A Beginner's Guide to

PIRATE

CONTENTS

1. INTRODUCTION

PIRATE is the Purdue Interactive Remote Access Terminal Environment system. It provides file creation and editing capabilities, remote batch job submission, and macro execution facilities. It is one of the subsystems available to a terminal user through the PROCSY system. The reader is referred to PUCC document LO-PROCSY for a description of how to log on to the PROCSY system and enter the PIRATE subsystem.

This document will describe the basic PIRATE facilities which will enable a user to create, submit, and retrieve jobs as well as editing, and storing of files from a terminal. Although this document discusses only a small subset of the PIRATE facilities, it should provide enough information to enable a user to use PIRATE effectively with a minimum of "learning time".

1.1 Command Prompt

Upon entering the PIRATE subsystem, a message like the following will be printed:

```
PIRATE VER. 2.29  08/02/73.  15.13.16.   0
+++
```

The "+++" is the PIRATE command prompt. It is typed (by PIRATE) to indicate that PIRATE is ready to accept any command (some of the basic commands are explained in this document).

1.2 Logging Off

Upon completion of a session at a terminal, the user must log off. This is done simply by typing the following command:

```
+++LOG
```

The user should note that all local files not saved will disappear. All jobs left in the batch system will complete execution (if not already completed) and will remain in the print queue for an unspecified short period of time.

1.3 Full Duplex/Half Duplex Mode Toggling

Normally when a character is typed at a terminal, the character is also printed at the terminal. On terminals operating in full duplex terminal mode, it is possible to cause the characters typed to be transmitted to the computer but not printed at the terminal. To do this, one must toggle the transmission duplex mode from full duplex to half duplex by typing

+++QTYPE,,HALF or +++QTY,,H

To then toggle the transmission duplex mode back to full duplex, one types

+++QTYPE,,FULL or +++QTY,,F

1.4 Command Syntax

In the following sections when describing the syntax of PIRATE commands square brackets ([and]) will be used to enclose optional parameters and separators. For example, the syntax for the DISPLAY command is

+++DISPLAY,filename[,first line[,last line]][,SUP]

One should note that in this example,

1) the file name must be given,

2) the optional parameters must be preceded by a separator as denoted by the comma,

3) a last line may be specified only if a first line is specified.

In the illustrations below of the use of this syntax, let SAMPLE be the name of the file, 3 be the first line number, and 17 be the last line number.

```
+++DISPLAY,SAMPLE
+++DISPLAY,SAMPLE,SUP
+++DISPLAY,SAMPLE,3,SUP
+++DISPLAY,SAMPLE,3,17
+++DISPLAY,SAMPLE,3,17,SUP
```

2. FORTRAN ROUTINES

This section explains the use of the PIRATE command FORTRAN. This command may be used to create, correct (edit), and execute FORTRAN programs. The user may then wait for the job output to return automatically to his terminal.

2.1 Generating and Executing a FORTRAN Program

A general form of the command to generate and execute a FORTRAN program is (note that user-supplied information is underlined):

```
+++FORTRAN
NEW OR FIX?  NEW
FILE-NAME?  filename
```

where filename is a user-chosen name which will be used to identify the program and its data. This name may be any alphanumeric character string up to seven characters in length, the first of which must be alphabetic.

An example of the use of this command is:

```
+++FORTRAN
NEW OR FIX?  NEW
FILE-NAME?  TEST

ENTER FORTRAN PROGRAM.
IF YOU HAVE DATA, TYPE #EOR TO SEPARATE PROGRAM FROM DATA.
TYPE #SEND TO SEND JOB FOR EXECUTION.

COLUMN:   1     7
   1.0000=      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
   2.0000=      WRITE (6,1)
   3.0000=    1 FORMAT('THIS IS A TEST')
   4.0000=      READ (5,2) X
   5.0000=    2 FORMAT(F10.0)
   6.0000=      Z = 2.0 - SIN(X)
   7.0000=      WRITE (6,3)  Z
   8.0000=    3 FORMAT(1X,E15.6)
   9.0000=      STOP
  10.0000=      END
  11.0000=#EOR
  12.0000= 0.543
  13.0000=#SEND

WAIT FOR JOB "Q0516" OR TYPE CTRL-B.

THIS IS A TEST
    1.483293E+00
```

```
.TRMNL,      ,ZZZ,CM45000,T8,P50.
.**
.FUN(S,0=0)
. CTIME 000.120 SEC. FUN MOD LEVEL 60H
.LOAD(LGO,RUNLIB)
.EXECUTE.
.CX       .367 SEC., NL  10100 WORDS
.STOP
.EXIT.
.CP      .388 SEC., IO       229 UNITS.
.LINES      2
.TC        3 TRACKS USED (MAX)
.CM      .056 MWD-SEC.,  FL  10100 WORDS
+++
```

The line numbers in the left-most columns are generated by PIRATE and are used only for editing files.

