HP 3000 MPE/iX Computer Systems

Using HP 3000 MPE/iX: Advanced Skills Tutorial

HP Part No. 32650-90872
Printed in U.S.A. 1998

Edition 1
E1098

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Preface

This is the first edition of *Using HP 3000 MPE/iX: Advanced Skills Tutorial*. It is a replacement for the *Using the 900 Series HP3000:Advanced Skills* which was printed in separate modules. This is intended for MPE/iX users who have completed *Using HP 3000 MPE/iX: Fundamental Skills Tutorial*, hereafter referred to as *Fundamental Skills*.

This self-paced course has hands-on exercises now contained in one booklet. There are six study modules composed of a number of lessons. Other chapters contain setup information and module solutions. There is a glossary of terms and an index.

Note

An accompanying lab tape is necessary for full use of the lessons. This lab tape should be installed by the system manager, using the "Setup Instructions" in this booklet.

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers. You will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories). You may also encounter references to MPE V, which is the operating system for HP 3000s, not based on PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.

Contents

Figures

Tables

1

Begin Here

The following instructions present the information you need to get started with *Advanced Skills*.

Materials List

The Student Kit consists of all parts listed below:

Begin Here	Your introduction to this course with instructions for setting up training session.
Module 1: Account Management	Managing accounts, groups, and users.
Module 2: File Management	Creating and using files.
Module 3: Batch Processing	Creating, controlling, and examining jobs.
Module 4: File Transfer and Storage	Copying files and storing and restoring files from tape media.
Module 5: User Commands	Creating and using user-defined commands and command files.
Module 6: Variables and Expressions	Creating and using variables and variable expressions.
Solutions Guide	Solutions to the questions and exercises in Modules 1 through 6.
Glossary of Terms	Glossary of terms found in *Advanced Skills* and *Fundamental Skills*
MPE/iX Lab Tape	The tape containing the files needed to set up and use *Advanced Skills* on your computer system.

Note

Whether you are taking this course alone at your desk, or sitting in a learning center with other students, you must read the following sections:

- "Course Overview" section of "Begin Here" for background information on the course.
- "Course Instructions" section in "Begin Here" for general instructions on how to take the course.

The person responsible for installing the lab tape must also read the following sections:

- "Setup Instructions" section of "Begin Here" for specific instruction on how to install the lab tape and set up student accounts.

Each lesson of each module contains questions and exercises. Solutions are found in the *Solutions Guide*. As you do the exercises, keep the Solutions Guide handy so that you can follow along and check your responses.

Course Overview

Course Description

This course gives you knowledge of the MPE/iX operating system and provide hands-on experience in the following areas:

- Creating an account structure.
- Creating and listing files and file equations.
- Creating and running jobs.
- Copying files across groups, accounts, and systems.
- Creating, modifying, and using command files and user defined commands.
- Using system and user-defined variables in command files and UDCs.

Course Goal and Objectives

After you complete this course, you will be able to do the following:

- Recognize and use MPE/iX commands to set up and manage accounts, run jobs, copy files, and store and restore files to and from tape.
- Customize your system environment by creating UDCs and command files, and use variables and expressions.
- Catalog and uncatalog UDC files.

Orientation

This course takes an estimated two full days to complete. The majority of the information is presented in tutorial fashion with text to read, instructions to follow, questions to answer, and exercises to complete. Answers to lesson questions and exercises, as well as review exercises are grouped by module and found in the *Solutions Guide*.

This course assumes that the user has successfully completed *Fundamentals Skills* and has mastered the following skills:

- Can log on and log off the MPE/iX system.
- Can use an editor to create, modify, and save files.
- Can explain users, groups, accounts.
- Can make offline listings of files.

Reference Documents

The following manual is used with the course:

- *MPE/iX Commands Reference Manual*

You may use the online Help Facility while studying any of the modules in this course. Enter HELP(Return) at the system prompt to start the Help Facility. When you want to leave the Help Facility, enter: EXIT(Return).

Conventions

Read this important information before you begin to do the modules.

These modules use instructions and different typefaces to clarify what to read and what to do. The following reflects the conventions used throughout this course:

Sample	Information for you to read and know.
Sample	A term you ought to recognize or understand.
Sample	Emphasis to bring something to your attention.
SAMPLE(Return)	Something that is in this typeface, followed by (Return), is what you should enter by typing the letters and pressing (Return).
SAMPLE	Something in this typeface, *without* (Return) shows you how something should look on your terminal screen. *Do not* enter anything you see in this typeface unless it is followed by (Return).
:SAMPLE	Something in this typeface preceded by a colon, is another way of showing you how something should look on your terminal screen. You will learn more about the colon a little later.

Course Instructions

1. When performing the exercises in this course, you must log on as follows:

```
HELLO USERx.ACCTx

Account password = APASSx
User password    = UPASSx
```

X Means

In each of these passwords, the *x* is a two-digit user identifier that should be provided by the person responsible for setting up accounts. That person may also have assigned you different account and user passwords due to system security. Find out what your user identifier is before you begin any of the hands-on activities.

Your home group is CLASS, where the majority of your files reside. You also have two other groups in the account, PUB and PROJECT. PROJECT is empty.

For more information on the files in your account, see the Setup Instructions in this booklet.

2. Make a copy of this page and fill in your user ID number and passwords. Then keep it with you for reference as you go through the activities in the various student booklets:

   Example:

```
USER ID#  1
ACCOUNT PASSWORD:  APASS1
USER PASSWORD:     UPASS1
LOGON:  HELLO USER1.ACCT1,CLASS
```

```
USER ID#

ACCOUNT PASSWORD:  APASS_ _

USER PASSWORD:     UPASS_ _

LOGON: HELLO USER_ _.ACCT_ ,CLASS
```

Setup Instructions

Note

The person responsible for setting up your accounts for this course, must have system manager (SM) capability to install the course material. Ask your system manager to install the material according to the following instructions.

This section helps you create the account structure for all users doing the MPE/iX exercises associated with this training.

Equipment

You should have the following equipment:

- One 900 Series HP 3000 (MPE/iX operating system)
- Two 7935 or 7937 disk drives (at least 1000 sectors per student account and 2000 sectors per course manager account)
- One tape drive (1600 bpi or 6250 bpi)
- One or two line printers
- One terminal per student, connected to the 900 Series HP 3000
- One 900 Series lab tape (STORE format)

Procedure Summary

There are nine steps in setting up the account structure for students taking the course. The following is a general description of the steps; each step is explained in greater detail on later pages, under "Step-by-Step Procedure."

1. Notify system management and operations of course needs (disk space, paper).
2. Restore the instructor's account from tape to disk.
   a. Modify the instructor's account capabilities.
   b. Verify the current job limit, session limit, job security level.
3. Log on as the instructor and activate UDCs.
4. Execute the ADVSETUP command file and its options to create the student accounts for either of the following courses:

   Fundamentals Skills Course:

   - Create the *Fundamental Skills* accounts from scratch.

   Advanced Skills Course:

   - Purge existing *Fundamental Skills* accounts and create new *Advanced Skills* accounts.

     - OR -

   - Create the *Advanced Skills* accounts from scratch.
5. Execute the MOVEFILE command file to set up *Advanced Skills* account files.

Note

Execute MOVEFILE only if you are setting up *Advanced Skills* accounts. This file creates job files that let you log on to each student account and move files over from the instructor account.

6. Execute the CLEANUP command file to perform file management activities.
7. Remove the tape from the tape drive and store it in a safe place.
8. Create new student accounts as needed.
9. Purge existing student accounts as needed.

Each of the steps will now be explained.

Step-by-Step Procedure

1. Notify system management and operations of the course requirements:

```
Disk space  = 2000 sectors for course manager account
              1000 sectors for each student account
Paper       = full supply of line printer paper
JOBSECURITY = LOW (or if HIGH, operator must
              ALLOW job management commands)
JLIMIT      = greater than the number of students
SLIMIT      = greater than the number of students
JOBFENCE    = 7
OUTFENCE    = 7
INPRI       = 7
OUTPRI      = 7
```

Note

If system management cannot set the JOBFENCE and OUTFENCE values as indicated, the student must check those values during module 3 to ensure that the jobs used in the activities behave as they should.

2. Restore the instructor's account from tape to disk.
   a. Mount the MPE/iX course tape.
   b. Log on as MANAGER.SYS (and stay logged on until told to do otherwise!)
   c. Use the RESTORE command as follows to restore the instructor's account to the system:

```
FILE T;DEV=TAPE
RESTORE *T;@.@.ACCT0;CREATE;SHOW
```

   (0 in ACCT0 is a zero, not the letter "o")

   You may have to reply at the system console as follows:

   REPLY=pin#,ldev*(tape drive)*

   d. Once the restoration is complete, place the tape drive back online by pressing the ON LINE key of the tape drive.

e. You should now have ACCT0, the instructor's account, on disk. Enter the following command to verify that there are 42 files in the account:

```
LISTFILE @.@.ACCT0,2

Account:  ACCT0 (instructor)      DISC SPACE:  1568 (SECTORS)
User   :  USER0
Groups :  CLASS (home)            PUB       PROJECT   MYGROUP

Files  :  ADVSETUP      MYUDC1    PFILE     (future)  FKEY1
          AFILE         MYUDC2                        PROMPT1
          AJOB          MYUDC3              (This     ROMAN1
          CF            P                   group
          CLEANUP       PR                  will be
          COPYFILE      PRX                 created
          DELVAR        PURMOVE             by the
          DOFILE        RECUR               ADVSETUP
          HIC           RELFILE             command
          HICOM         SECFILE             file.)
          JOB0          SJT
          JOB1          STATS
          JOB2          T
          JOB3          TERTIME
          LF            TESTFILE
          MODINST       WELCTEMP (template for welcome file)
          MOVEFILE
          MYFILE
          MYFILE1
          MYFILE2
          MYJOB1
          MYJOB2
```

f. Modify the instructor's capabilities, passwords, and home group with the following command file:

```
MODINST.CLASS.ACCT0
```

MODINST is a command file that contains the following commands:

```
ALTACCT ACCT0;CAP=SM,AM,AL,GL,ND,SF,BA,IA,PH;PASS=MORE
ALTUSER USER0.ACCT0;CAP=SM,AM,ND,SF,BA,IA,PH;PASS=MONEY;HOME=CLASS
ALTGROUP PUB.ACCT0;ACCESS=(R,x:ANY;W,A,L,S:AL,GU)
ALTGROUP CLASS.ACCT0;CAP=BA,IA,PH;ACCESS=(R,W,A,L,x,S:GU)
```

Enter the following command and compare the account, user, and group capabilities and passwords with those listed above:

```
LISTACCT ACCT0;PASS
LISTUSER USER0.ACCT0;PASS
LISTGROUP PUB.ACCT0;PASS
LISTGROUP CLASS.ACCT0;PASS
```

If the capabilities do not match, use ALTACCT, ALTUSER, or ALTGROUP to change them accordingly.

Note

The instructor account or manager of the course, using USER0 and ACCT0, now has extended capabilities beyond those of the student, in order to be able to monitor or amend student accounts as needed.

You may wish to replace MORE and MONEY with meaningful passwords of your choice. Use the `ALTACCT` and `ALTUSER` commands to do so.

3. Log on as the instructor and activate UDCs:
   a. Log on as `USER0.ACCT0`:

      ```
      HELLO USER0.ACCT0
      ```

      (home group = CLASS)

      ```
      Account password: MORE
      User password:    MONEY
      ```

   b. Verify that you are in the CLASS group:

      ```
      SHOWME
      ```

   c. Issue the following command to activate all your UDCs:

      ```
      SETCATALOG MYUDC1
      ```

   d. Issue the following command to verify that MYUDC1 is activated:

      ```
      SHOWCATALOG
      ```

4. Execute the ADVSETUP command file to create the students' accounts:
   a. Issue the following command:

      ```
      ADVSETUP
      ```

      or

      ```
      XEQ ADVSETUP
      ```

      You can now set up accounts for *either* the *Fundamental Skills* course *or* the *Advanced Skills* course. Each time you execute ADVSETUP, you can set up accounts for *one* type of course.

   b. You receive the following prompt:

      ```
      Are you setting up the Fundamental Skills course? (Y or N):
      ```

   c. Fundamental Skills:

      - Respond `Y` if you are setting up accounts for the *Fundamental Skills* course. You receive another set of prompts.

        ```
        Purge Create Quit Modify
        ```

      - Respond `C` to create the *Fundamental Skills* accounts.

Note `Q` lets you `QUIT`, `P` lets you `PURGE` the accounts. Use `P` to purge individual or all accounts in either course whenever necessary.

```
What is the number of the first
account/user you wish to create or
purge?
```

- Respond with `1` when you are first setting up the accounts. See step 7 if you are adding more accounts.

```
What is the number of the final
account/user you wish to create or
purge?
```

- Respond with a number representing the last account you wish created - usually the same as the total number of students. (See step 7 if you are adding more accounts.)
- You get this message when the accounts are created:

```
*** Creating  new account ACCTx for Fundamental Skills ***

*** x account(s) created ***
```

d. Advanced Skills (General):

- Respond `N` if you are not setting up the *Fundamental Skills* accounts. You receive another set of prompts:

```
Are you setting up the Advanced Skills course? (Y or N):
```

- Respond `Y` if you are setting up accounts for the *Advanced Skills* course.

e. Advanced Skills (*Fundamental Skills* accounts already exist):

```
************************************************************
*** ACCTx accounts may already exist for the Fundamental ***
*** Skills course.  Be careful not to purge any accounts ***
*** that may still be in use.  If any accounts on the    ***
*** system are no longer in use, you can purge them and  ***
*** reuse them for the Advanced Skills course by answer- ***
*** ing YES to the following prompt.                     ***
************************************************************

Do accounts/users from Fund. Skills already exist (Y or N):
```

- Answer `Y` *only* if you previously executed ADVSETUP and created student accounts for *Fundamental Skills*. This ensures that the old accounts are purged and the new ones are created.

  You get this message:

```
*** Press P to purge old Fund. Skills accounts. ***

Purge Create Quit Modify
```

- Respond `P` to purge the old *Fundamental Skills* accounts:

```
What is the number of the first account/user?
```

- Respond with the number of the first *Fundamental Skills* account you wish to purge.

```
What is the number of the final account you wish to create or purge?
```

- Respond with a number representing the last account you wish purged—usually the same as the total number of students.

  You now see this message:

  ```
  *** ACCTx through ACCTy will be purged ***
  Proceed with purging? (Y/N)
  ```

- Respond `Y` if you wish to continue purging the old accounts.

  You now get the following prompt(s) and message(s):

- Respond `Y` as long as you wish to purge accounts:

  ```
  ACCOUNT ACCTx TO BE PURGED? (YES/NO)
  ACCOUNT ACCTy TO BE PURGED? (YES/NO)

  *** Creating new account ACCTx for Advanced Skills ***
  *** Creating new account ACCTy for Advanced Skills ***

  *** x accounts created ***
  ```

f. Advanced Skills (*Fundamental Skills* accounts do *not* exist. Create new *Advanced Skills* accounts):

Note Skip to Step 5 for the rest of the procedure.

```
Do accounts/users from Fundamental Skills already exist
(Y or N):
```

- Answer `N` if this is the first time you are setting up accounts.

  ```
  Purge Create Quit Modify?
  ```

- Respond `C` to create the *Advanced Skills* accounts.

  ```
  What is the number of the first account/user?
  ```

- Respond with `1` if you are setting up *Advanced Skills* accounts for the first time. See step 7 if you are adding more accounts.

  ```
  What is the number of the final account/user?
  ```

- Respond with a number representing the last account you wish created - usually the same as the total number of students. See step 7 if you are adding more accounts. You see these messages:

  ```
  *** Creating new account ACCTx for Advanced Skills ***
  *** Creating new account ACCTy for Advanced Skills ***

  *** z accounts created ***
  ```

- Example #1:

  Suppose you wish to set up accounts for ten *Fundamental Skills* course students. Your responses to the prompts would be as follows:

  ```
  Are you setting up the Fundamental Skills course? Y
  Purge Create Quit Modify? C
  What is the number of the first account/user? 1
  What is the number of the final account/user? 10
  ```

```
*** Creating account ACCT1 for Fundamental Skills ***
*** Creating account ACCT2 for Fundamental Skills ***
.
.
.
*** Creating account ACCT10 for Fundamental Skills ***

*** 10 accounts created ***
```

- Example #2:

  Suppose those users now wish to take the *Advanced Skills* course. Your responses the prompts would be as follows:

```
Are you setting up the Fundamental Skills course? N
Are you setting up the Advanced Skills course? Y

************************************************************
*** ACCTx accounts may already exist for the Fund. Skills***
*** course.  Be careful not to purge any accounts        ***
*** that may still be in use.  If any accounts on the    ***
*** system are no longer in use, you can purge them and  ***
*** reuse them for the Adv. Skills course by answering   ***
*** YES to the following prompt.                         ***
************************************************************

Do accounts/users from Fund. Skills already exist (Y or N): Y

*** Press P to purge the old Fundamental Skills accounts  ***

Purge Create Quit Modify? P

What is the number of the first account/user? 1
What is the number of the final account/user? 10

*** ACCT1 through ACCT10 will be purged ***
Proceed with purging? (Y/N) Y

ACCOUNT ACCT1 TO BE PURGED? (YES/NO) Y
ACCOUNT ACCT2 TO BE PURGED? (YES/NO) Y
ACCOUNT ACCT3 TO BE PURGED? (YES/NO) Y
.
ACCOUNT ACCT10 TO BE PURGED? (YES/NO) Y

*** Creating account ACCT1 for Advanced Skills ***
*** Creating account ACCT2 for Advanced Skills ***

*** Creating account ACCT10 for Advanced Skills ***
```

- Example #3:

  Suppose you purged or never had the *Fundamental Skills* course student accounts and you now wish to create ten *Advanced Skills* course student accounts. Your responses to the prompts would be as follows:

```
Are you setting up the Fundamental Skills course? N
Are you setting up the Advanced Skills course? Y

************************************************************
*** ACCTx accounts may already exist for the Fund. Skills ***
*** course.  Be careful not to purge any accounts        ***
*** that may still be in use.  If any accounts on the    ***
*** system are no longer in use, you can purge them and  ***
*** reuse them for the Adv. Skills course by answering   ***
*** YES to the following prompt.                         ***
************************************************************

Do accounts/users from Fund. Skills already exist (Y or N): N

Purge Create Quit Modify? C
What is the number of the first account/user? 1
What is the number of the final account/user? 10

*** Creating account ACCT1 for Advanced Skills ***
*** Creating account ACCT2 for Advanced Skills ***
.
.
.
*** Creating account ACCT10 for Advanced Skills ***

*** 10 accounts created ***
```

g. When account creation is complete, you have the following student account structure for the respective courses. Issue the following command to verify that the accounts exist:

   LISTACCT ACCTx

   (x is the number of each student account)

```
Fundamental Skills Course:

        Account:  ACCT1 to ACCTx (students)
        User   :  USER1 to USERx
        Groups :  PUB                 MYGROUP          YOURGRP

        Files  :  empty               empty            empty

Advanced Skills Course:

        Account:  ACCT1 to ACCTx (students)
        User   :  USER1 to USERx
        Groups :  CLASS               PUB              PROJECT

        Files  :  empty               empty            empty
```

5. If you are creating *Advanced Skills* course accounts, after account creation, execute the MOVEFILE command file. MOVEFILE moves files from the instructor account to each student account. It does this by creating and streaming jobs that log on to each student

account and execute the COPYFILE command file. COPYFILE does the actual copying of files from instructor to student account.

a. Issue the following command:

```
MOVEFILE
```

b. Answer the prompts as you did when executing ADVSETUP:

```
*** ENTER A ZERO (0) TO TERMINATE WHEN ANSWERING PROMPTS ***

What is the number of the first student account you just added?

What is the number of the final student account you just added?
```

c. As MOVEFILE executes for each student account, you see several messages on the screen. The most important are shown below:

`MOVEFILE START1`	Executes MOVEFILE.
`#Jx STREAM MOVEPRx`	Streams job to log on to student accounts. *Press* (CR) *at this point.*
`COPYFILE IS MOVED OVER`	Copies COPYFILE command file to student account.
`ALL FILES COPIED TO ACCTx`	Executes COPYFILE and copies all files to student account.

These messages repeat for each account created earlier, not necessarily in order. The process is complete only when `ALL FILES COPIED ...` message appears for the last account.

When no more messages appear, press (Return) to return to the system prompt.

d. When MOVEFILE has completed, the student account structure should look like this. Issue the following command to verify:

```
LISTFILE @.@.ACCTx,2
```

```
          Advanced Skills Course:

Account:  ACCT1 to ACCTx (students)
User   :  USER1 to USERx
Groups :  CLASS              PUB            PROJECT

Files  :  AFILE              PFILE          empty
          CF
          COPYFILE
          DOFILE
          HIC
          HICOM
          JOB0
          JOB1
          JOB2
          LF
          MYFILE
          MYFILE1
          MYFILE2
          MYJOB1
          MYJOB2
```

```
MYUDC1
MYUDC2
MYUDC3
PR
PRX
RECUR
SJT
STATS
TERTIME
```

e. The account users are USER1 to USER*x*. The home group is CLASS, and the logon is:

```
HELLO USERx.ACCTx

Account password:  APASSx
User password:     UPASSx
```

6. All student accounts now contain their necessary files. At this point, perform file management activities to resecure files, delete variables, and purge temporary files created during MOVEFILE and COPYFILE execution. Execute this command:

   CLEANUP

   The CLEANUP command file actually executes three other command files (which can be executed separately on other occasions, if need be):

DELVAR	Deletes all variables used by MOVEFILE.
PURMOVE	Purges temporary files.
SECFILE	Secures all previously released instructor files.

7. The course account creation is now complete. To finish the installation, rewind and remove the tape from the tape drive.

   a. Fundamental Skills Course:

   You need the following amount of disk space for the initial student and instructor accounts:

```
GROUP and ACCOUNT          DISK SPACE BY SECTORS

Course Manager or Instructor:

        CLASS.ACCT0          2128
        PUB.ACCT0              16
        PROJECT.ACCT0           0
        MYGROUP                 0
        YOURGRP                 0

        SUBTOTAL:            2208 sectors

Student:

        PUB.ACCT                0
        MYGROUP                64
        YOURGROUP               0

        SUBTOTAL:               0 sectors
```

```
TOTAL:                          3136 sectors
```

b. Advanced Skills Course:

You need the following amount of disk space for the initial student and instructor accounts

```
GROUP and ACCOUNT          DISK SPACE BY SECTORS

Course Manager or Instructor:

        CLASS.ACCT0          2128
        PUB.ACCT0              16
        PROJECT.ACCT0           0
        MYGROUP                64
        YOURGRP                 0

        SUBTOTAL:            2208 sectors

Student:

        CLASS.ACCT           1552
        PUB.ACCTx              16
        PROJECT.ACCTx           0

        SUBTOTAL:            1568 sectors

TOTAL:                       3776 sectors
```

8. Create additional student account(s) as needed:

a. Use ADVSETUP to create the accounts. Respond as you did in step 4. Just remember to specify the correct first and final account number and specify the C option.

- Example #1:

Suppose you already have ten *Fundamental Skills* accounts and wish to add two more. You would respond to the prompts as follows:

```
Are you setting up the Fundamental Skills course? Y
Purge Create Quit Modify? C
What is the number of the first account/user? 11
What is the number of the final account/user? 12

*** Creating account ACCT1 for Advanced Skills ***
*** Creating account ACCT2 for Advanced Skills ***
.
.
.
*** Creating account ACCT12 for Advanced Skills ***

*** 12 accounts created ***
```

- Example #2:

Suppose you purged *Advanced Skills* accounts ACCT5 and ACCT6 earlier, and now two new students need those accounts:

You would respond to the prompts as follows:

```
Are you setting up the Fundamental Skills course? N
Are you setting up the Advanced Skills course? Y
```

```
************************************************************
*** ACCTx accounts may already exist for the Fund. Skills ***
*** course.  Be careful not to purge any accounts         ***
*** that may still be in use.  If any accounts on the     ***
*** system are no longer in use, you can purge them and   ***
*** reuse them for the Adv. Skills course by answering    ***
*** YES to the following prompt.                          ***
************************************************************

Do accounts/users from Fund. Skills already exist (Y or N): N

Purge Create Quit Modify? C
What is the number of the first account/user? 5
What is the number of the final account/user? 6

*** Creating account ACCT5 for Adv. Skills ***
*** Creating account ACCT6 for Adv. Skills ***

*** 2 accounts created ***
```

■ Example #3:

Suppose you already have ten *Fundamental Skills* accounts. USER2 and USER4 are ready to continue with *Advanced Skills*. You would respond to the prompts as follows:

```
Are you setting up the Fundamental Skills course? N
Are you setting up the Advanced Skills course? Y

************************************************************
*** ACCTx accounts may already exist for the Fund. Skills ***
*** course.  Be careful not to purge any accounts         ***
*** that may still be in use.  If any accounts on the     ***
*** system are no longer in use, you can purge them and   ***
*** reuse them for the Adv. Skills course by answering    ***
*** YES to the following prompt.                          ***
************************************************************

Do accounts/users from Fund. Skills already exist (Y or N): Y

*** Press P to purge old Fund. Skills accounts. ***

Purge Create Quit Modify? P

What is the number of the first account/user? 2
What is the number of the final account/user? 2

*** ACCT2 through ACCT2 will be purged ***
Proceed with purging? (Y/N) Y

ACCOUNT ACCT2 TO BE PURGED? (YES/NO) Y

*** Creating account ACCT2 for Adv. Skills ***

*** 1 account created ***

Are you setting up the Fundamental Skills course? N
Are you setting up the Advanced Skills course? Y

************************************************************
*** ACCTx accounts may already exist for the Fund. Skills ***
*** course.  Be careful not to purge any accounts         ***
*** that may still be in use.  If any accounts on the     ***
```

```
*** system are no longer in use, you can purge them and   ***
*** reuse them for the Adv. Skills course by answering    ***
*** YES to the following prompt.                          ***
************************************************************

Do accounts/users from Fund. Skills already exist (Y or N): Y

*** Press P to purge old Fund. Skills accounts. ***

Purge Create Quit Modify? P

What is the number of the first account/user? 4
What is the number of the final account/user? 4

*** ACCT2 through ACCT4 will be purged ***
Proceed with purging? (Y/N) Y

ACCOUNT ACCT4 TO BE PURGED? (YES/NO) Y

*** Creating account ACCT4 for Adv. Skills ***

*** 1 account created ***
```

b. For *Advanced Skills* accounts, enter MOVEFILE to copy files into the accounts. Just remember to specify the correct first and final account number as you did with the ADVSETUP command file.

9. Purge existing student account(s) as needed:

   - Use ADVSETUP to purge the accounts. Answer N to the prompts asking if you are setting up *Fundamental Skills* or *Advanced Skills* accounts. Then specify the P option for purge and give the correct first and final account number.

2

Module 1: Account Management

Module 1 presents the following basic and advanced account management concepts:

Lesson 1	Managing Your Account
Lesson 2	Managing Account Groups
Lesson 3	Managing Users

Challenge Test

1. Define the following account capabilities and note the ones that are default.

AM	AL
GL	ND
SF	BA
IA	PH

2. Indicate who (account manager, general user, or system personnel) can do the following:

   a. list information on their account

   b. change their account capabilities

   c. list their account password

   d. Change their account password

3. What two commands list information on account groups?
4. What command would create a new group, `OURGROUP`, with a password of PW, default capabilities, and user access limited to read, write, and execute for the group librarian?
5. Which commands allow you to change a group's attributes and delete a group?
6. Which command allows you to add ACD protection to the files that you own?
7. What commands create a new user, Jean, with a home group of `CLASS`, a password of OPEN, and default capabilities?
8. What software can a system manager use to interactively create accounts, groups, and users on the system?
9. What command allows you to change Jean's password to CLOSE? Delete Jean (user) from the account?

Lesson 1 Managing Your Account

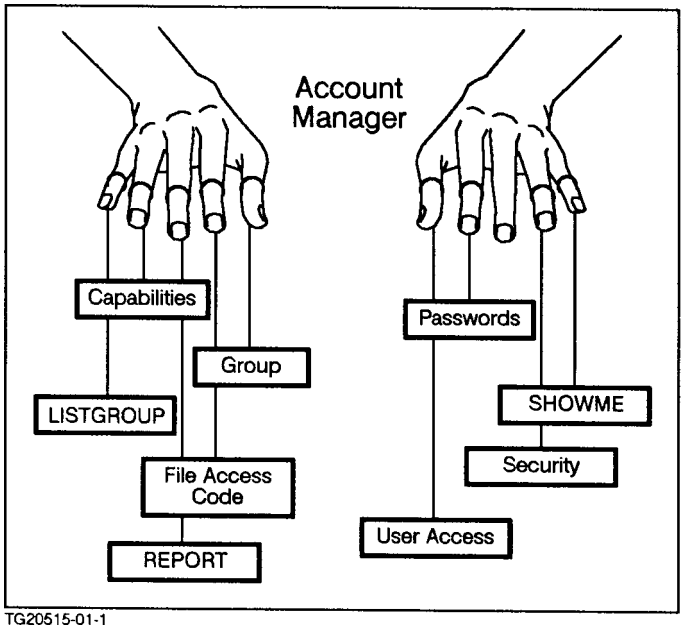


Figure 2-1. Account Manager

Introduction

Lesson 1 presents the following information about managing your account:

- listing account information
- meaning of access codes
- passwords—usage and security

In this and the following lessons, you learn more about account management, and how to become more effective in your interaction with the MPE/iX system. As an account manager for a particular account, you have the final responsibility for the management of that account.

As a general user, your capability is limited by account management; however, you are still responsible for being able to manage your own interaction with the system within user limits.

For the duration of this course, you have been given account manager (AM) capability for your account. By having this responsibility, you are to do a number of different tasks that are not permitted to users without this capability. Commands and capabilities reserved for account manager, as well as higher capabilities, are noted in this and the following lessons.

Note

When you complete this course, you may want to check with your system manager about getting an account with AM capability, if warranted, for your processing requirements.

Listing Account Information

If you haven't already done so, please log on to the account specified by your system manager or supervisor. What kind of capabilities are assigned to your account? What are the limits on your disk space? CPU time? User access?

To access this information, use the LISTACCT command. Enter that now to get information on your own account.

```
LISTACCT
```

Return

You should see a display similar to the following on your screen:

```
ACCOUNT:ACCTx

DISC SPACE:  1568(SECTORS)      PASSWORD: **
CPU TIME:    63(SECONDS)        LOC.ATTR. $000000000
CONNECT TIME:   01 MINUTES      SECURITY--READ    :AC
DISC LIMIT: UNLIMITED                     WRITE   :AC
CPU LIMIT: UNLIMITED                      APPEND   :AC
CONNECT LIMIT: UNLIMITED                  LOCK    :AC
MAX PRI  :150                             Execute :AC
GRP UFID: $05580000 $228970A9 $000C3360 $578210A2 $07949089
USER UFID:$05580002 $228790A9 $000FCCB0 $578210A2 $07949DC0
CAP:  AM,AL,GL,ND,SF,BA,IA,PH
```

Some of the information that you see is self-explanatory. Two things that may not seem so clear are the CAP information listed at the lower left and the SECURITY information listed in the right column.

CAP stands for "capability," and the abbreviations that follow indicate the capabilities assigned to your account. Explanations for these abbreviations are found in the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) and in the help facility under the NEWACCT command.

Note

If your manual set does not include the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) you may order it from your sales representative. It contains all of the command references that are found in the *Commands Reference Manual*.

Simply type HELP at the system prompt, and press Return to start the help facility.

After you review this information, list the definitions of the account capabilities below:

Q1-1	Account Capabilities: AM = AL = GL = *ND = *SF = *BA= *IA = PH =

Note

The capabilities noted with the asterisk in Q1-1 are default capabilities.

The default capability set, ND, SF, BA, and IA, is required in order to process, store, and print information from an MPE/iX system and peripheral devices such as printers, magnetic tape, and disk drives. Other capabilities, such as account librarian (AL) and group librarian (GL), allow you and other users with these capabilities to manage groups in the account. Remember this: no user or group within an account can have capabilities that exceed the capabilities assigned to that account.

Note

Only system operations may change account capabilities.

Q1-2	Given your account capabilities listed above, could any of the users in your account have PM (privileged mode) capability?

Access Codes

The security access codes displayed on the right by the `LISTACCT` command regulate the access to files within your account. This access is defined by system management and may be changed only at that level. The five access codes are:

R = Read
W = Write
A = Append
L = Lock
E = Execute

Next to each access code is an abbreviation for the type of user who may have that kind of access. There are six different types of users:

ANY = Any user (on the system)
AC = Member of this account only
GU = Member of this group only
AL = Account librarian only
GL = Group librarian only
CR = Creating user only

Check the `NEWACCT` command information for an explanation of the user code AC.

Q1-3	Use of your account has been restricted to which user(s)?

Logon Password

Files belonging to the users of an account should be protected from unauthorized personnel trying to log on to that account. You, as the account manager, can establish this protection through the use of passwords. MPE/iX allows you to enter your user, account, and group passwords either as part of your logon or in reply to a password prompt.

If you enter your passwords as part of your logon, there is a chance that their security may be compromised, as the passwords will be displayed on the screen; therefore, enter your passwords in response to the appropriate prompt. When done this way, the passwords are not displayed to the screen.

Listing Account Passwords

If you ever forget your account password while you are logged on to your account, you can list your password by using the `LISTACCT` command followed by the `PASS` parameter. Try that now.

```
LISTACCT;PASS
```

Note

Only users with at least AM capability may list their account passwords with the `LISTACCT` command.

Password Security

System management is responsible for setting up your account password. As an account manager, you are responsible for setting up and monitoring user and group passwords for your account. To ensure continuing security on your account, limit access to your passwords. Do not make passwords accessible or available to unauthorized users.

Better yet, have user and account passwords changed on a regular basis. Avoid using names or words that are easily associated with you, such as your first, last, or middle name; a nickname; the name of your husband or wife.

Users who wish to change their own user password may do so by using the `PASSWORD` command. The change does not become effective until the user logs off and logs back on. (Refer to Module 4, Lesson 5, “Changing User Passwords” in *Fundamendal Skills*.)

When you will be away from your terminal for a period of time, log off, ending your session. Logging off helps ensure the security of your account files and your passwords from unauthorized users.

Q1-4	Which practices identify good account security? a. changing passwords regularly b. not using well-known names, such as a nickname, for passwords c. protecting passwords from unauthorized users d. not entering passwords in your logon e. all of the above

Lesson Summary

1. Use the `LISTACCT` command to list information on your account.
2. Only system management may set and change session and job limits as well as account capabilities and account passwords.
3. Account security is every user's responsibility.
4. Account capabilities are assigned by system management and may only be modified at that level.
5. No user or group within an account may have capabilities exceeding those of the account.
6. The `ALTSEC` command allows users to add ACD protection to the files that they own.

Exercise 1-1: Lesson 1 Review

Indicate who can perform the following tasks:

	Task	Account Manager	User	System Operations
1.	List information for a user.			
2.	Set account passwords.			
3.	Change user passwords.			
4.	List account capabilities.			
5.	Show jobs and sessions currently running on the system.			
6.	Change account capabilities.			
7.	Use the `LISTACCT` command to list the account password.			
8.	Set session and job limits.			

********** End of Exercise 1-1 **********

Lesson 2 Managing Account Groups

Introduction

Lesson 2 presents information on:

- listing account groups
- defining capability and access attributes
- adding, modifying, and deleting account groups
- verifying program capabilities with VERSION

To help you keep track of the files in your account, MPE/iX allows account managers to create individual groups with specified attributes and capabilities. These capabilities include listing information on all groups within the account, creating and altering groups, and assigning users to specific groups.

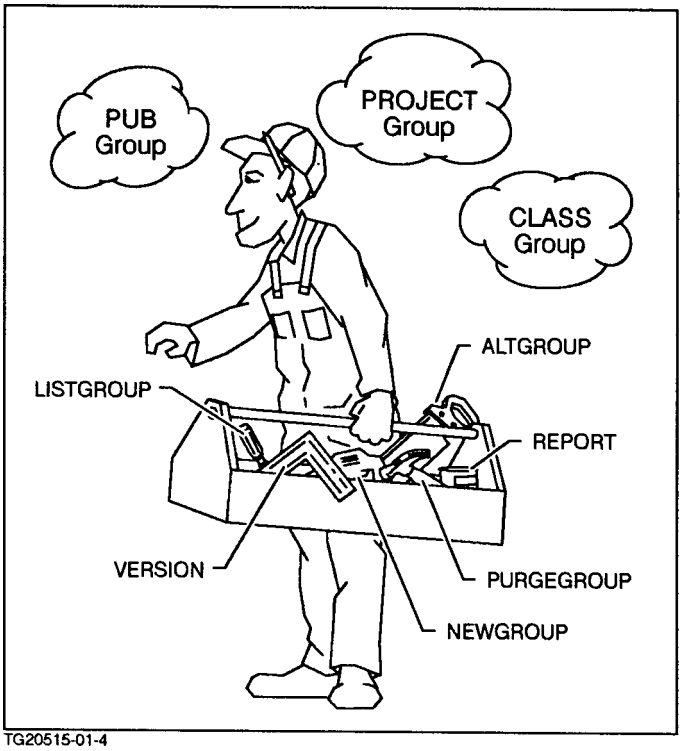


Figure 2-2. Managing Groups

Group Assignments (account manager)

With several users using one account, many account managers choose to assign individual users to specific groups. They also use the PUB group for files of interest to all users.

If you are the only user of your account, you may want to consider setting up individual groups for different projects or purposes. No limit exists on the number of groups that you may have in a single account.

Listing Groups in an Account

A quick way to list the groups within an account is to use the REPORT command. Enter that command now on your terminal to see which groups are currently in your account.

```
REPORT
```

Here's a sample of what you might see on a terminal screen:

```
Account  FileSpace  Sector    CPU-Seconds     Connect-Minutes
  /Group   Count    Limit    Count Limit        Count Limit
ACCTx      1000      **       234    **         2093    **
  /CLASS    950      **         6    **           90    **
  /PROJECT    0      **         8    **            3    **
  /PUB       50      **       220    **         2000
   **
```

Note

For general users, the REPORT command displays only the information about their current group.

The names of the account groups are listed in the far left column. Additional columns of information state the amount of file space occupied by the group, the cumulative total CPU time used by persons using that group, as well as their cumulative connect time to the system.

Q1-5 Which of the groups in the display above has done the following?

a. Used the most cpu time

b. taken the largest portion of file space

c. been connected for the least amount of time

More Information About Account Groups

What if you want more information about the groups in an account?

To get more detailed information about your account, use this command:

```
LISTGROUP
```

Return

Do that now so you can see group information for your account.

Does the display look familiar? It should. You're seeing similar information to that displayed by the LISTACCT command. This listing shows some of the information presented on the previous display as well as additional information on group capabilities and security.

Note

For general users the LISTGROUP command displays only information for their current group.

Q1-6	A user having account manager capability is able to list information for which groups? Users without account manager capability may only list information for which groups?

Group Attributes

Each group is assigned certain attributes selected by the account manager or by the system (default). These attributes can be assigned when the group is created or modified.

You can see all of the attributes if you read about NEWGROUP in the help facility (HELP NEWGROUP PARMS) or in the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364). Two of the most important attributes are capabilities and file access.

Group Capability Attributes

Group capability attributes are very similar to those for accounts and account users. The default capabilities for groups are IA (interactive access) and BA (batch access).

Q1-7	How many of the following attributes can you identify? (Need help? Check under NEWGROUP in the help facility.) PH = IA = BA = PM = MR = DS =

A capability cannot be assigned to a group if the capability has not been assigned to the account in which the group resides.

Use the LISTGROUP command again, and review the information displayed for your account groups. Capabilities are listed at the end of the left column as CAP.

Which of the group(s) in your account have default capability attributes (BA, IA)? Which group has process handling capability (PH) in addition to default capabilities?

Group File Access Attributes

File access attributes specify the mode of access that a type of user can have. When a group is created, each type of file access is noted with a single letter abbreviation as follows:

R = Read
L = Lock
A = Append
W = Write (implies A and L)
X = Execute
S = Save

As you remember from the first lesson, user types are also identified with an abbreviation.

Q1-8	How many of the following user abbreviations can you identify? (Need help? Use the help facility for the `NEWGROUP` command.) ANY = AC = GU = AL = GL = CR =

Use the `LISTGROUP` command and look again at the information provided for the groups in your account. File access information is displayed on the right side.

Which user(s) have read (R) and execute (X) access to the `PUB` group? Which users have write (W) access to the PROJECT group?

Here is a timesaver for account managers. To list only a single group, use the `LISTGROUP` command followed by the group name.

For example, how would you list only the information for the `PUB` group?

The command `LISTGROUP PUB` lists information only for the `PUB` group. If you have a large number of groups in your account, you will find it more efficient to use this command when you want information on a specific group.

Note

General users may list information only for their current group. A minimum of account manager capability is required to list information for any other group(s) in the account.

Access Control Definitions

Access codes provide one level of security. They provide security for accounts and groups.

Another level of security works on individual files. It allows you to protect specific files (or collections of files). This second level of security is called access control definitions (ACD). ACDs are provided to meet the requirements of National Computer Security Center.

You might not encounter—or need to use—ACDs unless your organization needs special security for your files. But, because you or someone in your organization *can* set ACDs for the files on your system, you should be aware of what they are.

You can create ACDs for any file that you (your *username.accountname*) have created.

With an ACD, you can prevent others from using, copying, or even looking at one of your files.

Account managers (AM) can create ACDs for any file in the account that they manage. System managers (SM) can create ACDs for any file on the computer system that they manage. This means that you cannot use an ACD to protect a file from use by your account manager or your system manager.

Creating an ACD

There are several ways to create ACD protection for a file. The way the most people will use is the `ALTSEC` command. The `ALTSEC` command is a shorthand way of saying "alter security." It creates or changes or removes ACD security provisions for a file.

Suppose that you have logged on as `JOHN.BUDGET,TAXES` (user `JOHN` in the group `TAXES` in the `BUDGET` account). You want to protect a file called `TAXRETRN`.

You could do this:

```
ALTSEC TAXRETRN;NEWACD=(R,W:JOHN.BUDGET)
```

You have just given yourself (`JOHN.BUDGET`) permission to read (look at) and write (to change) this file.

At the same time, you have denied everyone else permission to use this file in any way.

Note

If you are the owner (the creator) of the file TAXRETRN, you do not have to give yourself permission to use the file. The owner of this file, the user who logs on as `JOHN.BUDGET`, automatically has full access to the file. In fact, you—the owner of a file—cannot take away your own access to that file.

Before you attach an ACD to a file, be sure that you are not accidentally denying access to someone who *should* have access.

There are seven kinds of "permission" you can assign to a file when you decide to protect the file with an ACD.

Table 2-1. ACD Permissions

	Meaning	Permission
R	Read	The user who has this permission can look at or copy the file.
W	Write	The user who has this permission can add information anywhere in the file. This implies the ability to change existing information.
A	Append	The user who has this permission can add information to the file, but only at the very end of the file. Append permission does not give a user the ability to change existing information.
L	Lock	The user who has this permission can "lock" the file. Locking a file prevents two or more people from making changes to a file at the same time. This situation—concurrent access of a file—can arise if two or more people are updating (changing) information in the same file at the same time. This is less likely to happen with text files, but it can happen with database files . Whose changes are going to be the "real" changes to the file? Locking the file allows only one person at a time to make changes and ensures that two or more people are not competing for the same file at the same time.
X	Execute	The user who has this permission can run (execute) the program file to which this ACD is attached. Execute has no meaning for text and data files.
NONE	None	The user who has this permission has *no* permission to use the file at all.
RACD	Read/Copy	Read and copy the ACD permission file. The user who has this permission can copy the ACD security provision associated with the the file.

What an ACD means

An ACD has two parts. One part describes the access permission attached to a file. The other part describes the user, or users, who will have this access:

```
ALTSEC TAXRETRN;NEWACD=(R,W:JOHN.BUDGET)
```

The first part of this ACD attaches Read and Write permission to the file TAXRETRN.

```
ALTSEC TAXRETRN;NEWACD=(R,W:JOHN.BUDGET)
```

The second part of this ACD gives the permission to the user `JOHN.BUDGET`. Users who are not mentioned in this ACD definition

do not have any access to the file. (Account managers and system managers automatically have access to files: account managers "own" all of the files in their account(s); system managers "own" all of the files on their system.)

Notice that the two parts are separated by a colon (:).

You can make several ACD assignments at once, as in this example:

```
ALTSEC TAXRETRN;NEWACD=(R,W:JOHN.BUDGET;R:MARIA.BUDGET; &
NONE:MARK.BUDGET,@.SALES;)
```

Notice that each *pair* of assignments is separated by a semi-colon.

The inclusion of `@.SALES` means that all of the users who can log on to the `SALES` account have access to this file. (`@.@` means all of the users on a system.)

Adding ACDs to a File

You might decide later that another user should also have access to `TAXRETRN`. Perhaps user `BOB.ACCTNG` will need to be able to look at `TAXRETRN`.

To do that use the `ALTSEC` command again, this time with the `;ADDPAIR` keyword. Do this:

```
ALTSEC TAXRETRN;ADDPAIR=(R:BOB.ACCTNG)
```

Excluding or Removing Permission to Use a File

Someone might decide that the user `MARIA.ACCTNG` really should not have access to the file. To exclude this user from access to the file, enter this:

```
ALTSEC TAXRETRN;DELPAIR=(MARIA.ACCNTG)
```

Removing All ACDs

You might need to remove all access control definitions from a file. To do that with `TAXRETRN`, enter this:

```
ALTSEC TAXRETRN;DELACD
```

Now all ACD restrictions are removed from `TAXRETRN`.

More information in using `ALTSECT` to manage ACDs can be found in the *MPE/iX Commands Reference Manual* (32650-90003), which you may order from your sales representative.

Group Passwords (account manager)

If you ever forget a group password, you can use the `LISTGROUP` command with the `PASS` parameter to list the group's password to the screen. Try that command now for the `PROJECT` group.

```
LISTGROUP PROJECT;PASS
```

Adding a New Group (account manager)

If you have account manager (AM) capability or higher, you may add a new group to your account. The command that you use to do this is:

`NEWGROUP`

Exercise 1-2: Using NEWGROUP (account manager)

The purpose of this exercise is to illustrate how to create account groups with default attributes as well as those with attributes that you specify. The `NEWGROUP` command with its parameters is listed below.

```
NEWGROUP name;PASS=password;CAP=capabilities;ACCESS=(attributes:users)
```

Notice

Each of the attributes, `PASS`, `CAP`, `ACCESS`, specifies one or more values preceded by an equal sign: (`PASS=`*xxxx*;`CAP=`*xx,xx*). Access codes are enclosed in parentheses and separated by commas: `ACCESS=(R,L,A:GL)`. For example, to ensure that only the account librarian (AL) would have read (R) and execute (X) access to files in a group, you would enter `ACCESS=(R,X:AL)`.

1. Use the `NEWGROUP` command to create a group called STOP with default attributes.
2. Use the `LISTGROUP` command to list the attributes of this group.
   a. What default capabilities have been assigned?
   b. What default file access and user codes have been assigned?
   c. What command would you use to check on this group's password?

Note

Any mistakes that you may make can be corrected with the `ALTGROUP` command, which is covered later in this lesson.

3. Create a new group called GO with the following attributes:

Password:	Fast
Capabilities:	Interactive access (IA), batch access (BA), and process handling (PH)
File Access:	Default

4. Use the `LISTGROUP` command to verify the new group and its password.
5. Use the `NEWGROUP` command to create another group SLOW with the following attributes:

Password:	Down
Capabilities:	Default
File Access:	read (R), lock (L), append (A), write (W), execute (X), save (S) for the group librarian (GL) only.

6. Use the `LISTGROUP` command to check on the new group.

********** End of Exercise 1-2 **********

Altering Group Attributes (account manager)

The ALTGROUP command is used to change group attributes; however, like the NEWGROUP command, the ALTGROUP command also requires account manager level capability or higher for its use.

```
ALTGROUP groupname;attribute=
```

The ALTGROUP command can be used to change any attributes, provided that these attributes do not exceed the limits for your account. To change any attributes to default, list the attribute followed by an equal sign. For example, to change the file access attributes to default on the PROJECT group, use this command:

```
ALTGROUP PROJECT;ACCESS=
```

Exercise 1-3: Using ALTGROUP (account manager)

1. Use the ALTGROUP command to do the following:
   a. Change the password on the PROJECT group to PJ.

      (If you have forgotten the password, use the PASS parameter with the LISTGROUP command.)

   b. Change the capabilities on the GO group to default.
   c. Change the file access capabilities on the SLOW group to read, write, append, and execute for any user of the group.
   d. Remove the password for the SLOW group.
2. Check the groups that you created in the previous exercise and modify their attributes for additional practice.
3. Use the LISTGROUP command to check on each of the groups.

********** End of Exercise 1-3 **********

Exercise 1-4: Using ALTSEC

1. Use the ALTSEC command to give yourself Read and Write permission for one of the files that you have created.
2. use the ALTSEC command to give another user Read permission for this file.

