.
-admin User_id , -am User_id
selects requests of all users or of the specified user. Default is to list the user's own requests. Required r extended access to the queue(s), to read other users' requests.
-all, -a
searches all queues.
-brief, -bf
prevents printing of comment and request state in normal (not -long) mode.
-entry STR, -et STR
selects only requests whose entry names match STR. The star convention is allowed. Directory portions of request pathnames are not used for selecting requests. Incompatible with the path argument.
-id ID
selects only requests whose request_ids match ID. Type "help request_ids".
-immediate, -im
selects only I/O requests that are not deferred. With -position, ignores deferred requests when computing position.
-long, -lg
prints all information about each selected request, including long request_id and full pathname. Default is to print short request_id and entryname.
-long_id, -lgid
prints the long request_id.
-position, -psn
prints queue positions of each selected request. With -total, prints a list of queue positions. Requires r extended access to the queue(s), to read other users' requests.
-queue N, -q N
searches queue N. The default queue is generally 3 for I/O daemon requests, but can vary with request type.
-request_type STR, -rqt STR
searches the I/O daemon queues belonging to the specified request type. See "Notes".
-total, -tt
prints only the total number of selected requests and the total number in the queue. Incompatible with -long and -brief control arguments.

-user User_id
   selects only requests of the specified user. Requires r extended access to the queue(s).

Notes: Only request types belonging to the generic types "printer" or "punch" can be specified by the -request_type control argument when the -long argument is given. A list of these request types can be obtained by invoking the print_request_types command.

list_help

06/24/80 list_help, lh

Syntax: lh topics -control_args

Function: displays the names of all info segments pertaining to a given topic. Topics are specified by arguments to the list_help command. An info segment is considered to pertain to a given topic if the topic name appears in (i.e., is a substring of) the info segment name. The active function returns the selected names separated by spaces.

Arguments:
topics
   are strings to be searched for in info seg names.

Control arguments:
-absolute_pathname, -absp
   prints or returns full pathnames of info segs rather than entrynames.
-brief, -bf
   does not display the alternate names on the info segments. The default is to display them.
-all, -a
   displays the names of all info segments. The default is to display the names of only those info segments whose names match the topics specified.
-pathname path, -pn path
   specifies the pathname of a directory to search for applicable segments. The default is to search the directories in the info_segments search list. Multiple -pathname control arguments are allowed. See "Notes on Search List" below.

Syntax as active function: [lh topics -control_args]

Notes on search list: The list_help command uses the "info_segments" search list that has the synonyms "info_segs" and "info". The default "info_segments" search list is:

   >doc>iml_info
   >doc>info

These directories contain info segments provided by the site and those supplied with the system. Type "print_search_paths info_segments" to see what the current "info_segments" search list is. For more information about search lists, see the search facility commands, and in particular, the add_search_paths command description in Commands and Active Functions, AG92.

list_iacl_dir

08/30/79 list_iacl_dir, lid

Syntax: lid path User_ids -control_args

Function: lists entries on a directory initial access control list (initial ACL).

Arguments:
path
   pathname of a directory; if it is -wd, -working_dir, or omitted, the working directory is assumed. If it is omitted, no User_ids can be specified. The star convention can be used.
User_ids
   access control names that must be of the form Person_id.Project_id.tag. If no User_id is specified, the whole initial ACL is listed.

Control arguments:
-ring N, -rg N
   ring number (default is current ring).
-brief, -bf
   suppresses the message "User name not on ACL of path."

Notes: If this command is given without any arguments, the entire initial ACL for the current ring for the working directory is listed.
  Type "help acl_matching" for an explanation of the User_id matching strategy.

Syntax as an active function:
 [lid path User_ids -ring N]

Notes on use as active function:
returns the matching modes and access names separated by spaces, for example: "s One.B.* sma Two.B.a". The -brief control argument is assumed.

list_iacl_seg

08/30/79 list_iacl_seg, lis

Syntax: lis path User_ids -control_args

Function: lists entries on a segment initial access control list (initial ACL) in a directory.

Arguments:
path
   pathname of a directory; if it is -wd, -working_dir, or omitted, the working directory is assumed. If it is omitted, no User_ids can be specified. The star convention can be used.
User_ids
   access control names that must be of the form Person_id.Project_id.tag. If no User_id is specified, the whole initial ACL is listed.

Control arguments:
-ring N, -rg N
   ring number (default is current ring).
-brief, -bf
   suppresses the message "User name not on ACL of path."

Notes: If this command is given without any arguments, the entire segment initial ACL for the current ring for the working directory is listed.
  Type "help acl_matching" for an explanation of the User_id matching strategy.

Syntax as an active function:
 [lis path User_ids -ring N]

Notes on use as an active function:
returns the matching modes and access names separated by spaces, for example: "r One.B.* rw Two.B.a". The -brief control argument is assumed.

list_not_accessible

01/10/77 list_not_accessible, lnac

Syntax: lnac path User_id -control_args

Function: scans a directory and lists segments and directories that do not have a specified access relation to a named user.

Arguments:
path
   the directory to check (default: working dir)
User_id
   a standard access control name (default: User_id of calling process)

Control arguments:
-dir_mode STR
   lists directories to which the user does not have STR mode; STR can be any or all of the letters sma.
-seg_mode STR
   lists segments to which the user does not have STR mode; STR can be any or all of the letters rew.

Notes: If no control arguments are given, the command lists all segments and directories to which the user has null access.

list_ref_names

09/15/80 list_ref_names, lrn

Syntax: lrn paths -control_args

Function: lists the reference names associated with a specified segment; it accepts both segment numbers and pathnames as segment specifications.

Arguments:

paths
can be segment numbers or pathnames of segments known to the user's process. If path is a segment number, the pathname and reference names of the segment are printed. If path is a pathname, the segment number (in octal) and the reference names of the segment are printed. If a pathname looks like a control argument (i.e., if it is preceded by a minus sign) or a number, then path should be preceded by -name or -nm.

Control arguments:

-all, -a
prints the pathnames and reference names of all known segments, as well as the reference names of ring 0 segments. The -all control argument is equivalent to -from 0.

-brief, -bf
suppresses printing of the reference names for the entire execution of the command.

-from N, -fm N
allows the user to specify a range of segment numbers. This control argument can be used with the -to control argument. The pathnames and reference names of the segments in this range are printed. If -to is not specified, the highest used segment number is assumed.

-to
allows the user to specify a range of segment numbers. This control argument can be used with the -from control argument. The pathnames and reference names of the segments in this range are printed. If -from is not specified, the segment number of the first segment not in ring 0 is assumed, unless -all is used.

Notes: All of the above arguments (segment specifiers and control arguments) can be mixed. For example, in the command line:

```
! lrn 156 -from 230 path_one
```

the pathname and reference names of segment 156 and of all segments from 230 on are printed. The segment number (in octal) and the reference names of path_one are printed.

In the default condition, when called with no arguments, list_ref_names prints information on all segments that are not in ring 0.

When a pathname is specified, the segment number by which it is known is printed. When a segment number is specified, lrn also prints the pathname of the segment.

list_resources

01/12/81 list_resources, lr

Syntax: lr -control_args

Function: lists groups of resources managed by the Resource Control Package (RCP), selected according to criteria specified by the user.

Control arguments:

-acquisitions, -acq
   lists resources acquired by the user specified by the -user control argument. If this control argument is used, -type must also be specified.

-assignments, -asm
   lists resource assignments. This cannot be used with the active function.

-awaiting_clear
   lists those resources that are awaiting manual clearing.

-device STR, -dv STR
   lists device resources with the name STR. No other resources are listed. This cannot be used with the active function.

-logical_volume, -lv
   lists logical volumes that are currently attached. This cannot be used with the active function.

-long, -lg
   prints all the information known about each resource listed. If this control argument is not supplied, only the name is printed for each resource listed. This cannot be used with the active function. -lg has no effect if the -acq control argument has been specified.

-mounts, -mts
   lists resources currently mounted by the process. This cannot be used with the active function.

-reservations, -resv
   lists only device and volume reservations. This cannot be used with the active function.

-type STR, -tp STR
   lists resources of the type STR. See list_resource_types for information on obtaining the names of resource types.

-user User_id
   selects a particular user or group of users for whom resource information is to be printed. This control argument can be used only in conjunction with -acquisitions. The User_id can be any of the following forms--

   Person.Project
      specifies a particular Person_id and Project_id combination.
   *.Project
      specifies all users on a specified project.
   .

specifies all users (i.e., all acquired resources are listed).

free
specifies all resources in the free pool.

system
specifies all resources in the system pool.

**
specifies all users plus the free and system pools (i.e., all registered resources will be listed).

If this control argument is not specified, the User_id of the user invoking list_resources is assumed. See "Notes on Access Restrictions" below.

Notes on access restrictions: Access to rcp_admin_ is required to obtain information on other users. Read access to the PDT (Project Definition Table) of a specified project is required to obtain information for that project.

Notes: If this command is invoked without any arguments, all resources assigned and devices attached to the calling process are listed.

Syntax as active function: [lr -control_args]

list_retrieval_requests

02/02/79 list_retrieval_requests, lrr

Syntax: lrr path -control_args

Function: lists requests in the retrieval queue.

Arguments:
path
is the relative pathnme of one or more requests to be listed. The star convention is allowed. If this argument is not specified, all requests are listed. See also the -entry control argument.

Control arguments:
-absolute_pathname, -absp
prints the full pathname.
-admin User_id , -am User_id
selects requests of all users or of the specified user. Default is to list the user's own requests. Required r extended access to the queue(s), to read other users' requests.
-all, -a
searches all queues.
-brief, -bf
prevents printing of comment and request state in normal (not -long) mode.
-entry STR, -et STR
selects only requests whose entry names match STR. The star convention is allowed. Directory portions of request pathnames are not used for selecting requests. Incompatible with the path argument.
-id ID
selects only requests whose request_ids match ID. Type "help request_ids".
-long, -lg
prints all information about each selected request, including long request_id and full pathname. Default is to print short request_id and entryname.
-long_id, -lgid
prints the long request_id.
-position, -psn
prints queue positions of each selected request. With -total, prints a list of queue positions. Requires r extended access to the queue(s), to read other users' requests.
-queue N, -q N
searches queue N. The default queue is 3.
-total, -tt
prints only the total number of selected requests and the total number in the queue. Incompatible with -long and -brief control arguments.
-user User_id
selects only requests of the specified user. Requires r extended access to the queue(s).

locnet

09/12/83 locnet Utilisation du reseau local "LOCNET" du CNET/Paris.

INTRODUCTION:
Ces quelques lignes d'explication ont pour but de decrire les connexions possibles d'un terminal, relie au reseau local "LOCNET" du CNET/Issy, sur tout ordinateur relie a ce reseau, et de donner les principaux messages d'erreur dus, soit a un mauvais fonctionnement d'un element du reseau (ou de Transpac), soit a une saturation d'un de ces elements, ou encore a une mauvaise utilisation quelconque.

1) ACCES:
Mettre le terminal sous tension. Taper H (CR). Apparait alors le message: "LOCNET PARIS XX". Composer alors le numero d'acces au site desire.

Si la communication est etablie, apparait le message: "COM"

2) NUMEROTATION, DEPUIS LE RESEAU LOCAL, VERS LE CNET/ISSY:
(Le site destinataire est le DPS-8 MULTICS)

NUMERO	TYPE DE LIAISON	TYPE D'ACCES	VITESSES
59209055	via le DN 7103 (sous le "modele" ASCII)	X 25	110-4800
59209156	via le DN 7103 (qui demande le modele voulu)	X 25	"
59205201	via le DN 6678	EBVO	1200
59205302	"	"	"
59206204	"	"	"
59206306	"	"	"

En outre, on dispose des 2 mnemoniques : MA = acces EBVO via le DN 6678,
MB = " X 25 " le DN 7103.

3) NUMEROTATION, DEPUIS LE RESEAU LOCAL, VERS L'EXTERIEUR:

NUMERO TRANSPAC DU DESTINATAIRE	TYPE D'ACCES	VITESSES	SITE DESTINATAIRE
Exemples :			
12200013220	X 25	110-4800	IRIS 80 : CNET/LANNION
12200013440	X 25	110-4800	" : "
1380201326	X 25	110-4800	VAX : CNET/GRENOBLE

4) ACCES EXTERIEURS, DEPUIS TRANSPAC, VERS LE CNET/ISSY:
(Le site destinataire est le DPS-8 MULTICS)

NUMERO	TYPE DE LIAISON	TYPE D'ACCES	VITESSES
19202066855	via le DN 7103 (sous le "modele" ASCII)	X 25	110-4800
19202066856	via le DN 7103 (qui demande le modele voulu)	X 25	"
192020455	"	X 25	"
19202066801	via le DN 6678	EBVO	1200
19202066802	"	"	"
19202066804	"	"	"
19202066806	"	"	"

5) MESSAGES D'ERREURS LES PLUS COURANTS:

LIB DTE 008	Plus d'acces libre sur le site destinataire
LIB DTE 023	Lignes d'acces au calculateur hors service
LIB DTE 035	Erreur de mot de passe dans le numero
LIB NC 144 LIB NC 145 LIB DTE 004	PROBLEMES SUR LIAISONS D'ACCES
LIB DER XXX	Frontal en derangement
LIB NP XXX	Numero inconnu
RESET DTE 028	Pertes de caracteres (Par exemple, probleme de controle de flux sur un PAD)

6) INFORMATIONS COMPLEMENTAIRES:

Pour avoir des informations plus detaillees sur les messages de TRANSPAC, faire help reseau ; on trouvera des informations detaillees sur les profils des PAD, et des tableaux qui recapitulent les messages TRANSPAC, avec leur signification.

login

09/17/81 login, l

Syntax: l Person_id Project_id -control_args
or: l Person_id.Project_id -control_args

Function: used to gain access to the system. It is a request to the answering service to start the user identification procedure, and then either create a process for the user, or connect the terminal to an existing disconnected process belonging to the user. The login command line may be no more than 300 characters in length.

Arguments:

Person_id
is the user's registered personal identifier. This argument must be supplied. The personal identifier can be replaced by a registered "login alias" if the user has one.

Project_id
is the identification of the user's project. If this argument is not supplied, the default project associated with the Person_id is used. See the -change_default_project control argument below for changing the default project to the Project_id specified by this argument.

List of general control arguments: The following are permitted in any use of the login command:

-brief, -bf
suppresses messages associated with a successful login. If the standard process overseer is being used, the message of the day is not printed.

-change_default_auth, -cda
changes the user's registered default login authorization to the authorization specified by the -authorization control argument. If the authorization given by the user is valid, the default authorization is changed for subsequent logins, and the message "default authorization changed" is printed at the terminal. If the -cda control argument is given without the -auth argument, an error message is printed.

-change_default_project, -cdp
changes the user's default project to be the Project_id specified in this login request line. The default Project_id is changed for subsequent logins. If the -cdp control argument is specified without a Project_id argument, an error message is printed.

-change_password, -cpw
changes the user's password to a newly given password. The login request asks for the old password before it requests the new one. Passwords can be up to eight characters long and can not contain imbedded blanks.

-generate_password, -gpw
changes the user's password to a new password, generated for the

user by the system. The login request asks for the old password first. Then, a new password is generated and typed on the user's terminal. The user is asked to retype the new password, to verify having seen it.

-modes STR, -mode STR, -md STR
sets the I/O modes associated with the user's terminal to STR, where STR consists of modes acceptable to the tty_ I/O module (described in MPM Communications I/O). STR is usually a list of modes separated by commas and must not contain blanks.
-no_print_off, -npf
causes the system to overtype a string of characters to provide a black area for the user to type the password.
-no_warning, -nw
suppresses even urgent system warning and emergency messages from the operator, both at login and during the user's session. Use of this argument is recommended only for users who are using a remote computer to simulate a terminal, or are typing out long memoranda, when the process output should not be interrupted by even the most serious messages.

-print_off, -pf
suppresses overtyping for the password. The default for this control argument depends on the terminal type.
-terminal_type STR, -ttp STR
sets the user's terminal type to STR, where STR is any terminal type name defined in the standard terminal type table. This control argument overrides the default terminal type.

List of control arguments for process creation: The following arguments are to be used when requesting the creation of a new process.
-authorization STR, -auth STR
sets the authorization of the process to that specified by STR, where STR is a character string composed of level and category names for the desired authorization, separated by commas. STR cannot contain any embedded blank or tab characters. STR must represent an authorization that is less than or equal to the maximum authorization of Person_id on the project Project_id. If this control argument is omitted, the user's registered default login authorization is used.
-force
logs the user in if at all possible, provided the user has the guaranteed login attribute. Only system users who perform emergency repair functions have the necessary attribute.

-home_dir path, -hd path
sets the user's home directory to the path specified, if the user's project administrator allows this choice.
-no_save_on_disconnect, -nosave
causes the user's process to be logged out instead of being saved, if it becomes disconnected from its login terminal. This argument is used to override a default of -save_on_disconnect, if that default has been set by the user's project administrator.
-no_preempt, -np
refuses to log the user in if this can only be done by preempting

some other user in this user's load control group.

-no_start_up, -ns
instructs the standard process overseer not to execute the user's start_up.ec segment, if one exists, and if the project administrator allows this choice.

-outer_module path, -om path
attaches the user's terminal via the outer module named path rather than the user's registered outer module, if the user is allowed this choice.

-process_overseer path, -po path
sets the user's process overseer to the procedure given by the path specified, if the user's project administrator allows this choice. If path ends in the characters ",direct", the specified procedure is called directly during process initialization rather than by the standard procedure provided by the system. This means that the program specified by path must perform the tasks that would have been performed by the standard procedure.

-ring N, -rg N
sets the user's initial ring to be ring N, if this ring number is greater than or equal to the user's registered initial ring and less than the user's registered maximum ring.

-save_on_disconnect, -save
saves the user's process if it becomes disconnected from its login terminal because of a communications line hangup or FNP crash. Permission to use the process-saving facility, and the setting of whether or not the facility is enabled by default, are both under the control of the user's project administrator. See the description of the -no_save_on_disconnect control argument, and the descriptions of the save_on_disconnect and no_save_on_disconnect commands.

-subsystem path, -ss path
creates the user's process using the prelinked subsystem in the directory specified by path. The permission to specify a process overseer also governs the use of the -subsystem argument. To override a default subsystem specified by the project administrator, type -ss "".

List of control arguments for disconnected processes: The following are used to specify the disposition of disconnected processes.

-connect N
connects the terminal to the user's disconnected process. If more than one such process exists, the process number N must be specified.

-new_proc N
destroys the user's disconnected process and creates a new one. If more than one such process exists, the process number N must be specified.

-destroy N
destroys the user's disconnected process and logs out. If more than one such process exists, the process number N must be specified.

-create
creates a new process without destroying any disconnected processes. This is permitted only for users who are allowed to have multiple

interactive processes.

-list
lists the user's disconnected process, showing the process number, the time of the original login, and the ID of the channel and terminal that were last connected to the process.

Notes on disconnected processes: If a user's project administrator allows it, a user's process can be preserved when it becomes disconnected from its terminal because of a phone hangup or FNP crash. The user can call back any time before the (installation-defined) maximum inactive time and ask to be connected to this disconnected process. This feature is controlled by the -save_on_disconnect and -no_save_on_disconnect control arguments; the default is set by the user's project administrator.

Some users are permitted by their project administrators to have several interactive processes simultaneously. These users can have more than one disconnected process. Multiple disconnected processes are numbered consecutively starting with 1, in the order of their login times. These process numbers must be used as arguments when referring to one of a set of multiple disconnected processes. The number and login time of each is printed by the -list argument or the list preaccess request. The user can, however, anticipate the process numbering and use a number in an argument to the login command. The time listed and sorted on is the time of the original login from which the process is descended; this time is not affected by new_proc or reconnection.

List of actions:
A user with disconnected processes who does not specify, on the login line, the action to be taken with respect to the disconnected processes, is told of the existence of the disconnected processes and given a choice of the following actions:

list
to list the user's disconnected processes;

create
to create an additional process;

connect
to connect the terminal to a disconnected process;

new_proc
to destroy a disconnected process, create a new one with the same attributes, and connect the terminal to it.

destroy
to destroy a disconnected process and log out.

logout
to logout without affecting any process.

Notes:
The connect, new_proc, and destroy requests take an optional process number as an argument. The logout and destroy requests take an optional -hold (-hd) control argument, which prevents the breaking of the connection between the terminal and the answering service. The new_proc and destroy requests take an optional -immediate control argument, that causes the disconnected process to be destroyed immediately, without being sent a trm_ signal; this is useful for terminating a malfunctioning process. The help request, when issued

from a logged in but disconnected terminal, explains these options rather than explaining how to log in.

logout

04/24/81 logout

Syntax: logout -control_args

Function: terminates a user session and ends communication with the Multics system. It signals the finish condition for the process; and, after the default on unit for the finish condition returns, it closes all open files and destroys the process.

When used as a preaccess command (from a terminal not connected to a process), it terminates the user session without destroying any process. (See Disconnected Processes under the login request.)

Control arguments:

-hold, -hd
: the user's session is terminated. However, communication with the Multics system is not terminated, and a user can immediately log in without redialing.

-brief, -bf
: no logout message is printed, and if the -hold control argument has been specified, no login message is printed either.

long_date

12/30/80 long_date

Syntax: long_date dt

Function: returns a month name, a day number, and a year as a single string in the form "month day, year" (e.g., November 2, 1976).

Arguments:
dt
is any string acceptable to convert_date_to_binary_. The default is the current date. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [long_date dt]

long_message_format

05/26/77 long_message_format, lmf

Syntax: lmf -pn mbx_path

Function: causes messages from send_message to be printed in long format. A header giving the sender's name precedes every line received by the user or the owner of the specified mailbox.

mail

02/02/79 mail, ml

Syntax: (sending) ml path Person1.Project1 ... PersonN.ProjectN
-control_args
or: ml path -pn PATH -control_args

(printing) ml Person1.Project1 -control_args
or: ml path -control_args

Function: sends a message to another user or prints messages in a mailbox. If the recipient is using accept_messages, that user receives an immediate notification of the form "You have mail."

Arguments:
path
(sending) pathname of segment to be mailed, or "*" for terminal input. Exits from terminal input with a line consisting of ".".
(printing) pathname of mailbox; the mbx suffix need not be given.

PersonN name of a person to whom mail is to be sent. ProjectN project on which PersonN is registered.

Control arguments:
-brief, -bf
(reading) print total only. Print nothing if mailbox empty.
-header, -he
(reading) print header of messages only, not text.
-match Pers.Proj
(reading) print messages from Pers.Proj only (stars ok).
-exclude Pers.Proj, -ex Pers.Proj
(reading) exclude Pers.Proj messages.
-acknowledge, -ack
(sending) cause recipient to send message when mail read.
-pathname PATH, -pn PATH
send to or read a mailbox specified by pathname.
-no_notify, -nnt
(sending) suppress the "You have mail." notification.

Access required:
Access on a newly created mailbox is automatically set to adrosw for the user who created it, asw for *.SysDaemon.*, and aow for *.*.*. For more information on extended access, type "help ext_access".

Creating mailbox:
A default mailbox is created automatically the first time a user types "mail"; the default mailbox is:
>user_dir_dir>Project_id>Person_id>Person_id.mbx

memo

08/31/81 memo

Syntax: memo -memo_options memo_text
or: memo -action_arg -memo_options -selection_args

Syntax as active function: [memo memo_text]
or: [memo -list -totals]

Function: maintains a user-created reminder list in a memo segment, which is normally Person_ID.memo in the user's home directory.

Arguments:
memo_text
is the text of the memo being set. It may not be longer than 132 characters. It can be specified in one of the following forms:
STR
the first string that does not begin with a hyphen is taken as the beginning of the memo text. It and all succeeding strings form the memo text. No further arguments are accepted.
-memo STRs
treats all succeeding STRs as part of the memo text, whether or not they begin with hyphens.

List of memo_options: These control arguments are used to control various options of the memo being set, or to select memos being otherwise processed.
-alarm, -al
specifies that the memo is to be an alarm. It will be printed on the terminal, or executed if set with -call, when its timer goes off, if timers are enabled, rather than when memos are explicitly processed. An alarm memo is deleted immediately after it reaches maturity, unless it was set with -retain.
-call
passes the memo text to the command processor as a command line when the memo matures, rather than printing it.
-date DT, -dt DT
identifies a date (DT) for the memo to mature in a form suitable for input to the convert_date_to_binary_ subroutine. The DT is truncated to midnight preceding the date in which DT falls.

-expires DT, -exp DT
identifies a time (DT) at which the memo is to expire; this is treated as a delta from the maturity time (which it must be greater than) so that repeating memos with expiration times will work properly. When used as a selection_arg, all expiring memos are selected, regardless of the expiration dates.
-invisible, -iv
specifies that the memo is never to be mature and will never be printed during a normal memo print.

-no_retain, -nret
specifies that the memo will only be processed once, and then will be automatically deleted. This is the default for alarm memos.

-repeat DT, -rpt DT
identifies the interval at which the memo is to repeat where DT must be greater than or equal to 1 minute. The repeat interval is applied repeatedly until the new maturity time is greater than the current time, and then the new memo is set. When used as a selection_arg, all repeating memos are selected, regardless of the repeat intervals given. See "Notes on repeating memos".
-repeat_when_processed, -rwp
specifies that the repeat time of a repeating memo will be applied from the time the memo is processed, rather than the maturity time. This is useful for memos which are only significant within a single process.

-retain, -ret
causes an alarm memo to be kept as an ordinary printing (or executing, if set with -call) memo after it matures, rather than being deleted automatically. This is the default for non-alarm memos.
-time DT, -tm DT
identifies a time (DT) for the memo to mature in a form suitable for input to the convert_date_to_binary_ subroutine.

List of action_args: These control arguments control various options of the memo command. Only one may be specified, and they may not be combined with memo setting.
-brief, -bf
suppresses printing of the message "No memos." if no memos are found.
-delete -force , -dl -fc
deletes all memos selected by the optional arguments. At least one memo must be explicitly specified. Memo will query the user before deleting non-mature memos. when given with -force, causes memos to be deleted even if they are not yet mature, without querying the user.

-list, -ls
prints text and control information of selected memos; no memos are executed. If no memos are explicitly selected, all memos are listed. If -totals is also specified, only the total number of selected memos is printed.
-off
suppresses all memo alarms, until the next memo command with no explicitly specified action. The -on and -off control arguments may be combined with other actions.
-on
enables memo alarms without printing or executing nonalarm memos.

-pathname PATH, -pn PATH, -pathname -default, -pn -dft
changes the default memo segment to PATH if specified with no other action. Otherwise, the memo segment specified by PATH is used for the execution only of the current memo command. If -pathname is

used along with -on or -off, the default memo segment IS changed, and alarms are turned on or off, as appropriate, for the new segment. The suffix ".memo" need not be supplied. When given as -pathname -default, the default memo segment is reset to Person.ID.memo in the user's home directory.

-postpone DT, -pp DT
reschedules the maturity of the selected memos to the time specified by DT, if DT is later than the current maturity time. At least one memo must be explicitly specified.

-print, -pr
prints text of all selected memos. No memos are executed. If no memos are explicitly selected, only mature memos are printed.

-process
causes all mature memos to be processed, and alarms to be turned on, if not otherwise specified. This is equivalent to not explicitly specifying an action.

-status, -st
prints information about the current default memo segment. If -status is specified, it must be the only argument.

-totals, -tt
can only be specified in combination with the -list control argument. When it is used, the total number of memos selected is printed, rather than listing each of the memos.

Notes: The -delete, -list, -print, -postpone and -process actions are mutually exclusive.

List of selection_args: These arguments are used to select memos to be listed, printed, deleted, or postponed. Some memo_options may also be used to specify types of memos to be selected (see "Notes" below). When more than one selection_args are specified, only those memos that match all of the selection criteria are selected.

memo_number
is either a positive decimal number specifying a single memo (for example 32), or two such numbers separated by a colon, specifying a range of memos (for example 12:16).

