Department of Computing Science Working Paper Series

Faculty of Engineering and Information Sciences

1983

A tutorial introduction to the Unix to Univac communiation system

Cecily S. Bailes
University of Wollongong, uow@cbailes.edu.au

THE UNIVERSITY OF WOLLONGONG

DEPARTMENT OF COMPUTING SCIENCE

A TUTORIAL INTRODUCTION TO THE UNIX TO UNIVAC COMMUNICATION SYSTEM

Cecily S. Bailes

Department of Computing Science
University of Wollongong

Abstract

This tutorial describes the use of the UNIX to Univac 1100 communications software. The following facilities are provided: a "transparent link" to the Univac so that a UNIX terminal may be used as a Univac terminal; the submission of batch jobs to the Univac; and file transfers between the UNIX and Univac systems.

Preprint No 83-8 **August 15, 1983**

P.O. Box 1144, WOLLONGONG, N.S.W. AUSTRALIA
telephone (042)-282-981
telex AA29022

A Tutorial Introduction to the UNIX* to Univac Communication System

Cecily. S. Bailes

Department of Computing Science
University of Wollongong

1. INTRODUCTION

This document is a tutorial to introduce UNIX to Univac communications. The available commands are:

uvdct	for the establishment of a 'transparent link' to the Univac so that a UNIX terminal may be used as a Univac terminal.
uvb	for batch job submission.
uvcp	for inter-system file transfers.
uvftn	and **uvcob** provide all necessary job control commands for the submission of FORTRAN and COBOL batch jobs to the Univac.

These commands are documented in [1] and included in the Appendix for easy reference. The system was originally written by Charles M. Francis of the University of Wollongong in 1981. It was amended and extended by Cecily Bailes in 1982.

2. INTERACTIVE COMMUNICATION

The command

```
    uvdct
```

attempts to connect the user's UNIX terminal to the Univac, simulating a Univac DCT 1000 data communications terminal. Should the attempt fail, typically because all of the available lines to the Univac are in use, the message

```
    Univac dct1000 lines are not available
```

will be output, the appropriate response to which is patience (i.e. try again a little later). In the case of success, the following message will appear on the screen

```
    Site-id :csci0?
```

where ? stands for any single digit. The user may now interact with the Univac (if the Univac is running) beginning with the normal Univac sign-on procedure (see [2]). Input lines are copied verbatim to the Univac save for those commencing with the tilde ('~') character. On termination of a Univac session (see section 2.5), the user is returned to the UNIX system.

2.1. Execution of UNIX commands

If a line begins with the character pair tilde, exclamation mark '~!', then the remainder of the line is treated as a UNIX command, in much the same manner as the escape to the shell available in programs such as **ed**(1) and **mail**(1). For example,

```
    ~!pwd
```

will display the name of the user's current (UNIX) working directory.

*UNIX is a Trademark of Bell Laboratories.

2.2. Input/Output Re-direction

If a line begins with the characters '~<', then the next blank-free character string on that line is interpreted as the name of a UNIX file, the contents of which (subject to the user's access privileges) are transmitted to the Univac. For example,

```
~< classlist
```

transmits the contents of the file classlist in the user's current directory to the Univac as if it had been typed on the user's terminal. The transmitted characters are echoed on the user's terminal. Input re-direction ends when all characters of the nominated file are transmitted.

If a line begins with the characters '~>', then the next blank-free character sequence on the line is taken to be the name of a UNIX file to which subsequent output from the Univac is written. The previous content of the file (if it exists) is obliterated and the output from the Univac is also displayed on the user's terminal. If the line begins with '~>>' , then the Univac output is appended to the file. The previous content of the file is unchanged. To terminate output re-direction, the UNIX terminal user enters the line '~>' without a file name. For example,

```
~> output
```

creates the file output in the user's current directory and writes all subsequent output to this file as well as to the user's terminal.

```
~>> datalog
```

appends further output to the file called datalog in the user's current directory and writes output to the user's terminal as before. Whenever the user enters

```
~>
```

the prevailing re-direction is turned off.

2.3. File Transfers

The commands **~%take** and **~%put** are available for the simple transfers of files to the Univac from UNIX and vice versa. The line

```
~%take  file1  file2
```

copies a file called file1 from the Univac to the file named file2 on the UNIX system, subject to the conventions of both systems. Conversely,

```
~%put  file1  file2
```

copies file1 from the UNIX system to file2 on the Univac. While a transfer is taking place, all conversation with the Univac is suspended. Any attempt by the user to issue input to the Univac will cause the following message to be output:

```
Copy proceeding--to interrupt type 'stop'
```

As stated in the above message, termination of an on-going transfer may be brought about by entering the command line

```
stop
```

The file being created by the copy operation will then exist in truncated form. Whilst a tranfer is taking place, UNIX commands will continue to work normally, e.g.

```
~!pwd
```

will still cause the name of the current UNIX working directory to be displayed. At the termination of the transfer, the interaction with the Univac is resumed.