********** End of Exercise 1-4 **********

Purging a Group (account manager)

In order to delete a group from your account, use the PURGEGROUP command followed by the group name. Enter this now.

```
PURGEGROUP GO
```

The PURGEGROUP command can have devastating consequences. It erases the group and all files included therein. The system prompts you:

```
PURGE GO TO BE PURGED? (YES/NO)
```

Enter the word

```
yes
```

and press RETURN to complete the deletion.

Use the PURGEGROUP command to delete the STOP and SLOW groups.

Using the MKACCT Command File to Create Accounts, Groups, and Users

There is an alternative to using MPE/iX commands to create the accounts, groups, and users on your system. The command file MKACCT (MKACCT.MPEXL.SYS) provides you with an easy, interactive method of creating your account structure.

Figure 2-3 shows a sample worksheet that can be used to plan your structure or to record the structure that you have created. It shows how you might plan a new account that you wish to create. This sort of information will help you in deciding (and remembering) the kind of account structure that you want to create.

```
ACCOUNT Name:     PRACTICE_____       Password:    NOW_________
        Manager:  MGR_________        Password:    FY91________

        Group:    TAXES________       Password:    TX1992______
        Group:    BUGET________       Password:    ____________
        Group:    ____________        Password:    ____________
        Group:    ____________        Password:    ____________

        User:     BARB_________       Password:    ____________
                                      Home Group:  TAXES________
        LOGON:  HELLO BARB_________.PRACTICE______,____________
                        (user)        (account)       (group)

        User:     JIM__________       Password:    ____________
                                      Home Group:  PUB__________
        LOGON:  HELLO JIM__________.PRACTICE______,____________
                        (user)        (account)       (group)


        User:     ____________        Password:    ____________
                                      Home Group:  ____________
        LOGON:  HELLO ____________.______________,____________
                        (user)        (account)       (group)

        User:     ____________        Password:    ____________
                                      Home Group:  ____________
        LOGON:  HELLO ____________.______________,____________
                        (user)        (account)       (group)

        User:     ____________        Password:    ____________
                                      Home Group:  ____________
        LOGON:  HELLO ____________.______________,____________
                        (user)        (account)       (group)
```

Figure 2-3. Sample Worksheet

To create an account, groups and users

You must be logged on as MANAGER.SYS to use the MKACCT command file. Follow the MKACCT online (on screen) instructions, or use the instructions provided here to guide you along.

Caution

If you are taking notes as you create users, group, accounts and passwords, remember that it is the system manager's responsibility to keep this information secure.

1. Log on as MANAGER.SYS.
2. Type MKACCT.MPEXL at the system prompt, and press Return.

   :MKACCT Return

   The following MKACCT information appears on your screen:

```
*************************************************************************
*  MKACCT allows you to create: ACCOUNTS, GROUPS, and USERS.            *
*                                                                       *
*  ACCOUNTS contain GROUPS and USERS.  GROUPS contain your files.       *
*  The next screen shows a picture of an account structure.             *
*                                                                       *
*  All USERS in the same ACCOUNT can share files, but each USER         *
*  should have a GROUP where their own files reside.                    *
*                                                                       *
*  USER, ACCOUNT and GROUP names are needed to logon to your system,    *
*  for example:                                                         *
*                                                                       *
*     :hello USER.ACCOUNT,GROUP   or                                    *
*     :hello MANAGER.SYS,PUB                                            *
*                                                                       *
*  Each USER should be assigned a HOME GROUP, so that a group name is   *
*  not needed when the user logs on.  For example, the home group for   *
*  MANAGER.SYS is PUB so that the system manager can logon as:          *
*                                                                       *
*     :hello MANAGER.SYS                                                *
*************************************************************************
```

   Press the Return key to continue.

   The next screen shows you a diagram of the account structure: how files belong in groups, and groups belong in an account:

Note

When you are prompted for Yes/No/Exit, your options are these:

- Type YES or Y and press Return. This action tells MKACCT to continue with what it is doing.

  In most instances the YES response will be underlined: YES. Pressing Return is the same as responding with the underlined choice. If YES is underlined, Return is the same as Yes.

- Type NO or N and press Return. This action tells MKACCT to ahead to the next step, if there is one.
- Type EXIT or E and press Return. This action tells MKACCT to stop where it is without performing the next step, if there is one. MKACCT will ask you to confirm your intention to quit. Typing EXIT (or E) one more time will exit you out of the command file to the system prompt.

3. After viewing a diagram of an account structure, MKACCT asks if you want to create a new account> If you want to create an account, respond YES.

   Do you want to create a new account (Yes,No,Exit)?

   Again you are given three options to respond to this prompt.

- If you type NO, MKACCT will ask you if you want to add groups or users to an existing account. For more information on this topic, see the following task, "Using MKACCT to add groups or users to accounts."
- If you type EXIT, MKACCT will stop and ask you to confirm your intention of stopping MKACCT.
- If you type YES, MKACCT will start the process for creating an account.

4. When you are ready to create an account, type YES, or press (Return), at this prompt. You must respond to a sequence of prompts requesting a name for the account, a name for the account manager, and a password for the account and account manager. Passwords are optional; however, the lack of passwords diminish the security on your system. The following example displays the sequence of prompts that appear, along with sample user responses.

```
Do you want to create a new account (Yes,No,Exit)?Yes (Return)

Every ACCOUNT must be named and must include a manager (USER).
Every account and manager should have a password.  Accounts
and user without passwords are not secure.

If there are any problems, you will be asked to repeat the
process.

Please enter a name for this ACCOUNT: SALES (Return)

Please enter a password for account SALES: FY91 (Return)

Please enter a manager name for account SALES: YUKI (Return)

Please enter a password for account manager YUKI: KINU2 (Return)
```

After entering a password for the account manager, a screen similar to the following will appear:

```
The following account and user will be created:

Account name          :SALESh
Account password      :FY91

Manager name          :YUKI
Manager password      :KINU2

Create the ACCOUNT (Yes,No,Exit)?
```

5. Type YES, or press (Return). Doing so will finalize the creation of the account, the account's password, the account manager, and the account manager's password.
6. MKACCT then asks, "Do you want to create a new group in the *ACCTNAME* account (Yes,No,Exit)?".

   To create a new group, type YES, or press (Return), at this prompt.

   The MKACCT process for creating groups is similar to creating accounts. You must provide a group name and password (optional) when prompted.

Answer YES when the "Create this group (Yes,No,Exit)?" prompt appears to finalize the creation of the group.

MKACCT will continue to prompt you for new group names until you type NO at the "Do you want to create a new group ... ?" prompt. You do not need to create a PUB group. MKACCT does this automatically.

7. MKACCT then asks, "Do you want to create a new user in the *ACCTNAME* account (Yes,No,Exit)?".

   To create a new user, type YES, or press Return, at this prompt.

   The MKACCT process for creating users is similar to creating accounts and groups. You must provide a user name and password (optional). You will be asked if you want to see a list of groups in your new account. Answer accordingly.

   You will be asked to provide a home group for the user. Specifying a home group is optional; however, without a home group, the user must specify a group name each time they log on.

   Answer YES when the "Create this user (Yes,No,Exit)?" prompt appears to finalize the creation of the group.

   MKACCT will continue to prompt you for new user names until you type NO or EXIT at the "Do you want to create a new user ... ?" prompt.

   If you respond NO, MKACCT returns you to the prompt where you can start the process of creating another new account again.

8. To exit the MKACCT command file, type EXIT at any prompt that ends with the (Yes,No,Exit)? options. A screen similar to the following appears:

```
************************************************************
* MKACCT has kept a record of your work.  It produced two  *
* files in the PUB group that you may want to examine now  *
* or sometime later.  These files contain all of the work  *
* that you have done by using MKACCT.                      *
*                                                          *
*                                                          *
* CMDLOG   Contains the MPE/XL commands that were needed at *
*          each step along the way to create ACCOUNTS,     *
*          GROUPS, USERS and passwords.                    *
*                                                          *
* ACCTLOG  Contains a summary of the account structure that *
*          you have created, showing ACCOUNTS, their GROUPS, *
*          their USERS, all passwords, and the HOME GROUP  *
*          to which each user of each account was assigned. *
*                                                          *
* To see these two files enter:                            *
*                                                          *
* :PRINT cmdlog.pub                                        *
* :PRINT acctlog.pub                                       *
*                                                          *
* Since these files contain passwords, it is your          *
* responsibility as system manager to keep this information *
* safe from misuse.                                        *
*                                                          *
************************************************************
Please press RETURN to continue
```

9. After carefully reading this screen, press Return to view the next screen.

```
*************************************************************
*                                                           *
*                                                           *
* To create more GROUPS, USERS or ACCOUNTS just execute     *
* MKACCT again.                                             *
*                                                           *
* The following MPE commands may be used to list and        *
* modify your accounts:                                     *
*                                                           *
* :LISTACCT  AccountName      :ALTACCT    AccountName       *
* :LISTGROUP GroupName        :ALTGROUP   GroupName         *
* :LISTUSER  UserName         :ALTUSER    UserName          *
* :PURGEACCT AccountName      :PURGEGROUP GroupName         *
* :PURGEUSER UserName                                       *
*                                                           *
*  You can get HELP for any of these commands by typing:    *
*                                                           *
*  :HELP CommandName      e.g. :HELP ALTUSER                *
*                                                           *
*************************************************************

Please press RETURN to continue  Return
```

10. To get to the system prompt, press Return.

    The following message will appear on your screen and you will be returned the system prompt.

```
End of the MKACCT command file.
:_
```

    You may run MKACCT as often as you like. The ACCTLOG and CMDLOG files will accumulate information and show the date and time of each MKACCT session.

Using MKACCT to add groups or users to an existing account.

1. Log on as MANAGER.SYS.
2. Type MKACCT at the system prompt and press Return

```
:MKACCT  Return
```

3. At the "More information (Yes,No,Exit)?" prompt, type NO.
4. Type YES, or press Return, at the following prompt:

```
Do you want to add groups or users to an existing
ACCOUNT (Yes,No,Exit)? YES  Return
```

5. Provide the name of the account to which you wish to add groups or users at the following prompt:

```
Please enter the name of an existing ACCOUNT:
```

6. When prompted, supply a new group name and password. MKACCT will continue to prompt you for new group names until you type NO at the "Do you want to create a new group in the *ACCTNAME*?" prompt.

7. When prompted supply a new user name and password. MKACCT will continue to prompt you for new user names until you type NO or EXIT at the "Do you want to create a new user in the *ACCTNAME*?" prompt.
8. When you have finished adding new groups and users to your existing account, type EXIT at any prompt that ends with the (Yes,No,Exit)? options.

Making changes later

- To delete any portion of your account structure, use the PURGEACCT, PURGEGROUP or PURGEUSER commands.
- To modify capabilities or file access for any account, group, or user, use the ALTACCT, ALTGROUP, or ALTUSER commands.
- To change a password, use the PASSWORD command.

Problems with MKACCT

MKACCT is a command file that helps you establish user names, groups, accounts and passwords. If you have any problems with MKACCT you may restart it at anytime without damaging your system.

MKACCT is interactive, and if you make a mistake, it prompts you for the correct response.

You may verify the results of MKACCT with the LISTACCT, LISTGROUP, and LISTUSER commands. You change anything accomplished by MKACCT with the ALTUSER, ALTGROUP, and ALTACCT commands.

Capability Requirements for Applications and Programs

Before you move an application or program into a group, verify its capability requirements with the VERSION utility. In order for the application to execute, the group in which it is stored must have those capabilities.

For example, if you want to load the EDIT/3000 text editor application (henceforth referred to as *the editor*) into one of your account groups, first check the editor's capability requirements:

```
VERSION EDITOR.PUB.SYS
```

Notice the editor's capability requirements at the bottom of the display:

```
# SEG: 12
STACK: %10770
MAXDATA: %23420
TOTAL DB: %1735
DL:%0
CAP: BA,IA,DS
```

Any group in which the editor application will be stored must have BA, IA, and DS capabilities in order for the application to run successfully. Of course, any general user with default capabilities can use the editor successfully. Only the account and group where the editor resides, namely the PUB group of the SYS account, need have the special capabilities.

Lesson Summary

1. A minimum of account manager capability is required to do the following:
   a. Display information about all groups in your account.
   b. Create, modify, and delete account groups.
   c. List all passwords in your account.
2. The MKACCT command file allows a system manager to create accounts, groups, and users on the system in a question-answer format.
3. The REPORT and LISTGROUP commands provide information on groups within an account.
4. The capabilities for any group may not exceed those of the account.
5. Before loading an application or executable program into one of your account groups, check its capability requirements. If the requirements exceed those of the group, the application will not execute.

Exercise 1-4: Lesson 2 Review

Match the commands with what they do:

Command:
```
PURGEGROUP
NEWGROUP
VERSION
REPORT
LISTGROUP
LISTGROUP name;PASS
ALTGROUP
ALTSEC
```

a. Provides detailed information on each group in the account.

b. Allows you to change the attributes of a group.

c. Lists file space, CPU time, and connect time for all groups in an account.

d. Creates a new group in an account.

e. Lists detailed information on a group including the group's password.

f. Creates or changes ACD protection for files.

g. Checks capability requirements for applications and executable programs.

h. Deletes a group from an account.

********** End of Exercise 1-4 **********

Lesson 3
Managing Users

Introduction

Lesson 3 presents the commands associated with the following user-related tasks:

- displaying user information
- creating new users
- Altering characteristics of existing users

You have logged on to the system and started an interactive session. Previously, you used SHOWJOB to look at some of the users on the system. You will now learn more commands to help you manage and monitor those users.

The SHOWME command displays the account, group, and user information associated with your session. You can also display information about other users on the system or in your account with the LISTUSER command. You can use the online MPE/iX help facility to learn more about this command and other related account management commands.

Online help

As you know, an online help facility is a quick and easy way to obtain information about a command without referring to a reference manual; however, if you need in-depth information, the actual hard copy manual should be used, since the online help information is sometimes abbreviated.

To use online help, enter:

```
HELP
```

You should now get a "table of contents" of MPE/iX topics. Enter the appropriate topic to give you a listing of all of the classes of commands available, CLASS.

Now enter the correct class to get a listing of all resource management commands, namely, RESOURCES.

Command Syntax and the LISTUSER Command

You should now see a listing of resource management commands. examine the commands and find listuser. at the help prompt (>), you can enter individual command names.

Do so now to get information about LISTUSER.

Q1-9	According to online help, what does the LISTUSER command do?

Have you found a display of the syntax for LISTUSER? According to syntax convention, the command name and any other required keywords are displayed in CAPITAL letters. You may enter these words at your terminal in either uppercase or lowercase. All

parameter names are displayed in lowercase letters. If the parameters are required (no brackets), you must replace them by actual values.

Optional parameters are shown surrounded by brackets [ ]. You do not have to specify values for such parameters. If you do not specify an optional parameter in a command line, a default value(s) is assigned. When the syntax shows that a comma is associated with an optional parameter, you must omit that comma, too, when you omit the parameter. Omitted, optional parameters default to some specified value (as dictated by the syntax).

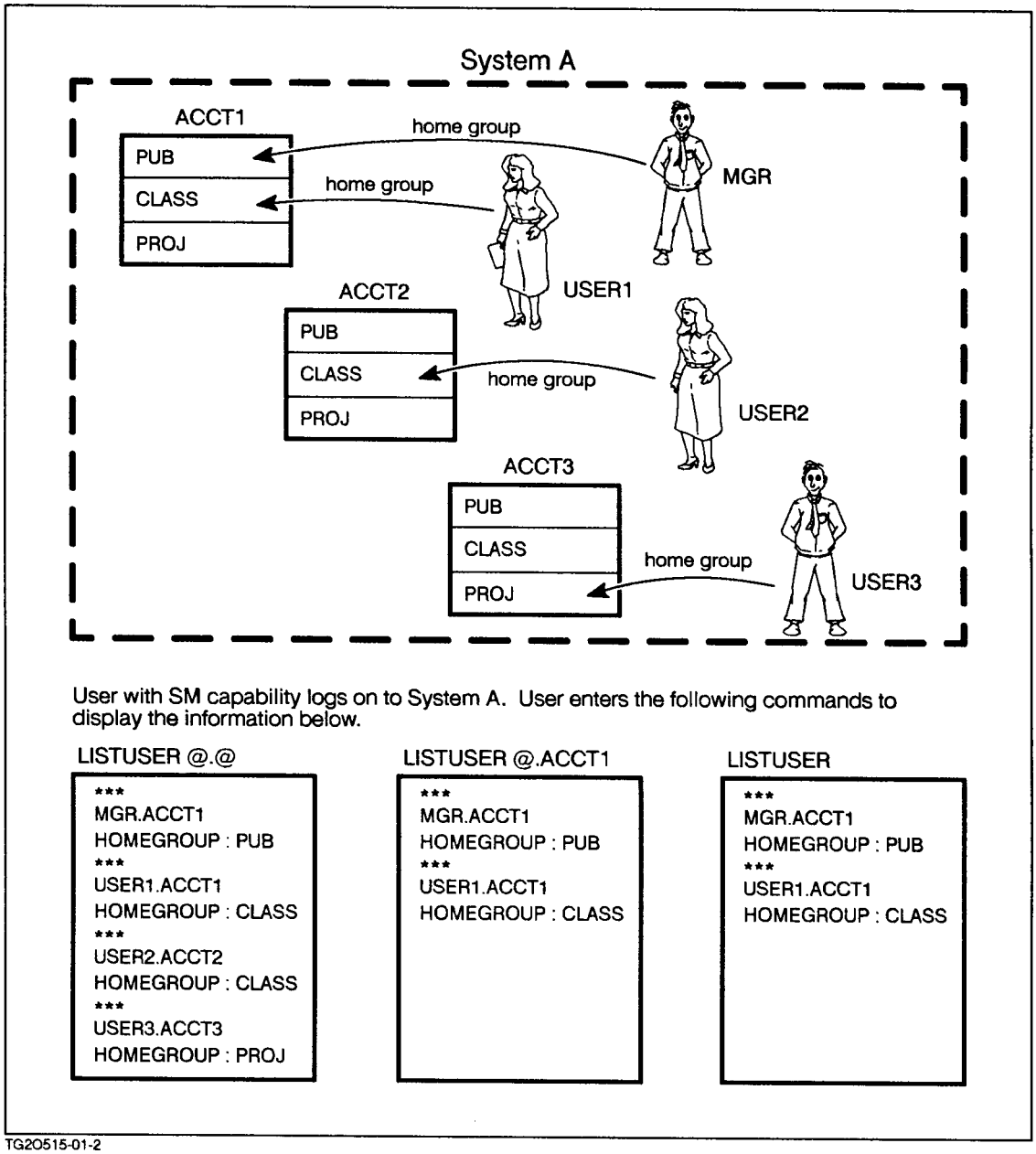


Figure 2-4. LISTUSER Command

Here is a simple example of command syntax:

```
LISTUSER [userset][,listfile][;PASS]
```

Here is an example of the corresponding command:

```
LISTUSER @.MYACCT
```

`userset`	Is all of the users (`@`) in the MYACCT account.
`listfile`	Defaults to the terminal (you see the listing there). Note that the comma (`,`) associated with `listfile` is omitted because `listfile` is omitted.
`PASS`	Is not specified, so no passwords are displayed. Note that the semicolon (`;`) associated with the password is omitted because `PASS` is omitted.

Note

More than likely, you will want to stop the `LISTUSER` listing occasionally to study the resulting terminal display. To do so, use the [STOP] key, if your keyboard has one—pressing [STOP] another time to let the display continue. Or you may use [CTRL] [S] to stop the display and [CTRL] [Q] to resume the display.

To get more information about parameters, their use, and examples associated with a specific command, simply type `PARMS`, `OPERATION`, or `EXAMPLE` at the help prompt.

To return to the original syntax display, enter the command again at the help prompt.

Exercise 1-5: Using LISTUSER

1. According to the `LISTUSER` syntax, which parameters are optional?
2. If you leave out these parameters, to what values do they default?
3. Use either of the following `LISTUSER` commands to list all of the users in your ACCTx account:

   ```
   LISTUSER @.ACCTx
   ```

   or

   ```
   LISTUSER
   ```

   a. How many users exist and who are they?

   b. What command would you use to list all of the users on the system? What capability must you have to do this?

4. According to the `LISTUSER` syntax, what would happen if you allowed the value of the `userset` parameter to default and entered the following command? What would happen if you left out the comma?

   ```
   LISTUSER,INFO;PASS
   ```

Hint: After executing the command, use the PRINT command to print the contents of INFO.

5. Try to list all of the users on the system. You get this message:

```
EXECUTING THIS COMMAND ON ALL ACCOUNTS REQUIRES SYSTEM MANAGER
CAPABILITY
(CIERR 724)
```

   You cannot look at all of the users because you do not have system manager capability. You will learn about system manager capability and other capabilities later in this lesson.

6. How would you modify the LISTUSER command you entered in question 3a so that it would display the passwords associated with the users in your account? What capabilities do you need to do this?

********** End of Exercise 1-5 **********

User Capabilities

Notice the type of information that is displayed for LISTUSER. Pay particular attention to the CAP information. The capabilities of the user are shown by two-letter capability codes.

Q1-10	What are the capability codes?
Q1-11	What capabilities must you have to list all users and their capabilities in your account? On the system?
Q1-12	If you do not have the capabilities specified in the previous question, what users can you list?

ALTUSER Command

The ALTUSER command changes a user's capabilities or passwords. This command is useful for increasing security so that other people logging on to your account must check with you for the current user password.

Users who wish to change their own user password may do so by using the PASSWORD command. (Refer to module 4, lesson 5, "Changing User Passwords," in *Fundamental Skills*).

The basic syntax of this command is

```
ALTUSER username;CAP=capabilities;PASS=password
```

If you change the capabilities or password of a user in your account, the changes do not go into effect until the specified user logs off and logs on again.

Use the HELP command to answer the following questions about ALTUSER:

Q1-13	Suppose that there is a user called MYUSER in your account, and you wish to change that user's password to MYPASS. Which command would you use? a. How would this affect MYUSER during the current session? After MYUSER logs off and logs on again? b. What capabilities must you have to change a user password for a user in your account? To change a user password for a different user in a different account?

Now change your own password to MYPASSX (X= user number). To do so, enter this:

```
ALTUSER USERx;PASS=MYPASSX
```

Use LISTUSER with the PASS option to verify that the change has occurred. Then change your password back to your original password. Verify the current password again:

```
LISTUSER USERx;PASS
```

NEWUSER Command

As account manager, besides changing an existing user's password or capability, you may also create a brand new user in an account. The NEWUSER command does this. Use the HELP command or the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) to determine the syntax for the NEWUSER command.

Notice that the syntax is the same for both ALTUSER and NEWUSER. Also, the required and optional parameters are the same.

Create a new user, USERxA (where X = your user number), in your account (ACCTx), with a home group of CLASS. Let its capabilities default. The password for this user will be UPASSxA.

```
NEWUSER USERxA.ACCTx;PASS=UPASSxA;HOME=CLASS
```

Q1-14	What are the capabilities of this new user?

Create another new user in your account, USERxB (where X= your user number). Give this user the same capabilities as USERxA, plus the capability to create volumes and do process handling. Let the home group be PUB. The password for this user will be UPASSxB.

```
NEWUSER USERxB.ACCTx;PASS=UPASSxB;CAP=ND,SF,CV,PH,BA,IA;HOME=PUB
```

Q1-15	What error message do you get and why? Has the user been created and given capabilities?

You now have an account with three users, each of whom has slightly different capabilities:

USERx: AM,IA,BA,SF,ND,PH

USERxA: IA,BA,SF,ND

USERxB: IA,BA,SF,ND,PH

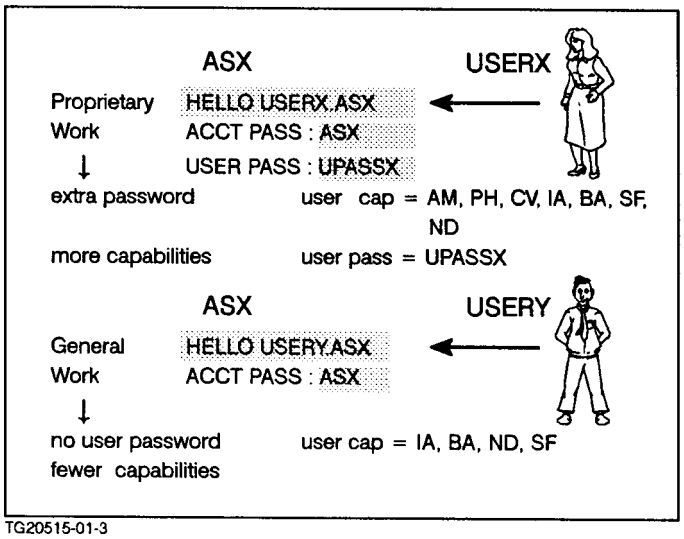


Figure 2-5. User Capabilities in One Account

As long as the users' capabilities do not exceed the account capability, an account may have multiple users who have different capabilities. This provides a way to implement a security scheme that controls the capabilities of individual users within the same account. In this way, people can log on as one user to do more proprietary work (they must know the user password) or log on as another user to do more general work (perhaps no user password is necessary).

PURGEUSER Command

You can also remove the users that you create in your account by using the PURGEUSER command.

At this point, remove USERxB from your account. To do so, enter:

```
PURGEUSER USERxB
```

Your account should now have two users associated with it, USERx and USERxA. In future lessons, please continue to log on as USERx until told otherwise.

Lesson Summary

1. The online help facility displays information on specific MPE/iX commands which are also described in the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).
.
2. The basic syntax conventions for MPE/iX commands are
   - Keywords are displayed in UPPERCASE letters.
   - Parameters are displayed in lowercase letters.
   - Optional parameters are displayed in lowercase letters, enclosed in square brackets [ ].
   - When commas (,) or semicolons (;) are associated with optional parameters, they must be omitted whenever the parameters are omitted and allowed to default.
   - Commas associated with required parameters must be supplied.
   - Required parameters have no brackets around them.
3. The `LISTUSER` command can display information about all of the users within your account.
4. The `ALTUSER` command changes a user's capabilities or password or (The `PASSWORD` command allows a user to
5. The `PURGEUSER` command deletes a specified user from your account.
6. The `NEWUSER` command creates a new user in your account.

3

Module 2: File Management

Module 2 presents the following lessons on creating and using files and file equations:

Lesson 1	Introducing Files
Lesson 2	Building Disk Files
Lesson 3	Creating System-Defined Temporary Files
Lesson 4	Using File Equations

Challenge Test

1. Which commands provide the following file information?
   a. record and (sector) space information
   b. creator and modification information
   c. hex file information
   d. logical record size information
   e. fully qualified file names by group
2. Identify a, b, and c as either a temporary file, formal file designator, or permanent file.
   a. file name 1-8 characters in length, stored on disk, created by user
   b. file name begins with $, lasts only for the duration of the session
   c. "file alias," refers to another file or device to which data is written
3. Write the command to build a disk file with the following characteristics: file name FILE1, record size of 80 bytes, fixed-length records, and ASCII file type.
4. Identify the following system file designators.
   a. `$STDIN`
   b. `$STDLIST`
   c. `$NULL`
   d. `$NEWPASS`
   e. `$OLDPASS`
5. What commands do you use for these?
   a. list file equations
   b. list temporary files
   c. cancel file equations

6. Write a file equation that designates file PRT as the line printer (LP). Then use the `PRINT` command to print the contents of MYFILE1 on the line printer. (The command must reference file PRT).

Note

Lesson 3 is required for programmers and anyone taking *Using the 900 Series HP-3000: Programmer Skills*.

Lesson 1
Introducing Files

Introduction

Lesson 1 presents the following information:

- LISTFILE parameters to display file information
- an introduction to permanent files
- an introduction to temporary files (user-defined and system-defined)
- An introduction to formal file designators

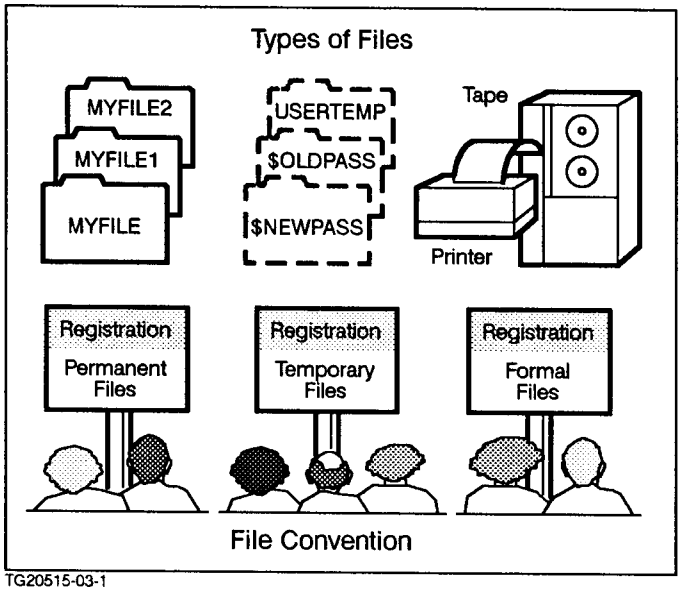


Figure 3-1. File Types

The MPE/iX file system controls the processing of all files, including their storage and transfer between various devices. Most of your work on the system involves files; therefore, you must know about the different kinds of files available for your use, as well as those used by the system.

In general, the term "file" is used in MPE/iX in three ways:

- permanent disk files
- temporary disk files (both user-defined and system-defined files)
- formal files (often called file "aliases")

Permanent files

Permanent files are sometimes referred to as "user files" since they are created directly by you, the user. You are probably already familiar with the file naming requirements, saving and purging commands, and basic LISTFILE commands to list your permanent files; however, as you continue to create and use more permanent files in MPE/iX, you need to know more about this kind of file, its characteristics, and how to get information about permanent files.

LISTFILE command options

Besides being able to provide you with sorted listings of permanent files within groups in your account, the LISTFILE command has a number option that can provide you with detailed information about any or all of your permanent files. The general form of the LISTFILE command is:

```
LISTFILE filename, #
```

Here, # represents an integer.

Often, the LISTFILE printout can be quite long. You can press BREAK to return to a system prompt.

Depending upon the syntax you use, the LISTFILE, # command gives you specific information:

- for all files in your account

  ```
  LISTFILE @.@,#
  ```

- for a single file in your current group

  ```
  LISTFILE filename,#
  ```

- for a single file in a group other than your current group

  ```
  LISTFILE filename.groupname,#
  ```

LISTFILE,1

Make sure that you are currently logged on to the CLASS group. Try entering this command from your keyboard:

```
LISTFILE,1
```

What you should see is an alphabetical listing of files in your CLASS group with their logical record size information.

How would you go about listing the same information only for MYFILE1? If you're in the CLASS group you only need enter the file name after the LISTFILE command.

```
LISTFILE MYFILE1,1
```

Your display should look like this:

```
ACCOUNT = xxxxxxxx          GROUP = xxxxxxxx

FILENAME   CODE   ---------------LOGICAL RECORD---------------
                  SIZE        TYP      EOF       LIMIT

MYFILE1            72B        FA       18           18
```

The display shows that the SIZE of MYFILE1 is 72 bytes. It also notes that MYFILE1 is a fixed-length ASCII file (TYP) and currently holds 18 records. (EOF means the end of the file.)

Can you get logical record information for the file, JOB1? Try that now.

Which record size is larger, JOB1 or MYFILE1? (Compare the sizes.)

Record size information is helpful when you want to join files together or use a word processing application to make changes to a file.

If the application cannot accept the file, you see the message:

```
CANNOT EDIT FILES OF THIS TYPE
```

Note You can enter `LISTFILE filename,SUMMARY` to get the same display provided by `LISTFILE filename,1`.

LISTFILE,2

Suppose that you want to see how much space your files are occupying on the disk. Try the `LISTFILE` command with option 2 for files MYFILE1 and JOB1.

```
LISTFILE MYFILE1,2
```

```
LISTFILE JOB1,2
```

Space information, expressed in terms of sectors, is provided in addition to the information from option 1. How would you modify this command so that you get information for all files in the `PUB` group of your account? Use wildcard options in the command.

```
LISTFILE @.PUB,2
```

Enter this command from your keyboard now. There is only one file in the `PUB` group.

The `LISTFILE,2` command option clearly can help you track the amount of permanent space used by your files.

Note You can enter `LISTFILE filename,DISC` and get the same display provided by `LISTFILE filename,2`.

LISTFILE,3

`LISTFILE,3` provides information on each file's structure, access codes, creation, and modification dates.

For example, how would you find out when MYJOB2 was last modified?

```
LISTFILE MYJOB2,3
```

Enter that now and review the information displayed on your screen.

When was MYJOB2 created? Last modified?

Note You can enter `LISTFILE filename,DETAIL` and get the same display provided by `LISTFILE filename,3`.

If you have SM capability, you can enter `LISTFILE filename, 3;PASS` and see the creator of the file and its lockword, if there is one.

Q2-1	Match the LISTFILE commands with the information that they display.	
	LISTFILE,1	File structure, creation, and modification information
	LISTFILE,2	Logical record information
	LISTFILE,3	File space information

LISTFILE,4

The purpose of the LISTFILE,4 option is to provide you with complete access information by account, group, and file, for each file in your current group.

By entering LISTFILE,4 you see a display of information for all files in your current group.

Q2-2	How would you modify the LISTFILE,4 command to list complete file access information for all files in your account?

Note You can enter LISTFILE filename,SECURITY and get the same display provided by LISTFILE filename,4.

LISTFILE,5

The LISTFILE,5 command option normally shows exactly the same information displayed by LISTFILE,3. But if any of the files specified are KSAM or spool files, it displays information specific to those specialized files. Since no KSAM or spool files have been created in this lesson, LISTFILE,5 duplicates the LISTFILE,3 information.

Note You can enter LISTFILE filename,DATA and get the same display provided by LISTFILE filename,5.

LISTFILE,6

This is an easy one. The LISTFILE,6 command option provides you with an alphabetical listing of fully qualified file names. This command can be very helpful if you have a large account with many groups, and need to quickly find the group location of a particular file.

Enter the LISTFILE command with option 6 to get a listing of all files in your account.

Did you remember to include the wildcard option?

If not, enter

```
LISTFILE @.@,6
```

Note You can enter LISTFILE filename,QUALIFY and get the same display provided by LISTFILE filename,6.

LISTFILE,7

The LISTFILE,7 command option displays file information exactly the way that LISTFILE,6 does, But if any of the files specified are KSAM or spool files, LISTFILE,7 displays information specific to those specialized files. Since no KSAM or spool files have been created in this lesson, LISTFILE,7 duplicates the LISTFILE,6 information.

Note

You can enter LISTFILE filename,UNIQUE and get the same display provided by LISTFILE filename,7.

LISTFILE,-1

The LISTFILE,-1 command provides you with hex information on your file(s). If you are a programmer or system manager, you may want to explore the displays of this LISTFILE option. Remember that the help facility describes the command parameters of the LISTFILE command.

Note

LISTFILE,-1 requires AM capability.

You can enter LISTFILE filename,LABEL and get the same display provided by LISTFILE filename,-1.

LISTFILE,-2

The LISTFILE,-2 command provides you with security information on specific files. It shows all of the access control definitions that are in effect for a particular file.

You can use LISTFILE,-2 to see the access control definitions for the files you that have created. An account manager (AM) can use it to see the access control definitions for the files in the account she or he manages. A system manager (SM) can use LISTFILE,-2 to see the access control definitions for the entire computer system.

In order to see the access control definitions for a file that someone else created, you must have RACD (Read and Copy ACD) permission for the that file you wish to examine; or you must have SM capability; or you must be the account manager (AM) of the account in which the file is found.

Temporary files

In contrast to permanent files, a temporary file is one that resides in the job or session temporarily, and exists no longer than the job or session that created it.

Temporary files can be explicitly created by you, the user, or implicitly created by the system.

User-defined temporary files are created with the TEMP option of the BUILD command. The BUILD command is discussed in lesson 2 of this module.

System-defined files

System-defined temporary files are also referred to as “system files.” These files are automatically created and named by the system. Two such files are $STDIN and $STDLIST.

When you log on, your system automatically defines a file called $STDIN that corresponds to your input device, which is your keyboard. It also defines a file called $STDLIST to which the system sends output. $STDLIST refers to your terminal screen. System files are also automatically created during compiling and linking.

These temporary files are used by the system to hold data generated during processing. For example, the different steps in compiling and linking a program create several $NEWPASS (output) files. $NEWPASS files automatically change to $OLDPASS (input) files when the files close at the completion of the process. Lesson 3 discusses temporary files.

Q2-3	Have you noticed a unique characteristic of the names of system defined files?

You can also create a temporary file to hold the output from a LISTFILE command. Enter the following command at your keyboard:

```
LISTFILE,6 > FILELIST
```

The > instructs MPE/iX to create a temporary file and enter the output from the command into that temporary file. FILELIST contains the file listing generated by LISTFILE,6. The > redirection of output works with any of the LISTFILE options and with almost every other MPE/iX command.

To view this temporary file, enter:

```
PRINT FILELIST
```

PRINT displays the first 23 lines of the file and pauses.

It shows the number of the next line to be displayed and the number of the last line in the file, and it asks if you want to continue with the display.

You see something like this on the bottom line of your screen:

```
(24/37) Continue?
```

The next line to be shown is 24. The last line that can be displayed is 37.

Press RETURN each time you want to see another 23 lines of the file.

If instead you want to see a particular line, enter that line number and press Return:

```
(24/37) Continue? 31Return
```

In this example, PRINT would display line 31 and the next 22 lines following it.

If you do not want to see more of the file, enter N (for No) and press Return.

Listing temporary files

How do you list temporary files? Use LISTFILE;TEMP. Enter that now and see if there are any temporary files in your account.

```
LISTFILE;TEMP
```

Unless you have been doing some programming or creating additional temporary files, you should only see the FILELIST that you just created.

```
TEMPORARY FILES FOR USERx.ACCTx,CLASS

FILELIST.CLASS.ACCTx
```

Purging temporary files

Purging temporary files is done with the PURGE command as follows:

```
PURGE filename,TEMP
```

Enter this command now to purge FILELIST.

```
PURGE FILELIST,TEMP
```

Formal files

Formal files differ from permanent and temporary files in that they do not exist on tape or on disk; however, they follow the same file naming conventions as permanent files. A formal file is an alternate label, or alias. It is a name used by a program or in a file equation and it refers to another actual file or device. This name is referred to as a *file designator*.

A common example of a formal file that you might use one day is the EDTLIST file in the following equation:

```
FILE EDTLIST;DEV=LP
```

The EDTLIST file is now the designated file alias for the line printer (LP). Because this formal file refers to a device, it is often referred to as a "device" file.

Listing file equations

Since formal files are not stored on disk, they cannot be listed with the LISTFILE command; however, file equations using formal file designators can be listed with the LISTEQ command.

Enter the example of a file equation presented earlier:

```
FILE EDTLIST;DEV=LP
```

Now enter LISTEQ to list the formal files on your system.

```
LISTEQ
```

You should see the following:

```
FILE EQUATIONS
FILE EDTLIST;DEV=LP
```

On some systems, you might also see additional file equations:

```
FILE MAILPRNT;DEV=LP;ENV=ELITE.HPENV.SYS
FILE SLLIST;DEV=LP;ENV=ELITE.HPENV.SYS
```

Each of the file designators listed, MAILPRNT and SLLIST, refers to a device. Lesson 4 provides more information regarding these file designators.

Q2-4	What is similar about the file naming characteristics of formal and permanent files?

Lesson summary

1. Use the `LISTFILE,#` options to get information on your account files:

`LISTFILE,1`	Logical record information
`LISTFILE,-1`	Hex information
`LISTFILE,2`	File space information
`LISTFILE,3`	Creation, modification, access information
`LISTFILE,4`	User access information
`LISTFILE,5`	Like 3, but adds KSAM or spool file information
`LISTFILE,6`	Fully qualified file names
`LISTFILE,7`	Like 6, but adds KSAM or spool file information

2. Permanent files are stored on disk and can be listed with the `LISTFILE` command.
3. Temporary files are created by the system or users, are stored on disk and are listed with the `LISTFILE;TEMP` command.
4. Formal files are designated file aliases and can be listed with the `LISTEQ` command.

Exercise 2-1: lesson 1 review

1. Check your understanding of the characteristics of permanent, formal, and temporary files by noting to which file type(s) each characteristic applies.

Characteristic	Permanent	Formal	Temporary
a. File names are 1-8 characters long.			
b. File names are aliases for devices.			
c. File names may begin with a $.			
d. Files may be listed with `LISTFILE`.			
e. Files may be listed with `LISTEQ`.			
f. File names must begin with an alpha character.			
g. File names are listed with `LISTFILE;TEMP`.			

2. From each pair of commands, select the correct LISTFILE command that displays the information requested.

a.	Logical record and space allocation for all files in account.	`LISTFILE @.@,2` or `LISTFILE @.@,3`
b.	Hex information for MYJOB1 (requires AM capability).	`LISTFILE MYJOB1,-1` or `LISTFILE MYJOB1,7`
c.	Logical record information for MYJOB1 file.	`LISTFILE MYJOB1,1` or `LISTFILE MYJOB1,-1`
d.	Creator information for all files in your current group (requires AM capability).	`LISTFILE,3` or `LISTFILE,5`
e.	All files within your logon group.	`LISTFILE` or `LISTFILE @.@,1`
f.	Complete access and lockword information for all files in your current group (requires SM capability).	`LISTFILE,3;PASS` or `LISTFILE,4`
g.	Fully qualified file names for all files in your account.	`LISTFILE @.@,6` or `LISTFILE @.@,2`
h.	Creation, access, and modification information on files in the `PUB` group of your account.	`LISTFILE @.PUB,3` or `LISTFILE @.PUB,1`
i.	KSAM or spool file information for those specialized files.	`LISTFILE,7` or `LISTFILE,5`

********** End of Exercise 2-1 **********

Lesson 2
Building Disk Files

Introduction

You have had a brief introduction to different types of files: permanent, temporary, and formal. In lesson 2, you will learn how to use the BUILD command and the following options to create customized files:

- record size
- blocking factor
- record type
- file type

The BUILD command is useful when you need to change the file's format so that another program or subsystem can accept the format.

For example, using the editor, you can only display and edit files that contain data in a particular format. If you "inherit" a file from another system, it may not be in the correct format. For example, suppose that you enter the command:

```
LISTFILE @,6 > LISTFILE
```

This command generates a temporary file called LISTFILE. The > causes MPE/iX to put the display from this command into a temporary file.

Note

If you try to edit LISTFILE you will find that the editor truncates some of the lines if they are too long for the editor. For many temporary text files, this presents no problem. If you suspect that truncation will cause a problem, then change the file format by building a new file with the appropriate format. Copy the temporary file into the new file. Then edit the new file. Your original temporary file keeps its format while you edit the copy.

The BUILD command allows you to specify the "shape" of a file that as yet contains no information. Using BUILD is much like creating a container into which liquid will be poured; the contents will conform to the dimensions of the container. In the BUILD situation, data will conform to the characteristics of the empty file created by the BUILD command. The following illustration shows how the data in a file will take on the characteristics of its "container."

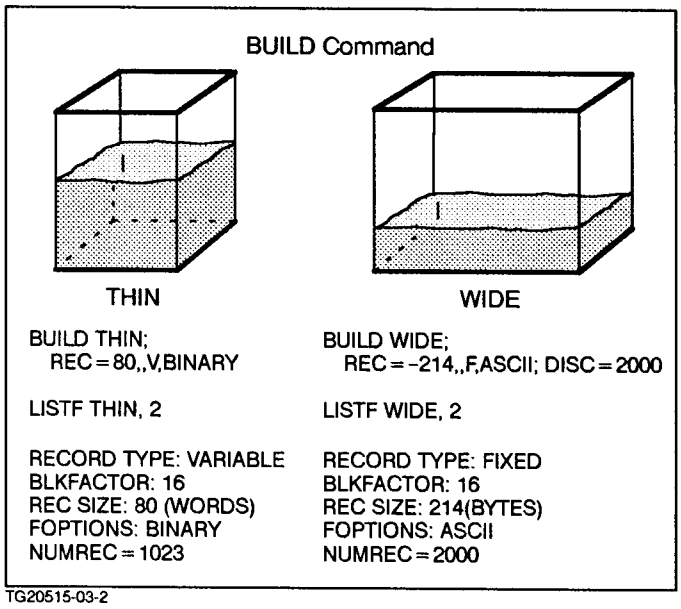


Figure 3-2. `BUILD` **Command**

Generally speaking, as a general user, you are not so much concerned with the characteristics of a file, as with the data in it. Usually, the subsystem that you are using (for example, some editor) automatically creates files with the appropriate characteristics that determine how the contents are stored on disk. As you become a more experienced user, you may become concerned with how to make programs run most efficiently by defining a file's characteristics more precisely.

Although you cannot alter the physical characteristics of an existing file, you can create a new file with the desired characteristics (using `BUILD`) and then copy the contents of the existing file into the new one (using `FCOPY`). The new file contents then have the appropriate characteristics. The `FCOPY` command is covered in module 4, File Transfer.

BUILD command

The `BUILD` command builds a new file. The options associated with the `BUILD` command determine the physical characteristics of the file.

Partial syntax for the `BUILD` command appears below. The complete syntax and option explanation can be viewed in the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) or in the online help facility.

```
BUILD filename[/lockword];REC=recsize,
blockfactor,rectype,filetype;options
```

Parameter	Description
filename	MPE/iX file name
lockword	Special password to restrict access to specific file
recsize	Record size (negative number is bytes, positive number is words)
blockfactor	Ratio indicating number of logical records per physical block (R/B on the LISTFILE columns)
rectype	Record type (F = fixed-length, V = variable length, U = undefined)
filetype	Type of data (ASCII,binary)
options	Device, file code, carriage control, and more

Consider a file that already exists. View the various characteristics specified by the BUILD command. Enter this command:

```
LISTFILE MYFILE,3
```

All questions about the BUILD command options refer to the display that you now see on the screen.

Note

Later, you might also wish to do a LISTFILE MYFILE,2 to compare the abbreviated titles with the full-length ones of the LISTFILE MYFILE,3 command.

Lockword

You've already worked with passwords at the account, group, and user level. MPE/iX provides a further level of security, lockwords. These are file "passwords," which restrict access to a particular file. This means that even if you know the account, group, and user passwords, you will not be able to access a particular file unless you know the lockword. When using the BUILD or RENAME command, use a forward slash (/) to delimit the lockword. The slash is not part of the lockword.

Note

A user with SM capability can use LISTFILE,3 with the ;PASS option to display the lockword.

For example, suppose you build a file called SECRETS:

```
BUILD SECRETS/KEY
```

Whenever you try to view SECRETS, you must also specify the lockword, KEY.

Be Careful

If you type a / when you are entering a file name, the system interprets everything after the / as a lockword. If that was *not* what you intended, you may have no recollection of it (and you may be unable to access the file).

For example, suppose that you accidentally type:

```
MY/FILE
```

You may think you have a file called MYFILE; however, the system interprets a file called MY with a lockword of FILE.

Record size

Record size refers to the number of bytes or words per record. Record size is important because certain applications can read only records of a certain size. You may find that if a file has records of a different size, you will not be able to display or edit that file.

This record size may vary according to the utility or program that created the file. For example, a file created with the editor will have a different record size than one created by `LISTFILE > filelist`. A positive record size indicates the number of two-byte *words* in the record. A negative record size indicates the number of bytes in the record.

Q2-5	What is the record size of MYFILE?

Blocking factor

This option is a holdover from a previous operating system, MPE V. At one time, the blocking factor was used to determine the number of logical records for each physically configured block. The more logical records per block, the more densely the data was packed. This helped to conserve space; however, blocking is rarely used on MPE/iX, so the blocking factor is usually allowed to default.

Note

When you do not build the file with the `BUILD` command, the blocking factor may vary according to the utility or program that created that file.

For files with variable-length records, the blocking factor always equals one.

Q2-6	What is the blocking factor for MYFILE?

Record type

Records are either fixed-length (length is a specific number of words or bytes), variable length (length changes with the data), or undefined. Generally speaking, editors can only read and work with fixed-length records.

Note

When you do not build the file with the `BUILD` command, the record type may vary according to the utility that created that file.

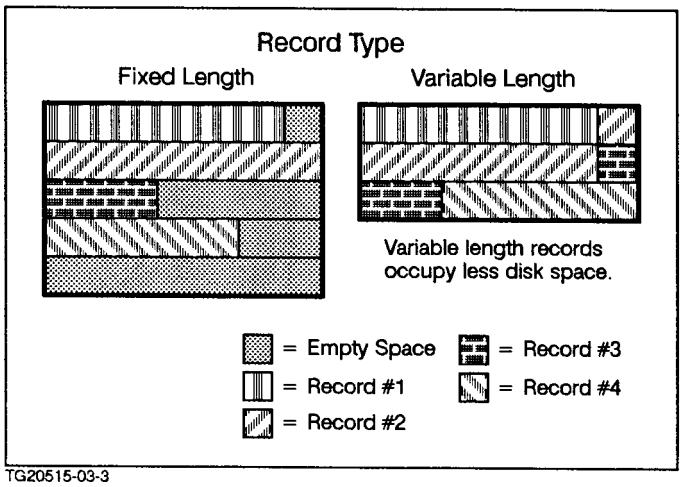


Figure 3-3. Record Type

Q2-7	What is the record type of MYFILE?

File data type

Text files that you can view and edit tend to be ASCII. Executable program files and graphics files tend to be binary. In the next lesson you will have an opportunity to create some binary files.