-from DT, -fm DT
selects all memos which mature on or after DT. -from may be combined with -to, each of which may only be specified once. This control argument is incompatible with -date and -time.

-match STRING
specifies a string against which memo texts are matched to select memos. STR may not be longer than 32 characters. Up to 40 strings may be specified; all memos which match at least one are selected.

-to DT
selects all memos which mature on or before DT. -to may be combined with -from. This control argument is incompatible with -date and -time.

Notes: No more than 5082 memos can be contained in a single memo segment. An individual memo may be no more than 132 characters long.

If no action is explicitly specified, and no memo is being set, all

mature memos are processed (printed or executed), and the alarm timer is turned on, enabling the processing of alarm memos.

The memo_options can also be used to specify types of memos to be selected; those which take a Date/Time interval (-repeat, -expires, but not -date or -time) will cause the selection of ALL repeating or expiring memos, as the time interval (which must be specified) is ignored.

Notes on default memo segment: The memo command operates on the default memo segment (unless -pathname is specified with one of the actions -delete, -list, -postpone, -print or -process). This default memo segment is also used when processing alarm timers, to find the memos which should be processed for the alarm. If the default memo segment has never been explicitly specified (by using -pathname without any other actions), it is the segment Person_ID.memo in the user's home directory.

The default memo segment is created if it does not already exist. If the default memo segment is changed, alarms are turned off for the old memo segment, and then turned on for the new one (if requested). Thus, only one memo segment may have alarms active at a time.

Notes on repeating memos: A repeating memo repeats by setting a new memo which is identical to the original one, and then turning off the repeat specification in the original memo.

An alarm memo which repeats will mature once, and then be automatically deleted, unless it was set with -retain, in which case it is turned into an ordinary, non-alarm memo, and lasts until it expires or is deleted.

Notes on expiring memos: Expired memos are deleted without being reprinted or executed. However, if they are repeating memos, they are repeated before being deleted. A sequence of repeating memos must be terminated manually (by deleting the current memo); the -expires control argument is not useful for this purpose.

Notes on active function: The memo active function can only be used to set and list memos. When a memo is set, the number assigned to the newly set memo is returned. When memos are listed, a string consisting of the memo numbers selected, separated by spaces, is returned; if -totals is specified, the total count is returned.

merge_ascii

05/12/81 merge_ascii, ma

Syntax: ma paths -control_args

Function: merges two or more related ASCII text segments.

Arguments:
paths
   are pathnames of segments to be merged as automatically as possible. The equal and :: conventions are allowed. Up to six segments can be merged, including those preceded by the -edit control_argument.

Control arguments:
-edit path
   merges the segment named path in a nonautomatic manner. Edit mode is entered each time a modification is found in the specified segment.
-minchars N
   specifies the minimum number of characters that must be identical for merge_ascii to assume blocks of text in different segments are identical. The default value of minchars is 25.
-minlines N
   specifies the minimum number of lines that must be identical for merge_ascii to assume blocks of text in different segments are identical. The default value of of minlines is 2.

-old_original path, -old_orig path
   identifies path as the pathname of a segment antecedent to the most recent common ancestor of the texts being merged and allows the automatic picking up of identical changes present in all the texts being merged.
-original path, -orig path
   identifies path as the pathname of a segment containing the original version of the text. The proper original is the most recent common ancestor of the texts being merged. Overlapping changes, even if identical, cause edit mode to be entered.
-output_file path, -of path
   put the merged output text in the segment named path. (no :: convention)

Notes: The merge_ascii program is typically used to merge texts that have been independently modified by several users. If an original version of the text is available, and if the user desires, merge_ascii performs the merge automatically, requiring user intervention only when overlapping modifications are detected. When user intervention is required, merge_ascii displays line-numbered blocks of text and then enters edit mode allowing the user to choose lines from any text or insert new lines.

When blocks of text are displayed, each line is preceded by a text identifier and a line number. The text identifier A is reserved for the original, whether supplied or not. The identifiers B-G are assigned to the texts being merged in the order in which their pathnames are encountered on the command line. The identifier M is used for the merged output, if printed while in edit mode.

The equal convention is allowed; equal processing is based on the first path argument in the command invocation.

Either the -original or -old_original (but not both) control argument may be used to enable automatic merging. If neither is supplied, edit mode is entered each time differences are found in the segments being merged. The -old_original control argument should be used judiciously, only if appropriate, and the user fully understands the relationships between the texts being merged.

List of edit requests: In the syntax of the edit requests, <text_id> is the lowercase letter corresponding to the text identifier used by merge_ascii; <line_no> is a line number in the text segment. Line numbers can be specified as "<" to address the first line or as ">" to specify the last line of a current block.

<text_id>k
   copy current block from specified text (e.g., bk copies current block from text B).

<text_id><line_no>k
   copy specified line from specified text (e.g., b5k copies line 5 from text B).

<text_id><line_no>,<line_no>k
   copy specified lines from specified text (e.g., b4,7k copies lines 4 through 7 from text B).

<text_id>p
   print current block from specified text (e.g., bp prints current block from text B).

<text_id><line_no>p
   print specified line from specified text (e.g., b6p prints line 6 from text B).

<text_id><line_no>,<line_no>p
   print specified line from specified text (e.g., b12,16p prints lines 12 through 16 from text B).

<text_id>d
   delete the current block in specified text (e.g., md deletes the current block in text M).

input
   enter input mode.

.
   return from input mode to edit mode.

go
   exit editor and continue comparison.

quit
   abort merge and return to command level. If this request is given during a merging procedure, all work is lost. Work is not saved unless merging is done from the beginning to the end of the

segments.

e
execute rest of line as a Multics command line.

x
display identifiers, current line numbers, and pathnames of each text.

help
print a list of the edit requests and a brief explanation of each one.

Notes on edit requests: In any invocation of edit mode the current block in each text is just the block of lines previously displayed. The current block in text M is initially empty, and is grown as the user selects or inputs lines.

The print (p) and copy (k) requests may address any lines in any text (A to M) known to merge_ascii. The delete (d) request can only be applied to the current block in text M, and has the effect of undoing all edit requests made since changes were last displayed.

Multiple edit requests, delimited by blanks, can be given on a single request line. However, the quit, go, input, and e requests must not be followed by other requests.

minute

12/30/80 minute

Syntax: minute dt

Function: returns the one- or two-digit number of a minute of the hour, from 0 to 59.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current time is used. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [minute dt]

modules

02/04/81 Commands, Active Functions, Subroutines

The help system provides individual info segments for each command, active function, and subroutine in the Multics system. These info segments are given the name of the particular system module (command, active function, or subroutine) with a suffix of "info". The resulting name is called an info_name. For example, the info segment describing the print command is named "print.info". But you need not type this suffix when using the help command; you can simply type

```
help print
```

Module Info_names:
If you are unsure of the name of a system module, you can get a list of possible names by using the list_help command with a word that describes what you are looking for. For example, if you want to know how to use the mail facility, you might type

```
list_help mail
```

For more information about the list_help command, type

```
help list_help
```

Subroutine entry points:
You can go directly to the description of a particular entry point in a subroutine by typing the name of the entry point with the help command. For example, by typing

```
help cu_$get_command_processor
```

you automatically bypass the 18 cu_ entry points described in alphabetical order before this one. If, on the other hand, you want to start printing at the beginning of the info seg for the command utility subroutine, you type

```
help cu_
```

month

12/30/80 month

Syntax: month dt

Function: returns the one- or two-digit number of a month of the year, from 1 to 12.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current month is used. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [month dt]

month_name

12/30/80 month_name

Syntax: month_name dt

Function: returns the full name of a month of the year.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current month is used. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [month_name dt]

move

10/28/80 move, mv

Syntax: move path1 path2...path1n path2n -control_arg

Function: causes a designated segment or multisegment file (along with its access control list (ACL) and all names) to be moved to a new position in the storage system hierarchy.

Arguments:
path1
   is the pathname of a segment or multisegment file to be moved. The star convention is allowed.
path2
   is the pathname to which path1 is to be moved in the target directory. The equal convention is allowed. If the last path2 segment is not specified, path1 is moved to the working directory and given the entryname path1.

Control arguments:
-acl
   moves the ACL. This is the default.
-all, -a
   moves multiple names and ACLs.
-brief, -bf
   suppresses the messages "Bit count inconsistent with current length..." and "Current length is not the same as records used...."
-chase
   moves the targets of links that match path1. (See "Notes".)
-long
   prints warning messages as necessary. This is the default.
-name, -nm
   moves multiple names. This is the default.

-no_acl
   does not move the ACL.
-no_chase
   does not move the targets of links that match path1. (See "Notes".)
-no_name, -nnm
   does not move the multiple names.

Access required: Read access is required for path1. Status and modify permission are required for the directory containing path1. Status, modify, and append permission are required for the target directory.

Notes:
The move command supports multiple moves on the same syntax line; path1 is moved to path2, path1n is moved to path2n, etc. If more

than one entry is to be moved to the working directory, path2 can be omitted, but path1. . .path1n must be contained in parentheses.

If an entry with the entryname path1 already exists in the target directory, the user will be asked whether the old (already existing) entry should be deleted. If the user answers "no", the move does not take place.

The default for chasing links depends on path1. If path1 is a star name, links are not chased by default, if path1 is not a star name, paths are chased.

If path1 is protected by the safety switch, the user is asked whether path1 is to be deleted from its old containing directory after it has been moved.

move_abs_request

10/08/80 move_abs_request, mar

Syntax: mar request_ids -control_args

Function: moves a request from one absentee queue to another. The request is always placed at the end of the target queue.

Arguments:
path
   is the full or relative pathname for the absentee input segment of requests to be moved. The star convention is allowed.
-entry STR, -et STR
   identifies requests to be moved by STR, the entryname portion of the absentee input segment pathname. The star convention is allowed.
-id ID
   identifies one or more requests to be moved by request identifier. This identifier may be used to further define any path or -entry identifier (see "Notes").

Control arguments:
-all, -a
   searches all queues except the target queue. This control argument is not compatible with -foreground and -queue control arguments.
-brief, -bf
   suppresses messages telling that a particular request_id was not found or which requests were moved when using star names or -all.
-foreground, -fg
   specifies that the foreground queue contains the requests to be moved. This control argument is not compatible with -all or -queue control arguments.
-queue N, -q N
   specifies that absentee queue N contains the requests to be moved. If not specified, the default queue searched is defined by the system administrator. This control argument is not compatible with -all or -foreground control arguments.

-sender STR
   specifies that only requests from sender STR should be moved. One or more request identifiers must be given.
-to_queue N, -tq N
   specifies which queue to move the request to. (Required)
-user User_id
   is a character string specifying the name of the submitter of the request to be moved, if not equal to the group id of the process. User_id can be of the form Person_id.Project_id, Person_id, or .Project_id. This control argument is primarily for the operator and administrators. Both r and d extended access to the queue are required. This control argument causes the command to use privileged message segment primitives that preserve the original identity of the submitter. The AIM ring_1 privilege is needed to

preserve the original AIM attributes. If ring_1 privilege is not present the AIM attributes of the user executing mar are used. The default is that only requests entered by the user executing mar are moved.

Access required: The user must have o extended access to the queue from which the request is being taken, and a access to the queue to which the request is being moved. The user must have r and d extended access to move a request owned by another user (see the description of the -user control argument above).

Notes: If any path of -entry STR request identifiers are given, only one -id ID request identifier will be accepted and it must match any requests selected by path or entryname.

Multiple -id ID identifiers can be specified in a single command invocation only if NO path or entry request identifiers are given.

When star names are not used and a single request_id matches more than one request in the queue(s) searched, none of the requests are moved. However, a message is printed telling how many matching requests there are.

If the request is already running, it is not moved and a message is printed to the user.

A complete description of User_id and AIM attributes can be found in the MPM Reference Guide.

move_daemon_request

10/08/80 move_daemon_request, mdr

Syntax: mdr request_identifiers -control_args

Function: moves a request from one I/O daemon queue to another. The move can be within the same request type or from one request type to another. The request is always placed at the end of the target queue.

Arguments:
request_identifiers can be chosen from the following:

path
identifies a request to be moved by the full or relative pathname of the input data segment. The star convention is allowed.

-entry STR, -et STR
identifies a request to be moved by STR, the entryname portion of the input data segment pathname. The star convention is allowed.

-id ID
identifies one or more requests to be moved by request identifier. This identifier may be used to further define any path or -entry identifier (see "Notes").

Control arguments:

-all, -a
searches all queues for the requests to be moved. This control argument is incompatible with the -queue control argument. The target queue is not searched by the -all control argument.

-brief, -bf
suppresses messages telling the user that a particular request identifier was not found or that requests were moved when using star names or the -all control argument.

-queue N, -q N
specifies that queue N for the specified request type contains the request to be moved, where N is an integer specifying the number for the queue. If this control argument is omitted, only the default queue for the request type is searched. This control argument is incompatible with the -all control argument.

-request_type STR, -rqt STR
specifies that the request moved is found in the queue(s) for the request type identified by STR. If this control argument is not specified, the default request type is "printer". Request types can be listed by the print_request_types command.

-to_queue N, -tq N
specifies which queue to move the request to. If not given, the default queue of the target request type is used.

-to_request_type STR, -to_rqt STR
specifies that the request should be moved to request type STR. If this control argument is not specified, the original request type is used. The target request types must be of the same generic type as the original request type.

-user User_id
specifies the name of the submitter of the requests to be moved. The default is to move only requests entered by the user executing the command. The User_id can be Person_id.Project_id, Person_id, or .Project_id. This control argument is primarily for the operator and administrators. Both r and d extended access to the queue are required. This control argument causes the command to use privileged message segment primitives that preserve the original identity of the submitter. If the process has access isolation mechanism (AIM) ring one privilege, the AIM attributes of the original submitter are preserved. Otherwise, the AIM attributes of the current process are used.

Access required: The user must have o extended access to the queue from which the request is being taken, and a access to the queue to which the request is being moved. The user must have r and d extended access to move a request owned by another user (see the description of the -user control argument above).

Notes: If any path or -entry STR request identifiers are given, only one -id ID request identifier will be accepted and it must match any requests selected by path or entryname.

Multiple -id ID identifiers can be specified in a single command invocation only if NO path or entry request identifiers are given.

When star names are not used and a single request identifier matches more than one request in the queue(s) searched, none of the requests are moved. However, a message is printed telling how many matching requests are found.

If the request is already running, it is not moved and a message is printed to the user.

See the MPM Reference Guide for a description of request identifiers.

move_dir

06/05/81 move_dir, mvd

Syntax: mvd source_dir target_dir entry_type_keys -control_args

Function: moves a directory and its subtree, including all of the associated attributes, to another point in the hierarchy.

Arguments:
source_dir
is the pathname of the directory to be moved. The star convention is allowed to match directory names. Matching names associated with other storage types are ignored. The source_dir cannot be contained in target_dir.
target_dir
is the new pathname for source_dir. If the entryname is different from one already on source_dir, it is added to the existing names. If target_dir is not specified, source_dir is moved to the working directory and given the same entryname. The equal convention is allowed.

Control arguments:
-brief, -bf
suppresses the printing of warning messages.
-force
continues execution when target_dir already exists, without asking the user. If the -force control argument is not specified, the user is queried.
-replace, -rp
deletes the contents of target_dir existing before the copying begins. If target_dir is non-existent or empty, this control argument has no effect. The default is to append the contents of the source directory to the target directory if it already exists.

List of entry_type_keys: These keys control what type of storage system entry is moved. The default is to move all entries. The keys are--
-branch, -br
-directory, -dr
-file, -f
-link, -lk
-multisegment_file, -msf
-non_null_link, -nnlk
-segment, -sm
If one or more entry_type_keys are specified, but not the -directory key, the subtree of source_dir will not be followed.

Access required: Status and modify permission are required for source_dir and all of the directories in its tree, and its containing directory. If target_dir does not exist, append permission is required

for its containing directory. If it does exist, modify and append permission for target_dir are required. This command does not force access.

The access control list associated with source_dir is moved to target_dir.

Notes: If target_dir is contained in source_dir, an appropriate error message is printed and control is returned to command level.

If name duplication occurs while appending the source_dir to the target_dir and the name duplication occurs between directories, the user is queried whether processing should continue. If the user answers yes, the contents of the directory are moved (appended) to target_dir, but none of the attributes of that directory are moved. If the answer is no, the directory and its subtree is skipped. If name duplication should occur between segments, the user is asked whether to delete the existing one in target_dir. (See the move command.)

Links are translated; that is, if there are references to a source directory in a link pathname, the link pathname is changed to refer to the target directory.

If part of the tree is not moved, problems with link translation may occur. If the target of the link in the source_dir tree was in the part of the tree not moved, there may be no corresponding entry in the target_dir tree. Hence, translation of the link (presumably originally non-null) will cause the link to become null.

See also the copy, move, and copy_dir commands.

move_quota

03/01/76 move_quota, mq

Syntax: mq path1 quota_change1 ... pathN quota_changeN

Function: moves storage quota between two directories, one immediately inferior to the other.

Arguments:

pathN
: pathname of a directory; -wd or -wdir can be used. The star convention CANNOT be used.

quota_changeN
: number of records to be moved between the containing directory quota and the pathN quota. The argument can be either positive or negative. If positive, quota is moved from the containing directory to pathN; if negative, the move is from pathN to the containing directory.

Access required: The user must have modify permission on both the directory specified by pathN and its containing directory.

new_proc

07/10/81 new_proc

Syntax: new_proc -control_arg

Function: destroys the user's current process and creates a new one, using the control arguments given initially with the login command, and the optional argument to the new_proc command itself. Just before the old process is destroyed, the "finish" condition is signalled. After the default on unit returns, all open files are closed. The search rules, I/O attachments, and working directory for the new process are as if the user had just logged in.

Control arguments:
-authorization STR, -auth STR
to create the new process at authorization STR, where STR is any authorization acceptable to the convert_authorization_ subroutine. (See the convert_authorization_ subroutine in the MPM Subroutines.) The authorization must be less than or equal to both the maximum authorization of the process and the access class of the terminal. The default is to create the new process at the same authorization.

Notes: If the user's initial working directory contains a segment named start_up.ec, and the user did not log in with the -no_start_up control argument, new_proc causes the command line:

```
! exec_com start_up new_proc interactive
```

to be automatically issued in the new process. This feature can be used to initialize per-process static variables.

no_save_on_disconnect

03/13/80 no_save_on_disconnect

Syntax: no_save_on_disconnect

Function: disables process preservation across hangups in the user's process, causing the process to log itself out automatically if its terminal channel hangs up.

Notes: This command is only meaningful if process preservation was in effect for the process at login time, either by default or because the -save_on_disconnect control argument was specified on the login command line.

—

on

—

05/22/81 on

Syntax:
```
   on conditions handler_com_line  -control_args
        subject_com_line
```

Function: establishes a handler for a specified set of conditions, executes an imbedded command line with this handler in effect, and then reverts the handler. The handler is another imbedded command line to be executed if the condition is signalled.

Arguments:

conditions
   is a list of condition names separated by commas to be trapped by the on command.

handler_com_line
   is the command line to be executed when one of the conditions contained in the list of condition names is raised. If handler_com_line contains spaces or other command language characters, it must be enclosed in quotes. If no command is to be executed when a condition is raised, handler_com_line must be given as "".

subject_com_line
   is the command line to be executed under the control of on. The subject_com_line consists of the remaining arguments and should be quoted if it contains parenthesis, brackets or semicolon.

Control arguments:

-brief, -bf
   suppresses the comment printed when a condition occurs.

-long, -lg
   prints a detailed message describing the condition raised, if one is available.

-restart, -rt
   continues execution of the subject_com_line after execution of handler_com_line, or if -cl is also specified, after the start command is executed.

-retry_command_line, -rcl
   causes subject_com_line to be aborted and exectued over again after executing handler_com_line.

-cl
   establishes a new command level after the execution of handler_com_line. The state of subject_com_line is preserved. If the start command is issued, the same action is taken as would have been if -cl had not been specified. This control argument is not allowed for the on active function.

-exclude STR, -ex STR
   prevents on from trapping the conditions given in STR. If more than one condition is listed, condition names are separated by commas.

This control argument is useful when handling the any_other condition.

Notes: The default action after executing handler_com_line is to abort the execution of subject_com_line; this is modified by the -restart and -rcl control arguments.

If a condition is raised and trapped by the on command while executing the handler_com_line, it is considered a recursive signal, and the entire invocation of the on command is aborted.

The message produced by the -long control argument is the same as the message printed by the reprint_error command. The -brief and -long control arguments are mutually exclusive. The -rcl and -restart control arguments are mutually exclusive.

See the MPM Reference Guide for a list of standard system conditions.

Syntax as active function:

```
[on conditions handler_com_line  -control_args
      subject_com_line]
```

Notes on active function:
The active function returns true if any of the specified conditions are signalled during the execution of subject_com_line, false otherwise.

print

12/21/81 print, pr

Syntax: print paths -control_args

Function: prints ASCII segments and multi-segment files on user_output.

Arguments:
paths
are the pathnames of the segments and multisegment files to be printed. The star and archive component (* and ::) conventions are accepted.

Control arguments:
-archive, -ac
treat each archive component as a new file for heading and line numbering. If any lines are printed from an archive component, and if -header is specified, print a header identifying the archive component name and the date of modification of the archive component, in the format--

ARCHIVE::COMPONENT date time

where date and time are those stored in the archive. -archive is the default if archive components were named with the :: convention, or if the entryname of the segment ends in ".archive", unless -no_archive is specified.

-chase
if a starname is specified, include links in the search. Do not complain about missing link targets for starnames.
-exclude STRING, -ex STRING
don't print lines containing STRING. Exclusion is done after matching. Thus, "-match A -exclude B" prints all lines with an A except those with a B.
-exclude /REGEXP/, -ex /REGEXP/
don't print lines containing a string matching the regular expression REGEXP.
-for N
print N lines from the file including the first line. The default is to print the whole file. If -to is also specified, printing stops when the first control argument is satisfied.

-from X, -fm X
begin printing from the X'th line. The default is line 1.
-from /REGEXP/, -fm /REGEXP/
begin with first line matching the regular expression REGEXP. See the writeup of the qedx command for the definition of regular expressions.
-from_page PP
start printing with the PP'th page, counting the first page as 1.

The default is to print starting with the first page.

-header, -he
print a header of the form--
NAME date time
before each segment. If -archive is specified, the header is printed before each archive component instead of before each segment. -header is the default if no other control arguments are given, or if multiple pathnames or the star convention are used.
-indent XX, -ind XX
print XX blanks before each line. This indents the printed output XX columns. The default is no indentation.
-left_col XX, -lc XX
don't print columns 1 to XX-1. This argument removes a slice from the left_hand edge of the file, and prints each line of the file starting with column XX. If a line has fewer than XX columns, a blank line is printed. The default is to print starting with column 1.

-line_length YY, -ll YY
format the page with a maximum physical line length of YY characters. Space generated by -indent and -number is not counted. If more than YY characters are in an output line, the line is split and continued on the next line. The default maximum line length is 1024 characters (larger values may be specified.)
-match STRING
print only lines containing the character string STRING.
-match /REGEXP/
print only lines containing a string matching the regular expression REGEXP.
-name NAME, -nm NAME
take NAME literally, even if it is all numeric or begins with "-".

-no_archive, -nac
even if the file being printed is an archive, do not print headings for individual archive components; treat it as a single segment for line numbering and heading.
-no_chase
do not include links when processing starnames. This is the default.
-no_header, -nhe
suppress the header before segments or archive components. This is the default if only one pathname is given and other control arguments are used.
-no_vertsp
simulate formfeed and vertical tab characters by outputting newline characters.

-number, -nb
print line numbers before each line. The line number and the spaces separating it from the line take up 10 spaces.
-page_length ZZ, -pl ZZ
start a new page after every ZZ lines from the file are printed. The default is no pagination, as if ZZ were infinite.
-phys_page_length ZZ, -ppl ZZ
format the page with a physical page length of WW lines. This causes a page eject to be generated every WW lines. These page ejects are

intended to skip over the perforation between physical pages; the default value of WW is 66. This value is also used to determine how many newline characters are used to simulate a formfeed of -no_vertsp is specified.

-right_col YY, -rc YY
   don't print columns past YY. This slices characters off the right-hand side of the file. The default is to print all columns.
-stop, -sp
   pause after each page until the user types a newline. Also pause before the first page.
-to X
   stop printing with line number X. The default is to print all lines.
-to /REGEXP/
   stop printing with the first line matching the regular expression REGEXP. The search for REGEXP begins after the first line printed.

-to_page PP
   stop printing after the PP'th page. The default is to print the whole file.
-vertsp
   send formfeed and vertical tab characters to the terminal. This is the default.
-wait, -wt
   pause before the first page until the user types a newline.

Notes: If any of -right_col, -line_length, -page_length, or -phys_page_length is specified, or -left_col is > 1, printing will be done via the printer conversion software: overstrikes will be replaced by multiple lines separated by CR (015) characters, and other control characters will be ignored.

Numeric arguments are processed specially for compatibility with previous versions of this command. If no file name has been found, a number is interpreted as a file name; other numeric arguments are interpreted as -from and -to in that order. The -name control argument can be used to indicate that a number is intended as a pathname.

More than one -match control argument and more than one -exclude control argument may be specified; a line is printed if any of the -match arguments select it, unless one of the -exclude arguments prevents it from being printed.

print_attach_table

02/02/79 print_attach_table, pat

Syntax: pat switch_names -control_args

Function: prints a list of I/O switches and information about them.

Arguments:
switch_names
   are starnames used to select the switches to be processed.
   If no switch_names are specified, all I/O switches that are currently
   attached are processed.

Control arguments:
-name switch_name, -nm switch_name
   causes the next argument to be interpreted as a literal switchname, even if
   it looks like a starname or control argument.
-brief, -bf
   suppresses the processing of the four standard switches (user_input,
   user_output, user_i/o and error_output)
-all, -a
   processes all switches, even those that are not attached.
-attached, -att
   processes only attached switches. This is the default.
-open
   processes only open switches.

Notes: The output from this command is a table listing the name of each switch processed, its attach description (if attached) and its open description (if open). The switches processed are selected by starname match and by whether they match the criteria specified by the control arguments.

Syntax as an active function:
 [pat -control_args switch_names]
returns a string containing the names of all the switches selected, separated by spaces.

print_default_wdir

02/26/76 print_default_wdir, pdwd

Syntax: pdwd

Function: prints the pathname of the current default working directory.

print_mail

08/21/80 print_mail, prm

Syntax: prm address -control_args

Function: prints all the messages in a mailbox, querying the user whether to delete each one.

Arguments:
address
   specifies the address of a mailbox. See "List of Addresses" below. If no address is specified, the user's default mailbox is assumed.

Control arguments:
-brief, -bf
   suppresses the printing of the informative messages.
-interactive_messages, -im
   operates on interactive messages from send_message as well as mail messages from send_mail. This is the default.
-list, -ls
   prints a summary of the messages in the mailbox before entering the request loop.
-no_interactive_messages, -nim
   operates on send_mail messages, not on interactive messages sent by send_message.

List of addresses:
-pathname path, -pn path
   where path is the pathname of a mailbox. The mbx suffix is assumed.
-user Person_id.Project_id
   specifies the Person_id and Project_id of a user whose mailbox is to be read.
STR
   is any argument not beginning with a minus (-) sign, and is interpreted as -pathname STR if it contains > or < characters. Otherwise, it is interpreted as -user STR.

Notes on query responses: After printing each message, print_mail asks the question:

   print_mail: Delete message N?

where N is the number of the message just printed.

Five responses are allowed:

yes
   the message is deleted and the next one is printed.
no

the message is not deleted and the next one is printed.

reprint
the message just printed is printed again, and the question is asked again.

quit, q
returns the user to command level after deleting the messages specified.

abort
returns to command level, without deleting any messages.

