MULTICS AT IPS

Information Processing Services
Massachusetts Institute of Technology
Memo MS-1

December 1, 1980

This memo replaces the previous MS-1 along with memos MS-7 and MS-18. The material has been extensively revised; this document, therefore, does not contain further marginal change indicators.

[illegible]

This memo provides IPS-specific information about Multics. It is not intended to be read linearly. The first section guides you to the Honeywell documentation and gives the dial-in phone number so you can get started. After you've read the introductory Honeywell manuals, you can come back to this one. Everyone should read Sections II, III, IV, and V, which cover such subjects as on-line consulting, access control, and off-line printing. People who use cards, want to retrieve lost data, or use Multics via networks should, in addition, read the appropriate sections on those topics.

The following documents are cited in this memo and are useful for additional information:

VENDOR DOCUMENTATION

New User's Introduction to Multics--Part I (CH24)
New User's Introduction to Multics--Part II (CH25)
Multics Programmers' Manual, or MPM:

Reference Guide (AG91)
Commands and Active Functions (AG92)
Subroutines (AG93)
Subsystem Writers' Guide (AK92)
Peripheral Input/Output (AX49)
Communications Input/Output (CC92)

Multics Pocket Guide Commands and Active Functions (AW17)
How to Use Telenet
Telenet U.S. Access Locations
How to Use TYMNET
TYMNET Telephone Numbers

IPS DOCUMENTATION

IPS User's Guide	
MS-12	Error Analysis and Recovery on Multics
MS-51	The Multics Installation-Maintained Library
MS-52	The Multics Author-Maintained Library
MS-99	Multics Bulletin Article Reprints
AP-1	Application Index and Descriptions

CONTENTS

I. INTRODUCTION

Multics is an easy-to-use, yet sophisticated and powerful operating system which IPS runs on its Honeywell 68/DPS computer. Multics is marketed by Honeywell, which does a fairly good job of explaining what Multics is and how to work it. We do not intend this memo to duplicate that effort. We urge you to buy and study Honeywell's documentation so you can learn to use Multics. This memo merely provides information about Multics that is specific to IPS, and touches on some topics given short shrift in the Honeywell documentation.

The rest of this section is not an explanation of how to get started on Multics, but rather a guide on how to get to the starting line.

THE BRASS TACKS

Before you can use Multics, you must have a Multics account with the IPS User Accounts Office (Room 39-213). Setting up an account is explained in the _IPS User's Guide_. Regardless of whether you set up the account or someone else does it for you, there are three things established at registration time that you must know before you can go any further: your person_id (the name that identifies you to Multics), your project_id (a short name for the group of users you're associated with), and a password.

The next thing you need is documentation. Start with Honeywell's _New Users' Introduction to Multics--Part I_ (Honeywell document CH24). It is available at the IPS Publications Office (Room 39-233). The manual covers nearly everything you need to know to get started working on Multics. Everything except our phone number.

THE PHONE NUMBER

To use a computer interactively, you type instructions, data, or whatever at a terminal. Some terminals are wired directly to a computer, but most must be connected via a telephone call. The _New User's Introduction to Multics--Part I_ explains in detail the process of connecting to Multics, but doesn't tell you what phone number to dial (because the number is specific to MIT, while the manual isn't). Some people, so that they can use Multics from afar without paying long-distance telephone charges, go through still another intermediary known as a network. If you are such a person, be sure to read Section IX, "Using Multics via Networks", for connection instructions.

To use IPS's Multics, dial:

258-8311

This number accommodates connections to Multics from any terminal in common use nowadays.* The number is the lowest number in a "hunt group". Since Multics cannot communicate with all its users via the same line, we provide many lines, which correspond to phone numbers in the range 258-8311 to 258-8366. If you call a line that is already in use, the computer automatically hunts this group for the next available line. You will hear a high-pitched tone when Multics is ready to "talk" to your terminal. When you've connected your terminal to Multics (see Section 2 of the _New Users' Introduction_) and are waiting for the Multics "greeting" message, press the RETURN key; this allows Multics to determine the transmission rate of your terminal and properly communicate with you during your session.

But what if you don't get an answer or high-pitched tone when you call? This may be because the hunting mechanism is not working properly, in which case you can still connect by trying a slightly higher number in the range. On the other hand, the computer may be unavailable ("down"), due to a "crash" or a scheduled shut-down. To find out if this is the case, call our recorded status message at 258-7700.

BON VOYAGE

Since the best way to learn any system is by using it, go get the _New Users Introduction to Multics--Part I_, read it carefully, find a terminal, and start using Multics. Let us only detain you long enough to say where you can get more information.

Consulting

If you need help, you can talk to an IPS consultant in person or by telephone. See the _IPS User's Guide_ for details on hours, phone numbers, and policies.

You can also talk to a consultant through Multics. By using the "olc" command, you can send a message from your terminal to an IPS consultant. See Section III or type "help olc" if you're

---

*As of this writing, IPS has phone lines for two other kinds of rarely-seen terminals. If you have a Selectric-type terminal which transmits at 134.5 baud (such as an IBM 2741), call 258-8215. If you are hooking in via a BELL 202-compatible modem, use 258-6845. Anyone else, including those using BELL 212-type and VADIC modems, should use 258-8311.

logged in, to learn how to use the "olc" command. The consultant can often answer your question by sending a message to your terminal.

Documentation

Multics documentation is extensive. The following IPS or Honeywell documents give general information about using Multics:

New Users' Introduction to Multics--Part I (CH24), as has been said so many times above, covers most of what a beginning user needs to know to use Multics. Essential.

New Users' Introduction to Multics--Part II (CH25), the sequel to the previous manual. Is there a class of Multics users who should read Part I but not Part II? If your work on Multics is limited to some self-contained subsystem, you can probably do without Part II. But nearly everyone else--programmers, for instance--should look into it.

Multics Pocket Guide Commands and Active Functions (AW17), a handy pocket-sized booklet which describes Multics commands and details their syntaxes. Nice for use at the terminal.

Multics Programmer's Manual (MPM), which comprises six sections:

Reference Guide (AG91)
Commands and Active Functions (AG92)
Subroutines (AG93)
Subsystem Writers' Guide (AK92)
Peripheral Input/Output (AX49)
Communications Input/Output (CC92)

The MPM contains reference descriptions of the Multics standard system software. This is the place to find complete, detailed information about Multics. For example, to learn the full capabilities of a command you've read about in the New Users' Introduction, see the MPM Commands and Active Functions (AG92). The MPM is big and expensive. (Its parts come separately--still, some of them are big and expensive.) Casual users should not buy it. Nevertheless, most users should have access to one (the IPS Reading Room, 39-233, has a copy). For sophisticated users, it is the bible; you cannot know the full power and flexibility of Multics without looking into the MPM.

The Multics Installation-Maintained Library (IPS Memo MS-51, forthcoming), which contains descriptions of the additional system facilities that have been designed and maintained by IPS. Memo MS-51 should be used as a supplement to the MPM Commands and Active Functions (AG92).

The Multics Author-Maintained Library (IPS Memo MS-52), which contains detailed descriptions of additional system facilities designed and maintained by MIT users. Like MS-51, MS-52 should be used as a supplement to the MPM Commands and Active Functions (AG92).

Multics Bulletin Article Reprints (IPS Memo MS-99), a series of pieces previously published in the IPS Bulletin that are of interest to Multics users. For example, novice users should read "Things That Nobody Told You About Your start_up.ec", while advanced users might enjoy "Writing Multics Commands".

Application Index and Descriptions (IPS Memo AP-1), a guide to software meant for specific tasks (such as numerical methods, statistics, and word-processing). AP-1 covers all application software available at IPS, not only that available on Multics.

Now go read the New Users' Introduction to Multics and come back to this document when you've finished.

II. LOCAL CUSTOMS

While the "standard" Multics system looks much the same at IPS as at any other Multics installation, there are aspects of the system that IPS can set as it chooses. These site-dependent defaults can make the system behave somewhat differently than what you might expect from reading the Honeywell documentation. This section covers several features of the Multics environment peculiar to IPS.

THE DEFAULT start_up.ec

A start_up.ec, as you know from reading New Users' Introduction to Multics--Part II, is a collection of commands that is executed each time you login. Typically, it sets aspects of your Multics environment that you want in effect for the duration of your session and handles certain chores, like checking mail, that are best done once per session.

If you log in and you do not have a start_up.ec (and your project does not have a default start_up.ec), Multics runs the system's default start_up.ec for you. Among other things, it prints the message of the day, enables you to receive messages from other users, and informs you if you have any mail. Be aware that, as it accepts messages, it creates a mailbox for you if you do not have one already. A mailbox is a segment, whose name is your person_id with the suffix ".mbx" appended, that serves as a holding area for messages and mail sent to you by the system or other users. It takes up one record of storage in your home directory. So don't be surprised to see the mailbox in the output when you give the "list" command.

The pathname of the default start_up.ec is >udd>sc1>start_up.ec.