Q2-8	What is the file type of MYFILE?
Q2-9	Based on the information from `LISTFILE` and on your understanding of `BUILD` options, what was the original `BUILD` command that created MYFILE?

Other BUILD options

There are several other options that you may occasionally wish to specify in the `BUILD` command:

`TEMP`	User-defined temporary file
`CCTL/NOCCTL`	Carriage control or no carriage control characters
`STD/MSG/CIR`	Standard, message, or circular file
`KSAMXL/SPOOL`	KSAM or spool file
`DEV`	Device
`CODE`	File code identifying the specific type of file (for example, the editor file or executable code).

Exercise 2-2: creating a file to specification

1. Write the correct BUILD command to create to specification each of the files listed below. Whenever default values are specified, let the parameter default in the BUILD syntax.

Note

A list of codes appears in the help facility (at the system prompt, enter: HELP FILE FILECODE). The same information is available in appendix F of the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

   a. FILE1 - should have 64 words/record, have fixed-length records, be an ASCII file, and be circular.

   b. FILE2 - should have variable-length records whose maximum size is 256 words and be a binary file.

   c. FILE3 - should have variable-length records whose maximum size is 88 bytes and be an ASCII file.

2. Build the following file and give it a unique name. (Suggestion: use your first name of eight characters or less.) Assume following the values:

   record size = 80 bytes
   record type = fixed-length
   file type = ASCII

Note

Make sure that you are logged on as USERx.ACCTx in the CLASS group.

   Use LISTFILE to confirm that the file exists. Use PRINT to confirm that the file is empty.

   You will later have the opportunity to copy the contents of an existing file into this empty file.

********** End of Exercise 2-2 **********

Lesson summary

1. The BUILD command lets you create a file with specific characteristics.
2. The following BUILD options can be used to specify the file characteristics:
   a. record size (REC=)
   b. blocking factor
   c. record type (F,V,U)
   d. file type (ASCII,BINARY)
   e. device (DEV)
   f. file code

Lesson 3 Creating Temporary Files

Note

System programmers must read this lesson. If you do not plan to do any programming, you may skip this lesson.

Introduction

Lesson 3 presents the following concepts related to compiling and linking files:

- displaying names of temporary files
- creating temporary files during compiling and linking
- using system-defined temporary files to redirect compilation output

Some files, other than those that you create, are created by the system or subsystem temporarily during program compiling and linking. These files are referred to as temporary, system-defined files. In this lesson, you will learn about several of these system-defined files:

`$NEWPASS`	Temporary file created automatically during compiling to which newly generated compiled code is written.
`$OLDPASS`	Temporary file created automatically when compiling is complete. Used to hold compiled code.
`$NULL`	Temporary file that is empty when used as input and meaningless when used as output. (The output essentially disappears into what is referred to as the *bit bucket*.)

Each of these files is discussed in this lesson.

$NEWPASS and $OLDPASS

The system creates temporary files to hold data generated during program processing. By doing this, output from one process can serve as input to another process. For example, compiler output serves as linker input.

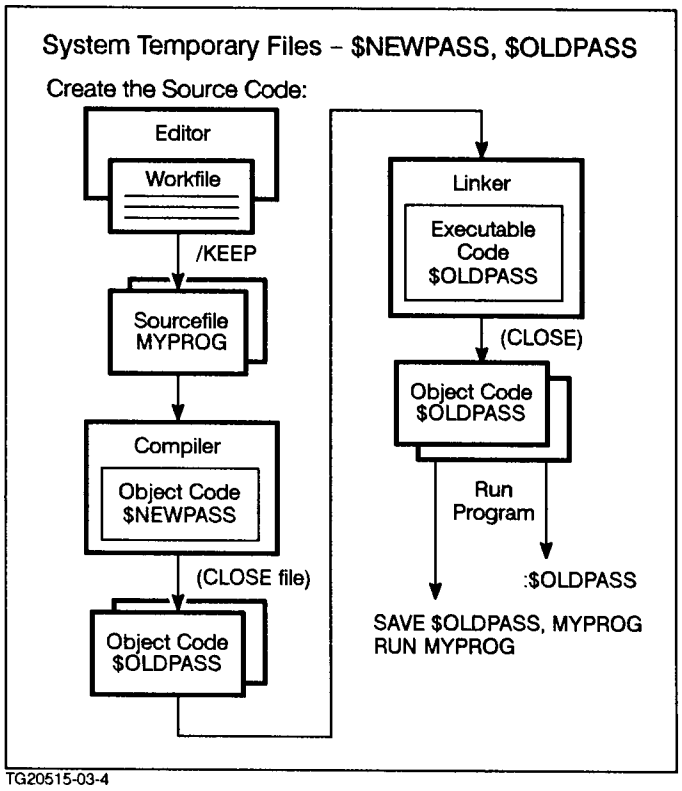


Figure 3-4. Temporary Files

When you compile and link a program, the following happens, as shown in figure 2-4:

1. Source code serves as input to the compiler.
2. The compiler builds `$NEWPASS` and writes compiled code to `$NEWPASS`.
3. When compiling is done, `$NEWPASS` is closed and renamed `$OLDPASS` (automatically). `$OLDPASS` contains the object code.
4. The linker uses `$OLDPASS` as input.
5. The linker builds `$NEWPASS` and writes linked, executable code to `$NEWPASS`.
6. When linking completes, `$NEWPASS` is closed and renamed `$OLDPASS` (automatically). `$OLDPASS` contains the executable code. `$OLDPASS` may be executed or saved with a new name by the user.

The `LISTFILE` command alone does not the names of these files. `LISTFILE` displays only permanent, disk files. To display the names of temporary files, such as `$OLDPASS`, you must use the `;TEMP` option of the `LISTFILE` command (`LISTFILE;TEMP`).

Here is an exercise that generates these temporary files.

Exercise 2-3: $OLDPASS

Follow the designated steps to generate $OLDPASS.

1. Make sure that you are logged on as USERx.ACCTx in the CLASS group.
2. Compile and link the C source code called HIC. You can list its contents with PRINT. The commands to compile and link are given below:

   ```
   CCXLLK source,program,listfile
   ```

   source: source file (HIC)

   program: compiled, linked code (default = $OLDPASS)

   listfile: errorlisting file (default = terminal screen)

   a. Allow the name of the executable program file (compiled, linked version) to default.

   b. Name the error listing file, HIERR.

3. To see if any temporary files were generated, enter:

   ```
   LISTFILE @.CLASS,2;TEMP
   ```

   a. Are any temporary files generated?

   b. What is their file type and file code? What does this mean? (A list of codes appears in the help facility (at the system prompt, enter: HELP FILE FILECODE). The same information is available in appendix F of the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364).

   c. What happens if you enter

      ```
      LISTFILE @.CLASS.ACCTx,2
      ```

      Why?

4. To save the $OLDPASS file as a permanent file, you must enter this command:

   ```
   SAVE $OLDPASS,filename
   ```

   a. Do so now to save $OLDPASS as a permanent file called HICP.

      ```
      SAVE $OLDPASS,HICP
      ```

   b. Use the LISTFILE command to verify that it is now a permanent disk file with the proper characteristics. Notice in particular that this is a binary file, ready for execution.

5. Use LISTFILE,3 to examine the characteristics of the HIERR file (especially the record size and the file type). Since HIERR is an ASCII file, you should be able to view it; however, the record size is too great for the editor. What might you do to remedy this?

********** End of Exercise 2-3 **********

When you are modifying, compiling, and linking several programs, over and over again, $OLDPASS always contains the latest version of whatever source code was processed. To prevent the possibility of

overwriting the current processed code, you must explicitly specify a program file name in the compile/link command line.

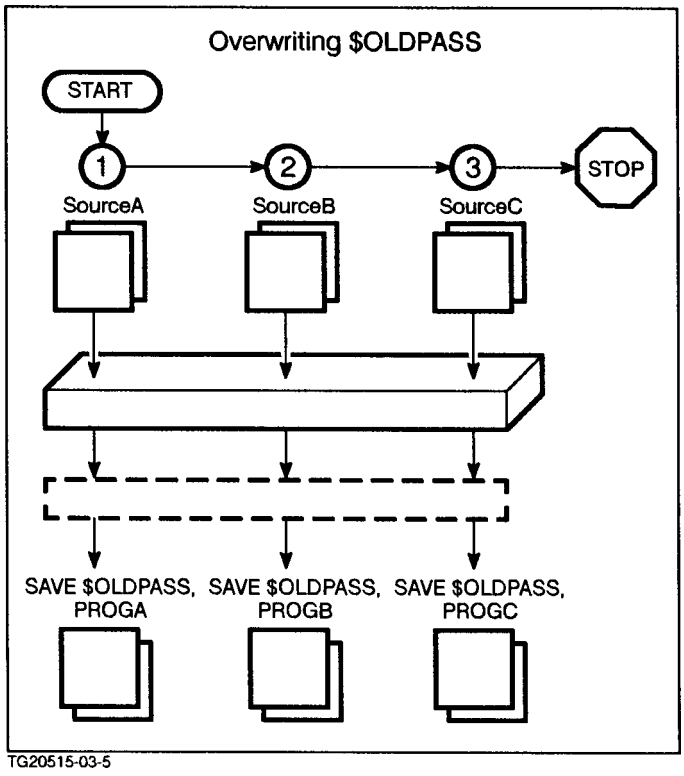


Figure 3-5. Overwriting `$OLDPASS`

Allow the name to default to `$OLDPASS` only if you are working with just one program. Make sure to save `$OLDPASS` under a different name when you finish, because temporary files only exist for the duration of the session!

$NULL If you wish your output to "disappear," `$NULL` is a handy file to use. Output that is written to `$NULL` goes nowhere ("bit bucket").

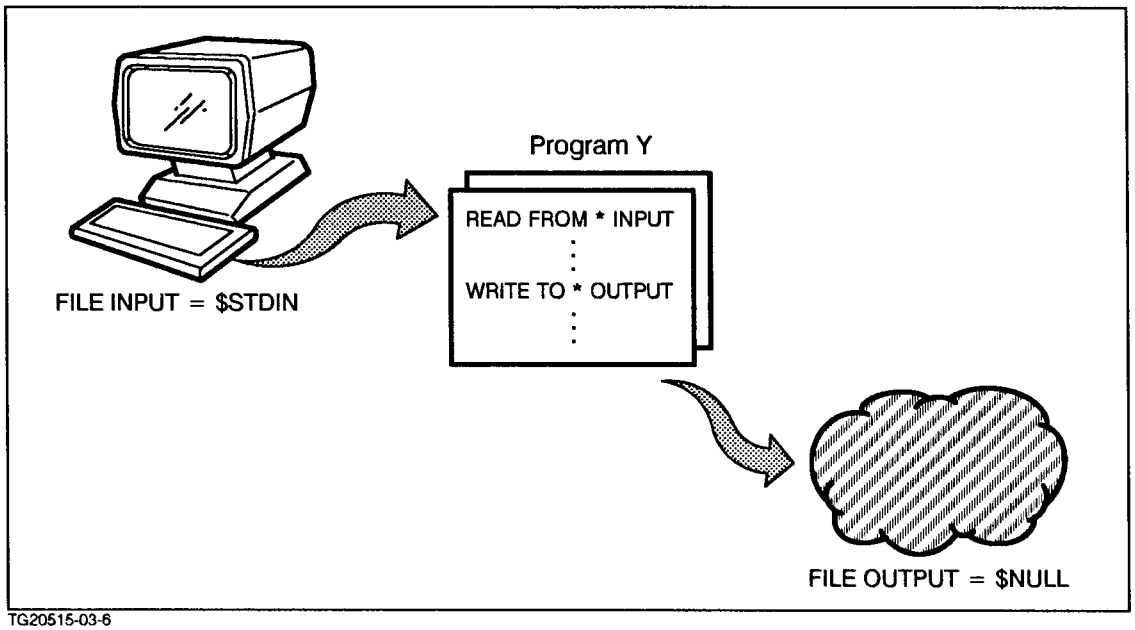


Figure 3-6. $NULL File

For example, suppose that you had a program that took user input, performed calculations, and wrote those calculations to the terminal and to a file. During the compiling/linking phase, you are interested only in whether or not the program is syntactically correct.

For testing purposes—when you run the program—you are interested only in whether or not it prompts correctly, and lists the results on the screen where you can examine them. You don't care whether the results are stored in a file or not. In this case, you can specify the output file to be `$NULL` and it will never show up on disk.

Note

For Programmers

When compiling a large program, specify `$NULL` as the list file. Compilation occurs much faster since no time is spent listing program lines on the screen.

Lesson summary

1. The `LISTFILE;TEMP` command displays the names of temporary files.
2. `$NEWPASS` and `$OLDPASS` are created during compiling and linking. Only `$OLDPASS` remains after the process.
3. `$NULL` can be used to direct program output to "disappear" during testing.
4. `SAVE` is used to save a temporary file as a permanent one.

Lesson 4 Using File Equations

Introduction

Lesson 4 presents the following file concepts:

- file equations
- device files
- formal file designators
- "backreferencing" file designators

You have worked with files that exist permanently or temporarily on disk. What about files that do not really exist, files that are not really files? These are *formal files*, files whose names are actually "aliases" for other files or devices.

Note

Before you read any further, make sure that you are logged on as USERx in the CLASS group of the ACCTx account (USERx.ACCTx,CLASS—where X is the student number assigned to you by your system manager or course manager).

Writing file equations

As you have already seen, if you do a LISTFILE or LISTFILE;TEMP, you see names of actual files. What you don't see are names that serve as aliases and "point" to other files or devices. These names are defined in *file equations*. Enter the following command to list the file equations currently available on your system.

```
LISTEQ
```

A file equation is something that equates a name (formal file designator) with a specific device or file. In other words, the formal file designator serves as an alias for an actual device or file.

One type of syntax equates a file designator with another system, subsystem, or user defined file:

```
FILE filedesignator=sysfilename;
REC=recsize,blkfactor,rectype,filedatatype
```

Example:

```
FILE IN=$STDIN
FILE OUTPUT=TESTFILE;REC=-80,,F,ASCII
```

Another type of syntax equates a file designator with a device:

```
FILE filedesignator;DEV=device;ENV=environment_file
```

Example:

```
FILE MAILPRNT;DEV=LP;ENV=ELITE.HPENV.SYS
```

Note An environment file (ENV) is a special file that contains basic information about page layout and type style.

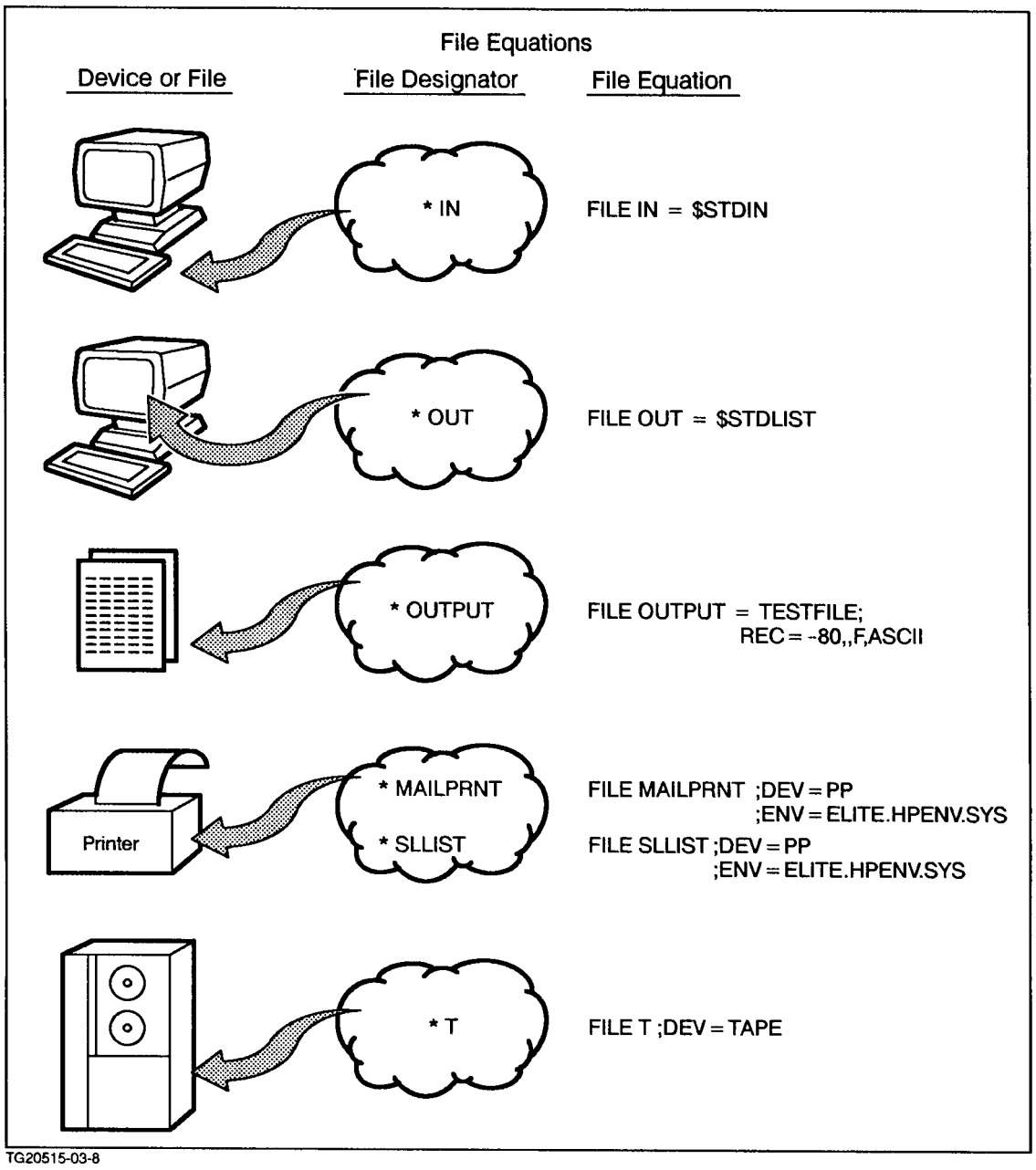


Figure 3-7. File Equations

Here are some examples of file equations that you might see on your system. All file equations begin with the keyword, `FILE`:

a. `FILE IN=$STDIN`

b. `FILE OUT=$STDLIST`

c. `FILE OUTPUT=TESTFILE;REC=-80,,F,ASCII`

d. `FILE MAILPRNT;DEV=LP;ENV=ELITE.HPENV.SYS`

e. `FILE T;DEV=TAPE`

Q2-10	According to the syntax, in the first file equation in the example (a), `IN` is actually an alias for `$STDIN`. What is `$STDIN`?
Q2-11	In the `OUT` equation (b), `OUT` is actually the alias for what?
Q2-12	The `MAILPRNT` equation (d) is used to print HP Desk messages. What device prints these messages?

Benefits of the FILE command

Why should you learn about the `FILE` command and use file equations? For two reasons:

- File equations let you define device files that treat a peripheral device (line printer, laser printer, or tape drive) as though it were a file.
- File equations also let you define formal files that reference a temporary or permanent file by a generic name, even though the actual file name may change. Formal files also give you a short name to use when referencing a lengthy file description.

Backreferencing

The process of referring to files and devices by a file designator is called *backreferencing*. The file designator always has an asterisk (*) in front of it for backreferencing.

For example, suppose that you define a file equation with a file designator called OUTPUT. This designator points to a file where results from a program are stored. You then write a program that executes a number of calculations, stores the results in a file, and prints the contents of that file.

In such a situation, you might enter this:

```
FILE OUTPUT=FILE1
```

The program would contain a code that would do the following:

write results to `*OUTPUT`

```
PRINT *OUTPUT
```

As a result, the program stores the results in FILE1 and prints the contents of FILE1, since FILE1 is pointed to by *OUTPUT. The program would *not* print the contents of a file called OUTPUT. The asterisk (*) in front of OUTPUT indicates that *OUTPUT is a file designator that refers to a file or device defined in a file equation.

You can demonstrate the above scenario yourself by entering the following commands. (You are assumed to be in the `CLASS` group.)

```
FILE OUTPUT=MYJOB1
```

```
PRINT *OUTPUT
```

You should see the contents of MYJOB1. If you change the file equation and reexecute the PRINT command, you see the contents of another file:

```
FILE OUTPUT=JOB2
PRINT *OUTPUT
```

The file designator, *OUTPUT, does not change, even if the file equation does. In a program, you might wish to change the file equation to redirect output to different files or devices. For example, suppose the first time you run the program, you want the output to go to the screen. Then, the second time you run the program, you want the output to go to a file called TEMP1. Finally, the third time you run the program, you want the output to go to a file called TEMP2. Rather than editing the program each time, and changing the actual file name, you can leave the program as is and refer to *OUTPUT.

Each time that you run the program, you need only enter a new FILE command to redefine the file equation. File equations last for the duration of the session, unless they are stored in a special file that goes into effect each time you log on. (You will learn about this in the UDC module.)

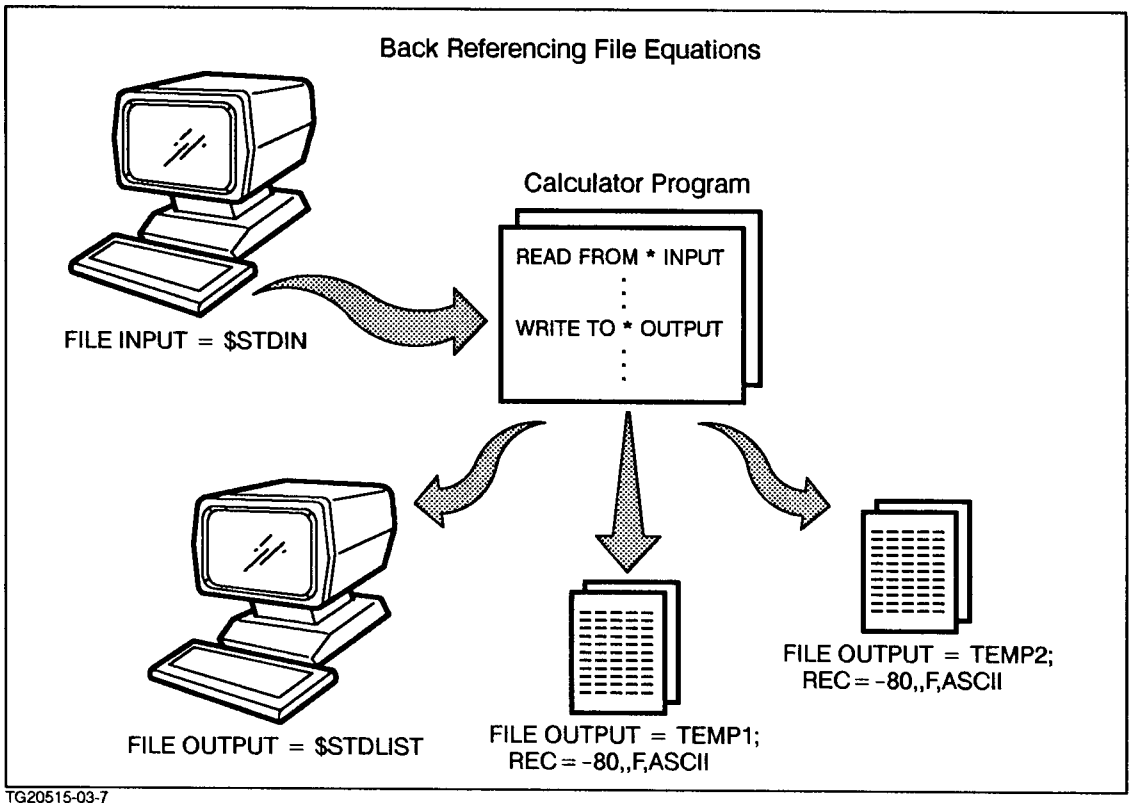


Figure 3-8. Backreferencing File Equations

Teaching exercise 2-4: redirecting output with file equations

This exercise provides hands-on experience redirecting output with file equations.

1. Create a file equation for a file designator called OUTPUT. Enter this file equation:

   ```
   FILE OUTPUT=$STDLIST
   ```

   This should direct the output to the standard output device, your terminal screen, whenever you refer to *OUTPUT.

2. Print the contents of the JOB1 file by using the PRINT command with its OUT option to direct the contents to *OUTPUT:

   ```
   PRINT JOB1;OUT=*OUTPUT
   ```

   Where does the output appear? Right. On your terminal screen.

3. Now change the file equation so that the output is redirected to a file called TEMP1.

   ```
   FILE OUTPUT=TEMP1
   ```

4. Enter the PRINT command again, using the *OUTPUT designator once more:

   ```
   PRINT JOB1;OUT=*OUTPUT
   ```

   Does the formatted output appear on your screen? It shouldn't. Instead, it should appear in the TEMP1 file. Use the PRINT command (without any options) to view this file:

   ```
   PRINT TEMP1
   ```

5. Lastly, change the file equation again so that the output is redirected to another file, TEMP2. Enter the PRINT command once more, using the same *OUTPUT designator:

   ```
   PRINT JOB1;OUT=*OUTPUT
   ```

6. The output from the PRINT command is directed to *OUTPUT, which actually refers to TEMP2, the file specified in the file equation.

   Take a look at TEMP2 by using the PRINT command again:

   ```
   PRINT TEMP2
   ```

   You never changed the `PRINT JOB1;OUT=*OUTPUT` command; yet, you were able to send the output of the PRINT command to different destinations. All that you changed was the file equation that defined OUTPUT. The formatted output was redirected each time you changed the equation.

********** End of Exercise 2-4 **********

System-Defined designators

When you create file equations, you can use system-defined file designators, subsystem-defined file designators, and user-defined file designators.

The system defines its own set of file designators, some of which were introduced in lesson 3 for programmers:

```
$STDIN
$STDLIST
$NULL
$OLDPASS
$NEWPASS
```

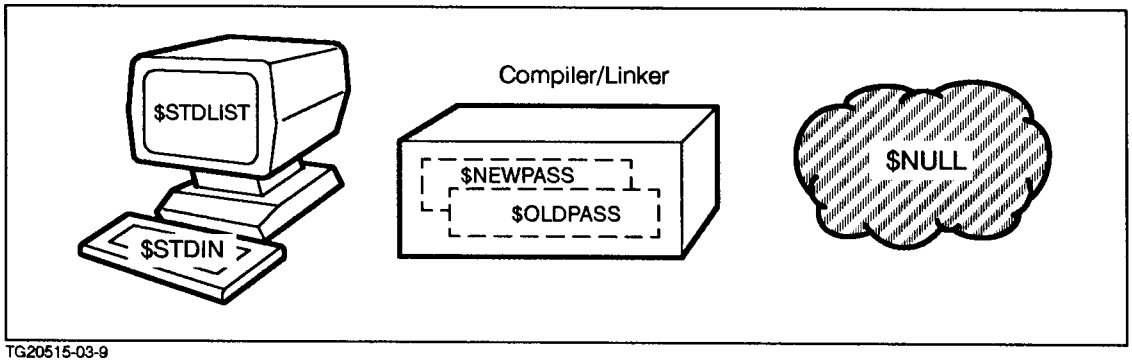


Figure 3-9. System-Defined File Designators

$OLDPASS and $NEWPASS are the system-defined temporary files generated during program compiling and linking.

$NULL, $STDIN, and $STDLIST are the system-defined temporary files that are useful in file equations for redirecting input and output.

System files cannot be equated to devices; they can only be equated to other file designators.

Example #1a: Invalid File Equation

```
FILE OUTPUT;DEV=$STDLIST
PRINT MYFILE1;OUT=*OUTPUT
```

Example #1b: Valid File Equation

```
FILE OUTPUT=$STDLIST
PRINT MYFILE1;OUT=*OUTPUT
```

You can equate a file designator to a device only with the DEV option of a file equation:

Example #2a: Invalid File Equation

```
FILE OUTPUT=LP
PRINT MYFILE1;OUT=*OUTPUT
```

Example #2b: Valid File Equation

```
FILE OUTPUT;DEV=LP
PRINT MYFILE;OUT=*OUTPUT
```

$STDIN refers to the standard input device, which is, by default, the keyboard.

$STDIN is a temporary system file, lasting only for the duration of the session. When writing programs, you can specify that the data be read from the standard input device as follows:

```
FILE TERM=$STDIN
   .
   .
   .
read from *TERM
```

$STDLIST refers to the standard output device, which is by default the terminal screen. The output device can also be the line printer. When writing programs, you can specify that the data be written to the standard output device as follows:

```
FILE OUTLIST=$STDLIST
  .
  .
  .
  write to *OUTLIST
```

$NULL is the system-defined file treated as an empty file ("bit bucket"). In the following example, one program reads from *INPUT and another program writes to *OUTPUT. Both INPUT and OUTPUT are file designators for $NULL. This means that one program reads from an empty file and receives an end-of-file marker.

The other program writes to *OUTPUT, and no physical output is actually produced (it writes to no device or file).

```
FILE INPUT=$NULL
.
.
.
read from *INPUT

FILE OUTPUT=$NULL
.
.
.
write to *OUTPUT
```

To illustrate this concept, use the FILE command to define a file designator called ERROR that refers to $NULL:

```
FILE ERROR=$NULL
```

Now compile and link the HIC program as shown below, and specify *ERROR as the error listing file. Normally the error listing would appear on the screen. Only this time, because the error listing has been written to *ERROR ($NULL), it has disappeared:

```
CCXLLK HIC,HICP,*ERROR
```

The resulting executable program is HICP. The error listing (*ERROR) does not exist because it was written to $NULL ("bit bucket").

User-defined files

You can create your own file equations using a file designator of your choice and an existing device or file of your choice. For example, when you run HP Desk (Hewlett-Packard's electronic mail facility) and enter PRINT, HP Desk assumes that you wish your mail messages printed on MAILPRNT (line printer using special environment file). This is because the following file equation already exists:

```
FILE MAILPRNT;DEV=LP
```

Q2-13 How might you change the file equation to redirect HP Desk messages to the terminal screen? How might you change the file equation to redirect HP Desk messages to the laser printer (designated as LP for this question)?

You could further edit the equation so that the mail messages print in a particular type style. Do you remember where type style information is stored? Yes, it is stored in the environment file. So, you must change the ENV part of the file equation. Of course you would need that environment file on your system to print the messages in that type style.

```
FILE MAILPRNT;DEV=LP;ENV=ELITE.PUB.SYS
```

Now, suppose that you also wish to run a program that reads from and writes to different files; however, you wish to refer to those files, generically, as FILEIN and FILEOUT within the program:

File Equation #1:

```
FILE FILEIN=MYFILEA;REC=-80,,F,ASCII
```

File Equation #2:

```
FILE FILEOUT=MYFILEB;REC=-80,,F,ASCII
```

Q2-14 According to the first file equation, what is the name of the actual file associated with FILEIN? What are its characteristics?

```
File name:
Record size:
Blocking factor:
Record type:
File type:
```

Q2-15 According to the second file equation, what is the name of the actual file associated with FILEOUT? What are its characteristics?

```
File name:
Record size:
Blocking factor:
Record type:
File type:
```

Clearing file equations

If you no longer want a file equation to be in effect, you can use the RESET command to cancel it. The syntax for this command is very simple:

```
RESET filedesignator
```

Q2-16	If you no longer want the FILEIN or FILEOUT equations to be in effect, what do you do?

Be Careful

Do not enter RESET @ (RETURN) unless you want to cancel *all* of the file equations in the current session or job.

Exercise 2-5: creating and using file equations

1. Make sure that you are logged on as USERx.ACCTx, CLASS.
2. Enter the command to list all file equations:

   You may see these equations:

   ```
   FILE IN=$STDIN
   FILE OUT=$STDLIST
   FILE OUTPUT=TEMP2
   FILE MAILPRNT;DEV=LP;ENV=ELITE.HPENV.SYS
   FILE ERROR=$NULL
   ```

3. Assume that a program reads from *IN and writes to *OUT. Modify each file equation as follows:

   a. Change the following OUT file equation so that output goes to the terminal instead of to the FILE2 file (as shown below):

      ```
      FILE OUT=FILE2;REC=-80,,F,ASCII
      ```

   b. Change the following file IN equation so that input is read from the FILE1 file, instead of from the terminal (as shown below):

      ```
      FILE IN=$STDIN
      ```

4. Identify each piece of the file equations (assume that LP is a line printer and that PP is a laser printer):

   a. FILE OUTPUT;DEV=LP

      What is the name of the file designator?

      What device does it refer to?

   b. FILE FORMAT;DEV=LP;ENV=PICA.HPENV.SYS

      What is the name of the file designator?

      What device does it refer to?

What is the environment file?

What does this file equation do?

c. `FILE INPUT=MYFILE1;REC=-80,,F,ASCII`

What is the name of the file designator?

What file does it refer to?

What happens if a program reads from *INPUT?

What size records are in the actual file?

d. `FILE OUTPUT=$NULL`

What is the name of the file designator?

What device or file does *OUTPUT refer to?

What happens if you write to *OUTPUT?

5. According to the following file equations, what happens when the specified commands are entered?

a. `FILE MAILPRNT;DEV=LP`
`HPMAIL`
`PRINT 10` *(this is the mail item number)*

b. `FILE OUTPUT=$STDLIST`
`PRINT MYFILE1;OUT=*OUTPUT`

c. `FILE INPUT=$STDIN`
`read FROM *INPUT`

d. `FILE INPUT=MYFILE`
`read FROM *INPUT`

e. `FILE OUTPUT=MYFILE;REC=-80,,F,ASCII`
`write to *OUTPUT`

6. *For programmers only*: Practice redirecting the error listing output when compiling/linking the HIC program.

Note

Throughout this exercise, you should never have to alter the compile/link command line, just the file equation.

a. Create a file equation that defines HIERR as the terminal screen.

b. Enter the following compile/link command line for the C program, HIC. Refer to the formal file designator for the error listing.

`CCXLLK HIC,,*HIERR`

Where does the listing print?

c. Change the file equation so that the error listing will be stored in a file called ERROR1 with default file characteristics.

```
FILE HIERR=ERROR1
```

d. Compile and link the program again. Where does the error listing print?

e. Change the file equation one more time to store the error listing in the ERROR2 file and compile and link the program. Where does the error listing print this time?

Lesson summary

1. System-defined files that can appear in file equations include the following:
   a. $STDIN
   b. $STDLIST
   c. $NULL
2. When using a system-defined file in a file equation, the format must be the following:

```
FILE filedesignator = system-defined file
```

3. When using a user-defined file in a file equation, the format must be the following:

```
FILE filedesignator= user-defined file
[;REC=.........]
```

4. When using a device in a file equation, the format must be the following:

```
FILE filedesignator;DEV=device
```

4

Module 3: Batch Processing

Module 3 presents the following lessons on using job files:

Lesson 1	Introducing Jobs
Lesson 2	Examining a Job File
Lesson 3	Creating and Streaming a Job File
Lesson 4	Monitoring Job Progress

Challenge Test

1. Which of the following characteristics refer to batch processing?
   a. interactive processing
   b. submits multiple commands at one time
   c. can be scheduled to run at specific times
   d. requires user supervision
2. What are the five job processing states?

3. Match the following terms with the descriptions below: input priority, job limit, jobfence, and session limit.
   a. establishes the priority for jobs to execute
   b. is the maximum number of sessions that can execute simultaneously
   c. must be above the jobfence for a job to run
   d. is the maximum number of jobs that can execute simultaneously
4. Write a short job file that prints `MYFILE1` on the line printer (assume that a file equation already exists defining LP as the line printer).

5. Write the `STREAM` commands that does the following:
   a. `JOBFILE` runs immediately.
   b. `JOBFILE` will run at 11:00 P.M. tonight.
   c. `JOBFILE` will run next Monday at 8:00 A.M.
   d. `JOBFILE` will run next Wednesday at 10:00 P.M.
6. What are the five commands that control job execution?
7. What are the ten states of a spool file?

Match the following output spool file states to the correct activity of the spool file:

`READY` Spool file is being printed.

`PRINT` Being created as output.

`CREATE` Spool file is ready to print.

8. What are the three most useful commands used to monitor job progress, and examine or delete completed spool files?

Lesson 1
Introducing Jobs

Introduction

Lesson 1 presents the following information about jobs and batch processing:

- comparison of session to job processing
- descriptions of `SHOWJOB` display information
- effect of the job limit and jobfence on batch processing

Until now your logon time in this course has been in session mode. In session mode, the computer acts on a single command at a time. However, in job mode or batch processing, you use the `STREAM` command to submit an entire series of commands as a "batch" or "job" file to the computer.

```
STREAM MYJOB
```

`MYJOB` is a job file containing a "batch" of commands. The computer processes these commands without additional input from you. Your results generally are sent to a printer.

Note

`MYJOB` and some other job files are ready for you to use with this module. Your system manager put them there for your use.

Other job files on your system are incomplete. You will have a chance to complete and run them as part of your study.

Advantages of job/batch processing

There are a number of advantages to job processing.

It's much more efficient than session processing. Jobs generally do not require continuous input or your supervision. You can process a number of job commands with a single command. While your job is processing, your terminal is free for other work.

One other advantage of job processing is that the processing can be scheduled to occur at a specific time. A job that requires a large portion of the system's resources may be run at a time when the system is not busy with interactive sessions or other jobs.

Comparison to sessions

How well can you distinguish between jobs and sessions? Label each item below to show which refer to jobs and which refer to sessions.

Q3-1 Which are characteristics of jobs? Of sessions?

a. interactive processing

b. synonymous with batch processing

c. does not require constant supervision

d. submits a single command at a time

e. can be scheduled to run at specified time

f. most time/resource efficient

g. executes with STREAM command

More SHOWJOB information

You already know that the SHOWJOB command lists all jobs and sessions currently on your system and provides you with the limits for each; however, as you begin working with jobs, you will be interested in the additional information displayed by SHOWJOB.

The following is an example of a SHOWJOB display:

```
JOBNUM    STATE  IPRI   JIN    JLIST  INTRODUCED    JOB NAME

#S201     EXEC          20      20    MON 7:05A     CONSOLE.SYS
#J30      WAIT   D6     10S     LP    MON 6:30A     JON.FINANCE
#S230     EXEC          55      55    MON 12:02P    USER2.CLASS
#J35      INTRO  8      10S     LP    MON 5:45A     STATS.PROJECT
#J32      EXEC          10S     LP    MON 4:30A     MARY.PROD

4 JOBS:

   1 INTRO
   1 WAIT
   3 EXEC; INCL 2 SESSIONS
   0 SUSP
  JOBFENCE=6; JLIMIT=11; SLIMIT=20

CURRENT:  7/20/89  17:30

JOBNUM    STATE  IPRI   JIN    JLIST  SCHEDULED-INTRO  JOB NAME

#J33      SCHED  9      10S    LP     7/20/89   20:00  MARY.PROJA
#J34      SCHED  8      10S    LP     7/20/89   22:00  USER2.CLASS

2 SCHEDULED JOB(S)
```

The important terms in the display are described below:

JOBNUM — Lists the number assigned by the system to sessions and jobs. Numbers assigned are sequential. J indicates a job; S indicates a session.

STATE — Shows the session or job's processing state.

The five possibilities for jobs are:

INTRO — The job is being submitted.

WAIT	The job is waiting for system resources, or the job limit may have been reached, or the job's priority is too low.
EXEC	The job is executing.
SUSP	The job was executing but is now suspended.
SCHED	The job is scheduled to execute at a time specified by the user.

Note

Questions 2 through 12 refer to the SHOWJOB display.

Q3-2	How many jobs in the preceding display are executing?
Q3-3	Which of the jobs in the display is being submitted?
Q3-4	Which job is awaiting execution?

JOBFENCE, JLIMIT, SLIMIT information

JOBFENCE	Establishes the priority for jobs to execute. A job must have a priority above the jobfence in order for the job to execute.
JLIMIT	Indicates the maximum number of jobs that can be executing simultaneously.
SLIMIT	Indicates the maximum number of sessions that can be logged on at any one time.

The values for each of these are set by system management and may be adjusted by them to meet the needs of system users.

IPRI (input priority)

This column displays a job's input priority only if that job is in the WAIT or INTRO state, or has been scheduled to run at a later time. D6 means the job has an input priority of 6, but is deferred for later execution. If a job is currently executing or suspended, the job's priority value is not displayed. You might see QUIET in this column. The QUIET means that the user has instructed the system *not* to interrupt with messages from other users. (The console operator can still interrupt a Quiet session with a WARN command message.)

Input priority values range from 1 (low priority) to 13 (high priority). If you enter HIPRI in the !JOB command line, the priority will be set to 15, but only if you have SM or OP capability; otherwise, the job will be submitted with the highest possible priority available at the time.

Jobs in the EXEC or SUSPEND state do not have IRPR values.

In order for a job to begin executing, the input priority value, IPRI, must be greater than the current jobfence. Once the job has entered the EXEC state, INPRI and the jobfence value have no further meaning for the job.

Q3-5	If the current jobfence is 7, which of the following jobs will execute? a. JOBA which has an input priority of 7 b. JOBB which has an input priority of 8 c. JOBC which has an input priority of 6
Q3-6	If the current JLIMIT is 5 and five jobs are executing, what will be the STATE of the next nonscheduled job submitted?

JIN and JLIST columns

These two columns show the device (LDEV) numbers assigned to input (JIN) devices and output (JLIST) devices. Since sessions are generally input and output from a terminal, these numbers are identical for sessions.

Because jobs are usually input from disk, and output is printed on the line printer (LP), the JIN and JLIST designators for jobs are different.

INTRODUCED and JOBNAME columns

The INTRODUCED column shows the date and time the session or job was started. For a job in the WAIT state, the date and time show when the job was introduced with the STREAM command. As soon as the job enters the EXEC state, the date and time show when the job entered the EXEC state.

The JOBNAME column shows the user and account where the job originated, preceded by the job name, if any.

If a job is scheduled for later submission, its state will be SCHED. It will appear at the end of the SHOWJOB listing. Under the column SCHEDULED-INTRO will be the date and time when the job is scheduled for submission.

Q3-7	At what time was Job 35 streamed?
Q3-8	From what account and by which user was Job 35 submitted?

Job and session summary

Just beneath the column information is a summary of information regarding jobs and sessions currently executing or awaiting execution on the system. The following information on the jobfence, job limit, and session limit is a list of scheduled jobs with the times they are scheduled to run.

Q3-9	How many jobs and sessions can be run concurrently on this system?
Q3-10	According to the JLIMIT information provided, how many more jobs could be executing at this time?
Q3-11	How many jobs have been suspended?
Q3-12	Which job will be the last to run on 7/20/89?

SHOWJOB parameters

Suppose that you want information about just one job.

```
SHOWJOB Jobnumber
```

Or perhaps you want to display only job processing information rather than job and session information to your screen.

```
SHOWJOB JOB=@J
```

The SHOWJOB command has a number of parameters that allow you to specify the kind of information to be displayed.

Q3-13 Review the parameter information for SHOWJOB in the Help facility. Then enter the SHOWJOB commands that do the following:

a. list job summary information

b. list jobs that have been scheduled

c. list information on all current jobs

d. list information on job 105

e. list the status of all jobs that you might have submitted from your account

Lesson summary

1. The SHOWJOB command lists all jobs and sessions currently executing or scheduled on your system.
2. In order for a job to execute, it must have an input priority above the jobfence.
3. Jobs submitted after the job limit has been reached will be assigned to a WAIT state until system resources are available for processing.
4. Unlike sessions, jobs allow you to submit a series of commands for processing, can be scheduled to run at specific times, and do not require constant user supervision.

Exercise 3-1:
lesson 1 review

Indicate which of the following statements are true.

1. If the jobfence is 6, jobs must have an input priority of at least 7 in order to execute.
2. When the job limit is reached, the system will not allow you stream any more jobs.
3. The job limit is the maximum number of jobs that can be executing at any one time by the system.
4. Scheduled jobs must also have an input priority above the jobfence in order to execute.
5. The job limit, session limit, and jobfence values may be set and adjusted by any user.

********** End of Exercise 3-1 **********

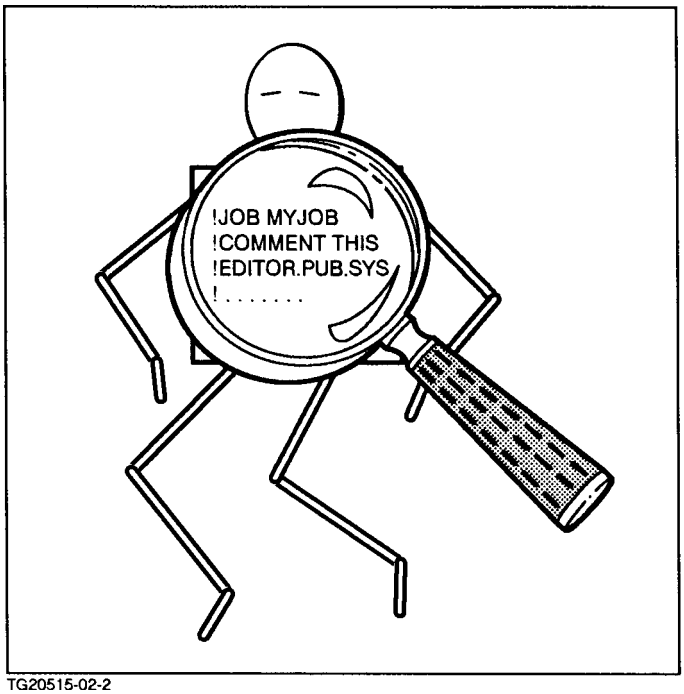
Lesson 2 Examining a Job File

Introduction

Lesson 2 provides the following information about the structure and use of a job file:

- job commands and how they are used
- input and output processing and printing priorities
- Modifying a job file.

In this lesson, you will examine a sample job file to see how job files are organized and what kinds of commands are used within these files.



TG20515-02-2

Figure 4-1. Examining a Job File

Start the editor and list the MYJOB (job) file to your screen. (MYJOB is in the CLASS group of your account.) This file contains examples of commands used in this lesson. The editor commands and job listing appear below.

```
EDITOR
/T MYJOB1
/L ALL
```

MYJOB listing:

```
1    !JOB MYJOB1,USERx/UPASSx.ACCTx/APASSx,CLASS;&
2    !INPRI=9;RESTART;OUTCLASS=LP,1;SPSAVE
3    !COMMENT MYJOB1 PRINTS MYFILE1
```

```
4    !CONTINUE
5    !EDITOR
6    T MYFILE1
7    L ALL,OFFLINE
8    EXIT
9    !TELL USERx.ACCTx  MYJOB1 IS DONE!!
10   !EOJ
```

Job files are generally created by using an editor or a word processor and may consist of four types of command:

- MPE/iX batch processing commands
- other MPE/iX commands
- user-defined commands (UDC) and command files (module 5)
- application commands (optional)

You will learn about user-defined commands and command files in module 5, "UDCs and Command Files."

Batch processing commands

These five MPE/iX commands are used almost exclusively in job/batch files are:

Command	Line Number in MYJOB File
JOB	1
COMMENT	3
CONTINUE	4
TELL	9
EOJ	10

When MPE/iX commands appear in a job file, they must be preceded by an exclamation point (!). The ! tells the system that what follows should be treated as an MPE/iX command.

Each of these commands is described and explained in both the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) and the Help facility. Select either resource for more information on the purpose and parameters of each of the commands as they are explained in this lesson.

JOB command

```
!JOB MYJOB1,USERx/UPASSx.ACCTx/APASSx,&
!CLASS;INPRI=9;RESTART;OUTCLASS=LP,1;SPSAVE
```

Note

When a JOB command is too long to be written on one physical line on the screen, it may be continued onto succeeding lines. To show the continuation, an ampersand (&) is entered at the end of the each physical line. Each line of the continued command *must* begin with an exclamation point (!). A command continued over several lines with & may consist of as many as 279 characters, including spaces and ampersands, but not including the ! in the column one position.

Every job file requires the inclusion of a `!JOB` command.

The purpose of the `JOB` command is to initiate a batch job. It is always the first executable line of any job and contains additional information regarding the location of the file materials, user and account names, and so on. (The `JOB` command is to batch processing what the `HELLO` command is to interactive processing.)

By using the required and optional parameters of the `JOB` command, you tell the system that the commands that follow should be executed as a job. Which of the `JOB` command parameters are required? Which are optional? Use your reference source and mark this information in the following table.

Q3-14

JOB Parameter	Required	Optional
User Name		
Account Name		
Group Name		
Job Name		
Time		
`INPRI`		
`RESTART`		
`OUTCLASS`		
`SPSAVE`		

Hint

`HELP JOB`

User/account names/passwords

```
!JOB MYJOB1,USERx/UPASSx.ACCTx/APASSx,CLASS;
```

Most `JOB` commands begin with the name of the job followed by the user name, account name, and logon group name. While not required, including the name of a job file in a `JOB` statement can provide for easy identification of the file.

Unlike sessions, where MPE/iX prompts for passwords, job files must contain the required user, account, and group passwords to work. Therefore, the passwords (if any) for the user, account, and group must follow each of these names. Notice that the passwords must be joined to the user, account, or group name by a backslash (/): `!JOB ... USERx/UPASSx.ACCTx/APASSx....`

Q3-15 What are the job name, user name, account name, and group name specified in the `JOB` command in the `MYJOB` file?

Q3-16 If the password for the `CLASS` group is LASER, will the job run with the `JOB` command stated as is?