Notes on creating a mailbox: A default mailbox is created automatically the first time a user issues print_mail, read_mail, accept_messages, or print_messages. The default mailbox is:

>user_dir_dir>Project_id>Person_id>Person_id.mbx

Notes on extended access: Access on a newly created mailbox is automatically set to adrosw for the user who created it, aow for *.SysDaemon.*, and aow for *.*.*. The types of extended access for mailboxes are:

add	a	add a message.
delete	d	delete any message.
read	r	read any message.
own	o	read or delete only your own messages.
status	s	find out how many messages are in the mailbox.
wakeup	w	send a wakeup when adding a message.

The modes "n", "null", and "" specify null access.

Notes on related commands: Special commands exist to create additional mailboxes and to change the attributes of mailboxes. These commands, described in the MPM Subsystem Writers' Guide, are:

mbx_create
create a mailbox.

mbx_delete
delete a mailbox.

mbx_add_name
add a name to a mailbox.

mbx_delete_name
delete a name from a mailbox.

mbx_rename
rename a mailbox.

mbx_list_acl
list the access control list of a mailbox.

mbx_set_acl
change or add entries to the ACL of a mailbox.

mbx_delete_acl
delete entries from the ACL of a mailbox.

mbx_set_max_length
set the maximum length of a mailbox.

mbx_safety_switch_on
turn on the safety switch of a mailbox.

mbx_safety_switch_off

turn off the safety switch of a mailbox.

For more information, see the read_mail and send_mail commands descriptions.

print_messages

07/17/81 print_messages, pm

Syntax: pm destination -control_args

Function: prints any interprocess messages that were received (and saved in the user's mailbox) while the user was not accepting messages, not logged in, or accept_messages -hold was in effect.

Arguments:
destination
can be of the form Person_id.Project_id to specify a mailbox. If destination contains > or <, it is the pathname of a mailbox. If no destination is specified, the user's default mailbox is assumed.

Control arguments:
-all, -a
prints all messages, including those held by -hold mode (see accept_messages). This is the default.
-call cmdline
for each message, instead of printing calls the command processor with the line:
cmdline number sender time message path
For more details, see the accept_messages command.
-last, -lt
reprints only the latest message received.
-long, -lg
prints the sender and date-time of every message, even when the same for two consecutive messages.

-new
when accept_messages -hold mode is in effect, prints only those messages that have not been printed before. The default is to print all held messages.
-pathname path, -pn path
specifies a mailbox by pathname. The mbx suffix is assumed. This control argument and the destination argument are mutually exclusive.
-short, -sh
prints messages as with accept_messages -short, omitting redundant sender names in favor of the prefix "=:". This is the default.

Notes: Messages are deleted after they are printed unless the -hold argument was given to the accept_messages command. The "last" message remains available for the life of the process or until redefined by a new message.

If messages are deferred, it is a good practice to print out pending messages periodically.

For a description of the mailbox, refer to the accept_messages and print_mail commands. See also the active functions last_message, last_message_sender, and last_message_time.

print_motd

06/25/81 print_motd, pmotd

Syntax: pmotd

Function: prints out changes to the message of the day since the last time the command was called.

Notes: The segment Person_id.motd is created in the user's home directory to hold the previous version of the message of the day.

Anonymous users use the segment Login_id.motd where Login_id is the name used on the enter or enterp line.

print_pdt

07/14/77 print_pdt

Syntax: print_pdt pdt_name user_name -control_args

Function: print contents of a PDT; can be used to make a PMF from a PDT.

Arguments:
pdt_name
   name of PDT. Can be the pathname of a PDT other than the live PDT found in >sc1>pdt.
user_name
   print only information about user_name.

Control arguments:
-brief, -bf
   print information derived from PMF whose values differ from the default.
-long, -lg
   print all information found in the PDT.
-pmf
   print in format suited for a PMF.
-no_header, -nhe
   do not print header.

Notes: The user must have access to the PDT; most users who are not project administrators do not have such access.
  If none of -pmf, -brief, or -long are given, all PMF specifiable attributes and the total amount spent are printed.
  To make a PMF out of a PDT, the following command line is recommended:
    fo projname; print_pdt projname -pmf; ro

print_search_paths

01/17/79 print_search_paths, psp

Syntax: psp search_lists -control_args

Function: prints the search paths in specified search lists.

Arguments:
search_lists
   are names of search lists to print. If none are specified, all search lists referenced in the process are printed.

Control arguments:
-expanded, -exp
   print all keywords except -referencing_dir and all unexpanded search paths as absolute pathnames.

Notes: All synonyms of a search list name are printed if no search lists are specified.

Syntax as an active function: [psp search_list -control_args]

print_search_rules

02/13/76 print_search_rules, psr

Syntax: psr

Function: prints the search rules currently in use.

print_terminal_types

08/25/77 print_terminal_types, ptt

Syntax: print_terminal_types ttt_path

Function: Displays the names of all currently-defined terminal types.

Arguments:
ttt_path
is the pathname of the terminal type table (TTT) to be used to find the names of the defined terminal types. If it is omitted, the system TTT is used.

print_wdir

02/13/76 print_wdir, pwd

Syntax: pwd

Function: prints the pathname of the current working directory.

probe

09/30/79 the probe command

Syntax: pb procedure_name

Function: provides symbolic, interactive debugging facilities for programs compiled with PL/I, FORTRAN, or COBOL.

Arguments:

procedure_name
can be the reference name of an initiated program, a pathname, or if not given, the procedure owning the frame in which the last condition was raised is assumed.

Notes: Probe is self-documenting. For further information, invoke probe and type "help". For a list of all requests, type "list_requests". For a list of all topics described by probe, type "list_help".

probe.gi

09/30/79 General information about probe

Probe is a symbolic debugger for programs written in PL/I, FORTRAN, and COBOL. It permits a user to interrupt a program at a specified location, to examine or modify program variables, to examine the stack history of block invocations, to display spurce lines associated with an object location. Expression of program variables may be aevaluated and external functions may be called.

Self-documentation:
By using the probe "help" request one can see probe information files. To see a list of them, type "list_help". To see a specific file, type "help file_name". This info file is available by typing "help".

probe interaction:
When probe has been invoked it accepts requests from the user. A probe request consists of a keyword (the name of the request to be performed) and its arguemnts, if any. More than one request may appear on a line if they are separated by semi-colons (";"). For a list of all requests, type "list_requests".

Breakpoints:
A breakpoint is a set of probe requests associated with a statement of a program. This set of requests is executed automatically by probe whenever the location in the object segment corresponding to the statement is executed. The most common request is "halt", which suspends execution of the program and listens for further requests. The user may then examine and modify the state of the suspended program, and resume (or abort) further execution. Other uses of breakpoints are to display the state of the program without halting, or to effect source-level patching by executing assignments. For more information, type "help breaks".

process_dir

07/23/80 process_dir, pd

Syntax: pd

Function: returns the pathname of the process directory of the process in which it is invoked.

Syntax as active function: [pd]

process_list

07/07/81 process_list, pls

Syntax: pls list_path form_path -control_args

Function: produces a document from all or selected records in a lister file.

Arguments:

list_path
is the pathname of the lister file to be processed. Lister must be the last component of the lister file name; however, if list_path does not have a suffix of lister, one is assumed.

form_path
is the pathname of the listform file that defines the format of the document. If form_path does not have a suffix of listform, one is assumed. If this argument is not specified, a listform file in the working directory is used that has the same entryname as list_path, with the entryname suffix of lister changed to listform.

Control arguments:

-argument STR, -ag STR
indicates that the listform segment requires arguments. If present, it must be followed by at least one argument. All arguments following this control argument on the command line are taken as arguments to the listform segment. Thus, if present, this must be the last control argument on the command line.

-brief_errors, -bfe
suppresses warnings about missing or extra arguments for the -ag control argument. Also suppresses the warning when no records were selected.

-output_file path, -of path
specifies that the document produced by this command is saved in the segment specified by path (see Sample Letter in "Sample List Processing Files" in Multics Wordpro Reference Guide, Order No. AZ98).

-extend, -ex
specifies that the document produced by this command is to be appended to the segment specified by path (-output_file must also be given). The default is to replace path completely.

-select STR, -sel STR
specifies the records selected for processing. If this control argument is not specified, then all records in the list are processed

-sort STR, -st STR
sorts the records processed according to sort_string, which is a string enclosed in quotes. The new ordering of the list is in effect only for the duration of the command. The lister file is not modified. If this control argument is not specified, then records are processed in the order in which they currently appear in the

lister file
-totals, -tt
prints the number of records processed.

Notes: The format of the document is defined in a listform file. Other text processors, such as compose, may be used to further format the document. By default, the document is printed on the user's terminal. Alternatively, it may be saved in a segment. For a description of the structure of a listform file and information on field insertion, angle bracket escapes, and the selection and sorting procedures (-select and -sort control arguments), see the Multics Wordpro Reference Guide, Order No. AZ98.

profile

12/22/80 profile, pf

Syntax: pf program_names -control_args

Function: a performance measuring tool that analyzes the time spent executing each source statement of a program, along with other parameters of interest, after the program is run.

Arguments:
program_names
are pathnames or reference names of programs to be analyzed. Any program_name that does not include "<" or ">" characters is assumed to be a reference name. They need not be specified if the -input_file control argument is used.

Control arguments: Control arguments apply to all programs specified, and can be given in any order.
-brief, -bf
used with -print to exclude from the output all information for statements that have never been executed. This is the default.
-comment STR, -com STR
used with the -output_file control argument to include STR with the stored profile data as a comment. This control argument can also be used with -plot. If STR is to include blanks or other characters recognized as special by the command processor, it should be enclosed in quotes. STR can be up to 128 characters long.
-first N, -ft N
used with -sort to print only the first N values.

-from N, -fm N
used with -print or -plot to begin the output with the data for line number N. The default is 1.
-hardcore, -hard
indicates that the specified programs are supervisor (hardcore) segments. The current (internal static) profile data for such programs is retrieved from the address space of the supervisor. Hardcore programs compiled with the -profile or -long_profile control arguments must be installed by generating a Multics System Tape and rebooting Multics. See System Programming Tools, Order No. AZ03, for a description of the generate_mst command. Note that the current (internal static) profile data for hardcore programs cannot be reset (zeroed).

-input_file path, -if path
causes the profile data to be retrieved from the profile data file specified by path. Use of this control argument causes the current (internal static) profile data, if any, to be ignored. The pfd suffix is appended to path if it is not already present. If any program_names are specified, they select a subset of the stored data

for analysis. If no program_names are specified, all data stored in the profile data file is used. This control argument is inconsistent with -output_file.

-line_length N, -ll N
used with -list to specify an output width of N characters. The default is 132.

-list, -ls
creates a profile listing for all specified programs. The profile listing file is given a name consisting of the first program name with the language suffix replaced by the pfl suffix. It is placed in the working directory. The information described above for the -print control argument is placed in columns to the left of each source line in the profile listing.

-long, -lg
used with -print to include in the output information for statements that have never been executed.

-max_points N, -mp N
used with -plot to specify the maximum number of points (line numbers) to be plotted (the graphics resolution). The default is 250. The Multics Graphics System is capable of plotting up to 1024 points.

-no_header, -nhe
used with -print to suppress column headings.

-output_file path, -of path
causes the profile data for the specified program_names to be stored in the profile data file specified by path. The file is created if it does not already exist and is overwritten if it already exists. The pfd suffix is added to path if it is not already present. The profile data is stored in a format acceptable to the -input_file control argument. The format of pfd data files is described by the PL/I include file pfd_format.incl.pl1. The stored data is determined by the program_names specified, the -comment control argument and whether the compilation was done using the -profile or -long_profile options. The name a program was compiled with is saved in the profile data file. If program_name specifies a bound object segment, profile data about each component of the bound object segment is saved.

-plot STR
plots a bar graph, on any supported graphics terminal, of the values of the specified field STR. STR can be any of the fields in the "List of Fields" section below. Use of this control argument requires that the site has installed the Multics Graphics System, and that the setup_graphics command has been executed. See the Multics Graphics System, Order No. AS40, for more information.

-print, -pr
prints the following information for each statement in the specified program(s):
1. Line Number.
2. Statement Number
   if more than one statement on the line.
3. Count
   the number of times the statement was executed.

4. Cost
an approximation to the accumulated execution time for the statement. Equal to the number of instructions executed plus ten times the number of external operators called.

5. Stars (asterisks)
an indication of the percentage of total cost (or time, for long_profile data) used in the statement. The number of stars is selected according to the table below.

```
4 stars:  20% to 100%
3 stars:  10% to 20%
2 stars:   5% to 10%
1 star:    2.5% to 5%
no stars:  0% to 2.5%
one period:  Statement was not executed.
```

6. Names of all external operators called by the statement.

For -long_profile (actual accumulated time) data, item 4 is changed to the following:

4a. Time
actual execution time for the statement in virtual CPU microseconds, including all time spent in any operators or subroutines invoked by the statement.

4b. Average Time
Time divided by Count (the average execution time for one execution of the statement).

4c. Page Faults
page faults incurred in executing the statement.

-reset, -rs
resets (zeros) all current (internal static) profile data for the named program(s). When -reset is specified, the resetting is done as the very last operation if -print, -list, -plot, or -output_file are also specified. This control argument is inconsistent with -input_file and -hardcore.

-search_dir path, -srhd path
used with -hardcore to add path to an internal search list of hardcore object directories. Up to 8 directories can be specified. If no search list is specified, >ldd>hard>o is searched for copies of the specified program(s).

-sort STR
used with -print to sort profile information into descending order of the specified field STR, which can be any of the fields in the "List of Fields" section below.

-source_dir path, -scd path
used with -list when the source segments to be listed have been moved from the directories in which they were compiled. If -source_dir is specified, only the directory specified by pathname path is searched for source segments.

-to N
used with -print or -plot to end the output with the data for line number N. The default is the line number of the last executable statement.

List of Fields:
count
   number of times statement was executed.
time
   vpcu time of statement (-long_profile).
cost
   approximate cost of statement (-profile).
page_faults (pfs)
   page faults taken during statement (-long_profile).

Notes: The program to be analyzed must be compiled using the -profile (-pf) control argument of the cobol, fortran and pl1 commands, or using the -long_profile (-lpf) control argument of the pl1 command. The -long_profile compiler control argument is used to acquire exact elapsed time statistics and is much more expensive to use than the -profile compiler control argument.

program_interrupt

07/25/81 program_interrupt, pi

Syntax: pi

Function: informs a suspended invocation of a subsystem that the user wishes to abort the current request.

Notes: The program interrupt command is used with interactive subsystems. Interactive subsystems are commands that accept user requests from the terminal. To abort a subsystem request the user uses the quit (or break) key to interrupt execution, and then gives the program_interrupt command. If the subsystem supports the use of the program_interrupt command, it will abort the interrupted request and ask the user for a new one. If the subsystem does not support the use of program_interrupt, the command will print an error message. The user may then either restart the interrupted operation with the "start" command, or abort the entire subsystem invocation with the "release" command.

List of subsystems:
   The following subsystems support the use of the
   program_interrupt command--
debug
edm
emacs
help
lisp
print_mail
probe
qedx
read_mail
send_mail
teco
ted

qedx

03/01/76 qedx, qx

Syntax: qx path optional_args

Function: context editor used to create and edit ASCII segments; also allows the user to create macros and use the editor as an interpretive programming language.

Arguments:

path
pathname of an ASCII segment from which the editor is to take its initial instructions; the qedx suffix must not be given.

optional_args
are appended, each as a separate line, to the buffer named args.

Requests: listed below in four categories giving, format, default in parentheses, and brief description. For value of ADR, see Addressing below; regexp, see Regular expression.

INPUT--these requests enter input mode and must be terminated with f.

ADRa (.a) append lines after specified line.

ADR1,ADR2c (.,.c) change existing line(s); delete and replace.

ADRi (.i) insert lines before specified line.

BASIC EDIT REQUESTS--

ADR1,ADR2d (.,.d) delete line(s).

ADR1,ADR2p (.,.p) print line(s).

ADR= (.=) print line number.

q exit from qedx editor.

ADRr path ($r path) append contents of path after specified line.

ADR1,ADR2s/regexp/string/ (.,.s/regexp/string/) substitute every regexp in the line(s) with string. If string contains &, & is replaced by regexp. First character after s is delimiter; it can be any character not in either regexp or string.

ADR1,ADR2w -path- (1,$w path) write lines into segment named path; if path omitted, default pathname used.

/regexp/ set the value of "." to the first line following the current line that contains regexp and print the line.

EXTENDED EDIT REQUESTS--

e <command line> execute command line without leaving editor.

ADR1,ADR2gX/regexp/ (1,$gX/regexp/) perform operation on lines that contain regexp; X must be d for delete, p for print, or = for print line numbers.

ADR1,ADR2vX/regexp/ (1,$vX/regexp/) perform operation on lines that do not contain regexp; X must be d for delete, p for print, or = for print line numbers.

BUFFER REQUESTS--
b(X) go to buffer named X.
ADR1,ADR2m(X) (.,.m(X)) move line(s) from current buffer into buffer named X.
x give the status of all buffers in use.
ADRn (.n) set value of "." to line addressed.
ADR" (.") ignore rest of line; used for comments.

Addressing: Most editing requests are preceded by an address specifying the line or lines in the buffer on which the request is to operate. Lines in the buffer can be addressed by absolute line number; relative line number, i.e., relative to the "current" line; and context. Current line is denoted by period (.); last line of buffer, by dollar sign ($).

Regular expressions:
The following characters have specialized meanings when used in a regular expression. The user can reinvoke the last used regular expression by giving a null regexp (//).

* signifies any number (or none) of the preceding character.

^ when used as the first character of a regular expression, signifies the character preceding the first character on a line.

$ when used as the last character of a regular expression, signifies the character following the last character on a line.

. matches any character on a line.

Escape sequences:
f exit from input mode and terminate the input request.

c suppress the meaning of the escape sequence or special character following it.

b(X) redirect editor stream to read subsequent input from buffer X.

r temporarily redirects the input stream to read a single line from the user's terminal.

read_mail

03/12/79 read_mail, rdm

Syntax: rdm INPUT_SPEC -CONTROL_ARGS

Function: Selectively lists, prints, deletes, saves and forwards messages and mail sent to a mailbox.

Arguments: INPUT_SPEC can be any one of the following; if no INPUT_SPEC is present, the users default mailbox is used:
>udd>Project>Person>Person.mbx

-mailbox PATH, -mbx PATH
   specifies the pathname of a mailbox.
-user Person.Project
   specifies the mailbox of a given user.
-save PATH, -sv PATH
   specifies a savebox PATH.sv.mbx
-log
   specifies the user's logbox, [hd]>Person.sv.mbx
STRING
   If STRING contains a single period, it is interpreted as a Person.Project identifier; otherwise, it is interpreted as a pathname.

Control arguments:
-brief, -bf
   Shorten informative messages from read_mail.
-interactive_messages, -im
   Include interactive messages, as from the send_message command.
-list, -ls
   List mailbox contents before entering request loop.
-long, -lg
   Print full informative messages. (default)
-no_interactive_messages, -nim
   Ignore interactive messages. (default)

-no_list, -nls
   Do not list msgs before entering request loop. (default)
-no_print, -npr
   Do not print msgs before entering request loop. (default)
-no_prompt
   Suppress the prompt in the request loop. (default is "read_mail:")
-own
   Read only messages sent by the user.
-print, -pr
   Print messages before entering the request loop.
-prompt STR
   Set the request loop prompt to STR.

-quit
   Exit read_mail after performing any operations given by the -list,
  -print, or -request control arguments.
-request STR, -rq STR
   Execute the requests in STR before entering the request loop.
-request_loop, -rql
   Enter the read_mail request loop even if there are no messages
   in the mailbox
-totals, -tt
   Print the number of the messages in the mailbox, and return. This
   argument is incompatible with -print and -list.

Default setting control arguments:
These arguments can be used to set defaults for the behavior of various read_mail requests. Their effect is to change the behavior of a particular request so that it behaves as if the specified control argument is always given. The defaults can all be overridden by using the negative form of the control argument.

Default setting for the print (pr) request:
  -no_header, -nhe
  -header, -he

Default setting for the reply (rp) request:
  -fill, -fi
  -include_authors, -iat
  -include_original, -io
  -include_recipients, -irc
  -no_fill, -nfi
  -no_include_authors, -niat
  -no_include_original, -nio
  -no_include_recipients, -nirc

Notes:
Request lines use () for iteration, "" for quoting, and [] to invoke read_mail active requests, listed below ("List of active requests").

Any request line which befgins with ".." will be passed directly to the Multics command processor with the leading ".." stripped off. This is the recommended method for executing Multics commands from within the read_mail subsysem, as the execute request is cumbersome.

Message specifiers:
Message specifiers (SPECS) are message numbers ("17"), ranges ("6:9"), string matches ("/STR/" or "/STR1/&/STR2/"), and expressions involving any of the keywords first (f), last (l), next (n), previous (p), current (c) and all (a), such as "6:last-3" or "all/STR/". A full description can be obtained with the read_mail request "help msg_specs".

Messages are not actually deleted from the mailbox until the "quit" request is issued, so that "deleted" messages can be retrieved with the "retrieve" ("rt") request.

```
List of requests:
   Only the most important aspects of the requests are documented here; in
   particular, many requests take more control arguments.  For further
   information on these requests, use the read_mail 'help' request.

?  lists the available read_mail requests and active requests.
.  identifies read_mail with version number, recursion level, mailbox
   pathname, message count, and current message number.


delete  SPECS , dl  SPECS
   Delete specified messages.
help  STR
   Print information about request names or topics. A list of available
   topics is produced by the request "help *".
list  SPECS   -CA's , ls  SPECS   -CA's
   Produce a summary listing of specified messages.
print  SPECS   -CA's , pr  SPECS   -CA's
   Print specified messages. -CA can be -header (-he) or -no_header
   (-nhe), among others.
quit  -CA's , q  -CA's
   Exit read_mail. -CA's can be -no_modify (-nm) to not change the
   mailbox, -force (-fc) to ignore newly arrived messages.
retrieve SPEC S , rt SPEC S
   Un-delete specified messages.


append  SPECS  PATH
   Write msgs at the end of an existing ASCII segment.
copy  SPECS  PATH, cp  SPECS  PATH
   Copy msgs intact to another mailbox.
forward SPEC ADDRESSES  -CA's
   Forward msgs specified by one specifier. ADDRESSES can be -user
   Person.Project, -mbx PATH, or STRING, as above.
log  -CA   SPECS
   Copy msgs to user's logbox, adding sender information to the message
   header if not already present. -CA can be -delete (-dl).
preface  SPECS  PATH
   Write msgs at the front of an existing ASCII segment.
save  -CA   SPECS  PATH, sv  -CA   SPECS  PATH
   Save msgs in a mailbox, adding sender information to the message
   header if not already present.
write  -CA   SPECS  PATH, w  -CA   SPECS  PATH
   Write messages to the end of a new or old ASCII segment.  -CA can be
   -truncate (-tc) to specify that the segment is to be truncated first.


execute STR's, e STR's
   Concatenate all STS's together and execute result as a Multics
   command line.  Active requests are replaced by their values
   before string is constructed.
reply  SPECS   -CA's , rp  SPECS   -CA's
   Send replies to the senders and recipients of specified messages by
   calling the send_mail subsystem.


List of active requests:
execute STR's, e STR's
   Concatenate STR's, execute as a Multics active function, and return
   its value.
```

current -CA's
   Return the number of the current message.
first -CA's
   Return the number of the first message.
last -CA's
   Return the number of the last message.
next -CA's
   Return the number of the next message.
previous -CA's
   Return the number of the previous message.
all -CA's
   Return a string consisting of all the message numbers.
mailbox, mbx
   Return the absolute pathname of the mailbox being read.

ready

06/20/80 ready, rdy

Syntax: rdy

Function: types out an up-to-date ready message whose format is optionally set by the general_ready command. The default ready message if general_ready is not used gives the time of day and the amount of CPU time and page faults used since the last ready message was typed. If the user is not at the first command level, i.e., if some computation has been suspended and the stack frames involved not released, the default ready message also contains the number of the current command level.

Notes: See the descriptions of ready_on, ready_off, and general_ready in MPM Commands and Active Functions, AG92.

ready_off

02/13/76 ready_off, rdf

Syntax: rdf

Function: turns off the ready messages printed at command level.

ready_on

02/13/76 ready_on, rdn

Syntax: rdn

Function: prints a ready message after each command line has been processed.

release

02/13/76 release, rl

Syntax: rl -control_arg

Function: releases the stack history preserved after a quit signal or unclaimed signal.

Control arguments:
-all, -a
to release the stack history preserved (and not already released) after all previous quit and/or unclaimed signals rather than after only the most recent quit or unclaimed signal.

release_resource

04/24/81 release_resource, rlr

Syntax: release_resource type STR1 ... STRn -control_arg

Function: The release_resource command releases a resource into the free pool. A resource may only be released by its accounting owner or privileged processes.

Arguments:

type
is a resource type defined in the resource type description table (RTDT).

STRi
is the unique identifying name of the particular resource being released. If STR looks like a control argument (i.e., if it is preceded by a hyphen), then it must be preceded by -name or -nm.

Control arguments:

-priv
specifies that the user wishes to perform a privileged release of this resource from the accounting owner, even though the user may not be the accounting owner (see "Access Restrictions" below).

Access Restrictions: The use of the -priv control argument requires execute access to the rcp_admin_ gate.

rename

01/10/77 rename, rn

Syntax: rn -control_arg path1 name1 ... -control_arg pathN nameN

Function: renames entries.

Arguments:

pathN
specifies the old name that is to be replaced; it can be a pathname or an entryname. The star convention is allowed.

nameN
specifies the new name that replaces the entryname portion of pathN. The equal convention is allowed.

Control arguments:

-name, -nm
causes the next path and name to be taken literally, without applying the star or equal conventions and disregarding special characters such as >.

reorder_archive

06/01/76 reorder_archive, ra

Syntax: ra -control_arg1 path1 ... -control_argN pathN

Function: reorder the contents of an archive segment according to a list specified by the user.

Arguments:
pathN
is the pathname of an archive segment; the archive suffix need not be given.

Control arguments:
-console_input, -ci
indicates user will list component names in order desired. See "Notes."
-file_input, -fi
indicates that a list of component names is to be found in the segment named XXX.order where XXX is the name of the archive segment. See "Notes" below.

Notes: When -ci control argument is used, user enters component names separated by line feeds in order desired. A period (.) on a line by itself terminates input; a two character line, .q, terminates the command without reordering the archive.

When -fi control argument is used, archive is reordered according to order specified in XXX.order. Errors in the list terminate the command without altering the archive.

reserve_resource

01/18/79 reserve_resource, rsr

Syntax: rsr -resource (-rsc) resource_description

Function: reserves a resource or group of resources for use by the calling process. The reservation takes effect immediately and lasts until cancelled by the cancel_resource command or by process termination.

Control arguments:
-resource STR, -rsc STR
this control argument must be present. The string, STR specifies a description of the resources to be reserved. If this resource description contains spaces or special characters, it must be enclosed in quotes. This resource description can also have control arguments and is described in detail below.

Examples: rsr -rsc "tape_drive -attr track=9,den=1600 -rsct tape_vol u309"

Resource Description:
A resource description describes certain devices and volumes by name or by attributes and an optional number. It has the following format:

-resource_type resource_spec1 ... -resource_type resource_specn

That is, a series of at least one resource spec where all but the first must be preceded by the -resource_type or -rsct control argument. The first one may or may not be preceded by the control argument.

The format of a resource_spec is as follows:

volume_type name1 names
or:
device_type names
or:
device_type -control_args

where:
volume_type
can be either tape_vol or disk_vol.
device_type
can be either tape_drive or disk_drive.

Control arguments:
-attributes STR, -attr STR
for tape drives STR can consist of a string of attributes with values separated by commas with no spaces. The attributes allowd for tape drives are: model=, track=, and den=. For disk drives the only attribute allowed is: model=.

-number N, -nb N
   is the number of identical resources of the type desired.