The #EOR is a special PIRATE word which means 'end-of-record' and it is equivalent to the 7-8-9 multi-punch on card input.

The #SEND is a special PIRATE word which is a signal that the input is terminated.

There are 72 print positions per line on the TTY and NCR terminals. The maximum length of line which can be typed in is 128 characters long. Since the PIRATE-generated line number takes ten characters, after typing the 62nd character of a line an automatic carriage return will be generated allowing the rest of the line to be typed in.

All output lines are truncated to a maximum of 72 characters and the printer format control character is preserved on each line.

The output following the results is called the day-file and it contains accounting information regarding the execution of the job.

At any time after #SEND has been typed, CTRL-B may be typed to obtain the "+++" immediately. If the job output had not yet started to print out,

```
+++ FORTRAN,GET,,job-name
```

may be used to obtain the job output.

2.2 Editing Directives

There are several directives which may be used to edit lines in a file, a few of which will be discussed here. For a description of more of the editing operators, see the section on "Editing a Local File". The general format of a subset of the editing directives is:

```
#OPERATOR,n,i
```

where:

OPERATOR is one of the following:

R or REPLACE	-	replace a single line
I or INSERT	-	insert a single line
A or ADD	-	add a block of lines
N or NORMAL	-	terminate the ADD operation
D or DELETE	-	delete a single line or a block of lines

n is a line number, where the decimal point is assumed to be at the right if not specified. It is not used in the #NORMAL directive.

i is the second argument used in #ADD and #DELETE directives:

- for #ADD, it is an increment signifying the amount to be added to each succeeding line number until the last line has been added.

- for #DELETE, it is a line number signifying the last of a block of lines to be deleted.

Some examples are:

#REPLACE,10	or	#R,10	- replace line number 10
#INSERT,3.1	or	#I,3.1	- insert line number 3.1
#ADD,10.1,0.1	or	#A,10.1,0.1	- add a block of lines, beginning with line number 10.1, then 10.2, then 10.3, etc. until #NORMAL is typed.
#DELETE,11,19	or	#D,11,19	- delete lines numbered 11 through 19

Note:

- The #ADD operation must be terminated by a #NORMAL rather than a "CTRL-B" or else the operation may be only partially completed.

An example of using an editing directive during the creation of a FORTRAN program is

```
1.0000=      PROGRAM MAIN(INPUT,OUTPUT,
2.0000=     1TAPE5=INPUT,TAPE6=OUTPUT)
3.0000=      REED%%AD (5,1)  I
4.0000=    1 FORMAT(I5)
5.0000=      X = I
6.0000=      DO 2 I=1,10
7.0000=      WRITE (6,2)  X
8.0000=    2 FORMAT(1X,G14.6)
9.0000=#REPLACE,6
6.0000=      DO 3 I=1,10
#ADD,9,1
9.0000=    3 X = 10.0*X
10.0000=     END
11.0000=#EOR
12.0000=   10
13.0000=#STOP
```

2.3 Editing and Resubmitting a FORTRAN Program

A general form of the command to edit (change) and resubmit a FORTRAN program is:

```
+++FORTRAN
NEW OR FIX?  FIX
FILE-NAME?  filename
```

