Advanced Introduction to UNIX/linux

Claude Cantin (claude.cantin@nrc.ca)
http://www.nrc.ca/imsb/rcsg

Research Computing Support Group
Information Management Services Branch
National Research Council

April 16, 2006

This page intentionally left blank.

Claude Cantin
National Research Council of Canada

History of printing:

Date	Copies
March 2003	200
March 2001	200
June 1999	200
November 1997	200
July 1996	200
November 1995	150
March 1995	150
February 1994	150
October 1993	100
August 1993	75
February 1993	75
November 1992	35
September 1992	40
February 1992	50
December 1991	50
April 1991	50
September 1990	40
January 1990	40

Table 0.1: Printings.

Contents

List of Tables

List of Figures

Chapter 1

Utilities

You have now been introduced to UNIX. A tour of the file system has been made, a series of common commands discussed, and two common editors mentioned. It is now time to introduce some of the most common utilities used by the UNIX operating system.

1.1 `Compression Utilities`

It is sometimes useful to reduce the amount of space taken by specific files. It is often done to files before they are transferred from one system to any other through a network, especially if the connection is slow.

In fact, most public domain packages downloaded using `ftp` or another method have been compressed with a utility.

Table 1.1 based on [5, p. 84] displays a number of compression utilities. `compress` is used with traditional UNIX-based systems whereas `pack` is found mostly on System V UNIX based systems; they are not bound on linux-based systems. `gzip` is now common to general UNIX (including linux) based systems, and `bzip2` is common to linux.

The other compression utilities in table 1.1 are either open source, public domain or freeware. Many were first written for the Windows/DOS world. They may be found in either linux or UNIX depending if they were installed/compiled on those platforms.

1.1.1 `compress, uncompress, zcat`

compress `[-v]` *filename...* (UNIX only)

where

Compression Programs	Decompression Program	File Suffix	Typical Filename
compress	uncompress or zcat	.Z	filename.Z
pack	unpack or pcat	.z	filename.z
gzip	gunzip or gunzip -c	.z or .gz	filename.gz
bzip2	bunzip2	.bz2	filename.bz2
stuffit	unsit	.Sit	filename.Sit
PackIt	unpit	.pit	filename.pit
zip (also archives)	unzip	.ZIP	filename.ZIP
zoo210	zoo210	.zoo	filename.zoo

Table 1.1: Common Compression Programs.

-v (verbose) displays the percentage of compression for each file.

uses "adaptive Lempel-Ziv coding" [10, p. 83] to compress files by more than 60%!

In creating the new files, the contents of *filename* is modified, and the compressed name is appended with `.Z`.

uncompress `[-vc]` *filename*`[.Z]`

where

-v (verbose) displays a message when the de-compression is done.

-c (cat) leaves the original file intact, but sends the results to standard output.

is used to restore the original name and content of *filename*.

The `.Z` is optional.

zcat *filename*

is equivalent to

uncompress -v *filename***.Z**

1.1.2 pack, unpack

Silicon Graphics systems also use `pack` to reduce the size of a file. It uses Huffman codes on a byte-to-byte basis to reduce the size of the file.

pack *filename* (System V UNIX only)

will modify *filename* and append `.z` to it. It will also display the compression ration.

`pack` has an efficiency similar to `compress`.

unpack *filename*`[.z]`

will restore the original *filename*, whereas

pcat *filename*`.z`

will send the resulting unpacked data to the standard output.

1.1.3 gzip, gunzip

Because of proprietary code in the `compress` utility, the Free Software Foundation, as part of their GNU project, wrote the `gzip`/`gunzip` utilities, and made them available to end users at no charge.

The result is that a large portion of public domain packages are now compressed using that utility, used not only by linux, but by the UNIX variants at large. `gzip` uses the Lempel-Ziv coding (LZ77) to reduce the size of the file.

Usage is very similar to `compress` and `pack`:

gzip `[-vc]` *filename*

where

`-v` displays the compression ratio.

`-c` sends the compressed output to standard output and leaves the original file intact.

`gunzip` can uncompress files originally compressed with `compress`, `gzip` or `pack`.

It can also take the "`-c`" option to send output to standard output.

In general `gzip` does a slightly better job at compressing files (when compared to `compress` and `pack`.

1.1.4 bzip2, bunzip2

bzip2 is similar with gzip in that is compresses files. It does, however, use a different algorithm (the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding), considered by many to be a better approach than the more conventional LZ77 algorithm used by gzip (this information taken from bzip2 man page).

Although mostly found on linux distributions, bzip2 is open source and may be compiled on UNIX variant systems.

Usage is very similar, but not all flags are the same, to gzip

bzip2 [**-vc**] *filename*

where

-v verbose.

-v may be replaced by --verbose.

-c sends the compressed output to standard output and leaves the original file intact.

-c may be replaced by --stdout.

If bunzip2 is used on *filename*.tbz2 or *filename*.tbz, the outfile filename will be *filename*.tar.