Example:
tape_vol 50102 u309 -rsct tape_drive -attr track=9,den=800 -nb 2

This describes four resources: two tapes, 50102 and u309; and two tape drives, both being 9-track and capable of 800 bpi operation.

resource_status

04/24/81 resource_status, rst

Syntax: resource_status type STR1 ... STRn -control_args

Function: The resource_status command prints selected information about the status of a given resource. This command can also be invoked as an active function (see "Notes" below).

Arguments:

type
   is a resource type defined in the resource type description table (RTDT).

STRi
   is the unique identifying name of the particular resource desired. If STR looks like a control argument (i.e., if it is preceded by a hyphen), then it must be preceded by -name or -nm.

Control arguments:

-access_class, -acc
   prints the AIM access class or access class range of the resource (see "Notes" below).

-acs_path
   prints the pathname of the ACS for this resource (see "Notes" below).

-all, -a
   specifies that all information maintained about this resource is to be printed. This control argument is not allowed in an active function invocation.

-alloc
   specifies that the state of the user allocation switch for this resource is to be printed.

-attributes, -attr
   prints the current and protected attributes of this resource.

-charge_type, -crgtp
   prints the charge type for this resource.

-comment, -com
   prints the user-settable comment associated with this resource.

-location, -loc
   prints the location field associated with this resource.

-lock
   prints the status of the resource lock for this resource. In an active function invocation, "true" is returned if the lock is on; "false" is returned if it is off.

-mode, -md
   prints the user's effective mode to the resource.

-owner, -ow
   prints the name of the owner of the resource.

-potential_access_class, -pacc
   prints the potential access class or potential access class range
   for this resource (see "Notes" below).
-potential_attributes, -pattr
   prints the potential attributes of this resource.
-priv
   specifies that a privileged call is to be made to obtain the status
   of this resource (see "Access Restrictions" below).

-release_lock, -rll
   prints the status of the lock which prevents the owner from
   releasing this resource. In an active function invocation, "true"
   is returned if the lock prevents the owner from releasing the
   resource; "false" is returned otherwise.
-uid
   prints the unique identifier of this resource.

Notes: When invoked as an active function, this command returns the value requested by the specified control argument (only one control argument may be specified in this usage).

Access Restrictions: The use of the -priv control argument requires execute access to the rcp_admin_ gate.

Syntax as active function: [resource_status type name -control_arg]

resource_usage

02/13/76 resource_usage, ru

Syntax: ru -control_arg

Function: prints a report of resource consumption for current billing period.

Control arguments: (One of the following)
-total, -tt
   prints only total dollar figures including the user's dollar limit stop and his month-to-date spending.
-brief, -bf
   prints the information selected by the -total control argument, preceding this information with a header and following it by total dollar figures depicting the user's interactive, absentee, and I/O daemon usage.
-long, -lg
   prints the most comprehensive picture of the user's resource usage.

Notes: If no control argument is specified, all dollar charges are printed but resource usage expressed as time is not printed.

revert_output

08/10/77 revert_output, ro

Syntax: ro -ssw switchname ... -all

Function: reverts the most recent file_output, syn_output, or terminal_output attachment, or all such, for each I/O switch specified.

Control arguments:
-ssw switchname
to specify an I/O switch. If no switchnames are specified, the default is user_output.
-all
reverts all fo, so and to attachments for all switches, or for the switchnames specified.

Examples:
ro -ssw user_output -ssw error_output
reverts both switches.

run

06/22/79 run

Syntax: run -control_args main_program program_args

Function: provides temporary environment for execution of programs.

Arguments:
main_program
   pathname of the main program for the run.
program_args
   arguments passed to the exec_com or main program.

Control arguments:
-exec_com path, -ec path
   specifies the exec_com to be executed.
-no_exec_com, -nec
   invokes the main program directly.
-limit n, -li n
   interrupts run every n seconds of virtual CPU time.
-copy_reference_names, -crn
   starts run with copy of reference names initiated before run and
   automatically terminates segments initiated only with the run unit.
-new_reference_names, -nrn
   uses a different reference name table and automatically terminates
   segments initiated only within the run unit. (DEFAULT)
-old_reference_names, -orn
   uses original reference name table directly and does not automatically
   terminate segments initiated during the run unit.

Notes:
   -crn, -nrn, and -orn are mutually exclusive.
   If neither -exec_com nor -no_exec_com control arguments are given, the exec_com segment main_program.run.ec is searched for in the same directory as the main program. If it is not found, the main program is invoked directly.
   When an exec_com is used, the main program name, if any, is passed as the first argument and the exec_com is responsible for invoking the main program.

runoff

08/30/79 runoff, rf

Syntax: rf paths -control_args

Function: types out text segments in manuscript form.

Arguments:
paths
are pathnames of input segments or multisegment files; the runoff suffix need not be given.

Control arguments:
-ball N, -bl N
convert output to a form suitable for an N typeball.
-character, -ch
create entryname.chars, listing page and line numbers of special characters, normally not printable, that must be drawn in by hand.
-device N, -dv N
prepare output compatible with device N.
-from N, -fm N
start printing at the page numbered N.
-hyphenate, -hph
call user-supplied procedure to perform hyphenation.
-indent N, -in N
set initial indentation to N.
-no_pagination, -npgn
suppress page breaks.
-number, -nb
print source segment line numbers in output.
-page N, -pg N
change the initial page number to N.
-parameter arg, -pm arg
assign arg as a string to the internal variable "Parameter".
-pass N
make N passes over the input.
-segment, -sm
direct output to the segment or multisegment file named entryname.runout, where entryname is the name of the input segment.
-stop, -sp
wait for a carriage return before each page.
-to N
finish printing after the page numbered N.
-wait, -wt
wait for a carriage return before the first page.

Control requests: are defined below. If the request has a default, it is in parentheses following the definition.
The following conventions are used to specify arguments of control requests.
integer constant

```
c      character
cd     character pair
exp    expression (either numeric or string)
n      integer expression
+/-n   +/- indicates update by n; if sign not present, set to n
f      segment name
t      title of the form 'part1'part2'part3'

.ad            right justify text (on)
.ar            arabic page numbers (arabic)
.bp            begin new page
.br            break, begin new line
.cc c          change special character from % to c (%)
.ce n          center next n lines (1)
.ch cd....     note "c" in chars segment as "d"
.ds            double space (off)
.ef # t        defines even footer line #
.eh # t        defines even header line #
.eq N          next N lines are equations (1)
.ex text       call command processor with "text"
.fh t          format of footnote demarcation line (underscore)


.fi            fill output lines (on)
.fo # t        equivalent to--  .ef # t, .of # t
.fr c          controls footnote numbering--  "t" reset each page,
               "f" continuous, "u" suppress numbering
.ft            delimits footnotes
.gb xxx        "go back" to label xxx
.gf xxx        "go forward" to label xxx
.he # t        equivalent to: .eh # t, .oh # t
.if f exp      segment f.runoff inserted at point of request;
               value of "exp" assigned to "Parameter"
.in +/-n       indent left margin n spaces (0)
.la xxx        define label xxx
.li n          next n lines treated as text (1)


.ll +/-n       line length is n (65)
.ma +/-n       equivalent to-- .m1 +/-n, .m4 +/-n (4)
.mp +/-n       print only every nth page (1)
.ms +/-n       multiple space of n lines (1)
.m1 +/-n.      margin above headers set to n (4)
.m2 +/-n       margin between headers and footers set to n (2)
.m3 +/-n       margin between last text line and last footer set to n (2)
.m4 +/-n       margin between first footer and page bottom set to n (4)
.na            does not right justify (off)
.ne n          need n lines; begin new page if not enough remain (1)
.nf            does not fill output lines;
               print them exactly as entered (off)


.of # t        defines odd footer line #
.oh # t        defines odd header line #
.op            next page number is odd
.pa +/-n       begin page n
.pi n          skip n lines if n remain; otherwise
               skip n on next page before any text (1)
.pl +/-n       set page length to n lines
.rd            read one line of text from the user_input I/O switch
```

```
            and process it in place of .rd line
.ro         roman numeral page numbers (arabic)
.rt         "return" from this input segment
.sk n       skip n page numbers (1)
.sp n       space n lines (1)
.sr sym exp assign value of "exp" to variable named "sym"


.ss         single space (on)
.tr cd....  translate nonblank character c into d on output
.ts n       process next input line only if n is not zero (1)
.ty xxx     write "xxx" onto error_output I/O switch
.un n       indent next text line n spaces less (left margin)
.ur text    substitute values of variables in "text",
            and scan line again
.wt         read one line of text from user_input I/O switch and
            discard it
.*          comment line; ignored
.           comment line; ignored, but included in chars
            output segment
```

Built-in symbols: runoff has over 50 internal variables, which are available to the user. In addition, the user can set his own variables with the .sr control request. See the runoff command in the MPM Commands for the list of built-in symbols.

Expressions: can be either arithmetic or string and consist of numbers and operators in appropriate combinations. The operators and order of precedence are--

```
^  (bit-wise negation), -(unary)
*,/,   (remainder)
+,-  (binary)
=, < , >, /, < , >  (all are comparison operators
                     that yield -1 for true or 0 for false)
&  (bit-wise AND)
|  (bit-wise OR), = (bit-wise equivalence)
```

Parentheses for grouping.

The values can have the following forms --

String

```
   <string>=<basicstring> | <concatenation> | <substr>; <basicstring>="xxx"
   <concatenation>=<string><basicstring>; <substr>=<string>(x,y)
   escape sequences - *b,*t,*n,*s,*",**,*cnnn (BS,HT,NL,space,",*, nnn)
```

Arithmetic adecimal number; # followed by octal digits - an octal number; followed by hexadecimal digits - a hexadecimal number

runoff_abs

11/13/81 runoff_abs, rfa

Syntax: rfa paths -rf_args -dp_args -control_args

Function: submits an absentee request to process text segments using the runoff command.

Arguments:
paths
   are the pathnames of segments to be processed by runoff.
rf_args
   are control arguments accepted by the runoff command.
dp_args
   are control arguments (except -delete and -indent) accepted by the dprint command.

Control arguments:
-queue N, -q N
   is the priority queue of the request. The default queue is defined by the system administrator. See the Notes for a description of the interaction with the dprinting of output files.
-hold
   do not dprint or delete any output files.
-output_file path, -of path
   put absentee output in segment path.
-limit N, -li N
   specifies time limit in seconds for the absentee job.

Notes:
Control arguments and paths can be mixed freely and can appear anywhere on the command line after the command.

Unpredictable results can occur if two absentee requests are submitted that simultaneously attempt to compile the same segment or write into the same absout segment.

If the -indent control argument is given to this command, it is interpreted as the runoff control argument; not as the dprint control argument.

If the -queue control argument is not specified, the request is submitted into the default absentee priority queue defined by the site and, if requested, the output files will be dprinted in the default queue of the request type specified on the command line. (If no request type is specified, the "printer" request type is used.)

If the -queue control argument is specified, and, if requested, the output files will be dprinted in the same queue as is used for the absentee request. If the request type specified for dprinting does not

have that queue, the highest numbered queue available for the request type is used and a warning is issued.

runoff_compose.differences

10/13/77 Differences between compose and runoff

This info file gives the differences between runoff and compose.
A "title" response is recommended, followed by a "search" for the section
of interest. The sections are not in any particular order.

Title Delimiter:
In order to implement free format header and footer blocks (allowing artwork and other text features), it is necessary to define a standard title part delimiter character in a manner similar to the Symbol Delimiter. This title part delimiter character is chosen as the vertical bar ("|") and may be changed by the user with the change-title-delimiter control. Any <title> not beginning with the current title delimiter character will be rejected with a diagnostic message.

Builtin Mapping:

```
Ad  ->     AlignMode = "both"
Ce  ->     --
CharsTable ->     ExcepTable
Charsw  ->     ExcepOpt
ConvTable  ->     --
Date  ->     Date
Device  ->     Device
DeviceTable ->     --
Eq  ->     --
Eqcnt  ->     Eqcnt
ExtraMargin ->     ExtraMargin
Fi  ->     FillMode
FileName  ->     FileName
Filesw  ->     OutputFileOpt
Foot  ->     Footcnt
FootRef  ->     --
Fp  ->     --
Fr  ->     FootReset = "r"
From  ->     From
Ft  ->     FootnoteMode
Hyphenating ->     Hyphenating
In  ->     Indent
InputFileName ->     InputFileName
InputLines ->     InputLines
LinesLeft  ->     LinesLeft
Ll  ->     PageWidth
Lp  ->     --
Ma1  ->     VMargTop
Ma2  ->     VMargHeader
Ma3  ->     VMargFooter
Ma4  ->     VMargBottom
Ms  ->     LineSpace
MultiplePagecount ->     PageSpace

NestingDepth ->     InsertIndex
```

```
Nl  ->    PageLine
NNP  ->    NextPageNo
NoFtNo  ->    FootReset = "u"
NoPaging  ->    Galley
Np  ->    PageNo
PadLeft  ->    --
Parameter  ->    Parameter
Passes  ->    Pass
Pi  ->    PictureCount
Pl  ->    PageLength
Print  ->    Print
Printersw  >    OutputFileOpt and
          Device = "ascii"
PrintLineNumbers ->    LineNumberOpt
Roman  ->    --
Selsw  ->    --
SpecCh  ->->    SymbolDelimiter
Start  ->    --
Stopsw  ->    StopOpt
TextRef  ->    --
Time  ->    Time
To  ->    To
TrTable  ->    TrTable
Un  ->    Undent
WaitOpt  ->    Waitsw
```

Symbol Delimiter:
The conventional use of the symbol delimiter (%) as a reference to the page counter has been removed; however, the remainder of the symbol delimiter parsing algorithm is unchanged from the algorithm used in runoff. This means that constructs for nesting and/or concatenation of variable values and literal strings should continue to work as they did for runoff.

File Suffixes:

```
 .runoff -> .compin
 .runout -> .compout
  .chars  -> .compx
```

Control Arguments:

```
     -check, -ck
     -device  name , -dv  name   ("ascii")
     -exception_graphics, -excep
     -from  n , -fm  n   (1)
     -galley  n1  ,n2 , -gl  n1  ,n2   (1,end-of-file)
     -hyphenate  n , -hyph  n , -hph  n   (3)
     -indent  n , -in  n   (0)
     -input_file path, -if path  (path is required)
     -linespace  n , -ls  n   (1)
     -noart, -noa
     -nofill, -nof
     -number, -nb
     -number_brief, -nbb
     -output_file  path , -of  path   ([wd]>input_file.compout)
     -pages  n   n,n , -pgs  n   n,n   (n is required)
     -parameter string, -pm string  (string is required)
     -pass  n   (1)
     -stop, -sp
```

```
      -to  n   (end-of-file)
      -wait, -wt


Control Mapping:
   .ad   .alb    .ar   .srm ar    .bp   .brp
   .br   .brf    .cc   .cdl    .ce   .bbe n
   .ch   .tre    .ds   .ls 2    .ef   .fle
   .eh   .hle    .eq   .bbe n    .ex   .exc
   .fh   .hlf    .fi   .fin    .fo   .fla
   .fr t .ftp    .fr f .ftr    .fr u .ftu
   .ft   .bbf/.bef    .gb   .go    .gf   .go
   .he   .hla    .if   .ifi    .in   .inl
   .la   .la    .li   .bbl n    .ll   .pdw
   .ml   .vmt    .m2   .vmh    .m3   .vmf
   .m4   .vmb    .ma   .vmt/.vmb    .mp   .ps
   .ms   .ls    .na   .all    .ne   .brn
   .nf   .fif    .of   .flo    .oh   .hlo
   .op   .brp o    .pa   .brp n|+ _n    .pi   .bbp n
   .pl   .pdl    .rd   .rd    .ro   .srm ro
   .sk   .brs    .sp   .spb    .sr   .srv
   .ss   .ls 1    .tr   .trf    .ts   .ts
   .ty   .ty    .un   .unl    .ur   .ur
   .wt   .wt
```

save_on_disconnect

03/13/80 save_on_disconnect

Syntax: save_on_disconnect

Function: reverses the effect of the no_save_on_disconnect command, re-enabling process preservation across hangups in the user's process.

Notes: This command is only meaningful if process preservation was in effect for the process at login time, either by default or because the -save_on_disconnect control argument was specified on the login command line.

send_mail

10/27/83 send_mail, sdm [Info Honeywell modifiee CNET].

ATTENTION: Segment d'info Honeywell avec modif CNET.
Cette modif (provenant en fait de l'INRIA) consiste en l'insertion d'une nouvelle section, intitulee "New control arguments", concernant le courrier inter-Multics.

Syntax: sdm addresses -control_args

Function: Send a message to one or more addresses, which are described below. It accepts text from the terminal, and optionally enters a request loop before sending to allow the user to edit or modify the message.

Notes:
In default mode, send_mail prompts "Subject:" and accepts a subject line from the terminal, then prompts "Message:" and accepts the message text. The text is terminated by "." on a line to send the message, " f" to enter the send_mail editor, or " fq" to enter the request loop.

Request lines use () for iteration, "" for quoting, and [] to invoke send_mail active requests.

Arguments: An address is any of the following:
-mailbox path, -mbx path
send to the mailbox specified by path.
-user Person_id.Project_id
send the message to the specified user.
string
If string contains either ">" or "< ", it is interpreted as a mailbox pathname (as in -mbx path); otherwise, it is interpreted as a Person_id.Project_id.

Any address may be followed by the sequence "-comment string", which causes string to be appended to the address in the header as a comment.

Control arguments:
-abort
Do not send message unless it can be sent to all recipients. (DEFAULT)
-acknowledge, -ack
Request an acknowledgement from the recipients.
-brief, -bf
Suppress "Mail delivered to ..." when message is sent.
-cc addresses
Specifies that addresses are secondary recipients of the message and adds them to the cc header field.
-fill
Reformat message text according to "fill-on" and "align-left" mode in compose.

-from addresses
   Specifes that addresses are the authors of the messages and adds them to the From field. (DEFAULT-- the user invoking send_mail)
-header, -he
   Generate a header. See "List of header fields". (DEFAULT)
-in_reply_to string, -irt string
   Add an In-Reply-To field containing string.
-input_file path, -if path
   Take message text from the specified file, rather than from the console.
-line_length N, -ll N
   Cause filling (fill request and -fill control arg) to be done with this line length. (DEFAULT-- 72)

-log
   Send a copy of the message to the user's logbox.
-long, -lg
   Print "Mail delivered to ..." when message is sent. (DEFAULT)
-message_id, -mid
   Add Message-ID field to header.
-no_abort
   Send message even it it can't be sent to all recipients.
-no_acknowledge, -nack
   Do not request an acknowledgement. (DEFAULT)
-no_fill, -nfi
   Do not reformat the message text. (DEFAULT)

-no_header, -nhe
   Add only those header fields to the message explicitly requested by the user.
-no_log
   Do not send a copy of the message to the user's logbox. (DEFAULT)
-no_message_id, -nmid
   Do not add a Message-ID field to the header. (DEFAULT)
-no_prompt
   Do not prompt the request loop. (DEFAULT-- prompt with "send_mail":)

-no_request_loop, -nrql
   Send the message immediately without entering the request loop. (DEAFULT)
-no_subject, -nsj
   Do not prompt for a Subject field. (DEFAULT-- prompt)
-prompt string
   Set the request loop prompt to string.
-reply_to addresses, -rpt addresses
   Add addresses to the Reply-To field.
-request string, -rq string
   Execute the requests in string after reading message text.

-request_loop, -rql
   Enters the request loop before sending the message.
-save path, -sv path
   Send a copy of the message to path.sv.mbx.
-subject string, -sj string
   Sets the subject of the message to string. (DEFAULT-- prompt for subject)
-terminal_input, -ti
   Prompt "Message:" and reads the text. (DEFAULT)

-to addresses
   Specifes that addresses are primary recipients of the message and adds these addresses to the To header field.

New control arguments:
-at host_id
   recipient (identified in the immediately preceding argument) resides on specified host. The host_id argument may be the standard host name, host abbreviation, or decimal host number. This control argument must follow a user name and may be used to temporarily override the -local or the -host control arguments.
   Available host_ids at CNET: IRIA for Rocquencourt, CICB for Rennes, CICG for Grenoble, CICT for Toulouse, CCVR for Cray-one.
-host host_id addrs
   recipients specified by addrs are resident on the specified DSA host. Each addr argument represents a user and must meet the requirements of a user name on the respective host. The host_id argument may be the standard host name, host abbreviation, or decimal host number.
-local addrs
   recipients specified by addrs are resident on the local Multics host. Each addr argument can be specified as either a Multics pathname or a User_id (Person_id.Project_id).

List of requests:
   This list lists only the most important features of each request. For more detailed information, use the send_mail request "help request_name" to see the info file on a particular request.

? List the available send_mail requests.
. Identify send_mail with version number, recursion level, and message info.
..line
   Execute line as a Multics command without further processing by the send_mail request processor.
help string
   Print information about send_mail requests or topics. For a list of topics, use the 'help *' request.

list, ls
   Print a message summary with lines, subject, and destinations.
print -control_arg , pr -control_arg
   Print the message. -control_arg can be -header (-he) or -no_header (-nhe).
quit -control_arg , q -control_arg
   Exit send_mail. -control_arg can be -force (-fc), which causes send_mail to be exited even if the message was modified since last sent.
send addresses -control_args
   Send message to addresses or to primary and secondary recipients if no addresses. -control_args can be -abort, -acknowledge (-ack), -brief (-bf), -header (-he), -long (-lg), -message_id, (-mid), -no_abort, -no_acknowledge (-nack), -no_header (-nhe), -no_message_id (-nmid), -no_notify (-nnt), or -notify (-nt) to override command line options.

fill -control_arg , fi -control_arg
   Reformat message text as in compose "fill-on" and "align-left"" mode.

-control_arg can be -line_length N (-ll N) to specify a line length for this request.

qedx -control_arg , qx -control_arg
Invoke qedx editor on the message. -control_arg be either -header (-he) or -no_header (-nhe).

append path
Write the message at the end of the ASCII segment, path.mail.

copy path, cp path
Copy the message to the mailbox, path.mbx.

log
Save the message in the user's logbox.

preface path
Write the message at the front of the ASCII segment, path.mail.

save path, sv path
Save the message in the savebox, path.sv.mbx.

write path -control_arg , w path -control_arg
Write the message to the end of the ASCII segment, path.mail.
-control_arg can be -truncate (-tc) or -extend (-ex).

apply -control_arg strings, ap -control_arg strings
Put the message into a temporary segment, concatenate all the strings and the pathname, and pass the result to the Multics command processor.
-control_arg can be -header (-he) or -no_header (-nhe)

execute strings, e strings
Execute strings as a Multics command line after evaluating send_mail active requests. As an active request, return the result of evaluating strings as a Multics active string.

HEADER REQUESTS

These requests (except for remove) print the contents of the specified header fields if invoked with no arguments.

cc addresses
Add addresses to the secondary recipients and cc field.

from addresses
Add addresses to the list of authors and From field.

in_reply_to string , irt string
Put string in the In-Reply-To field.

message_id, mid
Print Message-ID field, creating it if necessary.

remove addresses -control_args
Delete addresses from specified header fields. -control_args can be -cc, -from, -reply_to and -to and affect the following addresses.

reply_to addresses , rpt addresses
Add addresses to the Reply-To field.

subject strings , sj strings
Set the subject of the messages to strings. As an active request, returns the subject, re-quoted.

to addresses
Add addresses to the primary recipients and To field.

List of header fields:

Redistributed-Date, Redistributed-By, and Redistributed-To
   specify info about the forwarding of the message.
Date (required)
   shows the date and time the message was sent.
From (required)
   contains the list of authors. (DEFAULT-- user of send_mail command)
Subject
   describes the message contents. Supplied by the user.

Sender
   shows the actual sender of the message if different from From.
Reply-To
   lists addresses to which a reply should be sent.
To lists the primary recipients.
cc lists the secondary recipients.

Acknowledge-To
   gives an acknowledgement address if acknowledgement was requested. The
   presence of the field requests acknowledgement.
In-Reply-To
   describes the message to which this one is a reply, if any.
Message-ID
   contains a unique character string identifier from send_mail.

send_message

06/30/80 send_message, sm

Syntax: sm Person_id.Project_id message
or:
sm -pathname path message

Function: sends messages (one or more, always sent one line at a time) to a given user on a given project.

Arguments:
Person_id
is the registered name of the recipient.
Project_id
is the name of the recipient's project.
message
is an optional string. If message is missing from the command line, send_message types "Input." and accepts lines that it sends, one line at a time, with each newline character. In this case, input is terminated by a line consisting solely of a period.

Control arguments:
-pathname path, -pn path
causes messages to be sent to a mailbox specified by pathname. The mbx suffix is assumed.

Notes: For a description of the mailbox, refer to accept_messages and print_mail.

send_message_acknowledge

06/17/81 send_message_acknowledge, sma

Syntax: sma Person_id.Project_id message
or: sma -pn PATH message

Function: operates like the send_message command and requests that the recipient's process return an acknowledgement when the message is read.

Arguments:
Person_id
registered name of the recipient.
Project_id
name of the recipient's project.
message
string up to 132 characters; if omitted, send_message_acknowledge types "Input." and accepts lines that it sends, one at a time, with each newline character. In this case, input is terminated by a line consisting of a period.

Control arguments:
-pathname PATH, -pn PATH
specifies a mailbox pathname.

Access required: append and wakeup extended access on a mailbox in order to send an interactive message.

Notes: Parentheses, quotes, brackets, and semicolons in the command line have their usual command language interpretation.

send_message_express

06/17/81 send_message_express, smx

Syntax: smx Person_id.Project_id message
or: smx -pn PATH message

Function: operates like send_message, but adds message to the recipient's mailbox only if the message will be printed immediately (i.e., if the recipient is currently accepting messages).

Arguments:
Person_id
registered name of the recipient.
Project_id
name of the recipient's project.
message
string up to 132 characters; if omitted, send_message_express types "Input." and accepts lines that it sends, one at a time, with each newline character. In this case, input is terminated by a line consisting of a period.

Control arguments:
-pathname PATH, -pn PATH
specifies the pathname of a mailbox.

Access required: append and wakeup extended access on a mailbox in order to send an interactive message.

Notes: Parentheses, quotes, brackets, and semicolons in the command line have their usual command language interpretation.

send_message_silent

06/17/81 send_message_silent, sms

Syntax: sms Person_id.Project_id message
or: sms -pn PATH message

Function: operates like the send_message command but does not print an error message if the message cannot be sent or will not be received immediately.

Arguments:
Person_id
registered name of the recipient.
Project_id
name of the recipient's project.
message
string up to 132 characters; if omitted, send_message_silent types "Input." and accepts lines that it sends, one at a time, with each newline character. In this case, input is terminated by a line consisting of a period.

Control arguments:
-pathname PATH, -pn PATH
specifies the pathname of a mailbox.

Access required: append and wakeup extended access on a mailbox in order to send an interactive message.

Notes: Parentheses, quotes, brackets, and semicolons in the command line have their usual command language interpretation.

set_acl

12/22/80 set_acl, sa

Syntax:
sa path model User_il ... modeN User_idN -control_args

Function: manipulates the access control lists (ACLs) of segments, multisegment files, and directories. See "Access Control" in the MPM Reference Guide for a discussion of ACLs.

Arguments:
path
is the pathname of a segment, multisegment file, or directory. If it is -wd or -working_dir, the working directory is assumed. The star convention can be used and applies to either segments and multisegment files or directories, depending on the type of mode specified in modeN.
modeN
is a valid access mode. For segments or multisegment files, any or all of the letters rew; for directories, any or all of the letters sma with the requirement that if modify is present, status must also be present. Use null, "n" or "" to specify null access.
User_idN
is an access control name that must be of the form Person_id.Project_id.tag. All ACL entries with matching names receive the mode modeN. If no match is found and all three components are present, an entry is added to the ACL. If the last modeN has no User_id following it, the Person_id of the user and current Project_id are assumed.

Control arguments:
-chase
causes links to be chased when using the star convention. (Links are always chased when path is not a starname.)
-no_chase
causes links to not be chased when using the star convention. This is the default.
-brief, -bf
suppresses error messages of the form "No match for User_id on ACL of <path>", where User_id does not specify all components.
-no_sysdaemon, -nsd
suppresses the addition of a "rw *.SysDaemon.*" term when using -replace.

