INTERNAL DOCUMENT 290

IOS

DEACON LABORATORY

INTRODUCTION TO SHIPBOARD COMPUTING FACILITIES (UNIX SYSTEM)

T. P. LeBas

Natural Environment Research Council

INSTITUTE OF OCEANOGRAPHIC SCIENCES DEACON LABORATORY

INSTITUTE OF OCEANOGRAPHIC SCIENCES

DEACON LABORATORY

Wormley, Godalming,
Surrey, GU8 5UB, U.K.

Telephone: 0428 79 4141
Telex: 858833 *OCEANS* G
Telefax: 0428 79 3066

DIRECTOR: Dr. C. P. Summerhayes

Natural Environment Research Council

INSTITUTE OF OCEANOGRAPHIC SCIENCES
DEACON LABORATORY
INTERNAL REPORT NO. 290

INTRODUCTION TO SHIPBOARD COMPUTING FACILITIES
(UNIX SYSTEM)

T. P. LeBas

1989

DOCUMENT DATA SHEET

AUTHOR	LEBAS T.P.	*PUBLICATION DATE* 1989
TITLE	Introduction to Shipboard Computing Facilities (UNIX System)	
REFERENCE	Institute of Oceanographic Sciences Deacon Laboratory, Internal Document, No. 290, 28pp (Unpublished manuscript)	

ABSTRACT

This report is intended to be an introductory guide to aid a programmer or user on the UNIX A-B-C system. This system is available on most NERC ships and is supported by RVS.

ISSUING ORGANISATION	Institute of Oceanographic Sciences Deacon Laboratory Wormley, Godalming Surrey GU8 5UB. UK. Director: C.P. Summerhayes	*TELEPHONE* 0428 79 4141 *TELEX* 858833 OCEANS G *TELEFAX* 0428 79 3066

KEYWORDS	UNIX A-B-C Computing	*CONTRACT* *PROJECT* *PRICE*

CONTENTS Page

1. INTRODUCTION

This report is intended to provide a simple introductory guide to the use of the UNIX system. It is not intended to be highly detailed as further information is available from several sources: manuals and system-dedicated publications, and program documentations. It is hoped that this report will give the user a basic understanding of the UNIX system and the facilities the shipboard system can offer.

The report is based on the Plessey Microsystems System 68 on the R.R.S. *Charles Darwin.* A similar computer is available on the R.R.S. *Discovery* and *M.V. Farnella.* The system comprises:-

M68000 Processor Board

2 x 0.5 Mbyte RAM

Sky floating point processor

2 eight-port serial I/O boards

84 Mbyte Winchester Disc drive and controller

Floppy-disc drive and controller

Cambridge Ring LAN DMA interface

Half-inch 1600bpi standard tape drives(2) and controller

Peripherals e.g. Terminals, graphics terminals, plotters, digitiser and printer.

The UNIX operating system is a very simple and low-level programming environment. The shipboard facilities include:-