1.2 uuencode, uudecode

uuencode is used to convert binary-type files into an ASCII sequence, sendable by E-mail.

uuencode [*source*] *label* [> *encoded_file*]

where *source* is the input file (standard input is default), and *label* is the name of the file into which uudecode will put the resulting decoded file.

By default, the output of uuencode is displayed on the screen. To capture *encoded_file*, output redirection is necessary.

source is usually a binary or compressed file.

To re-create the original file:

uudecode *encoded_file*

where *encoded_file* is a file produced using uuencode.

The name of the resulting decoded file will be *label* (see uuencode above).

1.3 mt: Magnetic Tape

mt [**-t /dev/***tape*] *command* [*count*]

is used to control a magnetic tape drive (cassette). *tape* is the name of the device controlling the tape drive. The name of the default tape device is often `/dev/tape`.

The `-t` means that what follows is the name of the tape device.

Some of the most useful *command*s are:

1. `rewind`: rewinds the tape to the beginning.

2. `retension`: retensions the tape: forward to the end, then rewind back to the beginning. Using this command is recommended when a tape has not been written to/read from for a long time.

3. `erase`: erases the entire tape.

4. `fsf`: forward space one file if no *count* is given; forward space *count* files if this option is used.

5. `bsf`: backward space one file if no *count* given; backward space *count* files if this option is used.

6. `feom`: forward end of media; at the end of the last file on the tape.

7. `status`: returns the state of the tape drive.

The following table shows how to use the default tape drive (1/4 inch cartridge, also known as QIC) and an 8mm Exabyte tape drive on a Silicon Graphics Workstation QIC drives are not used anymore.

"`no rewind`" means that after executing the command, the tape drive will NOT rewind to the beginning. This allows the user to read/write two files in a row, or to forward space the tape one or more files.

Prior to IRIX 5.x on Silicon Graphics, writing files on cartridge tapes would change the order of the bytes (called byte-swapping). For a Sun to read those tapes, they need to have been written with the no-swap option set (i.e. for an SGI to read Sun tapes, the driver to use is "`/dev/tapens`" (now the default device in IRIX 5.x)).

Tape Action	Device Definition	
	default (QIC)	8mm Exabyte
tape access	/dev/tape	/dev/8mm
tape access, no rewind	/dev/nrtape	/dev/nr8mm
tape access, no byte-swap	/dev/tapens	/dev/8mmns
tape access, no rewind, no byte-swap	/dev/nrtapens	/dev/nr8mmns

Table 1.2: Typical cartridge tape devices

1.4 `tar`: Tape Archive

`tar` is a utility that packages a list of files (and/or directories) into a specific `tarfile`, whose name typically ends in `.tar`.

`tar` is invoked by

tar -c|t|x [**vf** *archive_file*] *file(s)...*
or
tar -c|t|x [**vzf** *archive_file*] *file(s)...* (linux only)

where *archive_file* is the name of the tape device or tar file, and *file(s)...* may be files and/or directories combined to produce the tar file. One of `c`, `t`, or `x` must be used.

-c (create): creates a new `tarfile`.

-t (table): lists the files on the `tarfile`.

-x (extract): extracts the files from the `tarfile`.

The three other options are:

-v (verbose): the size and name of each file put onto or extracted from the `tarfile` is displayed.

-z compress the archive with `gzip` as the files are read.

linux only. Archive file would end in `.tgz` instead of `.tar`.

If the `-x` or `-t` flag was used to extract or list the files part of the archive, the `-z` flag would `gunzip` the archive.

-f (archive_file): uses the next argument as the name of the `tarfile`. It is normally in the form *filename.tar* or *filename*`.tgz` (but could also be a device name, such as `/dev/8mm`).

For example, to write the files `utilities.tex` and `course.tex` onto file `save.tar`, use

tar -cvf save.tar utilities.tex course.tex

To read the files back onto another system (or elsewhere in the same system):

tar -xvf save.tar

To list the contents of the file:

tar -tvf save.tar

NOTES on tape devices (first 2 are for tape devices; the last 2 are for files):

- Not all UNIX machines create the exact same format when writing to/reading from tape. Prior to IRIX 5.x, Silicon Graphics used what is called `byte ordering`, or byte swapping whereas most other vendors did not. As a result, to read the two previous files (assuming they were written using a SUN) on a Silicon Graphics workstation, the following could be used (see next section for `dd`):

  dd if=/dev/tape conv=swab,noerror,sync | tar xf -

  or

  tar -xvf /dev/tapens (using the no-swap driver).

  To read a SGI standard tape on a Sun, `dd` would have to be used because Sun does not have a "no-swap" driver.

- Different systems may create tapes at different densities, depending on how old their tape drives are. If your system was purchased new in 1990 or 1991, chances are it is using a 150 MB tape drive. Otherwise, it is possible it is using a 60 MB tape drive.