Input priority

```
!JOB MYJOB1 ... ;INPRI=9 ...
```

Remember the importance of the input priority?

The input priority (INPRI) must be greater than the current jobfence in order for a job to execute.

What is the input priority for MYJOB?

It is 9. If you were going to run MYJOB now on your system, would it run? (Use SHOWJOB to check the jobfence on your system to find out.)

RESTART parameter

```
!JOB MYJOB1,USERx/UPASSx ... ;RESTART; ...
```

The RESTART parameter ensures that a job will restart in the event of a system halt and start. A job with this parameter restarts processing from the beginning; however, no work already done will be undone.

Note

RECOVERY Option

Your system operator may choose to restart a system with either the RECOVERY or the NORECOVERY option. RECOVERY is the default option, and it ensures that all jobs that include the RESTART option will be restarted. It also ensures the starting of any job that had *not yet* started at the time of the system halt.

No jobs of any kind are restarted if the system is brought back up with the NORECOVERY option.

Q3-17	What happens to your job if your system crashes and you do not have the RESTART parameter in your JOB command?

OUTCLASS parameter

```
!JOB MYJOB1 ... ;OUTCLASS=LP,1
```

The OUTCLASS parameter refers to a "job listing," not to the actual finished result you expect from a job. This parameter lets you set the following:

- device where the job listing is to be printed.
- job listing's outclass priority.
- number of copies of the job listing to be printed.

Outclass priority values range from 1 (low) to 13 (high). The output priority value must be set above the system outfence value in order to enable the job listing to print.

To check on the current outfence value for your system enter:

```
SHOWOUT
```

In order for a job listing to print, the outclass priority must be set above the system's outfence value.

Note

The job listing is a copy of the step-by-step processing of the job. Most users suppress the printing of the job listing unless the job has previously experienced some problems executing. When problems occur, review the job listing to find where the error(s) occurred.

SPSAVE parameter

A $STDLIST spool file is generated when you run a job. It is usually destined for the printer when the job has executed.

Normally, the spool file is automatically erased after it is printed. The optional `;SPSAVE` parameter instructs the system to save the $STDLIST spool file until you or the system manager explicitly purge it.

Hint:

A low output priority may prevent a spool file from printing, and the spool file will *appear* to be saved; but the spool file will disappear after it is allowed to print. The `;SPSAVE` option, however, saves the spool file after printing.

Saving a spool file can be useful for a job that is likely to execute quickly. By saving the spool file, you ensure yourself the opportunity of examining it with the `PRINT` command.

`MYJOB` is a small job and, if the system is not very busy, will execute quickly.

Q3-18 Assuming that the system outfence value is set at 6, which `!JOB` command ensures that the job listing will print?

a. `!JOBx.... .... INPRI=9;OUTCLASS=LP,5`

b. `!JOBx.... .... INPRI=9;OUTCLASS=LP,6`

c. `!JOBx.... .... INPRI=9;OUTCLASS=LP,8`

Q3-19 Where will the job listing from `MYJOB` appear?
Given a system outfence value of 4, will the job listing for `MYJOB` print?

COMMENT command

```
!COMMENT  MYJOB1 PRINTS MYFILE1
```

The `COMMENT` command inserts a comment into the command stream. The purpose of the comment is for the benefit of the user, since no execution is done. Comments may appear anywhere in a job file.

Including a comment regarding the purpose of a job can be helpful when you have multiple users of a single job.

CONTINUE command

```
!CONTINUE
```

How do you avoid a job aborting when it encounters an error?

The CONTINUE command enables a job to continue to process if it appears immediately before a command line that may cause an error during a job's processing; therefore, the job continues to execute even if an error occurs, and the job listing contains the error message where the error occurred.

Look at line 4 in MYJOB. Inserting the CONTINUE command on this line ensures that if the system encounters a problem with line 5, the job will not abort there.

TELL command and TELLOP command

```
!TELL USERx.ACCTx  MYJOB1 IS DONE!!
```

The TELL command can be inserted at one or more points in the job to send reports to the screen indicating your job's progress. The TELLOP command is used to alert system operators to perform some action regarding the job such as loading a tape or changing paper on the printer.

Q3-20	What prints on USERx's screen when MYJOB completes processing?

EOJ command

```
!EOJ
```

The !EOJ command terminates a job and displays the following information on the job's printout: CPU time (in seconds), time elapsed since the beginning of the job, and the date and time.

The !EOJ command is on the last line in a job file.

Other commands in job files

Some jobs may involve the use of a word processor or other kind of application or utility, for example EDIT/3000 or SORT. Any application that normally executes from a system prompt can be included in a job file preceded by an exclamation point:

```
!EDITOR
!SORT
```

Notice that after the editor is invoked (!EDITOR), the EDIT/3000 commands are entered in the job file exactly the way you would enter them if you were using the editor interactively at your terminal: T MYFILE1 and LIST ALL, OFFLINE. These commands are internal to the application (the editor) and must not have the ! ahead of them

Q3-21	What is the purpose of the !EDITOR line in MYJOB?
Q3-22	What is the purpose of the T MYFILE1 line in MYJOB?

Exercise 3-2: modifying the MYJOB2 file

Here's a chance to check your own knowledge about batch processing commands and their use.

Suppose that you have been given the following jobfile (MYJOB2). This job lists the contents of a text file titled MYFILE2; however, the job file needs to be modified in order to run correctly.

Use the editor to access and list MYJOB2 on your screen. (This file should be in your account.)

```
EDITOR
/T MYJOB2
```

MYJOB2 File

```
1  !JOB MYJOB2,USERx.FINANCE/PW,PUB;&
2  !INPRI=2;OUTCLASS=LP,8;SPSAVE
3  !COMMENT MYJOB2 PRINTS MYFILE2
4  !EDITOR
5  T MYFILE2.NEWS
6  L ALL,OFFLINE
7  EXIT
8  !EOJ
```

1. Modify the JOB command to show your user and account name as well as your passwords.
2. Verify the current jobfence. Modify the input priority in the JOB command to a value that enables the job to run.
3. You want to be sure that should the system halt while the job is processing, the processing restarts when the system is restarted. Enter that parameter in the JOB command to ensure that this happens.
4. Suppose that several times in the past a user has encountered some problems in processing MYFILE2. Therefore, you want to check the job listing before it is printed.

   Verify the current system outfence:

   ```
   SHOWOUT
   ```

   Modify the OUTCLASS parameter to a value that ensures that the job listing will not print.
5. MYFILE2 is now located in the CLASS group in your account. Modify the appropriate lines in the job file to reflect MYFILE2's correct location.
6. Add a line in the job file to provide a screen message noting that your job has completed processing.
7. Check over the file once again to ensure that all JOB commands are preceded by an exclamation point (!).
8. Check your modifications against those in the *Solutions Guide*. Then process MYJOB2 by entering

   ```
   STREAM MYJOB2
   ```

   If all the changes were made correctly, you should soon see a message similar to this:

```
FROM/Jxx USERx.ACCTx  MYJOB2 IS COMPLETE
```

Two sorts of errors can occur the jobs you have been working with.

a. The STREAM command detects a significant error in the job file itself—a missing or invalid !JOB command; a missing or invalid password; account or groupname; a missing or invalid device specification.

   Such errors will be reported on your screen. If the error is serious, the stream or the job execution may fail.

b. In the jobs that you have used so far, the editor detects an error—an invalid file name; an editor command to change something not found in your text.

   Such errors will turn up in your job listing.

That is why it is a good idea to specify ;SPSAVE in your !JOB command line. That way, you can use the PRINT command to examine your spool file to see what error the editor encountered. (In a later section, you will learn how to use LISTSPF and the PRINT command to check for errors.)

********** End of Exercise 3-2 **********

Lesson summary

1. The JOB command initiates a batch job and contains required information regarding the location of the relevant files, user and account names, and processing priority.
2. A job must have an input priority higher than the current jobfence in order to begin processing.
3. The job listing prints only if the job's output priority is above the outfence priority.
4. The CONTINUE command ensures that a job will continue to process if it appears before a command line that may cause an error during the job's processing.
5. The RESTART parameter ensures that an executing job will restart in the event of a system halt and system start, provided that the system is restarted with the RECOVERY option.

Lesson 3 Creating and Streaming a Job File

Introduction

Lesson 3 presents the following MPE/iX job-related tasks:

- create a job file according to specifications
- stream the job so that it runs at a scheduled time, day, or date
- suspend, resume, and alter a job
- Abort a job.

Up to this point you have worked with existing job files, examining them and modifying them. You will now create your own job files to perform specific tasks. Later in this lesson, you will actually stream the job files that you created and look at the results.

Teaching exercise 3-3: creating a job file

You've had the chance to examine and modify a job file. Now, you will step through an exercise that shows you how to create job files.

As you follow along, make sure that you are logged on as USERx.ACCTx in the `CLASS` group, where *X* = your student number.

You will be creating two job files according to specifications. These jobs will be streamed at a later time.

1. The job files will reside in the `CLASS` group and ACCTx account. You must replace the *X* to specify your account, user, and group passwords in the job files.

   Each job has the name specified in capital letters. The disk file name and the name of the job (specified in the `JOB` command) are the same.

   Example:

   File name = `JOB1`
   Job line in file = `!JOB JOB1`

2. Both jobs have the same general function:
   - Run the editor to modify a file called MYFILE, and keep it under a new name, MYFILEM.
   - List the contents of the modified file offline (for deferred printing).
   - Exit the editor and print a message telling the user the that job is done.

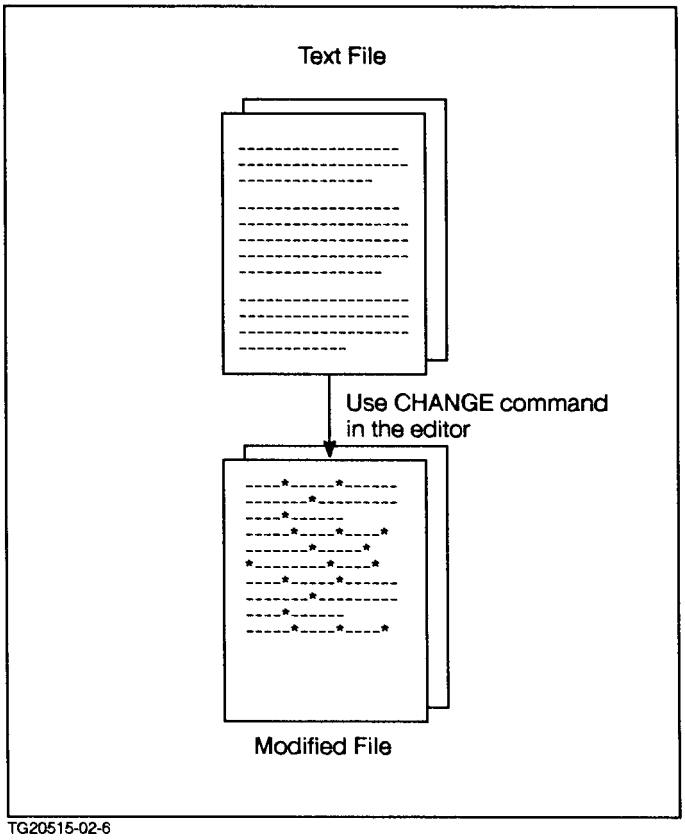


Figure 4-2. Modifying a File

3. Both jobs assume that the current jobfence is 7 and the outfence is 7. Any job with an output priority of 3 to 5, inclusive, will print after hours, when most of the employees have gone home.
4. Create JOB1 according to the following "skeletal" job file so that it performs the specified functions.

Note

Each letter refers to a function listed below the "skeletal" job file. The first three lines are the editor comments. *They do not belong in any job file you create.*

```
<<**************************>>
<<*FILE JOB1                >>
<<**************************>>
 a,b !
     !COMMENT JOB1 MODIFIES AND PRINTS MYFILE
     !CONTINUE
 c   !EDITOR
 d   T MYFILE.CLASS.ACCTx
 e   C "a" to "*A*" IN 1/4000
     C "b" to "*B*" IN 1/4000
     C "c" to "*C*" IN 1/4000
     C "d" to "*D*" IN 1/4000
     C "e" to "*E*" IN 1/4000
     C "f" to "*F*" IN 1/4000
     C "g" to "*G*" IN 1/4000
     C "h" to "*H*" IN 1/4000
 f   KEEP MYFILEM.CLASS.ACCTx;YES
```

```
g   L ALL,OFFLINE
h   EXIT
i   !
j   !
```

5. In your JOB1 file, complete lines a, b, d, e, f, g, h, i, and j of JOB1 to do the following:
   a. Specify an input priority high enough to guarantee that the job runs immediately. This means that the input priority must be above the current JOBFENCE (7).
   b. Suppress the printing of the job listing on the printer (LP). This means that the OUTCLASS priority (1) is less than the current OUTFENCE (7). Ensure that the job will restart in the event of a system halt.
   c. Start the editor.
   d. Text in a file called MYFILE.
   e. Use the CHANGE command to change the letters: a through h to *A* through *H* in lines 1 to 4000 of the file.
   f. Keep the modified file under the name MYFILEM.
   g. List the modified file offline.
   h. Exit the editor.
   i. Tell the user that the job is done.
   j. End the job.
6. You should now have a job file that looks like the JOB1 file in the following illustration.

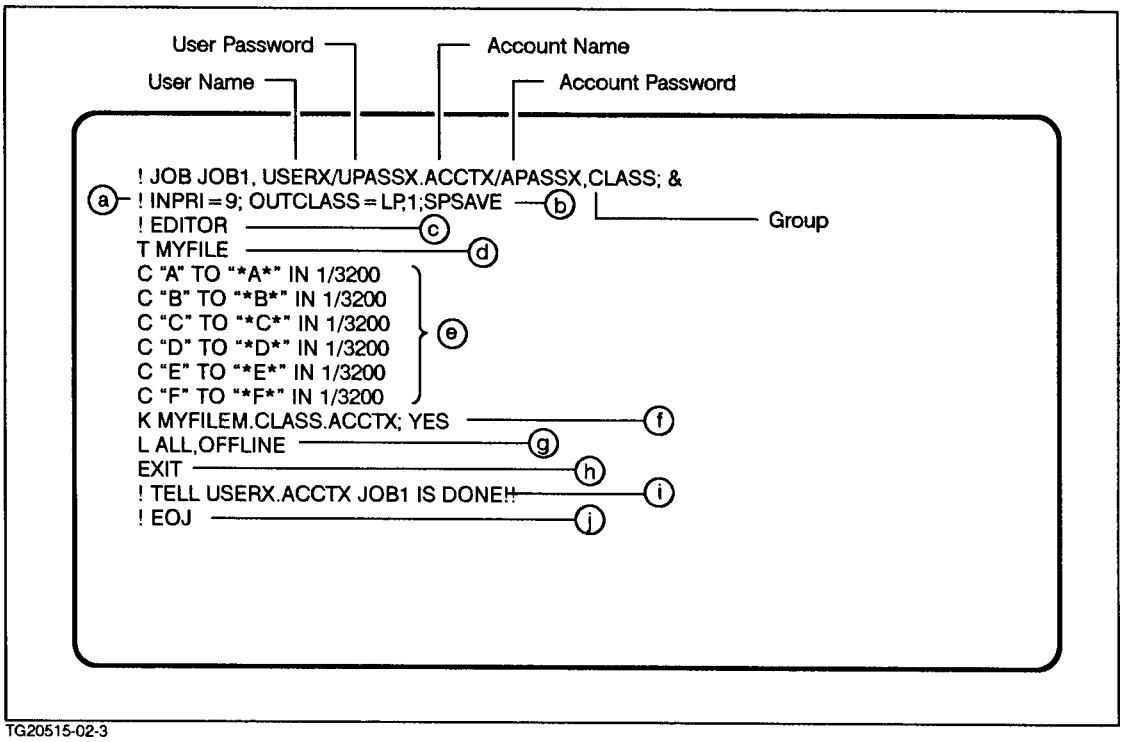


Figure 4-3. `JOB1` **File**

7. Using your JOB2 file, create JOB2 according to the "skeletal" job file below so that it performs the specified functions.

Note

Each letter refers to a function listed below the "skeletal" job file. The first three lines beginning with `<<` are the editor comments. *They do not belong in any job file you create.*

```
<<**************************>>
<<*FILE JOB2                >>
<<**************************>>
 a,b !
     !COMMENT JOB2 MODIFIES AND PRINTS MYFILEM
     !CONTINUE
 c   !
 d   T MYFILEM.CLASS.ACCTx
 e   C "*A*" TO "a" in 1/4000
     C "*B*" TO "b" in 1/4000
     C "*C*" TO "c" in 1/4000
     C "*D*" TO "d" in 1/4000
     C "*E*" TO "e" in 1/4000
     C "*F*" TO "f" in 1/4000
     C "*G*" TO "g" in 1/4000
     C "*H*" TO "h" in 1/4000
 f   K MYFILE.CLASS.ACCTx;YES
 g   L ALL,OFFLINE
 h   EXIT
 i   !
 j   !
```

8. Complete lines `a`, `b`, `c`, `d`, `e`, `g`, `h`, `i`, and `j` of JOB2 to do the following:

a. Specify an input priority that guarantees that the job runs now. This means that the input priority (9) is above the current JOBFENCE (7).

b. Suppress the printing of the error/job listing on the printer (LP). This means that the OUTCLASS priority (5) is less than the current OUTFENCE (7). Ensure that the job will restart in the event of a system halt.

c. Run the editor.

d. Text in a file called MYFILEM.

e. Use the CHANGE command to change the letters: *A* through *H* to a through h in lines 1 to 4000 of the file.

f. Keep the modified file under the name MYFILE.

g. List the modified file offline.

h. Exit the editor.

i. Tell the user that the job is done.

j. End the job.

9. You should now have a job file that looks like this:

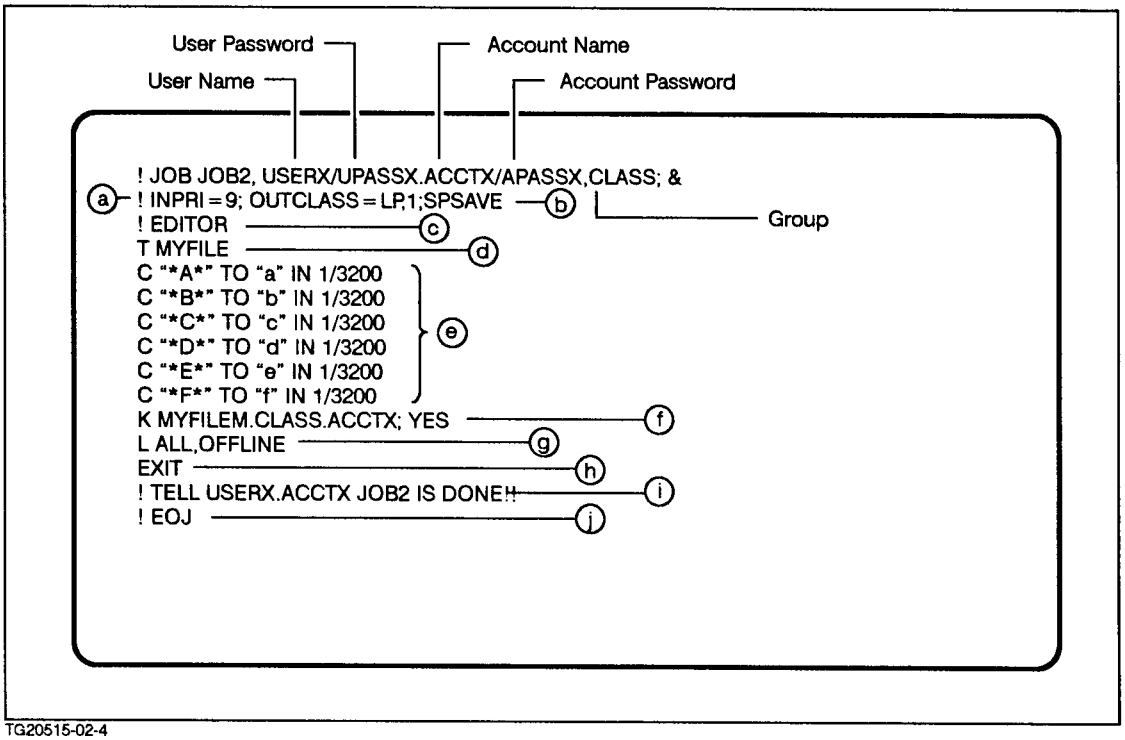


Figure 4-4. JOB2 File

********** End of Exercise 3-3 **********

Streaming a job

Now that you have had the chance to create job files on your own, let's learn what you can do to get those jobs to execute. The process of doing this is called streaming a job. This is accomplished with the `STREAM` command. The `STREAM` command has options that allow you to actually specify the time and date that you wish a job to run.

Note

Before you stream a job, check the job file to make sure that the user and account passwords are correct. You can use the `LISTUSER ... ;PASS` or `LISTACCT ... ;PASS` commands to do so. (You must have SM capability.)

Type this simple version of the `STREAM` command using the `JOB1` file:

```
STREAM JOB1
```

You will notice that the job number (`#Jxxx`) appears on the screen. Note this number.

`JOB1` immediately enters the job queue and executes when system resources are available. It will execute exactly as specified in the job file, as if you were entering the commands contained within it at the terminal.

Q3-24	Do a `SHOWJOB` to verify that the job is processing. What is the job's state?

Now suppose that you wish the job to execute late at night at a particular time when system resources are more available.

Please enter the following `STREAM` command to specify an exact time for your job to run after hours (specify a time at least one hour later than the current time):

```
STREAM JOB1; AT=03:00
```

In the example above, the job will run at 3:00 A.M. It will print several copies of a previous lesson later in the early morning.

Q3-25	Do a `SHOWJOB` to check the job state. What is it?

There are several other options that you can specify with the `STREAM` command:

`STREAM jobname;AT=` Lets you specify a *time* to run the job.

`;DAY=` Lets you specify a *day* to run the job.

`;DATE=` Lets you specify a *date* to run the job.

`;IN=` Lets you specify a *relative time, day, or date* to run the job (for example, two hours from now, or two days from now).

See the following figure for examples of the various options. These options are explained in depth in the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) and in the Help facility, under the STREAM command.

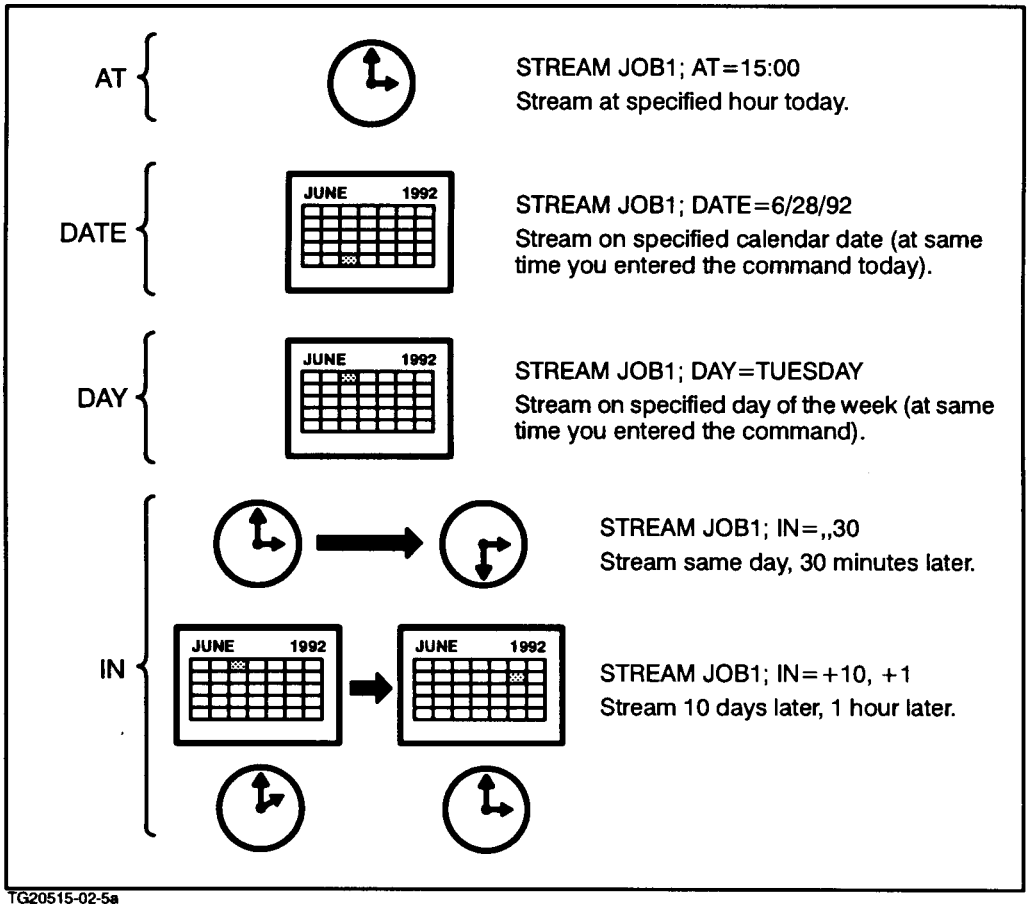


Figure 4-5. STREAM Command Options

If you specify a job start at 3:00 PM (15:00) and it is now 3:01 PM, your job will start at 3:00 PM the *next* day.

For a greater understanding of these options, study the following examples:

1. Assume it is now 8:00 P.M. Suppose that you want JOBFILE to run at midnight, 10 days from now. What STREAM command do you enter?

```
STREAM JOBFILE IN=+10,+4
```

2. Suppose that you want a JOBFILE to run today, three hours after you submit it. What STREAM command do you enter?

```
STREAM JOBFILE;IN=,+3
```

Match the descriptions in the following table with the correct STREAM command. (Assume it is Sunday, August 1, 1989, at 6:00 A.M., and jobs are currently executing on the system.)

Q3-26

Command		Description	
1.	`STREAM JOBFILE`	a.	Job will run on Monday, at 10:00 P.M.
2.	`STREAM JOBFILE ;AT=8:00`	b.	Job will run on August 9, at 8:00 P.M.
3.	`STREAM JOBFILE ;AT=20:00`	c.	Job will run today, at 8:00 A.M.
4.	`STREAM JOBFILE ;IN=,+8`	d.	Job will run immediately.
5.	`STREAM JOBFILE ;IN=+1,+8`	e.	Job will run on August 9, at the same time you execute the `STREAM` command today.
6.	`STREAM JOBFILE ;DATE=08/09/89 ;AT=20:00`	f.	Job will run today, 8 hours from now.
7.	`STREAM JOBFILE ;DAY=MONDAY ;AT=22:00`	g.	Job will run today, at 8:00 P.M.
8.	`STREAM JOBFILE ;DATE=08/09/89`	h.	Job will run tomorrow, 8 hours from now.

Q3-27 If you specify a day or date, but no time in the `STREAM` command, at what time will the job execute?

Aborting a job

After you enter the `STREAM` command, you may decide that you do not wish the job to execute. This may happen if you discover a text error in a file that you were planning to format, or in a program you were planning to run. In this case, you should abort the job. The following command does this:

```
ABORTJOB #J[jobnumber]
```

Note

You can abort a job only if it is your job, and if the system operator has set `JOBSECURITY` to `LOW` (or the operator issues the `ALLOW` command so that you can issue `ABORTJOB`.) Be aware that if you have several logons to the same account, a job submitted by one logon cannot be aborted when using another logon.

Determine whether `JOB1` has finished executing.

If it has not, use `ABORTJOB #J`*jobnumber*.

The job number should be the one you noted earlier. If the job has already completed, enter the following command.

```
STREAM JOB1
```

Note the resulting job number that displays after you enter the command.

Verify that the job is processing by entering the following command (note the job number):

```
SHOWJOB
```

Now abort the job using the job number and do a SHOWJOB again to verify that the job no longer exists.

Note

Sometimes ABORTJOB takes a few seconds to abort a job. Consequently, you may have time to execute a SHOWJOB and see your job still listed. Do not be surprised if that happens. ABORTJOB will abort your job.

The processed part of the job still prints out (whatever part of it was processed) even if you use ABORTJOB. In the next lesson you will learn how to prevent this from happening.

Q3-28 When you do a SHOWJOB #Jxxx, what message do you see after the job has been aborted?

Q3-29 Imagine that it is now 9:00 P.M. At 8:00 P.M. you entered the following STREAM command:

```
STREAM JOB1;AT=00:00
```

You have changed your mind and wish to abort the job. At what time is the job supposed to run? How would you abort it (since the job is not yet running)?

Suspending a job

Occasionally, after streaming a job, you may decide that the system resources are such that it will take too long to run the job (perhaps many other jobs are also running simultaneously and the CPU has to swap between them all). Rather than aborting your job and then restreaming it later, you can suspend it with the BREAKJOB command and resume it with the RESUMEJOB command:

```
BREAKJOB #Jjobnumber
RESUMEJOB #Jjobnumber
```

Take Care

A job may involve accessing databases or files that become locked during access. If you do a BREAKJOB while executing such a job, you run the risk of preventing others from accessing those databases or files. This is due to the fact that while your job is suspended, the locks remain on the databases or files that were previously being accessed. This can have detrimental effects on other users. Please be aware of the function of a job before you suspend or abort it.

By doing this, you do not lose any of the job processing you have accomplished. Plus, you do not lose your place in the job processing queue. All that happens is that system resources are freed to process other jobs ahead of you (so that less "swapping" has to occur). Once you see that those jobs are done, you can resume your job with the

RESUMEJOB command. Jobs that are suspended are listed as being in the SUSP state when you do a SHOWJOB.

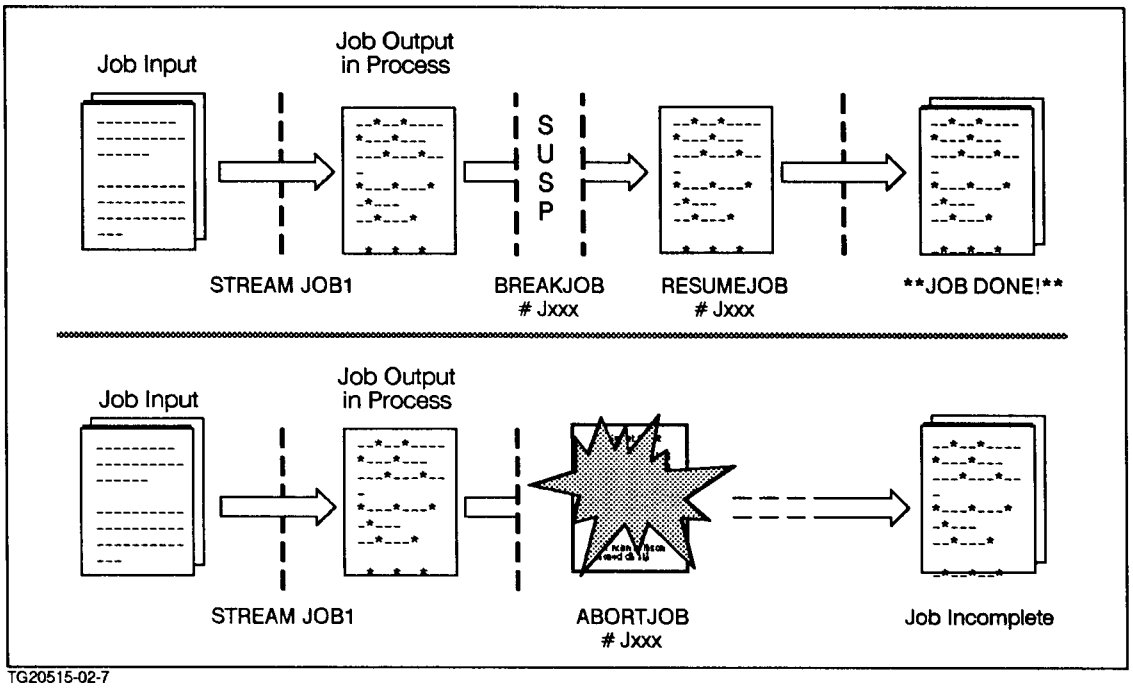


Figure 4-6. Resuming or Aborting Jobs

Note

You can suspend a job only if it is your job and the system operator has set JOBSECURITY to LOW (or the operator issues the ALLOW command so that you can issue BREAKJOB).

Q3-30	To test the BREAKJOB command, please STREAM JOB1. Allow the job to process by waiting about 10-15 seconds. Then enter the BREAKJOB command. Do a SHOWJOB to verify that the job has been suspended. What is the status of that job? To use BREAKJOB successfully, you may have to wait more or less than the 10-15 seconds, depending on the system load.
Q3-31	Now resume the job with the RESUMEJOB command and check its status with SHOWJOB. What is its status?

Altering a job

If a job is in INTRO, WAIT or SCHED state (has not run yet), you can alter its input priority or output device with the ALTJOB command:

```
ALTJOB #Jjobnumber [;INPRI=input_priority][;OUTDEV=device_name]
```

Make sure that JOB1 has completed. If it is still running, abort it. Now, test the ALTJOB command by restreaming JOB1:

```
STREAM JOB1
```

Note the job number. Try to change the input priority to 13. What happens? You'll find that you cannot ALTJOB because the job is

currently in the EXEC state. You must be in the INTRO, WAIT, or SCHED state.

To terminate JOB1, abort it:

```
ABORTJOB #Jxxx
```

Now, return to the editor and modify the JOB1 file so that the input priority is reduced to 1. Keep the file and restream JOB1:

```
STREAM JOB1
```

Note

You can alter a job only if it is your job and the system operator has set JOBSECURITY to LOW (or the operator issues the ALLOW command so that you can ALTJOB).

Q3-32	Do a SHOWJOB to check JOB1's current state. What is it?
Q3-33	Again, alter the input priority, this time to 9. Check on JOB1's state again. What is it now?

Finally, abort the job so that it neither continues processing, nor prints (remember, it only has an output priority of 5, which defers printing).

```
ABORTJOB #Jxxx
```

Using ALTJOB is a good way to ensure that a job runs, even if initially you set the input priority to a very low number. Also, suspending the job prevents it from continuing and from printing. In the next lesson, you'll learn how to purge the spool files that remain from suspended jobs that you no longer wish to resume.

Note

Make sure to use the editor once more to modify JOB1. Change the input priority in the INPRI line to a value that is high enough to guarantee that the job runs.

Lesson summary

1. The STREAM command uses the following options to schedule jobs:

   The ;AT option specifies an absolute time (hour:minute:second).

   The ;DATE option specifies an absolute date (month/day/year).

   The ;DAY option specifies a day of the week.

   The ;IN option specifies a relative day or time (day,hour,minute).

2. The following commands control job execution:
   - The BREAKJOB command suspends a job.
   - The RESUMEJOB command resumes a suspended job.
   - The ALTJOB command alters the input priority or the device or device class of a scheduled or waiting job (does not work on a suspended job).
   - The ABORTJOB command aborts a job.

Lesson 4
Monitoring Jobs

Introduction

Lesson 4 presents the following command: LISTSPF and SPOOLF. These commands let you monitor job progress:

You had the opportunity in the last lesson to STREAM jobs that were scheduled to run at various times, or that were purposely set *not* to print (deferred priorities). Now you will monitor the progress of the jobs and look at the job listings of your completed jobs. This allows you to determine whether or not the job was successful. And you will learn how to alter the output priority so that a job listing can print.

SPOOLER commands

The spooler commands LISTSPF and SPOOLF let you look at, manipulate, or delete spool files and job listings.

A spool file is an intermediate file that is awaiting printing. Both jobs and sessions may create spool files, such as $STDLIST (your job listing)—so can other processes or applications that you use on the computer. *Spool* actually stands for *simultaneous peripheral operations online.*

Spool Files are needed because output devices such as printers can handle only one thing at a time. They are *nonshareable*. Several jobs or sessions may attempt to use the printer simultaneously. That will not work. So, each job or session item that is destined for the printer is stored in a a spool file to await its turn. The system sends the spool files to the printer in order: the highest-priority items go first. If two items have the same priority, then the system prints them on a first-come-first-served basis.

Displaying spool file information

Note

Make sure that you are logged on as USERx.ACCTx, in the CLASS group. Also make sure that the file designator, LP exists on your system, so that your jobs print on some device (line printer).

First, you must stream a job so that a spool file is created. Enter the following command to execute JOB1, which modifies a file:

```
STREAM JOB1
```

(Note the job number that is displayed: #Jxxx.)

To look at the output (going to the printer) spool files for your *user.account*, enter:

```
LISTSPF
```

Unless no spool files exist at the moment, you should now see a display similar to this. Note the files associated with the JOB1 job number. In this example, the JOB1 job number is #J172.

```
SPOOLID    JOBNUM   FILEDES    PRI COPIES DEV STATE  RSPFN OWNER
```

```
#O1585     J172     $STDLIST     1      1 LP  CREATE       USER1.ACCT1

INPUT SPOOL FILES            OUTPUT SPOOL FILES
ACTIVE   = 0;                CREATE   = 1;              READY    = 0;
OPEN     = 0;                DEFER    = 0;              SELECTED = 0;
READY    = 0;                DELPND   = 0;              SPSAVE   = 0;
                             PRINT    = 0;              XFER     = 0
                             PROBLM   = 0;

TOTAL IN FILES   = 0;        TOTAL OUT FILES   = 1;
      IN SECTORS = 0;              OUT SECTORS = 2048;

OUTFENCE = 7
OUTFENCE = 7 FOR LDEV 6
```

Note the files associated with the JOB1 job number. In this example, the JOB1 job number is #J172.

Input and output spool files

Output spool files are identified by #O (pound sign and the letter "oh") followed by a number. Input spool files (those destined not for the printer but for some processing by the system) are identified by #I (pound sign and the letter "i" followed by a number).

The LISTSPF O@ command displays all the output spool files for your *user.account*. This is the default. The LISTSPF command show exactly the same information.

The LISTSPF I@ command displays all the input spool files for your *user.account*.

The LISTSPF @ command displays all the input and output spool files for your *user.account*.

Note

You see only the spool files associated with your *user.account*.

Most input spool files are "private." Access to them is restricted.

To see a listing of spool files belonging to other users, use this form:

```
LISTSPF @;SELEQ=[OWNER=user.account]
```

You must enter the brackets as part of your command.

The ;SELEQ=[...] option permits you to select (or filter) the LISTSPF display in several ways. The OWNER option is one of the selecting methods. You will find information about the others in the online help facility.

The information that you see on the screen helps you to determine whether your job has completed producing its output and whether the output is printing or not. Each job that you run is identified by a job spool file number. The various spool files associated with that job are identified by input and output identification numbers. The headings on the screen help you identify the information you see there. Examine the following table to learn more about the headings.

Heading	Definition
SPOOLID	A spool file number—also called a spool file ID number (typically there is more than one spool file per job): Input files begin with "I". Output files begin with "O"
JOBNUM	Job number.
FILEDES	The file designator for the spool file.
DEV	Device where spool file will eventually be printed.
PRI	Output priority. (This determines whether the spool file will print or not, depending on whether the priority is greater than or less than the OUTFENCE of the DEV.)
COPIES	Number of copies of the spool file that will be printed.
STATE	"Condition" of the file: CREATE = output spool file being created READY = ready to be printed ACTIVE = input spool file being created OPEN = input spool file being read by a job PRINT = output spool file being printed DEFER = output spool file in deferred state SPSAVE = spool file has been saved after printing PROBLM = spool file destination cannot be found DELPND = delete flag has been set—spool file will be deleted when the last person accessing it closes the spool file XFER = being transferred to another system to print
RSPFN	Flag indicating more spool file information: R—$STDIN input spool file is restartable (;RESTART). S—SPSAVE specified. P—a private (protected) spool file. F—special forms messages attached. File will print on a special form. N—spool file incomplete (insufficient disk space or system aborted during file creation).
OWNER	*user.account* in the !JOB command of the job file for a $STDLIST or $STDIN. Generally, the *user.account* that originated the file or generated the output.

Where spool files are kept

MPE/iX has one system account that holds most commonly generated spool files: HPSPOOL. Input spool files are found in the IN group. Output spool files are found in the OUT group.

Q3-34	How many files are associated with JOB1? What are they called?

More detailed information about your spool files is available with the ;DETAIL option of the LISTSPF command. To display more information about the output file #O1585, enter the LISTSPF command this way:

```
LISTSPF #O1585;DETAIL
```

You do not have to include #. You can leave out the O, too, since that merely identifies the file as an output file.

Your display should now look something like this:

```
SPOOLID JOBNUM   FILEDES  PRI     COPIES DEV    STATE RSPFN OWNER
                 FORMID   JOBNAME COPSRM SECTS  RECS  PAGES DATE  TIME

#O1585  J172     $STDLIST LP         1     LP    CREATE     USER1.ACCT1
                          JOB1       1   2048    0

INPUT SPOOL FILES            OUTPUT SPOOL FILES
ACTIVE   = 0;                CREATE   = 1;              READY    = 0;
OPEN     = 0;                DEFER    = 0;              SELECTED = 0;
READY    = 0;                DELPND   = 0;              SPSAVE   = 0;
                             PRINT    = 0;              XFER     = 0;
                             PROBLM   = 0;

TOTAL IN FILES   = 0;        TOTAL OUT FILES   = 1;
      IN SECTORS = 0;              OUT SECTORS = 2048;

OUTFENCE = 4
OUTFENCE = 4 FOR LDEV 6
```

The LISTFSP ;DETAIL display shows a second line of headings and new information. Note the following *new* headings:

Heading	Definition
FORMID	Identifies and special forms needed to print this file
JOBNAME	Job name of the job that created the spool file
COPSRM	Copies of the file remaining to be printed
SECTS	Total number of disk sectors used by the spool file
RECS	Total number of records in the spool file
PAGES	Number of physical pages in the spool file (the ~ indicates an approximate number)
DATE	Date when the spool file became READY
TIME	Time when the spool file became READY

For JOB1, $STDLIST should be READY when approximately 13144 lines have been written to it (all CHANGEs complete). At this point, do a LISTSPF, and an entry for EDTLIST may appear:

```
SPOOLID  JOBNUM   FILEDES  PRI COPIES DEV  STATE  RSPFN OWNER

#O1585   J172     EDTLIST  8      1   LP   READY        USER1.ACCT1
#O1585   J172     $STDLIST 1      1   LP   CREATE       USER1.ACCT1

INPUT SPOOL FILES           OUTPUT SPOOL FILES
ACTIVE   = 0;               CREATE   = 1;              READY    = 1;
OPEN     = 1;               DEFER    = 0;              SELECTED = 0;
READY    = 0;               DELPND   = 0;              SPSAVE   = 0;
                            PRINT    = 0;
                            PROBLM   = 0;

TOTAL IN FILES   = 1;        TOTAL OUT FILES   = 2;
      IN SECTORS = 36;             OUT SECTORS = 3205;

OUTFENCE = 7
OUTFENCE = 7 FOR LDEV 6
```

EDTLIST

EDTLIST is the editor's formal file designator for the offline listing of the modified file. It will disappear from the spool file listing after the text of MYFILE1 has printed.

To see if the job processed normally, look at the $STDLIST file (job listing). It shows you how the CHANGE commands operated on each line of MYFILE:

Using PRINT to check for errors

You may use the PRINT command to examine the contents of a spool file. Output spool files (begin with O—the letter "oh"—followed by digits).

To examine a spool file with the PRINT command,

```
PRINT spoolid.OUT.HPSPOOL
```

For example, to examine the spool file #O1585 shown above, you would enter:

```
PRINT O1585.OUT.HPSPOOL
```

Remember that the O is the letter "oh" and not a zero.

Note

PRINT may warn you that it has truncated some of the lines in the spool file as it displays the file. Don't be concerned. Nothing is being ruined.

PRINT displays the first 23 lines of the file and pauses.

It shows the number of the next line to be displayed and the number of the last line in the file; and it asks if you want to continue with the display.

You see something like this on the bottom line of your screen:

```
(24/340) Continue?
```

The next line to be shown is 24. The last line that can be displayed is 340.

Press (RETURN) each time you want to see another 23 lines of the file.

If instead you want to see a particular line, enter that line number and press (Return):

```
(24/340) Continue? 301(Return)
```

In this example, PRINT displays line 301 and the next 22 lines following it.

You can jump forward by 40 lines by entering +40; or jump backward by 40 lines by entering -40.

If you do not want to see more of the file, enter N (for No) and press (Return).

Note

Most commonly generated output spool files are found in OUT.HPSPOOL. Most commonly generated input spool files are found in IN.HPSPOOL. There are other kinds of spool files (called *unlinked* spool files) that may reside in other accounts and groups. Highly experienced users sometimes have need of creating unlinked spool files for special purposes.

At the top of the file, you see several lines identifying the job and its characteristics. Next comes your system's welcome message, followed by the command lines that were executed. If the job produced errors or warnings, they will be in the file, too. At the bottom, you should see a message indicating whether or not the job ran successfully.

```
/K MYFILEM
YES
L ALL, OFFLINE
*** OFFLINE LISTING BEGUN ***
EXIT
END OF SUBSYSTEM
:TELL USER USERx.ACCTx JOB1 IS DONE!!
```

The $STDLIST spool file for your job normally goes to the printer, unless you have set the output priority at or below the current outfence value.

In JOB1 the output priority is set below the outfence value, to prevent the spool file from printing. Once it prints, it will disappear, unless you have included the ;SPSAVE option on the !JOB command line.

Notice the STATE of the $STDLIST after you've accessed it and listed it.

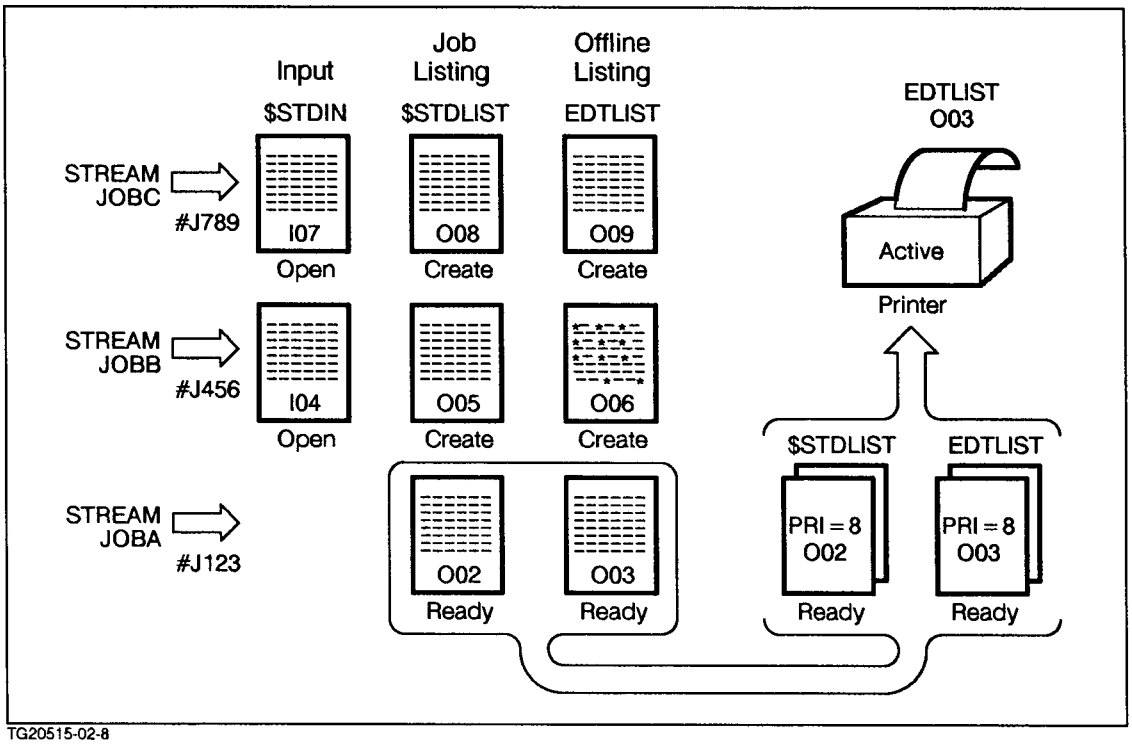


Figure 4-7.
Spool Files
EDTLIST Output from the Editor

Normally, if serious errors were indicated in $STDLIST, you would purge the spool file, return to the original job file, and correct the lines that were specified in the job listing. Then you would stream the job again.

Because the warnings were not serious (truncations), you need not correct them.

Note

You cannot use `PRINT` or the editor to look at the input spool file ($STDIN), since it is a private spool file.

Q3-35	What is the priority of the JOB1's $STDLIST spool file? Is it above or below the `OUTFENCE`?

Note

If you have forgotten the spoolid number, enter `LISTSPF` to display all the spoolid numbers.

To print the spool file, alter the priority so that it is above the `OUTFENCE`. Enter:

```
SPOOLF filenumber;ALTER;PRI=9
```

You have just altered the output priority from 5 (below the OUTFENCE), to 9 (above the OUTFENCE), and the file should be ready to print.

```
LISTSPF filenumber
```

If the state is PRINT, it is printing. If the state is READY, is *not* printing. Continue checking the status.

Q3-36	What happens when the spool file is no longer active on the printer?

As you may have noticed, the PRINT command gives you a way to determine if your processed job file had any errors.

Use LISTSPF again to see if there are any remaining spool files associated with your job. Purge any remaining spool files by entering the following command:

```
SPOOLF filenumber;DELETE
```

Warning

SPOOLF is a powerful command. Be certain of what you are doing when you use it.