An example using this is:

```
+++FORTRAN
NEW OR FIX? FIX
FILE-NAME? TEST

DO YOU WANT TO SEE LOCAL FILE "TEST"? YES
   1.0000=      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
   2.0000=      WRITE (6,1)
   3.0000=    1 FORMAT('THIS IS A TEST')
   4.0000=      READ (5,2) X
   5.0000=    2 FORMAT(F10.0)
   6.0000=      Z = 2.0 - SIN(X)
   7.0000=      WRITE (6,3)  Z
   8.0000=    3 FORMAT(1X,E15.6)
   9.0000=      STOP
  10.0000=      END
  11.0000=#EOR
  12.0000= 0.543
  13.0000=#EOR
ENTER CHANGES.
TYPE #SEND TO SEND JOB FOR EXECUTION.

COLUMN:   1     7
#DELETE 2,3
#REPLACE 8
   8.0000=    3 FORMAT(1X,F10.8)
#SEND

WAIT FOR JOB "Q7972" OR TYPE CTRL-B.

1.48329320

.TRMNL,     ,ZZZ,CM45000,T8,P50.
.**
.FUN(S,0=0)
. CTIME 000.121 SEC. FUN MOD LEVEL 60H
.LOAD(LGO,RUNLIB)
.EXECUTE.
.CX      .368 SEC., NL  10100 WORDS
.STOP
.EXIT.
.CP     .379 SEC., IO       229 UNITS.
.LINES      1
.TC       3 TRACKS USED (MAX)
.CM     .056 MWD-SEC.,  FL  10100 WORDS
+++
```

The answer to the question "DO YOU WANT TO SEE LOCAL FILE TEST?" could have been either "YES" or "NO". A "NO" answer would have caused the display of the program and data to be omitted.

Any of the editing directives could have been typed following the "#".

#SEND must be typed to terminate the changes and start execution.

3. LOCAL FILE MANIPULATION

This section explains how to create, display, edit (or change), and delete local files. A local file is not permanent—it vanishes when the terminal is logged off. It may be saved in PFILES storage, thereby making it permanent.

3.1 Creating a Local File

A general form of the command used to create a local file is

```
+++CREATE,filename
```

where "filename" is the name of the local file to be created. The name may contain up to seven alphanumeric characters, the first of which must be alphabetic.

An example of the use of this command is

```
+++CREATE,TEST2
   1.0000=T,12345,XYZ,CM50000,T8,P50.
   2.0000=**
   3.0000=FORTRAN.
   4.0000=LGO.
   5.0000=#EOR
   6.0000=      PROGRAM TWO(OUTPUT)
   7.0000=      X = ATAN(1.0)
   8.0000=      FOFX = (X-2.0)*X**2
   9.0000=      PRINT 1,  X, FOFX
  10.0000=    1 FORMAT(1X,2(E14.6,2X))
  11.0000=      STOP
  12.0000=#STOP
+++
```

In this example, the local file created is a FORTRAN job starting with a job card and ending with the STOP statement of a FORTRAN program. Actually this example contains an error because the last line of any FORTRAN program must be the END statement. Later the EDIT command will be discussed in which errors like this can be corrected and this file will be used as an example.

Notice that the jobcard contains an extra parameter at the beginning of the jobcard line. This parameter may be any alphanumeric string so long as it is there. If it is omitted, a jobcard error will result.

Any of the Purdue MACE control cards may be used when creating a file. Therefore the CREATE instruction permits the terminal user to generate and submit all jobs that a card-deck user might submit.

The CREATE command only creates a local file — it does not submit the file for execution. To submit local files for execution, see the section on Transmitting a Job to the Batch System.

During creation of a file, all of the PIRATE editing directives may be used. #STOP or #S must be typed at the beginning of a line to tell PIRATE that the file creation is to be terminated.

3.2 Displaying a Local File

The general form of the command to display a local file is

+++DISPLAY,filename[,first line[,last line]][,SUP]

where "filename" is the name of the local file to be displayed

"first line" is the number of the first line to be displayed. If null, the whole file will be displayed.

"last line" is the number of the last line to be displayed. If null, only the "first line" will be displayed.

"SUP" denotes that line numbers should not appear in the display.

An example of the use of this instruction is

```
+++DISPLAY,TEST2,SUP
T,12345,XYZ,CM50000,T8,P50.
**
FORTRAN.
LGO.
#EOR
      PROGRAM TWO(OUTPUT)
      X = ATAN(1.0)
      FOFX = (X-2.0)*X**2
      PRINT 1,  X, FOFX
    1 FORMAT(1X,2(E14.6,2X))
      STOP
+++
```

Notice that #STOP is not a line in this file. Once again the user is reminded that the END statement must be the last statement of any FORTRAN program.

3.3 Editing a Local File

The general form of the command to edit a local file is

+++EDIT,filename

where "filename" is the name of the local file to be edited.