-replace, -rp
deletes all ACL terms (with the exception of a default "rw *.SysDaemon.*" term unless -no_sysdaemon is specified) before adding the terms specified on the command line. The default is to add to and modify the existing ACL.
-sysdaemon, -sd
when -replace is specified, adds a "rw *.SysDaemon.*" ACL term

before adding the terms specified. (Default)

Either of the following control arguments can be specified to resolve an ambiguous choice between segments and directories that occur only when modeN is null and the star convention is used in path--

-directory, -dr
specifies that only directories are affected.

-segment, -sm
specifies that only segments and multisegment files are affected. This is the default.

Access required: The user needs modify permission on the containing directory.

Notes: The arguments are processed from left to right. Therefore, the effect of a particular pair of arguments can be changed by a later pair of arguments.

The strategy for matching an access control name argument is defined by three rules--

1) A literal component, including "*", matches only a component of the same name.

2) A missing component not delimited by a period is treated the same as a literal "*" (e.g., "*.Multics" is treated as "*.Multics.*"). Missing components on the left must be delimited by periods.

3) A missing component delimited by a period matches any component.

set_fortran_common

10/25/77 set_fortran_common, sfc

Syntax: sfc paths -control_arg

Function: initializes FORTRAN common blocks.

Arguments:
paths
are pathnames of FORTRAN object segments containing references to desired common blocks.

Control arguments:
-long, -lg
prints a warning whan an already allocated common block is smaller than one encountered in the list of object segments.

set_iacl_dir

08/30/79 set_iacl_dir, sid

Syntax: sid path model User_id1 ... modeN User_idN -control_arg

Function: adds entries to a directory initial ACL or modifies the access mode in a directory initial ACL entry.

Arguments:
path
directory in which the initial ACL should be changed; can be -working_dir or -wd. The star convention can be used.
modeN
mode associated with User_idN. It can be any or all of the letters sma, or null, n, or "" for null access. If m is given, s must also be given.
User_idN
access control name of the form Person_id.Project_id.tag.

Control arguments:
-ring N, -rg N
identifies ring number (default is current ring). It can appear anywhere on the line, except between a mode and its associated User_id, and affects the whole line.

Notes: Type "help acl_matching" for the User_id matching strategy.

set_iacl_seg

08/30/79 set_iacl_seg, sis

Syntax: sis path model User_id1 ... modeN User_idN -control_arg

Function: adds entries to a segment initial access control list (initial ACL) in a directory or modifies the access mode in an existing segment initial ACL entry.

Arguments:

path
directory in which the segment initial ACL should be changed; -wd or -working_dir for the working directory. The star convention can be used.

modeN
mode associated with User_idN; it can consist of any or all of the letters rew, or null, n, or "" for null access.

User_idN
access control name of the form Person_id.Project_id.tag.

Control arguments:

-ring N, -rg N
identifies ring number (default is current ring). It can appear anywhere on the line except between a mode and its associated User_id, and affects the whole line.

Notes: Type "help acl_matching" for the User_id matching strategy.

set_resource

04/24/81 set_resource, setr

Syntax: set_resource type STR1 ... STRn -control_args

Function: The set_resource command is used to modify parameters of a resource.

Arguments:

type
is a resource type defined in the resource type description table (RTDT).

STRi
is the unique identifying name of the particular resource being modified. If STR looks like a control argument (i.e., if it is preceded by a hyphen), then it must be preceded by -name or -nm.

Control arguments:

-access_class accr, -acc accr
sets the initial AIM access class parameters, where accr is the access class range. Users at any authorization within the access class range inclusive are allowed to read and write to the resource (provided they also meet other access requirements).

-acs_path path
specifies the pathname of the access control segment (ACS) for this resource. The ACS is not created by this command, but must be created by the accounting owner, and the desired access control list set (see "Notes" below). If this control argument is not given, the accounting owner of the resource is given rew access by default.

-alloc STR
sets the allocation state of the resource to free or allocated, where STR must be either "on" or "off". If this control argument is not given, the allocation state is free. on sets the allocation state to allocated; off sets the allocation state to free.

-attributes STR, -attr STR
specifies the desired values for the attributes of this resource (see "Notes" below).

-charge_type name, -crgtp name
specifies the name of the billing algorithm used to account for the use of this resource.

-comment STR, -com STR
specifies the desired value of the comment string for this resource.

-location STR, -loc STR
specifies a descriptive location for the resource, to aid the operator in locating it when it is stored in a special place (e.g., a vault, a different room, etc.).

-lock STR
locks or unlocks the resource, preventing or allowing use of that

resource, where STR must be either "on" or "off". If this control argument is not specified, the lock is off. on prevents any use of the resource; off allows use of the resource.

-priv
specifies that a privileged call is to be made to obtain the status of this resource (see "Access Restrictions" below).

-release_lock STR, -rll STR
specifies whether this resource may be released by the owner, or may only be released by a privileged process (see "Access Restrictions" below). If this control argument is not specified, the resource may be released by the owner (does not require special privilege). on resources may only be released by privileged process; off resources may be released by owner.

Notes: If multiple resources are specified to the set_resource command and an error occurs in the modification of one of these resources, none of the resources specified are modified.

Access Restrictions: The user must have write effective access to the resource named to perform any modification on the status of the resource. In addition, the user must have execute effective access to the resource named to modify protected attributes. Only the accounting owner may modify the ACS path. The user must have execute access to the rcp_admin_ gate in order to use the -access_class, -release_lock, -location, -charge_type, or -lock control arguments.

set_search_paths

07/08/80 set_search_paths, ssp

Syntax: ssp search_list search_paths -control_arg

Function: allows a user to replace the search paths contained in a specified search list.

Arguments:

search_list
   is the name of a search list. If this search list does not exist, it is created. A warning message is printed if a search list is created and it is not system defined.

search_paths
   are search paths to be added to the specified search list. The search paths are added in the order in which they are specified in the command line. The search path can be an absolute or relative pathname or a keyword. (For a list of acceptable keywords see add_search_paths in Commands and Active Functions, AG92.) If no search paths are specified, then the specified search list is set as if it were being initialized for the first time in the user's process.

Control arguments:

-brief, -bf
   suppresses a warning message for the creation of a search list not defined by the system.

-default, -df
   replaces the search list with its system-defined default. No search_paths can be specified with this control argument.

Notes: The specified search list is replaced by the specified search paths. It is an error to create a new empty search list.

For a complete list of the search facility commands, see the add_search_paths command description in Commands and Active Functions, AG92.

set_search_rules

10/08/80 set_search_rules, ssr

Syntax: set_search_rules path -control_arg

Function: sets the dynamic linking search rules of the user to suit individual needs with only minor restrictions.

Arguments:
path
is the pathname of a segment containing the ASCII representation of search rules. Search rules are absolute pathnames and any of the keywords listed below in "List of Keywords", one search rule per line. If path is not specified, the search rules must be reset to the default search rules by the -default control argument.

Control arguments:
-default, -df
resets the search rules to the default search rules, as set for a new process.

List of keywords:
initiated_segments
checks the already initiated segments.
referencing_dir
searches the containing directory of the segment making the reference.
working_dir
searches the working directory.
home_dir
searches the home directory.
process_dir
searches the process directory.
site-defined
expand into one or more directory pathnames. (An example of a site_defined keyword is system_libraries.) See the get_system_search_rules command for an explanation of the values of these keywords. The "default" keyword can be used to obtain the site-defined default rules.

Notes: A maximum of 21 rules is allowed. Leading and trailing blanks are allowed, but embedded blanks are not allowed.

If the user decides not to include the system libraries in the search rules, many standard commands cannot be found.

See also the descriptions of the print_search_rules, get_system_search_rules, add_search_rules, and delete_search_rules commands.

set_tty

01/11/82 set_tty, stty

Syntax: stty -control_args

Function: modifies the terminal type associated with the user's terminal and/or various parameters associated with terminal I/O. The type as specified by this command determines character conversion and delay timings; it has no effect on communications line control.

Control arguments:

-all, -a
   is the equivalent of specifying the four control arguments -print, -print_edit, -print_frame, and -print_delay.

-buffer_size N, -bsize N
   specifies the terminal's buffer size to be used for output block acknowledgement where N is the terminal's buffer size in characters.

-brief, -bf
   may only be used with the -print control argument and causes only those modes that are on plus those that are not on/off type modes (e.g., 1179) to be printed.

-delay STR, -dly STR
   sets the delay timings for the terminal according to STR, which is either the word "default" or a string of six decimal values separated by commas. If "default" is specified, the default values for the current terminal type and baud rate are used. The values specify vert_nl, horz_nl, const_tab, var_tab, backspace, and vt_ff, in that order. (See "List of delay types" below.)

-edit edit_chars, -ed edit_chars
   changes the input editing characters to those specified by edit_chars. The edit_chars control argument is a 2-character string consisting of the erase character and the kill character, in that order. If the erase character is specified as a blank, the erase character is not changed; if the kill character is omitted or specified as a blank, the kill character is not changed.

-initial_string, -istr
   transmits the initial string defined for the terminal type to the terminal.

-input_flow_control STR, -ifc STR
   sets the input_suspend and input_resume characters to those specified in STR, which is a string of one or two characters. If STR contains two characters, the first character is the input_suspend character and the second one is the input_resume character. If STR contains only one character, it is the input_resume character and there is no input_suspend character.

-io_switch STR, -is STR
   specifies that the command be applied to the I/O switch whose name is STR. If this control argument is omitted, the user_i/o switch is assumed.

-modes STR
   sets the modes for terminal I/O according to STR, which is a string of mode names separated by commas. Many modes can be optionally preceded by "^" to turn the specified mode off. Modes not specified in STR are left unchanged. For a list of valid mode names, type:
      help tty_modes.gi
-output_etb_ack STR, -oea STR
   sets the output_end_of_block and output_acknowledge characters to those specified in STR, which is a string of two characters. The first character of STR is the end_of_block character and the second one is the acknowledge character.

-output_suspend_resume STR, -osr STR
   sets the output_suspend and output_resume characters to those specified in STR, which is a string of two characters. The first character of STR is the output_suspend character and the second is the output_resume character.
-print, -pr
   prints the terminal type and modes on the terminal. If any other control arguments are specified, the type and modes printed reflect the result of the command.
-print_delay, -pr_dly
   prints the delay timings for the terminal.
-print_edit, -pr_ed
   prints the input-editing characters for the terminal.

-reset, -rs
   sets the modes to the default modes string for the current terminal type.
-terminal_type STR, -ttp STR
   sets the terminal type of the user to STR, where STR can be any one of the types defined in the terminal type table (TTT). The default modes for the new terminal type are turned on and the initial string for the terminal type, if any, is transmitted to the terminal. Refer to the print_terminal_types command for information on obtaining a list of terminal types currently in the TTT.

-frame STR, -fr STR
   changes the framing characters used in blk_xfer mode to those specified by STR, where STR is a 2-character string consisting of the frame-begin and the frame-end character, respectively. These characters must be specified in the character code of the terminal, and may be entered as octal escapes, if necessary. The frame-begin character is specified as a NUL character to indicate that there is no frame-begin character; the same is true for a frame-end character. These characters have no effect unless blk_xfer mode is on. It is an error to set the frame-end character to NUL if the frame-begin character is not also set to NUL.
-print_frame, -pr_fr
   prints the framing characters for the terminal.

List of delay types:
vert_nl
   is the number of delay characters to be output for all newlines to

allow for the linefeed (-127 <= vert_nl <= 127). If it is negative, its absolute value is the minimum number of characters that must be transmitted between two linefeeds (for a device such as a TermiNet 1200).

horz_nl
is a number to be multiplied by the column position to obtain the number of delays to be added for the carriage return portion of a newline (0 <= horz_nl <= 1).

const_tab
is the constant portion of the number of delays associated with any horizontal tab character (0 <= const_tab <= 127).

var_tab
is the number of additional delays associated with a horizontal tab for each column traversed (0 <= var_tab <= 1).

backspace
is the number of delays to be output following a backspace character (-127 <= backspace <= 127). If it is negative, its absolute value is the number of delays to be output with the first backspace of a series only (or a single backspace).

vt_ff
is the number of delays to be output following a vertical tab or formfeed (0 <= vt_ff <= 511).

set_tty.gi

11/04/83 Les options du stty

Vous trouverez ci-dessous une liste de 28 arguments de la commande stty
(ou : set_tty) pour lesquels il existe une info. en francais.
Ces "arguments" s'emploient :
- les uns comme "arguments de controle" proprement dits,
  par exemple : stty -edit <edit_chars>
- les autres comme des "modes" accompagnant l'argument de
  controle "-modes",
  par exemple : stty -modes ^capo,crecho

Pour acceder a cette info, veuillez taper la commande indiquee.
En general, cette commande est de la forme : help nom_de_l'argument
par exemple : help can
ou bien : help echoplex
etc...

LISTE DES ARGUMENTS - can,capo,crecho,ctl_char,default:

can	: Superposition de caracteres.	Commandez	: help can
capo	: Majuscules et minuscules.	"	: help capo
crecho	: Echo [CR] en reponse a [LF].	"	: help crecho
ctl_char	: Caracteres de commande ASCII.	"	: help ctl_char
default	: Groupe de six modes implicites.	"	: help defaut

LISTE DES ARGUMENTS (SUITE) - delay,echoplex,edit,edited,erkl:

delay (dly)	: Valeur des delais.	Commandez	: help delay
echoplex	: Echo.	"	: help echoplex
edit (ed)	: Caracteres d'effacement.	"	: help edit_chars
edited	: Caracteres inconnus du terminal.	"	: help edited
erkl	: Validation des carac. d'effacemt.	"	: help erkl

LISTE DES ARGUMENTS (SUITE) - esc,frame,hndlquit,iflow,lfecho:

esc	: Validation du caractere " ".	Commandez	: help esc
frame (fr)	: Debut et fin de trame.	"	: help framing_chars.
hndlquit	: Comportement en cas de break.	"	: help hndlquit
iflow	: Controle de flux en entree.	"	: help iflow
lfecho	: Echo [LF] en reponse a [CR].	"	: help lfecho

LISTE DES ARGUMENTS (SUITE) - ll,oflow,pl,polite,prefixnl:

ll	: Longueur de la ligne.	Commandez	: help ligne
oflow	: Controle de flux en sortie.	"	: help oflow
pl	: Longueur de la page.	"	: help pl
polite	: Interruption d'entree par sortie.	"	: help polite
prefixnl	: Position sur interruption.	"	: help prefixnl

LISTE DES ARGUMENTS (SUITE) - rawi,rawo,red,replay,reset:

rawi	: Traitement en entree.	Commandez	: help rawi
rawo	: Traitement en sortie.	"	: help rawo
red	: Couleur du ruban.	"	: help ruban

replay	: Reimpression apres interruption.	"	: help replay
reset (rs)	: Retour aux modes normaux.	"	: help reset

LISTE DES ARGUMENTS (SUITE) - scroll,tabecho,tabs:

scroll	: Controle de fin de page.	Commandez	: help scroll
tabecho	: Tabulations en entree.	"	: help tabecho
tabs	: Tabulations en sortie.	"	: help tabs

short_message_format

05/26/77 short_message_format, smf

Syntax: smf -pn mbx_path

Function: causes messages from send_message to be printed in short format. Successive messages from the same sender are preceded by := instead of by a header giving the sender's name.

sort

03/29/76 sort, merge

Syntax:
```
     sort  input_specs  output_spec   -control_args
     merge input_specs  output_spec   -control_args
```

Function: Sort (Merge) one or more files
according to the values of one or more key fields.

This info file applies to both the sort and the merge;
see the section "Differences Between Sort and Merge".

Arguments:
input_specs
   Specify each input file (up to 10) as -
     -if pathname Pathname in Storage System, or
     -ids "attach_desc" Attach description.

output_spec
   Specify just one output file as -
     -of pathname Pathname in Storage System, or
     -of -replace Replace input file by output file, or
     -ods "attach_desc" Attach description.

Control arguments:
     -ci Sort (Merge) Description input via terminal, or
     -sd pathname Pathname of Sort Description (sort command only), or
     -md pathname Pathname of Merge Description (merge command only).
     -td pathname Pathname of directory to contain work files;
  default is user's process directory (sort command only).
     -file_size f Estimated total amount of data to be sorted,
  in millions of bytes (sort command only).

Examples:
1) sort -ci -if in -of -rp Sort Description from user's terminal;
   input file is named in;
   output file will replace input file.
Input.   The Sort requests the Sort Description.
keys: char(10) 0; The single key is a character string
   whose length is 10 bytes,
   and which starts at the first byte
   of the record (word 0, bit 0).
.   A line consisting of "." terminates
   the Sort Description from a terminal.

2) sort -sd sort_desc -td >udd>pool
    Sort Description entered from a segment;
   work files will be contained in the
   directory >udd>pool;
   no input or output file is named.

```
     Assume the segment sort_desc contains -
key: bin(17) 1;  The key is fixed binary aligned, and
   occupies the second word of the record.
exits: input_file  user$input Input_file exit procedure is user$input.
    output_file user$output; Output_file exit procedure is user$output.

3)  sort -ids "tape_ansi_ V" -ods "record_stream_ -target vfile_ b" -ci
   Input file specified by attach description
   for a magnetic tape in ANSI format;
   output file specified to be unstructured
   (the Sort's record output will be
   transformed into stream output).

4)  merge -md merge_desc -if a.in -if b.in -of =.out
     Merge Description entered from a segment;
     input files are named a.in and b.in;
     output file will be named b.out.


Syntax of the Sort (Merge) Description:
     keys:  <key_description> ... ;
     exits:  <exit_description>  ... ;
If -ci is used, the additional line "." terminates input.
There may be up to 32 keys described.


Syntax of a Key Description:
     <datatype> (<size>) <word_offset> [(<bit_offset>)] [descending]
where -
     <datatype> Data type of a key field; can be -
  char, bit, fixed bin, float bin, dec, float dec.
     <size> Length of the key field (in decimal),
  in units appropriate to the data type.
     <word_offset> Offset in words from the beginning of the record.
  Words are numbered (in decimal) starting at 0.
     <bit_offset> Offset in bits from the beginning of the word.
  Bits are numbered (in decimal) starting at 0.
     descending Rank in descending order for this key field.


Syntax of an Exit Description:
     <exit_name>  <user_name>
where -
     <exit_name> Name of the exit point -
 input_file (sort command only)
 output_file (sort command only)
 compare
 input_record (sort command only
 output_record

      <user_name> Name of the entry point of the user procedure,
  in the same form as a command name.


Writing an Exit Procedure:
See the MPM descriptions of the subroutines sort_ and merge_
for a complete description of how to write a user exit procedure;
or type "help sort_" or "help merge_" for a summary.
```

Functions:
Sort or merge one or more files of records which are not ordered,
to create a new file of ordered (or "ranked") records.

Files Supported:
An input or output file can be specified
either by a pathname or by an attach description.
Its organization must be structured (record I/O is used).
Records can be either fixed length or variable length.
If the user names an input file or an output file,
it must be in the Multics Storage System.
(It can be either a segment or a multisegment file.)
If the user supplies an attach description,
any I/O module available at the installation can be used,
provided it supports sequential record I/O.

Input: The user can specify up to 10 input files.
The organization can be either sequential or indexed.
Alternatively, the user must name an input_file exit procedure,
which is then responsible for releasing records to the Sort.

Output: The user can specify one output file.
The organization must be sequential.
Alternatively, the user must name an output_file exit procedure,
which is then responsible for retrieving records
(ranked by the Sort) from the Sort.

Sort Description:
In addition to the arguments to the sort or merge command,
a Sort (Merge) Description is necessary
to specialize the Sort (Merge) for a particular execution.
It can be supplied either via the user's terminal, or via a segment.
A Sort Description can include the following statements -
keys Specifies key fields, used for ranking records.
exits Names user-written exit procedures.

Keys Statement: Up to 32 key fields can be specified.
Use any PL/I data type - except varying string, complex, or pictured.
Ordering can be ascending, descending, or mixed.
The original order of records with equal keys is preserved.

If key fields are not described via the keys statement,
then the user must name a compare exit procedure.

Exits Statement: User-written "exit procedures" can be supplied
at specific points in the sorting process.
The following exits are supported:
input_file Reads input file, releases records to the sort.
output_file Receives records in ranked order, writes output file.
compare Compares two records, decides which ranks first.
input_record Process each input record (delete, insert, or alter).
output_record Process each output record (delete, insert, alter,
or summarize data).

Differences Between Sort and Merge:
The merge command has the following restrictions -
1) -replace cannot be used for the output file.
2) -td and -file_size cannot be specified.
3) The following exit points are not provided -
 input_file
 output_file
 input_record

sort_seg

03/01/76 sort_seg, ss

Syntax: ss path -control_args

Function: orders the contents of a segment according to the ASCII collating sequence.

Arguments:
path
   pathname of input segment.

Control arguments:
-segment path, -sm path
   places sorted units in a segment whose pathname is path; incompatible with the -replace control argument.
-replace, -rp
   replaces original contents of input segment with the sorted units; this is the default.
-unique, -uq
   deletes duplicate sort units from the sorted results; default is to retain duplicated units.
-delimiter STR, -dm STR
   uses STR concatenated with a newline character as the string delimiter; default is a single newline character.
-block N, -bk N
   makes the sort unit a block of N strings where N must be a positive integer; default for N is 1.
-descending, -dsc
   makes the sort in descending order, according to the ASCII collating sequence; incompatible with the -ascending control argument.
-ascending, -asc
   makes the sort in ascending order, according to the ASCII collating sequence; this is the default.
-field field_spec, -fl field_spec
   specifies field (or fields) when sorting within a sort unit (see "Notes").
-ordered_field field_spec, -ofl field_spec
   specifies mixed ascending and descending fields. (see "Notes")
-all, -a
   entire sort unit is considered when sorting; this is the default.

Notes: The field_spec of the -field control argument is a pair of field specifications, S and L. S is the start position of the field in the sort unit (i.e., S is 1 if the field begins at the first character). L is the length of the field, in characters. Both S and L must be positive integers.
The first pair, S1 L1, defines the primary sort field, the second pair, S2 L2, defines the secondary sort field; and so forth. The use of -field control argument is incompatible with the use of the -all control argument.

The field_spec of the -ordered_field control argument is given in threes, S,

L, and O. S and L are the same as for the -field control argument. O is either the string "asc" for an ascending field or "dsc" for a descending field. Use of -ordered_field control argument is incompatible with the -ascending, -descending, and -field control arguments.

start

02/17/76 start, sr

Syntax: sr -control_arg

Function: resumes execution of the user's process from the point of interruption after a signal has suspended execution. It restores the attachments of the user_input, user_output, and error_output I/O switches, and the mode of user_io to their values at the time of interruption.

Control arguments:
-no_restore, -nr
   does not restore the standard I/O attachments.

Notes: The start command can be used to resume execution after an unclaimed signal, if the condition that caused the unclaimed signal either is innocuous or has been corrected.

The start command can be issued after a quit signal.

The release command discards the machine conditions for a suspended execution instead of restarting them.

status

01/29/81 status, st

Syntax: st paths -control_args

Function: prints selected detailed status information about specified storage system entries.

Arguments:
paths
   are the pathnames of segments, directories, multisegment files, and links for which status information is desired. The default pathname is the working directory, which can also be specified by -wd or -working_directory. The star convention can be used.

Control arguments: The following control arguments can be used with any type of entry, and can appear anywhere on the line after the command name and are in effect for the whole line.
-author, -at
   prints the author of the entry.
-chase
   prints information about the branch targets of links instead of the links themselves. An error occurs for a null link or a link to a null link.
-chase_if_possible, -cip
   prints information about the targets of links where branch targets exist, and for null links and links to null links prints information about the ultimate link in the chain. This control argument does not affect the processing of non_links.

-date, -dt
   prints all the relevant dates on the entry.
-date_time_dumped, -dtd
   prints the date-time-dumped by the hierarchy dumper.
-date_time_entry_modified, -dtem
   prints the date-time-entry-modified.
-date_time_used, -dtu
   prints the date-time-used.
-date_time_volume_dumped, -dtvd
   prints the date-time-dumped by the volume dumper.
-directory, -dr
   selects directories when using the star convention.
-link, -lk
   selects links when using the star convention.

-name, -nm
   prints all the names on the entry.
-no_chase
   prints link information about links. (Default)
-no_chase_if_possible, -ncip

prints link information about links. (Default)
-primary, -pri
prints the primary name on the entry.
-segment, -sm
selects segments when using the star convention.
-type, -tp
prints the type of entry: segment, directory, multisegment file, or link.

List of type specific control arguments: The following control_args can only be used for segments, multisegment files, and directories.
-access, -ac
prints the user's effective mode, ring brackets, access class (if different from the default), and safety switch (if it is on).
-access_class
prints the access class.
-bit_count, -bc
prints the bit count.
-bc_author, -bca
prints the bit count author of the entry.
-copy_switch, -csw
prints whether the copy switch is on or off.
-current_length, -cl
prints the current length in pages.

-damaged_switch, -dsw
prints whether the damaged switch is on or off.
-date, -dt
prints all the dates on the entry: i.e., date used, date contents modified, date branch modified, date dumped.
-date_time_contents_modified, -dtcm
prints the date-time-contents-modified.
-device, -dv
prints the logical volume on which the entry resides.

-length, -ln
for segments: prints the bit count, the number of records used, the current blocks (if different from records used), and the maximum length in words;
for multisegment files: prints the number of records used by the whole file, the sum of the bit counts of all components, and the number of components;
for directories: prints the number of records used and the bit count.
-logical_volume, -lv
prints the logical volume on which the entry resides. This control argument is the same as the -device control argument.

-long, -lg
prints all relevant information about the object.
-max_length, -ml
prints the maximum length of a segment.
-mode, -md
prints the user's effective mode.
-records, -rec
prints the records used.

-ring_brackets, -rb
   prints the ring brackets.
-safety_switch, -ssw
   prints whether the safety switch is on or off.
-unique_id, -uid
   prints the entry's unique identifier.

List of control arguments for segments:
-comp_volume_dump_switch, -cvds
   prints whether the complete volume dump switch is on or off.
-incr_volume_dump_switch, -ivds
   prints whether the incremental volume dump switch is on or off.
-usage_count, -use
   prints the number of page faults taken on the segment since
   creation.

List of control arguments for links:
-link_path, -lp
   prints the target pathname.
-long, -lg
   prints all relevant information about the link.

Notes: If no control argument is specified, the following information is printed for segments, multisegment files, and directories-- names, type, date used, date modified, date branch modified, bit count, records used, user's mode, access class.

If no control argument is specified, the following information is printed for links-- the pathname of the entry linked to, names, date link modified, date dumped. The -mode, -device, and -length control arguments are ignored for links.

Zero-valued dates (i.e., dates that have never been set) are not printed. In addition, attributes in the default state are not printed.

Syntax as active function: [st path -control_args]

switch_off

01/15/81 switch_off, swf

Syntax: swf keyword paths -control_args

Function: turns off a specified switch for one or more entries. For an MSF, the switch of the MSF directory (when possible) and those of all the components are turned off.

Arguments:
keyword
   specifies the name of a switch. See "List of keywords" below.
paths
   are the pathnames of segments, MSF's and directories for which it is possible to set the specified switch. The star convention is allowed, and includes links only if -chase is specified.

Control arguments:
-chase
   includes links and chases them when using the star convention.
-no_chase
   does not include links when using the star convention. (Default)

Access required: modify on the parent.