If you have AM capability, SPOOLF @;DELETE will delete *all* of the spool files in the account.

If you have SM or OP capability, or if you are working at the console, SPOOLF @;DELETE will delete *all* spool files on the system.

To restrict the deletion to the spool file of a particular *user* and *account*, use this form:

```
SPOOLF @;SELEQ=[OWNER=user.account];DELETE
```

You must enter the left and right brackets as part of the command. The SELEQ=[...] option works for SPOOLF as well as for LISTSPF. LISTSPF is a "safe" command: it can do no harm if you misuse it. SPOOLF without SELEQ=[...] is capable of ruining your spool files and those of others. Use caution.

To purge more than one spool file, use:

```
SPOOLF (filenumberone, filenumber ...
);SELEQ=[OWNER=user.account];DELETE.
```

You must enclose a list of *filenumber*s in parentheses to delete the spool files simultaneously.

You cannot delete input spool files.

Good housekeeping dictates that you clean up your spool files. Usually system management will purge spool files whose priority is less than a certain value each night. In case this doesn't happen, you should purge spool files periodically.

If you have any other spool files in the READY state, purge them now.

Aborting jobs

What happens if you check a job's progress, and decide that the job has a serious problem? Unfortunately, you cannot use the editor at that point to examine the error/job listing when the job is still executing and its listing spool file is in the CREATE (not READY) state; however, you can use PRINT O<*filenumber*>.OUT.HPSPOOL to display the spool file while it is still in the CREATE state.

In this situation, you should immediately lower the priority so that the job listing will *not* print. This will give you time to examine the job listing to determine what went wrong, without printing the erroneous file.

Then use ABORTJOB to stop the execution of the job. This allows its $STDIN spool file to become READY. If the spool file was originally deferred, it still will not print, so no paper resources will be wasted.

The $STDLIST of an aborted job always prints unless its priority is below the OUTFENCE.

To practice this process, complete the following exercise.

Exercise 3-4: aborting a job and using print

1. STREAM JOB2. This job modifies MYFILEM. This file contains some lengthy records that produce truncation errors that don't affect printing.
2. Use the PRINT command to access and view the job listing associated with JOB2. (Do so *before* the job has completed.) What happens?
3. Now, alter the priority to 5.

   ```
   SPOOLF filenumber;ALTER;PRI=5
   ```

   a. What would happen if you did not alter the priority?

   b. Let the job finish executing.

   c. Can you now access and view the job listing with the PRINT command?

   d. What message appears at the end of the job listing?
4. Check the status of the job. Use LISTSPF with the $STDLIST spoolid number:

   ```
   LISTSPF filenumber
   ```

   What is the state of the spool file now that the job has been aborted?
5. Get rid of any remaining jobs and spool files.

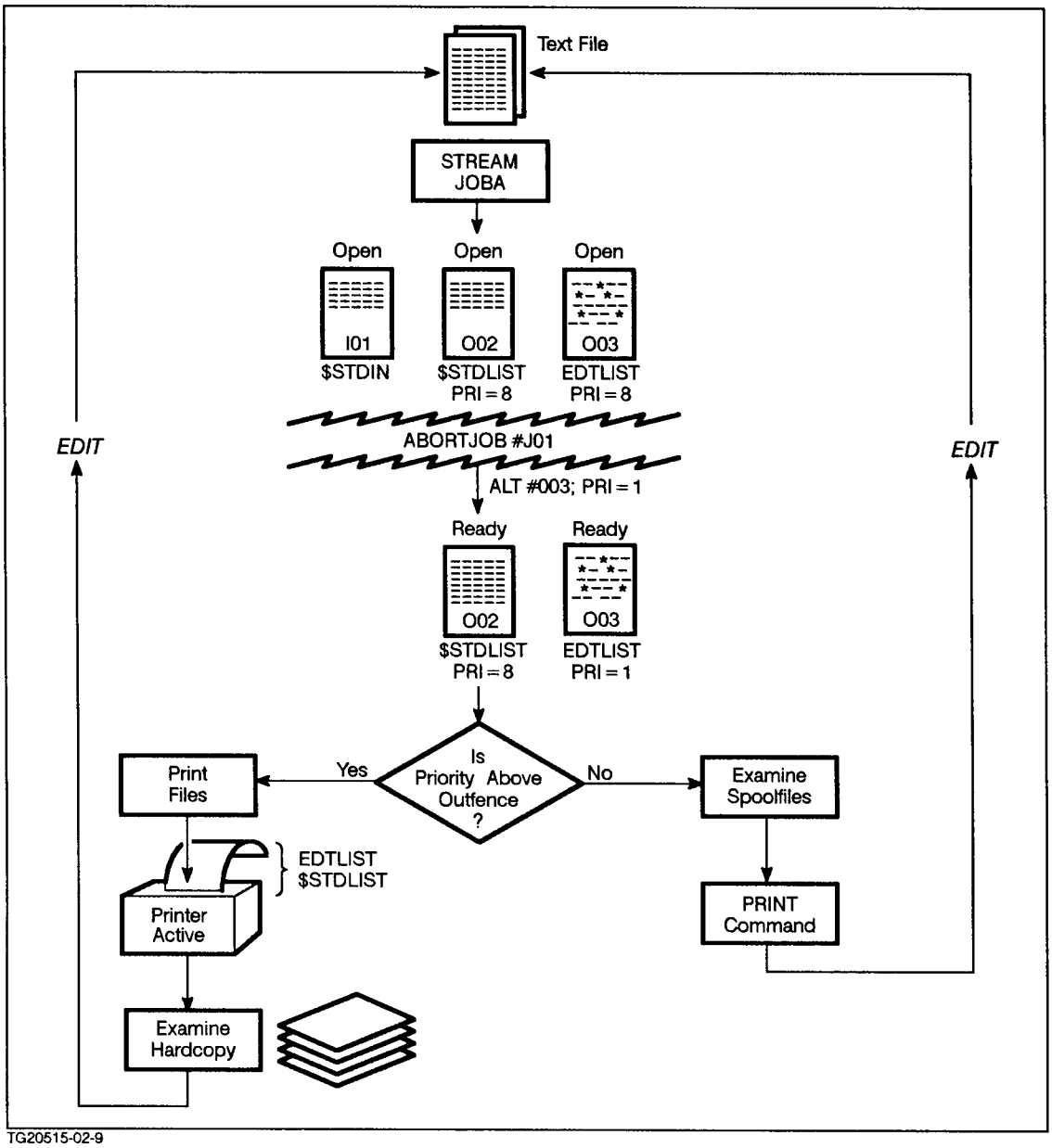


Figure 4-8.
SPOOLF and Aborted Jobs
EDTLIST Output from the Editor

Typically at this point, you would note where errors occurred. You could then purge the spool files, return to the editor, and make the necessary corrections to your job file. Please purge any remaining output spool files now:

```
SPOOLF @;SELEQ=[OWNER=USERx.ACCTx];DELETE
```

Warning

SPOOLF is a powerful command. Be certain of what you are doing when you use it.

If you have AM capability, SPOOLF @;DELETE will delete *all* of the spool files in the account.

If you have SM or OP capability, or if you are working at the console, SPOOLF @;DELETE will delete *all* spool files on the system.

6. Check to see if you have any additional spool files (LISTSPF @). These are probably associated with previously suspended or scheduled jobs. Please purge them now. If any $STDIN files remain, abort their associated jobs.

********** End of Exercise 3-4 **********

Any time you create a jobfile and STREAM it, use the LISTSPF and PRINT to monitor its progress and check for processing errors.

Lesson summary

1. The contents of a typical LISTSPF display include these:
   a. spoolid number
   b. job number
   c. file name
   d. state of spool file
   e. device (or device class) on which file is to be printed
   f. output priority
   g. owner of the spool file
   h. using the ;DETAIL option provides more information on your spool file(s)
2. All jobs create two kinds of spool files:
   a. input file ($STDIN)
   b. job listing file ($STDLIST)

   There are other kinds of spool files, but these two are basic to all jobs.
3. The common commands used to display and manage spool files are:
   a. The LISTSPF command shows all spool files associated with your logon user and account.
   b. The LISTSPF *filenumber* command monitors the progress of a particular spool file.
   c. The PRINT command displays the job listing to help you determine if the job file contained errors.
   d. The SPOOLF *filenumber*;DELETE command purges spool files.
   e. The ABORTJOB command aborts a job and causes its job listing to enter the READY state.
   f. The SPOOLF *filenumber*;ALTER;PRI=*outpri* command changes the output priority associated with a spool file.

g. The `SPOOLF` *filenumber*`;DELETE` command deletes the specified spool file.

Warning

`SPOOLF` is a powerful command. Be certain of what you are doing when you use it.

If you have AM capability, `SPOOLF @;DELETE` will delete *all* of the spool files in the account.

If you have SM or OP capability, or if you are working at the console, `SPOOLF @;DELETE` will delete *all* spool files on the system.

5

Module 4: File Transfer and Storage

Module 4 presents lessons on transferring files using FCOPY and STORE/RESTORE.

Lesson 1 Using FCOPY

Lesson 2 Storing and Restoring Files

Challenge Test

1. What `FCOPY` command would you use to copy the file `MYJOB1` from the `PUB` group to the `CLASS` group and not overwrite a file with the same name in the `CLASS` group?
2. What `FCOPY` command would allow you to copy the STATS file from the `PUB` group in the `ADVUSER` account to the `CLASS` group in your account, ACCTx?
3. Write the file equation and `FCOPY` command that will allow you to append three files, `FILE1`, `FILE2`, and `FILE3` to a new file, `FILEA` (assume you'll need 10,000 records in the new file).
4. Write the file equation and `FCOPY` command to copy `MYJOB1` to a tape where the tape is designated as `T`.
5. What `STORE` option would you add to the `STORE` command if you were going to restore this tape to an MPE system?
6. How would you modify the earlier file equation (4.) to allow the tape to be restored on a tape drive requiring 1600 bpi format?
7. Which `STORE` command parameters provide the following information?
   a. displays a message on a scheduled basis showing the progress
   b. prints a message every time a file is stored
   c. deletes a file as soon as the file is copied to tape
8. What file equation and `RESTORE` command would restore all the files noted in question 5?

Lesson 1
Using FCOPY

Introduction

Lesson 1 presents the following information on FCOPY:

- Comparison to COPY command.
- Copying files within and across accounts.
- Appending files.
- Copying files to and from other devices.

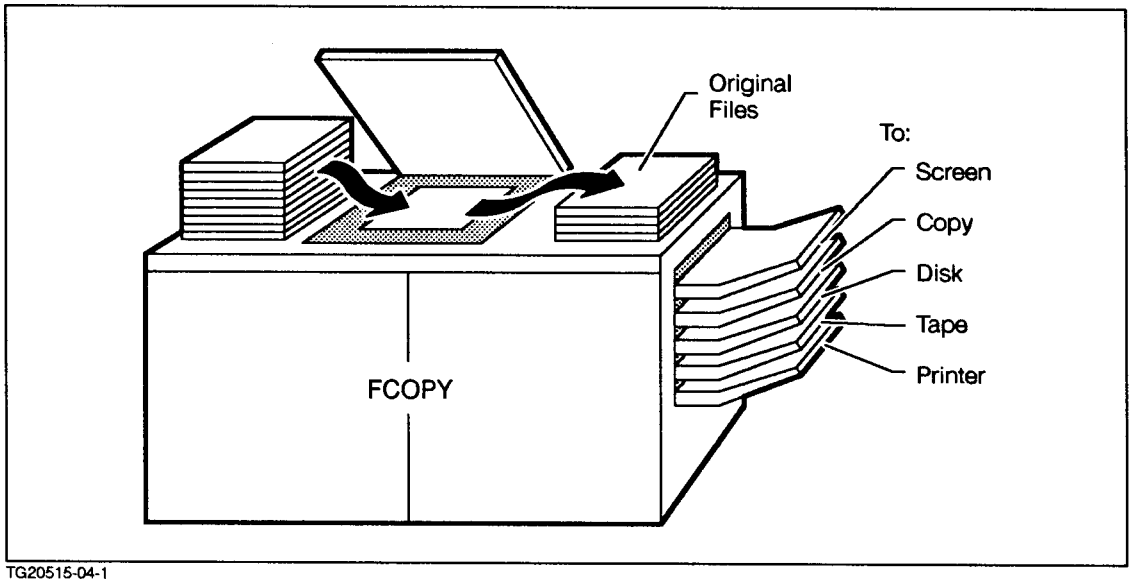


Figure 5-1. FCOPY

You have probably used the COPY command for copying files within an account. In this lesson you'll be learning about the FCOPY utility, which can perform additional copying functions within an account and give you more flexibility in copying files to other locations. This utility can be treated as a command by entering all of its parameters in one command line. This is the way you will be treating it throughout this module.

Be sure that you are logged into the CLASS group of your account.

FCOPY and COPY commands

FCOPY is used to copy single files from one location to another. Usually, this means copying files from one disk location to another; however, FCOPY can also be used to copy files from or to devices other than disk. However, for now concentrate on copying files within your account. When using FCOPY as a command, rather than as a utility, the syntax for the FCOPY command is similar to that of the COPY command:

```
COPY:        COPY FROM=filename;TO=filename
```

```
FCOPY:        FCOPY FROM=filename;TO=filename
```

Notice the locations of the equal signs (=) following the words FROM and TO. Many users make mistakes here.

The FROM information tells your system the name of the source file and its location. The TO information tells the system the name of the new or destination file and its location.

Adding the NEW option at the end of the FCOPY command assures you that the command will not copy the file over a previously existing file with the same name.

```
FCOPY FROM=filename;TO=filename;NEW
```

Adding the NO option at the end of the COPY command ensures the same thing as the FCOPY NEW option.

```
COPY FROM=filename;TO=filename;NO
```

Also note that semicolons are entered at the end of both the FROM and the TO information to separate the two parameters from each other.

FCOPY and COPY also differ in the way they copy files. The COPY command purges the file, then creates a new file with the characteristics of the source file. FCOPY copies into the file and tailors the source to destination file characteristics. This is why you cannot use the COPY command to alter the record size of a file. However, FCOPY allows this by truncating the records and fitting them into the destination file.

Copying files within an account

Like the COPY command, the FCOPY command can be used to copy files within an account and between accounts.

Suppose you want to duplicate the MYJOB1 file currently in the CLASS group and call the duplicate copy MYJOB1A. The command syntax looks like this:

```
FCOPY FROM=MYJOB1;TO=MYJOB1A;NEW
```

Enter that command now on your terminal.

Unlike COPY, which displays another prompt following a successful copy, FCOPY provides detailed information regarding the copy process at its successful conclusion.

Did you get the following information on your display noting that the copy was completed successfully? (If not read the following section.)

```
HP32212A.03.24 FILE COPIER (C) HEWLETT-PACKARD CO. 1984

EOF FOUND IN FROMFILE AFTER RECORD 9

10 RECORDS PROCESSED *** 0 ERRORS

END OF SUBSYSTEM
```

If you get an error message

If you got a syntax error, check your FCOPY command again to be sure that you have entered the command correctly.

```
*54* SYNTAX ERROR
```

If the syntax error message appeared, check the equal sign (=) after the FROM and TO, and the semicolon (;) before the TO.

```
*144* NEW OPTION:  FILE ALREADY EXISTS
```

If this NEW option error message appears change the new name for your file to MYJOB3 in your FCOPY command.

Copying files between groups

As an account manager you can copy files across groups in your account even though you are not the creator of the file or are not in the destination group.

For example, as the account manager you may want to keep a copy of a file (PFILE) in each of your account groups. The master copy of this file would remain in the PUB group.

Go ahead and copy this file (PFILE) from the PUB group to each of the groups in your account.

```
FCOPY FROM=PFILE.PUB;TO=PFILE.groupname;NEW
```

Note

Insert the name of the group (PROJECT or CLASS) for "groupname" when you want to copy the file to that group.

Q4-1	How is the the syntax of the FCOPY command that is used to copy files between groups different from the syntax of the FCOPY command that is used to copy files within a group?

Copying files between accounts

There are times when you will need to copy files between different accounts. The FCOPY command will let you do this with the following provisions.

- The file must be released at its source location.
- The file must be copied from the destination location.
- The source file name must be fully qualified.

Teaching exercise 4-1: copying files across accounts

This exercise will walk you through the procedure to copy a file from one account to another.

Note

If you are taking this course by yourself, ask someone who has an account on the system you are using to release a nonessential file for you to use for this practice exercise. Substitute the name of that file in the procedure.

Step 1. Ask another student to release a copy of the MYJOB1 file from their CLASS group. Find out their account name.

```
RELEASE MYJOB1
```

Step 2. If you are not already logged on to your own account, log on now.

Step 3. Enter the following FCOPY command on your keyboard substituting the other student's account name for ACCTx.

```
FCOPY FROM=MYJOB1.PUB.ACCTx;TO=MYJOB1.PROJECT;NEW
```

Step 4. Errors? Check your syntax and reenter. Check the next section, "Problems," if needed.

Step 5. Use the LISTFILE command to make sure the file transfer was completed successfully.

```
LISTFILE MYJOB1.PROJECT
```

Step 6. Ask the other student to secure the file.

```
SECURE MYJOB1
```

Go through the same procedure again for more practice. Copy the same file to your account; however, this time rename the file MYJOBX.

1. What command must be entered before a file can be copied to another account?
2. What FCOPY command should you enter to copy the file to your account and rename the file MYJOBX?
3. What command should you enter to make sure the file transfer was completed successfully?
4. What command should the owner of the file enter to secure the file following the copy process?

********** End of Exercise 4-1 **********

Q4-2	What additional information regarding the file to be copied is required when copying a file from a different account rather than from a different group?

Problems?

Check this section of the lesson if you had problems; otherwise, congratulations on your file transfer!

The most frequent error message is

```
CAN'T OPEN FROM FILE
```

To correct this, ask your partner who has the source file to again release the MYJOB1 file before repeating steps 3 through 6.

If you get a syntax error, examine your file command line again to make sure that it is written correctly.

Exercise 4-2: appending one file to another

There will be times when you want to append several files together. FCOPY can do this by creating a new file and then appending additional files to this file. In this exercise you will combine JOB1, JOB2, JOB3, and JOBO in a single JOBX file.

1. Determine the total number of records used by each file using LISTFILE, 1. Add these together.

   ```
   LISTFILE JOB?, 1
   ```

   The question mark tells LISTFILE to display all files that have JOB as the first three letters of the name and any fourth letter or number. The ? is a wildcard character.

2. Enter a FILE command to set up the new file JOBX. Give this file Append access.

   Enter the total number of records, plus 10 (to be on the safe side) as DISC=*number-of-records*.

   ```
   FILE F1=JOBX;REC=-88,,F,ASCII;ACC=APPEND;DISC=72
   ```

3. Copy JOB1 to the JOBX file. Backreference the file designator and use the NEW option in the FCOPY command. (You will get a warning indicating that the FROMFILE has records that are 80 bytes long, while the TOFILE has records that are 88 bytes long. You will then be asked if you wish to continue. Answer Y for Yes).

   ```
   FCOPY FROM=JOB1;TO=*F1;NEW
   ```

   You will get a warning indicating that the FROMFILE has records that are 80 bytes long, while the TOFILE has records that are 88 bytes long. You will then be asked if you wish to continue. Answer Y for Yes.

4. Use the LISTFILE,2 command to check on the existence of the new JOBX file. How many records does it have?

   ```
   LISTFILE JOBX,2
   ```

5. Append JOB2 to the JOBX file. Again backreference the file designator. Do not use the NEW option in the FCOPY command.

   ```
   FCOPY FROM=JOB2;TO=*F1
   ```

6. Use the LISTFILE,2 command to show the increased size of the JOBX file. How many records does it have?

   How much larger is the JOBX file now?

7. Append JOB3 to the JOBX file. What FCOPY command would you use to do this?

8. Again, check the size of the JOBX file.

   How large is the JOBX file now?

9. Append the JOBO file to the JOBX file.

   What FCOPY command would you use to do this?

10. Use the LISTFILE command to check on the size of the JOBX file.

********** End of Exercise 4-2 **********

Copying files from disk to other devices

To use the FCOPY command to copy files from disk to other devices (such as tape or peripherals), you must do the following:

- Use a FILE command to define a file name for your device. For example, assign the name LIST to the line printer (LP).

```
FILE LIST;DEV=LP
```

- Use the device's filename as the TO filename. Precede the file name with an asterisk to backreference it.

```
FCOPY FROM=MYJOB1.CLASS.ACCTx;TO=*LIST
```

Q4-3 Suppose you have assigned the name OUT to the line printer. How would you write the file equation to define the device? What FCOPY command would copy the MYJOB2 file to this device?

Copying files to and from devices other than disk

What if you want to copy files from a tape to a line printer? The FCOPY command can do this kind of copying as well. No file release is required. However, you must do the following:

- Define your device files.

```
FILE TAPEFILE;DEV=TAPE;REC=-80,,F,ASCII
```

```
FILE PRINTER;DEV=LP
```

- Use the two formal file designators as the FROM file and TO file in an FCOPY command. Backreference the filenames with asterisks.

```
FCOPY FROM=*TAPEFILE;TO=*PRINTER
```

Note

TAPEFILE and PRINTER are the formal file designators in the FCOPY command. TAPE and LP are device class names for a magnetic tape unit and a line printer, respectively. In this case you must define the size of the records on the tape.

Q4-4 Assuming that the file information and file name designators are correct, will the following file equations and FCOPY command copy MYJOB1 to tape?

```
FILE T;DEV=TAPE;REC=-80,,F,ASCII

FCOPY FROM=MYJOB1.CLASS.ACCTx;TO=T
```

Lesson summary

1. The FCOPY and COPY commands are very similar in syntax; however, FCOPY uses the NEW option to prevent overwriting a file; COPY uses a NO option.

```
COPY FROM=MYFILE;TO=MYFILE.PUB;NO

FCOPY FROM=MYFILE;TO=MYFILE.PUB;NEW
```

2. Both FCOPY and COPY allow you to copy a file from another account to your current account. You issue the respective command from the destination account and fully qualify the source file name.

```
FCOPY FROM=MYFILE.PUB.ACCTx;TO=MYFILE;NEW
```

3. When using FCOPY to copy files to and from other devices, first define your device files and then backreference the file designators in your FCOPY command.

```
FILE PRINTER;DEV=LP

FCOPY FROM=MYFILE;TO=*PRINTER
```

4. To append files, first create a formal file with the FILE command and the APPEND option. Backreference the file designator in each FCOPY command to append each successive file.

```
FILE F=FILE1;REC=-80...,;ACC=APPEND:DISC=number-of-records
FCOPY FROM=FILE1;TO=*F;NEW
FCOPY FROM=FILE2;TO=*F
```

Exercise 4-3: lesson 1 review

1. Note which tasks you are required to do when copying files:

Tasks		Within Group	Within Account	Account to Account	Device to Device
a.	Release file before copying.				
b.	Define the device file(s).				
c.	Fully qualify the file names (file.group.account).				
d.	Backreference device file names with asterisks.				
e.	Require only the FROM filename and the TO filename.				
f.	Be in the destination account when you issue the FCOPY command.				
g.	Secure the file after copying.				
h.	Note the group names for the source and destination file locations.				

2. What FCOPY command would you use to copy the DEPTCEN file from the PUB group in the FINANCE account to the PROJECT group in your account?
3. Suppose you would like to copy MYFILE1 currently in your CLASS group to tape. Write the file equation to define the device (tape) and the FCOPY command to copy the file.
4. Correctly sequence the steps to copy a file from one account to another. (Assume you are logged on to the source account.)

   a. Log on to the source account and secure the original file.

   b. Release the file.

   c. Enter the FCOPY command.

   d. Log on to the destination account.

   e. Use LISTFILE to check the file transfer.

********** End of Exercise 4-3 **********

Lesson 2 Storing and Restoring Files from Tape

Introduction

Lesson 2 presents the STORE/RESTORE utility:

- using STORE to copy files to tape
- using RESTORE to copy files from tape to a system

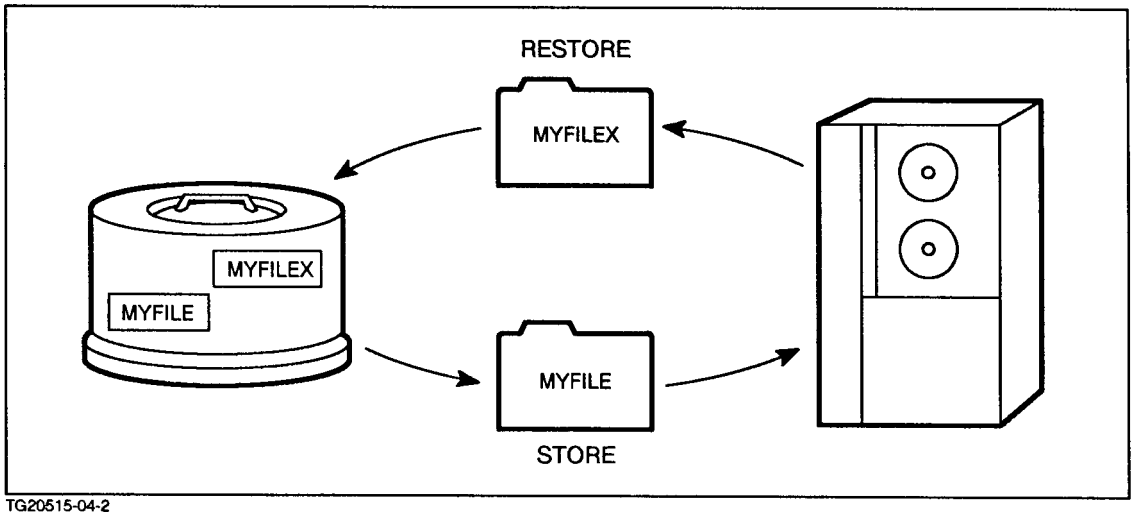


Figure 5-2. STORE/RESTORE Utility Subsystem

Note

The user procedures established by system management for storing and restoring files to and from tape may vary from facility to facility. The user procedures described in this lesson assume that system operators load and unload tapes from the tape drives, and that users are responsible for entering the equation to define the device and the command to store or restore the file(s).

Prior to using the STORE/RESTORE subsystem utility, check with your system management for specific procedures for your facility.

Consider the following situations:

- *Scenario 1:* You are moving to a new account on your system and want to take all of your files from the PUB group to the new account. How can you transfer these files with a minimum of effort?
- *Scenario 2:* Your department is going to be on an HP 3000 MPE V system not connected to your present system. How can you move files in your account safely and efficiently to this new system?
- *Scenario 3:* Now that the MIDAS project is complete, system operations would like you to remove the MIDAS files from your system in order to conserve disk space. You know that the MIDAS

project will be resumed in one year. How can you store the files on tape now, and then reload them on disk when the project resumes?

Sooner or later you will encounter similar situations where you will need to store files on tape for immediate or future use. The most efficient way to store files for possible future use or to transport files between accounts is by using the STORE/RESTORE utility.

Store utility

The STORE utility is used to copy files from disk to magnetic tape. The primary purpose is to back up files in the event of a system failure or to transfer files to other locations.

Note

Although there are other methods to copy disk files to tape, you should always use the STORE command for backup and transfer between locations.

The STORE command does the following:

- copies files in a format that is optimal for backup and transport
- creates a directory of files on the tape

Using STORE

After you have logged on to your account, you can enter the commands to store files to tape from your terminal.

Note

Call system operations prior to completing the exercises in this lesson to get their cooperation in loading tapes for you. Otherwise, complete the exercises with pencil and paper.

Step 1 Send a message to your system operator asking that a blank tape be mounted.

```
TELLOP PLS MOUNT TAPE.  REPLY WHEN MOUNTED.
```

Step 2 When the tape has been mounted, enter the following file equation: (Use "T" as the file designator for the tape or create your own tapename.)

```
FILE T;DEV=TAPE
```

Step 3 Enter the STORE command followed by the name of the file(s) to be copied to the tape. Backreference (*) the tapename.

```
STORE filename;*T
```

Consider how this works for scenario 1. You want to move all the files from the PUB group to another account on the same system. Your system operator has mounted a tape for you, and you enter the following in order to initiate the STORE process:

```
FILE T;DEV=TAPE
STORE @.PUB.ACCTx;*T
```

If the STORE is completed successfully, here's what you should see on your display:

```
>> TURBO-STORE/RESTORE  VERSION  A.21.01  HP30319A <<
 (C) 1986 HEWLETT-PACKARD CO.
(TODAY'S DATE, TIME)

FILES STORED :                              19
```

Q4-5 How would you change the preceding STORE command to copy all the files in your account to a tape as in scenario 3?

Transporting files to a new system

Sometimes it is necessary to use STORE in order to transport files to another HP 3000 system.

Check with that system operations personnel for the following information:

Tape Drive Density — Any density other than the default (6250) should be specified as:

```
FILE T;DEV=TAPE;DEN=density value
```

System Type — If the destination system is not an MPE/iX system, the TRANSPORT parameter should be added to the STORE command.

```
STORE @.@.ACCTx;*T;TRANSPORT
```

Now consider scenario 2—transferring files from an MPE/iX system to an MPE V system. You find that the tape drives for the new system require a 1600 bpi tape format. Therefore, your file equation should look like this:

```
FILE T;DEV=TAPE;DEN=1600
```

The new system is an MPE V which means you need to add the TRANSPORT parameter to your STORE command to ensure a readable tape format:

```
STORE @.@.ACCTx;*T;TRANSPORT
```

Q4-6 Suppose you only need to store those files beginning with the letter M for transfer to an MPE V system. How would you write the STORE command to do this?

Note

It is a good practice to note on the tape label any special options used when creating the tape using STORE. This may save you time when you restore the files. If you used a special density in the file equation or the TRANSPORT option, write these on the tape label.

STORE parameters

Besides the TRANSPORT parameter, the STORE command has a number of other parameters, which allow you to copy different sets of files as well as monitor the progress of the STORE process.

Go into the Help Facility now and list the STORE parameters to your screen. Your awareness of the additional capabilities of this command is important for future use.

Hint

```
HELP STORE PARMS
```

Several of the parameters are discussed below. For others see the Help Facility or the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) .

PROGRESS

```
STORE @.@.ACCTx;*T;PROGRESS=5
```

When storing a large number of files to a tape, use the PROGRESS parameter to monitor the process. The STORE command above will display a progress message every five minutes.

SHOW

```
STORE @.@.ACCTx;*T;SHOW
```

The SHOW parameter prints a message on your terminal screen every time a file is stored. In the event of problems, the SHOW parameter displays an error message. You can direct the SHOW listing to another device if you choose. For example, to include a printout of file names with the tape, use:

```
STORE @.@.ACCTx;*T;SHOW=OFFLINE
```

PURGE

```
STORE @.@.ACCTx;*T;PURGE
```

The PURGE parameter erases the original file from the disk as soon as it is copied onto tape.

Be Careful

Use caution with the PURGE option. If the tape is defective the data is lost.

Using RESTORE

The RESTORE process essentially reverses the STORE process. Consider scenario 3—reloading files from a STORE tape onto your system.

To restore files from a tape to your account, you would give the tape to your system operator to be mounted, and enter the following at your terminal:

Step 1 Enter the same file equation entered for the STORE process.

```
FILE T;DEV=TAPE
```

Step 2 Enter RESTORE followed by a backreference to the tape name you just defined, followed by the file name(s) to be restored.

```
RESTORE *T;@.@.ACCTx
```

Q4-7	What RESTORE command would restore only the files from the PROJECT group of your account?

Lesson summary

1. To store files on tape, use a file equation to define the destination device. Use a STORE command so that the name(s) of the file(s) to be stored is backreferenced to the tape:

```
FILE T;DEV=TAPE
STORE filename;*T
```

2. Add the TRANSPORT parameter to your STORE command if the STORE tape will be reloaded on a system other than MPE/iX:

```
STORE @.ACCTx;*T;TRANSPORT
```

3. Add the DENSITY parameter to the tape file equation when storing files that will be restored on a tape drive with a different density.

```
FILE T;DEV=TAPE;DEN=1600
```

4. Add the SHOW or PROGRESS parameter to your STORE command to monitor the STORE process; add the PURGE parameter to delete files after they are stored only if they need to be purged.

```
STORE @.@.ACCTx;*T;SHOW
STORE @.@.ACCTx;*T;PROGRESS=3
STORE @.@.ACCTx;*T;PURGE
```

5. To restore files from tape, enter the file equation used for the STORE procedure followed by the RESTORE command.

```
FILE T;DEV=TAPE
RESTORE *T;@.@.ACCTx
```

Exercise 4-4: lesson 2 review

Now that you have taken the Advanced Skills Course, your manager has asked you to assist other members of the department to store files on tape. By Monday afternoon, a number of problems have come up. How many can you solve? (Assume that all the other members are working in the PUB group of the ADVUSER account.)

1. The use of the following equation and the STORE command gives the user an error message on the terminal screen. What's wrong?

```
FILE T;DEV=TAPE
STORE @.@.ADVUSER;T
```

2. Someone else has a problem trying to restore a tape made earlier today to another system where the user has an account. The destination system is unable to "read" the tape. What might be the problem?

3. A friend has asked you to help store and then restore some files from the PUB group of an ADVUSER account on an MPE/iX system to an ADVUSER account on an MPE V system.

The tape density of the current system is 6250 bpi; however, the tape drives for the new system will accommodate only 1600 bpi tape.

How would you write the file equations and the `STORE` and `RESTORE` commands to complete the process?

4. Another member of the department used the `PURGE` parameter when storing files on tape. Unfortunately, the tape has proven to be defective. Is there any way to recover the files?

********** End of Exercise 4-4 **********

6

Module 5: User Commands

Module 5 presents the following lessons on user commands, including command files and UDCs:

Lesson 1	Comparing UDCs and Command Files
Lesson 2	Understanding Search Priorities
Lesson 3	Cataloging UDCs
Lesson 4	Using UDC Options
Lesson 5	Using UDC Parameters
Lesson 6	Using the Command Interpreter

Challenge Test

1. TRUE OR FALSE:

   To execute a command file, you must enter its file name.

2. TRUE OR FALSE:

   To execute a UDC, you must enter its file name.

3. What does the abbreviation "UDC" mean?

4. Which executes in the following situations: the UDC, the command file, or the system command?

   a. a UDC and command file both having the same name

   b. a command file and system command both having the same name

5. What command would you use to display a particular UDC in a given UDC file?

6. What catalog command must you enter before you attempt to edit a UDC file?

7. What catalog command can you enter to add a UDC file to the UDC directory?

8. What UDC option prevents a UDC from being executed within a program?

9. What UDC option lets you reference one UDC within another, even though the referenced UDC is cataloged *before* the other UDC?

10. Examine the following UDC and then answer the questions.

```
***
PR
PARM FILE1=MONEY,FILE2="$NULL",FILE3="$NULL"
PRINT !FILE1
PRINT !FILE2
PRINT !FILE3
***
```

a. What happens when you enter this command:

```
PR
```

b. What happens when you enter this command:

```
PR DOLLARS,CENTS
```

11. What does the following UDC do when you enter this? `LSF FILEX,2`

```
UDC:

LSF MYFILE1,OPTION
LISTFILE !MYFILE1,!OPTION
```

12. What benefit does the CI provide you when developing programs and command files?

Lesson 1 UDCs and Command Files

Introduction

Lesson 1 presents a review of command files and an introduction to UDCs. You will learn how to:

- distinguish between command files and UDCs
- print a UDC file and a UDC
- ensure that a UDC goes into effect at logon

Note

In order to complete the exercises in this module, you need to be in the CLASS group of your account.

Review of user commands

"User command" is a term for a user-specified short-hand command that causes one or more MPE/iX commands to execute. Here are some samples of user commands. The names of the user commands are the choice of the user.

```
User          MPE/iX
Command       Commands

LF--------->LISTFILE
PR--------->PRINT
SM--------->SHOWME
SJ--------->SHOWJOB
SHOW------->SHOWME & SHOWJOB
```

As you know from previously creating command files, their main purpose is to save users time and keystrokes when executing MPE/iX commands.

Types of user commands

There are two types of user command: command files and user- defined commands, called UDCs.

Both command files and UDCs have the following features:

- contain one or more commands that MPE/iX can execute
- are useful for creating and executing frequently used commands or command sequences that are long or complex
- utilize parameters and options
- can be easily changed to the other's format
- may invoke other files
- can be created with a text editor

UDCs, however, are sophisticated command files that have some added features and differ in various ways from command files:

- need to be stored in a UDC file (which in turn is cataloged)
- can use two options not functional in command files
- execute when their command header is entered

Comparison of command files and udcs

Consider how the command files listed previously would appear as UDCs in a UDC file, MYUDC1.

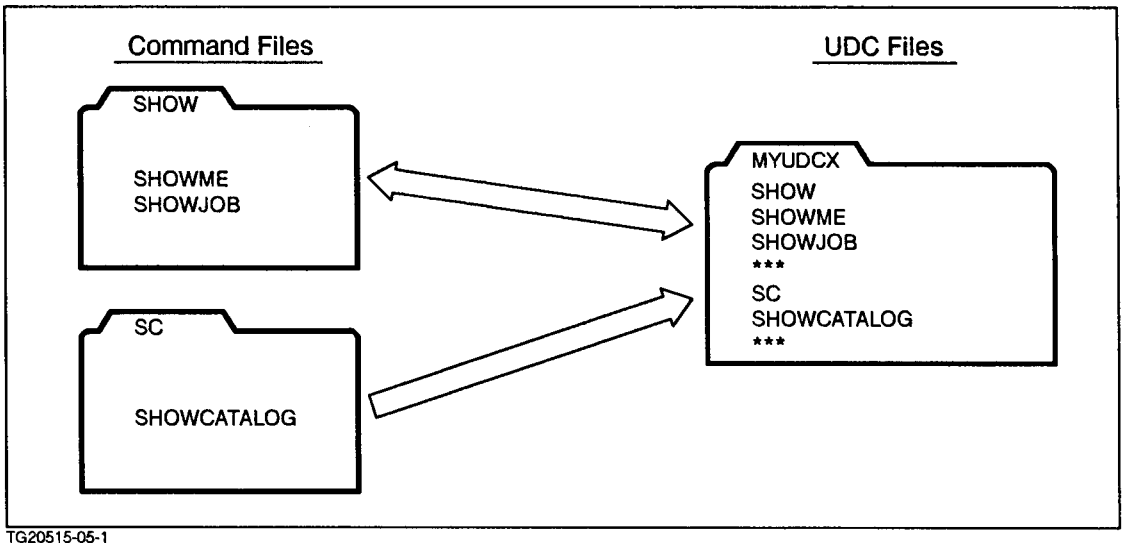


Figure 6-1. Command Files vs UDC File

Some of the differences you will notice are:

- Command file names (SHOW and SC) are not part of the command file.
- UDC names (SHOW and SC) are part of their respective UDCs and are referred to as command header names. UDCs are invoked by their command headers.
- Individual UDCs are separated by asterisks within a UDC file. Asterisks must begin in column one.

Why use a command file for user commands?

There are many user benefits to using a command file for user commands. First, command files are immediately executable after being kept. UDC files must be cataloged before they will execute.

Second, command files can be easily modified and, if necessary, purged. UDC files must first be uncataloged to be modified, and then recataloged to be executable again. UDCs cannot be purged while they are cataloged.

Third, because command files are easy to use, many users create their UDCs initially as command files. After they are sure that the file "works," users transform the command file to a UDC file.

Why use a UDC file for user commands?

Despite the added effort to create and catalog UDC files, they provide some features unavailable with command files.

One advantage is the order of execution. Commands stored in a UDC file execute before those stored in a command file. (You will learn about search and execution priorities in Lesson 2.)

Another important advantage is that UDCs can use a logon option that allows the UDC to execute immediately upon the user's logon.

Yet another advantage of UDCs is that a UDC file is opened only once, and then any of the UDCs within it are available for access at any time. (In the case of command files, every time you call a command file, it must be opened.)

UDCs use less disk space than the same number of individual command files. They may also provide better organization of user commands since they are all contained in the same file. Finally, UDC files have an additional safety feature—they cannot be easily purged accidentally.

Listing UDC files

Consider the UDC files in your account. UDC files must be cataloged in order for the UDCs to execute. This is done with the SETCATALOG command. All cataloged files are displayed with the command:

```
SHOWCATALOG
```

Enter that command now from your keyboard.

```
SHOWCATALOG
```

Do you get the following listing of your UDC files? You should see the fully qualified file name for MYUDC1 and these UDCs:

```
MYUDC1.CLASS.ACCTx
   STARTUP          USER
   SETEQ            USER
   SHOWCAT          USER
   SM               USER
   SJ               USER
```

Reviewing a UDC file

Study the UDC file, MYUDC1. To list the contents of a UDC file, enter:

```
PRINT udcfilename
```

Do that now for MYUDC1.

```
PRINT MYUDC1
```

Here's what you should see on your screen:

```
STARTUP
OPTION LOGON
SHOWJOB
SETEQ
***
SETEQ
FILE IN=$STDIN
FILE OUT=$STDLIST
FILE OUTPUT=TESTFILE;REC=-80,,F,ASCII
FILE MAILPRNT;DEV=LP;ENV=ELITE.HPENV.SYS
FILE SLLIST;DEV=TAPE
***
SHOWCAT
```

```
SHOWCATALOG
***
SM
SHOWME
***
SJ
SHOWJOB
***
```

As mentioned earlier, UDC files may contain one or more UDCs with the individual UDCs separated by asterisks. (Remember? No asterisks are included in command files.)

How many UDCs are in MYUDC1?

If you said five, you're right. STARTUP, SETEQ, SHOWCAT, SM, and SJ are the five UDCs contained in MYUDC1.

UDC characteristics

Each UDC begins with its command header or UDC name, for example, STARTUP, followed by commands to be executed. (Command files are executed by their file name; UDCs are executed by their command header.)

Logon UDCs

How could you ensure that a UDC would go into effect automatically when you log on?

If you insert a line OPTION LOGON following the command header, the UDC executes when you log on.

The logon option is useful for displaying information regarding your current session, specifically the number of jobs and sessions currently running on your system.

UDC files can use OPTION LOGON, but command files cannot.

Which of the UDCs in the MYUDC1 file goes into effect when you log on? If you log on again, you should notice that the SHOWJOB command automatically executes.

How to list a UDC

There may be times when you want to look at a UDC to find out which commands are included. But what if you don't know under which UDC file name the UDC was cataloged? Then you can use the HELP command to list the contents of individual UDCs. For example, to list the contents of a UDC called SETEQ, you would enter:

```
HELP SETEQ
```

UDCs can also be used to set up file equations. By including your most common file equations in a UDC, you can avoid retyping the equations every time you want to use them. Instead, you only need to execute that UDC.

Lesson summary

1. UDCs and command files share a number of features; however, unlike command files, UDCs are stored in a UDC file, which must be cataloged before the UDCs may execute.
2. Use the following commands to get information about UDC files and UDCs stored in your account:

`SHOWCATALOG`	Displays cataloged UDC files
`PRINT udcfilename`	Displays contents of a UDC file
`HELP udcname`	Displays contents of a UDC

Exercise 5-1: lesson 1 review

1. Check your knowledge of the characteristics of command files and UDC files by marking which characteristics belong to each:

	Characteristics	Command Files	UDC Files
a.	Must be cataloged.		
b.	Uses commands and parameters.		
c.	Command definitions must be separated by asterisks.		
d.	Can easily be converted to the other.		
e.	Executes the `LOGON OPTION`.		
f.	Can be created with a text processor.		
g.	Invoked by a command header.		
h.	Invoked by a file name.		

2. A sample UDC file and command file are listed below. See if you can answer the following questions regarding each of the files:

`MYCOMM1.CLASS.ADVUSER`

```
SHOWME
SHOWJOB
SHOWTIME
```

`MYCOMM2.CLASS.ADVUSER`

```
BEGIN
OPTION LOGON
SHOWJOB
*****
SM
SHOWME
*****
```

   a. Which file is a command file?

   b. Which file is a UDC file?

   c. If both files are in your account, which MPE/iX command(s) execute when you log on?

d. Which of the following are command headers in the example above?

```
SM
SHOWCAT
MYCOMM2
MYCOMM1
BEGIN
SHOWME
```

3. Select the correct command to do each of the following:

	Function	HELP UDC Name	SHOW CATALOG	PRINT Filename
a.	List all UDC files currently cataloged for an account.			
b.	List the contents of a specified UDC.			
c.	List the contents of a specified UDC file.			

********** End of Exercise 5-1 **********

Lesson 2 Understanding Search Priority and Search Path

Introduction

This lesson presents information on the following topics:

- MPE/iX search priorities
- the use of the `XEQ` command

To be sure that your user commands execute, you need to be aware of how MPE/iX interprets what is entered at the prompt.

How does MPE/iX search for UDCs, commands, programs, and command files? How does MPE/iX know whether you have entered a command or a UDC, for example? What is printed if what you have entered is not found in the search path?

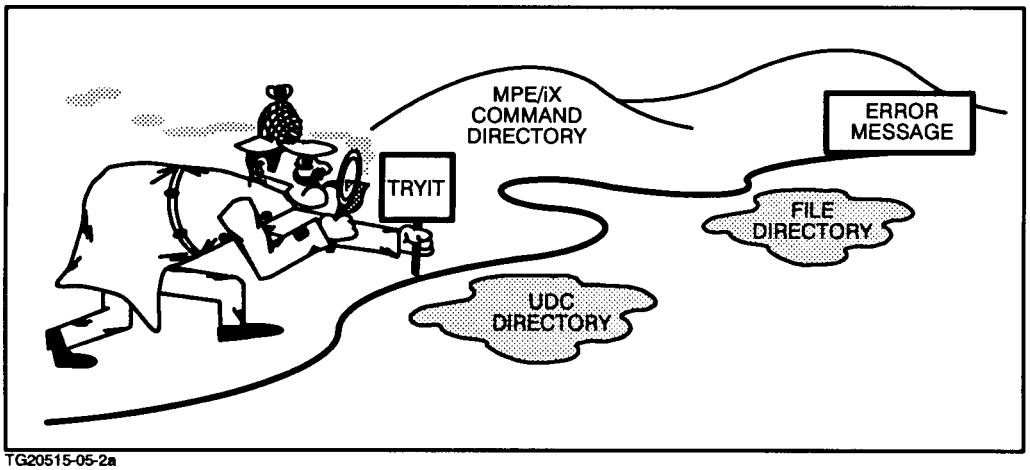


Figure 6-2. MPE/iX Directory Search

MPE/iX interprets everything entered at the prompt to be a command. It prioritizes its search for that command in the following order:

- UDCs first
- MPE/iX commands second
- file names (including command files and programs) last

This hierarchy is called the *search priority*.

The following detailed diagram illustrates the process MPE/iX uses to search for a command called `TRYIT`.

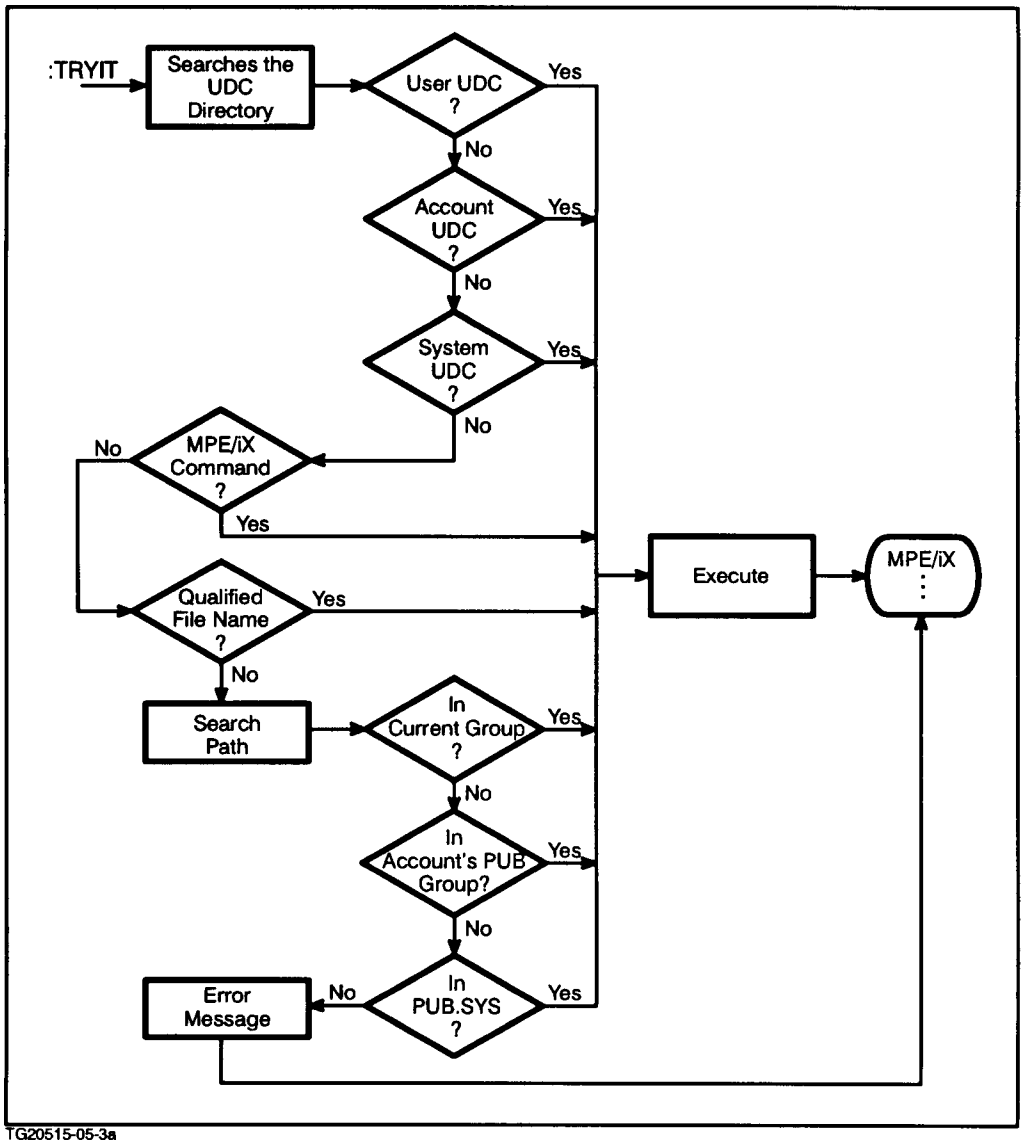


Figure 6-3. MPE/iX Search Priority

UDC directory

Suppose you were to enter the following at the MPE/iX prompt:

```
TRYIT
```

The system would first search for `TRYIT` in the directory of UDC files. (This directory is where all cataloged UDCs are kept.) User created UDCs are searched first, account UDCs next, and system UDCs last. If `TRYIT` were a UDC, the search would end and `TRYIT` would be executed.

