UNIVERSITY OF QUEENSLAND

Computer Centre

WEEKLY NEWSLETTER

Date : Week ended 14 January 1971
Authorization : Director of Computer Centre

1. OPERATION

1.1 PDP-10 SYSTEM

Schedule this week : Maintenance 0700-0930
Timesharing 1000-1200, 1400-1600
Batch 1000-1630

Friday	8 January	Errors in reading data 0930-1000
		Air-conditioning plant maintenance 1000-1030
Tuesday	12 January	System loading and disk refresh 0955-1200
Wednesday	13 January	Line Printer maintenance 1000-1030
Thursday	14 January	Maintenance diagnostics 1410-1430

This week has been the first week running under the new system. It has not always been possible to keep to the notified times but it has been a fairly successful week.

1.2 GE-225 System

Normal schedule : Maintenance 0700-0900
Operations 0900-2400

2. SECOND SHIFT

As from Monday 18 January computer operations at the Centre are being extended to include a second shift. The proposed schedule will be as follows:

Timesharing 1000-1200, 1400-1600
Batch 1000-1200, 1400-2300

Although major development testing is done during week ends, the period 1200-1400 is usually reserved for short interval tests and checks. On any day that this reserved time is not required it will be made available for both Timesharing and Batch operation. Users will be advised of the availability of this time via the schedule board and messages through the terminals.

3. DATA FILES

3.1 General

All information within the PDP-10 system is kept in the form of files.

These files can be classified into two very broad categories. Many files contain programs in one form or another and as such are files which can be placed into execution by means of some 'processor program'. These comprise the first category.

The second category consists of files which are simply used for holding information to be read or written by a program. The information in these files is of such a nature that the file could not be placed into execution. These are commonly called data files.

3.2 Data Filenames

The general form of a filename is

name/processor program name

where the name is the identification given to the file by the user. and the processor program is the name of the routine necessary to place the program file into execution.

As a data file cannot be placed into execution a data filename has no 'processor program' part. It is simply an identification of the form

name

examples:

(i) 'DATA'
(ii) 'INPUT'
(iii) 'OUT'
(iv) 'SMØ1Ø'

3.3 Creation of Data Files

Data files can exist in one of two forms; as ASCII character files (which must be read and written under format control), or as binary records (read and written without format control).

In general terms data files can, at present, be created in one of the following ways.

(a) By Editor

The editor program can be used to create data files. The user simply gives the command.

```
.CREATE filename<cr>

INPUT:
```

and then types in the required data records. These records are ASCII character strings, each a maximum of 72 characters long, and each terminated by a carriage return.

When the user's program reads this file the records must be read under format control in an identical manner to reading input data cards.

example:

Suppose an input record to the data file was

1234<cr>

Reading this record with the instructions

```
      READ(1Ø, 2) M
2     FORMAT (14)
```

would give M the value of 1234
However the same record, read by

```
      READ (1Ø,4) I, G
4     FORMAT (I1, IX, F1.1)
```

would give the result I = 1
G = 3.4

(b) From Cards

If a user has a deck of cards, he can transfer these to a disk file which can then be processed as any other ASCII file.

Facilities will be available in a new version of Batch to enable a user to create a disk file from cards by running a batch job.

As an temporary, interim service until the new Batch is available, the Centre will transfer card decks to disk files for remote terminal users.

(c) By a program

A user's own program can output named data files to the disk. These files can be binary or ASCII and can be kept on the disk for later use by other programs.

3.4 Use of Data Files in FORTRAN

To be able to read or write data files with a FORTRAN program, it is necessary to have some mechanism by means of which a specific data file can be assigned to a FORTRAN logical unit number.

For example the user may wish to make the FORTRAN statement

READ (1Ø, 3) A, B, C, J

read its data records from the user's data file named DATIN. To do this the user must specify that, in this program, the file DATIN is assigned to the logical unit number 10.

In the PDP-10 system these assignments are made by means of the subroutines IFILE and OFILE. File assignments can only be made to the FORTRAN logical unit numbers 10, 11, 12 and 13. The assignment of a file to a logical unit number can be broken by the RELEAS subroutine.

(a) Subroutine IFILE

The IFILE subroutine assigns a named file to a logical unit number as an input file. This file can be read only on that unit number; it cannot be written.

The calling sequence for IFILE is as follows:

CALL IFILE (n, filename)

n is the FORTRAN logical unit number to which the file is being assigned (10, 11, 12 or 13)

filename is a literal string or a variable containing the name of the file to be assigned in ASCII.

examples:

(i)
```
CALL IFILE (1Ø, 'DATIN')
```
This assigns the file DATIN to FORTRAN logical unit number 10. Each read on unit 10 will read the next sequential record from DATIN.

While this assignment holds unit 10 cannot appear in a WRITE statement.

(ii)
```
DATA INF/'INPUT'
CALL INFILE (12, INF)
READ (12) ARRAY
```

Each execution of the READ statement will read the next sequential record from the file INPUT into ARRAY. In this example the file INPUT is a binary file.