2.4. Miscellaneous

To transmit a text line beginning with a '~', begin the line with the character pair '~~'. For example,

```
~~xyz
```

transmits the line

```
~xyz
```

Any lines starting with a '~' but not in one of the above formats (i.e. not beginning with the strings '~!' '~>' '~>>' '~<' or '~~') will not be transmitted, but will cause the following warning message to be displayed:

```
To send a line beginning with '~' type '~~'
```

2.5. *Termination*

Dialogue is terminated by either the Univac **@@term** command, or a line containing the control-d character after which the user is returned to the UNIX system.

3. BATCH JOB SUBMISSION

The **uvb** command provides a facility to add jobs to the batch queue of the Univac system and to manipulate the queue.

3.1. Adding to the Queue

The UNIX command

```
uvb  file1 ... filen
```

takes each of file1 ... filen in turn and submits it as a runstream to the Univac, removing any lines beginning with the character pair '@@', end-of-input holding true if any one of the following conditions occurs:

(i) an **@fin** card is detected

(ii) another **@run** card is detected

(iii) end-of-file is detected

Prior to submission, the user is prompted for a Univac user-id, unless one of the following options (for example) is used.

(i) `uvb -i cmf filea fileb filec`

will submit the runstream from files filea fileb and filec with the user-id cmf.

(ii) `uvb -c filea`   `uvb -m fileb`

will submit the runstream in filea (respectively fileb) with the standard computing science (respectively mathematics) user-id consisting of the string '04' (respectively '17') prepended to the UNIX user-id.

The **uvb** command will automatically generate a Univac **@run** card with the following format

@run,/nb *user-id*,d04card81/csci,*user-id* unless the user specifically uses the −r option and provides an **@run** card as follows

```
uvb −r filea
```

The file filea should contain an **@run** card as its first input line.

3.2. Manipulating the Univac Batch Submission Queue

The options for the **uvb** command, which may be used to manipulate the Univac batch job submission queue are:

−q display the identifying numbers of jobs currently in the queue

−k delete one or more jobs belonging to a particular user, except those that are currently being processed. The command

```
uvb −k
```

deletes all the jobs in the queue belonging to the user. If the command is of the form

```
uvb −k j1 ... jn
```

only jobs with numbers *j1* ... *jn* will be deleted. The job numbers may be obtained with the **−q** option.

−f attempts to fetch output immediately (see Section 3.3).

3.3. Results

Results of batch submissions are transmitted from the Univac to UNIX periodically (at 5 minute intervals). This process may be expedited by the −f option to **uvb**. A UNIX user is notified of the availability of results by normal system mail (see **mail**(1)). The results are placed in a file (nominated by the communication system) in the user's home directory. If this is not possible, because the UNIX user-id of the initiator of the submission has been corrupted on the Univac by truncation to conform to Univac size limits, the output will remain in the file directory /usr/spool/uvout.

4. INTER-SYSTEM FILE TRANSFERS

The **uvcp** program is a special purpose program to handle file transfers between the UNIX system and the Univac. It may be viewed as a variation of the **uvb** program to effect the**%take** and**%put** commands available under **uvdct**. It differs from these commands in that the transfers take place as a submitted batch job, not as part of an interactive dialogue. The command

```
uvcp filea fileb
```

copies filea into fileb. The nature of the name of the file identifies which system (UNIX or Univac) to use as the source and/or destination. A file name of the form

```
unix!prog.c
```

refers to file prog.c on the UNIX system, and likewise

```
univac!prog.for
```

refers to file prog.for on the Univac. A file without either of the prefixes 'unix!' or 'univac!' signifies a UNIX file. Standard system file name conventions and defaults are otherwise observed.

For purposes of identification to the Univac, the −i, −c and **−m** options of **uvb** are available (see Section 3 above). Similarly, if no user-id is provided, the user is prompted for one. The user is notified of the outcome of a copy operation by mail.

4.1. Examples

The command

```
uvcp -i cmf unix!olddata univac!newdata
```

copies the file olddata on the UNIX system into the data file cmf*newdata on the Univac. The command

```
uvcp -c univac!output results
```

(assuming the UNIX user-id is cmf) will copy the Univac file 04cmf*output into the UNIX file named results. It is possible to copy one Univac file into another, as shown by

```
uvcp -i cmf univac!original univac!newcopy
```

which copies cmf*original to cmf*newcopy (both files being on the Univac) For completeness, it is possible to copy a UNIX file into another UNIX file:

```
uvcp -i cmf fromfile tofile
```

copies the UNIX file named fromfile to a UNIX file named tofile using the **cp**(1) program.