List of keywords:
copy_switch, csw
   (segments) If ON, allows processes lacking write access to modify a copy of the segment in the process directory.
damaged_switch, dsw
   (segments) If ON, the segment is assumed to have been damaged by a device error or system crash.
complete_volume_dump_switch, cvds
   If ON, the entry is dumped during a complete volume dump of the physical volume on which it resides.
incremental_volume_dump_switch, ivds
   If ON, the entry is dumped during an incremental dump cycle of the volume dumper.

perprocess_static_switch, ppsw
   (object segment) If ON, the segment's internal static storage is not initialized when a run unit is created.
safety_switch, ssw
   If ON, the delete command and delete_ subroutine query the user before deleting the entry.
transparent_paging_device_switch, tpds
   If ON, storage system pages of the entry are never allowed to reside on the bulk store unit.

switch_on

01/15/81 switch_on, swn

Syntax: swn keyword paths -control_args

Function: turns on a specified switch for one or more entries. For an MSF, the switch of the MSF directory (when possible) and those of all the components are turned on.

Arguments:
keyword
   specifies the name of a switch. See "List of keywords" below.
paths
   are the pathnames of segments, MSF's and directories for which it is possible to set the specified switch. The star convention is allowed, and includes links only if -chase is specified.

Control arguments:
-chase
   includes links and chases them when using the star convention.
-no_chase
   does not include links when using the star convention. (Default)

Access required: modify on the parent.

List of keywords:
copy_switch, csw
   (segments) If ON, allows processes lacking write access to modify a copy of the segment in the process directory.
damaged_switch, dsw
   (segments) If ON, the segment is assumed to have been damaged by a device error or system crash.
complete_volume_dump_switch, cvds
   If ON, the entry is dumped during a complete volume dump of the physical volume on which it resides.
incremental_volume_dump_switch, ivds
   If ON, the entry is dumped during an incremental dump cycle of the volume dumper.

perprocess_static_switch, ppsw
   (object segment) If ON, the segment's internal static storage is not initialized when a run unit is created.
safety_switch, ssw
   If ON, the delete command and delete_ subroutine query the user before deleting the entry.
transparent_paging_device_switch, tpds
   If ON, storage system pages of the entry are never allowed to reside on the bulk store unit.

tape_archive

10/02/80 tape_archive, ta

Syntax: ta key table_path args

Function: manages offline archival storage of files on magnetic tape.

Arguments:
key
   specifies the archival operation. Valid keys are listed below.
table_path
   is the pathname of the table of contents for the tape_archive. If the table does not exist, it is created.
args
   are additional arguments that vary according to the key used.

Notes:
Requests to move files between the storage system and tape are not performed immediately, but are queued within the table until the "go" key is performed by the user.

The cancel key can be used to cancel a pending request before the tapes are processed.

List of generic arguments for keys:
paths
   are pathnames of segments. The star convention is honored.
components
   are names of components of the archive. The star convention is honored.

List of file management keys:
a table_path -control_args paths
   appends a file to the archive.
ad table_path -control_args paths
   like a, but deletes the file from the storage system when done.
adf table_path -control_args paths
   like ad, but deletes forcibly.
r table_path -control_args paths
   replaces a file in the archive.
rd table_path -control_args paths
   like r, but deletes the file from the storage system when done.
rdf table_path -control_args paths
   like rd, but deletes forcibly.

u table_path -control_args paths
   updates a file in the archive if the DTBM has changed.
ud table_path -control_args paths
   like u, but deletes the file from the storage system when done.
udf table_path -control_args paths

```
   like ud, but deletes forcibly.
x table_path  -control_args  components
   extracts a file from the archive.
xf table_path  -control_args  components
   like x, but forcibly deletes an existing file of the same name from the
   storage system.


d table_path  -control_args  components
   deletes a file from the archive.
df table_path  -control_args  components
   like d, but deletes forcibly.
cancel table_path  -control_args  components
   cancels outstanding requests for a component.


List of control arguments for file management keys:
-mode ascii, -mode ebcdic, -mode binary
   causes a file to be archived using the specified recording mode.
-single_name, -snm
   causes additional names on a file not to be recorded/extracted.


List of miscellaneous keys:
t table_path  components   -control_args
   prints a table of contents for the archive.
go table_path  -control_arg
   causes the volume set to be mounted and all queued requests to be performed.
alter table_path parameter value
   alters the specified parameter to the specified value.
compact table_path
   schedules compaction of the volume set.
load_table table_path  -io_module modulename   -retain all   volume_ids
   causes a copy of the current online table to be retrieved from the tapes.
direct table_path  -control_args
   enters an interactive mode in which each line typed is interpreted as a key
   followed by the arguments to that key.


List of control arguments for the t key:
-brief, -bf
   prints contents in short form.
-long, -lg
   prints long form of contents.
-no_header, -nhe
   omit header information; list components only.
-header, -he
   prints header information only, if no components specified.
-pending
   lists only those components with pending requests.
-all, -a
   lists all components, even those previously deleted or replaced but still
   physically resident on the volume set.


List of control arguments for the go key:
-long, -lg
   causes printing of a message as each operation is being performed on the
   tape.
-retain all
```

leave volume set mounted after go request completes.
-retain none
demount volume set after go request completes. (DEFAULT).

List of control arguments for the direct key:
-retain all
leave the volume set mounted between go requests. Demount volume set and exit direct mode only when quit request is given.
-retain none
demount the volume set and exit direct mode after the go request completes. (DEFAULT).

List of parameters for the alter key:
module tape_ansi_/tape_ibm_
changes the tape I/O module used.
warning_limit fraction
prints warning whenever tape waste exceeds fraction.
auto_limit fraction
schedules compaction whenever waste exceeds fraction.
volume old_volume new_volume -alternate ,
volume -number N new_volume -alternate
changes the specified reel ID.
compaction off
unschedules an upcoming compaction.
density N
sets the density of the volume set to N bpi.

List of requests accepted in direct mode:
(In addition to the above requests)
save
save pending requests.
quit
discard requests since last "save" and exit without processing tape.
go
causes processing of the volume set, and exists direct mode, unless -retain all was specified.
.
causes tape_archive to identify itself.
..command_line
causes command_line to be passed to the command processor.

tape_control_language.gi

12/22/80 Tape Control Language

The TCL source file, written in the Tape Control Language (TCL) is the control file that governs file transfer with the tape_in or tape_out commands. For information on these commands, type:

help tape_in or help tape_out

The file is actually a program, written by the user, the contents of which describe the file transfer. When the user issues the tape_in or tape_out command, the control file named in the command line by the path argument is compiled and, if the compilation is successful, the generated code is interpreted to accomplish the desired file transfer(s). The same control file can be used with both the tape_in and tape_out commands.

Notes on creating a TCL control file:
The TCL control file consists of a list of statements of the form:

<keyword>: <argument(s)>;
or
<keyword>;

These statements are combined to form file-groups and file-groups are combined to form volume-groups. A TCL control file consists of one or more volume-groups.

Notes on file-group: A file-group is a list of statements that define one file transfer. A file-group must begin with a File statement and must contain a path statement. In addition, it may contain one or more local statements. A file-group is terminated by a global statement, an End statement, or another File statement.

Notes on volume-group: A volume-group is a series of statements that specify the file transfer(s) to be performed between the storage system and a particular tape volume-set. A volume-group must begin with a Volume statement, contain one or more file-groups, and terminate with an End statement. In additon, a volume-group may optionally contain one or more global statements, which apply to all the file-groups within the volume-group that follow the global statement.

List of TCL control file statements:
All TCL control files must have at least four statements-- a Volume statement, a File statement, a path statement, and an End statement. All other TCL statements are optional.

Volume: <volid>;
specifies the tape volume to be used in file transfer. This

statement causes a tape volume whose volume identifier is <volid> to be mounted on a 9-track drive. The "Volume" keyword must begin with an upper case letter. <volid> must consist of from 1 to 6 ASCII characters. If <volid> contains any of the following characters, it must be enclosed in quotes.

1. any ASCII control character
2. : ; , or blank
3. the sequence /* or */
4. if <volid> itself contains a quote character, the quote itself must be doubled and the entire <volid> string enclosed in quotes

(See the tape_ansi_ and tape_ibm_ info files for more details on volume specifications.)

File: <fileid>;
specifies the tape file to be read or written. The "File" keyword must begin with an upper case letter. The tape file is identified by <fileid> and must be from 1 to 17 characters for ANSI labeled tapes, and a valid DSNAME for IBM labeled tapes. The File statement marks the beginning of any local attributes for a given tape file transfer.

path: <pathname>;
specifies the pathname of the storage system file to be read or written. <pathname> can be either a relative or absolute pathname.

End;
marks the end of the TCL statements for that volume. "End" must begin with an upper case letter followed by a semicolon.

List of global statements:
A global statement changes a volume-group default. (See Tape Defaults below.)

Block: <blklen>;
specifies the tape file (maximum) physical block length, in bytes, to be used with subsequent file-groups.

Density: <den>;
indicates the density in which the volume is to be recorded. This statement may appear only once within a volume-group or an error is indicated.

Expiration: <date>;
specifies the expiration date of files to be written (created). <date> is of a form acceptable to the convert_date_to_binary subroutine, for example, "09/12/79".

Format: <form>;
specifies the tape record format to be used with subsequent file_groups.

Mode: <mode>;
specifies the tape mode and character code to be used with subsequent file-groups.

Record: <reclen>;
specifies the tape file (maximum) logical record length, in bytes, to be used with subsequent file-groups.

Storage: <structure>;
states the internal (logical) structure of the storage system

file(s) to be specified by subsequent file-groups. An unstructured file is referenced as a series of 9-bit bytes, commonly called lines; a sequential file is referenced as sequence of records, each record being a string of 9-bit bytes. <structure> must be either unstructured or sequential.

Tape: <tape-type>;
specifies the kind of tape that is processed. This statement may appear only once within a volume-group or an error is indicated.

List of local statements:
A local statement overrides the volume-group defaults in effect at the time a file-group is evaluated. A local statement has no effect outside of the file-group in which it occurs and may appear anywhere within the file-group.

block: <blklen>;
expiration: <date>;
format: <form>;
mode: <mode>;
record: <reclen>;
storage: <structure>;
these local statements operate exactly as do their global statement counterparts, except that they affect only the file-group in which they occur.

generate;
causes the entire contents of a file on an ANSI tape to be replaced while retaining the structure of the file itself and incrementing the file generation number.

modify;
causes the entire contents of a file on an ANSI or IBM labeled tape to be replaced while retaining the structure of the file itself.

number: <number>;
specifies the file sequence number of the file to be used in the file transfer. <number> must be either an integer between 1 and 9999 inclusive, or the character "*".

replace: <fileid>;
replaces an ANSI or IBM standard labeled tape. The file to be overwritten is identified by <fileid> in the replace local statement and the new file to be written is identified by <fileid> in the File statement.

storage_extend;
extends an already existing file in the storage system.

tape_extend;
allows new data records to be appended to an existing file on an ANSI or IBM standard labeled tape without in any way altering the previous contents of the tape file.

Tape Defaults:
If no Tape statement is specified in the control file, ANSI standard labeled tape will be assumed. If, however, a Tape statement is specified, the tape characteristics for that tape-type will preside as default until overridden.

Tape-type ANSI: (this is the default)
1) density: 800 bpi
2) file expiration: immediate

3) storage system file format: unstructured
4) mode: ascii character code
5) tape file record format: variable length records, blocked
6) physical block length: 2048 characters (maximum)
7) logical record length: 2048 characters (maximum)

Tape-type ibmsl, ibmnl, ibmdos:
1) density: 1600 bpi
2) file expiration: immediate
3) storage system file format: unstructured
4) mode: ebcdic
5) tape file record format: variable length records, blocked
6) physical block length: 8192 characters (maximum)
7) logical record length: 8188 characters (maximum)

Control File Comments:
Comments may be inserted anywhere within the TCL program by surrounding the comment text with the comment delimiters. /* is the delimiter that begins a comment, and */ is the delimiter that terminates a comment.

Notes:
To read files on a labeled tape, where the file names are not known, the <fileid> "*" can be used in the TCL File statement with tape_in only.

If it is wished to append a file to a given tape volume, it is not necessary to know how many files are on the tape if the tape is labeled. In such a case, the character "*" can be used in the TCL number statement if a valid file name is specified in the TCL File statement. This appending feature cannot be used to create a completely new volume.

Either tape_in or tape_out supports processing of unlabeled tapes, provided that the tapes are structured according to the OS standard.

For a more comprehensive description of the tape_io commands and the TCL, see the MPM Peripheral I/O, Order No. AX49.

tape_in

11/13/81 tape_in, tin

Syntax: tape_in path -control_args

Function: transfers files from magnetic tape to the storage system.

Arguments:
path
is the pathname of the control file that governs the file transfer. If path does not end with the .tcl suffix, it is assumed.

Control arguments:
-check, -ck
specifies that only semantic checking be done on the TCL control file. No tapes are mounted if this option is specified.
-ring
mounts volumes of the volume-set with write permit rings.
-severityN, -svN
causes the compiler's error messages with severity less than N (where N is 0, 1, 2, 3, or 4) not to be written into the "error_output" I/O switch. The default value for N is 0.

Notes on the TCL source file:
The control file that governs the file transfer is actually a program, written by the user, in the Tape Control Language (TCL). the contents of this control file describe the file transfer. The same control file can be used with both the tape_in and tape_out commands.

For additional information on the TCL, type help tcl.gi.
See also the MPM Peripheral I/O, Order No. AX49.

tape_out

11/13/81 tape_out, tout

Syntax: tape_out path -control_args

Function: transfers files from the storage system to magnetic tape.

Arguments:
path
is the pathname of the control file that governs the file transfer. If pathname does not end with the .tcl suffix, it is assumed.

Control arguments:
-check, -ck
specifies that only semantic checking be done on the TCL control file. NO tapes are mounted if this option is specified.
-force, -fc
specifies that the expiration date of a tape file to be overwritten is to be ignored. This control argument extends unconditional permission to overwrite a tape file, regardless of the file's "unexpired" status. This unconditional permission suppresses any query made by the I/O module to inquire about tape file's expiration date.

-ring
mounts volumes of the volume-set with write permit rings.
-severityN, -svN
causes the compiler's error messages with severity less than N (where N is 0, 1, 2, 3, or 4) not to be written into the "error_output" I/O switch. The default value for N is 0.

Notes on the TCL source file:
The control file that governs the file transfer is actually a program, written by the user, in the Tape Control Language (TCL). The contents of this control file describe the file transfer. The same control file can be used with both the tape_in and tape_out commands.

For additional information on the TCL, type help tcl.gi.
See also the MPM Peripheral I/O, Order No. AX49.

teco

04/20/76 teco

Syntax: teco path outpath

Function: character-oriented text editor provides simple editing requests, macro definitions, iterations, and conditional statements.

Arguments:
path
input segment.
outpath
output segment.

Notes: This command invokes the editor, searches for a start_up macro, and executes it. The default start_up macro reads the segment path into the buffer and puts the pointer at the beginning of the buffer. If outpath is given, q* is set to outpath, otherwise, it is set to path. If neither path or outpath is given, nothing is done. For more information about the editor, see the Tools PLM, Order No. AN51.

New Entry Point: teco$macro macro_name This entry point invokes teco, searches for a macro whose name is macro_name and executes it. The argument macro_name must be supplied. Additional arguments may be provided and are available to teco commands through the pushdown stack. As an example, the command line:

teco arg1 arg2 arg3

is equivalent to--

teco$macro start_up arg1 arg2 arg3

It differs from the standard entry point in several ways. First, if the teco commands executed encounter an error condition, the invocation of teco is aborted with an error message. Second, if an "eq" command is executed, teco never reachs its command level and no prompt message (Z) is printed. Finally, if teco command level is reached without errors, the macro mode is disabled and teco functions normally.

This entry point is useful for application programs written in teco, such as abbreivation editors.

New Features:

Q-register q" is set to the value of the last quoted string seen by teco. For the "n" command, q" is set to the actual text matched.

The "n" command is a search command that searches forward for a qedx-type regular expression. It is identical in syntax to the "s" command.

The teco command edits segments of any length. The buffer size is determined by the length of the text being edited.

Multics "e" commands:

eb/path/ where /path/ is a quoted string, copies the segment path to ==.bak and then writes text to the segment path. The command takes arguments and interprets them like the

"t" command, except that no arguments is equivalent to heb/path/.

ec/cmd/ where /cmd/ is a quoted string, passes the quoted string to the Multics command processor.

ei/path/ where /path/ is a quoted string, inserts the segment path immediately to the left of the text pointer.

em/macro/ where /macro/ is a quoted string, uses the teco search rules to find macro.teco and executes it. Any arguments to the "em" command are available to the macro invoked.

eo/path/ where /path/ is a quoted string, writes text to the segment path. The command takes arguments and interprets them like the "t" command, except that no arguments is equivalent to heo/path/.

eq exits from teco.

esn/name/ where n is a text q-register name and /name/ is a quoted string, calls the segment name passing it the arguments to the "es" command and the text q-register n. The segment called can modify the text q-register and return a numeric value.

Multics vs. PDP-10 teco:

Multics teco treats the entire segment as a single buffer. There are no equivalents to the PDP-10 append, yank, "n" search, or "w" commands, or any need for them.

Exiting from teco does not automatically write the buffer back to the segment. The "eo" or "eb" command must be used.

A search that fails does not change the current text pointer position.

Multics q-register names are one character in length and can be any one of the 95 printable ASCII characters, including blank.

Multics quoted string are of the form:

/string/ or qn

where / is any character except a letter or a digit and n is a q-register name. The form qn allows the contents of a q-register to be specified as a quoted string.

Multics command lines are terminated by the two character sequence dollar sign ($) newline. The altmode character is not used in Multics.

The Multics "s" command always gives an error message if the search fails.

Multics teco uses | to denote the logical or operator.

Multics teco treats -1< as an error.

Multics teco expressions are evaluated somewhat differently. See the documentation or use the "=" command to print out the value of a questionable expression. Multics teco "=" command takes zero, one, or two arguments.

The numeric value of a text q-register is equal to the number of characters of text.

Multics error messages can be of two forms, long or short. Short messages are eight characters and long ones are up to fifty characters. The user can control error message length.

ted

01/29/81 ted 2.6

Syntax: ted ted_com -control_args

Function: ted can be used to create and edit ASCII segments. ted can do many kinds of text processing. ted can be used recursively to a depth of 14 and it can be called as an active function.

Arguments:

ted_com

If supplied, the contents of the named segment is read into buffer exec and then executed. If the contents of b(exec) is exhausted, then request lines will be read from user_input unless the -com option was present.

Control arguments:

-reset level

used to break out of a ted_com loop and return to ted request level. "level" is a 2-digit number specifying the level to be returned to. If level is not specified, then the most recent invocation is used.
This argument is mutually exclusive with all others.

-abort, -com

This causes ted to execute a ted_com so that if it has an error, it will exit instead of returning to ted request level.

-pathname XXX, -pn XXX

Begin execution by reading segment XXX into b(0).

-arguments ARGS, -ag ARGS

This causes all remaining arguments to be made available to a ted_com. The arguments may be referenced either in buffers (arg1), (arg2), etc. or as line 1, line 2, etc. of buffer (args).

-read

This causes the read option to be set on. (Default)

-no_read

This causes the read option to be set off. SEE: o

-safe

causes ted to place its work segments in your default working directory so as to be able to survive loss of the process.

-restart

If ted was called with the -safe option and the system crashed, the terminal goes off-line, etc. this call is used to restart where you left off. You must restart in the same level as when you called ted -safe. Due to the great variety of reasons why the system crashes, occasionally the restart will not work. This is infrequent.

-status, -st

causes a list of the environments which exist. [This is an interim facility at this point in time.
This argument is mutually exclusive with all others.

-blank
   This requires a blank to be necessary between multiple requests on a line. Due to the fact that a,c,i,d,r,e,w, requests use up the rest of the line, the blank is required just after the request character. This is the suggested mode of operation.
-part_blank
   This requires the blank on only the a,c,i,d,r,e,w requests.
-no_blank
   This requires no extra space. (default)

-label
   Enable tracing of labels processed.
-no_label
   Disable tracing of labels. (default)
-trace_edit
   Display each ted request line in EDIT mode before it is executed.
-trace_input
   Display each line of INPUT mode data before it is inserted.
-trace
   Combination of -trace_edit and -trace_input.
-no_trace
   No tracing. (default)

-debug, -db
   Tells ted to type out "Edit." before accepting the first line from user_input. Useful for detecting when a ted_com "falls" back to request level.
-break
   Enable break processing.
-no_break
   Disable break processing. (default)
-pause
   This call is used to interrupt a ted_com execution. This causes ted to believe it has just encountered a breakpoint. The next time a request line is fetched, the input routine will enter the break sequence.
      This argument is mutually exclusive with all others.

-Jset XX...
   Set the collating sequence for the J sort request. When XX... is present, each X represents a mapping pair in one of these forms:
    CC   map first character to second
    X->YC   map range of characters to second
    X->YX->Y map range of characters to range of characters. Ranges
       must be of equal size.
   (When a character is mapped to '777 it means to ignore it.)
    C is any of 3 forms
      'ooo   ooo is 3 octal characters
      'Z   Z is any character
      Z   Z is any character except ' (which must be entered as '')
    X->Y means the contigious characters X through Y. (X< Y in the
         9-bit collating sequence. X and Y are any of the C forms.
   when XX... is absent it sets to (uppercase=lowercase):
     '000->'177'000->'177'200->'777'777a->zA->Z

-Jshow

Display the current collating sequence for the J sort.

Notes: This info segment is used by both the help command and the help request of ted. Only the external entries are known to the help command. All the rest is formatted in a way which is used by the help request. The whole segment may be dprinted for user browsing.

List of requests: Only the most important aspects of the requests are documented here. The default address is shown inside a "[]" pair at the beginning of the explaination. If nothing is there, there no address is allowed. If there is "." or "$" then 1 address is allowed. If there is ".,." or "1,$" then 2 addresses are allowed. If there is "*" it is a special case which does not fit the rules just stated. RE stands for a regular expression. (B) stands for a buffer name.

"
   [.] ignore rest of line
%(B) etc
   [ ] call a buffer, with optional arguments
line
^# line
   [*] execute line if buffer status is/is not true. May test for buffer empty, on specified line or within specified range.
*/RE/ line
^*/RE/ line
   [.,.] execute line if regexp is found/not found in addressed range
:C
:(L)
   [ ] label define, must be at BOL

=
   [.] print current line number
...
   [*] evaluate
>x
   [ ] transfer of control (goto), x may have the form "C", "(str)", "+n", "-n"
^>x
   [ ] goto if error during rest of line's execution, x is the same form as above.
|xxx
   [ ] call external request ted_xxx_

   [ ] return from buffer

a TEXT f
   [.] append
!a
   [.,.] bulk append, terminated by "." line
b(B)
   [ ] change current buffer
b(B,adr,adr)
   [ ] change current buffer to a window of a buffer
!b(B)
!b(B,adr,adr)

```
   [   ] remember current buffer, then change to new one
b()
   [   ] change to remembered buffer
^b(B)
   [   ] not-buffer (delete buffer)


c TEXT f
   [.,.] change
!c
   [.,.] bulk change
d
   [.,.] delete
e line
   [   ] execute command line
!e
   [   ] print, then execute a command line (also E)
.. line
   [   ] execute command line without input function expansion, BOL
   only


f(B)
   [   ] file-out into a buffer, auto reversion
!f(B)
   [   ] file-out to a buffer, no auto reversion (also F)
!f<NL>
   [   ] revert previous form of file-out (also F)
g=/RE/
   [1,$] global, linenumber of all lines which match RE
gd/RE/
   [1,$] global, delete all lines which match RE
gp/RE/
   [1,$] global, print all lines which match RE
g!p/RE/
   [1,$] global, print/linenumber sll lines which match RE


g*SELECTION ACTIONs
   [1,$] global, do ACTIONs on all lines which match SELECTION
h/T,c,c.../
   [.,.] process out pseudo-tab
help
   online information
i TEXT f
   [  .] insert
!i
   [  .] bulk insert
j/spec/
   [1,$] sort
!j/spec/
   [1,$] sort, with settable collating sequence (also J)


k(B)
   [.,.] kopy
!k(B)
!k(B,adr)
   [.,.] kopy-append to end of/specified place in any buffer (also K)
l
```

```
   [    ] linefeed to user_output
!l
   [    ] linefeed to error_output (also L)
m(B)
   [.,.] move
!m(B)
!m(B,adr)
   [.,.] move-append to end of/specified place in any buffer (also M)


n
   [ * ] nop
o
   [    ] option/modes
p
   [.,.] print
!p
   [.,.] print with linenumbers (also P)
q
   [    ] quit
!q
   [    ] quit, without modified buffer check (also Q)


r path
   [  .] read a segment into current buffer
!r path
   read w/abbrev expansion of pathname (also R)
r (B)
r (B,adr,adr)
   [  .] read all or part of a buffer into current buffer
^r path
   [    ] not-read (force pathname)
s/RE/REPL/
   [.,.] substitute occurances of RE with REPL
!s/RE/REPL/
   [.,.] no-fail substitute


t/xxx/
   [    ] type string to user_output
!t/xxx/
   [    ] type string to error_output (also T)
u/RE/
   [.,.] lowercase what matches RE
!u/RE/
   [.,.] uppercase what matched RE
v
   [1,$] inverse form of global, i.e. do when NO match


w path
   [1,$] write from current buffer into a segment
!w path
   [1,$] write w/abbrev expansion of pathname (also W)
w (B)
   [1,$] write from current buffer into another buffer
wm
   [    ] write-modified (BLANK mode only)
x
```

```
   [    ] status of all buffers
!x(B)
!X< NL>
   [    ] status of named/current buffer
xm
   [    ] status of all modified buffers (BLANK mode only)


y
   [.,.] replace SPs with HTs where possible, remove trailing
   whitespace
zdump
   [.,.] dump octal/ASCII
z.fi.ad
   [.,.] fill/adjust
z.fi.na
   [.,.] fill/no-adjust
zif  ...  line
   [ * ] execute line if result of evaluation is not "0" or "false


 b
   invoke buffer (input function)
 r
   read line from user_input without NL (input function)
  ...
   evaluate (input function)
```

terminate_refname

06/25/81 terminate_refname, tmr

Syntax: tmr ref_names -control_args

Function: removes a segment or a single reference name (tmsr) from the user's address space and resets links to the terminated segment. It is commonly used prior to initiating a different version of a program.

Arguments:
ref_names
   are the reference names of segments to be terminated.

Control arguments:
-brief, -bf
   suppresses the error message normally printed when a segment to be terminated is not known (initiated).
-long, -lg
   does not suppress the above error message. (Default)
-name STR, -nm STR
   specifies an entryname (terminate) or reference name (tmr and tmsr) STR that begins with a minus sign, to distinguish it from a control argument.

List of terminate commands with short names:
terminate, tm
terminate_segno, tms
terminate_refname, tmr
terminate_single_refname, tmsr

Notes: The tmr command allows termination by reference name rather than by pathname. The segment itself is terminated, not merely the reference name specified.