(b) Subroutine OFILE

The OFILE subroutine assigns a named file to a logical unit number as an output file. This file can be written only on that unit number.

The calling sequence for OFILE is similar to that for IFILE, namely

CALL OFILE (n, filename)

n is the FORTRAN logical unit number to which the file is being assigned (10, 11, 12 or 13).
filename is a literal string or a variable containing the filename in ASCII.

examples:

(i) CALL OFILE (1Ø, 'DOUT')

This assigns the file DOUT to the FORTRAN logical unit number 10. Each write on unit 10 will write the next sequential record onto DOUT .

While this assignment holds unit 10 cannot appear in a READ statement.

(ii)
```
DATA NIT, INFO/13, 'DOUT'/
CALL OFILE (NIT, INFO)
WRITE (NIT, 7) ARRAY
```

Each execution of the WRITE statement will output the contents of ARRAY, under format statement 7, as the next sequential record on the file DOUT.

(c) Subroutine RELEAS

The RELEAS subroutine breaks the assignment of a named file to a logical unit number. That unit number can then be reassigned by another call to IFILE or OFILE.

The calling sequence for RELEAS is simply

```
CALL RELEAS (n)
```

where n is the logical unit number to be released.

example:

```
      DATA K, L/1Ø, 11/
      CALL IFILE (K, 'DIN')
      CALL OFILE (L, 'OUTPT')
      DO 2 I = 1, 5Ø
      READ (K, 1) M
1     FORMAT (I6)
2     WRITE (L) M
      CALL RELEAS (L)
      CALL IFILE (L, 'OUTPT')
      DO 3 I = 1, 5Ø
      READ (L) M
3     WRITE (6, 4) M
4     FORMAT (1X,16)
      CALL EXIT
      END
```

This sample program reads 50 records from the ASCII file DIN and copies them in binary form to the file OUTPT. The OUTPT file is then read back and printed.

3.5 Some Details on Data File Usage

There are a number of details to be remembered in the use of data files via the IFILE and OFILE routines.

(a) Files can only be read sequentially or written sequentially. The first READ or WRITE after the file assignment will process the first reoord of the file. Subsequent READ and WRITE statements process successive records on a file.

(b) Files can be in ASCII or binary. They should always be read in the same form as they were written.

(c) A REWIND statement on a logical unit number can be given, but it will, at present, simply act as a call to RELEAS and break the assignment of named file to logical unit number.

(d) Files cannot currently be Eackspaced. The BACKSPACE statement is ignored.

(e) One file can be used for both input and output by assigning it to two different logical unit numbers.

For example suppose the following assignments had been made.

```
CALL IFILE (1Ø, 'DATA')
CALL OFILE (11, 'DATA')
```

Information could then be read from unit 10, processed, and written to unit 11. As both input and output files have the same name, this operation will cause the file assigned to unit 10 to be superseded by that assigned to unit 11. At the completion of this operation the file named DATA will contain only the information output on unit 11. The original information on the file DATA input on unit 10 will have been irretrivably destroyed. This facility should therefore be used with care.

(f) FORTRAN logical units can be used either as input units or as output units. However a given unit cannot be assigned for both input and output simultaneously.

For example the two statements

```
CALL IFILE (1Ø, 'DATA')
CALL OFILE (1Ø, 'DOUT')
```

are invalid unless separated by a call to RELEAS.

3.6 Scratch Files

Output can be done to, (and input from) the logical unit numbers 10, 11, 12 and 13 without first assigning a named file to these units.

In this case the system will automatically assign the following default filenames.

Unit No.	Filename
10	FOR1Ø
11	FOR11
12	FOR12
13	FOR13

If the first operation on a given unit number is a WRITE, a file, named as above, will be written onto the user's disk area. If the first operation on a given unit is a READ an attempt will be made to find the file, named as above, on the user's disk area. If none exists an error message,

```
?FILENAME FORTR NOT ON DEVICE DSK
```

will result.

Once files of these names have been produced the user can treat them as ordinary files. Users should be careful not to keep files under these names as they could easily be inadvertently destroyed.

Items (a) (b) (c) and (d) noted in section 3.5 above apply to these scratch files.

3.7 Availability

These facilities have been implemented and can now be used from remote terminals.

Users are advised that whenever improved file or assignment facilities become available , there may be changes in the operation of the routines outlined above.

4. A SUGGESTION ON REMOTE TERMINAL I/O WITH FORTRAN

Two problems arise with the use of FORTRAN on the present remote terminal system:

(a) the entry of data to a program
(b) the output of a large volume of results on the Teletype

Both these problems may be alleviated somewhat by the use of data files on disk.

Input data files can be created by use of the EDITOR and read by the User's program.

Similarly for a large volume of output, the output can be written to a disk file, instead of the Teletype and the output file contents inspected with the EDITOR. Note however that the EDITOR will truncate lines of more than 80 characters.