living computers
museum+labs

CDC 6500 Survival Guide

The CDC 6000 series [https://en.wikipedia.org/wiki/CDC_6000_series] was a family of mainframe computers designed by Seymour Cray and James E. Thornton and manufactured by Control Data Corporation in the 1960s. The computers in the series were extremely fast for their time and considered to be the first supercomputers.

Living Computers [http://www.livingcomputers.org] has recently refurbished and put on-line a CDC 6500 supercomputer. The system was originally used at Purdue University from 1967 to 1989.

This document is a short guide for those interested in exploring the 6500's interactive command environment.

(Note: The primary author worked briefly with a CDC NOS computer 30 years ago, but all information below is based on study of the referenced manuals and experimentation on LC's 6500. We apologize for any inaccuracies and welcome corrections and improvement suggestions.)

Connecting

1. Start your Secure Shell (SSH) client and connect to `cdc6500@tty.livingcomputers.org` .
2. After the `USER NUMBER` prompt enter your user name (your input will be concealed by the row of Xs). You may need to type more slowly than you are used to with other systems.
3. After the `PASSWORD` prompt type your password (your input will be concealed by the row of Xs).
4. After the `RECOVER /SYSTEM:` prompt enter a subsystem name (see below) or other command. (Enter `RECOVER` to recover previously interrupted session.)

```
[CONNECTED TO MUX6676 LINE 1]


 16/08/24. 20.43.27.
 LIVING COMPUTER MUSEUM 6500.          NOS 1.3-485/11.
USER NUMBER
XXXXXXXXXXXXXXX                                          ← 2.
PASSWORD
XXXXXXXX                                                 ← 3.
PORT    :      5, TTY
RECOVER /SYSTEM:
```

Subsystems

The 6500's Network Operating System (NOS) time-sharing system, the Interactive Facility (IAF), includes a number of subsystems, similar to applications on other computer systems, to provide various functions to the user. Subsystems can be selected explicitly by command (see below), or automatically when you select a primary file.

Although the Null subsystem provides a general-purpose environment with file and system utilities, in most cases you can move directly between subsystems as needed without going through the Null subsystem.

Subsystem	**Function**
`ACCESS`	Messaging with other users connected to the 6500.
`BASIC`	Interactive BASIC language programming environment.
`BATCH`	Create and run batch jobs (non-interactive programs) or interactively run system commands. This subsystem more closely resembles modern command-line shell environments.
`EXECUTE`	Execute previously compiled programs.
`FORTRAN`	Interactive Fortran language programming environment. Uses FORTRAN Version 5 compiler[1].
`FTNTS`	Interactive Fortran language programming environment. Uses FORTRAN Extended Version 4 compiler.
`NULL`	A general-purpose environment for working with files and system status and settings (although most general commands are available in other subsystems as well).

File System

A basic understanding of the 6500 file system is necessary to use most IAF comands.

Permanent files

Permanent files are preserved by the system from their creation until their deletion by user command. Each permanent file is owned by a single user and is normally accessible only by its owner, though it is possible to make files accessible by other users as well.

Two types of files are supported:

Direct access: Used for random access and database applications.

Indirect access: Common sequential access files, including program source files.

Temporary files

Whenever an indirect access file is accessed, or a new indirect access file created, the system first makes a temporary copy of the file, to which are directed all references and updates to the file. The temporary file must be saved in order for changes to be applied to the permanent file. Unsaved changes will be discarded at the end of the user's session.

A temporary file is not created when a direct access file is opened. Changes made to a direct access file are immediately applied to the permanent file.

Primary file

The primary file is the temporary file, usually a program source file, that a user is viewing, modifying, or executing at a given time. Although a user session may simultaneously open multiple temporary files, a given user may have only one primary file at a time.

The currently selected primary file automatically determines the user's active subsystem. When a new indirect access file is saved as a permanent file for the first time, the file system records the active subsystem at the time. Then when the file is selected as primary sometime later, the system automatically activates the original subsystem.