There are a number of editing directives, some of which were discussed in the previous section entitled "Editing Directives" in chapter 2. Before we discuss the editing directives, it is important to discuss one point.

The PIRATE editor keeps a "current line" pointer which is changed upon successful completion of an editing operation. If, for example, one uses the #INSERT,12.1 directive, upon successful completion of that operation the "current line" pointer would be set to 12.1.

Now back to the narrative. Probably the most useful single editing directive has the form

```
[n[,i]]=  (any text)
```

where n is the (first) line number
i is the line number increment for subsequent lines
(any text) is to be inserted as line n.

This directive combines the functions of the REPLACE, INSERT, and ADD operators. One may note that omitting both the n and the i parameters causes the "current line" to be changed.

Let us illustrate the use of this directive.

```
+++EDIT,TEST2
#3.5,0.5=MAP(PART)
   4.0000=LOADX(LGO,RUNLIB2)
   4.5000=#12=      END
#S
+++
```

The first directive requested that the text following the "=" should be inserted as line 3.5 and that subsequent lines be added with line number incremented by 0.5. The second directive (given on line 4.5) caused the insertion of line 12.

The editing operators below may be used with two alternative directive formats. The alternative forms of the directives are

```
#OPERATOR[,n[,i]]
      or
#[n[,i]]OPERATOR
```

where OPERATOR is one of the following:

R	or	REPLACE	- replace a single line
I	or	INSERT	- insert a single line
A	or	ADD	- add a block of lines
N	or	NORMAL	- terminate the ADD operation
D	or	DELETE	- delete a block of lines
P	or	PRINT	- print a block of lines (with line numbers)
S	or	STOP	- terminate editing

n is a line number (the decimal point is assumed to be at the right if not specified). It is not used with the NORMAL or STOP operators.

i is the second argument used with ADD, DELETE, and PRINT

- with ADD, it is an increment signifying the amount to be added to each succeeding line number. The default value for i is 1.
- with DELETE and PRINT, it is a line number indicating the last of a block of lines to be deleted or printed.

Note that if no line number parameter is given, the "current line" is implied.

The following few examples illustrate the use of the alternate directive forms:

```
#7,26D   or  #D,7,26
#6A      or  #A,6
#1,100P  or  #P,1,100
#12.6I   or  #I,12.6
```

Two of the PIRATE editing operators are context-oriented; that is, they are used to find and change particular strings of characters within a local file. The FIND operator is used to find the first occurrence of a given character string. The CHANGE operator is used to find the first occurrence in the "current line" of a given character string and replace it with another given character string. The scanning of the file starts with the line after the "current line" and proceeds "down" the file until either the character string is found or the end of the file is reached.

The format of the directive using the FIND operator is

$[n_1[,n_2]]$F,'string'[col]

or

$[n_1[,n_2]]$FIND,'string'[col]

where n_1 is the line number after which the scan of the file is to start (n_1 may be zero).

n_2 is the line number of the last line to be scanned.

string is the string of characters to be found.

col is the line column where the left-most character of "string" is to be found; if null, all columns of each line are scanned.

If the given character string is found, the "current line" pointer is set to the number of the line in which it occurs and the line is printed at the terminal. If the character string is not found, the message "NO SUCH LINE" is printed.

The format of the directive using the CHANGE operator is

[n]C,'$string_1$'[col]='$string_2$'

or

[n]CHANGE,'$string_1$'[col]='$string_2$'

where n is the number of the line in which the character string change (substitution) is to be made. If not present, the "current line" is used.

$string_1$ is the string of characters which is to be changed.

$string_2$ is the string of characters which is to replace "$string_1$".

col denotes the column where the left-most character of "$string_1$" is to be found; if null, the whole line is scanned.

If "$string_1$" is found, the character string "$string_2$" is substituted in its place. If it is not found, the message "NO SUCH SUBSTRING" is typed.

Since the CHANGE operator works on a single line, either the user must specify the line number or the "current line" is used. The #FIND directive is used to find the desired string and to set the "current line" pointer.