In order to put a UDC in the UDC directory, it must be part of a UDC file that has been cataloged with the `SETCATALOG` command. `SETCATALOG` is described in the next lesson.

Q5-1	Which UDC in each of the following pairs would execute? 1. the user UDC `SM` or the system UDC `SM`? 2. the system UDC `LF` or the account UDC `LF`? 3. the account UDC `TRYIT` or the user UDC `TRYIT`?

Command directory

But what if `TRYIT` were not found in the UDC directory? Then the search would continue to see if `TRYIT` were in the MPE/iX command directory If it were, the system would execute it.

Q5-2	Suppose one of your cataloged UDCs has the same name as an MPE/iX command. If you enter the name, will the UDC or the MPE/iX command execute?

File directory

If `TRYIT` were not found in the command directory, the search would continue on to the file directory. If `TRYIT` were a qualified file name of a valid executable file, for example, a command file, `TRYIT` would execute.

Q5-3	Suppose you have both a command file and a cataloged UDC with the same name, MYFILEX. When you enter MYFILEX at the prompt, which of these will execute?

Search path

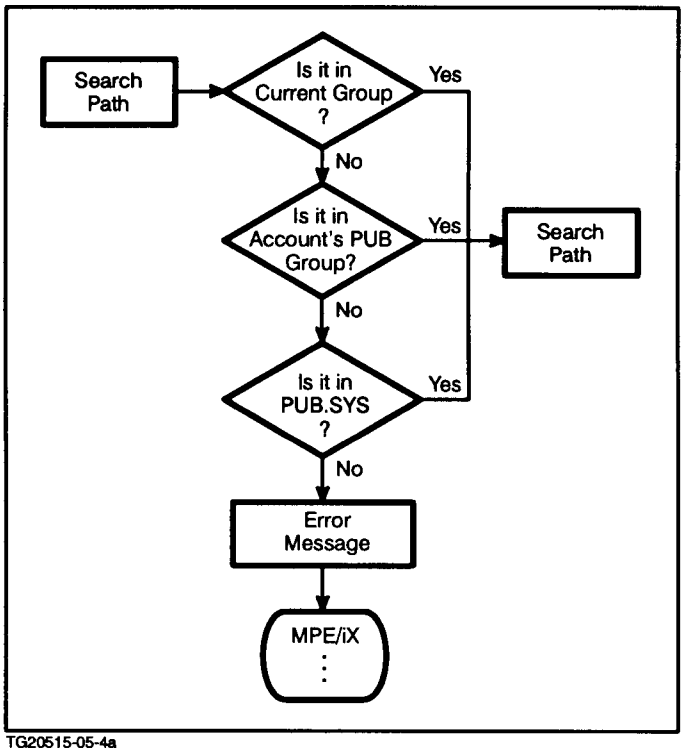


Figure 6-4. Search Path

Since the file name TRYIT has not been qualified, the system follows a designated search path looking for a matching name.

By default, the system first looks in your current group, next in your PUB group, and finally in PUB.SYS.

If TRYIT were not found in the search path or were found to be a nonexecutable file, the system would issue an Error Message and return you to the MPE/iX prompt.

```
UNKNOWN COMMAND NAME (CIERR 975)
```

XEQ command

Occasionally you may *want* to have the system execute a command file or program that has the same name as a UDC. The UDC always executes first unless you precede the name of the command file or program with the XEQ command.

The XEQ command ensures execution of valid command files and programs despite name duplications. XEQ plus the name of a program causes that file to execute.

For example, suppose you had a UDC and a program file with the same name, MYFILE.

Entering

```
MYFILE
```

executes the UDC.

Entering

```
XEQ MYFILE
```

ensures that your program file MYFILE executes.

Lesson summary

1. MPE/iX interprets everything entered at the prompt to be a command and prioritizes its search for that command as follows:
   a. UDCs first
   b. MPE/iX commands second
   c. file names last
2. Use the `XEQ` command to ensure that the system executes a command or program file having the same name as a cataloged UDC.

Note

`XEQ` also causes your command file to override any system command of the same name.

Exercise 5-2: lesson 2 review

Use the search path diagram in figure 6-3 as a guide. Check your understanding of search path and execution priority. Which executes?

1. the `LISTFILE` system command or a command file named `LISTFILE`?
2. a command file called `DELFILE` in your PUB group or a program called `DELFILE` in your current group?
3. a program called PRINT or the MPE/iX command `PRINT`?
4. `XEQ MYLIST`, a command file, or a UDC called MYLIST?

********** End of Exercise 5-2 **********

Lesson 3
Cataloging UDCs

Introduction

Lesson 3 covers the following topics dealing with cataloging UDCs for use:

- cataloging process
- SETCATALOG command
- APPEND option of SETCATALOG
- DELETE option of SETCATALOG
- uncataloging UDCs

Reviewing UDCs

Before you begin cataloging UDCs, see how much you remember.

What command lists the currently cataloged UDCs and the files in which they reside?

```
SHOWCATALOG
```

What is the difference between a UDC and a user command file?

They are both inherently the same except that the user command file has no header line, and is invoked by its file name. The UDC is invoked by the name listed in its header. Also, a UDC must be part of a UDC file that is cataloged in the UDC directory.

Creating or modifying a UDC

To create a UDC, you must first create a UDC file with an editor. Then you can add to it the commands that make up your UDC. (Just make sure all UDCs are separated by asterisks ***.) Another way to create UDCs is to modify or add to the commands in an existing UDC file.

Anytime you modify any existing UDC file, you must first uncatalog all UDC files by issuing the command SETCATALOG (without any file name following it). You can then use the text editor to modify the file. After you have saved the file, you must use SETCATALOG again, but this time specify the UDC file you are cataloging. You can then use the UDCs within it.

Note

Typically, you create a command file first to test your user command. When it works as planned, you may wish to add it to an existing UDC file. Command files are easier to modify than UDCs, so in an environment where change is common, you may wish to use command files instead of UDCs.

For example, suppose you want to create a UDC that lets you change your group to PUB by entering:

```
CHPUB
```

You might first test the command by creating a command file named CHPUB. Create such a command file with only one line:

```
CHGROUP PUB
```

Now execute the CHPUB command file. Do a SHOWME to verify that you are in the PUB group.

Change back to your CLASS group. You can use the MPE/iX CHGROUP command, or create another command file named CHCLASS to do this. Enter:

```
CHGROUP CLASS
```

Since the CHPUB and CHCLASS commands work, you can add them to one of your UDC files. Do a SHOWCATALOG to see what UDC files are currently cataloged. To add the command to MYUDC1, you have to uncatalog MYUDC1. To do so, type:

```
SETCATALOG
```

Do a SHOWCATALOG to verify that no user UDC files are cataloged. (System UDC files remain cataloged).

Note

Entering a SETCATALOG command without any file name uncatalogs all your UDC files. This means that the UDC files are removed from the UDC directory. They are not purged from the system!

Invoke an editor, and edit the MYUDC1 file to add the CHPUB and CHCLASS UDCs:

```
CHPUB
CHGROUP PUB
***
CHCLASS
CHGROUP CLASS
***
```

Keep the UDC file and purge the old CHPUB command file. Now try to execute the CHPUB UDC. Does it work? Of course not! You haven't yet recataloged the UDC file of which CHPUB is a part.

Exit the editor and enter:

```
SETCATALOG MYUDC1
```

Now reexecute the CHPUB command. Are you back in the PUB group? Of course!

Use the CHCLASS command to return to your CLASS group.

Cataloging and uncataloging UDC files

As you may have noticed, before a UDC can be used, the UDC file in which it resides must be cataloged in the UDC directory. The SETCATALOG command syntax is quite simple:

```
SETCATALOG udcfile1, udcfile2, ... udcfilex
```

When you want to uncatalog your UDC files (the ones which are currently cataloged), the SETCATALOG command syntax is even simpler:

```
SETCATALOG
```

Just leave off the names of the UDC files, and all of the UDC files will be uncataloged. This allows you to edit any of your UDC files and then to catalog them when you're ready. Remember, uncataloging

only removes the UDC files from the UDC directory; it does not purge them from the system.

Do a SHOWCATALOG to verify that no UDC files are currently cataloged. Now catalog only the MYUDC2 file:

```
SETCATALOG MYUDC2
```

Adding UDC files

When using the SETCATALOG command you must specify *all* the UDC files you wish to put in the UDC directory. Each time you use the SETCATALOG command, any UDC files specified replaces those currently in the UDC directory. However, sometimes you may wish to catalog only one UDC file and then add others later. What can you do?

You can use the APPEND option:

```
SETCATALOG udcfile;APPEND
```

To illustrate this option, type:

```
SETCATALOG MYUDC1;APPEND
```

Do a SHOWCATALOG to verify that MYUDC1 and MYUDC2 are cataloged.

Whenever you catalog UDC files, remember to use the group name for a UDC file if you try to catalog it from a different group.

Notice that MYUDC2 has a special UDC that lets you append UDC files: SETCATA. All you need to do is specify the UDC file you wish appended after the name, SETCATA. Try this command now:

```
SETCATA MYUDC3
```

Verify the effect of this UDC by using SHOWCATALOG.

MYUDC1, MYUDC2 and MYUDC3 should all be cataloged.

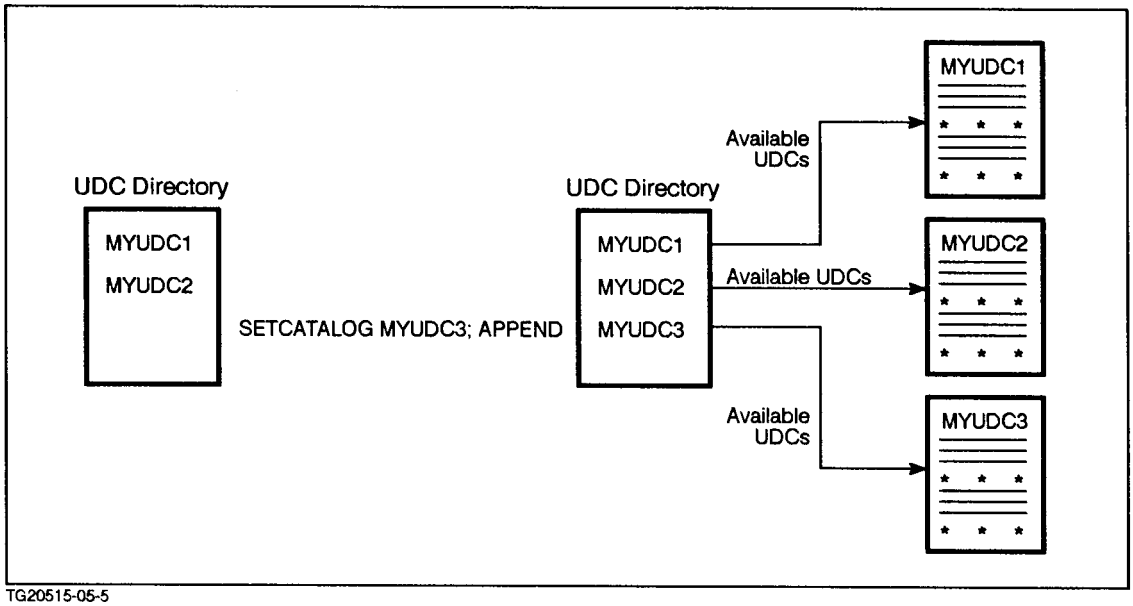


Figure 6-5. SETCATALOG APPEND Option

Deleting UDC files

`SETCATALOG` without parameters removes *all* of your UDC files from the UDC directory. Suppose you want to remove and edit just one of your UDC files?

Use the `;DELETE` option of the `SETCATALOG` command.

```
SETCATALOG UDCfilename;DELETE
```

You may remove one or more UDC files from the catalog. To remove more than one, use this form:

```
SETCATALOG UDCfilename1, UDCfilename2...;DELETE
```

When you remove one or more UDC files from the catalog, the remaining files stay in the catalog, undisturbed.

Q5-4 If the files `TOKYO`, `LONDON`, `DALLAS`, and `LIMA` were cataloged UDC files in the UDC directory, how would you do the following?

a. delete all the UDC files from the UDC directory?

b. add the UDC files `MOSCOW`, `NAIROBI`, and `BRASILIA` to the UDC directory without affecting other cataloged files?

c. replace existing UDC files in your UDC directory with the following files: `NEWYORK`, `BOSTON`, `BOMBAY`, and `ZURICH`?

d. remove `BOSTON` and `BOMBAY` from the UDC directory without affecting any other UDC files.

Lesson summary

1. The following commands are used when dealing with UDC files:
   - `SETCATALOG UDCfilename` - catalogs one or more UDC files. (These files replace whatever files were previously cataloged.)
   - `SETCATALOG` (Return) - uncatalogs all UDC files.
   - `SETCATALOG UDCfilename(s);APPEND` - catalogs one or more UDC files by adding them to the UDC directory. (Files which were previously cataloged are still cataloged).
   - `SETCATALOG UDCfilename(s);DELETE` - uncatalogs one ore more UDC files by removing them from the UDC directory (the remaining files remain cataloged).
2. Before you edit a UDC file, you must uncatalog it.

Lesson 4
Using UDC Options

Introduction

Lesson 4 presents the following options associated with UDCs:

- LOGON/NOLOGON
- LIST/NOLIST
- HELP/NOHELP
- RECURSION/NORECURSION
- PROGRAM/NOPROGRAM''
- BREAK/NOBREAK

These options affect the way that the UDC is used and the way that it acts. Typically, the option must be specified in the line directly after the UDC header (name). Several of these options can also appear within the body of the UDC. (This is discussed later in this lesson.)

Note

Most of the options discussed appear in MYUDC1 and MYUDC2.

LOGON option

As you recall, the LOGON option causes a UDC to execute immediately upon logon. You need not invoke it by name.

MYUDC1 contains a UDC called STARTUP that uses this option. Notice how you automatically get a SHOWJOB display when you logon. OPTION LOGON is good for any UDCs that you want automatically executed at the beginning of your session.

Example:

```
STARTUP
OPTION LOGON
SHOWJOB
SETEQ
***
```

The default is OPTION NOLOGON.

Note

The LOGON/NOLOGON option must appear immediately after the UDC header on a separate line.

LIST/NOLIST option

This option specifies whether or not the MPE/iX commands that make up a specific UDC are displayed when the UDC is executed. If OPTION LIST is specified, each line of the UDC will be displayed, as well as the results of that line.

With the LIST option, the UDC command lines are displayed.

There are two UDCs in MYUDC2: ST1 and ST2. Both do the same thing, with only one minor difference -
the LIST/NOLIST option:

```
ST1
OPTION LIST
```

```
SHOWTIME
***

ST2
OPTION NOLIST
SHOWTIME
***
```

Verify that MYUDC2 is still cataloged. If it is not, please use SETCATALOG to do so.

Execute ST1 and ST2. What happens? Notice that with ST1, both the MPE/iX command (SHOWTIME) and its results are displayed on the screen. With ST2, only the results of SHOWTIME are displayed. ST1 has the LIST option, and ST2 does not.

Normally, you don't need a listing of the actual commands in the UDC. However, if you have a particularly complex UDC and wish to be reminded of the commands that compose it, use OPTION LIST.

The default is OPTION NOLIST.

Note

The LIST/NOLIST option can appear in the body of the UDC, as well as immediately after the header. This is unlike LOGON/NOLOGON, which can only appear immediately after the header.

HELP/NOHELP option

As you recall, you can use the HELP command to look at the description of a particular UDC in a UDC file. By using HELP, you do not have to PRINT the entire UDC file to view only selected UDCs. Type:

```
HELP SETCAT
```

You should see a description of the SETCAT UDC. Now type:

```
HELP SETCATA
```

Why don't you see a description? HELP only works with UDCs that are defined with the HELP option. To examine SETCATA, then, you must print the entire MYUDC2 file.

```
PRINT MYUDC2
```

Find the SETCATA UDC and notice that it is defined with the NOHELP option.

Unless security restrictions prevent you from allowing others to examine a UDC, you should avoid the use of the NOHELP option.

The default is OPTION HELP.

Note

The HELP/NOHELP option must appear immediately after the UDC header.

RECURSION/ NORECURSION option

Normally, in your UDC file you can use a UDC within the definition of another UDC, only if the one you are referencing appears *after* the one you are currently using. For example, this is a valid UDC sequence within a UDC file:

UDCFILE1:

```
LOGON
  command
  command
  command
**********
UDC1
  command
  UDC2
  command
**********
UDC2
  command
  command
  command
***********
```

UDC2 is called within UDC1, and that is acceptable because UDC2 is defined *after* UDC1 in UDCFILE1. What if you wanted to call UDC1 within UDC2? That would not be possible *unless* you used the RECURSION option.

Not only must you be concerned with the order in which UDCs are defined within one UDC file, you must be concerned with the order in which the UDC files themselves are cataloged. When you catalog your UDC files, UDCs are cataloged in the order in which the UDC file names were entered on the SETCATALOG command line.

For example, suppose you cataloged your UDC files in this order:

```
SETCATALOG UDCFILE1,UDCFILE2
```

By specifying the RECURSION option after the UDC header, you can reference any UDC in the UDC directory, even if it was defined in a UDC file that was cataloged prior to the current one. In the next example, you can see the RECURSION option used properly for UDC2 and UDC3: UDCFILE1: UDCFILE2:

```
UDC0                          LOGON
  command                       command
  command                       command
  command                       command
**********                    ***********
                              UDC1
                                command
                                command
                                UDC2
                                command
                              ***********
                              UDC2
                              OPTION RECURSION
                                command
                                command
                                UDC1
                                command
                              ***********
                              UDC3
                              OPTION RECURSION
                                command
                                UDC0
                              ************
```

The RECURSION option, as illustrated, ensures that UDC2 is properly executed, even though UDC1 precedes it in the file. The option also ensures that UDC3 is properly executed, even though UDC0 precedes it in the directory.

Note UDC0 is in UDCFILE1. UDCFILE1 was cataloged *before* UDCFILE2.

The RECURSION option is only effective for the UDC in which it is specified.

The RECURSION option on a UDC eliminates the need to put a UDC file in any specific order in the catalog. When RECURSION is used, the recursive UDC finds the other UDCs it calls, regardless of the order of the UDC file. Therefore, the order in which the UDC files are cataloged is not important.

Note It is good practice to put a comment in a UDC file to indicate whether or not it must be cataloged in a certain order with other UDC files because of recursion dependencies.

Exercise 5-3: RECURSION option

Study the following UDC file and indicate how you should change this file to have the UDC SO call upon and execute UDC ST without changing the order of the UDCs in the file:

```
ST
SHOWTIME
***
SM
```

```
SHOWME
***
SO
SHOWOUT
***
```

********* End of Exercise 5-3 *********

Because recursion allows two UDCs to call each other and enables a UDC to call itself, limitations have been put in place to prevent endless loops.

The maximum number of times that a UDC can call itself, or that two user commands can call each other is 30 total times. When that maximum is reached, the system interrupts the process with an error message. This is a system-imposed restriction.

Example:

Suppose you add the following UDC to `MYUDC2`, recatalog `MYUDC2`, and then execute `MYTIME`.

```
MYTIME
OPTION RECURSION
SHOWTIME
MYTIME
```

Notice that `MYTIME` executed 30 times!

The default is `OPTION NORECURSION`.

Note

The `RECURSION` option can appear anywhere in the body of the UDC, as well as immediately after the header.

PROGRAM/ NOPROGRAM option

This option lets you specify whether or not a UDC can be executed from within a program. If you specify `NOPROGRAM`, the UDC cannot be executed from within a program.

Consider the following UDC in the `MYUDC2` file:

```
TIME
OPTION NOPROGRAM
SHOWTIME
***
```

This means that if you called `TIME` from within a program or utility, such as VOLUTIL (a volume management utility), the following would happen:

```
volutil:  :TIME
UNKNOWN COMMAND NAME.  (CIERROR 975)
```

Notice that from the utility, you must enter the system prompt (:) before the UDC. You must do this so that the command interpreter can respond, even though you are in a subsystem.

Then exit VOLUTIL as follows:

```
volutil:  EXIT
```

The default is OPTION PROGRAM. However, even with OPTION PROGRAM, some subsystems and utilities won't recognize UDCs, depending on how they were programmed.

Note

The PROGRAM/NOPROGRAM option must appear after the UDC header and the PARM line.

Q5-5 Study the following UDC and indicate what you would do to this file to have the UDC LF *not* be executable from within a program:

```
LF
LISTFILE
***
```

BREAK/NOBREAK option

This option lets you specify whether or not the execution of a UDC will be "breakable". With the BREAK option specified, when you press (BREAK), execution of a command stops and cannot be resumed. However, if the command happens to invoke a subsystem such as the editor or FCOPY, you may use the RESUME command after pressing (BREAK) to continue execution.

For example, LISTFILE is a "regular" command. If you press (BREAK) while LISTFILE is executing, all output stops. You cannot use the RESUME command to continue its execution. On the other hand, if you are executing VOLUTIL commands (VOLUTIL is a subsystem) and press (BREAK), you will be put in "break mode."

This means that you can then execute other MPE/iX commands or RESUME or ABORT the subsystem.

If you specify the NOBREAK option, pressing (BREAK) will have no effect on either regular MPE/iX commands or subsystem commands.

To illustrate this option, create the following two command files, BREAK1 and BREAK2.

BREAK1 command file:

```
OPTION BREAK
LISTFILE @.CLASS.ACCTx,2
```

BREAK2 command file:

```
OPTION NOBREAK
LISTFILE @.CLASS.ACCTx,2
```

Now, execute BREAK1 and immediately press (BREAK). The listing should stop. Notice also that you *cannot* resume the listing by typing RESUME. (BREAK) effectively terminates the execution of BREAK1.

Do the same with BREAK2. What happens? The (BREAK) key should have no effect, and the files in the account should continue listing.

Note

The BREAK/NOBREAK option must appear immediately after the UDC header.

Position of options

By now you've probably noticed that most options appear on the lines following a UDC header, or on the first lines of a command file. There are two options that can appear both after the UDC header *and* in the UDC body. What are they?

```
OPTION RECURSION/NORECURSION
OPTION LIST/NOLIST
```

When an option appears within the body, it takes precedence over the option in the header.

Study the DOFILE example when answering the following question. It illustrates what happens when OPTION LIST appears in the header, and OPTION NOLIST appears in the body:

DOFILE UDC File:

```
DOIT1
OPTION LIST
SHOWTIME
OPTION NOLIST
LISTFILE
***
DOIT2
SHOWME
***
DOIT3
LISTFILE
OPTION RECURSION
DOIT2
***
```

Q5-6 DOFILE.CLASS is on your system. Add it to the user directory by cataloging it.

a. Execute DOIT1. What MPE/iX commands are displayed?

b. Execute DOIT2. What MPE/iX commands are displayed? Why?

c. Execute DOIT3. What MPE/iX commands are displayed? Why?

In DOIT1, OPTION LIST is "activated" in the UDC header and then "deactivated" in the UDC body when OPTION NOLIST is specified. This means that when you enter DOIT1, you will see the MPE/iX command, SHOWTIME, but you will not see the command, LISTFILE.

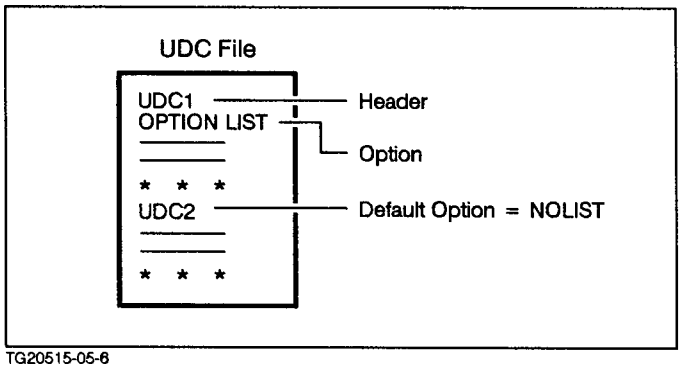


Figure 6-6. Default Options

Exercise 5-4: UDC options

1. Add a *single* UDC called SLS to the MYUDC2 file. Have the UDC do the following in the order given:

   a. Execute the SHOWTIME command without listing the SHOWTIME command on the screen.

   b. Execute the LISTFILE command without listing the LISTFILE command on the screen.

   c. Execute the SHOWME command and list the SHOWME command on the screen.

Note

Make sure that you uncatalog MYUDC2 before adding SLS to it.

2. What is wrong with the following UDC?

```
LISTER
OPTION NOLIST
FILE OUT;DEV=LP
PRINT MYFILE;OUT=*OUT
OPTION LIST
OPTION NOHELP
LISTFILE MYFILE,2
***
```

3. Put an X next to the correct default value assumed by the system when no option is specified in a UDC:

LIST	NOLIST
HELP	NOHELP
RECURSION	NORECURSION
PROGRAM	NOPROGRAM

********** End of Exercise 5-4 **********

Lesson summary

1. The following options control what you can do with your UDCs and command files (only `LOGON/NOLOGON` and `RECURSION` are valid *only* for UDCs):

   a. `LOGON/NOLOGON` - determines whether or not the UDC goes into effect automatically upon logon (default = `NOLOGON`).

   b. `LIST/NOLIST` - determines whether or not the commands making up the UDC, as well as its results, are listed when the UDC is executed (default = `NOLIST`).

   c. `HELP/NOHELP` - determines whether or not you can display the commands making up the UDC by using the `HELP` command (default = `HELP`).

   d. `RECURSION/NORECURSION` - determines whether or not you can reference a UDC within the definition of another, when the referenced UDC was defined *before* the other, either in the same file or elsewhere in the directory (default = `NORECURSION`).

Note

Command files can always be invoked recursively. The `RECURSION` option has no meaning, except for UDCs.

   e. `PROGRAM/NOPROGRAM` - determines whether or not you can execute a UDC within a program (default = `PROGRAM`).

   f. `BREAK/NOBREAK` - determines whether or not the execution of a UDC is be “breakable” (default = `BREAK`).

Lesson 5
Using Parameters

Introduction

Lesson 5 presents the following information about UDC and command file parameters:

- parameter definition.
- correctly using required and optional parameters.
- correctly specifying parameters in UDCs and command files.

Passing values by parameters

You may have noticed, when examining the `MYUDC2` file, that some UDCs have more than just a command name in the header. Some also have parameters. Up to now you've read about user commands (both UDCs and command files), which are composed solely of simple MPE/iX commands. To invoke such a user command, you simply enter one command name.

It is also possible for a user command to contain parameters, so that when you execute the command, you can enter the name of the UDC or command file, *followed* by one or more values. The values are passed to the parameters. Parameters act as placeholders in the command syntax and are replaced by user-supplied values upon execution. In a UDC, the parameters appear in *either* the command header, or on the `PARM` line directly preceding any options, but not in both places. In a command file, the parameters appear only in the PARM line, which is the first line of the file, directly preceding any options.

UDC:

```
Header
Parameters
Options
Body
```

Command File:

```
Parameters
Options
```

UDC Example:

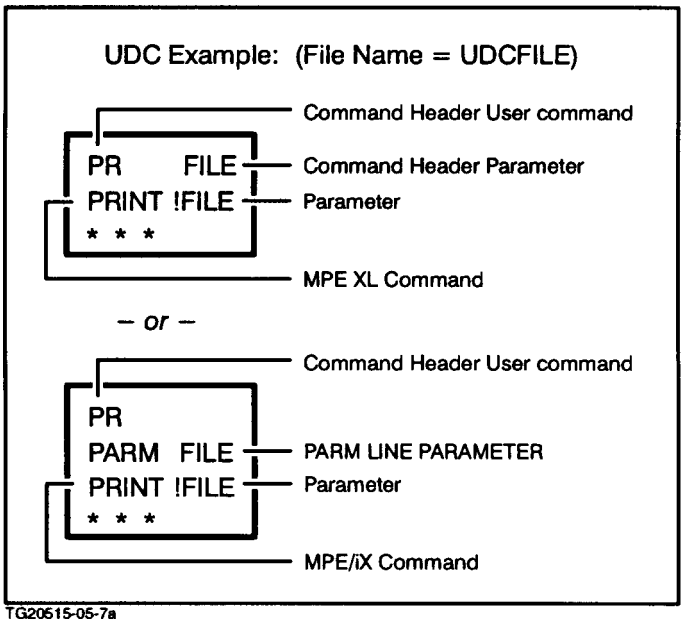


Figure 6-7. Parameters in a UDC

Command File Example:

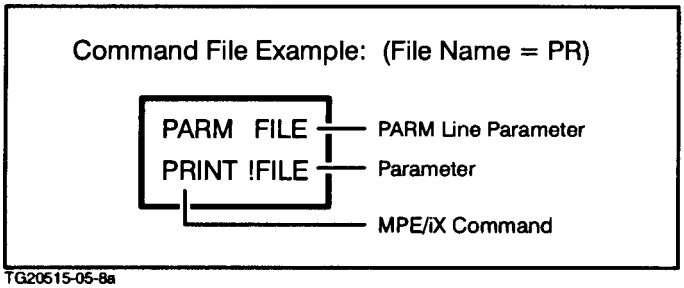


Figure 6-8. Parameters in a Command File

In these examples, `FILE` is the parameter. The user command name is `PR`. The syntax of this user command indicates that `PR` must always be followed by a value. This value will be some file name supplied by the user. The system knows that `FILE` is a parameter because of the PARM line. The system also knows that `FILE` is to be replaced with a user-supplied value, since there is an exclamation point (`!`) in front of the parameter name (`!FILE`).

The system command, `PRINT`, is executed when you supply a file name along with the `PR` user command:

```
PR JOB1
```

This actually executes the following system command:

```
PRINT JOB1
```

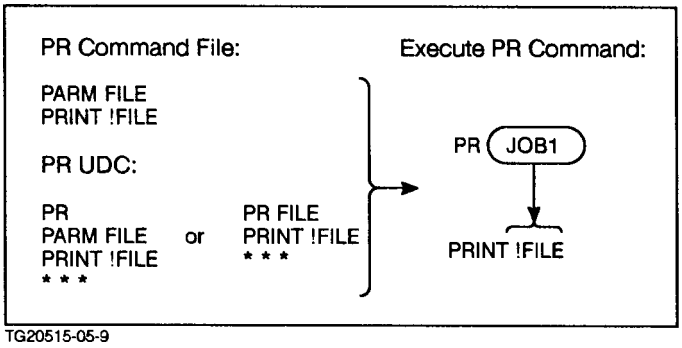


Figure 6-9. PR Command File

Try it and see!

What happens if you just type:

```
PR
```

You get an error because, according to the UDC definition, the system expects a value after the PRINT command. In this lesson you will learn how to avoid such errors by defining default parameter values.

What if you wanted to print several files? What happens if you type:

```
PR JOB1,JOB2
```

Once again, you get an error because, according to the UDC definition, only one parameter value is expected. You may write UDCs that allow more than one parameter. You will be examining their use later in this lesson.

Optional and required parameters

Parameters are either required or optional. If required, the user *must* specify values for the parameters when executing the user command.

Required parameters are those for which no default value is specified. Required parameters are specified by listing the parameter name without a default value in the UDC. This means that when the UDC is invoked, some value must follow the command name, if a required parameter is specified in the definition. You can now examine two different examples, the PR command file (which you've already seen), and the SETCAT UDC.

PR Command File Example

```
PARM FILE1
PRINT !FILE1
```

Explanation

The purpose of this command file is to print one file (FILE1.)

The exclamation point (!) tells the system that the file name that you specify when you execute the PR user command is be passed to the parameter, FILE1.

FILE1 is a required parameter. You cannot enter PR without specifying a value after it. What happens if you enter only PR? The system tells you:

```
FILE1 PARAMETER IS REQUIRED.
PARAMETERS DO NOT MATCH USER COMMAND DEFINITION.
```

SETCAT UDC Example:

```
SETCAT UDCFILE="MYUDC1"
SETCATALOG !UDCFILE
***
```

Explanation:

The purpose of this UDC is to catalog a specified UDC file.

The exclamation point (!) tells the system that when you execute the SETCAT user command, the file name that you specify is to be passed to the parameter, UDCFILE.

UDCFILE is an optional parameter. If you enter SETCAT without specifying a value after it, MYUDC1 is assumed to be the value, by default.

Note

Required parameters have no values assigned them in the UDC or command file. Optional parameters do.

First, use SHOWCATALOG to verify that both MYUDC1 and MYUDC2 are cataloged. If they are not, catalog them.

To test the default value of SETCAT, first uncatalog MYUDC1 so that only MYUDC2 is in effect:

```
SETCAT MYUDC2''
```

Now enter SETCAT without any file name after it and see what happens. When you do a SHOWCAT, MYUDC1 should be in effect again, not MYUDC2. Indeed, the system did execute this command:

```
SETCATALOG MYUDC1
```

Now, add MYUDC2 to the UDC directory and verify that both MYUDC1 and MYUDC2 are cataloged:

```
SETCATALOG MYUDC2;APPEND
SHOWCAT
```

Here you cannot use SETCATA because it is a UDC defined in the MYUDC2 UDC file. Once MYUDC2 is cataloged, however, you can use SETCATA again.

Optional parameters with sensible default values are useful if the user doesn't specify a value.

Multiple parameters

What if you want to use the PR command file to print multiple files? It is also possible to specify more than one parameter in a UDC or command file. This lets you enter additional values after the UDC name when executing the command. By doing so, you pass values to multiple parameters in one command.

Consider the following command file, PRX, and the UDC, SETCATX. Each is modified version of the PR command file and the SETCAT UDC.

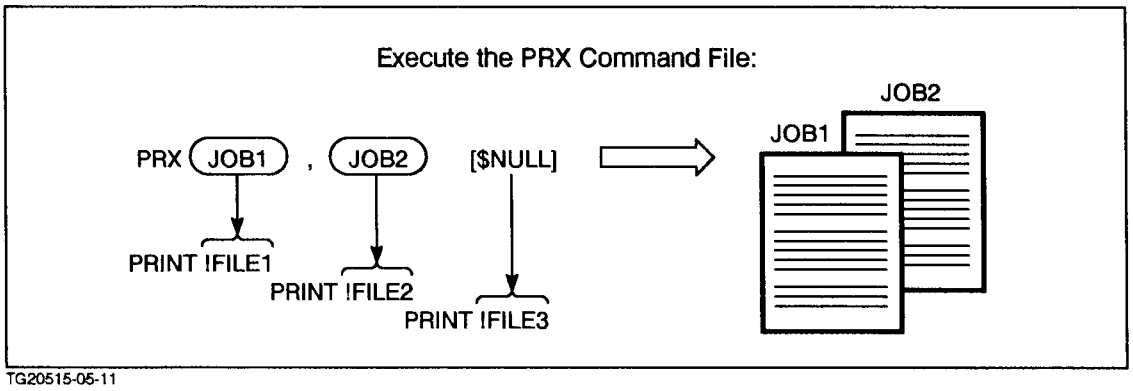


Figure 6-10. PRX Command File

PRX Command File:

```
PARM FILE1=$NULL,FILE2=$NULL,FILE3=$NULL
PRINT !FILE1
PRINT !FILE2
PRINT !FILE3
```

Explanation:

The command in the previous figure lets you print up to three files. All three parameters default to `$NULL` (the system designator for an empty file), so that only a `$NULL` value with no data is passed to them. In this example, assigning a parameter value of `$NULL`, no error message results if you forget to specify a value for that parameter. Without the `$NULL` default, you would get the message `THE xxx PARAMETER IS REQUIRED.`

To test how the `PRX` user command works with one or more values, enter the following:

```
PRX JOB1,JOB2
```

You should see a listing of each file and no error message.

What happens if you type only `PRX`? You shouldn't get an error message. Instead, nothing is printed (since `$NULL` is an empty file).

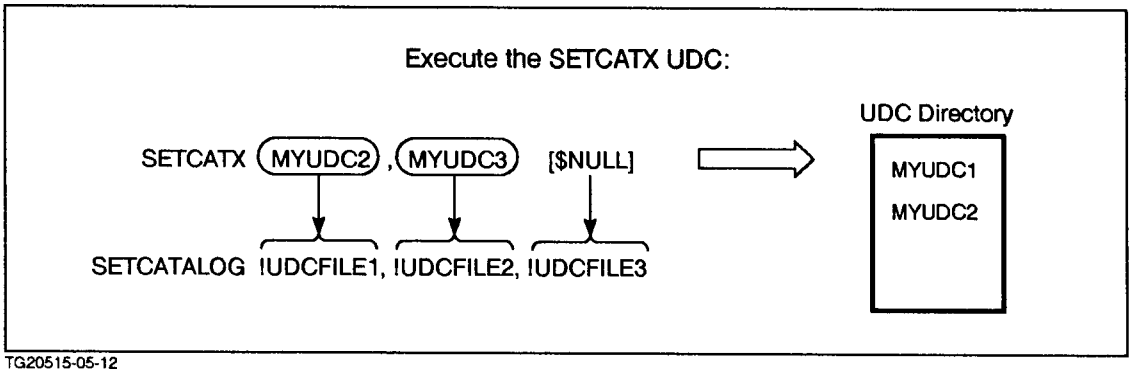


Figure 6-11. SETCATX UDC

SETCATX UDC (in MYUDC2 file):

```
SETCATX UDCFILE1="MYUDC1",UDCFILE2=$NULL,UDCFILE3=$NULL
SETCATALOG !UDCFILE1,!UDCFILE2,!UDCFILE3
***
```

Explanation:

The UDC in the figure above lets you catalog up to three files. The first parameter defaults to MYUDC1. The second and third parameters default to $NULL (empty file).

To test how the SETCATX UDC works with one or more values, start by cataloging only MYUDC2:

```
SETCATX MYUDC2
```

Do a SHOWCAT to verify that only MYUDC2 is cataloged. So, SETCATX works with only one value. How about three values?

Enter the following:

```
SETCATX MYUDC1,MYUDC2,MYUDC3
```

Do a SHOWCAT to verify that all three UDC files are now cataloged.

What happens if you type only SETCATX? Try it and see. You should not get an error message, since the first parameter (UDCFILE1) defaults to MYUDC1. Instead, MYUDC1 should get cataloged, since it is the default value. Notice also that all the other UDC files were "uncataloged."

Do a SHOWCAT to verify that *only* MYUDC1 is in effect.

Now recatalog MYUDC2 and MYUDC3, and then verify that all three UDC files are once again in effect:

```
SETCATALOG MYUDC2,MYUDC3;APPEND
SHOWCATALOG
```

Using multiple parameters and $NULL default parameter values in place of file names gives you flexibility in a user command. By doing so, you can enter less than the total number of parameters specified in the command without receiving an error message.

$NULL must not be used as a default value if you are supposed to pass a numeric value to a parameter. Use a number instead. In the example below, $NULL is the default value for a file name, whereas 2 is the default value for the numeric option.

Example:

```
LS FILE1="$NULL",OPTION="2"
LISTFILE !FILE1,!OPTION
```

Note

The quotation marks are not required.

Benefits of parameters

Putting parameters in your UDCs can provide a short-cut method for entering lengthy commands that change frequently. For example, suppose that you run a program that writes its results to *OUT. You must change the OUT file equation each time you run the program so that the output goes to one of three different files: OUTFILE1, OUTFILE2, and OUTFILE3. Consider the typing effort:

```
FILE OUT=OUTFILE1;REC=-80,,F,ASCII
FILE OUT=OUTFILE2;REC=-80,,F,ASCII
FILE OUT=OUTFILE3;REC=-80,,F,ASCII
```

Wouldn't it be nice to have a UDC that would let you change the destination easily? Here is how it's done.

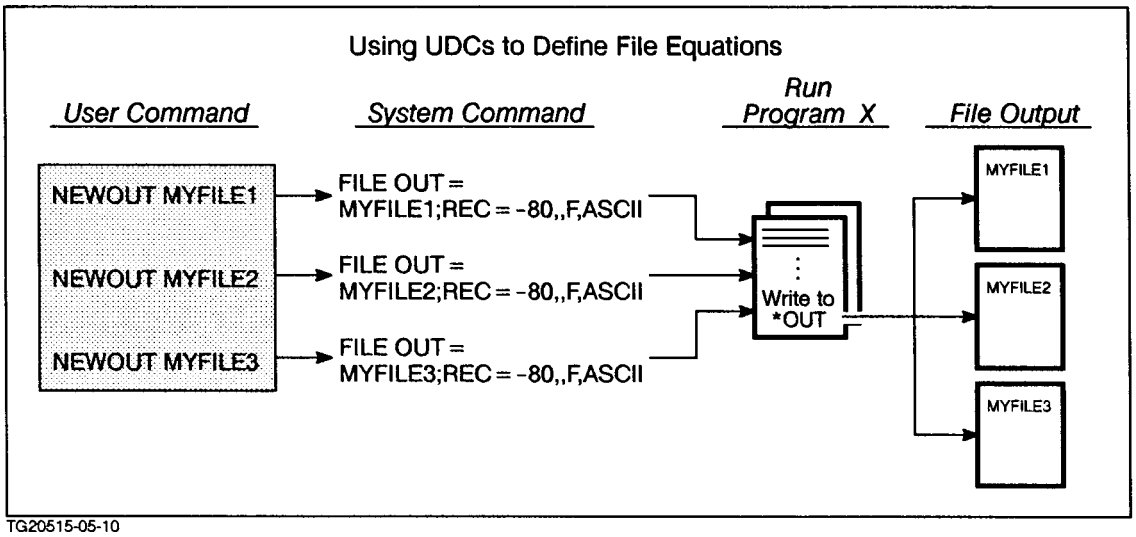


Figure 6-12. NEWOUT UDC

```
NEWOUT FILE
FILE OUT=!FILE;REC=-80,,F,ASCII
***
```

To change the output destination to OUTFILE1, you only have to type

```
NEWOUT OUTFILE1
```

Then, when you run the program, and it writes its output to *OUT, the output is directed to OUTFILE1.

Exercise 5-5: using parameters

1. Create a command file or UDC called LS that has the following characteristics:
   a. performs a LISTFILE,1 or LISTFILE,2 or LISTFILE,3 on a user-specified file
   b. the user *must* specify a file name; however, LISTFILE,1 will always be assumed unless otherwise specified Hint: Use the PARM keyword as the first line of the file, and make sure to specify a default value for the numeric LISTFILE option.

Note

When using parameters in a UDC, it's a good idea to create a command file first and test it, and then convert it to a UDC.

2. Test and execute `LS` on file JOB1 as follows:

   a. Enter

      `LS`

      without any numeric option value. What happens?

   b. Enter

      `LS`

      with parameter value 1; then with parameter value 2; then with parameter value 3.

3. When the `LS` command file is working, convert it to a UDC and add it to `MYUDC3`. This time, do not use the `PARM` line.

   HINT: Make sure to use `SETCATALOG` first so that `MYUDC3` is not in the user directory; otherwise, you will be unable to modify it.

4. Recatalog `MYUDC3` in the user directory. Then test and execute `LS`. Even though the command file and the UDC are the same, the UDC is executed because the system searches the user directory first.

********** End of Exercise 5-5 **********

Lesson summary

1. Optional (default) parameters should be used in a UDC or command file when it seems sensible that a user might not enter values for all the parameters.
2. Required parameters should be used in command files and UDCs when values must be specified with the UDC. Typically, required parameters come before optional ones.
3. In command files, parameters must be specified in the `PARM` line only (before options and before the body of the file).
4. In UDCs, parameters can be specified in the command header *or* the `PARM` line, but not in both.

Lesson 6 The Command Interpreter

Introduction

Lesson 6 presents two basic uses of the command interpreter:

- toggling between an editor and the operating system when compiling and linking a program
- toggling between an editor and the operating system when testing a command file

Note

This lesson is primarily for programmers and system managers. You can skip the lesson if you perform neither programmer nor system manager functions.

If you plan to write programs or command files, you will often use the editor to edit the file, keep the file, exit the editor, and return to the operating system to compile and run the program or execute the command file. This process can get tedious when you must do continual editing and error checking. It would be most convenient at such times to be able to toggle between the editor and the operating system; however, pressing the Break key or pressing CTRL Y does not allow you to toggle between the editor and the operating system, and actually execute any commands at the operating system level. So, what does? The *command interpreter* (**CI**).

Note

It is also possible to toggle by entering :*MPE/iX command-name* at the editor prompt.

In this lesson, you will learn to do the following:

- use the CI to toggle between the editor and the operating system when modifying source code or command files

Using the CI

The command interpreter (CI) is the program that acts as the interface between you and the operating system. It checks the MPE/iX commands that you enter for spelling and syntax errors. The CI then passes the command along to the appropriate system procedure for execution. Following execution, control returns to the CI, which becomes ready for another command.

You can run the CI from within the editor and then enter the MPE/iX commands necessary to compile, link, and run a program. Then, by entering `EXIT`, you can immediately return to the editor. You can also run the CI from within the CI.

Note

To run the CI, either the user *or* the program must have PH capability. The editor does not have PH capability (confirmed earlier with the `VERSION` command), so the user must have PH capability. To use nested levels of the CI, the user must have PH capability.

You can enter SHOWVAR HPCIDEPTH to learn your CI level. When CI is run from the editor, HPCIDEPTH=2. The main CI level (root) is HPCIDEPTH=1.

To run the CI, enter the following command at the editor prompt (/):

```
:RUN CI.PUB.SYS
```

or

```
:XEQ CI
```

To exit the CI and return to the editor, enter:

```
EXIT
```

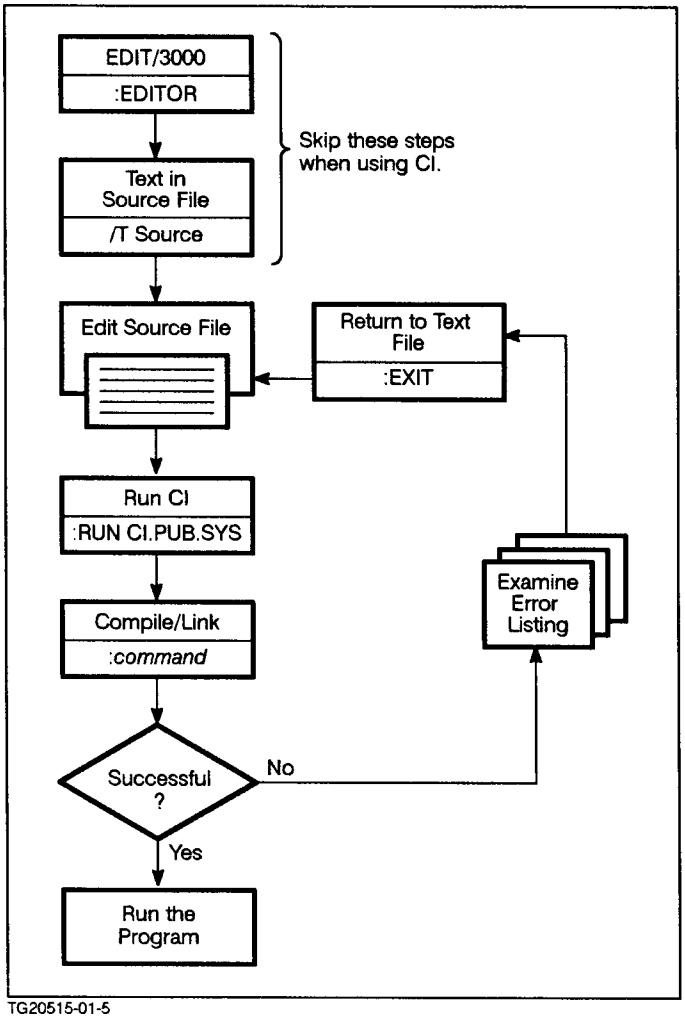


Figure 6-13. The CI

To demonstrate the assistance that the CI can provide you during program development, follow along with the teaching exercise below. If you are a system manager, skip ahead to teaching exercise 5-7.