Caution must be exercised not to unintentionally terminate a segment of the command language interpreter or another critical piece of the environment. Fatal process errors usually result from such an action.

texto

05/31/83 texto

ACCES A TEXTO:
Ajouter (dans le start_up) la regle de recherche suivante
asr >udd>Chemdata>Texto
puis taper la commande
texto

ASSISTANCE DOCUMENTATION:
Dominique SOURGEN Centre de Calcul piece 014d tel 638 5057
Francoise GILLET Centre de Calcul piece 018d tel 638 5514

CARACTERISTIQUES PRINCIPALES:
texto est un progiciel de gestion documentaire developpe par la societe Chemdata de Lyon.
Les caracteristiques principales en sont
---> mode conversationnel
---> utilisable par non-informaticiens
---> souplesse de structure (fichiers modifiables a tout moment)
___> 2 procedures d'interrogation
simple (resultats donnes immediatement)
composee (chaque question genere un ensemble de reponses memorise et reutilisable)
---> editions elaborees
---> possibilites de creer des index, d'effectuer des tris, de creer des sous-fichiers

FONCTIONNEMENT:
3 types de fichiers
---> le CATALOGUE memorise divers types de documents utilitaires
*documents de parametres (permettent de structurer les fichiers documentaires)
*documents d'edition (definition des modeles d'edition)
*dialogues (enchainement de commandes texto pour utilisation ulterieure)
---> les FICHIERS DOCUMENTAIRES sont les fichiers proprement dits
Ils sont structures grace aux documents de parametres.
Ils sont composes d'une suite de documents numerotes.
ce sont des fichiers sequentiels indexes Multics.
---> les INDEX pouvant etre
*des fichiers inverses permettant une interrogation rapide et economique sur de gros volumes
*des index de tris pour presenter des editions triees

EXEMPLES D'APPLICATION TEXTO:
---> fichiers de documentation interne de l'entreprise
fichiers d'adresses

fichiers de rapports scientifiques ...
---> fichiers de gestion de bibliotheque
gestion de la reception et de la ventilation
des revues
---> constitution de banques et de bases de donnees

time

12/30/80 time

Syntax: time dt

Function: returns a four-digit time of day in the form "hh:mm" where 00 <= hh <= 23 and 00 <= mm <= 59.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current time is used. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [time dt]

topics

02/04/81 General Information

The Multics help system includes a number of info segments that contain general information about features or use of the system. The info_names of these segments end with a suffix "gi.info" (gi for general information). For example, acl_matching.gi.info describes how Access Control List (acl) entries are matched with User_ids in access control commands such as set_acl. As with all info_names, when typing "help" with the info_name of one of these gi segments, you need not type the suffix ".info". In addition, you may also leave off the suffix ".gi". So, to get the info segment on acl_matching, you need only type

```
help acl_matching
```

gi Info_names:
To get a complete list of the info segments whose names have the suffix "gi.info", use the list_help command:

```
list_help gi
```

If you want a list of info segments that pertain to a particular subject, type a word (or a string of characters that is part of a word) that describes that subject and the list_help command will print a list of all info_names containing that word or partial word. For example, if you want a list of info segments that describe the Multics mail facility, type

```
list_help mail
```

For more information about the list_help command, including its method for matching character strings, type

```
help list_help
```

total_output_requests

12/07/81 total_output_requests, tor

Syntax: tor request_types -control_args

Function: prints the total number of requests in one or more I/O Daemon queues.

Arguments:
request_types
   name the request types whose totals are to be listed. The default is to list totals in the queues of the default printer request type.

Control arguments:
   may only be given when invoked as a command.
-brief, -bf
   omits request types which are empty.
-long, -lg
   includes request types which are empty. (default)
-all, -a
   lists totals for all I/O Daemon request type queues.
-inhibit_error, -ihe
   suppresses error messages for request type queues to which the user does not have access. Totals for such queues are printed as *****.

Access required: To use tor for a given request type, you must have s (status) extended access to the queue segments for that request type.

Syntax as an active function: [tor request_type]

Notes: When invoked as an active function, tor returns the count of entries in each queue defined for the request type. A request type may have up to 4 queues, numbered from 1 through 4. The tor active function returns 1, 2, 3 or 4 numbers, representing the totals entries in queues 1, 2, 3 and 4 respectively. If an error occurs while accessing any of the queues, a * is returned for the total in that queue.

No control arguments may be given when tor is invoked as an active function.

Examples:

```
  ! tor -all -brief

    printer:              0     0     3     2
    manuals:              0     3
```

```
fl1x8:                    0     2
hdsa_prt:              *****
 Incorrect access on entry.  hdsa_prt queue 1
mit_pps_2sided:           0     0     1     0
punch:          2
```

trace_stack

12/22/80 trace_stack, ts

Syntax: ts -control_args

Function: prints a detailed explanation of the current process stack history in reverse order (most recent frame first).

Control arguments:
-brief, -bf
   suppresses listing of source lines, arguments, and handlers. This control argument cannot be specified if -long is also specified as a control argument.
-long, -lg
   prints octal dump of each stack frame.
-depth N, -dh N
   dumps only N frames.
-stack_ptr PTR, -sp PTR
   starts tracing from stack frame at PTR where PTR is a virtual pointer acceptable to cv_ptr_. PTR points to the stack frame at which tracing is to begin.

Notes: The trace_stack command is most useful after a fault or other error condition. If the command is invoked after such an error, the machine registers at the time of the fault are also printed, as well as an explanation of the fault. The source line in which it occurred can be given if the object segment is compiled with the -table option.

For a description of stack frames, see "Multics Stack Segments" in the MPM Subsystem Writers' Guide.

truncate

11/13/81 truncate, tc

Syntax: tc -control_arg path length
or: tc segno length

Function: truncates a segment to a specified length, and resets the bit count accordingly.

Arguments:
path
is the pathname of a segment. The star convention is NOT allowed.
segno
is an octal segment number.
length
octal integer indicating the length of the segment in words after truncation; zero by default.

Control arguments:
-name, -nm
specifies that the octal number following it is a pathname.

Access required: write access on the segment to be truncated.

Notes: If the segment is already shorter than the specified length, its length is unchanged, but the bit count is set to the specified length.

This command should not be used on segments that are (or are components of) structured files.

vfile_adjust

9/16/75 vfile_adjust, vfa

Syntax: vfa path -control_arg

Function: adjusts a storage system file that may have been left in an inconsistent state by an interrupted opening.

Arguments:
path
   is the pathname of a single file to be adjusted.

Control arguments:
  (one specified if and only if file is unstructured)
-set_nl
   append a newline char if file does not end with one
-use_nl
   truncate file after last new_line character set_bc set bitcount and truncate
   at last nonzero byte in the file
-use_bc N
   truncate at byte specified by bitcount of component N (last nonzero
   component if N not specified)

Notes: a sequential or blocked file is adjusted by truncation after the last complete record. An indexed file is adjusted by completing any interrupted operation.
  The condition of a file can be determined by using the vfile_status command.
  See documentation of the vfile_ I/O module in the MPM Subroutines for further details.

vfile_status

9/16/75 vfile_status,vfs

Syntax: vfs path

Function: prints the apparent type and length of storage system files. Additional info is provided for structured files.

Arguments:
path
is the pathname of a file. The star convention is allowed.

Notes: for structured files (sequential,blocked, or indexed), the state of the file is printed (if busy). The following statistics are also provided for indexed files--

1. number of records in the file, including those of zero length
2. number of nonnull records, if different from the above
3. total length of the records (in bytes)
4. number of blocks in the free space list for records
5. height of the index tree (zero for empty files)
6. number of nodes in the index (each occupies a single 1K page)
7. total length of all keys (bytes)
8. number of keys
9. number of duplicate keys
10. total length of duplicate keys

Additional information about a file can be obtained using the status command. See documentation of the vfile_ I/O module in the MPM Subroutines for further details.

walk_subtree

05/10/78 walk_subtree, ws

Syntax: ws path command_line -control_args

Function: executes a command line in a given directory (starting node) and in inferior directories.

Arguments:
path
   starting node; -wd specifies working directory.
command_line
   command line to be executed. Command lines containing blanks must be quoted.

Control arguments:
-first N, -ft N
   where N is the first level in the storage system hierarchy at which the command line is to be executed; by default N is 1.
-last N, -lt N
   where N is the last level in the storage system hierarchy at which the command line is to be executed; by default N is 99999.
-brief, -bf
   suppresses printing the names of directories in which the command is executed.
-bottom_up, -bu
   starts execution of the command line at the last level and proceeds upward through the storage system.

Notes: The command has a cleanup handler. If, after the user quits out of the command, she types rl, her working directory is changed back to what it was before walk_subtree was invoked.

where

07/16/81 where, wh

Syntax: wh names -control_args

Function: uses the standard search rules to search for a given segment or entry point.

Arguments:
names
are segment and entry point names. The star convention is NOT allowed.

Control arguments:
-all, -a
lists the pathnames of all segments and entry points with the specified names that can be found using the current search rules, the user's effective access to each segment or entry point, and the name of the search rule used to find each segment or entry point.
-brief, -bf
prints only the pathname of each entry found. (Default)
-entry_point, -ep
searches for entry points. If a name argument does not contain a dollar sign ($), the where command searches for the entry point name$name.
-inhibit_error, -ihe
does not print an error message if no segments can be found for a given name. For the where command, no output is printed; for the active function, null string is returned.

-long, -lg
prints the name of the search rule used to find each segment and the user's effective access to the segment, in addition to the pathname. This control argument is incompatible with -all.
-no_inhibit_error, -nihe
prints an error message if no segments can be found for a given name. (Default)
-segment, -sm
searches for segments. This is the default, unless name contains a dollar sign.

Notes: The command prints out the full pathname of the segment, using its primary name and the entry point name if one is requested. If the segment or entry point is not in the search path, an error message is printed.

The primary name of a storage system entry is the name that is first in the list of names on that entry.

If the -all control argument is not specified, the where command prints

information only about the first matching segment or entry point encountered (using the standard search rules).

The -entry_point and -segment control arguments are mutually exclusive. If one of these control arguments is used, all the name arguments are assumed to be of the type specified.

If neither the -entry_point nor -segment control argument is specified, the where command scans the name arguments. Any name arguments that contain a dollar sign are assumed to be names of entry points; all others are assumed to be names of segments.

Syntax as active function: [wh name -control_args]

Notes on active function: The active function returns the pathname of the segment, as found by the search rules. Only one name can be specified. The -all, -brief and -long control arguments are not allowed. Unless -inhibit_error is specified, an error occurs if no segment can be found.

where_search_paths

01/17/79 where_search_paths, wsp

Syntax: wsp search_list entryname -control_args

Function: finds occurrences of an entryname in a search list.

Arguments:
search_list
   is the name of the search list to search.
entryname
   is the entryname to search for.

Control arguments:
-all, -a
   print all occurrences of the entryname in the search list.
   The default is to just print the first occurrence.

Access required: To find an entryname in the search list, the user must have s access on the containing directory or non-null access to the entry.

Syntax as an active function: [wsp search_list entryname -control_args]

who

05/22/81 who

Syntax: who User_ids -control_args

Function: lists User_ids and other information about current users of the system.

Arguments:
User_ids
are match names where:
Person_id
lists users with the name Person_id.
.Project_id
lists users with the project name Project_id.
Person_id.Project.id
lists users with the specified person and project.

Control arguments:
-absentee, -as
lists absentee users. Absentee users are denoted in the list by an asterisk (*) following Person_id.Project_id.
-brief, -bf
suppresses the printing of the header. Not allowed for the active function.
-daemon, -dmn
lists daemon users.
-interactive, -ia
lists interactive users.

-long, -lg
prints the date and time logged in, the terminal identification and the load units of each user, in addition to the user's name and project. The header includes installation identification and the time the system was brought up. If available, the time of the next scheduled shutdown, the time when service will resume after the shutdown, and the time of the previous shutdown are printed. Not allowed for the active function.
-name, -nm
sorts the output by the name (Person_id) of each user.
-project, -pj
sorts the output by the Project_id of each user.

Notes: If the control_args -interactive, -absentee, or -daemon are not specified, the default is to list all three types of users. If one or more of these control_args is specified, only users of the specified type(s) are listed.

If the who command is specified with no arguments, the system responds with a two-line header followed by a list of interactive users sorted

according to login time.

If the -project and -name control arguments are omitted, the output is sorted on login time. Both arguments cannot be used together, because the sort is performed on one key at a time.

If a User_name is specified, the header is suppressed even if the -long control argument is specified.

It is possible to prevent the user's own name from being listed by all users' invocation of who; to do this, the user should contact the project administrator.

Syntax as active function: [who User_ids -control_args]

Notes on active function: The active function returns a list of Person_id.Project_id pairs, requoted and separated by spaces. Control arguments can be used to select and sort.

working_dir

07/23/80 working_dir, wd

Syntax: wd

Function: returns the pathname of the working directory of the process in which it is invoked.

Syntax as active function: [wd]

year

12/30/80 year

Syntax: year dt

Function: returns the two-digit number of a year of the century from 01 to 99.

Arguments:
dt
is a date-time in a form acceptable to convert_date_to_binary_. If no argument is specified, the current year is used. See date_time_strings.gi.info for valid dt arguments.

Notes: See the MPM Subroutines for a complete description of convert_date_to_binary_. See date_and_time.info for other date/time commands and active functions.

Syntax as active function: [year dt]

V- LISTE ALPHABETIQUE DE TOUTES LES COMMANDES D'INFORMATION GENERALES

1 FAST.gi (FAST)
2 absentee.gi (abs,absentee)
3 access_isolation.gi (access_isolation)
4 acl_matching.gi (acl_matching)
5 acl_primitives.gi (acl_primitives,acl_entries.gi)
6 aim_use.gi (aim_use)
7 allocation_storage.gi (allocation_storage)
8 apl_context_editing.gi
9 audit_.gi
10 autobaud (autobaud.gi)
11 bad_area_format.gi (bad_area_format)
12 basic_files.gi (basic_files)
13 canonicalization (canonicalization.gi)
14 card_access_control.gi (card_access_control)
15 card_input.gi (card_input)
16 channel_names.gi
17 cobol_implementation.gi (cobol_implementation,cobol.implementation.gi)
18 cobol_mcs.gi (cobol_mcs,cobol.mcs.gi)
19 compose.artwork.gi (compose.art.gi,comp.artwork.gi,comp.art.gi)
20 compose.macros.gi (documentation.macros.gi)
21 control_arguments.gi (control_args.gi)
22 copy_on_write.gi (copy_on_write)
23 correspondence.gi (correspondence)
24 damaged_keyed_files.gi
25 damaged_segments.gi (damaged_segments)
26 date_time_strings (date_time_strings.gi)
27 dict.gi (dict)
28 documentation.gi (doc.gi)
29 ed.gi
30 editing.gi
31 ep_basic.gi (ep_basic)
32 extended_access.gi (ext_access)
33 external_storage.gi
34 fast_topics.gi (fast_topics)
35 fort_options.gi (fort_options)
36 fortran.gi
37 fortran_77.gi
38 gcos.gi
39 graphic_fonts.gi
40 hardcore_wait_events.gi
41 help_infos.gi (help_infos)
42 help_system.gi (help_system)
43 hunt_dec.gi
44 info_seg.gi
45 instance_tags.gi (instance_tag.gi)
46 ioa_control.gi (ioa_control,format_line_control.gi,fl_control.gi)
47 linkage_errors.gi
48 linking.gi (linking)
49 lister.gi
50 listform_segment.gi (listform_segment)
51 listin_segment.gi (listin_segment)
52 load_control.gi (load_control)
53 logical_volumes.gi
54 ltsm.gi

```
55  lv_attaching.gi
56  mail_system.gi (mail_system,mlsys.gi)
57  manuals.gi (manuals)
58  master_directories.gi
59  mode_string_.gi
60  modules.gi (modules,module.gi)
61  new_audit.gi
62  new_fortran_conversions.gi
63  new_fortran_differences.gi (new_fortran.differences)
64  new_fortran_extensions.gi (new_fortran.extensions)
65  new_fortran_optimizer.gi (nfopt.gi)
66  order_manuals.gi (order_manuals)
67  pascal.gi
68  pllr24 (pllr24.gi)
69  print_mail (print_mail.gi)
70  printed_output (printed_output.gi)
71  probe.gi
72  process_overseers.gi (po.gi)
73  process_preservation.gi
74  protection_notices.gi
75  questions.gi
76  request_ids.gi (request_ids)
77  set_tty.gi (stty.gi)
78  severity_indicators.gi
79  speedtype.gi (speedtype)
80  star_equal.gi (star_equals,star_equals.gi)
81  star_system_links.gi
82  static_handlers.gi (static_handlers)
83  tape_control_language.gi (tcl.gi)
84  topics.gi (topics,topic.gi)
85  tss_basic.gi (tss_basic)
86  tss_fortran.gi (tss_fortran)
87  tty_modes.gi
88  util.fortran.gi
89  virtual_entries.gi (virtual_entries)
90  virtual_pointers (virtual_pointers.gi)
91  volume_names.gi
```

VI- LISTE ALPHABETIQUE DE TOUTES LES COMMANDES EXPLOITABLES PAR HELP

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
1	963	029	26		
2	AF.compin	active_function.compin	35		
3	Answering_Service.errors	answering_service.errors AS.errors as.errors	455		
4	FAST.gi	FAST	39		
5	MAP		24		
6	PPS_Support.errors		8		
7	abbrev	ab	69		
8	abbrev.errors	ab.errors	149		
9	abbrev_		41	3	
10	absentee.gi	abs absentee	166		
11	absolute_pathname_		87	2	
12	accept_messages	am	94		
13	access_isolation.gi	access_isolation	33		
14	acl_matching.gi	acl_matching	21		
15	acl_primitives.gi	acl_primitives acl_entries.gi	107		
16	acquire_resource	aqr	103		
17	active_fnc_err_		93	2	
18	add_bit_offset_		36	1	
19	add_char_offset_		55	1	
20	add_dict_words	adw	66		
21	add_epilogue_handler_		25	1	
22	add_name	an	13		
23	add_pnotice		41		
24	add_search_paths	asp	59		
25	add_search_rules	asr	46		
26	add_symbols	asb	42		
27	adjust_bit_count	abc	27		
28	adjust_bit_count_		43	1	
29	adjust_mrds_db	amdb	61		
30	after	af	10		yes
31	aim_check_		81	3	
32	aim_use.gi	aim_use aim	69		
33	allocation_storage.gi	allocation_storage	71		
34	alm		257		
35	alm_abs	aa	51		
36	and		11		yes
37	answer		85		
38	apl		80		
39	apl.errors		279		
40	apl_context_editing.gi		76		
41	apl_convert		51		
42	apl_cube		55		
43	apl_end		6		
44	apl_erf_		17		
45	apl_erf_.pl1		126		
46	apl_get_list_nums_		40		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
47	apl_help_available		10		
48	apl_ioa_		34		
49	apl_linalg		62		
50	apl_linprog		46		
51	apl_pickup_float_bin_2_		68		
52	apl_read_segment_		29		
53	apl_start		27		
54	apl_summary		187		
55	apl_system_commands		20		
56	apl_system_functions		19		
57	apl_system_variables		27		
58	apl_tekplot		189		
59	apl_vs_vlapl		215		
60	apl_vs_vsapl.extensions		91		
61	apl_vs_vsapl.incompat		95		
62	apl_ws		14		
63	apl_ws_index		13		
64	apl_wsfns		10		
65	append_list	als	27		
66	archive	ac	165		
67	archive_		320	5	
68	archive_sort	as	16		
69	archive_table	act	33		yes
70	area_info_		110	1	
71	area_status		21		
72	argument.compin	arg.compin	36		
73	argument_list.compin	arg_list.compin	20		
74	arithmetic_af	arithmetic	11		
75	arithmetic_to_ascii_		58	1	
76	arpanet.errors		41		
77	as_meters		40		
78	ascii_to_ebcdic_		42	2	
79	ask_		219	15	
80	assign_		17	1	
81	assign_resource	ar	147		
82	attach_audit	ata	41		
83	attach_lv	alv	17		
84	audit_		151		
85	audit_.gi		187		
86	audit_editor		196		
87	autobaud	autobaud.gi	166		
88	backquote	lisp.backquote	143		
89	backup_cleanup	bc	38		
90	bad_area_format.gi	bad_area_format	6		
91	basic		23		
92	basic_files.gi	basic_files	95		
93	basic_system	bs	39		
94	before	be	11		yes
95	binary	bin	18		yes
96	bind	bd	221		
97	bit_offset_		27	1	
98	bool		43		yes
99	branches		21		yes
100	bullet.compin		14		
101	byte		17		yes
102	calc		93		yes

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
103	calendar		133		
104	calendrier		92		
105	can		37		
106	cancel_abs_request	car	89		
107	cancel_carry_request	ccr	36		
108	cancel_cobol_program	ccp	17		
109	cancel_daemon_request	cdr	86		
110	cancel_resource	cnr	27		
111	cancel_retrieval_request	crr	70		
112	canonicalization	canonicalization.gi	414		
113	canonicalize	canon	31		
114	capo		20		
115	card_access_control.gi	card_access_control	78		
116	card_input.gi	card_input	631		
117	carry_load		65		
118	ceil		10		yes
119	change_default_wdir	cdwd	16		
120	change_error_mode	cem	19		
121	change_symbols	csb	37		
122	change_tuning_parameters	ctp	39		
123	change_wdir	cwd	12		
124	change_wdir_		15	1	
125	channel_names		54		
126	channel_names.gi		56		
127	char_bit_offset_fcns_		20		
128	char_offset_		32	1	
129	character_string		25		
130	character_string_af		25		
131	check_iacl		27		
132	check_info_segs	cis	68		yes
133	check_mst	ckm	98		
134	check_short_strings		43		
135	check_star_name_		25	2	
136	clear_partition		40		
137	clear_resource	clr	21		
138	clock_		15	1	
139	close_file	cf	25		
140	cobol		113		
141	cobol_abs	cba	51		
142	cobol_implementation.gi	cobol_implementation cobol.implementation.gi cobol.implementation	16		
143	cobol_mcs.gi	cobol_mcs cobol.mcs.gi cobol.mcs	100		
144	collate		10		yes
145	collate9		11		yes
146	collating.compin	coll_page.compin coll_cont.compin	87		
147	com_err_		120	2	
148	command.compin		32		
149	command_language.changes	cl.changes	126		
150	command_processor_.errors	command_processor.errors	38		
151	command_query_		227	3	
152	command_usage_count	cuc	67		
153	comp_dir_info		80		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
154	compare		37		yes
155	compare_ascii	cpa	75		
156	compare_dump_tape		60		
157	compare_mst		35		
158	compare_object	cob	42		
159	compdv		31		
160	compile_gct		32		
161	compile_gdt		34		
162	component_info_		27	2	
163	compose	comp	170		
164	compose.artwork.gi	compose.art.gi comp.artwork.gi comp.art.gi	171		
165	compose.builtins	comp_builtin_list	87		
166	compose.changes	comp.changes cchng	744		
167	compose.controls	comp_ctl_list	177		
168	compose.errors	csts.info.status comp.errors	174		
169	compose.macros.gi	documentation.macros.gi	251		
170	condition_interpreter_		17	1	
171	connect_help		26		
172	console_output	co	21		
173	console_report		43		
174	contents		10		yes
175	continue_to_signal_		16	1	
176	control_arguments.gi	control_args.gi	241		
177	convert_aim_attributes_		16	1	
178	convert_authorization_		105	6	
179	convert_characters	cvc	73		
180	convert_date_to_binary_		50	2	
181	convert_dial_message_		16	1	
182	convert_ec	cvec	111		
183	convert_numeric_file		21		
184	convert_runoff	cv_rf	19		
185	convert_status_code_		17	1	
186	convert_tsoapl_ws	ctw	27		
187	convert_v1_fdump		36		
188	copy	cp	87		
189	copy_acl		25		
190	copy_acl_		31		
191	copy_cards	ccd	26		
192	copy_characters	cpch	10		yes
193	copy_dir	cpd	107		
194	copy_dump_tape		71		
195	copy_file	cpf	110		
196	copy_iacl_dir		25		
197	copy_iacl_seg		23		
198	copy_list	cpls	73		
199	copy_mst	cpm	19		
200	copy_names		24		
201	copy_names_		30		
202	copy_on_write.gi	copy_on_write	32		
203	copy_seg_		36		
204	copy_switch_off	csf	12		
205	copy_switch_on	csn	12		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
206	correspondence.gi	correspondence	40		
207	count_dict_words	cdw	12		
208	cours		92		
209	cpu_time_and_paging_		16	1	
210	create	cr	14		
211	create_area		34		
212	create_daemon_queues	cdq	28		
213	create_data_segment	cds	22		
214	create_data_segment_		15	1	
215	create_dir	cd	43		
216	create_ips_mask_		15	1	
217	create_list	cls	31		
218	create_mrds_db	cmdb	111		
219	create_mrds_dm_include	cmdmi	78		
220	create_mrds_dm_table	cmdmt	92		
221	create_mrds_dsm	cmdsm	95		
222	create_wordlist	cwl	39		
223	crecho		20		
224	cross_reference	cref	254		
225	cross_ring_	cross_ring_io	18		
226	cross_ring_io_		14	1	
227	ctl_char		15		
228	ctlarg.compin	controlargument.compin	36		
229	ctlarg_list.compin	controlargument_list.compin	20		
230	cu_		2663	39	
231	cu_.changes		83		
232	cumulative_page_trace	cpt	80		
233	cv_bin_		43	3	
234	cv_dec_		15	1	
235	cv_dec_check_		16	1	
236	cv_entry_		19	1	
237	cv_hex_		15	1	
238	cv_hex_check_		16	1	
239	cv_oct_		15	1	
240	cv_oct_check_		16	1	
241	cv_ppscf		26		
242	cv_ptr_		31	2	
243	cv_ttf		26		
244	damaged_keyed_files		67		
245	damaged_keyed_files.gi		72		
246	damaged_segments.gi	damaged_segments	19		
247	damaged_sw_off	dsf	27		
248	damaged_sw_on	dsn	16		
249	date		22		yes
250	date_and_time		15		
251	date_compiled	dtc	41		yes
252	date_deleter		33		
253	date_time		25		yes
254	date_time_		28	2	
255	date_time_after	dtaf	23		yes
256	date_time_before	dtbe	23		yes
257	date_time_equal	dteq	22		yes
258	date_time_strings	date_time_strings.gi	121		
259	datebin_		261	15	
260	day		22		yes
261	day_name		22		yes