The following example illustrates the use of the PRINT, FIND, and CHANGE operators:

```
+++EDIT FILE
#1,100P
   1.0000=THIS IS A LOCAL FILE WHICH WILL BE USED TO
   2.0000=ILLUSTRATE THE CONTEXT-EDITING FEATURES OF
   3.0000=THE 'PIRATE' EDITOR.
   4.0000=THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
   5.0000=NOW IS THE TIME FOR ALL GOOD MEN TO COME TO THE
   6.0000=AID OF THEIR COUNTRY.
   7.0000=E PLURIBUS UNUM.
#OF,'X'
   2.0000=ILLUSTRATE THE CONTEXT-EDITING FEATURES OF
#F,'BROWN'
   4.0000=THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG.
#C,'BROWN'='RED'
   4.0000=THE QUICK RED FOX JUMPED OVER THE LAZY DOG.
#F,'NU'
   7.0000=E PLURIBUS UNUM.
#S
+++
```

The #1,100P directive was used to display the local file to be edited. The #OF,'X' directive was used to find the first occurrence of the character X in the file. Notice that zero was used to cause the scan to start with

line 1. The #F,'BROWN' directive was used to find the character string BROWN. Since no line number was given in the directive, the scan started with the line after the "current line". Note that the "current line" was line 2. The #C,'BROWN'='RED' directive caused the first occurrence of the character string BROWN to be replaced with the character string RED. Since no line number was specified, the "current line" was used. The #F,'NU' directive found the first occurrence of the character string NU as the file was scanned starting after the "current line" (which was line 4). Notice that NU was found in the word UNUM.

3.4 Deleting a Local File

The general form of the command to delete a local file is

```
+++RELEASE,filename
```

where "filename" is the name of the local file to be deleted.

An example of the use of this command is

```
+++RELEASE TEST

REMAINING LOCAL FILES:
 TEST2    I    13.0000
```

Note that the remaining local files are listed along with two characteristics of the files. The characteristics are format of the file (either internal (I) or external (E)) and the "current line" number of the file.

3.5 Listing Local File Names

The form of this instruction is

```
+++FILES
```

An example of using this instruction is

```
+++FILES
ACTIVE LOCAL FILES

 TEST2    I    13.0000

FILE INDEX

  INPUT       INPUT
  PIRMAC     *PERM CM
  TEST2       LOCAL
+++
```

PIRATE keeps only a small number of files (currently 5) in an active status at any one time. The files are kept in a file stack which is updated at every file reference, keeping the most-recently used file name at the top. When space is needed for a new file, the bottom file is removed from active file status but still remains a local file to the terminal.

The FILES command causes the names of the active local files to be listed followed by a FILE INDEX which lists the name and type of all files associated with the terminal.

In the example above, the only active local file is TEST2. The file INPUT (of type INPUT) provides terminal communication. The file PIRMAC (of type PERM CM) contains the PIRATE command macroes.

4. JOB MANIPULATION

This section is concerned with entering jobs into the batch system and retrieving job output from the batch system. In order to submit a program for execution, it must first be a local file.

4.1 Transmitting a Job to the Batch System

A general form of the command to transmit a job to the batch system is

+++XMIT,jobname,filename[,filename]

where "jobname" is a user-chosen name which will be used to identify the program's output. It may be any alphanumeric character string whose first character is alphabetic. Only the first four characters are significant. Never choose a jobname the same as any existing filename.

"filename" is the name of a local file which is to be sent to the batch system. The first named file must contain a control card record as its first logical record.

If line 2.0 of the first-named file contains an asterisk in column 1, XMIT will replace the line with a password card and then send the job. It then changes line 2.0 to "**" so that you can safely display your control cards. If, on the other hand, an asterisk is not found or you don't have a password, XMIT leaves the file alone and sends it.

The purpose of the automatic password-card insertion is to help users to protect their passwords from onlookers and file-snoopers. It also allows users to keep files containing control cards in their PFILES storage without having to use READ keys to protect their passwords. Even if you don't have a password (you really should), it's a good idea to have line 2.0 of your control card files to contain "**" so that your lack of a password won't become general knowledge.

An example of using this command is

```
+++XMIT,MINE,TEST2
JOB "MINE" SENT
+++
```

This command only transmits a job to the batch system. It does not retrieve job output.

When a job is submitted to the batch system, a batch jobname is formed by appending the specified jobname to the user's three-character user id. In the example above, if the terminal is logged on with user id of XXX, the batch jobname is XXXMINE.