5. SPECIAL PURPOSE COMMANDS

Commands **uvftn** and **uvcob** provide a sequence of standard control commands for the submission of FORTRAN and COBOL batch jobs to the Univac. The command

```
uvftn file1.f ... filen.f data1 ... datan
```

concatenates the files file1.f ... filen.f, which should contain FORTRAN source code, and passes the resulting file to the Univac ASCII FORTRAN compiler. The files data1 ... datan are concatenated and used as data to the compiled program. Two options, **-n** and -r, are available.

```
uvftn -n prog1.f datafile
```

suppresses the FORTRAN compiler listing. Alternatively

```
uvftn -r prog1.f datafile
```

produces a cross-reference table with the compiler listing for the file prog1.f.

```
uvcob file1.cob ... filen.cob data1 ... datan
```

has a similar effect but with respect to the Univac ASCII COBOL compiler. The same options that apply to **uvftn** also apply to **uvcob**.

```
uvcob -n prog2.cob dataset
```

produces no compiler listing, and

```
uvcob -r prog2.cob dataset
```

produces a cross-reference listing.

6. ACKNOWLEDGEMENT

My thanks go to Professor Juris Reinfelds for his comments on drafts of this document.

7. REFERENCES

[1] UNIX Programmer's Manual, Seventh Edition, Volume 1, The University of Wollongong.

[2] SPERRY UNIVAC Series 1100 Executive System, Volume 2, EXEC, Programmer Reference.

8. APPENDIX

The relevant pages of Volume 1 of the UNIX programmer's Manual now follow.

NAME
uvb – Univac remote batch submission

SYNOPSIS
uvb [–option ...] [file ...] [job# ...]

DESCRIPTION
uvb provides a user interface to the batch submission and queueing for the Univac. *uvb* is used to submit a run stream to the Univac for batch execution. The file is first copied into a temporary file, and screened for lines beginning with '@@', then it is queued for submission to the Univac. *uvmpx* maintains the queue, and manages a daemon (*uvd*) which does the actual submission. When the job has been submitted, it will remain on the queue until its output has been retrieved by the daemon. When output from a job is received by *uvd*, an attempt will be made to identify the user based on the RUNID of the job. A heuristic approach is used, looking first for the run-id in the queue, then searching the password file for the name (after stripping the first 2 characters of the department code), then the *gecos* field of the password file. When the user has been identified, the output will be placed in his directory under a unique file name, and a message sent to him via *mail*(1). If the user was not identified, the output will be left in */usr/spool/uvout/**.

The user-id (which is actually used for the run-id and the project-id) must be supplied. By default, *uvb* will prompt for a user-id, but this may be altered by the following options.

–u the user-id should be read from the *gecos* field of the password file.

–i the user-id is the next argument on the command line.

–c a computer science user-id should be used, consisting of `04' followed by the UNIX user name.

–m a math user-id should be used, consisting of `17' followed by the UNIX user name.

By default, an `@run' card will be automatically generated by the submission daemon, based on the supplied user-id. If the user wishes to supply his own `@run' card, he should specify the option –r. (The actual user-id used by *uvd* is 'csci'.)

In addition to submission of jobs, *uvb* provides several options to manipulate the queue:

–q will cause the Univac batch submission queue to be displayed. (requests from *uvcp*(1) will also be displayed)

–k will cause all of a user's jobs (except those actively transmitting) to be deleted from the queue. If *job#* is given, only the specified jobs will be killed. (A list of numbers is supplied with the –q option.)

–f provides a specific request to the daemon to fetch any queued output (the daemon will automatically try to fetch output every 5 minutes when there are jobs on the queue waiting for output).

–z restart the daemon. This may only be executed by gurus, and is useful if the daemon appears to be hung, or dies prematurely. The old daemon should be killed first, to release the line to the Univac.

FILES

/usr/spool/uv/mpx	multiplexed file to spooler
/tmp/uv/*	directory for temporary spooled files

SEE ALSO
uvdct(1), uvd(8), uv(4), uvcp(1), mail(1)

AUTHOR

Charles M. Francis,
University of Wollongong.

BUGS

Unix users are identified by a heuristic approach, which doesn't always succeed.

Various unusual conditions can probably cause behaviour from the Univac that is unexpected, and hangs the daemon.

NAME

uvcp – Univac batch file copy

SYNOPSIS

uvcp [–option [user–id]] source–file destination–file

DESCRIPTION

uvcp copies the *source–file* into the *destination–file*, using the Univac remote batch queueing system. The file name may be a standard UNIX file name, or it may have the form

system–name ! file–name

where 'system–name' is either **unix** or **univac**. UNIX file names will be expanded with the current working directory name prefixed where necessary. The Univac file name may consist of filename.eltname or just filename. If no qualifier is supplied, the default will be the user–id.