ECHOPLEX MODE

When you're working at a terminal logged in to Multics, and you press, for instance, the "Q" key, a signal which corresponds to that letter is sent to Multics. But it is useful if the character is also printed at your terminal so you can see what you typed; this is called echoing. Echoing may be done by the terminal or by the computer. In the first case, called "local echoing", your terminal prints the character while sending it to Multics. However, it is also possible for Multics to transmit a "copy" of every character it receives back to the terminal; the character is then displayed at your terminal just like any other output generated by Multics. (Of course, this happens so fast that your keystroke seems to be immediately responsible for the appearance of the letter.) You can have Multics perform echoing or not; when it does, it is said to be operating in "echoplex mode".

The trend nowadays is to let the computer do the echoing. Advanced screen editors, like Emacs, depend on computer echoing, and some new terminals, such as the VT100, are incapable of local echoing. Accordingly, IPS has made echoplex mode a system default. Your terminal, therefore, should be set to full duplex mode. If it isn't, you will see each character you type echoed twice--once by the terminal and once by Multics. If you do not want to run in full duplex mode, you may turn off echoplex mode before you give the "login" command by typing:

```
modes ^echoplex
```

Alternatively, you can give the following command after you log in (or put it in your start_up.ec):

```
stty -modes ^echoplex
```

PROCESS PRESERVATION

Users of time-sharing systems occasionally encounter the annoyance of accidental disconnection. Say you're working on Multics, diligently typing in some data, when a clumsy colleague bumps into your acoustic coupler and disconnects you from the computer before you've had a chance to "save" your work. On many systems, such work would be gone for good. However, Multics can hold pending work in suspension when you are disconnected and give you a chance resume your work from the point of interruption. Thus, all is not lost when you are disconnected by a terminal, phone, or network failure. (You can not, of course, recover work lost when Multics itself "crashes".) At IPS, your Multics sessions are automatically subject to this "process preservation" facility.

Reconnecting to a suspended process is not difficult. Just log back; Multics coaches you on what to do. There are, however, some subtle points that can make reconnecting go less than smoothly. See Section for all the details.

For now, keep in mind two things. First, you will get a disconnected process--your interrupted session's disembodied self--if you are disconnected accidently or if you deliberately hang up without logging out. Second, any disconnected process you have will chew up one more hour of connect charges before Multics concludes you are not going to reconnect. (In fact, disconnected processes can be kept alive even longer--see Section .)

One last thing: process preservation does not yet work for users logged in over ARPANET. ARPANET users, to prevent getting disconnected processes that they cannot reconnect to, should use the "-nosave" control argument when they log in or put the following in their start_up.ecs:

```
          &if [equal [user line_type] TELNET]
          &then no_save_on_disconnect
```

III. IN AND OUT OF TROUBLE

If you need help when working on Multics, assistance is often at your fingertips. This section looks at some of the on-line facilities for getting help, including the IPS-written commands for on-line consulting, error analysis, and problem reporting.

INFO SEGMENTS

There is a lot of information about Multics kept in publicly available files (called "info segments"). To peruse this on-line documentation, use the "help" command; it locates the file and prints portions of it at your terminal. To find out, for example, about the "dprint" command, type:

```
help dprint
```

When you don't know exactly the name of the file you're looking for, you can use the "list_help" command to list the info segment names that match a word (or word fragment) you specify. Thus, to list the help files that pertain to networks, you might type:

```
list_help net
```

See the New Users' Introduction to Multics--Part I (CH24) for more information on the "help" and "list_help" commands.

While the bulk of the info segments are descriptions of Multics commands, both "help" and "list_help" can locate and print general-information files. IPS provides a number of these. You can type "help rates" to see our current rates, or "help schedule" to find out the operating schedule. Among other topics that are covered by info segments are: "news", a log of changes to system; "pending_changes", which lists significant changes planned for the future; and "pl1.status", which tells about new features and bugs in the PL/I compiler.

ON-LINE CONSULTING

If you are working on Multics and become stumped or confused by some aspect of the system, you can send a distress cry to an IPS consultant by using the "online_consultant" (or "olc") command. Like the "send_message" command, "olc" can be used in either of two ways. If you type your message on the same line as the command, "olc" sends your message to the on-line consultant and returns to command level. For example:*

```
     => olc  How can I allow colleagues to read my mailbox?
        r 14:11 0.201 73
```

Alternatively, you can type "olc" on a line by itself. Then everything you type is sent to the consultant until you enter a period (".") on a line by itself. For example:

```
     => olc
        Input.
     => Can I enable other people on the same Project
     => as me to read stuff in my mailbox?  I tried
     => set_access but that doesn't work.
     => .
        r 14:11 0.504 95
```

This second mode is intended to facilitate user-consultant conversation. Don't be surprised if you receive replies from the consultant while you are still entering message input lines. A typical dialogue:

```
     => olc
        Input.
     => I get the message "Error:  Coolant underflow condition
     => by chevy$engine|396 (>udd>RACER>RPetty>chevy)"
     => This program has always run ok in the past.
        From Scott.ARCS (olc)  05/30/80  1011.0 edt Mon:
        Do you call the system subroutine cu_$radiator?
     => Yes.
        From Scott.ARCS (olc) 05/30/80  1012.7 edt Mon:
        The arguments have changed. Declare them fixed bin(21)
     => OK.  I'll check it out.  Thanks.
     => .
        r 10:18 0.559 245
```

Remember, to stop transmitting to the consultant, type a period all by itself at the beginning of a line, followed by a carriage return. If you don't, anything you type continues to be sent to the on-line consultant until he or she is able to get a word in edgewise and tell you what to do, or until you hit the BREAK key.

On-line consulting is available every weekday from 9:30 AM to 12:30 PM and 1:30 AM to 4:30 PM (and often at unscheduled times). If no consultant is signed on when you use "olc", the command types a message to inform you of this and saves your message for the next consultant. You can find out whether a consultant is signed on or the next time one is scheduled by using the

---

*Throughout this memo, lines that you type are preceded by an arrow (=>), to distinguish them from Multics responses.

"online_consultant_status" (or "olcs") command. IPS Memo MS-51 (forthcoming) gives full information on the "olc" and "olcs" commands. If you have a Multics question and you're not logged in (or simply don't want to use the "olc" command), you can call 253-8424.

ON-THE-SPOT ERROR INTERPRETATION

The "explain_error" command ("ee" for short) can help you with errors that you encounter when running programs. Suggestions for correcting the problem and preventing future occurrences are given.

The "explain_error" command can only help you with errors that are reported in messages beginning with "Error:". Unfortunately, it can't interpret other error messages, such as those reported by the system program "com_err_", which simply prints a message without creating an error condition. Typically, such error messages are preceded by the name of the program encountering the error. For instance:

```
=> print last_minute_votes
   print: Entry not found. >udd>CARTER>Jimmy>last_minute_votes
   r 20:00 1.980 4
```

In general, the errors reported by "com_err_" are self-explanatory.

Let's suppose, however, that the last command you typed, instead of producing the expected output, stopped abnormally, leaving you at a new command level and generating the following message:

```
Error: record_quota_overflow condition by put_in_bank|20
(>udd>OILCORP>Glutton>rake_in_profits)
referencing >udd>OILCORP>Glutton>our_piggy_bank
r 20:06 7.622 11 level 2
```

Try typing:

```
explain_error
```

Invoked with no arguments, "explain_error" looks for the most recent error that occurred in your session and types the explanation via the "help" command's routines. During long explanations, these routines pause and ask if you want more help; respond with any valid "help" reply, such as "yes", "no", or "skip".

Running programs or commands after the error is reported may prevent "explain_error" from finding it. In this case, you can get an explanation by explicitly giving "explain_error" the name of the error. For instance, to get information on the error in the example above, type:

```
        explain_error record_quota_overflow
```

Type the error name exactly as Multics did. By giving the name of the error as an argument in this way, you can get an explanation for an error condition at any time.

Information is not currently available for all errors; however, IPS keeps a log of errors for which an explanation is requested but not available. We will write new explanations where needed.

For full information on "explain_error", see IPS Memo MS-51 (forthcoming). For more help in understanding Multics errors, see Memo MS-12.

TROUBLE REPORTING

Although IPS tries hard to keep its systems problem-free, you may occasionally encounter what you consider to be a system bug. Or you may have a request, comment, or suggestion to pass along to IPS. Use the "trouble_report" (or "tr") command to report problems or make suggestions from your terminal. For example:

```
     => trouble_report
        Your name?  => Sal Dali
        Your phone number?  => (801) 333-1801
        Your address?  => Barcelona
        Now describe your problem.
        Begin each line representing console printing with a *.
        Type a "." on a new line when done.
     => Something is wrong with the "time" command.
     => Whenever I give this command, my terminal melts into
     => a pool of silly putty and drips onto my lap.
     => .
        How urgent is
        your problem?  => Very, I'm running out of terminals.
        Can a reply be sent to your Multics mailbox?  => yes
        Are you sending labeled console output to
        to the Consultant, 39-421?  => no
        On one line give path names of relevant segments
        (set the ACLs to re *.*.*):
     => >udd>ModART>Dali>watches.ec
        r 15:21 0.888 56
```

IPS will process these reports and get back to you. We encourage you to use this facility; we cannot fix a bug unless we know it exists. We also want to know which of our services you consider most important and what new ones you would like to see. We can't promise to provide every service requested, but we will consider each request.