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
262	deactivate_seg		33		
263	debug	db	112		
264	decat		27		yes
265	decimal	dec	18		yes
266	decode		29		
267	decode_clock_value_		33	2	
268	decode_definition_		44	3	
269	decode_descriptor_		17	1	
270	default		14		yes
271	default_wdir	dwd	12		yes
272	defaut		8		
273	defer_messages	dm	25		
274	define_area_		15	1	
275	delay		29		
276	delete	dl	55		
277	delete_		26	2	
278	delete_acl	da	38		
279	delete_dict_words	ddw	28		
280	delete_dir	dd	63		
281	delete_external_variables	dev	16		
282	delete_iacl_dir	did	41		
283	delete_iacl_seg	dis	42		
284	delete_message	dlm	33		
285	delete_name	dn	31		
286	delete_old_pdds		26		
287	delete_search_paths	dsp	30		
288	delete_search_rules	dsr	18		
289	delete_symbols	dsb	12		
290	delete_volume_quota	dlvq	16		
291	deregister_resource	drr	23		
292	describe_list	dls	50		yes
293	detach_audit	dta	20		
294	detach_lv	dlv	13		
295	dial	d	24		
296	dial_manager_		115	7	
297	dial_manager_call		106		
298	dict.gi	dict	103		
299	directories	dirs	21		yes
300	directory		10		yes
301	directory_commands		7		
302	disban	bandis	157		
303	discard_output	dco	24		
304	display_account_status	das	27		
305	display_audit_file	daf	122		
306	display_cobol_run_unit	dcr	29		
307	display_comp_dsm	ddsm	32		
308	display_component_name	dcn	14		
309	display_dump_events		48		
310	display_entry_point_dcl	depd	73		yes
311	display_ioi_data		75		
312	display_kst_entry		24		
313	display_label		33		
314	display_lisp_object_segment		43		
315	display_list	dils	33		yes
316	display_mrds_db_access	dmdba	60		
317	display_mrds_db_population	dmdbp	51		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
318	display_mrds_db_status	dmdbs	74		
319	display_mrds_db_version	dmdv	29		
320	display_mrds_dm	dmdm	107		
321	display_mrds_dsm	dmdsm	77		
322	display_mrds_open_dbs	dmod	27		
323	display_mrds_scope_settings	dmss	31		
324	display_mrds_temp_dir	dmtd	52		
325	display_mstb		16		
326	display_pllio_error	dpe	11		
327	display_pnotice		34		
328	display_psp		41		
329	display_pvte		34		
330	display_timers		14		
331	display_tsoapl_ws	dtw	18		
332	display_ttt		31		
333	divide		12		yes
334	dmd_	mrds_dmd_	208	6	
335	do		70		yes
336	do_subtree		165		
337	documentation.gi	doc.gi documentation doc	11		
338	dot_fig.compin		50		
339	dot_fig_get_no.compin		41		
340	dot_fig_index.compin		30		
341	dot_page.compin	dot_page_off.compin	91		
342	dot_tab.compin		48		
343	dot_tab_get_no.compin		41		
344	dot_tab_index.compin		30		
345	dpl	dple	125		
346	dprint	dp	96		
347	dprint_		15	1	
348	dpunch	dpn	68		
349	dsl_	mrds_dsl_	887	24	
350	dsmd_		158	5	
351	dump_gcos	dgc	25		
352	dump_partition		49		yes
353	dump_seg_		20		
354	dump_segment	ds	130		yes
355	dump_segment_		17	1	
356	ebcdic_to_ascii_		15	1	
357	echoplex		14		
358	ed		13		
359	ed.gi		138		
360	edit_chars		18		
361	edited		18		
362	editing.gi		59		
363	edm		66		
364	emacs		40		
365	emacs.changes		304		
366	emacs.errors		409		
367	emacs.status		218		
368	encode		32		
369	encode_clock_value_		35	2	
370	enter	e	39		
371	enter_abs_request	ear	78		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
372	enter_carry_request	ecr	78		
373	enter_lss		22		
374	enter_retrieval_request	err	85		
375	enterp	ep	39		
376	entries		19		yes
377	entry		10		yes
378	ep_basic.gi	ep_basic use_ep_basic use_sp_basic	22		
379	equal		22		yes
380	equal_name	enm	15		yes
381	erkl		14		
382	error_handlers.errors		41		
383	error_table_conversion		23		
384	esc		42		
385	ex_lineno.compin	example_line_number.compin	25		
386	example.compin	example_off.compin	44		
387	examples.compin		15		
388	excerpt_mst		26		
389	exec_com	ec v2_exec_com v2ec	493		yes
390	execute_epilogue_		15	1	
391	executive_mail	xmail	22		
392	executive_mail.errors	xmail.errors	63		
393	exercise_disk		55		
394	exists		111		yes
395	expand_cobol_source	ecs	50		
396	expand_device_writer	xdw	54		
397	expand_list	els	38		
398	expand_pathname_		191	4	
399	expand_symbols	esb	17		
400	exponent_control		46		
401	exponent_control_		144	5	
402	extended_access.gi	ext_access	19		
403	external_storage.gi		20		
404	fast		18		
405	fast_topics.gi	fast_topics	65		
406	fig.compin		68		
407	fig_get_no.compin		50		
408	fig_index.compin		40		
409	fig_on.compin		45		
410	file_output	fo	37		
411	files		21		yes
412	find_condition_info_		36	1	
413	find_dict_words	fdw	37		
414	find_include_file_		45	3	
415	find_symbols	fsb	28		
416	floor		10		yes
417	format_cobol_source	fcs	28		
418	format_document	fdoc	58		
419	format_document.errors	fdoc.errors	16		
420	format_line	fl	25		yes
421	format_line_nnl	flnnl	38		yes
422	format_pl1	fp	943		
423	fort_options.gi	fort_options	122		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
424	fortran	new_fortran ft	94		
425	fortran.bugs		57		
426	fortran.gi		52		
427	fortran.new_features		931		
428	fortran_77.conversions		119		
429	fortran_77.differences		192		
430	fortran_77.gi		79		
431	fortran_abs	fa	51		
432	framing_chars		20		
433	fs_chname		27		
434	ftf.errors		8		
435	gcos	gc	110		
436	gcos.errors		36		
437	gcos.gi		83		
438	gcos_build_library	gcbl	52		
439	gcos_build_patchfile		19		
440	gcos_card_utility	gcu	105		
441	gcos_create_file		18		
442	gcos_extract_module	gcem	39		
443	gcos_fms	gfms	83		
444	gcos_library_summary	gcls	20		
445	gcos_list_patchfile		16		
446	gcos_pull_tapefile	gcpt	65		
447	gcos_reformat_syslib		32		
448	gcos_set_environment	gse	39		
449	gcos_set_tape_buffer_size	gstbs	20		
450	gcos_sysprint	gsp	33		
451	gcos_syspunch	gspn	16		
452	gcos_tss	gtss	25		
453	general_ready	gr	99		yes
454	generate_mst		230		
455	generate_pnotice		114		
456	get_authorization_		15	1	
457	get_bound_seg_info_		18	1	
458	get_default_wdir_		15	1	
459	get_definition_		17	1	
460	get_ec_version_		29		
461	get_entry_name_		16	1	
462	get_entry_point_dcl_		20	1	
463	get_equal_name_		86	2	
464	get_group_id_		28	2	
465	get_initial_ring_		15	1	
466	get_ips_mask		24		
467	get_library_segment	gls	98		
468	get_line_length_		27	2	
469	get_lock_id_		15	1	
470	get_max_authorization_		15	1	
471	get_mode		20		yes
472	get_pathname	gpn	25		yes
473	get_pdir_		15	1	
474	get_privileges_		15	1	
475	get_process_id_		15	1	
476	get_quota	gq	24		
477	get_ring_		15	1	
478	get_shortest_path_		16		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
479	get_system_search_rules	gssr	7		
480	get_temp_segment_	get_temp_segments_	15	1	
481	get_wdir_		34	1	
482	graphic_fonts.gi		103		
483	graphic_macros_	gmc_	194		
484	graphics_editor	graphic_editor ge	172		
485	greater		20		yes
486	gtss_file_attributes		14		
487	hangup		9		
488	hardcore.errors		1218		
489	hardcore_wait_events.gi		60		
490	harwell		31		
491	have_mail		16		
492	hcs_		8377	80	
493	hello		9		
494	help		347		
495	help_		54	4	
496	help_infos.gi	help_infos	23		
497	help_system.gi	help_system	60		
498	hexadecimal	hex	23		yes
499	high		10		yes
500	high9		10		yes
501	hndlquit		15		
502	home_dir	hd	10		yes
503	host_table		9		
504	hour		22		yes
505	how_many_users	hmu	31		
506	hp_delete_vtoce		56		
507	hunt		48		yes
508	hunt_dec		36		
509	hunt_dec.gi		59		
510	if		17		
511	iflow		29		
512	immediate_messages	im	28		
513	indent	ind	90		
514	index		10		yes
515	index_set		28		yes
516	info_seg.gi		232		
517	init.compin	init_plm.compin init_mpm.compin init_photo.compin	65		
518	initiate	in	38		
519	initiate_file_		145	2	
520	instance_tags.gi	instance_tag.gi instance_tag	22		
521	intermultics_copy	imc	84		
522	io_call	io	235		yes
523	io_daemon.errors	iod.errors	70		
524	ioa_		263	5	
525	ioa_control.gi	ioa_control format_line_control.gi format_line_control fl_control.gi fl_control	143		
526	iod_info_		56	4	

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
527	iox_		948	25	
528	ipc_		663	13	
529	l0exact.compin		40		
530	l0h.compin		97		
531	l0setup.compin	l1setup.compin l2setup.compin l3setup.compin l4setup.compin	42		
532	l0toc.compin	l1toc.compin l2toc.compin l3toc.compin l4toc.compin	33		
533	l1exact.compin	l2exact.compin l3exact.compin l4exact.compin	38		
534	l1h.compin	l2h.compin l3h.compin l4h.compin	77		
535	l1hbox.compin	l2hbox.compin l3hbox.compin l4hbox.compin	14		
536	l1mh.compin	l2mh.compin l3mh.compin l4mh.compin	28		
537	l1mhbox.compin	l2mhbox.compin l3mhbox.compin l4mhbox.compin	15		
538	l6_ftf		72		
539	laser		51		
540	last_message	lm	25		yes
541	last_message_sender	lms	34		yes
542	last_message_time	lmt	25		yes
543	length	ln	16		yes
544	less		18		yes
545	lex_error_		17	1	
546	lex_string_		39	2	
547	lfecho		24		
548	library_descriptor	lds	46		
549	library_fetch	lf	67		
550	library_info	li	73		
551	library_pathname	lpn	58		yes
552	ligne		6		
553	line_length	ll	11		
554	link	lk	23		
555	link_unsnap_		15	1	
556	linkage_errors.gi		106		
557	linking.gi	linking	17		
558	links		19		yes
559	linus		36		
560	linus.changes		140		
561	linus.errors		97		
562	lisp		86		
563	lisp.manual_update		778		
564	lisp_compiler	lcp	107		
565	list	ls	166		
566	list_abs_requests	lar	156		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
		list_abs_request			
567	list_accessible	lac	38		
568	list_acl	la	45		yes
569	list_carry_requests	lcr	27		
570	list_daemon_requests	ldr	63		
571	list_dict_words	ldw	29		
572	list_dir_info		45		
573	list_dir_info_		15	1	
574	list_external_variables	lev	23		
575	list_help	lh	52		yes
576	list_iacl_dir	lid	38		yes
577	list_iacl_seg	lis	38		yes
578	list_mdir	lmd	52		
579	list_mst		23		
580	list_not_accessible	lnac	27		
581	list_partitions		42		
582	list_pgs_contents	lpc	25		
583	list_pnotice_names		22		
584	list_ref_names	lrn	59		
		list_refnames			
585	list_resource_types	lrt	31		
586	list_resources	lr	81		yes
587	list_retrieval_requests	lrr	50		
588	list_sub_tree	lst	33		
589	list_symbols	lsb	23		
590	list_tape_contents	ltc	33		
591	list_temp_seg	lts	25		
		list_temp_segments			
592	lister.gi		66		
593	lister.new_features	lister_new_features	48		
594	listform_segment.gi	listform_segment	22		
595	listin_segment.gi	listin_segment	51		
596	load_control.gi	load_control	50		
597	locate_words	lw	37		
598	locnet		107		
599	logical_volumes.gi		24		
600	login	l	236		
601	logout		24		
602	long_date		22		yes
603	long_message_format	lmf	8		
604	long_year		21		yes
605	low		13		yes
606	lower_case		19		yes
607	lowercase		19		yes
608	ltrim		16		yes
609	ltsm		146		
610	ltsm.gi		115		
611	lv_attached		16		yes
612	lv_attaching.gi		11		
613	mail	ml	52		
614	mail_system.errors	print_mail.errors	606		
		read_mail.errors			
		send_mail.errors			
		mlsys.errors			
		sdm.errors			
		rdm.errors			

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
		prm.errors			
615	mail_system.gi	mail_system	28		
		mlsys.gi			
		mlsys			
616	make_commands		48		
617	make_pps_tape		29		
618	manage_volume_pool	mvp	60		yes
619	manuals.gi	manuals	253		
620	master_directories	mdirs	22		yes
621	master_directories.gi		11		
622	match_star_name_		15	1	
623	max		10		yes
624	mbx_commands		10		
625	mbx_create	mbcr	27		
626	mbx_delete_acl	mbda	23		
627	mbx_list_acl	mbla	19		yes
628	mbx_set_acl	mbsa	28		
629	mcs.errors		267		
630	mcs_version		14		yes
631	mdc_		95	7	
632	memo		221		yes
633	menu.errors		9		
634	menu_		432	9	
635	menu_create		72		
636	menu_delete		17		
637	menu_describe		31		yes
638	menu_display		21		
639	menu_get_choice		71		yes
640	menu_list		25		yes
641	merge_ascii	ma	138		
642	merge_ascii.errors		17		
643	merge_list	mls	40		
644	merge_mst		35		
645	message.compin	msg.compin	17		
646	message_facility		21		
647	message_segments.errors	ms.errors	22		
648	mhcs_		30	2	
649	min		10		yes
650	minus		10		yes
651	minute		22		yes
652	mmi_	mrds_mmi_	196	8	
653	mod		10		yes
654	mode_string_		173	6	
655	mode_string_.gi		137		
656	modes		12		
657	modify_list	mdls	37		
658	module.compin		23		
659	modules.gi	modules	41		
		module.gi			
		module			
660	monitor_log		90		
661	monitor_quota		64		
662	month		22		yes
663	month_name		21		yes
664	motd	message_of_the_day	230		
665	move	mv	69		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
666	move_abs_request	mar	85		
667	move_daemon_request	mdr	95		
668	move_dir	mvd	86		
669	move_names		20		
670	move_names_		28		
671	move_quota	mq	22		
672	mpc_data_summary		50		
673	mrds		40		
674	mrds.builtins		33		
675	mrds.changes		88		
676	mrds.cmdb_source		69		
677	mrds.cmdsm_source		99		
678	mrds.commands		44		
679	mrds.errors		250		
680	mrds.recovery		32		
681	mrds.scope		29		
682	mrds.security		53		
683	mrds.selection_expressions		76		
684	mrds.versions		189		
685	mrds_call	mrc	72		
686	mrpg		18		
687	msf_manager_		296	8	
688	msfs		21		yes
689	msmi_	mrds_msmi_	125	5	
690	mvp.errors		27		
691	nag		99		
692	nequal		14		yes
693	new_audit.gi		67		
694	new_fortran_conversions.gi		123		
695	new_fortran_differences.gi	new_fortran.differences	209		
696	new_fortran_extensions.gi	new_fortran.extensions	123		
697	new_fortran_optimizer.gi	nfopt.gi	113		
698	new_proc		30		
699	new_user		115		
700	ngreater		10		yes
701	nless		10		yes
702	no_save_on_disconnect		14		
703	noecho		8		
704	nonbranches		20		yes
705	nondirectories	nondirs	19		yes
706	nonfiles		19		yes
707	nonmaster_directories	nmdirs	23		yes
708	nonmsfs		19		yes
709	nonnull_links	nnlinks	20		yes
710	nonsegments	nonsegs	19		yes
711	nonzero_files	nzfiles	23		yes
712	nonzero_msfs	nzmsfs	23		yes
713	nonzero_segments	nzsegs	23		yes
714	not		10		yes
715	notes.compin		15		
716	nothing	nt	19		
717	null_links		20		yes
718	numeric_to_ascii_		16	1	
719	object_info_		42	3	
720	octal	oct	18		yes
721	oflow		23		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
722	old_fortran		62		
723	old_fortran.bugs		38		
724	on		82		yes
725	option_symbols	osb	66		
726	or		11		yes
727	order_manuals.gi	order_manuals	147		
728	overlay	ov	24		
729	p0f.compin	p1f.compin p2f.compin p3f.compin p4f.compin p5f.compin p6f.compin p7f.compin p8f.compin p9f.compin	28		
730	page_trace	pgt	37		
731	pagebox.compin	midbox.compin	20		
732	par.compin	p0.compin p1.compin p2.compin p3.compin p4.compin p5.compin p6.compin p7.compin p8.compin p9.compin	28		
733	par_flush.compin	pf1.compin	28		
734	par_hanging.compin	phg.compin p1h.compin p2h.compin p3h.compin p4h.compin p5h.compin p6h.compin p7h.compin p8h.compin p9h.compin	41		
735	parse_file_		152	8	
736	pascal		174		
737	pascal.gi		1108		
738	pascal_area		32		
739	pascal_bugs		16		
740	pascal_convert_source	pcs	169		
741	pascal_display		20		
742	pascal_files		9		
743	pascal_indent		81		
744	pascal_set_prompt_char		9		
745	path		10		yes
746	pathname_		127	3	
747	pathname_manipulation_af		15		
748	pause		12		
749	peo	peo1	8		
750	perprocess_static_sw_off		7		
751	perprocess_static_sw_on		8		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
752	picture	pic	34		yes
753	pind		81		
754	pl		22		
755	pll		145		
756	pll.errors		811		
757	pll_abs	pa	51		
758	pll_code_changes		171		
759	pll_io_		25	2	
760	pll_new_features		91		
761	pllr24	pllr24.gi	122		
762	pllr25		96		
763	plus		11		yes
764	polite		17		
765	pps_		50		
766	preaccess		16		
767	preface.compin	pf.compin pf_cont.compin	48		
768	prefixnl		16		
769	prepare_mc_restart_		40	3	
770	print	pr	160		
771	print_attach_table	pat	40		yes
772	print_auth_names	pan	23		
773	print_bind_map	pbm	26		
774	print_default_wdir	pdwd	6		
775	print_error_message	pem pel	8		
776	print_link_info	pli	38		
777	print_linkage_usage	plu	7		
778	print_log		96		
779	print_mail	print_mail.gi prm	118		
780	print_messages	pm	59		
781	print_motd	pmotd	14		
782	print_pdt		33		
783	print_proc_auth	ppa	20		
784	print_request_types	prt	16		
785	print_sample_refs		22		
786	print_search_paths	psp	25		yes
787	print_search_rules	psr	6		
788	print_symbols_path	psbp	6		
789	print_terminal_types	ptt	13		
790	print_tuning_parameters	ptp	29		
791	print_wdir	pwd	6		
792	print_wordlist	pwl	46		
793	printed_output	printed_output.gi	59		
794	probe	pb	19		
795	probe.gi		35		
796	process_cobol_report	pcr	46		
797	process_compout	pco	129		
798	process_dir	pd	10		yes
799	process_list	pls	67		
800	process_overseer_		8		
801	process_overseers.gi	po.gi po process_overseers	133		
802	process_preservation.gi		78		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
803	profile	pf	194		
804	program_interrupt	pi	37		
805	progress	pg	77		
806	proj_usage_report		47		
807	project_start_up_		15		
808	projet		126		
809	protection_notices.gi		80		
810	pspj		198		
811	ptsr		11		
812	qedx	qx	98		
813	query		52		yes
814	questions.gi		44		
815	quiesce_mrds_db	qmdb	48		
816	quote		11		
817	quotient		11		yes
818	random_		177	12	
819	rank		24		yes
820	rawi		12		
821	rawo		14		
822	read_allowed_		32	1	
823	read_list_		98		
824	read_mail	rdm	188		
825	read_password_		15	1	
826	read_tape_and_query	rtq	231		
827	read_tsoapl_tape	rtt	41		
828	read_write_allowed_		30	1	
829	ready	rdy	17		
830	ready_off	rdf	6		
831	ready_on	rdn	6		
832	rebuild_dir		25		
833	record_stream_		60		
834	record_to_sector		17		yes
835	record_to_vtocx		42		
836	register_resource	rgr	69		
837	release	rl	14		
838	release_area_		15	1	
839	release_resource	rlr	29		
840	release_temp_segment_		15	1	
841	release_temp_segments_		16	1	
842	remove_graphics	rg	18		
843	rename	rn	21		
844	reorder_archive	ra	30		
845	repeat_query	rq	10		
846	replay		20		
847	reprint_error	re	20		
848	request.compin		23		
849	request_ids.gi	request_ids	42		
850	request_list.compin	rqst_list.compin	20		
851	reseau		1641		
852	reserve_resource	rsr	63		
853	reset		9		
854	reset_external_variables	rev	20		
855	reset_ips_mask		33		
856	resetcopysw		12		
857	resolve_linkage_error	rle	35		
858	resource_report		73		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
859	resource_status	rst	83		yes
860	resource_usage	ru	22		
861	response		36		yes
862	retain_symbols	rsb	27		
863	retrieve		177		
864	reverse	rv	10		yes
865	reverse_after	rvaf	26		yes
866	reverse_before	rvbe	26		yes
867	reverse_decat	rvdecat	34		yes
868	reverse_index	rvindex	26		yes
869	reverse_search	rvsrh	26		yes
870	reverse_verify	rvverify	26		yes
871	revert_output	ro	21		
872	revise_words	rw	46		
873	ring0_get_		66	5	
874	ring_zero_peek_		205	5	
875	rtrim		16		yes
876	ruban		10		
877	run		40		
878	run_		13	1	
879	run_cobol	rc	27		
880	runoff	rf	162		
881	runoff_abs	rfa	56		
882	runoff_compose.differences	rf_comp.differences rf_comp.diffs	138		
883	safety_sw_off	safety_switch_off ssf	13		
884	safety_sw_on	safety_switch_on ssn	13		
885	sample_refs		32		
886	save_dir_info		21		
887	save_on_disconnect		13		
888	scroll		22		
889	sdmc		74		
890	search		11		yes
891	sector_to_record		17		yes
892	secure_mrds_db	smdb	40		
893	segments	segs	21		yes
894	select		34		yes
895	send_mail	sdm	259		
896	send_mail_		56	4	
897	send_message	sm	32		
898	send_message_		54	4	
899	send_message_acknowledge	sma	33		
900	send_message_express	smx	33		
901	send_message_silent	sms	34		
902	set_acl	sa	87		
903	set_bit_count	sbc	16		
904	set_bit_offset_		34	1	
905	set_cc		34		
906	set_char_offset_		37	1	
907	set_dir_ring_brackets	sdrb	37		
908	set_fortran_common	sfc	18		
909	set_iacl_dir	sid	28		
910	set_iacl_seg	sis	30		
911	set_ips_mask		33		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
912	set_lock_		82	2	
913	set_max_length	sml	39		
914	set_mdir_account	smda	17		
915	set_mdir_owner	smdo	17		
916	set_mdir_quota	smdq	21		
917	set_mrds_temp_dir	smtd	31		
918	set_resource	setr	81		
919	set_ring_brackets	srb	41		
920	set_search_paths	ssp	41		
921	set_search_rules	ssr	53		
922	set_severity_indicator	ssi	15		
923	set_system_storage		23		
924	set_time_zone	stz	52		
925	set_translator_search_rules	stsr	16		
926	set_ttt_path		21		
927	set_tty	stty	139		
928	set_tty.gi	stty.gi	62		
929	set_user_storage		23		
930	set_volume_quota	svq	18		
931	setcopysw		12		
932	setup_graphics	sg	57		
933	severity		42		yes
934	severity_indicators.gi		40		
935	short_message_format	smf	8		
936	show_symbols	ssb	12		
937	signal_		15	1	
938	slave		14		
939	slug_off.compin		14		
940	sort	merge	183		
941	sort_items_		61	5	
942	sort_items_indirect_		93	6	
943	sort_list	sls	37		
944	sort_seg	ss	58		
945	special_active_functions		16		
946	speedtype.gi	speedtype	58		
947	star_equal.gi	star_equals star_equal star_equals.gi equal_names star_names	78		
948	star_system_links.gi		79		
949	start	sr	22		
950	start_up.ec		23		
951	static_handlers.gi	static_handlers	20		
952	status	st	154		yes
953	stop_cobol_run	scr	16		
954	stop_run		7		
955	storage_system_af		18		
956	string		13		yes
957	strip		18		yes
958	strip_entry	spe	18		yes
959	sub_err_		85	1	
960	substr		14		yes
961	suffix		10		yes
962	suffixed_name_		29	2	
963	sweep_disk_		60	2	

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
964	switch_off	swf	53		
965	switch_on	swn	53		
966	syn_output	so	23		
967	system		99		yes
968	system_commands		461		
969	system_functions		120		
970	system_info_		1134	23	
971	system_logs.errors		8		
972	system_variables		229		
973	tab.compin		47		
974	tab_get_no.compin		50		
975	tab_index.compin		40		
976	tab_on.compin		45		
977	tabecho		15		
978	tabs		40		
979	tape_archive	ta	157		
980	tape_control_language.gi	tcl.gi	215		
981	tape_i/o.errors	tape.errors	8		
982	tape_in	tin	37		
983	tape_nstd_		146		
984	tape_out	tout	45		
985	tc_io_		87		
986	teco		103		
987	teco.errors		27		
988	ted		321		
989	ted.errors		41		
990	ted_msgs		6		
991	term_		62	5	
992	terminal_output	to	18		
993	terminal_type	ttp	11		
994	terminate	tm	38		
995	terminate_commands		11		
996	terminate_file_		64	1	
997	terminate_refname	tmr	41		
998	terminate_segno	tms	40		
999	terminate_single_refname	tmsr	42		
1000	test_archive		12		yes
1001	test_io_daemon		25		
1002	test_tape		80		
1003	texto		59		
1004	time		22		yes
1005	timer_manager_		228	11	
1006	times		11		yes
1007	titlepage.compin	tp.compin	50		
1008	toc_on.compin		39		
1009	topics.gi	topics topic.gi topic	35		
1010	total_cpu_time_		15	1	
1011	total_output_requests	tor	59		yes
1012	tp.changes		305		
1013	tp_cancel		16		
1014	tp_change_deadline		17		
1015	tp_cvsct		14		
1016	tp_display_command_table	tpdct	13		
1017	tp_display_current_xcns	tpdcx	11		

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
1018	tp_display_input_queue	tpdiq	13		
1019	tp_display_master_table	tpdmt	13		
1020	tp_display_output_queue	tpdoq	13		
1021	tp_get_xcn_status	tpgxs	32		
1022	tp_io_start		30		
1023	tp_list_pending_requests	tplpr	26		
1024	tp_meters		22		
1025	tp_pre_create.ec	tp_pre_create	11		
1026	tp_reset_xcn_num		19		
1027	tp_rollback_transaction_		19	1	
1028	tp_shrink_q		37		
1029	tp_start		27		
1030	tp_stop		17		
1031	tp_user		46		
1032	tp_verify_transaction_		22	1	
1033	tp_who		27		
1034	tp_worker_init_tcf		16		
1035	tp_worker_start		17		
1036	trace		185		
1037	trace_meters	tmt	49		
1038	trace_stack	ts	33		
1039	transaction_call	trc	54		
1040	transaction_call_		84	6	
1041	translate		13		yes
1042	translate_ids_schema	tis	22		
1043	translator_info_		15	1	
1044	trim_list	tls	49		
1045	trim_wordlist	twl	31		
1046	trunc		11		yes
1047	truncate	tc	33		
1048	tss_basic.gi	tss_basic	105		
1049	tss_fortran.gi	tss_fortran	125		
1050	tssi_		79	6	
1051	ttt_info_		112	8	
1052	tty.new_features		83		
1053	tty_		445		
1054	tty_modes.gi		194		
1055	unassign_resource	ur	22		
1056	underline		11		yes
1057	unique		18		yes
1058	unique_bits_		15	1	
1059	unique_chars_		15	1	
1060	unlink	ul	34		
1061	unpopulate_mrds_db	umdb	43		
1062	unwinder_		15	1	
1063	update_mrds_db_version	umdbv umdv	55		
1064	upper_case	uppercase	30		yes
1065	use_symbols	usb	13		
1066	user		147		yes
1067	user_ftp	ftp	220		
1068	user_info_		669	19	
1069	user_ring_io.errors		13		
1070	user_telnet		195		
1071	util.fortran.gi		73		
1072	vl_exec_com	vlec	160		yes

num.	Commandes	Synonyme(s)	lines info	entry pts	act. func.
1073	v1_exec_com.differences	v1ec.diffs	286		
1074	v2_exec_com.differences	v2.diffs v2ec.diffs	228		
1075	valid_decimal_		15	1	
1076	valid_pictured_data	vpd	19		
1077	validate_info_seg	vis	79		yes
1078	value_		634	12	
1079	value_defined	vdf	41		yes
1080	value_delete	vdl	91		
1081	value_get	vg	73		yes
1082	value_list	vls	115		yes
1083	value_path	vp	15		yes
1084	value_set	vs	155		yes
1085	value_set_path	vsp	34		
1086	values_afs		12		
1087	verify		20		yes
1088	vfile_		1090		
1089	vfile_adjust	vfa	32		
1090	vfile_find_bad_nodes		212		yes
1091	vfile_status	vfs	30		
1092	vfile_status_		15	1	
1093	video.errors		42		
1094	video_data_		34	1	
1095	video_utils_		74	2	
1096	virtual_cpu_time_		15	1	
1097	virtual_entries.gi	virtual_entries virtual_entry	35		
1098	virtual_pointers	virtual_pointers.gi virtual_pointer	54		
1099	volume_dump_switch_off	vdsf	27		
1100	volume_dump_switch_on	vdsn	27		
1101	volume_names.gi		53		
1102	vtoc_pathname		45		
1103	vtocx_to_record		19		yes
1104	walk_subtree	ws	34		
1105	where	wh	76		yes
1106	where_search_paths	wsp	26		yes
1107	who		75		yes
1108	window_		1598	22	
1109	window_call	wdc	268		yes
1110	window_io_		203		
1111	working_dir	wd	10		yes
1112	write_allowed_		32	1	
1113	wsfns		201		
1114	year		22		yes
1115	zero_segments	zsegs	23		yes