4.2 Obtaining Status of Jobs

A general form of the command to obtain the status of all jobs submitted by a user is

```
+++STATUS[,SUP]
```

An example of using this command is

```
+++STATUS
 STATUS AT  15:31:20
   XXXMINE IS EXECUTING  ( 345), PRIORITY = 6306
     CM =  15000, TIME USED =     0 SEC
          RFL(50000)

 ACTIVE LOCAL FILES:

  TEST2    I   13.0000
 +++
```

Including the optional "SUP" parameter will delete the listing of active local files.

The information given in the status response gives the current status of the jobs submitted by the user (assuming the terminal is logged on with the user's user id). The job status may be that it is in the input queue, in execution, rolled out, in the print queue, in the punch queue, or not there at all. The number in parentheses on the first line specifies the file name table entry number of the job. The priority given is the queue priority of the job (see ZO-QP for an explanation of queue priority). If present, the second line gives the current job field length (CM) and the number of seconds the job has run so far. The third line displays the current control card being processed.

Since there are problems when two jobs of the same name are submitted by the same user, the STATUS command should be used before choosing any jobname. In this way potential job name conflicts may be avoided.

4.3 Displaying Job Output at the Terminal

When a job is in the print queue, it may be displayed at the terminal or printed on a line printer or "thrown away" (purged). The PREVIEW command causes the job dayfile and job output to be printed at the terminal while still saving it in the print queue so it may later (at the user's command) be printed on a line printer. The form of the PREVIEW command is

```
+++PREVIEW,jobname
```

where "jobname" is the name of the job as specified in the XMIT command.

As seen in the example below, PREVIEW displays the job dayfile and then the job output. One may notice that the job dayfile is also printed with the job output. One may suppress printing of the second dayfile by hitting a 'CTRL'-'B' when it starts to print.

```
+++PREVIEW MINE

JOB DAYFILE:

**
FORTRAN.
 CTIME 000.104 SEC. FUN MOD LEVEL 60H
LGO.
CX       .226 SEC., NL    5400 WORDS
STOP
CP      .233 SEC., IO        201 UNITS.
LINES      14
TC        3 TRACKS USED (MAX)
CM      .081 MWD-SEC.,  FL    5400 WORDS

JOB OUTPUT:

1
1                PROGRAM TWO(OUTPUT)
 000002          X = ATAN(1.0)
 000004          FOFX = (X-2.0)*X**2
 000007          PRINT 1,  X, FOFX
 000017        1 FORMAT(1X,2(E14.6,2X) )
 000017          STOP
 000021          END
OPROGRAM LENGTH INCLUDING I/O BUFFERS
 001074
OUNUSED COMPILER SPACE
 007100
0
#EOR
   7.853982E-01   -7.492275E-01
#EOR
#EOF
#EOR

1          XXXMINE.    07/31/73.PURDUE MACE 07/22/73.
R
 16.38.00.T,      ,XXX,CM50000,T8,P50.
 16.38.00.**
 16.38.00.FORTRAN.
 16.38.03. CTIME 000.104 SEC. FUN MOD LEVEL 60H
 16.38.04.LGO.
 16.38.04.CX       .226 SEC., NL    5400 WORDS
 16.38.04.STOP
 16.38.04.CP      .233 SEC., IO        201 UNITS.
 16.38.04.LINES     14
 16.38.04.TC       3 TRACKS USED (MAX)
 16.38.04.CM      .081 MWD-SEC.,  FL    5400 WORDS
#EOR
+++
```

4.4 Printing and Punching Job Output

Job output in the print (or punch) queue may be printed on a line printer (or punched on the card punch) by using the following command:

+++PRINT,jobname[,destination]

where "jobname" is the name of the job as specified in the XMIT command

"destination" is "AT,ENAD" or "AT,KRAN"
to have the output printed and to be picked up at ENAD or room 760 Krannert, respectively.

An example of the use of this command is

```
+++PRINT,MINE
AT 16:42:21  07/31/73
  XXXMINE  = PR001EB
+++
```

Job output may be picked up in room B22 in the basement of the Mathematical Sciences building or at ENAD or in room 760 of the Krannert building. It will be filed under the identification provided by the PRINT command. In the example above, the job output would be identified as PR001. The two suffix characters are the terminal identifier characters and have no other purpose or effect.

If job output includes punched cards, they may also be obtained in the same place as the list output. The punched card job identification is the same PR number.