For more information on the "trouble_report" command, see IPS Memo MS-51 (forthcoming).

IV. ACCESS

The bulk of this section concerns access to entries in the storage system--how you control who can look at or modify a particular segment or directory belonging to you. The secure yet flexible access control that Multics provides, however, is worthless if some rascal can log in and use Multics under your person_id. So let us digress a moment to discuss account security.

The security of your account depends on your password. (If you don't know what a password is, you are out of your depth--go back to Section I.) Never tell anyone your password. Change it regularly. If you suspect that someone has discovered your password, change it immediately.

To learn how to change your password, see Section II of the New Users' Introduction to Multics--Part I (CH24), type "help login", or check out the description of the "login" command's "-change_password" control argument in the MPM Commands and Active Functions (AG92). Keep in mind that you have only one password, even if you are registered on more than one project. If you have forgotten your password, ask for a new one at the IPS User Accounts Office (Room 39-219, 253-4118).

Now let us turn to access control for segments and directories.

ACCESS CONTROL LISTS

Every segment and directory on Multics has its own access control list (ACL). The ACL is a list of user identifiers, each with corresponding access modes. For example, the ACL for a segment might be:

```
rw    Juliet.Capulet.*
r     Nurse.Capulet.*
rw    *.SysDaemon.*
```

"Juliet.Capulet.*" is a user identifier, while "r" represents an access mode. To determine your access to a given segment or directory, Multics tries to match you to a user identifier and accords you the corresponding access modes.

The user identifier consists of three parts: a person_id, a project_id, and an instance tag. (The instance tag is a letter that reflects how a user is using Multics. For interactive users, the instance tag is "a"; for absentee users, "m"; and for system processes or "daemons", "z".) An asterisk in place of one of the parts means that Multics can ignore that component when checking whether you match the user identifier.

Suppose your person_id is "Juliet", your project is "Capulet", and you are logged in at a terminal. (These assumptions about you will be carried on throughout this section.) Thus, you are known to Multics as "Juliet.Capulet.a", and would match any of the following user identifiers in an ACL:

```
Juliet.Capulet.a
*.Capulet.*
Juliet.*.a
Juliet.Capulet.*
*.*.*
```

(Note that "*.*.*" matches any user.) You would NOT match:

```
*.CAPULET.*
Juliet.Capulet.m
Juliet.Montague.*
Tybalt.Capulet.*
```

Multics arranges the user identifiers in an ACL in order of specificity, considering the person_id, project_id, and instance tag in that order. If you match more than one user identifier, the first of these--the most specific one--determines your access. For example, suppose the ACL for a segment is:

```
r     Juliet.Capulet.a
re    Juliet.Capulet.*
rew   Juliet.*.a
rew   *.Capulet
re    *.*.*
```

Although you match all the entries in this ACL, you have only "r" access, determined from the access mode of the first entry, not the broader "rew" access given in some of the other entries.

But what's the meaning of these access modes, anyway? Since they depend on whether the ACL is for a segment, a directory, or a mailbox (which, although it is a segment, is treated specially), they are discussed in the appropriate sections below.

ACCESS TO SEGMENTS

Access Modes for Segments

The four access modes that pertain to segments (except for mailboxes) are:

r	(read)
e	(execute)
w	(write)
n	(null)

You need "r" access to a segment to examine or copy its contents. To execute a segment (provided it is a compiled program), you need "e" access to it, as well as "r" ("e" access alone is not enough). With "w" (along with "r") access to a segment, you can change its contents as well as look at it. If you have "n" access to a segment, you can't do anything with it. Access on a segment can be any one of these or a combination such as "re", "rew", or "rw". Access mode "n" cannot be given in combination with any of the others.

Controlling Segment Access

To print a segment's ACL, use the "list_acl" command. For example, to see the ACL for a segment named "balcony", type:

```
=> list_acl balcony
   rw    Juliet.Capulet.*
   r     *.Capulet.*
   n     *.Montague.*
   rw    *.SysDaemon.*
   r 10:23 0.873 9
```

Your access is derived from the access modes listed for the first user identifier in the ACL that matches you. Thus you, as Juliet, have "rw" access to "balcony", while any other user on "Capulet" project has "r" access. If you do not match any user identifier in the ACL, your have "null" access. Therefore, the "null" access set for "*.Montague.*" in the above ACL is overkill.

You can use "list_acl" to see the access for a particular user identifier by typing it after the segment name:

```
=> list_acl balcony *.Capulet
   r     *.Capulet.*
   r 10:24 0.913 8
```

If you omit the project_id or the instance tag, "list_acl" takes it to be "*".

To set access for someone, use the "set_acl" command. For instance, to add "rw" access for "Romeo.Montague", type:

```
          set_acl balcony rw Romeo.Montague
```

You can also use "set_acl" to change the access for a user identifier already in the ACL.

You can specify more than one pair of access modes and user identifier in a command line. Be sure to specify the access modes <u>before</u> each user identifier. For example:

```
          set_acl balcony rew Romeo.Montague re Nurse.Capulet
```

To remove a user identifier and its access modes from an ACL, use the "delete_acl" command. For example, to delete the access to "balcony" of "*.Capulet", type:

```
          delete_acl balcony *.Capulet
```

You can use the star convention in the segment name with any of these commands.

To use the "list_acl" command, you need at least "s" access to the directory containing the segment of interest. To use "set_acl" and "delete_acl", you need at least "sm" access to the containing directory.

Suggested Access Settings for Segments

People often give "r" or "re" access on all their segments to "*.*.*" (everyone), except for segments containing private information. Unless you give other users "w" access to a segment, there is no danger of their changing its contents. If you are registered on more than one project, you may find it convenient to give "rw" access on your segments to "Juliet.*.*"--assuming Juliet is your person_id--allowing you to use them from any project.

Giving a user only "r" access to a program does not prevent him from executing it; he can copy it into his own directory and execute that copy.

By default, "*.SysDaemon.*" is given "rw" access to all your segments. This is necessary in order to print segments on the line printer, back them up on tape, etc. You should not delete this access; it presents no security problems for most users.

ACCESS TO DIRECTORIES

Access Modes for Directories

The four access modes that pertain to directories are:

s	(status)
m	(modify)
a	(append)
n	(null)

You need "s" access to find out the names and characteristics of the entries in a directory (though not necessarily their contents). With "m" access, you can change the characteristics of or delete existing entries; "a" lets you add new entries. If you have "n" access to a directory, you cannot do anything to it. Access on a directory may be any one of these or a combination, such as "sm", "sa", or "sma". Access mode "n" cannot be given in combination with any of the others.

Having "n" access to a directory doesn't prevent you from using segments in that directory, if you have appropriate access to them. For example, you can't list the segments in a directory without "s" access to the directory, but you can print a particular segment if you have "r" access on the segment. The catch is that you have to know the name of the segment beforehand.

Controlling Directory Access

To list, set, and delete access to directories--as with segments--you use the commands "list_acl", "set_acl", and "delete_acl". The only difference between their use with segments and with directories is the access specification: trying to give someone "rw" access to one of your directories just won't work.

If you give the "list_acl" command with no arguments, it prints the ACL for your working directory.

In general, you have "sma" access to your home directory, but not to your project directory (the next higher one). This means that you cannot change the ACL for your home directory; if you need to do so, contact your project administrator.

To use the "list_acl" command, you need at least "s" access to the directory just above the directory of interest. To use "set_acl" and "delete_acl", you need at least "sm" access to the higher directory.

Suggested Access Settings for Directories

You may wish to give "s" access on your directories to "*.*.*". This allows anyone to print the name and ACL of entries in the directory; it does not give them automatic access to the contents of segments in the directory. If you are registered on more than one project, you may find it convenient to have your project administrators set "sma" access for "Juliet.*.*" in each of your home directories (assuming your person_id is "Juliet"). This permits you to work with any of your directories regardless of what project you are logged in on.

By default, "*.SysDaemon.*" has "sma" access to your directories. As with segments, this access is necessary to perform various system chores; you should not delete it.

ACCESS FOR MAILBOXES

Access Modes for Mailboxes

The group of access modes that apply to mailboxes is known as extended access. The seven modes of extended access are:

a	(append)
d	(delete)
r	(read)
o	(own)
s	(status)
w	(wakeup)
n	(null)

You need "a" access to a mailbox to add messages to it. Modes "d" and "r" give you access to anyone's messages in the mailbox; "d" lets you delete them, and "r" lets you read them. On the other hand, "o" access to a mailbox lets you delete and read only your own messages (i.e., messages that you have sent). With "s" access to a mailbox, you can find out how many messages it contains. If you have "w" access, you can send a "wakeup" to a user accepting messages from the mailbox; that is, you can send him interactive messages. If you send messages without having "w" access, they go into the mailbox without being printed immediately. Access for mailboxes is usually given in combinations such as "aow", "arsw", and "adrosw". Access mode "n" can only be given by itself.

Controlling Mailbox Access

Three special commands are used for listing and changing mailbox access:

mbx_list_acl	(mbla)
mbx_set_acl	(mbsa)
mbx_delete_acl	(mbda)

They are used exactly like "list_acl", "set_acl", and "delete_acl".