a) A 'C' compiler and debugger

b) The Shell

c) The text editors *ed* and *vi*

d) Other language processors including FORTRAN and BASIC

e) Text processing and document preparation e.g. *nroff*

f) Graphics and plotting

g) General library of programs for data manipulation

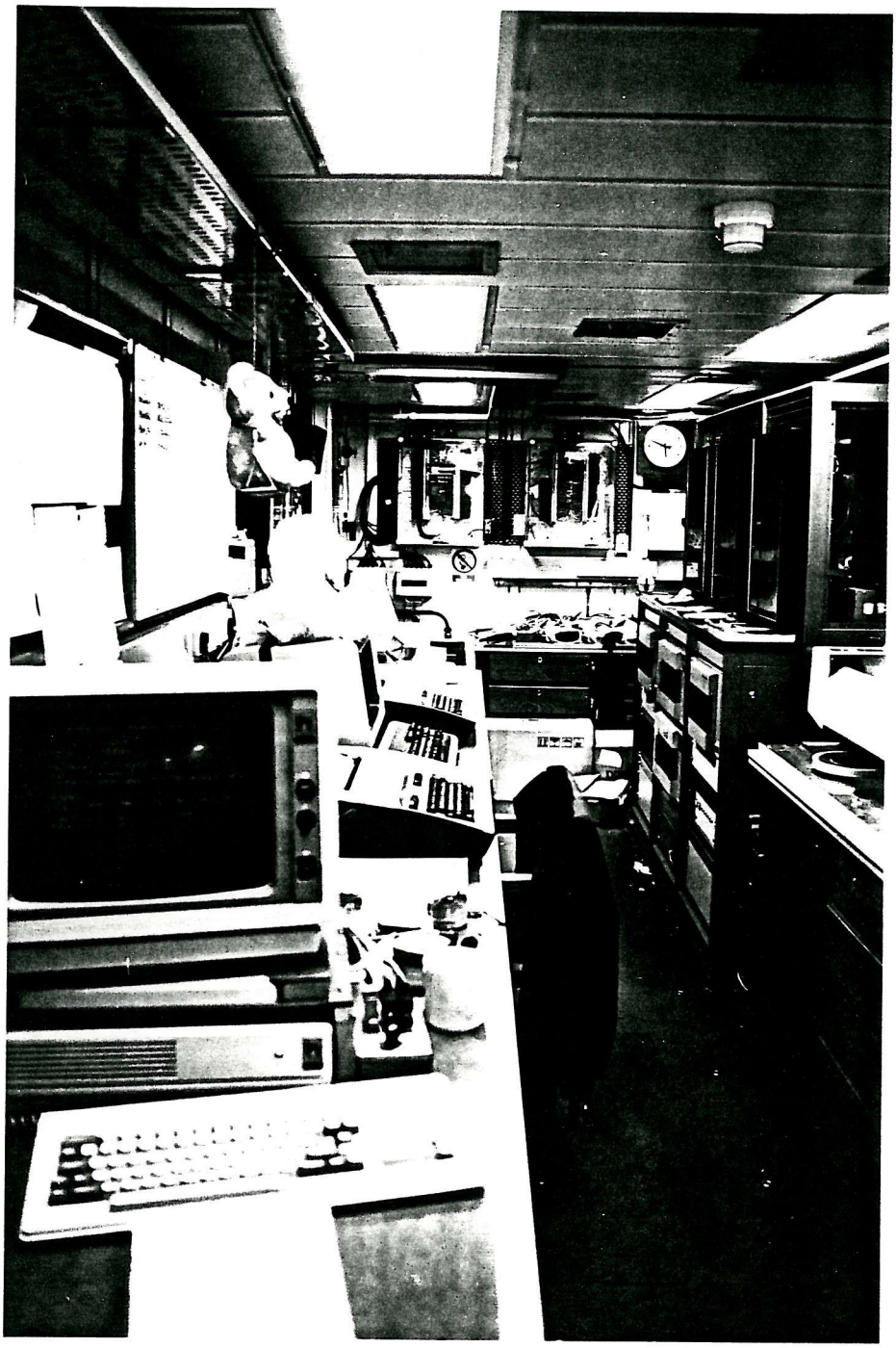


Figure 1. The computing room aboard the R.R.S. *Charles Darwin* housing the UNIX A-B-C system, with the disk and tape drives on the right hand side and some terminals opposite.

C is the programming language used to write the UNIX operating system and most of its commands. It is a very low-level programming language and consequently extremely complex. It is not discussed here since it is of little use to high-level programmers. However if the user wishes to program in this language there are many good texts for both novice and expert programmers (e.g. Kernighan & Ritchie 1978). The command level of the system is called the shell and can be a programming language itself as well as a command language. The command interpreter provides a user interface to the UNIX operating system and is similar to function calls in C.

The programming language used and explained here is Fortran. No programming techniques are given as it is assumed that the user already has an understanding of the Fortran language. However for those users who are not programmers a section is devoted to the programs already available and supported, which give various scientific calculations and plotting facilities. In addition all parameters that can be logged by the computer system are explained, though in any one cruise not all parameters will be present. Access to these data for use in the user's own programs is also covered.

As the shipboard system is under constant use, programs are constantly being updated and altered. Some of the information in this report therefore may be outdated though should be retrievable if the newer programs do not serve the user's exact requirements.

2. THE UNIX SYSTEM

2.1 System Commands

This section covers some of the basic commands of the UNIX system. Logging in is the first problem and will generally be different according to circumstances. The user may be either given a personal username and password to sign onto the system or given access to a communal username (such as 'guest'). The main UNIX operator or 'superuser' has general system administrative responsibility and can provide whichever type of i.d. the user requires.

Once you have logged in a prompt will appear, with any messages of the day. The prompt can vary considerably from just a $ to a serialised command number and word (e.g. 1GO>). Learning a new computer system in the abstract is difficult and so the best way to learn how to use the UNIX system is to relax in front of a connected terminal and experiment.

Three useful control codes to mention at the outset are:-

the ERASE character is	Del or Control-H
the LINE KILL character is	Control-U
the BREAK character is	Control-C

2.2 Directories

In the UNIX system there are two types of files, namely directories and ordinary files. These are linked in a tree structure (Figure 2). A directory is a file which in itself does not take any space though may contain other files and directories which will. A directory can therefore be regarded as a branching point within the file structure. Thus any file which is owned by the user may appear to be in the base level arrived at when logging in, but may be already several branches down the tree. The following commands can show how to move around these directories and their use.

pwd	Prints current working directory
cd <option>	Changes directory

Examples:

cd	Changes to logon directory
cd model	Changes down to directory 'model'
cd ..	Changes up to parent directory
cd /nerc/work/model	Changes to given directory
mkdir <name>	Creates a directory within the current directory.
rmdir <name>	Removes an empty directory

2.3 Files

Any name may be given to a file (or directory) though it is often good practice to use standard names. Standard format for a file is filename followed by a dot and followed by the filetype (e.g. 'xxxxxx.xxx'). Various names are suggested here as the UNIX software from time to time requires special filetype markers. It is also recommended that the user only uses alphanumeric characters as some commands can use file patterns.

name.for	Fortran programs
name.dat	Data files
name	Files for execution
name	Directories
name.c	C programs
name.bak	Backup or Archive files
name.txt	Text files
name.o	Compiled C programs
name.obj	Compiled Fortran programs