Teaching exercise 5-6 for programmers: using the command interpreter

1. There is a C program in your ACCT*x* account and CLASS group called HIC. You will be compiling, linking, modifying, and running the program using the CI and the editor. It purposely has some hidden bugs in it so that it does not compile successfully the first time that you try to do so.
2. First, compile and link the program using the command line shown below:

   ```
   CCXLLK HIC,HICP
   ```

   where:

   a. HIC is the source code.

   b. HICP is the compiled and linked, executable code.

   c. The error listing will appear on the screen and not be stored in a file. (No error listing file was specified in the command line.)

   d. When compiling and linking is completed, an END OF PROGRAM message will appear on the screen.

   The general syntax for compiling and linking a program is

   ```
   CCXLLK source_code,
   executable_code[,error_listing]
   ```

Note

Since errors do occur during compiling and linking, you will eventually have to modify the source code and correct the errors so that the program will run successfully.

3. Edit the source file, make the necessary changes, and save the file. The error is clearly noted at the beginning of the source code after the DEFINE section. The correction that you should make is also explained.

   ```
   EDITOR
   /T HIC
   /LIST 8/30
   ```

   Notice that on or near line 23 there is an error. You can now make the specified changes using standard editing commands (M[ODIFY], R[EPLACE], I[NSERT])

   Now keep the file:

   ```
   /K HIC
   ```

4. Toggle back to the operating system by using the CI. Type:

   ```
   :RUN CI.PUB.SYS
   ```

   or

   ```
   :XEQ CI
   ```

   Compile and link the program as you did before. It should compile and link without errors.

5. Run the program by entering the following:

   HICP

6. The program should prompt you for your logon user name, and then print out the following message:

```
Enter your user name>

HELLO user name

WELCOME TO THE MPE/iX ADVANCED SKILLS COURSE!!!

YOUR CURRENT SEARCH PATH is ***

TODAY IS date.

GOOD LUCK!
```

7. Return to the editor by entering:

   EXIT

   You should see the editor prompt (/).

   a. Type:

      LIST 80/LAST

   b. Modify the source code on or near line 91 so that the program prints out the current search path. To do this, replace the asterisks (***) in the program's PRINTF search path statement with:

      %s

Note

Don't remove the single quotes, however.

   This indicates that a text string showing the value of HPPATH will be displayed. HPPATH is a system variable that you will learn about in the "Variables and Expressions" module.

   c. Keep the file.

8. Use the CI again to do the following:

   a. compile and link the program

   b. run the program

   c. return to the editor

9. EXIT the editor to finish. You should see the system prompt.

   If you exit the CI prematurely by entering BYE instead of EXIT, you will terminate your session. You might also leave behind the editor work file whose name consists of a K followed by a series of digits.

********** End of Exercise 5-6 **********

Teaching exercise 5-7 for system managers: using the CI

1. There is a command file in your account called HICOM. You will edit and execute this file using the editor and the CI.

2. First, execute the command file by entering the following:

   ```
   HICOM
   ```

Note

An error message results. Eventually, you will have to modify the source code and correct the errors so that the program will run without error.

3. Edit the command file, make the necessary changes, and save the file. The first line of the file needs a semicolon (;) between USER and PROMPT:

   ```
   INPUT USER;PROMPT="Enter user name> "
   ```

   To make the change and save the file, follow these instructions:

   ```
   EDITOR
   /T HICOM
   /LIST ALL
   ```

   Make the necessary change to the first line of the file using the standard editing commands (M[ODIFY], R[EPLACE], I[NSERT]).

   When you are finished making changes, press (Enter) and keep the file:

   ```
   /K HICOM
   ```

4. Toggle back to the operating system by using the CI. Enter:

   ```
   :RUN CI.PUB.SYS
   ```

   or

   ```
   :XEQ CI
   ```

   Execute the command file as you did before. The file should prompt you as follows, and then print out the following message:

   ```
   Enter your user name>

   HELLO user name

   WELCOME TO THE MPE/iX ADVANCED SKILLS COURSE!!!

   YOUR CURRENT SEARCH PATH is ***

   TODAY IS date.

   GOOD LUCK!
   ```

5. Return to the editor.

   ```
   EXIT
   ```

   You should see the editor prompt (/).

   a. Type:

      ```
      LIST ALL
      ```

   b. Modify the source code on or near line 7 so that the command file prints out the current search path. To do this, replace the asterisks (***) in the search path ECHO statement with:

```
!HPPATH
```

This indicates that a text string showing the value of HPPATH is to be displayed. HPPATH is a system variable that you will learn about in the "Variables and Expressions" module.

c. Keep the file and then use the CI again to execute the command file. The message that you receive should now display a search path.

You can also execute the command file by entering

```
:XEQ HICOM
```

d. Return to the editor and then leave the editor

```
EXIT
```

to finish.

********** End of Exercise 5-7 **********

Lesson summary

1. The command interpreter (CI) provides a quick way to toggle between the editor and the operating system.
2. The command line :RUN CI.PUB.SYS (or :XEQ CI) lets you use the CI from the editor; EXIT returns you to the previous level of the CI (in this lesson, usually the editor).
3. The CI is useful when writing and testing programs or command files.

7

Module 6: Variables and Expressions

Module 6 presents the following lessons on using variables and expressions in command files and UDCs:

Lesson 1	Understanding Variables
Lesson 2	Using Variables and Expressions

Challenge Test

1. Which of the following is a legal variable name:

   a. `VAR_1`

   b. `VAR-1`

   c. `VARIABLE#1`

   d. `VARIABLE_1`

   e. `VAR 1`

   f. `1_VAR`

2. Which of the following system-defined variables have values that can be altered?

   a. `HPPROMPT`

   b. `HPJOBLIMIT`

   c. `HPSESLIMIT`

   d. `HPPATH`

3. TRUE OR FALSE

   User-defined variables can have real numbers (decimals or fractions) as values.

4. TRUE OR FALSE

   You can set a variable to a specific value in this manner:

   ```
   SETVAR VARIABLE1,5
   ```

5. How would you display the value set in question 4?

6. How would you print this message on the screen with or without using variables:

   ```
   HELLO -- WELCOME TO THE ADVANCED SKILLS COURSE
   ```

7. How would you prompt a user for his or her name and then print this message:

   HELLO *username* -- WELCOME TO THE ADVANCED SKILLS COURSE

8. Which series of statements is correct for prompting a user for an integer value (VAR_A), adding 2 to that value, assigning the result to VAR_B, and then displaying VAR_B?

a.
```
INPUT VAR_A;PROMPT="What is the value of variable
A?
SETVAR VAR_B,VAR_A + 2
SHOWVAR VAR_B
```

b.
```
INPUT VAR_A;PROMPT="What is the value of variable
A?"
SETVAR VAR_B,!VAR_A + 2
SHOWVAR VAR_B
```

c.
```
INPUT VAR_A;PROMPT="What is the value of variable
A? "
SETVAR VAR_A,!VAR_A
SETVAR VAR_B,VAR_A + 2
SHOWVAR VAR_B
```

9. Compare the following command files, CFA and CFB. Answer the questions about them:

CFA:

```
INPUT NAME;PROMPT="What is your name (enter // to stop): "
IF NAME <> '//' THEN
   INPUT COUNTRY;PROMPT="What is your native country &
   (8 characters or less): "
   ECHO *** HELLO !NAME FROM !COUNTRY ***
ELSE
   ECHO ***SORRY, NO MESSAGE***
ENDIF
```

CFB:

```
INPUT NAME;PROMPT="What is your name (enter // to stop): "
WHILE NAME <> '//' DO
   INPUT COUNTRY;PROMPT="What is your native country &
   (8 characters or less): "
   ECHO *** HELLO !NAME FROM !COUNTRY ***
   INPUT NAME;PROMPT="What is your name (enter // to stop: "
ENDWHILE
ECHO ***SORRY, NO MESSAGE***
```

If you respond in the following manner to CFA and CFB, what happens in each case?

	Responses	CFA	CFB
a.	JULIO SPAIN		
b.	//		
c.	GINA ITALY //		

Lesson 1 Understanding Variables

Introduction

Lesson 1 covers the following topics dealing with variables and how to define them:

- how to create user-defined variables
- how to modify user-defined and some system-defined variables
- SETVAR
- SHOWVAR
- DELETEVAR

As you become more experienced, you will probably find yourself creating more and more specialized user commands. In doing so, you will probably find the use of variables invaluable. Variables are similar to the parameters that you have already used in the past lessons. Throughout the lessons in this module, you will learn how to use both system-defined and user-defined variables in your user commands.

Command files in this lesson

In this lesson, you will execute command files that use variables. The tasks performed by the command files could also be performed by UDCs; however, this would involve constantly uncataloging and recataloging UDC files to add or modify UDCs. The LF and the STATS command files will be used to illustrate various aspects of variables and expressions. Also, the contents of each file serve as good review for system-defined formal files, file equations, and backreferencing.

Note

Each of these command files could become a UDC in a UDC file if you wished.

Don't expect to understand everything about the LF and STATS command files immediately. You'll be looking at the pieces that make up the files as you progress through the lesson.

STATS command file:

This command file lets you check the number of jobs on the system and compare them to the current job limit. If the number of jobs has approached the limit (anything over three less than the job limit), the command file tells you not to stream the job.

```
ECHO *** CURRENT JOBS = !HPJOBCOUNT
ECHO *** CURRENT JOB LIMIT = !HPJOBLIMIT
IF HPJOBCOUNT<=HPJOBLIMIT-3
   ECHO *** NUMBER OF JOBS IS STILL REASONABLE --- STREAM JOB NOW ***
ELSE
   ECHO *** APPROACHING JOB LIMIT --- STREAM JOB LATER ***
ENDIF
```

LF command file:

This command file lets you list the names and characteristics of all files in a specified group and account, using LISTFILE,1 or LISTFILE,2 or LISTFILE,3 commands.

```
SETVAR FILENAME,"//"
SETVAR RESPONSE,"N"
SETVAR OPTION,"1"
INPUT RESPONSE;PROMPT="DO YOU WISH TO LIST ANY FILE(S)? (Y OR N): "
WHILE RESPONSE='Y' DO
   INPUT FILENAME;PROMPT="ENTER FILE.GROUP.ACCT (@ ALLOWED,// TO END): "
   IF FILENAME <> '//' THEN
      INPUT OPTION;PROMPT="ENTER LISTFILE OPTION (1,2,3): "
      LISTFILE !FILENAME,!OPTION
   ELSE
      SETVAR RESPONSE,'N'
   ENDIF
ENDWHILE
DELETEVAR FILENAME,RESPONSE,OPTION
```

Assigning and displaying variable values

Now that you have had a chance to look at the command files, you'll be examining the variables within them. An MPE/iX variable is just like a programming variable. It is a placeholder that appears in command syntax. When the actual command is entered, a value is specified to replace the placeholder.

To define (create) a variable, only a SETVAR statement is required:

```
SETVAR var_name,var_value
```

To display the variable value, all that is required is a SHOWVAR statement:

```
SHOWVAR var_name
```

Note

There are other ways to display a variable's value, but the SHOWVAR command suffices for this lesson.

Note

For Programmers

Unlike a programming variable, an MPE/iX variable is declared and assigned a value in one step. No type statement is required, since the variable type is automatically defined when the value is assigned. An exception to this rule is that any variable used with the INPUT command is considered type string, regardless of the value that it contains.

To demonstrate the ease of assigning and displaying values, enter the following statements:

x = your user number

```
SETVAR USER,"USERx"
SETVAR GROUP,"CLASS"
SETVAR ACCT,"ACCTx"
SHOWVAR USER,GROUP,ACCT
```

You should see the following:

```
USER = USERx
GROUP = CLASS
```

```
ACCT = ACCTx
```

Note

Either single or double quotation marks may be used. In special cases, you must use double quotes. For example, you may wish to use a single quotation mark within a character string:

```
SETVAR WHEN,"The meeting's at 8:00"
```

Also, the variable name and value hmay be separated by a space or a comma:

```
SETVAR WHEN,"The meeting's at 8:00"
```

or

```
SETVAR WHEN "The meeting's at 8:00"
```

Here are some simple examples of assigning variable values in the command files:

The STATS command file makes reference to a variable called HPJOBLIMIT.

The value of this variable is the number of jobs (job limit) that can run at one time. This is the same value that you see as JLIMIT when you do a SHOWJOB. This is a system-defined variable which happens to be read only. Its initial value is set by the LIMIT command.

This value can be changed only by the operator or system manager at the system console.

The LF command file creates and references several variables, one of which is RESPONSE. It is a user-defined variable that refers to an answer that the user must make to a specific prompt. Near the end of the file, it is automatically set to N (No):

```
SETVAR RESPONSE "N"
```

Note

Notice also, that three of the variables are assigned initial values at the beginning of the file, just in case the variables already exist and have some other values. This is a good programming practice.

Text values for variables require quotation marks around them; integer values require that you do not use quotation marks.

Variable names

Regardless of how it is defined, a variable must have a name that follows standard naming conventions:

- begin with an alpha character or an underscore (_)
- may contain 1 to 255 alphanumeric characters (including an underscore)

Examples of valid names are:

```
PACIFIC_TIME
A
_007
NUM_array
employee1
HPSESLIMIT
```

```
HPJOBCOUNT
```

Examples of invalid names are:

```
#$%&
MAX#
ERROR_#
error flag
123
```

If you look in the two command files, you will see these variables:

STATS file:

```
HPJOBLIMIT
HPJOBCOUNT
```

LF file:

```
RESPONSE
FILENAME
OPTION
```

Q6-1	Which of the following are acceptable names for variables?
	a. $DOLLARS
	b. CIERROR
	c. FRED_FLINTSTONE
	d. HPINPRI
	e. RINGO*
	f. _JAMES_BOND
	g. VARIABLE-1

Variable types

The value assigned to the variable determines the variable type, as follows:

- *string value* is a sequence of characters or digits surrounded by quotation marks.

  Example:

  ```
  SETVAR MOVIE "FANTASIA"
  SETVAR DATE,"1988"
  ```

- *integer value* is a sequence of digits preceded by +, -, $, %, or nothing.

  Example:

  ```
  SETVAR TEMP,-20
  SETVAR STOCK_DIF,+30
  SETVAR SALARY,100000
  ```

  - *hexadecimal integers* are preceded by $.

    Example:

    ```
    SETVAR DEGREES_F,$D4
    ```

  - *octal integers* are preceded by %.

    Example:

    ```
    SETVAR FLAVORS,%37
    ```

Note

Variables cannot have real number values. For example, 100000 is an acceptable value for the SALARY variable because it is an integer value. 100000.00 is not an acceptable value for the SALARY variable because it is a real value. Decimal points are not allowed in any numeric variable value.

- *boolean value* is TRUE or FALSE.

  Example:

```
SETVAR RESPONSE,TRUE
```

Q6-2 According to the values assigned them (by the user or the system), what are the types of the following variables?

```
HPJOBLIMIT (number of jobs allowed to run):

HPJOBCOUNT (number of jobs currently running):

RESPONSE   (user response - either Y or N):

FILENAME   (user-supplied file name):
```

There are two classes of variables:

- *System-defined variables* are generally used to store system-assigned information. (These are used in the STATS command file. All but two of them begin with the letters "HP".)
- *User-defined variables* are generally used to store user-assigned information. (These are used in the LF command file.)

The variables in both classes can be of any of the three types mentioned earlier.

System-defined variables

System-defined variables are variables whose names are predefined by the system, following the standard naming conventions. In general, system-defined variables are "read only," having values that are assigned and modified only by the system or system manager. Their initial values are set during system configuration and can be changed only at the system console, using special commands.

Values of a few system-defined variables can be modified by a user for that user's session.

Some of the more useful system-defined variables are shown below. A comprehensive list appears in appendix A of the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364), indicating whether such a variable is read only (not modifiable) or read/write (modifiable).

The same information is available in the online Help facility. Enter HELP VARIABLES[Return] to see the information on system-defined variables.

```
Variable            Read only?          Read/Write?

HPJOBLIMIT          YES                 NO
```

```
HPSESLIMIT          YES                 NO
HPPROMPT            NO                  YES
HPPATH              NO                  YES
HPVERSION           YES                 NO
HPUSER              YES                 NO
HPSESCOUNT          YES                 NO
```

Q6-3 Use the Help facility to look up each of these variables. What do each of these variables do? What are their current values?

Hint: Use SHOWVAR to display the value of each.

```
HPJOBLIMIT    =

HPSESLIMIT    =

HPPROMPT       =

HPPATH          =

HPVERSION      =

HPUSER          =

HPSESCOUNT    =
```

Q6-4 How could you list all of the system-defined and user-defined variables?

Hint: Use a "wildcard" character in the same manner that you did when listing files or users/accounts/groups.

To illustrate the use of SETVAR with a modifiable, system-defined variable, please enter the commands as shown in this example:

Example:

Suppose that that you want to change the prompt to "Hi There: ". You can change the value of the system-defined variable, HPPROMPT, by entering:

```
SETVAR HPPROMPT,"Hi There: "
```

Do this now. Press RETURN. You should see the new prompt. The new prompt remains in effect until you log off. You modify it later.

Use the SHOWVAR command to verify the value of the current prompt:

```
SHOWVAR HPPROMPT
```

Use the SHOWVAR command again to determine your current capabilities:

```
SHOWVAR HPUSERCAPF
```

Your capabilities should be AM, IA, BA, ND, SF, and PH.

Q6-5	What are the system-defined variables in the STATS command file?
Q6-6	Are those variables user-modifiable?

User-defined variables

User-defined variables are variables with names and values that the user specifies. Once again, names must follow standard conventions. You can read, modify, or delete any user-defined variable. User-defined variables are session dependent. They are good only for the one user during that one session. They remain as long as the user who created them does not log off. They remain even if the user changes groups using the CHGROUP command.

Enter this command:

```
CHGROUP PUB
```

What prompt do you get? Yes, it remains the same even when you change groups.

Note

You can also use CG to perform a CHGROUP. It is a UDC works as long as MYUDC3 is one of your cataloged UDC files.

Return to the CLASS group:

```
CG CLASS
```

Q6-7	In the LF command file, what are the three user-defined variables?

Deleting variables

To delete any user-defined variable and its value, you need only issue the DELETEVAR command:

```
DELETEVAR variable_name
```

You cannot, however, delete any system-defined variable. Its value may be changed, but its definition remains the same.

To illustrate this, delete the variables that you defined earlier:

```
DELETEVAR USER,GROUP,ACCT
```

Now use SHOWVAR to verify that they are gone.

```
SHOWVAR USER,GROUP,ACCT
```

You should get error messages indicating that the variables do not exist.

Now try to do the same with these system-defined variables:

```
DELETEVAR HPPROMPT,HPJOBLIMIT,HPSESLIMIT
```

What message do you get?

```
VARIABLE IS NOT DELETABLE
```

Exercise 6-1: variables

1. Define a variable called `MYPROMPT` and assign it the value currently stored in `HPPROMPT`.
2. Display the value of `MYPROMPT`.
3. Delete `MYPROMPT`. Verify that it is gone. Try to delete `HPPROMPT`. Why can you delete `MYPROMPT`, but not `HPPROMPT`?

********** End of Exercise 6-1 **********

Lesson summary

1. System-defined variables have system-assigned names and values. Some of them have values that can be modified by the system manager. A few of them can be modified by the user.
2. User-defined variables have user-defined names and values.
3. `SHOWVAR` and `SETVAR` are used to display and assign variable names and values.
4. `DELETEVAR` deletes a specified user-defined variable and its value.

Lesson 2 Using Variables and Expressions

Introduction

Lesson 2 presents the following information about how the variables can actually be used, once they have been defined:

- assigning values to variables both interactively and with programs
- displaying and manipulating variable values by using expressions

Using variables in the STATS command file

Look at how the variables HPJOBLIMIT and HPJOBCOUNT are used in this command file.

```
ECHO *** CURRENT JOBS = !HPJOBCOUNT
ECHO *** CURRENT JOB LIMIT = !HPJOBLIMIT
IF HPJOBCOUNT<=HPJOBLIMIT-3 THEN
   ECHO *** NUMBER OF JOBS IS STILL REASONABLE --- STREAM JOB NOW ***
ELSE
   ECHO *** APPROACHING JOB LIMIT --- STREAM JOB LATER ***
ENDIF
```

ECHO and variable dereferencing

ECHO displays whatever information follows it. ECHO is often used to display text messages on the screen. In the STATS file, one line that is to be displayed is:

```
*** CURRENT JOBS = !HPJOBCOUNT
```

The exclamation point will not be displayed, nor will the word "HPJOBCOUNT." Instead, the current value of HPJOBCOUNT will replace the word and appear on the screen. This indicates how many jobs are currently running.

For example, suppose that there are 25 jobs executing on the system. The ECHO command STATS produces this message on the screen:

```
*** CURRENT JOBS = 25
```

Note

When the ECHO command is issued, the CI searches the line for the special symbol, "!", substitutes a value for the named variable, and executes the command.

This is called *variable dereferencing*. Variable dereferencing refers to the act of substituting the value of a variable in place of the variable name before executing the command that contains that variable name. There are two ways to do this (examples appear on the following pages): *implicit dereferencing* and *explicit dereferencing*.

Explicit dereferencing. This involves using an exclamation point(!). This method is similar to the method used with parameters of user commands. When the system encounters an exclamation point in front of a variable name, it substitutes the value for the variable name. This method is available with *any* command line.

Implicit dereferencing. This does *not* involve an exclamation point. It is done automatically, without an exclamation point. Implicit deferencing occurs in the commands SETVAR, IF-THEN-ELSE-ENDIF, WHILE-ENDWHILE, and CALC.

(You will look at IF-THEN-ELSE-ENDIF and WHILE-ENDWHILE later in this lesson.) The CALC command is a simple calculator command that will not be discussed; however, its syntax and description are in the online help facility and in the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364)

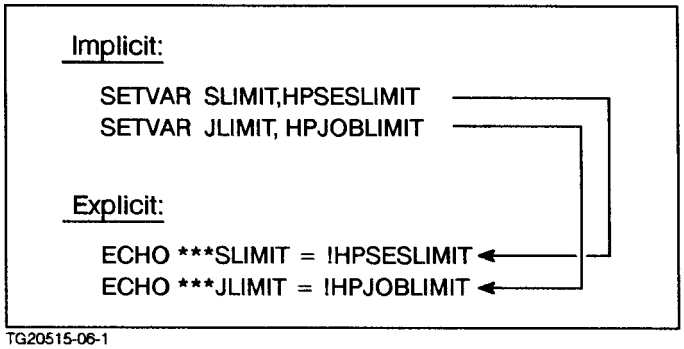


Figure 7-1. Implicit and Explicit Dereferencing

Q6-8	Which method of dereferencing is used in the two ECHO statements of the STATS command file?

Dereferencing examples. You have already seen examples of value assignment with SETVAR, like the following, where no exclamation point was necessary:

```
SETVAR USER,HPUSER
```

HPUSER requires no exclamation point.

If you wanted to set your prompt so that it would echo your user name, as stored in HPUSER, you would have to use an exclamation point:

```
SETVAR HPPROMPT,"Hi There !HPUSER"
```

HPUSER does require an exclamation point in this case. An exclamation point is necessary if a variable name must be used within the quotation marks following SETVAR. Without the exclamation point, the new prompt looks like this:

```
Hi There HPUSER:
```

Q6-9 In an earlier lesson you defined three variables, USER, GROUP, and ACCT. How would you define them again if they were to be given the same values as those currently stored in the system-defined variables, HPUSER, HPGROUP, and HPACCOUNT?

```
SETVAR USER,

SETVAR GROUP,

SETVAR ACCT,
```

Note

You may explicitly dereference a variable in a statement that requires only implicit dereferencing.

Now look at another case where no exclamation point is necessary, the IF-THEN-ELSE statement.

IF-THEN-ELSE-ENDIF statement

In an IF-THEN-ELSE statement, when the IF condition is true, the THEN action is performed; otherwise (if it is not true), the ELSE action is performed.

In the STATS command file, a simple IF-THEN-ELSE statement is used to compare the job limit with the current number of jobs running in order to determine what action to take.

This means that if the current number of jobs is getting close to the job limit, a message is displayed indicating that your job should be streamed later. Otherwise, a different message is displayed. ENDIF marks the end of the statement.

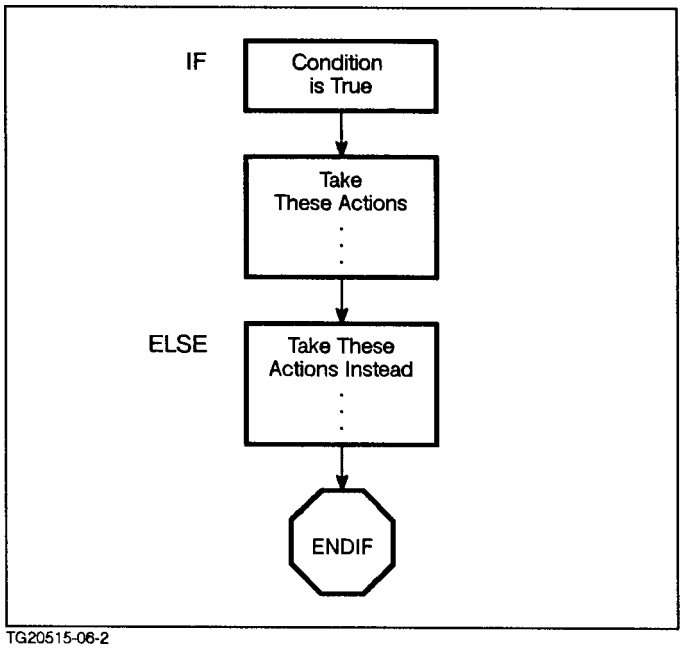


Figure 7-2. IF-THEN-ELSE-ENDIF Statement

```
IF condition is true THEN
  take this action
ELSE
  take a different action
ENDIF
```

- Note that IF is followed by an optional THEN.
- ELSE is optional and can be included if you have another choice.
- ENDIF must "close" the IF statement.
- IF, ELSE, and ENDIF keywords are not indented, but actions associated with THEN and ELSE are indented. This helps improve readability, but is not required.
- No exclamation point is needed in front of the variable names used in the IF or ELSE conditions:

  ```
  IF HPJOBCOUNT <= HPJOBLIMIT-3 THEN
  ```

  Translation:

  If HPJOBCOUNT is less than or equal to 3 less than HPJOBLIMIT, then . . .

Q6-10 According to the IF-THEN statement in the STATS file, what message is displayed in each of the following situations:

a. The current number of jobs is 10, and the job limit is 12.

b. The current number of jobs is 5, and the job limit is 10.

Expressions. Variables in the IF-THEN-ELSE statement are used in conjunction with certain math operations. These operations are represented by symbols. A combination of variables, constants, and operators are referred to as an *expression*. Variables that appear in expressions do not require exclamation points (!) to be deferenced. Dereferencing is done implicitly:

```
IF HPJOBCOUNT <= HPJOBLIMIT-3 THEN
...action1...
ELSE
...action2...
```

The first symbol (<=) represents a comparative (relational) operation: less-than-or-equal-to.

The second symbol (-) represents an arithmetic operation, subtraction.

The following is a basic list of the operations that you can perform in MPE/iX variables, and the symbols that represent those operations. For a more exhaustive list, refer appendix B of the *MPE/iX Commands Reference Manual* (32650-60002).

Arithmetic Operations:

+	Addition
-	Subtraction
*	Multiplication
/	Division

Relational Operations:

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
<>	Not equal to

Exercise 6-2: IF-THEN-ELSE statement

1. Write a command file containing an IF-THEN-ELSE statement that checks to see how many people are on the system. If the current number is greater than or equal to three-fourths (3/4) of the session limit, do the following:

a. Print on the screen the current number of sessions and the session limit.

b. Print the message, ***TOO MANY PEOPLE ON SYSTEM --- LOG OFF NOW***

Otherwise, do the following:

c. Print on the screen the current number of sessions and the session limit.

d. Print the message, ***CURRENT SESSION LIMIT NOT EXCEEDED***.

Note

Hints are on this page if you need them. Try not to use the hints until you absolutely have to.

Hints:

- Use ECHO with explicit dereferencing to display the message.
- Use expressions and implicit dereferencing with the IF condition.
- Use (3*HPSESLIMIT)/4 to represent three-fourths of the session limit value. Remember, variables can only take on integer values, and 3/4 is not an integer value.

********** End of Exercise 6-2 **********

Using variables in the LF command file:

Let's look at how the variables RESPONSE, FILENAME, and OPTION are used in the LF command file:

```
SETVAR FILENAME,"//"
SETVAR RESPONSE,"N"
SETVAR OPTION,"1"
INPUT RESPONSE;PROMPT="DO YOU WISH TO LIST ANY FILE(S)? (Y OR N): "
WHILE RESPONSE="Y" DO
  INPUT FILENAME;PROMPT="ENTER FILE OR FILE.GROUP.ACCT &
  (@ ALLOWED, // TO END): "
  IF FILENAME <> "//" THEN
    INPUT OPTION;PROMPT="ENTER LISTFILE OPTION (1,2,3): "
    LISTFILE !FILENAME,!OPTION
  ELSE
    SETVAR RESPONSE,"N"
  ENDIF
ENDWHILE
```

INPUT command

Other commands may be used to assign a value to a variable. The INPUT command lets the user interactively assign or change a variable value.

This command can optionally display a prompt on the screen, and the value that is entered by the user is assigned to the specified variable.

For example, in order for the LF command file to operate correctly, it must ask the user whether or not it should list a file. INPUT specifies the variable (RESPONSE) whose value will be supplied by the user; the PROMPT keyword specifies the prompt that the user sees:

```
INPUT RESPONSE;PROMPT="DO YOU WISH TO LIST ANY FILE(S)? (Y OR N):
```

The user sees this:

```
DO YOU WISH TO LIST ANY FILE(S)? (Y OR N):
```

If the PROMPT option were not specified, the user would only see a blank line and have to guess what to input. It makes more sense to prompt for input in an intelligent manner, by printing some type of message.

INPUT treats all user responses as string values (series of alphanumeric characters). This means that if the user enters a digit, it will be considered a string value, not a numeric value.

To use such digits from an INPUT statement as integers, simply dereference the variable explicitly, as shown below:

Example:

Suppose that you wish the user to enter a number so that an addition operation can be performed. Please create a small command file called ADD1:

```
INPUT NUMBER;PROMPT="Enter a whole number >  "
SETVAR A,!NUMBER + 2
SHOWVAR A
DELETEVAR NUMBER,A
```

Execute the ADD1 command file. It performs the addition because NUMBER is treated as an integer (2) since you explicitly dereferenced it as !NUMBER.

Now create a new command file called ADD2. In it, deliberately do *not* explicitly dereference the number:

```
INPUT NUMBER;PROMPT="Enter a whole number (no fractional part)> "
SETVAR A,NUMBER + 2
SHOWVAR A
DELETEVAR NUMBER,A
```

Execute the ADD2 command file. You get this error:

```
SETVAR A,NUMBER + 2
                  ^

ILLEGAL CHARACTER FOUND, EXPECTED A STRING (CIERR 9815)
```

ADD2 does not perform the addition, since NUMBER is treated as a string value, not as a numeric value.

Note

Good practice dictates that you use DELETEVAR to delete your variables when you are done using them.

WHILE loop

The WHILE loop is similar to the IF-THEN-ELSE-ENDIF statement in that it involves conditions and actions. As the name implies, the actions are performed over and over again (in a loop) until the conditions are no longer true.

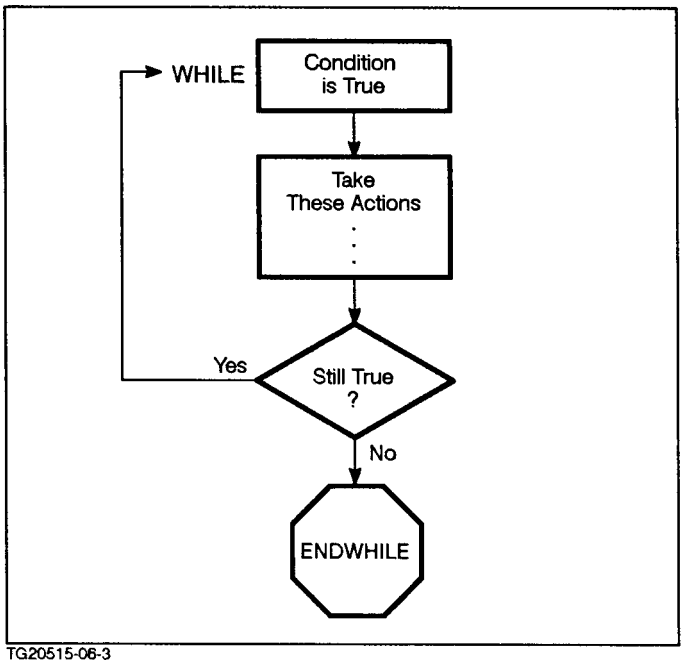


Figure 7-3. WHILE Loop

```
WHILE condition is true DO
  this action
ENDWHILE
```

As long as the condition is true, the action continues. As soon as the condition is no longer true, the action stops. Note the following about the `WHILE-ENDWHILE` syntax:

- The `WHILE` statement must always be matched by an `ENDWHILE` statement.
- The action associated with `WHILE` is indented for improved readability.
- The keyword `DO` is optional.

Q6-11	According to the LF command file, you are initially asked a question. Until you give a certain response, the system continues to prompt you for file names. What causes this to occur?
Q6-12	According to the LF command file, once you are prompted for a file name, what response can you give to terminate further prompting?

Exercise 6-3: while loop

1. Write a command file called CF that does the following:
   a. Asks if you would like to change your prompt.
   b. If you answer no, it shows you the current prompt.
   c. If you answer yes, it asks you what new prompt you would like in place of the current prompt character(s) on your screen. If you enter (Return), it assumes that the current prompt is okay and displays the current prompt with this message:

      ```
      *** Current prompt remains in effect ***
      ```

      Hint: A (Return) could be represented as "" since the two quotation marks with nothing between them indicate that nothing is entered. Make sure that the variable that you define to hold the (Return) value is *not* an integer.
   d. If you enter a new prompt, CF prints the new prompt and asks if it is acceptable. If it is not acceptable, CF asks for a new prompt again and continues to do so until the prompt is acceptable.
   e. Deletes all of the variables at the end of the file.
2. Execute CF and enter a new prompt, ABC, then another, DEF, and then finally, !HPUSER: What prompt is finally displayed?
3. Use SETVAR HPPROMPT to change HPPROMPT to the initial system prompt (:).

********** End of Exercise 6-3 **********

Lesson summary

1. Explicit dereferencing (with an exclamation point) should be used to display or assign variable values in any command *other than* `SETVAR`, `IF`, `WHILE`, and `CALC`.
2. Implicit dereferencing (no exclamation point) should be used with `SETVAR`, `IF-THEN-ELSE-ENDIF`, `WHILE-ENDWHILE` and `CALC`.
3. `DELETEVAR` should be used to delete user-defined variables (both the name and the value).

8

Solutions

Solutions to Module 1: Account Management

Check Your Answers

	Is Your Answer ... ?	If Not, Start With
1.	AM=Account Manager AL=Account Librarian GL=Group Librarian ND=*Nonshareable Devices SF=*Save File BA=*Batch Access IA=*Interactive Access PH=Process Handling * *indicates default capability*	Lesson 1
2.	Indicate account manager, general user, or system manager: 1. List information on their account (AM, GU, SM) 2. Change their account capabilities (SM) 3. List their account password (AM, SM) 4. Change their account password (SM)	Lesson 1
3.	`REPORT` and `LISTGROUP`	Lesson 2
4.	`NEWGROUP OURGROUP;PASS=PW; ACCESS=(R,W,X:GL)`	Lesson 2
5.	`ALTGROUP` and `PURGEGROUP`	Lesson 2
6.	`ALTSEC`	Lesson 2
7.	`NEWUSER JEAN;PASS=OPEN;HOME=CLASS`	Lesson 3
8.	The `MKACCT` command file	Lesson 3
9.	`ALTUSER JEAN;PASS=CLOSE` `PURGEUSER JEAN`	Lesson 3

Lesson 1 Managing Your Account

Q 1-1

Code	Capability
AM	Account manager
AL	Account librarian
GL	Group librarian
ND	Nonshareable devices (such as a tape drive or hot printer)
SF	Save files
BA	Batch access (for running jobs)
IA	Interactive access
PH	Process handling (needed for programming functions)

Q 1-2 No. Since PM is not one of your account's capabilities, no user may have this capability.

Q 1-3 Access is restricted to member of this account.

Q 1-4 e. all of the above.

Exercise 1-1: Lesson 1 Review

	Task	Account Manager	User	System Operations
1.	List information for a user's account.	X	X	X
2.	Set account passwords.			X
3.	Change user passwords.	X	X	X
4.	List account capabilities.	X	X	X
5.	Show jobs and sessions currently running on system.	X	X	X
6.	Change account capabilities.			X
7.	Use the `LISTACCT` command to list the account password.	X		X
8.	Set session and job limits.			X

********** End of Exercise 1-1 **********

Lesson 2 Managing Groups

Q 1-5

a. Most CPU time: `PUB` Group

b. Most file space: `CLASS` Group

c. Least connect time: `PROJECT` Group

Q 1-6

A user having account manager capability is able to list information for all groups within an account. Users without account manager capability may only list information for their current group.

Q 1-7

Code	Capability
PH	Process handling
IA	Interactive access
BA	Batch access
PM	Privileged mode
MR	Multiple RINS
DS	Extra data segments

Q 1-8

Code	Capability
ANY	Any user (on the system)
AC	Any user in the account
GU	Group user
AL	Account librarian
GL	Group librarian
CR	Creating user

Exercise 1-2: Using NEWGROUP

1. `NEWGROUP STOP`

2a. Default capabilities assigned: IA and BA

2b. Default file access and user codes: Read, Lock, Append, Write, Execute, and Save for any user in the group.

2c. To check on the group's password: `LISTGROUP STOP;PASS`

3. `NEWGROUP GO;PASS=FAST;CAP=IA,BA,PH`

4. `LISTGROUP GO;PASS`

5. `NEWGROUP SLOW;PASS=DOWN;`
`ACCESS=(R,L,A,W,X,S:GL)`

6. `LISTGROUP SLOW;PASS`

********** End of Exercise 1-2 **********

Exercise 1-3: Using ALTGROUP

1a. `ALTGROUP PROJECT;PASS=PJ`

1b. `ALTGROUP GO;CAP=`

1c. `ALTGROUP SLOW;ACCESS=(R,W,A,X:GU)`

This results in no user having Lock or Save access to the group. More specifically, account users (except AM) cannot create permanent files without Save access to the group.

1d. `ALTGROUP SLOW;PASS=`

3. `LISTGROUP` or `LISTGROUP groupname`

********** End of Exercise 1-3 **********

Exercise 1-4: Using ALTSEC

1. for example:

```
ALTSECT !!myfile!!;NEWACD=(R,W:!!mylogon.myaccount!!)
```

2. for example:

```
ALTSECT !!myfile!!;ADDPAIR=(R:!!username.accountname!!)
```

********** End of Exercise 1-4 **********

Exercise 1-4: Lesson 2 Review

a. `LISTGROUP`

b. `ALTGROUP`

c. `REPORT`

d. `NEWGROUP`

e. `LISTGROUP name;PASS`

f. `ALTSEC`

g. `VERSION`

h. `PURGEGROUP`

********** End of Exercise 1-4 **********

Lesson 3 Managing Users

Q 1-9 LISTUSER displays information about one or more users in an account or on a system.

Exercise 1-5: Using LISTUSER

1. These LISTUSER parameters are optional:

   The name(s) of the user(s) and account(s), the output file name, and the keyword, PASS.

2. The parameters default to these values:

   The name of the user and account will be all of the users in your logon account.

   PASS will be ignored, and no password will be displayed.

3a. There is one user: USER*x* (*x*= your particular student number)

3b. You would use this command to list all the users on the system:

```
LISTUSER @.@
```

   You would need system manager (SM) capability in order to do this.

4. The system writes user information to the INFO file. Check this by entering PRINT INFO. If you leave out the comma, the system attempts to locate the user INFO and display this message:

```
NON-EXISTENT USER
```

5. LISTUSER @.@ executes only if you have SM capability.

6. This LISTUSER command lets you display the passwords associated with the users in the account:

```
LISTUSER @.ACCTx;PASS
```

   You need at least AM or SM capabilities to list the passwords.

********** End of Exercise 1-5 **********

Q 1-10 The capability codes are: AM, ND, SF, BA, IA, PH

The capability codes mean the following:

a. AM - account manager

b. ND - nonshareable devices

c. SF - save files

d. BA - batch access

e. IA - interactive access

f. PH - process handling

Q 1-11 To list all the users on your account, you must have AM capabilities.

To list all the users on the system, you must have SM capabilities.

Q 1-12 If you have neither AM nor SM capabilities, you can list the capabilities of your current user.

Q 1-13 To change MYUSER's password to MYPASS, you would use this command:

```
ALTUSER MYUSER.ACCTx; PASS=MYPASS
```

a. After you make this change, MYUSER remains logged on; however, when MYUSER logs off and tries to log on again, he or she must specify the new password.

b. You must have AM capabilities.

   You must have SM capabilities if you are in a different account.

Q 1-14 The new user has the following capabilities:

```
ND, SF, BA, IA.
```

Q 1-15 You get this error message:

```
USER CAPABILITY REQUESTED EXCEED ACCOUNT
CAPABILITIES
CV, UV NOT GRANTED (CIWARN 794)
```

Since the ACCT*x* account does not have CV capabilities, no user in the account can have those capabilities.

The user, USER*x*B, has been created; however, it does not have CV capabilities. It has instead ND, SF, BA, IA, and PH capabilities.

Solutions to Module 2: File Management

Check Your Answers

Is Your Answer ... ?	If Not, Start With
1. a. `LISTFILE,2` b. `LISTFILE,3` c. `LISTFILE, -1` d. `LISTFILE,1` e. `LISTFILE,6`	Lesson 1
2. a. permanent file b. temporary file c. formal file designator	Lesson 1
3. `BUILD FILE1;REC=-80,,F,ASCII`	Lesson 2
4. a. keyboard b. terminal screen c. "bit bucket" d. created during the compiling of a program e. created when compiling is complete	Lesson 3
5. 1. `LISTEQ` 2. `LISTFILE;TEMP` 3. `RESET`	Lesson 4
6. `FILE PRT;DEV=LP` `PRINT MYFILE1;OUT=*PRT`	Lesson 4

Lesson 1 Introduction to Files

Q 2-1

Command	Information
`LISTFILE,1`	Logical record information
`LISTFILE,2`	File space information
`LISTFILE,3`	File structure, creation, and modification information

Q 2-2 `LISTFILE @.@,4`

Q 2-3 System-defined file names begin with $.

Q 2-4 Formal files and permanent files both have from one to eight characters, begin with an alpha character and cannot include noncharacters ($%"&"*., etc).

Exercise 2-1: Lesson 2 Review

1.

	File Characteristics	Permanent	Formal	Temporary
a.	File names are 1-8 characters long.	X	X	X
b.	File names are aliases for devices.		X	
c.	File names may begin with a $.			X
d.	Files may be listed with LISTFILE.	X		
e.	Files may be listed with LISTEQ.		X	
f.	File names may begin with an alpha character.	X	X	
g.	Files are listed with LISTFILE ;TEMP.			X

2a. `LISTFILE @.@,2`

2b. `LISTFILE MYJOB1,-1`

2c. `LISTFILE MYJOB1,1`

2d. `LISTFILE,3`

2e. `LISTFILE`

2f. `LISTFILE,4`

2g. `LISTFILE @.@,6`

2h. `LISTFILE @.PUB,3;PASS`

2i. `LISTFILE,5` *and* `LISTFILE,7`

*******End of Exercise 2-1 ************

Lesson 2 Building Disk Files

Q 2-5 The record length is 80 bytes (80B).

Q 2-6 The blocking factor (R/B) is 16.

Q 2-7 The record type is fixed length (F).

Q 2-8 The file type is ASCII (A).

Q 2-9 The original BUILD command that created MYFILE was the following:

```
BUILD MYFILE;REC=-80,16,F,ASCII
```

Exercise 2-2: Creating a File to Specification

1. The following are the BUILD commands necessary to create each of the specified files:

   a. `BUILD FILE1;REC=64,,F,ASCII;CIR`

b. BUILD FILE2;REC=256,,V,BINARY

c. BUILD FILE3;REC=-88,,V,ASCII

2. The following is the BUILD command necessary to create a file with the following characteristics:

```
record size   = 80 bytes
record length = fixed length
file type     = ASCII
BUILD myname;REC=-80,,F,ASCII
```

********** End of Exercise 2-2 *********

Lesson 3 Creating Temporary Files

Exercise 2-3: $OLDPASS

Follow the designated steps to generate $OLDPASS and $NEWPASS:

2. The compile and link strings for the HIC program are given below:

```
CCXLLK HIC,,HIERR

HIC     = name of source file
,       = default name of compiled, linked code ($OLDPASS)
HIERR   = name of errorlisting file (default = terminal screen)
```

3a. The temporary file that was generated is:

$OLDPASS

3b. The file type is NMPRG, the code is 1030. This is a native mode program file.

3c. If you do a LISTFILE @.ACCT*x*,2, the following happens:

You do *not* see $OLDPASS because it is a temporary file.

4a. SAVE $OLDPASS,HICP

4b. LISTFILE HICP,2

5. Since the HIERR record size is too large for the editor, you might wish to BUILD another file with the proper record size and copy the contents of HIERR into it.

********** End of Exercise 2-3 **********

Lesson 4 Using File Equations

Q 2-10 $STDIN is the standard input device, which is normally the terminal.

Q 2-11 OUT is the alias for $STDLIST, the standard output device.

Q 2-12 According to the MAILPRNT equation, the HP Desk messages will be printed on LP (line printer).

Q 2-13 To redirect HP Desk messages to the terminal screen, you might change the file equation as follows:

```
FILE MAILPRNT=$STDLIST
```

To redirect HP Desk messages to the line printer, you might change the file equation as follows:

```
FILE MAILPRNT;DEV=LP
```

Q 2-14 According to the first file equation, the name and characteristics of the file associated with FILEIN are the following:

```
File name       = MYFILEA
Record size     = 80 bytes
Blocking factor = (default)
Record type     = fixed length
File type       = ASCII
```

Q 2-15 According to the second file equation, the name and characteristic of the file associated with FILEOUT are the following:

```
File name       = MYFILEB
Record size     = 80 bytes
Blocking factor = (default)
Record type     = fixed length
File type       = ASCII
```

Q 2-16 If you no longer want the FILEIN or FILEOUT equations to be in effect, you would do the following:

```
RESET FILEIN
RESET FILEOUT
```

Exercise 2-5: Creating and Using File Equations

2. Enter the command to display all existing file equations:

```
LISTEQ
```

3a. To redirect output to the terminal instead of to FILE2, change the OUT file equation as follows:

```
FILE OUT=$STDLIST
```

3b. To redirect input from FILE1, instead of from the terminal, change the IN file equation as follows:

```
FILE IN=FILE;REC=-80,,F,ASCII
```

4a. `FILE OUTPUT;DEV=LP`

What is the name of the file designator? OUTPUT

What device does it refer to? line printer

4b. `FILE FORMAT;DEV=LP;ENV=PICA.HPENV.SYS`

What is the name of the file designator? FORMAT

What device does it refer to? line printer

What is the environment file? PICA

What does this file equation do? It lets you print something on the line printer using pica type.

4c. `FILE INPUT=MYFILE1;REC=-80,,F,ASCII`

What is the name of the file designator? `INPUT`

What file does it refer to? `MYFILE1`

What happens if a program reads from `*INPUT`? It reads the contents of `MYFILE1`.

What size records are in the actual file? 80 byte records

4d. `FILE OUTPUT = $NULL`

What is the name of the file designator? `*OUTPUT`

What device or file does `*OUTPUT` refer to? Nothing

What happens if you write to `*OUTPUT`? Output disappears

5a. Prints HP Desk message #10 on the line printer.

5b. Prints the contents of `MYFILE` to the standard output device (usually the terminal screen).

5c. Reads input from the standard input device (usually the keyboard).

5d. Reads input from an already existing file called `MYFILE`.

5e. Writes output to a new file called `MYFILE` (with the specified characteristics).

6a. This file equation defines `HIERR` as the terminal screen:

```
FILE HIERR=$STDLIST
```

6b. According to the file equation, by directing the error listing to `*HIERR`, the listing should print on the screen.

6c. To redirect the error listing to a file called ERROR1 with default file characteristics, change the file equation as follows:

```
FILE HIERR=ERROR1
```

6d. According to the file equation, by directing the error listing to `*HIERR`, the listing should print in the `ERROR1` file.

6e. To redirect the error listing to a file called `ERROR2`, change the file equation as follows:

`FILE HIERR=ERROR2;REC=-80,,F,ASCII`

According to the file equation, the error listing should print in the ERROR2 file.