To use the "mbx_list_acl" command, you need at least "s" access to the directory containing the mailbox. To use "mbx_set_acl" and "mbx_delete_acl", you need at least "sm" access to the directory containing the mailbox.

Suggested Access Settings for Mailboxes

Your default mailbox is in your home directory; its name is your person_id with ".mbx" appended. (Thus Juliet's is

>udd>Capulet>Juliet>Juliet.mbx.) It is automatically created the first time you invoke "print_mail", "read_mail", "accept_messages", or a similar command. You get a default ACL entry of the form:

```
          adrosw   Juliet.*.*
```

To receive mail and messages from other users, you should give them at least "aw" access to the mailbox. If you give them "aow" access, they can read and delete their own mail (messages they have sent). By default, "*.SysDaemon.*" has "aow" access to your mailbox. You should not change this, since it allows you to receive system messages.

INITIAL ACCESS CONTROL LISTS

When a segment or directory is created, it acquires a default ACL. You can specify additional entries for this default ACL through the use of an initial access control list (IACL). An IACL is useful when you want to share your programs and data with other users because it relieves you of having to explicitly set access for these users each time you create a new segment or subdirectory.

IACLs are associated with directories. Every directory has two IACLs. One IACL specifies the ACL for any segment created in the directory. The other specifies the ACL for any directory created in the directory. (There is no IACL for mailboxes.)

Suppose you wanted user Laurence.Friars to have "rw" access to segments subsequently created in your home directory. Give the "set_iacl_seg" (or "sis") command specifying first the directory and then the ACL entry:

```
          set_iacl_seg >udd>Capulet>Juliet rw Laurence.Friars
```

If that directory is your working directory, you can give the command more simply:

```
          sis -wd rw Laurence.Friars
```

Thereafter, the ACL of any segment created in your home directory would contain the default entries ("rw" for you and "*.SysDaemon") plus "rw" for "Laurence.Friars.*". This is so for segments created by an editor or the "copy" command. However, the "move" command uses the original ACL of the segment; the IACL of the new directory is ignored.*

---

*You can override the default behavior of "copy" and "move" with control arguments.

To print a directory's IACL for segments, use the "list_iacl_seg" (or "lis") command. For example:

```
          list_iacl_seg -wd
```

To delete entries from a directory's IACL for segments, use the "delete_iacl_seg" (or "dis") command. For example:

```
          delete_iacl_seg -wd rw Laurence.Friars
```

Changes to an IACL are not retroactive: deleting an entry from the IACL for segments in a particular directory does not effect the ACLs on any segments already in the directory. Of course, you can modify the the access for a segment (or directory) after it has been created by using "set_acl" or "delete_acl".

For controlling the IACL for directories, there is another group of commands:

```
          set_iacl_dir     (sid)
          list_iacl_dir    (lid)
          delete_iacl_dir  (did)
```

Usage of these commands is identical to those intended for segment IACLs.

ACCESS ERRORS

The cause of most access errors is simple enough: you don't have enough access to some segment or directory. Occasionally, you have enough access to a directory "higher" in the storage system than the inaccessible segment or directory to correct the problem by setting yourself access. Otherwise, you'll have to find a user enpowered to set you the access you need.*

When a command or program you're running tries to violate the restrictions imposed by the access controls, you find yourself in one of two situations: a restartable error or a non-restartable error.

Restartable Errors

With some programs, Multics detects attempts to violate the access controls and you "take a fault". An explanatory message

---

*The person most likely to have sufficient access is your project administrator. Ask him or her to set you the access you need (unless IPS User Accounts is your project administrator--they won't).

beginning with "Error:" is printed, and you get a new command level. After correcting the problem, you can usually type "start" ("sr") to continue.

Non-restartable Errors

Other programs flag an access violation before trying to do their task; these programs print an error message beginning with the program name. You may or may not get a new command level; in either event, you must correct the error which caused the message and start the program from the beginning. Common messages of this type are:

Incorrect access on entry.

> You do not have sufficient access to operate on a segment. Examples of this would be trying to print a segment without having "r" access, or trying to write to a segment without "w" access.

Incorrect access on directory containing entry.

> You have tried to perform an operation on a segment, directory, or mailbox without having sufficient access on the directory containing it. Examples of this would be trying to list the contents of a directory on which you do not have "s" access, or attempting to delete a segment in a directory on which you lack "m" access.

Insufficient access to return any information.

> You do not have access to perform an operation and also lack the necessary access to find out anything about your situation.

Validation level not in ring bracket.

> You are trying to perform an operation which you are not privileged to do. An example would be trying to use commands such as "set_acl" on a mailbox.

V. GETTING PRINTED OUTPUT

The New Users' Introduction to Multics--Part I describes how to use the "dprint" command to print a copy of a segment on the high-speed printer. This section discusses the process in more detail and covers IPS-specific information, such as where to pick up your output and acceptable "-destination" and "-request_type" control arguments for dprint. At IPS, you can also get printed output on microfiche. The section ends with a discussion of this service.

DPRINT REQUESTS

On Multics, printing is actually performed not by you, but by special pseudo-users called "daemons" which control the printers. You ask a daemon to print something for your by giving the "dprint" (for daemon-print) command. This causes a "dprint request" to be "queued" for action by the daemon.

You can specify that your request go to one of three printing queues. Queue 1 provides the fastest turnaround, at premium prices; queue 3 gives reduced service at the lowest rates (and is the default); queue 2's service is in between that of the other two.

Since printing is performed by the daemons (not by you), they must have access to the segments to be printed. In general, system defaults provide proper access for the daemons. If access has been changed from these defaults, you must set access for "*.SysDaemon"; provide at least "s" (status) access to the directories involved, and at least "r" (read) access to segments to be printed. (The daemons are logged in under the project_id "SysDaemon".) See Section IV for more information on Multics access control.

The "dprint" command can be used only with character files, such as source programs or text segments. If you attempt to dprint something other than a character file (e.g., segment which contains a direct-access file), the output is likely to be long, expensive, and useless. An operator who notices this happening may cancel the printout.

To dprint the contents of a segment, issue the "dprint" command followed by any control arguments and the segment's pathname. For example:

```
          dprint -delete snaggletooth_snippet
```

Give control arguments before the pathname. For other matters of syntax, particularly the question of how control arguments apply to pathnames when you specify more than one of each, see the description of "dprint" in the MPM Commands and Active Functions (AG92). Many control arguments are supported; all of them are detailed in the MPM description. The ones that are most commonly used specify:

> That the segment be deleted after it is printed (-delete)*
>
> That more than one copy of the output is to be produced (-copy N)**
>
> The number of the request queue, as detailed above (-queue N)
>
> The "request type", which is used to specify paper and type of printing (-request_type XXX)
>
> The header and destination for the printout (-header XXX and -destination XXX)

These last two pieces of information control what is printed on the front of the output. The header is usually your person_id; the destination may be your project_id (the default), or may designate the courier drop-off point to which the printout is to be delivered. See "Destinations and Output Delivery", below, for detailed information about headers and destinations at MIT.

One other control argument, "-notify", can be used to tell the daemon to send you a message when the request is processed.

---

*If you specify the "-delete" control argument, the file is not immediately deleted after printing. A "period of grace" (currently set to one hour) allows for a reprint in case a hardware problem is noticed later by the operator. Also, if you modify the segment at any time after printing begins, the daemon does not delete it.

**The "-copy N" control argument produces multiple copies by printing the output several times, not by using carbon forms. A maximum of four copies may be requested in one dprint command.

Examples

(1) A user wishes to print out the PL/I source program "delta.pl1" from her working directory as quickly as possible. Entering her request into the high-priority queue, she types:

```
     dprint  -queue 1  delta.pl1
```

(2) User LCostello wishes to print two copies of the segment "antics" from his working directory. He types:

```
     dprint  -copy 2  antics
```

To dprint a segment located in another directory, you must have permission (e.g., from the owner of the segment) to access the segment. For example, if LCostello wants to dprint the segment "whosonfirst" from user BAbbott's directory, BAbbott must give LCostello "read" access by typing:

```
     set_acl  whosonfirst  r  LCostello
```

Assuming BAbbott's working directory is >udd>ZANIES>BAbbott, user LCostello may then type:

```
     dprint  >udd>ZANIES>BAbbott>whosonfirst
```

IN THE INTERIM

Since the "dprint" command queues your request to print a segment, not the segment itself, you must not delete, rename, or alter a segment to be printed until the request has been processed or cancelled. You may check whether or not a dprint request has been processed by invoking the "list_daemon_requests" command. You may cancel a pending request with the "cancel_daemon_request" command. Note that requests being processed may or may not appear in the list given by the "list_daemon_requests" command and cannot be cancelled. If you just want to know when the printing is done, use the "-notify" control argument when you give the "dprint" command; the daemon sends you a message when your request is processed. For more information on these commands, see their descriptions in the MPM Commands and Active Functions (AG92) or use the "help" command (e.g., type "help list_daemon_requests").

DESTINATIONS AND OUTPUT DELIVERY

The default destination for printed and punched output at MIT is the Dispatch Area on the second floor of Building 39. Unless arrangements have been made with the IPS Operations staff (as for the courier service described below), you should not specify a destination outside Building 39. Output is filed by project_id; ask for it by the project_id used to print or punch it.