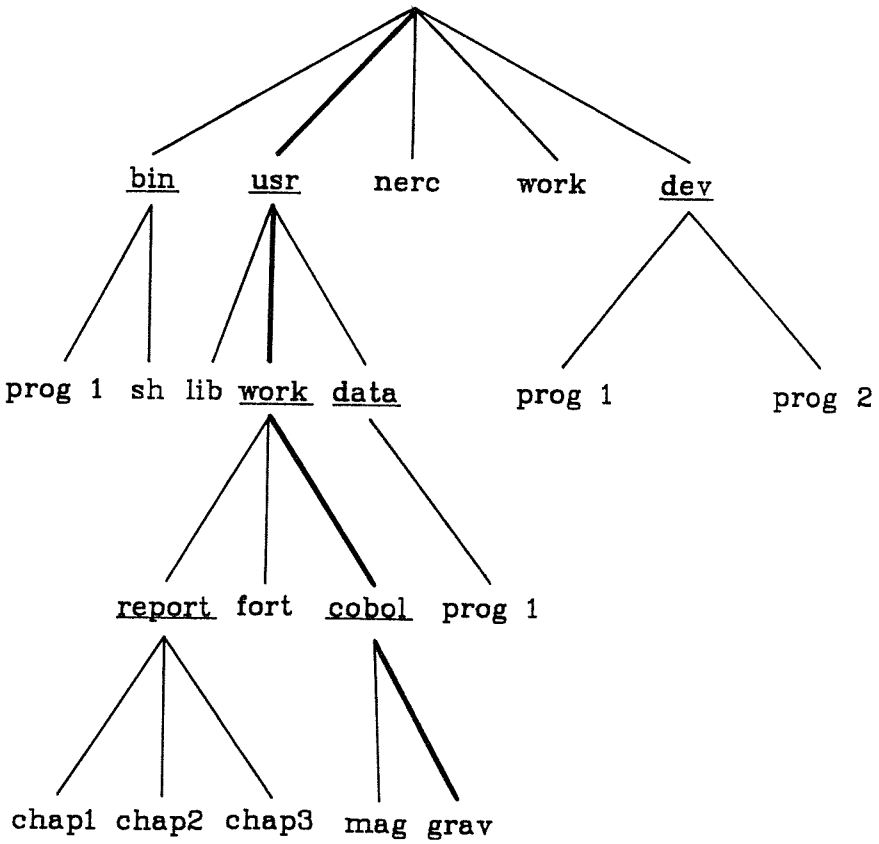


Figure 2. Example of a hypothetical file directory tree structure. Underlined names are directories and non-underlined names are files. Thus file 'grav' has a full name of /usr/work/cobol/grav.

Creation of files is done via the editor (see Section 4) and their execution is generally done by typing the filename. There are no formal run or exec commands or statements.

File control is relatively straightforward and the commands are as follows:-

ls	Lists the current directory
ls -l	Lists the current directory giving full file information e.g. directory permissions, ownership, size, last modification time
cat <filename>	Lists file contents
more <filename(s)>	Lists file(s) contents with a scroll stop every 24 lines (space to continue)
lp <filename(s)>	Lists file(s) contents on line printer
mv <file1> <file2>	Renames (moves) file1 to file2
cp <file1> <file2>	Copies file1 to file2
rm <filename(s)>	Delete (removes) file(s)
sd <filename(s)>	Selective delete of files (from filename pattern)

2.4 Filename patterns and wildcards

File patterns and wildcards are generally used to catenate commands when working with a library of files. A name is used which has a common feature with the other parts masked by wildcards or 'metacharacters'. There are three metacharacters:-

*	any number of characters
?	any single character
\	ignore the meaning of the next character (e.g. if it is another metacharacter)

Examples:

rm ?race.*	Would delete trace.for and brace.obj but not fastrace.for nor tracer.for	<2.4.1>
more ray*t.?	Would list 'ray*t.c' and 'ray*t.d' but not 'rayit.c'	<2.4.2>

Output can be directed in various ways. Default output is generally directed to the terminal. But this can be changed by giving the command a directive path and thus putting normal output to a file. Two possible path directions are:-

> file	Output is written to the file named
>> file	Output is appended to end of the file named

Examples:

cat test2 >>test1	appends test2 to test1	<2.4.3>
ls -l >dirlist	saves a long directory listing in file 'dirlist'	<2.4.4>

2.5 Repeating commands

When typing in long commands it is sometimes annoying to keep repeating the line when, say, only minor changes are required. The UNIX system allows for this with a 'history' command. This command lists the last 20 commands of the present session. These commands are numbered and are thus referencable and so can be rerun. The history commands are:-

history or h	Prints the last 20 lines of commands
!!	Reruns the previous command
!n	Reruns the command numbered n
!cde	Reruns the last command which begins with 'cde'
^cde^fg	Changes 'cde' in previous command to 'fg' and then rerun

Another way to keep repeating commands is to make a command file. Creation of a command file is in the usual way via the editor. Execution however is only possible by having the mode changed. This can only be done by the owner of the file who must use the chmod command. There are 3 levels of user: owner, group and public. Within these 3 levels there are 3 types of permission: read, write and execute. This therefore gives 9 possible permission tracks. Normally the owner is the only person who requires read and write access but it may be required that all users have execute access, this would give a permission mode of rwx--x--x or 711. Figure 3 shows how to calculate the mode for the various permissions. The syntax is as follows:-