Local files

Local files are the set of temporary files and direct access files (permanent files) currently open by a particular job or user session.

File names may be up to seven alphanumeric (upper-case) characters and may not begin with a numeric character. The following file names are reserved by the system:

INPUT	P8	SCR3
OUTPUT	SCR	SCR4
PUNCH	SCR1	ZZZZZ*
PUNCHES	SCR2	

File Commands

`CATLIST`
`CATLIST,LO=F`
	List your permanent files. The second form generates a detailed list.

`ENQUIRE`
`ENQUIRE,F`
	Prints users terminal port number, current subsystem, primary file. The second form lists user's local files

`NEW,`*file*	Create a new file with the current subsystem and select as your primary file.
`OLD,`*file*	Select existing permanent file as primary. Change subsystems if file type is different from current primary file.
`LIST` `LNH`	List primary file contents. The second form omits printing the system header.
`GET,`*file* `LIST,L=`*file*	Use GET and LIST,L together to list the contents of a file other than primary file.
`SAVE` `SAVE,`*old*=*new*	Save contents of primary file as permanent file (creates permanent file). May need to use PACK first. Note: Use only the first time you create a file. The second form saves the primary file named old in a new permanent file named new.
`REPLACE`	Replace permanent file with current contents of primary file. May need to use PACK first. Note: Use when modifying previously created permanent file.
`PACK`	Compress multiple logical records in local file into single logical record.
`PURGE,`*file*	Delete a permanent file
`RENAME,`*oldname*`,`*newname*	Change the name of a permanent file

Programming Commands

There are several ways to develop programs using NOS 1.3 on the CDC 6500. Two of them are:

(1) Using the BATCH subsystem the typical edit, compile, run development process can be used with ALGOL, BASIC, COBOL, FORTRAN 4 Extended, Minnesota FORTRAN, and Pascal. Refer to details on the BATCH subsystem, below.

(2) Using the BASIC, FTNTS, and FORTRAN subsystems. These provide environments for interactively creating, modifying, debugging, and running programs:

`BASIC`	Enter the BASIC subsystem. Uses the BASIC programming language interpreter.

`FTNTS`	Enter the FTNTS subsystem. Uses the FORTRAN Extended Version 4 compiler.
`FORTRAN`	Enter the FORTRAN subsystem. Uses the FORTRAN Version 5[2] compiler.

These three subsystems provide a development environment similar to the original Dartmouth BASIC and versions of Microsoft BASIC for microcomputers and early PCs.

Program source files

For all subsystems, program source code is stored in indirect access files. The file's subsystem flag, set when first created, indicates the file's source programming language (and compiler version for Fortran source files).

Create new source files, list and select existing source files, view contents, and save changes to source files using the file commands documented above.

Each programming subsystem includes a simple editing function for entering and modifying program source code in the primary file as follows:

To add a source code line: Type a new line sequence number followed by a space and then a program statement. Source code lines will be stored and compiled/executed in the order of line sequence numbers, so lines can be entered in any order.

To modify a source code line: Type line sequence number of the program statement to be changed followed by a space and the modified program statement.

To delete a source code line: Type the line sequence number followed immediately by the `Enter` key.

- In the BASIC subsystem, use line sequence numbers as the targets of `GOTO` and other statements.
- In the Fortran language subsystems (FORTRAN and FTNTS), the line sequence numbers are not the same as the line label numbers used by Fortran PRINT/WRITE/FORMAT, GO TO, and other statements. Line label numbers should be inserted in necessary lines between the line sequence number and program statement.
- In the Fortran language subsystems, program statement text Following the line sequence numbers may conform to or ignore standard Fortran column rules at the programmer's choice.
- In the BASIC subsystem, program statement without a preceding line sequence number will be executed immediately and are not added to the primary file. This is useful for testing statements before modifying your program. Use `PRINT` statements to display results of calculations, etc.