If arrangements have been made for special filing procedures (other than by project_id), specify the pre-arranged name in the "-destination" control argument of the dprint or dpunch request and call for the output by that name.

The "-header" and "-destination" control arguments can be used to send output to another user's folder (provided such a folder exists). Normally, you should specify the recipient's project_id as the "-destination". A "-header" may be used to label such output with the recipient's person_id; note, however, that output is filed by its destination label <u>only</u>. For example, to dprint the segment mercury.info and have the output sent to user ArtooDetoo on project DROID, give the command:

```
     dprint  -ds DROID  -he ArtooDetoo   mercury.info
```

IPS operates a courier service to deliver printed and punched output to three locations other than the Building 39 Dispatch Area. To specify that output be delivered to one of these locations, give a "-destination" from the list below. Output delivered to these locations should be called for by the header on the listing (normally the person_id, unless another header was specified with the "-header" control argument).

CISL-575TS	Honeywell
EASTCMPS	East Campus Facility
LCS-545TS	Lab for Computer Science (formerly Project MAC)

For example, to print the segment "snooker.runoff" and send the output to the East Campus Computing Facility (Building E52), give the command:

```
          dprint  -ds EASTCMPS  snooker.runoff
```

For a fee, IPS will mail you your output. Contact User Accounts and get an OSM (outside mailing) number. Then specify the OSM number with the "-destination" control argument when you dprint something you want mailed to you.

REQUEST TYPES

The "dprint" command line may include a "-request_type" argument to specify special paper or handling for the output. Available request types differ from time to time. For a list of currently-available request types, type "print_request_types", or "prt".

Request types in active use at this writing are listed below. Note that request types other than "printer" are processed only at intervals; their turnaround is normally about 24 hours.

printer	Produces output on standard MIT forms (on 11-by-15-inch white paper). This is the default request type (no "-request_type" control argument given).
x1200	Passes output to the Xerox 1200 printer. This printer prints 132 characters per line, 66 lines per page on 3-hole punched 11-by-8.5 inch paper in landscape format (sideways). This printer cannot overstrike characters, so underscoring does not show up. For more information, type "help x1200".
8by11	Produces output on 8.5-by-11-inch paper (the same size and format as the page you are reading). Paper is perforated near the pin-feed holes, and measures 8.5 by 11 inches after trimming at the perforations. When using these forms, you must ensure that output will not run off the edges.
11by8	Produces output on 11-by-8.5 inch paper--the same size as the sheet you are reading, but with the characters printed in landscape format (i.e., running the long way).
ibmpt	Passes output to the IBM printer. Some users prefer this more legible typeface for final copy. The paper used is standard-sized (11-by-15-inch), unlined, heavy-weight paper. You must specify the "-queue 1" or "-queue 2" control argument. (This is a pricing policy intended to help recover extra handling costs.)
ibm3ply	Passes output to the IBM printer and uses three-ply forms (i.e., with carbons). You must specify the "-queue 1" or "-queue 2" control argument.
stu_cen	Passes output to the 24-hour SIPB printer in the Student Center Library. This printer, a Decwriter 1200, prints 140 characters per line, 88 lines per 8.5-by-11 page. Do not use this printer for printed output longer than 50 pages. Output is not filed--you have to tear your output off the printer yourself. Direct

questions on this service to SIPB (Room 39-200, 253-7788), not to Operations.

text	Passes output to the SIPB letter-quality printing service. Output is printed 85 characters per line on 20-pound unlined 8.5-by-11 continuous-form paper by a NEC Spinwriter 5520 or a Diablo 1620 using a multi-strike carbon ribbon. You may either pick up your output at the SIPB office, Room 39-200, or have output sent to IPS Operations for normal filing or courier delivery. To do the latter, use the "-destination" control argument in the your dprint command, specifying "IPS/" followed by the destination you want. Standard "dprint" rates are charged for this service; however, because of the extra handling required, the default queue is 2. For information on this service, type "help lq_printer".
text_12	Same as "text", except that the printing is 12-pitch, i.e., your output is printed 102 characters per line.
labels	Produces output on continuous-form pressure-sensitive mailing labels. You must specify the "-queue 1" or "-queue 2" control argument. In addition, you must give the "-no_endpage" control argument, unless the output segment contains the form-feed character. Type "help labels" for more information.

GETTING OUTPUT ON MICROFICHE

Generating reams of output is seldom difficult, but sometimes storing them is. You can diminish this problem by getting your output on microfiche. To copy the contents of a segment onto microfiche, simply give the "com_tape" command followed by the pathname of the segment. For example:

```
          com_tape  annals_of_imperial_rome
```

Multics responds by transferring onto an IPS tape the segment or multi-segment file you want copied. The Operations staff automatically processes the tape and makes it available for daily pickup by Syner Graphics, Inc. of Belmont. Syner Graphics makes the microfiche and returns it two days later, and Operations files it in your Multics folder.

Originals are reduced 24 times to produce four-by-six inch microfiche. Each microfiche contains up to 80 frames (input pages), including a 10-frame header that's readable to the naked eye. You specify this header, as well as captions and other information, by means of control arguments to "com_tape". For complete information, type "help com_tape" or see IPS Memo MS-51 (forthcoming).

All costs are charged to your project; see IPS Memo RT-2 for current rates.

VI. IF YOU MUST DEAL WITH CARDS

Try not to use cards. An unseemly vestige of computing's dark ages, they are a discredited data-storage medium, bulky, easily dropped, lost, or damaged. However, there are those who have data on cards they want to move to Multics on-line storage, or who must send information stored on Multics to some unfortunate colleague who insists on cards. If you have one of these excuses, this section is for you.

Cards may be processed (read in or punched out) by Multics in three formats: Multics Card Code, 7punch, and raw. The three formats are described in detail in the MPM Reference Guide (AG91). Briefly, Multics Card Code (MCC) is character information in a punch format approximately equivalent to IBM (EBCDIC) and ANSI card code standards. "7punch" is a special binary format useful solely in backing up information onto cards; in general, 7punch format can be read by Multics only. "raw" is a one-for-one column binary card format. Unlike 7punch, "raw" cards are neither checksummed nor sequenced.

READING CARD DECKS INTO SEGMENTS

You submit card decks to be read into Multics segments to the Dispatch Counter on the second floor of Building 39. To submit a card deck, complete the top part of an Input form (these are located in a tray on the counter in the Input Area). Fold the form and place it at the front of the card deck. Leave the card deck in the Input Tray. If there are more cards than will fit in the tray, inform one of the Dispatching personnel. After the deck is read, it is placed on the shelves next to the file cabinets in the Dispatch Area for you to pick up.

Each deck must begin with three (or more) keypunched control cards. These are used to identify the submitter to Multics and describe the deck name and format. Control cards begin with the characters "++" in columns one and two, followed immediately by a keyword that identifies the type of control card.

In general, card decks submitted to Multics will be read within 24 hours. For protection, segments are created in System Pool Storage rather than in your directory. Once the data has been read, copy the card-image segment(s) into your directory with the "copy_cards" command; you must do this within a reasonable time (usually one week), since segments are periodically deleted from System Pool Storage. (See the MPM description of the "copy_cards" command.)

Control cards should be produced on a standard 029 keypunch. All characters on the control cards are mapped to lower case except those immediately after an escape character (¢).* For example, ¢FIDDLE¢STICKS is mapped to FiddleSticks.

The control-card formats are described in detail and explained in the MPM Reference Guide (AG91). The example below does not show all possible control-card formats, but should be adequate for the typical user submitting data cards at the Dispatch Counter.

Before the first time you submit a deck of cards, you should create a Multics mailbox (if you do not already have one). You will be notified via Multics mail when your cards have been processed. To create a mailbox, or to check the contents of an existing one, type:

```
print_mail
```

Example

User FWray, working on project KingKong, wishes to read a FORTRAN source deck into a segment named "gorilla.fortran". She does this as follows:

- From her terminal, she verifies that gorilla.fortran does not currently exist, typing the Multics command:

```
list **.fortran
```

- From a keypunch, she sets up the source deck with the appropriate control cards, as follows:

```
++DATA   GORILLA.FORTRAN  ¢F¢WRAY ¢KING¢KONG
++PASSWORD        (password field is blank)
++FORMAT  MCC     (this card is optional; MCC is default)
++INPUT
          .
          .
(source program deck in MCC format)
          .
          .
```

- She submits the deck at the Dispatch Counter, Room 39-260. When the job is completed, she retrieves the deck from the Dispatch Area and checks for comments from the operator.

---

*Note that, on some keypunches, the backslash (\) must be used in place of the cent sign (¢).

- From her terminal, she then issues the Multics command:

```
copy_cards  gorilla.fortran
```

  to copy the cards into her working directory.

NOTE: Assuring the accuracy of the read-in process is the user's responsibility. If the cards cannot be read, an error message is sent to your Multics mailbox. You may need to ask for help from the Operations staff.

Deck Size

Decks may not exceed the maximum length of a Multics segment. For raw-code reading, the actual maximum is 9,792 cards. For Multics Card Code, the actual maximum depends on the number of characters read, since trailing blanks on cards are ignored by default. Assuming all 80 columns are punched on each card, the maximum is 13,055 cards. For 7punch decks, the length of the created segment depends on the length of the original data. The typical 7punch card represents 22 words, but it may represent as many as 4,096 words if the original data contained that many consecutive words of identical contents.