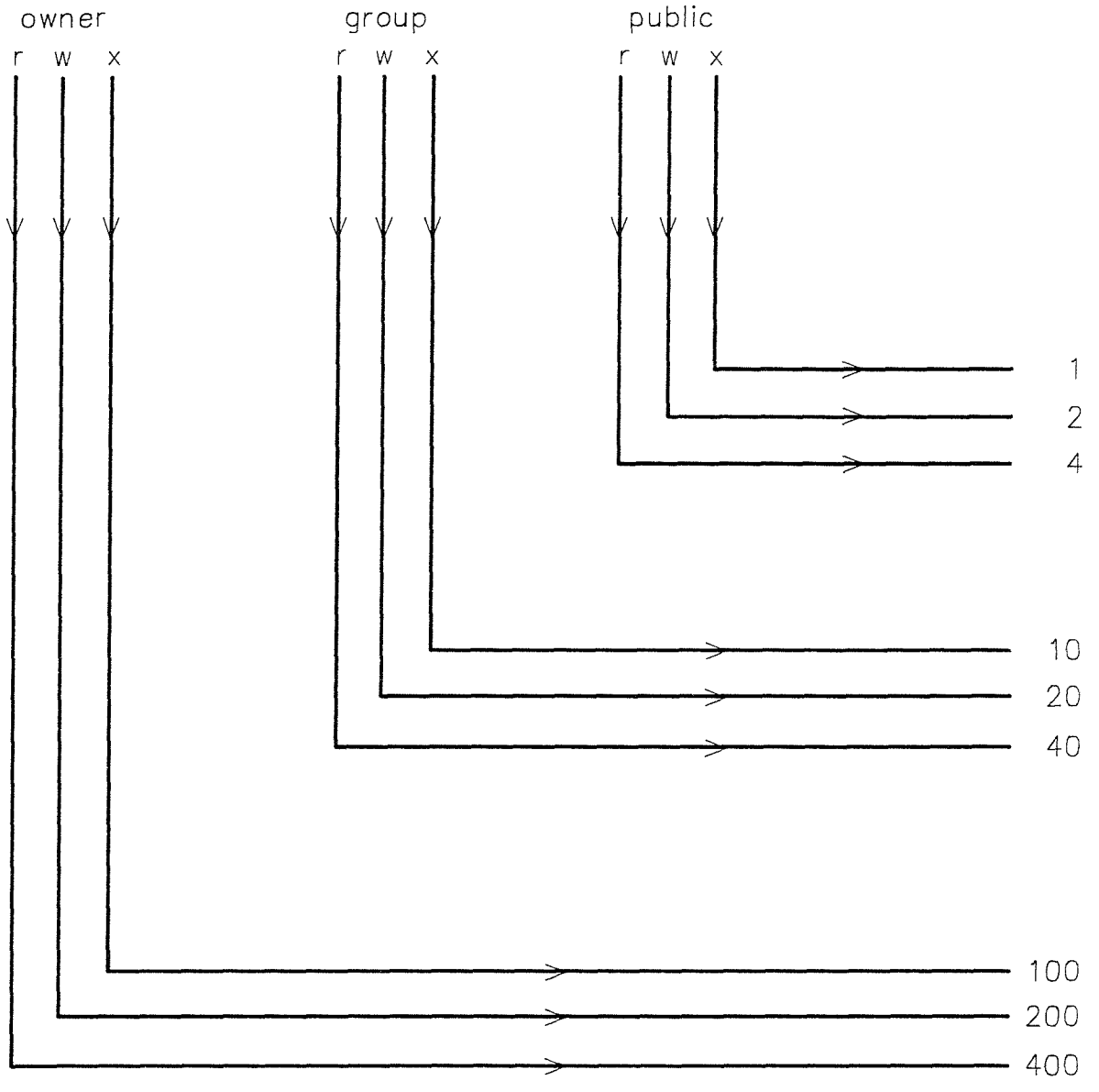
chmod <mode> <file>	Changes the mode of the file

Example:

chmod 777 test	Gives full access to everyone on file 'test'	<2.5.1>

2.6 External and Foreign Tapes

Due to the simple programming environment it is relatively easy to recover programs and/or data from foreign or external tapes and dump them onto file. The 'dd' command copies and converts the source to a file from tape (or vice-versa if required). The syntax for this command is:-



This example is mode for owner read, write and execute permissions together with group and public execution permissions.

	Mode		Permissions
	400	-	owner read
	200	-	owner write
	100	-	owner execute
	10	-	group execute
	1	-	public execute
mode =	711		

Figure 3. Translation diagram for chmod permissions.

dd <option=value>

options:

if=file		Input file (or device)
of=file		Output file (or device)
ibs=n		Input block size n byte (default 512 bytes)
obs=n		Output block size n bytes
cbs=n		Conversion buffer size
skip=n		Skip n records before cop
conv=	ascii	Convert EBCDIC to ASCII
	ebcdic	Convert ASCII to EBCDIC
	ibm	Slight different map of ASCII to EBCDIC (for IBM)
	lcase	Alphabetics to lowercase
	ucase	Alphabetics to uppercase
	swab	Swap every pair of bytes
	noerror	Do not stop processing on an error

Example:

dd if=/dev/rmt0 of=mtdx ibs=800 cbs=80 conv=ascii,swab <2.6.1>

This will read a raw EBCDIC tape blocked in ten 80 byte EBCDIC card images per record into the ASCII file 'mtdx'

Example 2.6.2 shows a step-by-step process for transferring a working Fortran program 'mtxrot.for' from a non-UNIX machine to a onboard A-B-C UNIX computer.