Several options can apply to the user–id (which is used for both the run–id and the project–id):

–u the user–id should be read from the *gecos* field of the password file.

–i the user–id is the next argument on the command line (before the file names)

–c A computer science user–id should be used, consisting of '04' followed by the UNIX user name.

–m A math user–id should be used, consisting of '17' followed by the UNIX user name.

The default is to prompt the user for the user–id when the command is executed.

Copies from UNIX to UNIX will be done by executing *cp*(1). All other copies will be queued using the batch submission queue (see *uvb*(1)) and copied by the batch submission daemon. When the copy has completed, the user will be notified by *mail*(1).

For copies from UNIX to the Univac, the editor will be used, and UNIX default tabs supplied. To override the tab specification, the file being copied should contain the command:

@EDIT SET tab1 tab2 ...

FILES

/usr/spool/uv/mpx	multiplexed file to spooler
/tmp/uv/*	directory for temporary spooled files

AUTHOR

Charles M. Francis,
University of Wollongong.

SEE ALSO

uvb(1), uvdct(1), uvd(8), uv(4), cp(1), mail(1)

BUGS

Only printable characters will be reliably transferred between systems, as the Univac editor is used for the transfer. Similarly, control cards beginning with '@@' or '@EOF' will not be copied, as they would put the editor out of input mode. Lines beginning with EOF: on files being copied from the Univac to the local UNIX system will cause the file to be truncated.

NAME

uvdct – Univac dct1000 emulator

SYNOPSIS

uvdct [device siteid]

DESCRIPTION

uvdct connects to a UNIVAC 1100 EXEC–8 operating system as a DCT1000 terminal, and manages an interactive conversation with the Univac. If *device* and *siteid* are specified, an attempt will be made to open the specified line, otherwise it cycles through all of the available DCT lines, and connects to the first free one.

It then runs as two parallel processes, one of which reads from the Univac, and writes to the terminal, and the other reads from the terminal and writes to the Univac. (The two processes intercommunicate by means of a pipe.)

Any line beginning with a ˜ is specially interpreted by the program. The following combinations have meaning:

˜!cmd ...	run the command on the local UNIX system (via **sh –c**)
˜>[>] file	redirect output from the Univac to the specified file on the local UNIX system. If '>>' is used, the output will be appended to the specified file.
˜>	terminate redirection of output.
˜< file	read the specified file on the local UNIX system, and send to the Univac as input.
˜%take file1 file2	copy 'file1' from the Univac into 'file2' on the local UNIX system.
˜%put file1 file2	copy 'file1' from the local UNIX system to 'file2' on the Univac. For both ˜**%put** and ˜**%take** all of the necessary files should already be assigned on the Univac. The transfer in both cases will be done using the Univac editor.
˜˜...	send the line '˜...' to the Univac.

Any other lines beginning with ˜ will receive a warning message and not be sent.

While a file is being copied by **put** or **take** any commands directed to the Univac will be ignored. Commands may still be run on the local UNIX system using ˜!. To abort a file copy, type 'stop'.

FILES

/dev/uv? lines to the Univac /dev/dsa? data set synchronous link adapter

SEE ALSO

uvb(1), uvcp(1), uv(5), cu(1)

BUGS

If the Univac is down, the program tends to hang.

While executing a UNIX shell, periodic messages or prompts may appear from the Univac (this does have the advantage that it reminds the user he's logged on to the Univac).

For **put** and **take** certain combinations of lines in the files can cause trouble. When putting files on the Univac, any lines beginning with '@' will be executed, and terminate the copy. When taking files from the Univac, a line beginning with 'EOF:' will cause the file to be truncated.

The syntax is borrowed from cu(1).

NAME

uvftn, uvcob – Univac Fortran and COBOL

SYNOPSIS

uvftn [–options] prog.f [sub.f ...] [datafile ...]
uvcob [–options] prog.cob [sub.cob ...] [datafile ...]

DESCRIPTION

uvftn and *uvcob* prepare a run stream to compile and execute the specified program on the Univac. The result is then submitted through *uvb*(1) to the Univac batch submission queue. When the output is ready, it will be returned to the user's directory, and he will be notified through *mail*(1). All of the program and subroutine files will be concatenated together in the order supplied and submitted to the ASCII FORTRAN or ASCII COBOL compilers as required. The data files will be concatenated together and supplied as input to the programs. Source files are distinguished from data files by the suffix. The options supplied for the Univac are 'ISC', which produces a source listing, and runs compile and go. A limited set of these options can be overridden by specifying as follows:

–n	Suppress source listing.
–r	Produce a cross-reference listing.

If a different protocol is required, the user must set up his own run stream, and submit the job directly through *uvb*(1).

SEE ALSO

uvb(1)

BUGS

The routines are deliberately restrictive to make them simple.