Errors

The operator will return the bottom part of the Input Form with the deck if any errors occurred when the cards were read. (You may then correct the errors and resubmit the deck.)

PUNCHING CARD DECKS FROM SEGMENTS

On Multics, punching (like printing) is not actually performed by you, but by special psuedo-users called "daemons" which control the punches. You ask a daemon to punch a segment for you by giving the "dpunch" (for daemon-punch) command. This causes a "dpunch request" to be "queued" for action by the daemon.

You can specify which of the three punching queues your request goes to. Queue 1 provides the fastest turnaround, at premium prices; queue 3 gives slower service at the lowest rates (and is the default); queue 2's turnaround time and price are somewhere in between. Since so few people bother to punch cards, queue 3 is usually more than adequate.

Since punching is performed by daemons (not by you), they must have access to the segments to be punched. In general, system defaults provide proper access for the daemons. If access has been changed from these defaults, you should set access for "*.SysDaemon"; provide at least "s" (status) access to the directories involved, and at least "r" (read) access to the segments to be punched. See Section IV for information on Multics access control.

When you give a "dpunch" command to punch a segment, you may specify any of the card formats described at the beginning of this section; MCC is the default. For example, to punch two copies of the FORTRAN source program named "pisces.fortran" in Multics Card Code, give the command:

```
dpunch -copy 2 pisces.fortran
```

Or, to punch the segment "gemini" in 7punch format, type:

```
dpunch -7punch gemini
```

If you are using "MCC" format to transfer information to another machine, keep in mind that many other machines cannot read cards punched with lowercase characters. Lowercase characters in a Multics segment may be converted to uppercase prior to issuing the dpunch request by invoking the "convert_characters" command. (Type "help convert_characters" for more information.)

Is It Done Yet?

Since the "dpunch" command queues your request to punch a segment, not the segment itself, you must not delete, rename, or alter a segment to be punched until the request has been processed. You may check whether a dpunch request has been processed by giving the "list_daemon_requests" command. You may cancel a pending request with the "cancel_daemon_request" command. Requests actually in the process of being punched may or may not appear in the queue listing given by the "list_daemon_requests" command and cannot be cancelled. If you just want to know when the punching is complete, use the "-notify" control argument when you give the "dpunch" command; the daemon sends you a message when your dpunch request is processed. For more information on these commands, see their descriptions in the MPM Commands and Active Functions (AG92) or use the "help" command (e.g., type "help dpunch").

The Finished Product

Normally, your punched output is placed on the shelves next to the file cabinets in the Dispatch Area on the second floor of Building 39; pick up your output there. Just as with printed output, however, you can arrange for punched output to be delivered to other locations or sent to other users. See "Destinations and Output Delivery" in Section V for information.

The card deck produced as a result of the dpunch command contains additional cards before and after the data. These cards are used to identify the deck and its owner; they are punched in flip-card format, which you can read by the punch pattern (without using the card interpreter). The complete deck looks like this:

```
SEPARATOR CARD
INFO CARDS  -  (usually more than one) punched in flip-card format
SEPARATOR CARD
User's Data
END OF DECK -  punched in flip-card format
SEPARATOR CARD
```

IMPORTANT: The extra cards are not understood by the card-reading program, and must be removed from the card decks before the decks are used again, as they will damage most card interpreters.

VII. PROCESS PRESERVATION

When you log onto Multics, you get something called a "process". Roughly defined, your process is your Multics session, in particular those of its activities whose effects persist while you are logged in but go away after you log out or get a new process. Search rules and search paths, for instance, are established on a per-process basis. Another example is changes made to a file in an editor but not yet "saved"; such pending work is part of your process.

As we said in Section II, Multics automatically saves your process when you are disconnected by a terminal, phone, or network failure. This "process preservation" facility allows you to reconnect to your suspended process and resume work interrupted by the accidental disconnection.

RECONNECTING TO SUSPENDED PROCESSES

Let's say you were disconnected and want your preserved process back. Log back in on the same project within an hour of being disconnected. If your process was successfully saved, Multics informs you that you have a disconnected process, and offers you a number of options. For example:

```
         Multics 34.29: MIT, Cambridge, Mass.
         Load = 49.0 out of 100.0 units: users = 49, ...
      => 1 AOakley
         Password
      =>    <---[Enter your password]
         You have 1 disconnected process.
         AOakley OLDWEST logged in 11/24/80 1344.3 ...
         Last login 11/24/80 1310.4
         Please give instructions regarding your disconnected
         processes(s).
         Please type list, create, connect, new_proc, destroy,
         logout, or help.
```

Tell Multics you want to reconnect; type:

```
         connect
```

Multics responds:

```
         Your disconnected process will be connected to
         this terminal.
         Wait for QUIT.
            <---[Sit tight for a few seconds.]
         QUIT
         r 13:48 0.288 level 2
```

Now you are in the same state as if you had pressed the BREAK or ATTN key, and you can restart or abort what you were doing when you were disconnected.

To restart most programs, give the "start" command. However, some programs, particularly interactive programs like editors, debuggers, and the mail commands, should be re-entered by means of the "program_interrupt" (or "pi") command. This aborts the operation in progress, which may be in an incomplete or unknown state, without exiting from the program.

To abort a program, type "release" (or "rl"). Also type "release" if you were at command level when you were disconnected. This returns you to that level, and accordingly reduces by one the "level" number given in your ready message.

Before you do resume your work, however, you may have to re-establish terminal settings. Terminal settings control such functions as editing characters and end-of-page processing. When Multics reconnects you, it resets your terminal settings to the defaults for the terminal on which you are making the reconnection. Consequently, settings that you added since your login (including any established by your start_up.ec) are nullified. To restore them, give appropriate "set_tty" commands after you reconnect. See the MPM *Communications Input/Output* (CC92) for complete information on terminal settings.

RE-ENTERING EDITORS

One virtue of process preservation is that you can continue with work that was pending in an editor when you were disconnected. Be careful, however, how you recommence an editor session.

If you were in edm "Input" mode, and you re-enter using "pi", edm throws away everything you typed since you entered input mode. You should re-enter the editor with the "start" command, and type the three character sequence: newline, period, newline. Print the current line and repair the damage. Then, re-enter input mode.

Emacs users should log back in on the same kind of terminal. If you reconnect with a different kind, and don't alert Emacs, it responds as if you were using the previous kind. Typically, the results are graphically interesting and totally useless. This is especially a problem if you try to re-enter a suspended editing session, since Emacs can't change terminal types in mid-invocation. But also watch out if you begin a new Emacs session: Emacs remembers your terminal type between invocations, and will be confused unless you signal the terminal type change by giving the "-reset", "-query", or "-ttp" control argument with your "emacs" command.

WARNINGS

One ramification of process preservation is that the only correct way to end a Multics session is to give the "logout" command. If you just hang up, you accrue one more hour of connect charges before the system concludes you are not coming back and logs you out. When you do log out, wait for Multics to type your usage figures before you hang up or you may wind up with a disconnected process and be charged for that extra hour.

Although Multics logs out disconnected processes after an hour of inactivity, they may be kept alive even longer. Disconnected processes can still receive mail and messages, and Multics regards this as "activity". So a garrulous friend can keep your disconnected process chewing up connect-time charges indefinitely. Not only that, "send_mail" and "send_message" offer your friend no clue as to whether your process is disconnected or not--you may be unjustly accused of unsociability.

NOTE: process preservation does not work for processes logged in over ARPANET. (See page II-2.)

CIRCUMVENTING PROCESS PRESERVATION

To disable process preservation for a particular session, give the "-nosave" control argument with your "login" command. If you've already logged in with process preservation in effect, you can disable it by giving the "no_save_on_disconnect" command, and, if you wish, subsequently re-enable the facility with the "save_on_disconnect" command. Remember, if you do disable the process preservation, that it will be turned back on if you get a new process.

VIII. BACKUP AND RETRIEVAL

Suppose you delete or muck up a segment by accident. Or suppose a system failure leaves one of your segments damaged. Do such occurrences mean your work is gone for good? Probably not. On Multics, all user files are automatically copied onto magnetic tapes at regular intervals. (This process is called "backup" or "dumping".) Consequently, you can restore a lost or damaged segment by getting a previous "good" version of it off these tapes. You do this by submitting a "retrieval request".

Backup happens automatically, and most users do not have to know anything about it, or read further in this chapter--until they want to get something back. Data retrieval is complicated by the fact that there are two Multics backup systems, and you must know which one to call upon to recover your segment. So before we tell you how to retrieve segments, we have to discuss how they're backed up.

THE TWO BACKUP SYSTEMS

The two backup systems are the hierarchy dumper and the volume dumper. The older of the two is the hierarchy dumper, having been part of Multics since its inception. The volume dumper was developed later, in response to certain deficiencies in the hierarchy dumper.