i) Copy the Fortran source program from the non-UNIX machine to a magnetic tape

ii) Note the tape format (e.g. 800 bpi, EBCDIC, 72 characters per record, 16 records per block, first file on tape, byte swapped)

iii) Transfer the tape to the ship and load on the tape drive 0

iv) Read the tape to disk with the command:

dd if=/dev/rmt0 of=mtxrot.for ibs=1152 cbs=72 conv=ascii,swab

v) Check the source program for any differences in programming that may need changing (e.g. Input/Output streams and/or Graphics)

vi) Compile and link the program: f77 mtxrot.for

vii) Run the program: mtxrot <2.6.2>

3. PROGRAMMING ON THE UNIX SYSTEM

3.1 Fortran

Fortran is one of the most widely used scientific languages to date. The UNIX system implementation is the American National Standards Institute (ANSI) standard FORTRAN-77. This section is a user's guide to Fortran execution as opposed to a Fortran reference manual (Metcalf 1985). But some graphics and input/output routines are included since these are a common area for error and are usually machine-dependant. To run a Fortran program it will first have to be entered into memory either from the editor or from some external source such as disc or tape. After this it must then be compiled and linked so that an executable form can be created, 'an executable module'. Compiling, linking and run-time error numbers are not listed here; firstly the error message is usually explicit and secondly the Fortran language manual covers them adequately (Unisoft systems, Fortran, Appendix A).

3.2 Compiling

The syntax of the compiler is:-

f77 <option> <filename>

options :	-c	to get object module (.obj)
	-o ofile	to get executable module called 'ofile'

Examples:

f77 -c test.for	produces object module from program 'test.for' called 'test.obj' ready for linking	<3.2.1>
f77 trial.for	produces executable module called 'trial' from program 'trial.for' - no linking required	<3.2.2>
f77 -o tr1 trial.for	produces executable module called 'tr1' from program 'trial.for' - no linking required	<3.2.3>

Examples 3.2.2 and 3.2.3 automatically link to the ordinary Fortran libraries but not the graphics libraries. Thus with programs requiring graphics the linking process must be executed separately. When compiling many subroutines in different files it is usually good programming practice to compile each file on its own and subsequently join them with the link. This should also help in the debugging of programs at compilation stage.

3.3 Linking

This is required to link to the Fortran libraries, any graphics libraries needed and possibly some external user subroutines not included in the main program. This produces an executable module. There are two commands to this process, 'ulinker' (UNIX object code formatter) and 'cc' (C compiler). The syntax for ulinker is :-

ulinker listingfile outputfile inputfile(s)

The listing file is not necessary and can be suppressed
by using the option -l
The output file must be present and of type '.o'
The input files must include the Fortran libraries as well
as any special subroutines or libraries - they must all be of type '.obj'

The syntax of cc is:-

cc <options> <filename(s)>

options: -o output Names the final executable output file
Default is a.out

Examples:

ulinker -l test.o test.obj ftnlib.obj paslib.obj spisubs.obj cc -o test test.o <3.3.1>

This links compiled program test.obj to the Fortran libraries
and to the graphics library and then creates an executable
module called 'test'. No listing file is produced.

ulinker trial.lst trial.o trial.obj ftnlib.obj paslib.obj
cc trial.o <3.3.2>

This links compiled program trial.obj to the Fortran libraries,
creates a link listing 'trial.lst' and creates an executable module
called 'a.out'.

Examples 3.3.1 and 3.3.2 assume that ulinker, ftnlib.obj, paslib.obj and spisubs.obj are in the current directory. This is not always the case and thus example 3.3.1 could look like this:-

/usr/lib/ulinker -l test.o test.obj/nerc/work/guest/fort/spisubs.obj
/usr/lib/ftnlib.obj /usr/lib/paslib.obj
cc -o test test.o <3.3.3>

3.4 Execution

The syntax of executing a program once it has been compiled and linked is:-

filename [| filter] [<input] [>(>)output]

options:	| filter	The pipe symbol | followed by an appropriate filter will translate plotting such as a graphic display, plotter or printer
	<input	Will take default input from file (instead of the terminal)
	>output	Will send default output to file (instead of the terminal)
	>>output	Will append default output to end of file (instead of the terminal)

Examples:

trial	Runs program 'trial'	<3.4.1>
trial >trial.out	Runs program 'trial' and all output goes to file 'trial.out'	<3.4.2>
trial <trial.in	Runs program 'trial' taking all input from file 'trial.in'	<3.4.3>
test | ca1039	Runs program 'test' and outputs the plot on the calcomp 1039	<3.4.4>
/nerc/geoph/mag2d	Runs program 'mag2d' which is not stored on the current directory but in directory '/nerc/geoph'	<3.4.5>

NOTE

On the R.R.S. *Charles Darwin* computer system certain of the mentioned files are held in the following directories and may not be present on other systems:

FILE	DIRECTORY	USE
ulinker	/usr/lib	Linker
ftnlib.obj	/usr/lib	Fortran library
paslib.obj	/usr/lib	Fortran library
spisubs.obj	/nerc/levc/ref/src/spif/fortran	Graphic library
SPI routines	/nerc/levc/ref/src/spif	Graphic sources

3.5 Graphics

The graphics available on the UNIX system is the SPI language. The subroutines available are listed below. When using graphics the linking process must include the graphics libraries (e.g. spisubs.obj). Programs can be run in several ways depending on the output path. Generally there are three paths: the screen, the plotter, or a file. Each of these requires some kind of filter or path which must be specified.