`AUTO`	Enter automatic line sequence number mode. Enter program source statements in sequence after the system-generated line numbers. Exit the mode by pressing `Esc` key followed by `Enter` key.
`RESEQ`	Renumber the source code lines in the primary file, maintaining the original sequence. In BASIC subsystem, target line numbers of GOTO and other statements are adjusted appropriately.

`RUN`
`RNH`
`RUN,MI=12625`
	For the BASIC subsystem, runs the primary file program. The second form omits printing of the system header. The third form may be needed to compile and run more complex programs. (The `MI` argument sets the compiler field length. As of this writing it is unknown if value 12625 is valid for all programs.)

FORTRAN subsystem example Here the FORTRAN subsystem is selected, an existing program is loaded, listed, and run.

```
FORTRAN
READY.

OLD,TEST2
LIST

 99/09/10. 21.13.26.
PROGRAM   TEST2
```

```
00100 PROGRAM TEST2(OUTPUT)
00110 INTEGER J
00120 REAL Z,W,Q,Q2,Q3
00130 Z=0.0
00140 Q=0.0
00150 DO 210 J=1,25
00160 Z=SQRT(Q)+Z
00170 W=SQRT(Q)
00180 Q2=Q*Q
00190 Q3=Q*Q*Q
00200 PRINT 190,J,W,Q2,Q3
00210 190 FORMAT(1X,I9,3(3H   ,F12.6))
00220 Q=Q+1.0
00230 210 CONTINUE
00240 W=Z/25.0
00250 PRINT 230,W
00260 230 FORMAT(1X,5HAVG  ,F12.6)
00270 END

READY.

RUN

 99/09/10. 21.13.38.
PROGRAM   TEST2

         1       0.              0.              0.
         2       1.000000        1.000000        1.000000
         3       1.414214        4.000000        8.000000
...
        24       4.795832      529.000000    12167.000000
        25       4.898979      576.000000    13824.000000
 AVG      3.225351
 END.

SRU      0.447 UNTS.

RUN COMPLETE.

READY.
```

FTNTS subsystem example Here the FTNTS subsystem is selected, an existing program is loaded, listed, and run. Note the way the FORTRAN line numbers 190, 210, and 230 as specified in addition to the editing line numbers which do not make up part of the program.

```
FTNTS
READY.

OLD,TEST1
READY.

LNH

00100 PROGRAM TEST1(OUTPUT)
00120 INTEGER J
00130 REAL Z,Q
00140 Z=0.0
00150 Q=0.0
00160 DO 210 J=1,25
00170 Z=SQRT(Q)+Z
00180 PRINT 190,J,SQRT(Q),Q*Q,Q*Q*Q
00190 190 FORMAT(1X,I9,3(3H   ,F12.6))
00200 Q=Q+1.0
00210 210 CONTINUE
00220 PRINT 230,Z/25.0
```

```
00230 230 FORMAT(1X,5HAVG: ,F12.6)
00240 END

READY.

RUN

 99/09/10. 20.42.34.
PROGRAM   TEST1

S
1   LOAD MAP - TEST1    CYBER LOADER 1.4-485  99/09/10. 20.42.37.  PAGE     1

...
    .549 CP SECONDS              22100B CM STORAGE USED     3 TABLE MOVES
...

S
         1       0.000000       0.000000       0.000000
         2       1.000000       1.000000       1.000000
         3       1.414214       4.000000       8.000000
         4       1.732051       9.000000      27.000000
         5       2.000000      16.000000      64.000000
...
```

Text Editor

Although the programming subsystems include basic functionality for entering and modifying program source code in the primary file, the NOS Text Editor program (`EDIT`) provides both advanced editing functions and the ability to edit other local files besides the primary file.