The main difference between the two systems is how they approach the data to be backed up. The hierarchy dumper's way of referencing data is similar to yours. It traverses the tree structure of segments and directories ("hierarchy", to use the Multics term). The hierarchy dumper, however, is inefficient due to the nature of its scanning and is very clumsy when used to reload the data lost when a disk pack is damaged. So, to provide quick restoration of data after such failures, the volume dumper was developed. It references data by volume, i.e., by the physical disk drive/pack that contains the segments or directories. (The layout of data on the actual disks does not correspond to the tree structure.)

Both dumpers have three modes of operation: incremental, consolidated, and complete. The mode determines which segments are in fact copied by a dumper. Thus, in incremental mode, a dumper copies only those segments that have changed since the last incremental dump, while in complete mode, the dumper copies all segments. See the table on page VIII-6 for full information on the various modes.

You should also be aware that the hierarchy dumper, as it scans the tree, enters the pathnames of those segments it copies in a list or "map", which is printed and kept in the Dispatch Area on the second floor of Building 39. The volume dumper does not generate maps. Instead it keeps on-line tables that tell what disk volumes were dumped on what tapes and when.

WHICH ONE DO I USE?

In most cases, use the retrieval facility of the hierarchy dumper to recover a segment. Hierarchy backup, besides providing the most efficient and complete retrieval capabilities, makes available maps that allow you to check that a segment was indeed copied and can be reloaded.

There are only three times you should use the volume retriever. One is to recover segments that have been in their desired state for less than 24 hours. The hierarchy dumper, which makes a daily pass over the storage system, would have no chance to catch such a segment, but the volume dumper, which makes hourly passes, might have copied it.

Also use the volume retriever to restore a damaged segment, i.e., a segment whose "damage switch" is on. Multics sets this switch when a system error destroys part of a segment. If you try to use a segment whose whose damage switch is on, you get this message:

```
Entry has been damaged.
Please type "help damaged_segments.gi".
```

After looking at the help file cited, you'll be able to judge whether you have to retrieve the segment. If so, leave the damaged segment on line so that the retriever can obtain the date/time modified from the segment's directory entry.

Finally, at times you may be unable to find a segment you want to retrieve listed in the maps produced by the hierarchy dumper. If you nevertheless know the exact time the segment was on the system, use the volume retriever to get it back.

MAKING HIERARCHY RETRIEVALS

To do a hierarchy retrieval, either go to Building 39 and fill out a Retrieval Request Form or use the "retrieval_request" ("rr") command. You must specify the <u>primary</u> pathnames for the segments and directories you are retrieving. (The primary name is the first name listed by a "list" or "status" command.) Therefore, to make sure you have the correct pathname, you should go to Building 39 and check the maps before submitting the request. Along with the pathname, the request (whether done with the "retrieval_request" command or on the form) must specify the dump to be used (e.g., "complete dump from last weekend"). Refer to the table on page VIII-6 to get an idea which dump would include a copy of the segment you want back. If the segment hadn't been changed recently, specify a complete dump; to retrieve a recently-modified version of a segment, specify an incremental dump.

To retrieve a subtree--all the segments and directories stored under a particular directory--put ">**" after the name of the directory. For example, user JChild, to retrieve the contents of her subdirectory "recipes", specifies ">udd>WGBH>JChild>recipes>**". If you omit the ">**", only the directory (with the links and access control lists it contains, but not its segments) is retrieved.

Type "help retrieval_request" for more information on submitting on-line hierarchy retrieval requests.

Bringing It Back Under a Different Name

Sometimes you may want to reload a segment or directory under a new name. To do this, follow the original pathname with an equal sign (=) and the new pathname. (Do not put spaces around the "=".) For example, to retrieve a file called "flush.pl1" and store the retrieved copy in a file called "flush.pl1.retrieved", you'd specify:

```
    >udd>POKER>Jack>flush.pl1=>udd>POKER>Jack>flush.pl1.retrieved
```

You can do the same thing with subtrees. For example, to retrieve a subdirectory "winnings" as a new directory named "memories", specify:

```
    >udd>POKER>Jack>winnings>**=>udd>POKER>Jack>memories
```

Notice that you do not include the ">**" with the new directory.

It is possible, by means of the renaming process, to retrieve segments and subtrees into new directories--including those of a different user. In order to prevent unauthorized retrievals, the Operations staff checks with the owner of the segment or directory whenever the retrieval will go to another user. This may delay the retrieval for a day or so while permission is being obtained.

MAKING VOLUME RETRIEVALS

To submit a volume retrieval request, use the "enter_retrieval_request" ("err") command. If the file to be retrieved has been on line with its "bad" contents long enough for them to be backed up, you should include the "-previous" control argument with your "err" command to specify that you want the previous version of the file. If the segment is no longer on line, you must also specify a time range covering the time it was last modified (use the "-from" and "-to" control arguments). Otherwise, the retriever mounts the most recent tape and works backwards until it finds your file. Since we generate about 20 tapes a day, the operator may get worn out mounting tapes before the retriever finds your file.

The volume retriever attempts to notify you of what it has done. You may get a notice that a branch is being appended. This means that the retriever has found the directory entry for your file and has appended the entry to your directory. If you try to use the file before the "object is loaded" you get an error message saying that the contents are not in the VTOC (Volume Table of Contents). This means that data has not been loaded onto disk yet. If the volume retriever cannot load your file, it reports that the object was not found. This might be due to an incorrect time range specification or to the tapes not being saved.

For more information on submitting a volume retrieval request, type "help err" or see the description of "enter_retrieval_request" in the MPM Commands and Active Functions (AG92).

WHEN DOES IT HAPPEN AND HOW MUCH DOES IT COST

IPS Operations processes retrieval requests twice a day on weekdays (noon and 8 PM) and once a day on weekends (about 2 PM). (The times are not exact and depend upon the workload of the operator.)

There is no charge for the retrieval of data lost because of a system error. See IPS Memo RT-2 or type "help rates" for the cost of redressing your own mistakes.

EVADING BACKUP

If you are dealing with super-secure data and do not want to have your files put on our backup tapes at all, you may prevent the dumpers from backing up your files. The hierarchy dumpers respect normal access control. Therefore, you may set the ACL of the segments you don't want dumped to "null" for the appropriate SysDaemon. (Backup.SysDaemon for the incremental dumper, Dumper.SysDaemon for the complete dumper, or *.SysDaemon for all SysDaemons including the IO SysDaemon.) The volume dumpers, on the other hand, dump volumes and thus bypass normal access control. To keep them from dumping your files, use the "volume_dump_switch_off" command with either the "-incremental" or "-complete" arguments to specify that you don't want incremental or complete volume dumps for the file.

DO-IT-YOURSELF BACKUP

If you want to do your own backup, use "tape_archive". This command allows you to append and extract files from a tape much as you would do with a regular archive command. We feel that this is the best solution for the general problem of keeping files off line. If you are dumping a whole hierarchy (because, for instance, you are going away for a year), you may want to use "backup_dump". This is the same code as used by the hierarchy dumper. It is reasonably easy to use and uses Multics standard format tapes which are a bit more reliable than other formats. However, these tapes can only be read on a Multics system and you cannot add more files to the tape later. Also the code has less support than does "tape_archive". If you want to use "backup_dump", see one of our consultants.

TYPE AND MODE		SEGMENTS COPIED	WHEN DONE*	LENGTH OF TIME TAPES ARE KEPT
Hierarchy	incremental	those changed since last incremental dump	once every day	one month
	consolidated	those changed since operator-specified time	not currently done	-----
	complete	all	once a week (usually Friday)	six months, except the first one done each month is kept one year and the first one done each year is kept six years
Volume	incremental	those changed since last incremental dump	once an hour	five days or until the next complete dump is done
	consolidated	those changed since last consolidated dump	once a night	one month
	complete	all	first and third week of each month	until the next complete dump is done

FIGURE VIII-1: MULTICS BACKUP SYSTEMS

(*Although Operations tries to maintain the schedule, circumstances may require changes. To make sure a particular dump was done, contact Operations at 253-7739.)

IX. USING MULTICS VIA NETWORKS

Suppose you're not in the Boston area, but you want to use IPS's Multics. In theory, you could hook up to it via a long-distance phone call; however, this is expensive and does not always provide reliable data transmission. But there's a better alternative: networks. A network allows you to make a local phone call in cities across the country and link to a computer far away. The network performs the same function as a direct telephone call to Multics. Once you've completed the network connection procedure, you log in exactly as if you _had_ made a direct call.

Networks are for distant users only. Since there are a limited number of access channels ("ports") available to the networks, please do not use them if you can dial Multics locally.* It would cost you to be discourteous, anyway: a connect-time surcharge applies for use of Multics via networks. See IPS Memo RT-2, or type "help rates", for current rate schedules.

The two networks through which you can use IPS's Multics are Telenet and TYMNET. Use the one that provides an access location--phone number--nearest you. (See _Telenet U.S. Access Locations_ and _TYMNET Telephone Numbers_.)

TELENET

Connecting to Multics via Telenet is relatively simple. First, you call a local Telenet phone number and hook up your terminal to the network. Next, tell Telenet that you want to connect to Multics by typing commands that include a _connection address_ for IPS's Multics. Then login to Multics. Details on this process are given in two Telenet brochures: _How to Use Telenet_ explains the mechanics of using the network, and _Telenet U.S. Access Locations_ lists the phone number you call. You also need to know the connection addresses for IPS's Multics, which are given in the box below.