Examples:

test | m2250	plots program 'test' results on the Micrographics m2250 terminal screen	<3.5.1>
test | ca1039	plots program 'test' results on the Calcomp 1039 plotter	<3.5.2>
test >plot.plo	saves all program output in a file 'plot.plo' - plotting can then be done using cat plot.plo | m2250	<3.5.3>

Since all the output is piped on a certain path, there must be no mixing of graphics output with alphanumeric output. Alphanumeric output should be written to file within the program thus not confusing the two outputs. This also holds for inputs.

The graphics routines currently available are:-

anumb(degs,nflds,ndp,itype,size,angle,icont,ierr)	Write an angle or Lat./Long.
cench(n,size)	Plot centred mark number n
devend	Terminate graphics activity and close all devices
enumb(r,nc,ndp,size,angle,icont,ierr)	Write a number in fixed E- format
fnumb(r,nc,ndp,size,angle,icont,ierr)	Write a number in fixed F- format
inumb(i,nc,size,angle,icont,ierr)	Write an integer
linby2(xr,yr)	Draw a line by xr, yr from the current SPI pen position
linto2(xp,yp)	Draw a line to xp, yp
movby{2xr,yr)	Move the pen by xr, yr from the current SPI pen position
movto2(xp,yp)	Move the pen to xp, yp
nc836	Select the nicolet 836 driver (needed for all SPI programs)

pen(npen)	Select a new SPI current pen SPI Pen 1 = Black SPI Pen 2 = Red SPI Pen 3 = Green SPI Pen 4 = Blue
piccle	Closes the old page and opens a new one, without resetting parameters
shift2(xp,yp)	Shifts the origin by xp, yp
text (string,nchars,size,angle,icont)	Write nchar characters of text in A1 format from array string

Availability of hardware on the R.R.S. *Charles Darwin* can be variable. In general there are two graphics outputs: a Calcomp 1039 plotter and some Micrographics m2250 graphics terminals. Unfortunately they appear to have different scales; the Calcomp having 1 unit per millimetre and the m2250 having an area of 2100 units in the x-direction and 1600 units in the y-direction. In the course of various trials it was observed that the m2250 would invert the y-axis if plotting began at figures less than 36 hence a shift2 command is required at the outset.

Example graphics program to draw a small picture (Figure 4)

```
C
C  * * * Test.for * * *
C
C  Requires spisubs.obj when linking
C
      call nc836                    ;Nicolet driver
      call piccle                   ;Clear screen
      call shift2(0.0,36.0)         ; Fudge ! (see above)
      call movto2(30.0,80.0)        ; Move into the central area
      call pen(2)                   ; Change pen to red
      call linby2(0.0,-50.0)
      call linby2(30.0,0.0)
      call linby2(0.0,50.0)
```