4.5 Purging Job Output

Jobs in the print queue which the user has PREVIEWed and does not want to have printed on the line printer should be purged from the print queue. The form of the command to do this is

+++PURGE[,$jobname_1$][,$jobname_2$]$\cdots$[,$jobname_n$]

where "$jobname_i$" denotes the XMITted jobname of the job to be purged from the print queue.

If no jobname is specified, then all of the user's terminal-submitted batch jobs in the print and punch queues will be purged.

Typing

+++PURGE,LOGOFF

causes all of the user's jobs in the print and punch queues to be purged and the terminal to be logged off.

5. SEMI-PERMANENT FILE MANIPULATION

A user's semi-permanent files are stored in his PFILES storage space. They are semi-permanent in that if a given file is not used within a 15-day period, it will be automatically purged from PFILES storage.

This chapter describes the various PFILES functions the user may perform from PIRATE. Further documentation may be found in PUCC document QO-PFILES.

5.1 Saving a Local File in PFILES Storage

A local file may be copied into semi-permanent file storage by using the following command:

```
+++PUT,filename[/wkey[/rkey]]
```

where "filename" is the name of the local file to be stored

"wkey" is the file write key.
"rkey" is the file read key.

The following example illustrates how local file TEST2 is stored with a write key of WWW and read key of RRR:

```
+++PUT,TEST2/WWW/RRR
DONE   FILE SIZE     24 WORDS
         1 FILE(S) COPIED
+++
```

The read and write keys are file protection keys. They may be formed with any alphanumeric string of characters up to four characters in length. To replace a file, one must specify the proper write key. To retrieve a file, one must specify the proper read key.

A file which is PUT remains as a local file.

Names of semi-permanent files must be unique within any one PFILES storage area. The names may be a maximum of seven alphanumeric characters in length, the first of which must be alphabetic.

5.2 Retrieving a PFILES File

To retrieve a file in semi-permanent file storage and make it a local file, one uses the command

```
+++GET,filename[/rkey]
```

where "filename" is the name of the file in PFILES storage. The file will be made a local file of the same name.
"rkey" is the file read key specified when it was first placed in storage.

If the read key in the GET command is incorrect for the file, the file will not be retrieved and an error message will be printed at the terminal. A file which is retrieved by a GET command is copied from the user's PFILES storage and is made a local file--it also remains in PFILES storage.

An example of the use of the GET command is

```
+++GET,TEST2/RRR
DONE     1 FILE(S) COPIED
+++
```

5.3 Replacing a PFILES File

To place a new edition of a file in PFILES storage, one simply uses

```
+++PUT,filename[/wkey]
```

where "filename" is the name of the file to be replaced
"wkey" is the write key specified with the initial storage of the file.

The following example illustrates this:

```
+++PUT,TEST2/WWW
DONE    FILE SIZE    25 WORDS
          1 FILE(S) COPIED
+++
```

5.4 Deleting a PFILES File

To delete a file from PFILES storage one uses the command

```
+++DELETE,filename[/wkey]
```

where "filename" is the name of the file to be deleted
"wkey" is the write key specified with the initial storage of the file.

When a file is deleted from semi-permanent file storage, it just "goes away". It does not become a local file to the terminal.

An example of the use of this command is

```
+++DELETE,TEST2/WWW
  DELETED TEST2
+++
```

5.5 Listing PFILES Filenames

To obtain a list of files in one's semi-permanent file storage, one uses the command

```
+++INDEX
```

An example of the use of the INDEX command is

```
+++INDEX

1  FILE INDEX AT  15.35.58  07/30/73 XXX
0  SPACE =   103 WDS, LIMIT =    640, USE CNT =    8
   GET CNT =    4,     PUT CNT =   4, ERR CNT =    0

     FILE      SPACE    GET   PUT   DAYS   FILE
     NAME    IN WDS     CNT   CNT   IDLE    AGE

   TEST2        25       1     2      0      4
   DATA         21       3     1      1      3
   SOLVE        57       0     1      2      2

+++
```

In the heading, SPACE and LIMIT refer to the number of words of storage in use and available, respectively. The various counts (CNT's) are self-explanatory and are totals. For each file in storage, its name, size, number of retrievals (GETs), number of PUTs, number of days idle, and age are printed. When the number of DAYS IDLE reaches 15, the file is automatically purged.