`EDIT` `EDIT`,*file* `EDIT`,*file*,*mode*	Edit the primary file with Text Editor. The second form is for editing other local files. The third form is for specifying a file mode; `N`=normal, `AS`=ASCII.
`RESET`	Reset current line pointer to first line.
`SET`;n	Set the current-line pointer by moving it +/- "n" lines relative to the current position.
`F`:/abc/	Find line with text "abc" and move the current-line pointer there.
`LIST`;n or `L`;n	List "n" number of lines from the current line-pointer (pointer does not change).
`EXTRACT`	Copies the current line to the string buffer (a.k.a. "clipboard")
`ADD`	Add one or more lines after the current line-pointer position
`ADD`*<CR>$<CR>*	Pastes the string buffer content below the current line
`DELETE`;n or `D`;n	Delete one or more lines beginning with the current line
`RS`:/abc/,/xyz/;n	Replace string "abc" with "xyz" in the current line "n" times
`END`	Exit Text Editor

Here is an example session using the `EDIT` command to create and edit a simple ASCII text file.

```
/EDIT,test,AS
 BEGIN TEXT EDITING.
? add
 ENTER TEXT.
? /This is a test file.
? It has multiple lines of text.
? This file demonstrates the edit command.
? This is line 4./
 READY.
? list;*
This is a test file.
It has multiple lines of text.
This file demonstrates the edit command.
This is line 4.
 -END OF FILE-
? add
 ENTER TEXT.
? /Pointer at line 1, we insert a new line 2./
 READY.
? list;*
This is a test file.
Pointer at line 1, we insert a new line 2.
It has multiple lines of text.
This file demonstrates the edit command.
This is line 4.
 -END OF FILE-
? set;4
? list
This is line 4.
? delete
 -END OF FILE-
? add
 ENTER TEXT.
? /This is line 5./
 READY.
? reset
? list;*
This is a test file.
Pointer at line 1, we insert a new line 2.
It has multiple lines of text.
This file demonstrates the edit command.
This is line 5.
 -END OF FILE-
? end
 END TEXT EDITING.
$EDIT,TEST,AS.
/SAVE,test
```

Batch

The BATCH subsystem provides the ability to create and submit batch jobs as well as run system commands interactively. It is probably the closest to the modern shell or command-line environment.

The typical process is as follows,

- Create a new, empty, file
- Add the program source to the file using the EDIT or XEDIT text editors
- Save the program source file to permanent storage
- Invoke the compiler for the program source language
- Correct any reported errors
- Recompile
- Replace the file in permanent storage with the revised file
- Invoke the compiled loadable-object file

COBOL Here is an example of creating, compiling, and running an example COBOL program. It waits for entry of two integer values, calulates the product and prints the result.

```
/new,sample1
```

```
/xedit,sample1,c
 XEDIT 3.0.20
?? tabs 8,12,16,20
?? deftab ;
??
 INPUT
? ;identification division.
? ;program-id products.
? ;environment division.
? ;configuration section.
? ;source-computer. CDC 6500.
? ;object-computer. CDC 6500.
? ;data division.
? ;working-storage section.
? ;01 result pic 9(9).
? ;01 mult1 pic 9(5).
? ;01 mult2 pic 9(4).
? ;procedure division.
? ;first-par.
? ;;accept mult1.  accept mult2.
? ;;multiply mult1 by mult2 giving result.
? ;;display "Answer: " result.
? ;;stop run.
?
 EDIT
??
?? quit
SAMPLE1 IS A LOCAL FILE

/save,sample1

/cobol,i=sample1

/lgo

S
? 65432
? 1065
ANSWER: 069685080

/
```

Pascal Here is an exmample of loading an existing Pascal source file, listing it, compile, run. The program displays a table of squares and square roots of the integers 1 to 20.

```
/OLD,SAMPLE3

/LIST
PROGRAM SMPL3(INPUT/,OUTPUT);
VAR
   V1,V2,V3 : REAL;
   J : INTEGER;
BEGIN
   WRITELN('                    N                   N*N                SQRT(N)');
   FOR J := 1 TO 20 DO
   BEGIN
     V1 := J;
     V2:= SQR(V1);
     V3:= SQRT(V1);
     WRITELN(V1:22:12,V2:22:12,V3:22:12);
   END;
END.

/PASCAL(SAMPLE3)
...
```

```
     0.263 CP SECS,  44276B CM USED.

/LGO
...
S
                    N                   N*N                SQRT(N)
        1.000000000000        1.000000000000        1.000000000000
        2.000000000000        4.000000000000        1.414213562373
        3.000000000000        9.000000000000        1.732050807569
        4.000000000000       16.000000000000        2.000000000000
        5.000000000000       25.000000000000        2.236067977500
...
     0.129 CP SECS,   5725B CM USED.
/
```

Messaging

Use the ACCESS subsystem to communicate with other users connected to the 6500.