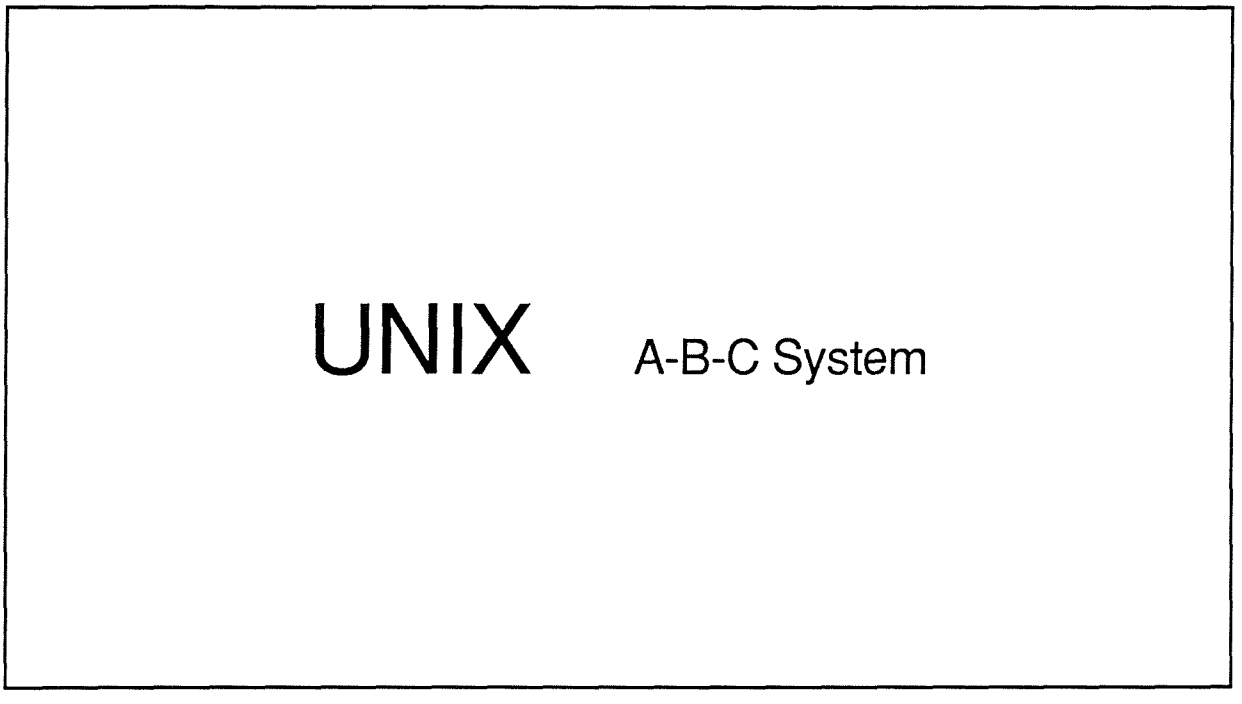


Figure 4. Graphical output from program example

```
call movto2(80.0,30.0)
call pen(3)
call linto2(80.0,80.0)
call linto2(110.0,30.0)
call linto2(110.0,80.0)
call movby2(20.0,-50.0)
call pen(4)
call linby2(0.0,50.0)
call movby2(20.0,0.0)
call pen(5)
call linto2(180.0,30.0)
call movby2(-30.0,0.0)
call linto2(180.0,80.0)
call movby2(100.0,-40.0)
call pen(1)
call text('A-B-C system',12,2.5,0.0,3)
call devend
end
```

3.6 Input/Output

Input and Output routines in Fortran can differ from machine to machine. Standard input and output channels are referred to as unit 0 or * and their default is formatted and sequential. The OPEN statement is the main command discussed here. The other commands are similar to any other system e.g. CLOSE, REWIND, ENDFILE and their syntax is available in the Fortran language manual.

OPEN(options)

options: UNIT=u	u is the unit specifier
ERR=s	s is the exit statement label
FILE=fname	fname is the filename
STATUS=sta	sta is the status - must be 'OLD', 'NEW', 'SCRATCH' or 'UNKNOWN'
ACCESS=acc	acc is the access code - must be 'SEQUENTIAL' or 'DIRECT'

FORM=fm	fm is the format code - must be 'FORMATTED' or 'UNFORMATTED'

Default is 'SCRATCH', 'SEQUENTIAL' & 'FORMATTED'.

4. THE EDITOR

4.1 Editing

There are two commonly available editors *ed* and *vi*. Of these *ed* is the more basic editor initially designed for use on dumb terminals such teletypewriters. *vi* is a screen editor and takes full advantage of the video screen. In this section only *vi* is described as it is powerful and more useful than *ed* which is adequately described in Unisoft Systems User Guide.

4.2 The *vi* commands

Entry and Exit

vi filename	Entry into file 'filename'
:x	Exit *vi* saving changes
:q	Exit *vi* with no save
:q!	Exit *vi* not saving any changes
:w file	write (copy) to 'file'

Cursor Movement

Return	Down 1 line
-	Up 1 line
Space	Right 1 character
Backspace	Left 1 character
$	Move to end of line
0	Move to beginning of line
cntl-D	Move down half a page
cntl-U	Move up half a page
cntl-F	Move forwards a whole page
cntl-B	Move backwards a whole page
/cde/	Search forwards for the pattern 'cde'
?cde?	Search backwards for the pattern 'cde'
nG	Go to line n

Insert (until ESC)

i		Insert text before current character
a		Insert text after current character
O		Insert text before current line
o		Insert text after current line

Change/Replace

r		Replace current character
cc		Change current line (until ESC)
:s/X/Y/opt		Substitute Y for the first occurence of X - options are:-
	g	change every occurrence in line
	c	confirm each change
	p	print each change

Delete

x	Delete current character
nx	Delete n characters starting on the current character
dd	Delete current line
ndd	Delete n lines starting on the current line

Undo

u	Undo last operation
U	Restore current line

Global Commands

:g/X/s//Y/opt		Globally find the string X on every line and substitute for the string Y
:g/X/s/Y/Z/opt		Globally find the string X on every line and substitute the string Z for the string Y on those lines - options are:-
	g	change every occurrence in line
	c	confirm each change
	p	print each change

5. SHIPBOARD FACILITIES

5.1 Shipboard Programs

The following list gives an idea of what programs are available for data editing and processing on the UNIX system on all NERC ships. Further information is available in the Level-C Operators Manual (Shipborne Computer Group). It is suggested that these programs are only accessed with the permission of the operator or under their supervision.

ANNOT	Produces annotation specification
ANTPLOT	Produces an annotated plot
COMPASS_C	Plots compass rose (colour)
COMPASS_M	Plots compass rose (monochrome)
DEPCODE	Produces depth codes for Gloria replays
DIGREAD	To generate files of converted digitiser data
DXFMT	Produces merge-merge format 3 magtape for data transfer
EDSATS	Edits satellite fix file satfix
EDSTATUS	Edits status of data on most data files
GF3	Produces gf3 format file on disc or tape for data transfer
GPSNAV	Combines relative navigation with GPS fixes
GRID	Produces the grid specification
GRIDPLOT	Produces an annotated plot
LISTIT	General purpose data listing program
LIVENAV	Produces live navigation for plot
LIVEPREP	Sets up grid limits for live plotting
LIVEPLOT	Plots the live data
LORIN	Manual entry of Loran fixes
LORNAV	Combines relative navigation with Loran fixes
MANDEP	Manual entry of depth data
PRINTDEP	Prints depths - corrected and uncorrected
PRINTIT	Prints navigation data and/or other parameters
PRINTGPS	Prints GPS fixes
PRINTLOR	Prints Loran fixes
PRINTSAT	Prints satellite fixes
PRODEP	Corrects depths
PROGRAV	Calculates free air anomaly (FAA)
PROMAG	Calculates magnetic anomaly (RESIDMAG)