6. MORE PIRATE INFORMATION

This chapter describes how one may obtain further information about PIRATE.

6.1 Syntax

The SYNTAX command may be used to display the proper syntax for a PIRATE command. To use it, one types

```
+++SYNTAX, command name
```

where "command name" is the name of the command whose syntax is desired.

An example of the use of the SYNTAX command is

```
+++SYNTAX,PUT
PUT [(OPTION)],FILE[/W-KEY[/R-KEY[/ID]]],[FILE(S)]
+++
```

6.2 Help

To obtain help when in need of information, the following command may be used:

```
+++HELP, keyword
```

An example of the use of the HELP command is

+++HELP,HELP

HELP:

'PIRATE' IS A SELF DOCUMENTING SYSTEM IN THAT BASIC DOCUMENTATION FOR THE SYSTEM IS CONTAINED IN DATA FILES THAT ARE ACCESSIBLE TO USERS OF THE SYSTEM. THE COMMAND 'HELP' IS THE MEANS WHEREBY A USER MAY GAIN ACCESS TO THE INFORMATION IN THESE FILES. MUCH OF THE MATERIAL CONTAINED WITHIN THESE DOCUMENT FILES WILL BE PUBLISHED AS A REFERENCE MANUAL AVAILABLE AT THE COMPUTING CENTER.

FORMAT:

HELP COMMAND/TYPE, COMMAND/TYPE, ...

WHERE:

COMMAND = THE NAME OF A 'PIRATE' COMMAND OR PRIMITIVE OPERATOR FOR WHICH DOCUMENTATION IS DESIRED.

TYPE = AN OPTIONAL SPECIFICATION INDICATING WHAT PART OF THE DOCUMENTATION FILE IS DESIRED. PRINTOUT STARTS AT THE SECTION INDICATED AND CONTINUES TO THE END. WHEN SPECIFIED, 'TYPE' WILL BECOME THE DEFAULT FOR SUBSEQUENT COMMANDS.

DEFAULTS:

COMMAND = (MUST BE SPECIFIED)

TYPE = (COMPLETE DOCUMENTATION)

EXAMPLES:

+++HELP CREATE, EDIT, XMIT, PREVIEW, PURGE
PRINT FULL DOCUMENTATION FOR THE LISTED COMMANDS

+++HELP FORTRAN/EXAMPLES, PRINT
LIST EXAMPLES AND NOTES FOR THE 'FORTRAN' AND 'PRINT' COMMANDS

NOTES:

-IF 'COMMAND' IS OMITTED, THE USER IS PROMPTED FOR COMMANDS.

-MUCH OF THE 'PIRATE' DOCUMENTATION IS THE SAME AS THE PROCSY 1.0 DOCUMENTATION. THIS DOCUMENTATION IS SLOWLY BEING UPDATED TO REFLECT CHANGES IN THE 'PIRATE' SYSTEM.

-THE NAMES FOR THE DOCUMENTATION FILES ARE THE COMMAND NAME PREFIXED BY A 'Q'. IF THE USER HAS A LOCAL FILE BY THE SAME NAME, THE HELP COMMAND WILL NOT WORK PROPERLY.

+++

6.3 Examples

To obtain examples of the use of a PIRATE command, one uses

```
+++HELP, command/EXAMPLES
```

where "command" is the name of the command for which examples are desired.

An example of the use of this to obtain examples of the use of the CREATE command is

```
+++HELP,CREATE/EXAMPLES

CREATE:

EXAMPLES:

    +++CREATE,MYFILE

    +++CREATE

    LOCAL FILES
    JOBCA

    FILE-NAME? MYFILE
    ENTER FILE (MYFIL).
    TYPE #STOP TO TERMINATE INPUT.

NOTES:

    - IF NAME IS NOT GIVEN, A 'FILES' IS PERFORMED, THE
    USER IS PROMPTED FOR A FILE NAME AND CERTAIN IN-
    STRUCTIONAL DATA IS PRINTED.


    - SEE 'HELP QCREATE' FOR FURTHER INFORMATION.

+++
```

6.4 News

To obtain descriptions of the most recent changes to the PROCSY and PIRATE systems, one uses

```
+++NEWS,PROCSY
```

The news items are displayed in reverse chronological order from the most recent to those four months old. 'CTRL'-'B' may be typed during the "NEWS" to suppress printing of the rest of the news file.