********** End of Exercise 2-5 *********

Solutions to Module 3: Batch Processing

Check Your Answers

	Is Your Answer ... ?	If Not Start With
1.	b,c	Lesson 1
2.	SCHED, WAIT, INTRO, EXEC, SUSP	Lesson 1
3.	a. jobfence b. session limit c. input priority d. job limit	Lesson 1
4.	!JOB JOBx,USERx/UPASSx.& !ACCTx/APASSx;INPRI=8;OUTCLASS=LP,1 !PRINT MYFILE1;OUT=*LP !EOJ	Lesson 2
5.	a. STREAM JOBFILE b. STREAM JOBFILE;AT=23:00 c. STREAM JOBFILE;DAY=MON; AT=8:00 d. STREAM JOBFILE;DAY=WED;AT=22:00	Lesson 3
6.	BREAKJOB, ALTJOB, RESUMEJOB, STREAM, ABORTJOB	Lesson 4
7.	CREATE, READY, ACTIVE, OPEN, PRINT, DEFER, SPSAVE, PROBLM, DELPND, XFER In PRINT state, file is printing. READY means that the spool file is ready for printing. CREATE means that the spool file is being created as output.	Lesson 4
8.	LISTSPF, SPOOLF, PRINT	Lesson 4

Lesson 1 Introducing Jobs

Q 3-1

	Characteristic	Jobs	Session
a.	interactive processing.		X
b.	synonymous with batch processing.	X	
c.	does not require constant supervision.	X	
d.	submits a single command at a time.		X
e.	can be scheduled to run at specified time.	X	
f.	most time/resource efficient.	X	
g.	Executes with STREAM command.	X	

Q 3-2 Only J32 is executing.

Q 3-3 Job J35 is being submitted.

Q 3-4 J30 is awaiting execution.

Q 3-5 (b) Only JOBB executes since its input priority is above the jobfence.

Q 3-6 The job will be listed as WAIT.

Q 3-7 J35 was streamed at 5:45 A.M.

Q 3-8 J35 was submitted from the PROJECT account by the STATS user.

(Note: It *is* possible for a user other than STATS, working in an account other than PROJECT, to submit this job.)

Q 3-9 Currently, a maximum of 11 jobs and 20 sessions can be run on this system.

Q 3-10 Since only two jobs are executing, nine more jobs could be submitted for execution.

Q 3-11 No jobs have been suspended.

Q 3-12 Job #J34 will run at 10:00 P.M.

Q 3-13 SHOWJOB command parameters

a. list job summary information: SHOWJOB STATUS

b. list jobs which have been scheduled: SHOWJOB SCHED

c. list information on all jobs: SHOWJOB JOB=@J

d. list information on Job 105: SHOWJOB #J105

e. list status of all jobs that you might have submitted from your account: SHOWJOB JOB=USERx.ACCTx

Exercise 3-1: Lesson 1 Review

1. TRUE
2. FALSE
3. TRUE
4. TRUE
5. FALSE

********** End of Exercise 3-1 **********

Lesson 2 Examining a Job File

Q 3-14

JOB Command Parameter	Required	Optional
User Name	X	
Account Name	X	
Group Name		X
Job Name		X
Time		X
INPRI		X
RESTART		X
OUTCLASS		X
SPSAVE		X

Q 3-15 MYJOB1 is the job name; ACCT*x* is the account name; USER*x* is the user name; CLASS is the group name.

Q 3-16 No. The group password should be inserted in the !JOB command in order for MYJOB1 to run.

```
!JOB MYJOB1,USERx/UPASSX...CLASS/LASER..
```

Q 3-17 If the system crashes and you do not have the RESTART parameter in your !JOB command, your job will not continue processing when the system is restarted, even if the operator uses the RECOVERY option.

Q 3-18 (c) The outclass priority must be set above the outfence.

Q 3-19 The job listing from MYJOB1 is scheduled to be printed on the line printer (LP). The outclass priority is 1 (which means the job listing will not print).

Q 3-20 MYJOB1 IS DONE!! prints on USERx's screen when MYJOB1 completes processing.

Q 3-21 This line accesses the editor.

Q 3-22 This line texts in MYFILE1 so that it can be listed with the L ALL, OFFLINE command.

Q 3-23 The print priority of the output is 1. The line printer (LP) is scheduled to print the output; however, the output does not print until the priority, currently 1, has a value above the outfence.

Exercise 3-2: Lesson 2 Review

1. `!JOB MYJOB2,`
   `USERx/UPASSX.ACCTx/APASSX`

2. `SHOWJOB STATUS` (to verify jobfence)

   `!JOB MYJOB2...;INPRI=10  (or 1 greater than the jobfence);...`

3. `!JOB MYJOB2,....ACCTx/APASSX,CLASS;INPRI=10;RESTART;....`

4. `!JOB MYJOB2.....;OUTCLASS=LP,1`

5. The line following `!EDITOR` should read `T MYFILE2.CLASS`

6. Add a line after EXIT: `!TELL USERx.ACCTx  MYJOB2 IS COMPLETE!!`

The modified `MYJOB2` file should now look like the following:

```
!JOB MYJOB2,USERx/UPASSX.ACCTx/APASSX,CLASS;&
!INPRI=10;RESTART;OUTCLASS=LP,1
!COMMENT MYJOB2 PRINTS MYFILE2
!CONTINUE
!EDITOR
T MYFILE2.CLASS
L ALL, OFFLINE
EXIT
!TELL USERx.ACCTx MYJOB2 IS COMPLETE!!
!EOJ
```

********** End of Exercise 3-2 **********

Lesson 3 Creating and Streaming a Job File

Q 3-24 The state of the job is `EXEC`.

Depending on how your terminal is set up, the state of your session may be `EXEC QUIET`. This means that no messages can be sent to your terminal screen.

Q 3-25 The job state is `SCHED`. This means that the job is scheduled to run at a later time.

Q 3-26 You should match the descriptions and command as follows:

	Command		Description
1.	`STREAM JOBFILE`	d.	Job runs immediately.
2.	`STREAM JOBFILE ;AT=8:00`	c.	Job will run today, at 8:00 A.M.
3.	`STREAM JOBFILE ;AT=20:00`	g.	Job will run today, at 8:00 P.M.
4.	`STREAM JOBFILE ;IN=,+8`	f.	Job will run today, 8 hours from now.
5.	`STREAM JOBFILE; IN=+1,+8`	h.	Job will run tomorrow, 8 hours from now.
6.	`STREAM JOBFILE; DATE=08/09/89; AT=20:00`	b.	Job will run on August 9 at 8:00 P.M.
7.	`STREAM JOBFILE ;DAY=MONDAY ;AT=22:00`	a.	Job will run on Monday, at 10:00 P.M.
8.	`STREAM JOBFILE; DATE=08/09/89`	e.	Job will run on August 9, at the same time you execute the `STREAM` command today.

Q 3-27 The job will execute on the day specified, at the time that matches the time when the job was submitted.

Q 3-28 When you do a `SHOWJOB #J`*xxx*, you should see this message after the job has been aborted:

```
NO SUCH JOB(S)

JOBFENCE=x;JLIMIT=y;SLIMIT=z
```

Q 3-29 The job is supposed to run at *midnight*. Even though the job has not yet run, use the `ABORTJOB` command. It works on scheduled jobs, as well as executing jobs.

Q 3-30 The status of the job is `SUSP`.

Q 3-31 The status of the job is now `EXEC`.

Q 3-32 The job is in `WAIT` state.

Q 3-33 The job is in `EXEC` state.

Lesson 4 Monitoring Job Progress

Q 3-34 There is one file associated with JOB1:

$STDLIST

Note

If you do a LISTSPF at just the right moment, a spool file called EDTLIST may appear in the display. EDTLIST is a transitory spool file associated with the editor.

Q 3-35 The priority of the $STDLIST spool file is 1. This should be below the outfence, which is generally 7.

Q 3-36 When the spool file is no longer active the spool file "disappears" from the LISTSPF display.

Exercise 3-4 Aborting a Job and Using PRINT

2. If you use PRINT and attempt to access the job listing spool file before the job has completed, you will be unable to look at the file because it is not Ready.

   You can use PRINT to examine the spool file while it is still in the CREATE state.

3a. If you did not alter the priority, the file, with any errors, would print on the line printer.

3c. Yes, you can now access and view the job listing.

3d. The following message appears at the end of the job listing.

4. The state of the spool file (now that the job has been aborted) is Ready.

```
*** JOB ABORTED BY SYSTEM MANAGEMENT ***
```

5. Display remaining spool files and purge them.

```
LISTSPF @;SELEQ=[OWNER=USERx.ACCTx]
```

Abort any job that still as a $STDIN spool file. Use ABORTJOB.

```
SPOOLF (!!filenumber, filenumber, ...!!)SELEQ=[OWNER=USERx.ACCTx];DELETE
```

Purge those spool files.

```
>EXIT
```

**********End of Exercise 3-4 **********

Solutions to Module 4: File Transfer

Check Your Answers

Is Your Answer ... ?	If Not, Start With
1. `FCOPY FROM=MYJOB1.PUB ;TO=MYJOB1.CLASS;NEW`	Lesson 1
2. `FCOPY FROM=STATS.PUB.ADVUSER ;TO=STATS.CLASS;NEW`	Lesson 1

3. `FILE FILEA;REC=-80,,F,ASCII ;ACC=APPEND ;DISC=10000`
`FCOPY FROM=FILE1;TO=*FILEA;NEW`
`FCOPY FROM=FILE2;TO=*FILEA`
`FCOPY FROM=FILE3;TO=*FILEA`
	Lesson 1

4. `FILE T;DEV=TAPE` `FCOPY FROM=MYJOB1;TO=*T`	Lesson 1
5 `STORE @.PUB.ACCTA;*T;TRANSPORT`	Lesson 2
6. `FILE T;DEV=TAPE;DEN=1600`	Lesson 2
7. a. `PROGRESS` b. `SHOW` c. `PURGE`	Lesson 2
8. `RESTORE *T;@.PUB.ACCTA`	Lesson 2

Lesson 1 Using FCOPY

Q 4-1 When you copy files between account groups, you need to include source and destination group information.

Q 4-2 When copying a file to another account, you need to include the file's current group and account location.

```
FCOPY FROM=MYJOB1.PUB.ACCTx
;TO=MYJOB1.group;NEW
```

Exercise 4-2 Appending One File to Another

4. `JOBX` consists of 16 records.

6. `JOBX` consists of 27 records.

7. `FCOPY FROM=JOB3;TO=*F1`

8. `JOBX` consists of 38 records.

9. `FCOPY FROM=JOBO;TO=*F1`

10. `JOBX` now consists of 49 records.

11. `PURGE JOBX`

********** End of Exercise 4-2 *******

Q 4-3

```
FILE OUT;DEV=LP
FCOPY FROM=MYJOB2.PUB.ACCTx;TO=*OUT
```

Q 4-4 You need to add an asterisk before the T (tape file name) in the FCOPY command in order to copy the file to tape.

```
FCOPY FROM=MYJOB1.PUB.ACCTx;TO=*T
```

Exercise 4-3: Lesson 1 Review

1.

	Tasks	Within Group	Within Account	Acct to Acct	Device to Device
a.	Release file before copying.			X	
b.	Define the device file(s).				X
c.	Fully qualify the file names (*name.group.account*).			X	
d.	Backreference device file names with asterisks.				X
e.	Require only the FROM file name and the TO file name.	X			
f.	Be in the destination account when you issue the FCOPY command.			X	
g.	Secure the file after copying.			X	
h.	Note the group names for the source and destination file locations.		X		

2.

```
FCOPY FROM=DEPTCEN.CLASS.FINANCE
;TO=DEPTCEN.PROJECT;NEW
```

3.

```
FILE T;DEV=TAPE;REC=-80,,F,ASCII
FCOPY FROM=MYFILE1.CLASS.ADVUSER;TO=*T
```

4. The correct sequence follows:

Step	Action
b.	Release the file.
d.	Log on to the destination account.
c.	Enter the FCOPY command.
e.	Use LISTFILE to check the copy transfer.
a.	Log on to the source account and secure the original file.

********** End of Exercise 4-3 **********

Lesson 2 Storing and Restoring Files from Tape

Q 4-5 To store all the files in your account to tape you would enter:

```
STORE @.@;*T
```

Q 4-6 To store all files on a tape to be restored to an MPE V system, you would enter:

```
STORE M@.@;*T;TRANSPORT
```

Q 4-7

```
RESTORE *T;@.PROJECT.ACCTx
```

Exercise 4-4: Lesson 2 Review

STORE and RESTORE problems:

1. The tape name needs to be backreferenced in the STORE command:

   ```
   STORE @.@.ADVUSER;*T
   ```

2. Check the tape drive densities and system types. The problem may result from different tape densities or different systems or both. If either of these conditions exist, you need to go through the Store procedure again, adding the DEN= and/or TRANSPORT parameters as necessary.

3. To store the files:

   ```
   FILE T;DEV=TAPE;DEN=1600
   STORE @.PUB.ADVUSER;*T;TRANSPORT
   ```

   To restore the files:

   ```
   FILE T;DEV=TAPE
   RESTORE *T;@.PUB.ADVUSER
   ```

4. This is one of the potential problems resulting from using the PURGE parameter. Use it cautiously. In the event of tape problems, check with system management for help with the tape. Also, check with them for the availability of system backup tapes made recently in order to recover copies of the files.

********** End of Exercise 4-4 **********

Solutions to Module 5: User Commands

Check Your Answers

Is Your Answer ... ?	If Not, Start With
1. TRUE	Lesson 1
2. FALSE	Lesson 1
3. An abbreviation for User-defined command.	Lesson 1
4. a. UDC executes. b. The system command executes.	Lesson 2
5. `HELP UDCname`	Lesson 3
6. `SETCATALOG`	Lesson 3
7. `SETCATALOG UDCfile;APPEND`	Lesson 3
8. `OPTION NOPROGRAM`	Lesson 4
9. `OPTION RECURSION`	Lesson 4
10. a. The file `MONEY` prints out. b. The files `DOLLARS` and `CENTS` print out.	Lesson 5
11. It executes a `LISTFILE` of the specified file using the specified option (-1,0,1,2,3,4,6).	Lesson 5
12. The command interpreter lets you toggle between EDIT/3000 and the operating system so that you can edit and test programs or command files.	Lesson 6

Lesson 1 UDCs and Command Files

Exercise 5-1: Lesson 1 Review

1.

	Characteristics	Command Files	UDC Files
a.	Must be cataloged.		X
b.	Uses commands and parameters	X	X
c.	Command definitions must be separated by asterisks.		X
d.	Can easily be converted to the other.	X	X
e.	Executes the `OPTION LOGON`.		X
f.	Can be created with a text processor.	X	X
g.	Invoked by a command header.		X
h.	Invoked by a file name.	X	

2.

a. `MYCOMM1`
b. `MYCOMM2`
c. `SHOWJOB`
d. `SM, BEGIN`

3.

	Function	HELP UDC name	SHOW CATALOG	PRINT UDC file name
a.	List all UDC files currently cataloged for an account.		X	
b.	List the contents of a specified UDC.	X		
c.	List the contents of a specified UDC file.			X

********** End of Exercise 5-1 **********

Lesson 2 Understanding Search Priority and Search Path

Q 5-1
a. user UDC SM
b. account UDC LF
c. user UDC TRYIT

Q 5-2 The UDC file executes.

Q 5-3 The UDC MYFILEX executes.

Exercise 5-2: Lesson 2 Review

Search path and execution priority problems include:

1. system command LISTFILE
2. program file DELFILE in your current group
3. MPE/iX command PRINT
4. command file MYLIST

********** End of Exercise 5-2 **********

Lesson 3 Cataloging UDCs

Q 5-4

a. To delete all the UDC files from the UDC directory, you would enter the following command:

```
SETCATALOG
```

b. To add the UDC files MOSCOW, NAIROBI, and BRASILIA to the UDC directory without affecting other cataloged files, you would enter the following command:

```
SETCATALOG MOSCOW,NAIROBI,
BRASILIA;APPEND
```

c. To replace existing UDC files in your UDC directory with the following files: NEWYORK, BOSTON, BOMBAY, and ZURICH, you would enter this command:

```
SETCATALOG NEWYORK,BOSTON,
BOMBAY,ZURICH
```

d. To remove BOSTON and BOMBAY from the directory without affecting other UDC files, enter this command:

```
SETCATALOG BOSTON,BOMBAY;DELETE
```

Lesson 4 Using UDC Options

Exercise 5-3: RECURSION Option

In order to allow UDC SO to call upon and execute UDC ST (without changing the order of the UDCs in the file), you would have to use OPTION RECURSION as follows:

```
ST
SHOWTIME
***
SM
SHOWME
***
SO
OPTION RECURSION
```

```
SHOWOUT
ST
***
```

********** End of Exercise 5-3 **********

Q 5-5 To make the UDC not executable from within a program, you need to add the NOPROGRAM option:

```
LF
OPTION NOPROGRAM
LISTFILE
***
```

Q 5-6

a. SHOWTIME is displayed. The LISTFILE command is not displayed because it is preceded by OPTION NOLIST.

b. No MPE/iX commands are displayed because the OPTION NOLIST is still in effect.

c. No MPE/iX commands are displayed because the OPTION NOLIST is still in effect.

Exercise 5-4: UDC Options

1. The SLS UDC should look like this:

```
SLS
SHOWTIME
LISTFILE
OPTION LIST
SHOWME
***
```

2. The problem with the following UDC is that OPTION NOHELP can only appear directly after the header.

```
LISTER
OPTION NOLIST
FILE OUT;DEV=LP
PRINT MYFILE, OUT=*OUT
OPTION LIST
*OPTION NOHELP
LISTFILE MYFILE,2
```

3. Default values for the options are marked with an asterisk (*):

```
LIST            NOLIST*
HELP*           NOHELP
RECURSION       NORECURSION*
PROGRAM*        NOPROGRAM
```

********** End of Exercise 5-4 **********

Lesson 5
Using Parameters

Exercise 5-5: Using Parameters

1. The LS command file should look like this:

```
PARM FILE NAME,OPTION="1"
LISTFILE !FILE NAME,!OPTION
```

2a. When you enter LS without any options, the following happens:

LISTFILE,1 executes by default.

2b. When you enter LS with option 1, the following happens:

LISTFILE,1 executes.

When you enter LS with option 2, the following happens:

LISTFILE,2 executes.

When you enter LS with option 3, the following happens:

LISTFILE,3 executes.

3. The LS UDC should look like this:

```
LS FILE NAME,OPTION="1"
LISTFILE !FILE NAME,!OPTION
***
```

4. To uncatalog MYUDC3 and edit it, do the following:

Uncatalog MYUDC3:

```
SETCATALOG
```

Edit MYUDC3:

```
LS FILE NAME,OPTION="1"
LISTFILE !FILE NAME,!OPTION
***
```

Recatalog MYUDC3:

```
SETCATALOG MYUDC3
```

To execute LS, do the following:

```
LS JOB1
LS JOB1,2
LS JOB1,3
```

********** End of Exercise 5-5 **********

Solutions to Module 6: Variables and Expressions

Check Your Answers

	Is Your Answer ... ?	If Not, Start With
1.	`VAR_1` `VARIABLE_1`	Lesson 1
2.	`HPPROMPT` or `HPPATH`	Lesson 1
3.	FALSE	Lesson 1
4.	TRUE	Lesson 1
5.	`SHOWVAR VARIABLE1` or `ECHO !VARIABLE1`	Lesson 1

6.	`ECHO HELLO --- WELCOME TO THE ...`
	- *OR* -
	`SETVAR A "HELLO -- WELCOME TO THE ... "`
	`SHOWVAR A` *or* `ECHO !A`
		Lesson 2

7.	`INPUT USERNAME;PROMPT="What is your name?"` `ECHO HELLO !USERNAME -- WELCOME TO THE ...`	Lesson 2
8.	You must explicitly dereference the value of `VAR_A` by using the statements in b and c.	Lesson 2
9.	a. `CFA`: It prints: `*** HELLO JULIO FROM SPAIN ***` and quits. `CFB`: It prints: `*** HELLO JULIO FROM SPAIN ***` and reprompts. b. `CFA`: It prints `*** SORRY, NO MESSAGE ***` and quits. `CFB`: It prints `*** SORRY, NO MESSAGE ***` and quits. c. `CFA`: It prints: `*** HELLO GINA FROM ITALY ***` and quits. It never gives you a chance to enter the `//`. `CFB`: It prints: `HELLO GINA FROM ITALY ***` and quits after you enter `//`.	Lesson 2

Lesson 1 Understanding Variables

Q 6-1 The following are acceptable names for variables:

	Variable Name
b.	CIERROR
c.	FRED_FLINTSTONE
d.	HPINPRI
f.	_JAMES_BOND

Q 6-2 According to the values assigned them (by the user or the system), the variables listed below are of the specified type:

Variable		Type
HPJOBLIMIT	(number of jobs allowed to run):	integer
HPJOBCOUNT	(number of jobs currently running):	integer
RESPONSE	(user response - either Y or N):	string
FILE NAME	(user-supplied file name):	string

Q 6-3 Hint: HELP VARIABLES(Return). According to the Help facility, each of the following variables are defined.

Hint: SHOWVAR variablename(Return)

Their current values depend on your system:

Variable	Current Value	Definition
HPJOBLIMIT	7	Current job limit.
HPSESLIMIT	60	Current session limit.
HPPROMPT	:	The initial system prompt.
HPPATH	HPGROUP,PUB, PUB.SYS	Current search path
HPVERSION	X.11.50	Current version of operating system.
HPUSER	USER1	User name.
HPSESCOUNT	varies with the system	Current number of sessions.

Q 6-4 To list *all* the system-defined and user-defined variables, enter this command:

```
SHOWVAR @
```

Q 6-5 The system-defined variables in the STATS command file are the following:

```
HPJOBLIMIT
HPJOBCOUNT
```

Q 6-6 No, neither variable is user-modifiable.

Q 6-7 In the LF command file, the three user-defined variables are:

```
RESPONSE
FILE NAME
OPTION
```

Exercise 6-1: Variables

1. To define a variable called MYPROMPT and assign it the value currently stored in HPPROMPT, enter the following:

   ```
   SETVAR MYPROMPT HPPROMPT
   ```

2. To display the value of MYPROMPT, enter the following:

   ```
   SHOWVAR MYPROMPT
   ```

3. To delete MYPROMPT and verify that it is gone, enter the following:

   ```
   DELETEVAR MYPROMPT
   SHOWVAR MYPROMPT
   ```

   ```
   MYPROMPT is a user-defined variable.
   HPPROMPT is a system-defined variable.
   You cannot delete system-defined variables.
   ```

********** End of Exercise 6-1 **********

Lesson 2 Using Variables and Expressions

Q 6-8 The method of dereferencing used in the two ECHO statements of the STATS command file is *explicit* dereferencing (exclamation points are used).

Q 6-9 To give the three user-defined variables, USER, GROUP, and ACCT, the same values as those currently stored in the system-defined variables, HPUSER,HPGROUP, and HPACCT, you would do the following:

```
SETVAR USER, HPUSER
SETVAR GROUP, HPGROUP
SETVAR ACCT, HPACCT
```

Q 6-10 According to the IF-THEN statement in the STATS file, the following messages are displayed in each of the following situations:

a. The current number of jobs is 10 and the job limit is 12.

```
***APPROACHING JOB LIMIT ---
STREAM JOB LATER***
```

b. The current number of jobs is 5 and the job limit is 10.

```
***NUMBER OF JOBS IS STILL
REASONABLE --- STREAM JOB NOW***
```

Exercise 6-2: IF-THEN-ELSE Statement

1. Your IF-THEN-ELSE statement should look like this:

```
IF HPSESCOUNT >= (HPSESLIMIT*3)/4 THEN
   SHOWVAR HPSESCOUNT
   SHOWVAR HPSESLIMIT
   ECHO ***TOO MANY PEOPLE ON SYSTEM --- LOG OFF NOW***
ELSE
   SHOWVAR HPSESCOUNT
   SHOWVAR HPSESLIMIT
   ECHO *** CURRENT SESSION LIMIT NOT EXCEEDED ***
ENDIF
```

********** End of Exercise 6-2 **********

Q 6-11 As long as you respond with anything other than //, you will be prompted continually for file names.

Q 6-12 To terminate being prompted for any more file names, you must respond with //.

Exercise 6-3: WHILE Loop

1. Your CF command file should look like this:

```
INPUT RESPONSE;PROMPT="Do you want to change the prompt? (Y or N): "
   IF RESPONSE = 'Y' THEN
      SETVAR MYRESP 'N'
      WHILE MYRESP = 'N' DO
          INPUT HPPROMPT;PROMPT="Enter your new prompt (10 char. max): "
          ECHO !HPPROMPT
          INPUT MYRESP;PROMPT="Is this prompt OK? (Y or N): "
      ENDWHILE
   ELSE
      ECHO Current prompt remains in effect !HPPROMPT
DELETEVAR RESPONSE,MYRESP
```

2. When you enter your final prompt, this prompt should be displayed:

   USERx:

3. Use this command to change HPPROMPT to the initial system prompt:

```
SETVAR HPPROMPT,":"
```

********** End of Exercise 6-3 **********

Glossary

This glossary defines terms found in the *Fundamental Skills* and *Advanced Skills* courses. Terms that appear only in the *Advanced Skills* course are identified by the notation (*AS*) following the definition.

abort — A procedure that terminates a program or session if an irrecoverable error, mistake, or malfunction occurs, or if the system manager requests termination.

access codes — Access codes are assigned by the system manager to accounts. Access codes are assigned by the account manager to groups and users. Access codes regulate who has the ability to read, write, append, lock, save, or execute a given file.

account — An account is a collection of users and groups. Each account on the system has a unique name. A user must specify a valid account to access the system.

account librarian capability — AL capability: Assigned by the account manager to one or more users within an account. An account librarian is allowed special file access modes to maintain specified files within the account.

account manager capability — AM capability: Assigned by the system manager to one user within each account who is then responsible for establishing users and groups.

account structure — The account structure provides organization, security, and billing for the system. It is used to allocate use of system resources such as central processor time, online connect time, and file space. The account is the principal billing entity for the use of system resources.

append — To join all or part of one existing file to the end of another existing file.

application program — An application program is a set of computer instructions that executes a specific set of tasks. Applications include spreadsheets, word processing,

graphics, database management, and data communication programs.

ASCII — American Standard Code for Information Interchange: ASCII is the standard method of representing character data (seven data bits plus one that can be used for parity). This method was established by the American National Standards Institute (ANSI) to achieve compatibility between data devices during an exchange of information. (*AS*)

attribute — Attributes enable the computer to determine what functions it will or will not allow a user, group, or account to perform. They include file access codes and special capabilities. They enable the computer to determine what functions it will or will not allow a user, group, or account to perform.

backreference — A technique of referencing a previously defined file. Backreferencing uses an asterisk (*) before a formal file designator to indicate it that has been previously defined with the `FILE` command. (*AS*)

backup — A process that duplicates computer data to offline media such as magnetic tape. Backups protect data if a system problem occurs.

batch processing — A method of submitting a job for processing. A job, which is submitted as a single entity, can consist of multiple commands such as program compilation and execution, file manipulation, or utility functions. Once submitted, no further interaction between the user and the job is necessary.

binary — A method of representing numbers, alphabetic characters, and symbols in digital computers. Binary is a base two number system that uses only two digits, 0's and 1's, to express numeric quantities. (*AS*)

Boolean — A data type with a value that is either TRUE or FALSE (binary 1 or 0). (*AS*)

byte — A combination of eight consecutive bits treated as a unit. A byte represents one letter or number. The size of memory and disk storage is measured in bytes. (*AS*)

capability — A method for determining what commands account members are allowed to execute.

	Capabilities are assigned to accounts, groups, and users to provide system security and access to the operating system. Account capabilities are assigned by the system manager when the account is created. The account manager then assigns capabilities to groups and users within the account.
carriage control characters (CCTL)	Carriage control characters determine such things as double spacing, vertical line spacing, and page ejects.
catalog	A system file containing information such as a listing of user-defined commands. A UDC file is cataloged with the `SETCATALOG` command and the information is added to the system file `COMMAND.PUB.SYS`. (*AS*)
central processing unit	CPU: A part of a computer system that interprets and executes instructions and contains all or part of internal storage.
CI	See command interpreter.
command	A system-reserved word that directs the operating system, subsystem, or utility program to perform a specific operation.
command file	A file a user creates to execute multiple MPE commands (except `DO` and `REDO`). To execute commands or UDCs referenced with the command file, enter the command file name at the MPE/iX system prompt.
command interpreter	CI: An MPE/iX program that reads command lines entered at the standard input device; interprets them; determines if they are valid, and if so, executes them.
command line history stack	See **history stack**.
communication	The ability of one computer system to access or "talk" to other computer systems by way of telecommunication devices.
compile	The process of changing a program written in a source language (for example, BASIC, C, FORTRAN) into a machine language routine that the computer can understand. The compiled routine is then ready to be loaded into storage and run.
computer	A device that accepts information, processes it, and produces an output. A computer usually contains memory, a control unit, arithmetic and logical manipulators, and a means for input and output.

configuration — The way in which a computer and peripheral devices are programmed to interact with each other.

console — A terminal, usually designated logical device 20, given unique status by the operating system. The operator uses the console to monitor and manage jobs, sessions and resources, respond to requests, and communicate with other user terminals. It is used to boot the system and receive system loader error messages, system error messages, and system status messages.

continuation character — An ampersand (`&`) character entered as the last character of a command line. A continuation character tells the command interpreter that the command is longer than one line and is continuing onto a second or subsequent lines. (*AS*)

cursor — A character, such as a flashing rectangle or blinking underline character, on a terminal screen. The cursor marks the position where text or data can be entered, modified, or deleted.

database — A collection of logically related data files, and structural information about the data.

default — A predefined value or condition that is assumed, and used, by the operating system if no other value or condition is specified.

delimiter — A special character used to mark the end of a string of characters. Common delimiters are the semicolon (;), equal sign (=), Return, or the comma (,).

dereferencing — Dereferencing substitutes the value of a variable in place of the variable name. See also **explicit dereferencing** and **implicit dereferencing**. (*AS*)

device — See **peripheral**.

device file — A file associated with a nonshareable device (a spool file). Input and output spool files are identified by a number in the `DFID` (device file identification) column of the `SHOWIN` and `SHOWOUT` command displays. A device file may also refer to any nondisk device, such as $STDIN and $STDLIST, the default input and output device files for a terminal. (*AS*)

directory — A system table defining where groups, users, accounts, and files are located. A

directory may also contain information such as file size, creation date, modification dates, creator, or security information. (*AS*)

disk drive A peripheral device that reads information from and writes information to the disk.

echo To display on the terminal screen data being typed on the keyboard. If echo is turned off, the computer receives the data but nothing appears on the screen. (*AS*)

editor A word processing application used to prepare, modify, or delete text and program files. EDIT/3000 is the text editor used with MPE/iX.

environment file A compiled disk file containing all the specifications for a printed page of data. These specifications, which are not a part of the data, may include the page size, character fonts, and forms to be used in conjunction with the laser printer. (*AS*)

error listing A report generated by the system describing the step-by step processing of the job. (*AS*)

error message A system message describing errors that occurred during either an interactive session or batch job. The messages appear on the standard list device, which is usually a terminal for a session, or a line printer for a job.

execute When a command is entered, the computer carries out the instructions or performs the routine indicated.

explicit dereferencing When the command interpreter encounters an exclamation point immediately before a variable name, it substitutes the value for the variable name. Explicit dereferencing may be used in any MPE/iX command. See also **implicit dereferencing**. (*AS*)

expression A statement consisting of variables, constants, and operators. (*AS*)

FCOPY An HP 3000 subsystem that allows the user to copy, append, translate data from one type to another (for example, ASCII to EBCDIC), verify, and compare files. The subsystem is activated with the MPE/iX `FCOPY` command. (*AS*)

file A group of related records that represent either ASCII text (text files) or binary data (such as executable code).

file equation — A method of equating a file name to a device or another file. The MPE/iX `FILE` command is used to establish the relationship of the file to the device. Generally used to direct the input or output of a program, job, or session to a particular device by referencing the device class, such as `TAPE` or `LP`. (*AS*)

file name — An MPE/iX file name is a string of up to eight alphanumeric characters, the first of which must be an alphabetic character. The file name is assigned when the file is created or first saved.

formal file designator — An "alias" file name that is used either programmatically or in a file equation to reference a file. The formal file designator is not the file name found in the system file directory. (*AS*)

fully qualified file name — A complete file description that includes the file name, the group to which the file belongs, and the account to which the group belongs. The fully qualified file name of the `LETTER` file in the `PUB` group of the `SYS` account is expressed as `LETTER.PUB.SYS`.

function keys — Special keys on the terminal keyboard that are labeled sequentially, F1, F2, F3 . . . and correspond to the windows that appear at the bottom of the terminal screen. Function keys are used to perform various activities.

group — A group is part of an account and is used to organize the account's files. All files must be assigned to a group; and within an account, each group has a unique name. Groups are the smallest entity for which use of system resources is reported. A `PUB` group is designated for each account when it is created. Additional groups are created within the account, as needed, by the account manager.

group librarian capability — GL capability: Assigned by the account manager, to a user within an account. A group librarian is allowed special file access modes for the maintenance of certain files within the user's home group.

hard reset — A method to reset the computer or a terminal. A hard reset erases all information in memory. See also **soft reset**.

hardware — All the physical components of the computer including the CPU cabinet, tape drives, disk drives, terminals, and other peripherals.

header — The first page printed when output is directed to a line printer. It contains the session name (if designated), session number, logon identification, day of the week, date, and time. It corresponds to the trailer printed as the last page of the output.

Help facility — An online utility providing information on all MPE/iX commands. Information can be accessed by topic areas or by task.

hexadecimal — A method of representing a single character with a combination of four bits. Hexadecimal also describes the base 16 number system, in which the first ten digits are 0 through 9, and the last six are A through F. A number written in base 16 is preceded by a dollar sign ($). For example, $F3 is the hexadecimal representation for the decimal number 243. (*AS*)

history stack — The history stack is a CI table that contains, by default, the 20 most recent commands entered at the system prompt during a session. The history stack is used with the `REDO` and `DO` commands. To display the commands in the history stack, use the `LISTREDO` command.

home group — A home group may be assigned to each user. If no other group is specified with the `HELLO` or `JOB` command, users are logged on to their home group by default. If no home group is assigned, the user must always specify a group when logging on. The account manager assigns the home group when a user's name is first defined.

HP Desk — HP Desk is Hewlett-Packard's electronic mail product.

IF-THEN-ELSE statement — A statement used to determine what action occurs. When the IF condition is true, the THEN action is performed. When the IF condition is false the ELSE action is performed. (*AS*)

implicit dereferencing — A way of substituting the value of a variable in place of the variable name. Implicit dereferencing may be used with the `CALC`, `IF`, `SETVAR`, and

	WHILE commands. See also **explicit dereferencing**. (*AS*)
input priority	A number in the range of 1 (lowest priority) to 14 (highest priority) assigned to input jobs. The input priority can be assigned by the system (default is 8) or by the user. Jobs with an input priority less than or equal to the system jobfence (default 7) are deferred. (*AS*)
integer	A data type that is either a positive or negative whole number, or zero. (*AS*)
integer value	A sequence of digits preceded by a +, -, $, or % sign. When neither a + nor - sign is provided a positive number is assumed. A $ indicates a hexadecimal integer and a % indicates an octal integer. (*AS*)
interactive	An interactive session allows users to enter commands and data at the terminal and receive an immediate response. Sessions are useful for data entry and retrieval, text editing, or program development when direct dialog with the computer is preferred.
job	A job is a method of submitting multiple operating system and utility commands for processing with a single command. Once submitted, the job executes independently of the user's session. Jobs are used to compile source programs, modify files, or perform other functions that do not require user interaction. See also **batch processing** and **stream**.
jobfence	A limit established to manage jobs. If a job has an input priority higher than the jobfence, it executes. If it has an input priority less than or equal to the jobfence, it does not execute. (*AS*)
job file	A job file is used to define a job to the system. It must start with a JOB command and end with an EOJ command. (*AS*)
job limit	A limit set to manage jobs. The system manager or operator can restrict system usage by limiting the number of jobs allowed to run on the system. If the LIMIT command is used to set the job limit to 0 (zero), no additional jobs can log on to the system.
job listing	See **listing**.

job number A system assigned identification number given to each job when it is submitted for processing.

keyboard A keyboard is attached to a terminal and used to input data to communicate with the system.

keyword A word assigned a specific meaning by the operating system, a subsystem, computer language, or utility.

K file A recovery file created by EDIT/3000, with a name in the form `K#######`, where the first three character `(ddd)` show the Julian day, and the next four `(hhmm)` show the time in hours and minutes when work began on the files. A new K file is created every time a new editor file is created or an existing file is loaded for editing. If a system problem occurs, the data in the new or loaded file is saved to the K file for recovery purposes.

laser printer A hardware device used for system output. A laser printer prints output one page at a time.

LDEV number See **logical device number**.

line editor A line editor requires you to press Return to end one line of text and to begin another. EDIT/3000 is an example of a line editor.

line printer A hardware device used for system output. A line printer prints output one line at a time.

link To merge a compiled file and its libraries to create an executable file that allows a program to run.

listing A listing is the output of a job usually in the form of a printed document.

local mode A standalone method of terminal operation. A terminal is operating in local mode when it is not connected to the computer. See also **remote mode**.

lockword A word used as a security device on files. A lockword can be assigned to a file when it is created or renamed, and must be supplied to regain access to the file. The word may be from one to eight alphanumeric characters long and must begin with an alphabetic character. (*AS*)

logical device number — An LDEV number is assigned to all peripherals of a computer system and is used for identification purposes.

log off — A method of terminating a session. To log off MPE/iX enter the `BYE` or `EXIT` command.

log on — A method of initiating a session. To log on to MPE/iX enter the `HELLO` command and a valid user and account name, plus a group name if necessary, and any required passwords.

log on identity — A security device used to verify users to the system. A log on identity includes a valid user name and account name in the form *user.account*.

log on prompt — A system prompt (`MPE XL:`) that indicates the computer is ready to initiate a session. See also **prompt**.

magnetic tape — A data storage device used to duplicate online data to offline media. The duplicated data may also be copied from the tape back to disk. MPE/iX supports the use of magnetic tape in reel form.

MPE/iX — Multiprogramming executive with integrated POSIX: The operating system for the 900 Series HP 3000 computers. MPE/iX manages all system resources and coordinates the execution of all programs running on the system.

$NEWPASS — A system-defined file. $NEWPASS is used as temporary storage for processes such as compilation. Only one $NEWPASS file may exist during a single job or session. $NEWPASS is the name temporarily assigned to any file referenced by $OLDPASS. (*AS*)

nonshareable device capability — ND capability: Assigned to accounts and users allowing account members to own nonshareable devices such as unspooled tape drives, line printers, serial disks, and volumes.

$NULL — A system-defined file. $NULL is a nonexistent file that is treated as an empty file. When referenced as an input file by a program, that program receives only an end-of-file indication upon first access. When referenced as an output file, the associated write request is accepted by MPE/iX but no physical output is actually performed. $NULL can be used to discard

unneeded output from an executing program. (*AS*)

octal Octal is a base eight number system in which digits 0 through 7 are used. One octal digit can be represented by three binary digits. Octal numbers are preceded by a percent sign (%); for example, %775 is the same decimal 509. (*AS*)

$OLDPASS A system-defined file. $OLDPASS is the name of the temporary file last closed as $NEWPASS. (*AS*)

online Data stored in memory that is updated as soon as it changes and is, therefore, constantly current and accessible.

operating system A software program that enables the computer to run. It contains programs such as basic file and I/O manipulators. All subsystems run upon the operating system.

optional parameter A parameter that is not required when entering a command. In MPE/iX reference manuals, optional parameters appear within brackets ([ ]). (*AS*)

outclass priority A value in the range of 1 to 13 used to determine if a job's error listing will print. If the outclass priority is higher than the system outfence value, the error listing will print.

outfence A number in the range of 1 (lowest priority) to 14 (highest priority) used to control access to the system printer. If a job does not have an output priority higher than the system outfence (default 7), it will not print. (*AS*)

output priority A number in the range of 1 (lowest priority) to 13 (highest priority) assigned to an output spool file either by the system (a default value) or by a user. The output priority is used by MPE/iX to determine the order in which files are be printed. (*AS*)

overwrite A method of erasing and replacing an existing file. If a file is saved under a name that already exists on disk, the new file overwrites the existing file.

paging A method to limit the amount of data appearing on the terminal screen to one full screen ("page") of information rather than having the data scroll.

parameter — A value passed to a procedure, which then uses it in calculations or to determine operations within the procedure.

partly qualified file name — A designation identifying the group in which a file resides. A partly qualified file name may be used to access a file in another group of the account to which you are logged on.

password — A password is a string of predefined ASCII characters. The system uses the string as a security device to verify the identity of a user, group, or account.

path — See **search path**.

peripheral — Hardware devices that are attached to and controlled by the computer. Peripherals include terminals, disk drives, or printers.

permanent file — A permanent file is stored on disk and has an entry identifying it in the system directory. To delete a file from the system, use the PURGE command. (*AS*)

printer — A hardware device used for system output. There are various types of printers available for use with MPE/iX, including line printers and laser printers.

program — A program is a sequence of instructions that tells the computer how to perform a specific task.

programmer — A person who writes sets of instructions (programs) telling the computer how to perform a specific task. (*AS*)

prompt — A character(s) displayed on the terminal screen, indicating that the system is ready for a command. The default system prompt is a colon (:). Other subsystems have different prompts. See also **log on prompt**.

PUB group — The public group of an account. Programs and files available to all account users reside here.

PUB.SYS — The public group of the system account. Programs and applications available to all users of the system reside here.

queue — A job management technique. Jobs waiting in a line (queue) are usually processed on a first in, first out basis or by priority, if specified. For example, the output produced by a program is generally stored on disk in a queue until a printer becomes available. As each output is printed,

	the next job in priority is selected and processed.
quiet mode	When a session is running in quiet mode, messages sent from other jobs or sessions are not displayed on the terminal screen. Warning messages from the system console override quiet mode and are displayed on the terminal screen.
record	A collection of data treated as a unit, residing in a file. (*AS*)
recursion	The ability of a procedure or function to call itself.
redo stack	See **history stack**.
reexecute	A second or subsequent execution of a command.
remote mode	A method of terminal operation. A terminal is operating in remote mode when it is transmitting to and receiving data from a remote (or host) computer. See also **local mode**.
required parameter	A parameter that is required when entering a command. In MPE/iX reference manuals, required parameters appear within braces ({ }). (*AS*)
RESTORE subsystem	A system utility program allowing the retrieval of user files from `SYSGEN` or Store tapes and writing them to disk. The `RESTORE` subsystem is activated with the `RESTORE` command. (*AS*)
save file capability	SF capability: Assigned to users and accounts allowing users to save the files that they create.
scheduling	A method of determining when a job will be processed by the computer. Jobs are scheduled using the `STREAM` command.
scroll	A way to roll data up or down on the terminal screen.
search path	The hierarchy by which MPE/iX searches the system for UDCs, commands, programs, and command files. When you enter a command at the system prompt the standard MPE/iX search path is UDCs first, MPE/iX commands second, and file names (including command files and programs) last. (*AS*)
security	MPE/iX provisions to protect the system from unauthorized use. The most basic level of security includes organizing files

	into groups and users into accounts, both of which may be assigned a password. Security also refers to the ability to read, write, append, lock, save, and execute files. Assigned to accounts by the system manager and to groups and users by the account manager.
session	An interactive way of communicating with a computer. In a session, commands are entered by way of the keyboard, and the computer responds with an action or a message to the terminal's screen. A session is initiated with the `HELLO` command and is terminated with the `BYE` command.
session limit	The maximum number of sessions allowed to log on at any given time. This is set with the `LIMIT` command. (*AS*)
session name	An optional identification method for a session. A session name is specified when logging on in the form *session,user.account*.
session number	A system assigned identification number given to each new session as it is logged on to the system.
soft keys	See **function keys**.
soft reset	A method of resetting a computer or terminal. A soft reset initializes various terminal functions but does not reset the memory. See also **hard reset**.
software	A set of computer instructions. Software programs are concerned with the operation of a computer and provide it with instructions on how to perform specific operations.
SORT-MERGE/iX	An MPE/iX utility program. The SORT program allows you to sort data alphabetically or numerically. The MERGE program allows you to combine previously sorted data into a new file.
source code	A file(s) containing the instructions of a program. It must be compiled into machine-readable data (object code) and linked before it can be executed by the computer. (*AS*)
spool file	A file on a storage device (usually a disk drive) that is either spooled from an input device or spooled to an output device. Spool Files may be in any of the following states: `OPEN`, `ACTIVE`, `READY`, `CREATE`,

PRINT, DEFER, SPSAVE, PROBLM, DELPND, or XFER. These states describe different stages of the spooling process depending upon whether the file is an input or an output spool file. (*AS*)

spooling A method of managing jobs. Multiple users can send output to a nonshareable device, such as a tape or line printer, and their output is directed to spool files on disk. The output is printed on a priority basis as the printer becomes available. Users can proceed with other processing activities without waiting for the printer.

standalone See **local mode**.

$STDIN A system-defined file name. $STDIN refers to the standard input device used to initiate a session or job; usually a terminal or tape drive. (*AS*)

$STDLIST A system-defined file name. $STDLIST indicates the standard job or session listing file corresponding to the particular input device being used. The listing device is usually a printer for batch jobs and a terminal for sessions. (*AS*)

STORE subsystem An HP 3000 subsystem used to save files to magnetic tape. The Store subsystem is executed by using the STORE command. (*AS*)

stream A method of running a batch job. A batch job is initiated from a session or a job by using the STREAM command. Once a job is streamed, it executes as a separate process without any further user input or supervision.

string value A sequence of characters surrounded by quotation marks. (*AS*)

subcommand A command performed under another command. For example, the EDIT/3000 MODIFY command enables you to use the D (delete), I (insert), and R (replace) subcommands.

subsystem A system supported utility or program. MPE/iX subsystems include FCOPY, SORT-MERGE/XL, and EDIT/3000.

syntax A set of rules defining the structure of a language, instruction, or command.

SYS account A special account on the 900 Series HP 3000 that is included with the system when it is first installed. It contains all

	the files for system supported subsystems, utility programs, and compilers.
system console	See **console**.
system-defined files	Files defined by MPE/iX and made available to all users. They indicate standard input or output devices, special temporary files, and files opened for output that do not perform an actual write operation. (*AS*)
system-defined variables	Command interpreter variables are used to store system assigned information. MPE/iX system-defined variables usually begin with the letters "HP". They have a string, integer, or Boolean value. (*AS*)
system manager capability	SM capability: A user-assigned system manager capability is responsible for installing the computer, creating accounts, and assigning capabilities and resource limits to each account created.
system operator	A user associated with the system console. The system operator monitors the system console and uses it to manage jobs and sessions, store and restore data, and perform system backups.
system processing unit	SPU: The SPU contains all the components of the computer system including the central processing unit. SPU does not refer to the system console or any other peripheral devices. See also **central processing unit**.
system prompt	See **prompt**.
tape density	Tape density is measured in bits-per-inch. Hewlett-Packard supported tape densities are 800, 1600, and 6250 bits-per-inch. (*AS*)
tape drive	A hardware device used to store and restore data from disk to magnetic tape and from magnetic tape back to disk.
temporary file	A file that exists only for the duration of a session or job. There is no entry in the system directory for a temporary file. (*AS*)
terminal	A hardware device consisting of a keyboard and a display screen. It is used for entering data to and receiving data from the computer.
trailer	The last page printed when output is directed to a line printer. It contains the session name (if specified), session number, log on identification, day of the week, date, and time. It corresponds to

	the header printed as the first page of a document.
truncate	To cut off or shorten data. If too many characters appear on a line, they may not all be recognized by the system or printed as output.
user	A person logged on to the computer. Each user is identified by a user and account name. A user logs on to a session and can access files in the logon group.
user command	A user command is a UDC or command file created to execute one or more MPE/iX commands. (*AS*)
user-defined command	UDC: A file, or portion of a file, containing user-defined commands. A UDC can be used to execute one or more specified MPE/iX commands to perform a specific task. The header (name) of the UDC is defined before the commands are listed and is then used to execute the UDC.
user-defined variable	Used to store user-assigned information. Variables can be used to assign a string value, an integer value, or a Boolean value. (*AS*)
utility	A utility is a system program that performs specific functions such as copying files, sorting or merging data, memory dump analysis, or monitoring available disk space.
variable	Variables can be system-defined or user-defined. They are specified with the `SETVAR` command and the values they contain may be constant or dynamic. Variable names are a string of one or more alphanumeric characters beginning with an alphabetic character. (*AS*)
variable dereferencing	Variable dereferencing refers to the act of substituting the value of a variable in place of the variable name. See also **explicit dereferencing** and **implicit dereferencing**. (*AS*)
VERSION	A subsystem utility used to display the attributes of an executable file.
VOLUTIL	Volume Utility: A subsystem that allows for the management of volume sets (disk drives).
Warn message	A message sent from the system console to all users. A Warn message interrupts all

	sessions on the system, including those running in quiet mode.
Welcome message	A message created by the system operator that is displayed each time a user logs on to the system. It usually contains a greeting and important system information.
WHILE loop	A statement used to determine what action occurs. When the conditions are true, the action is performed repeatedly in a continuous loop. A WHILE statement is terminated with an ENDWHILE statement. (*AS*)
wildcard character	Wildcard characters are used to replace a character or set of characters. MPE/iX uses the pound sign (#), the at sign (@), and the question mark (?) as wildcard characters. Other subsystems may use different symbols as wildcard characters.
word	A word consists of 32 bits (4 bytes) of information on the 900 Series HP 3000 (MPE/iX). A word consists of 16 bits (2 bytes) of information on other HP 3000 systems (MPE V). (*AS*)
word processor	A word processor is a utility program that supports the creation, modification, or deletion of letters, memos, reports, and other written documents.

Index