PRONAV	Combines relative navigation with fixes
SATFUDGE	Forces satellite to give position to corresponding finalnav
SATNAV	Combines relative navigation with satellite fixes SPOT draws markers on a grid
TITSIL	Writes data into a data file (reverse of LISTIT)
TRACK	Produces the track specification
TRACKPLOT	Produces a track plot
XYPREP	Creates input file for XYPLOT
XYPLOT :	
TPLOT	Plot data against time (live)
TPLOT_H	Plot data against time (hardcopy)
DPLOT	Plot data against distance (live)
DPLOT_H	Plot data against distance (hardcopy)
OPLOT	Plot data against any other variable (live)
OPLOT_H	Plot data against any other variable (hardcopy)

Example:

Four programs are required to produce a navigation plot on a grid. The first two grid and track are form programs allowing data to be input using forms and thus are relatively easy to operate. To enter the parameters in each form window on the screen the following keys are used:-

TAB	Move on one window
ESC	Move back one window
LINEFEED	Move on one space
BACKSPACE	Move back one space

All the other keys act as normal.

When both data forms are filled, plotting can start. The grid is plotted first, followed by the track plotted on top of this grid.

The series of commands thus could look like this:-

```
grid g.tarea
track t.tarea
gridplot g.tarea | ca1039
trackplot -g g.tarea - t.tarea | ca1039
```

The first line creates a form file g.tarea holding the user-defined grid parameters and the second line similarly creates a form file t.tarea holding the data to be plotted on the grid. The last two lines plot the grid and track respectively on the Calcomp 1039.

On the R.R.S. *Charles Darwin* all this must be done in directory /nerc/data/maps, and the user must have the appropriate access permissions. These are available from the main UNIX operator.

5.2 The Digitiser

Hardware - Complot Series 7000 Digitiser (+ a terminal)

There is one interface program for the digitiser - DIGREAD. It is a menu driven program and self-explanatory. To run this program the user must have the appropriate permissions for directory /nerc/levc/ref/src/digitiser and be in that directory. Within the program the user can use the digitiser predefined boxes (at the top of the tablet) to select the opening of the correct and input the scaling in x-y coordinates. The digitising tablet should be set to point or stream mode via the keys on the side of the tablet. Point mode is probably easier, especially in heavy seas! Use the digitising tablet mouse keys 0-9 to enter the X and Y values. From there the terminal will give instructions to the program's use. When finishing, use the exit and confirm predefined tablet boxes to stop the input.

The terminal attached to the digitiser has no graphics capabilities itself thus verification of the digitisation is difficult. Therefore when digitising it is advisable to take note of which horizons or contours have been digitised.

Program DIGREAD now writes directly to data files. Therefore they must be created before execution by using program CREDAT. Seismic data is written with variables y and button number, contour data is written with variables x and y, and Mercator data is written with variables lat and lon (though at present Mercator values are treated as a linear scale).

5.3 Data files and their variables

All these files are kept in binary files within directory /nerc/data/cruise*n*. Some of these files are recirculatory which means that if the data file fills, it recycles itself overwriting the original version. This saves space and gets rid of unwanted raw and unprocessed data (e.g. gyro) after it has been processed. The recycling period is commonly about a week though can vary from a month to a single day. This data is available if required in other programs via the LISTIT command program.

BESTNAV	lat lon vn ve cmg smg dist_run heading source
EM_LOG	speedfa speedps
EVNT_LOG	evnt_no
GPSFIX	lat lon svc
GPSNAV	lat lon vn ve cmg smg dist_run heading

GRAVITY	grav_a
GYRO	heading
LIVENAV	lat lon vn ve cmg smg dist_run heading
LORFIX	lat lon td1 td2
LORNAV	lat lon vn ve cmg smg dist_run heading
MAGNET	magfld
MX1107	lat lon slt slnel it ct dist dir sat r
PROCTD	press temp cond trans oxyc raw salin psu sigmat svanom
	potemp sigmap depth soundv oxygen oxypc
PRODEP	uncdepth cordepth cartarea
PROGRAV	gravfld gravfaa gravet latcor bouguer
PROMAG	magfield maganomaly
RAWCTD	pressure temp cond trans oxyc oxyt spare1 spare2
RAWDEP	uncdepth
RELMOV	vn ve pfa pps pgyro
SATNAV	lat lon vn ve cmg smg dist_run heading
SIMRAD	lat lon
702FIX	lat lon crse speed it el slt sln rms whdg wspd sat

All these data files have a machine encoded time code which can be used if required.

BIBLIOGRAPHY

Banahan M. & Rutter A., (1982) UNIX - the Book, Sigma Technical Press.

Bourne S.R., (1983) The UNIX System, Addison-Wesley.

Kernighan B.W. & Ritchie D.M., (1978) The C Programming Language, Prentice-Hall.

McGilton H. & Morgan R., (1983) Introducing the UNIX System, McGraw-Hill.

Metcalf M. (1985) Effective Fortran 77, Clarendon Press.

Shipborne Computer Group, RVS (1983) Level C Operators's Manual, NERC.

Unisoft Systems, (1984) Uniplus+ System V :

User Guide

User's Manual Section I

Fortran - Language Manual for Uniplus+

ACKNOWLEDGEMENTS

I would like to thank the RVS Shipboard Computer Group and especially Martin Beney for their assistance while using this computer system. I would also like to thank Roger Searle for his observations and recommendations in preparing this text.