Telenet Connection Address:

for 100- to 300-baud terminals:	617 138
for 1200-baud terminals:	617 139

*IPS believes that enough access ports are available; access delays should be negligible. If you encounter persistent delays, please call the Associate Director of IPS at (617) 253-7184.

Logging In

Dial your local Telenet access number (listed in Telenet U.S. Access Locations), and place the telephone receiver into the coupler (or press the DATA button). Next type a sequence of characters that depends on what type of terminal you have. If yours is an ASCII terminal, type two carriage returns (NOT line feeds).* Telenet prints its name and an identifying message.

```
TELENET
415 8A
```

Telenet prompts for a terminal code, typing "TERMINAL=". Type the code for your terminal, listed in How to Use Telenet. If your terminal is not listed, just type a carriage return. In the example here, the user types DS16, the terminal code for a Diablo 1620 terminal. Each character prints twice, because the terminal is set to half-duplex and Telenet assumes full-duplex. Don't worry about this now.

```
TERMINAL=ddss1166
```

Telenet prints an "at" sign (@), indicating it is ready to accept commands. If each character of your terminal identifier line printed twice, as shown above, enter the Telenet command HALF, to set the network to half-duplex and make your typing easier to read.

```
@hhaallff
```

Telenet again responds with an "at" sign. Type the connection command, giving the address for IPS's Multics system. Telenet chooses an available network port and prints a connection message.

```
@c 617 138
617 138B CONNECTED
```

NOTE: If you are using a 1200-baud (120-character-per-second) terminal, use network address "617 139" instead of "617 138".

---

*If, rather than an ASCII terminal, you have an EBCDIC (or Selectric-type) terminal, type a carriage return, a semicolon, and another carriage return. However, if you are using the APL character set, you have to use yet a different sequence. If you're using an APL ASCII device, type a carriage return, a right parenthesis, and another carriage return. If you have an APL EBCDIC terminal, type a right parenthesis and a carriage return.

Multics then prints its login herald. At this point, you may proceed through the normal login procedure as described in Section II of New Users' Introduction to Multics--Part I (CH24).

```
Multics 34.12: MIT, Cambridge, Mass.
Load = 43.0 out of 85.0 units: users = 43, 01/21/80 ...
login Odysseus
Password:
[redacted]
You are protected from preemption.
Odysseus FAROFF logged in 01/21/80  1552.9 est Mon ...
Last login 01/17/80  1021.1 est Thu ...
```

Talking to Telenet While Connected to Multics

Once you've connected to Multics, you can do your work as usual. However, sometimes you may need to alter your terminal environment by giving commands to Telenet. You do this by entering Telenet command mode, giving your network commands, and returning to Multics. To enter Telenet command mode, type an "at" sign (@) on a line by itself, followed by a carriage return (NOT a line feed):

```
@
```

Telenet responds:

```
TELENET
@
```

You are now at Telenet command level and may issue network commands. Telenet responds to your network commands with messages (described in How to Use Telenet) and the prompt "@". Unlike Multics, Telenet prints all its messages in CAPITAL LETTERS. See How to Use Telenet for the meaning of standard network messages.

If, after you issue a network command, Telenet responds:

```
 ?
@
```

it means your command was not understood; try again.

To return to Multics from Telenet command mode, type:

```
CONT
```

You may enter Telenet command mode from almost any Multics environment (command level, editors, debuggers, mail, etc.), and are returned to the same environment when you type CONT.

Some Slight Adjustments

There are a number of adjustments you can make from Telenet command mode to make your session run a little smoother. For instance, suppose you neglected to give the Telenet command HALF before logging in and everything you type prints twice (see page IX-2). To belatedly identify your terminal as half-duplex, type:

```
@
 <---[Telenet prompt]
HALF
 <---[Telenet prompt]
CONT
```

Or, conversely, to identify your terminal as full-duplex, type:

```
@
 <---[Telenet prompt]
FULL
 <---[Telenet prompt]
CONT
```

You can speed up the echoing of carriage returns by entering Telenet command mode immediately after you receive the Multics herald (see page IX-3), and issuing the Telenet commands:

```
INSERT.LF.ON ECHO
INSERT.LF.ON INPUT
CONT
```

Then use the "-modes" control argument, in your login command or a later set_tty command (see the MPM), to set terminal modes to "^lfecho", "^crecho", and "fulldpx".

Depending on your location and terminal, you may need to issue the first two commands in reverse order:

```
INSERT.LF.ON INPUT
INSERT.LF.ON ECHO
```

or combine them on the same line:

```
INSERT.LF.ON INPUT ECHO
```

to achieve the correct result (not too many extra linefeeds, not too few).

In some cases, due to inadequate buffering in your terminal, the first few characters of the lines Multics types will not print. To correct this, enter Telenet command mode and give the ENABLE PADDING command:

```
@
 <---[Telenet prompt]
ENAB PADD
 <---[Telenet prompt]
CONT
```

Major and Minor Unpleasantries

A message such as:

```
617 138B LOCAL NETWORK OUTAGE
```

means, among other things, that you have been disconnected. You can reconnect without redialing. (Start with the connection command C 617 138, at the "@" prompt.) The problem is usually temporary, but until it goes away you can sometimes avoid it by specifying a different Telenet port explicitly in the connection command, for example:

```
c 617 137c
```

This connects you to the specific port 137C (if it's not busy), rather than letting Telenet choose a port from those available.

The message:

```
****POSSIBLE DATA LOSS****
```

alerts you to an unspecified difficulty--potential, spurious, fleeting, or imminent. Multics may be about to crash. Multics may have crashed. Telenet may be having problems of its own. In many cases, you may have lost little or no data. (If using an editor, check the last lines input or modified.) This message does not usually disconnect you.

Certain Multics luxuries are not available via Telenet, to wit: the terminal modes "polite" and "replay" (used to delay the interruption of lines you type, and to retype them when they _are_ interrupted) may not work when you are using Multics via Telenet. You are also not provided the convenience of having Multics transmit its official goodbye ("MULTICS NOT IN OPERATION AT 16:27") when it crashes.

Logging Out

Log out as described in Section II of _New Users' Introduction to Multics--Part I_. Multics prints its standard logout message.

```
logout
Odysseus FAROFF logged out 01/21/80  1650.9 est Mon
CPU usage 3 sec, memory usage 83.7 units.
hangup
```

Telenet may inform you that Multics has dropped the phone line and print another "at" sign (@) as a prompt for another connection command.

```
617 138B DISCONNECTED 8:52:27 678 938
@
```

Turn off the terminal and hang up the phone.

TYMNET

Connecting to Multics via TYMNET is detailed in the TYMNET brochure How to Use TYMNET. Basically, the steps are as follows. You dial the appropriate telephone number (listed in TYMNET Telephone Numbers), and give TYMNET the "terminal identifier", "user name", and "password" shown in the box below. Once you receive the connection message from both TYMNET and the Multics herald, you can login to Multics as described in Section 2 of the New Users' Introduction to Multics--Part I (CH24).

terminal identifier:	a
"user name":	mitmul
"password":	(none; just type a carriage return)

Logging In

Dial the local TYMNET access number and place the telephone receiver into the coupler (or press the DATA button). TYMNET responds by asking for a "terminal identifier". (On 100-baud terminals, this printing will be garbled.) Always type "a". TYMNET prints some numbers of its own (node and port IDs).

```
please type your terminal identifier a
-1247-01--
```

The network then prompts you for a "TYMNET user name". Before you respond, make sure you and TYMNET agree on whether your terminal is operating in full-duplex or half-duplex mode. By default, the network assumes full-duplex operation. Therefore, if your terminal is set to half-duplex, you must now alert TYMNET of that fact. To do this, type a backspace after TYMNET asks you to log in but before you type the user name. If you fail to identify a half-duplex terminal to TYMNET, every letter you type during your session prints twice. You can't recover from this; if you see this happening, hang up and start over.

Respond to the prompt with the TYMNET user name for IPS's Multics; type "mitmul":

```
     please log in: mitmul
```

TYMNET then asks for a password. There isn't one; type a carriage return. TYMNET prints a port identifier ("17" in this example) and connects to Multics ("MULT").

```
     password:  <---[Type a carriage return]
     p 17
     MULT is online
```

Multics will then print its login herald. At this point, proceed through the normal login procedure as described in Section II of CH24.

```
     Multics 34.12: MIT, Cambridge, Mass.
     Load = 44.0 out of 85.0 units: users = 44
     login Odysseus
     Password:
     [redacted]
     You are protected from preemption.
     Odysseus FAROFF logged in 02/01/80  1552.9 est Fri ...
     Last login 01/30/80  10.21.1 est Wed ...
```

Logging Out

Log out as described in Section II of the *New Users' Introduction to Multics--Part I* (CH24). Multics prints its standard logout message.

```
     logout
     Odysseus FAROFF logged out 02/01/80  1649.0 est Fri
     CPU usage 3 sec, memory usage 126.7 units.
     hangup
```

TYMNET tells you Multics has dropped the line, and asks for another connection ("please log in").

```
     dropped by host system
     please log in:
```

Turn off the terminal and hang up the phone.

INDEX