`ACCESS`	Enter the ACCESS subsystem
`USER,`*userID*	Find connected user's terminal port number
`DIAL,`*port*`,`*message*	Send message to user.

Other System Commands

`DAYFILE`	List user's activity log.
`PASSWOR,`*old*`,`*new*	Change user password from old to new

Disconnecting

`BYE`
`GOODBYE`
`LOGOUT`
	End user session and disconnect

`HELLO` `LOGIN`	End user session and return to log-in screen

File Sharing

Share permanent files with other 6500 users.

`CHANGE,`*file*`/CT=PU,M=`*mode*	Give access mode *mode* to *file* to all 6500 users. Permission modes: `A`=append, `E`=execute, `M`=modify, `N`=remove perm., `R`=read, `RA`=read+append, `RM`=read+modify, `W`=write
`PERMIT,`*file*`,`*user*[=*mode*]…	Give access mode *mode* to *file* to *user*. Default *mode* is read.
`CATLIST,UN=`*user*	List permanent files in *user*'s catalog for which you have been granted access permission.
`OLD,`*file*`/UN=`*user*	Open permanent file *file* in *user*'s catalog (must have been granted access permission).

File Transfer

In order to transfer data to or from the 6500, your best option is to copy and paste. If you're having trouble pasting in programs, you may want to try using TeraTerm after altering the line delay under Setup → Additonal settings → Copy and Paste → Paste delay per line to about 250ms.

References

1. BASIC Language Version 2 Reference Manual [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/lang/basic/19980300B_BASIC_Language_Version_2_Reference_Nov74.pdf] (PDF). Control Data Corporation. Oct. 1974. Accessed Nov. 11, 2017.
2. FORTRAN Extended Reference Manual Version 4 [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/lang/fortran/60305600A_FTN_Extd_V4_Oct71.pdf] (PDF). Control Data Corporation. Oct. 22, 1971. Accessed Nov. 11, 2017.
3. Interactive Facility Version 1 Reference Manual [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/nos/60455250C_Interactive_Facility_Version_1_Reference_Aug79.pdf] (PDF). Control Data Corporation. Aug. 10, 1979. Accessed Nov. 11, 2017.
4. NOS Version 1 Reference Manual Volume 1 of 2 [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/nos/60435400C_NOS_Version_1_Reference_Manual_Volume_1_Dec76.pdf] (PDF). Control Data Corporation. Dec. 3, 1976. Accessed Nov. 11, 2017.
5. NOS Version 1 Text Editor Reference Manual [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/nos/60436100C_NOS_Version_1_Text_Editor_Ref_Mar76.pdf] (PDF). Control Data Corporation. Mar. 8, 1976. Accessed Nov. 11, 2017.
6. COBOL Reference Manual [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/lang/cobol/60191200_COBOL_Reference_Jun67.pdf] (PDF). Control Data Corporation. Jun. 1967. Accessed Sep. 10, 2019.
7. Pascal Version 1 Users Manual [http://bitsavers.trailing-edge.com/pdf/cdc/cyber/lang/pascal/60497700A_Pascal_Version_1_Users_Man_Sep83.pdf] (PDF). Control Data Corporation. Sep. 1983. Accessed Sep. 9, 2019.

Future Improvements

The author would like to make the following improvements to this document:

- Document compilation and execution of programs with FORTRAN subsystem.
- XEDIT text editor command guide.
- Batch subsystem guide continued, add FORTRAN and Pascal examples.
- Execute subsystem guide.

---

1) , 2)
Minnesota FORTRAN, similar to FORTRAN Version 5