  150 MB tape drives can read tapes written by a lower density drive, but cannot write the lower density, or write on lower density tapes. 60 MB tapes can only read and write the lower density tapes.

- A `tarfile` may exist as a file in your directory. `tar` by itself only creates an archive from a file or files (to to this, put a file name on the command line instead of *device*). The archive may be on disk, or in a directory. Such a file typically ends with `.tar`.
- If the file name is replaced by a hyphen on the creation of the `tar`file, data is sent to the standard output. Similarly, a hyphen on the extraction flag means the data will be read from the standard input. The `dd` example above illustrates that concept.

  You could laso use tar to copy a local directory to a remote system, using:

  `tar -cvf -` *local_dir* `| ssh` *remote_system* `tar -xvf -`
  OR `tar -cvf -` *local_dir* `| (ssh` *remote_system* `cd` *some_directory* `\;`
  `tar -xvf -`

1.5 `dd`: Image Copy

This utility is "useful for making an *image copy* of any media. An image copy is an exact byte-for-byte copy" [8, p. 80].

dd [**if**=*input-file*] [**of**=*output-file*] [*option*]

where both *input-file* and *output-file* are optional. If omitted, *input-file* will default to the standard input, whereas *output-file* will default to the standard output.

option can be one or more of:

- `bs` (block size): default is 512 bytes.
- `conv=`*arg...* where *arg* is one or more of the following, separated by commas:
  - `ascii`: convert EBCDIC to ASCII.
  - `ebcdic`: convert ASCII to EBCDIC.
  - `swab`: swap every pair of bytes.
  - `noerror`: do not stop processing on detecting an error.
  - `sync`: pad every input record to (default) 512 bytes.
- see your system manual for more details.

To read an older Silicon Graphics tape on a Sun, the `dd` command is:

dd if=/dev/tape conv=swab,noerror,sync | tar xf -

The input of `dd` is the file on the cartridge tape. `dd` converts that file by swapping every pair of bytes, and padding every record to 512 bytes (while not stopping if an error occurs). The resulting output file is fed (through the `pipe`) to tar, which then reads the file.

When a hyphen (`-`) is used instead of a file name on the `tar` command, the input is taken from the standard input on the *extract* and put to the standard ouput on the *create* function.

1.6 grep: Global Regular Expression Printer

`grep` finds a string in a specified list of file(s).

grep [**-in**] *expression* [*filename...*]

where *expression* is a literal to be searched for which may contain metacharacters (in which case, the literal must be in quotes), and *filename...* may be one or more files.

Some useful options are:

`-i` : ignore case, i.e. treat upper and lower case the same.

`-n` (numbers): print the line number of the expression in the file.

For example, to find, in every file ending with `.dat`, every occurrence of `Silicon Graphics`, use

grep -i "Silicon Graphics" *.dat

Or, to find the string `Claude Cantin` in files `personnel` and `payroll.list`, use

grep "Claude Cantin" personnel payroll.list

Finally, to find all occurrences of `prog2` to `prog9` in all the files of the directory, use

grep "prog[2-9]" *

1.7 `sed`: Stream Editor

`sed` is like a non-interactive `ed`: it accepts `ed` commands, but never changes the contents of the input file:

sed -f (*script_file*) *filename*
or
sed -e (*command*) *filename*

where *filename* is the file on which *command* will act. Output is directed to the standard output.

`-f` : what follows is the name of the file from which a series of *command*s will be taken.

The format of *command* is

[*line*[**,***line*]]*operation*[*parameter*]

For example, to change the string `cancer society` to `United Way` in file `letter.cancer`, and store the output in file `letter.united`, one would use:

sed -e "s/cancer society/United Way/" letter.cancer > letter.united

The next example is taken from [7, p. 221]. Pretend that file `people` contains a list of persons followed by an identification number. You want to:

- change Sally Smith's name to Sally White.
- delete the line containing Henry Morgan.
- add a line containing James Walker at the end of the file.

To do that, a file, `changes`, is created, using:

`prompt`**> cat > changes**
/Sally/s/Smith/White/
/Henry/d
**$a **
James Walker
^ D

The first line says: search for the first line containing `Sally`, then substitute `White` for `Smith`.

The second line searches for the line containing `Henry`, then deletes it.

Finally, the third line says: go to the end of the file ($), append what follows—when append is used, a \ has to be appended to each line, *except* the last one. In this case, only one line is appended, so the \ is put at the end of the command line.

To make these changes, and save the changed file in `new.people`,

sed -f changes people > new.people

As a last example, to save lines 10 to 30 in a file called `20.people`, `changes` would contain:

```
1,9d 21,$d
```

and to run:

sed -f changes people > 20.people

More information on `sed` can be found in [7, pp. 219-233].

1.8 sort: Sorting a File

sort [**-r**] *file*

This is a much simplified `sort` command syntax.

`sort` will sort the input file in the ASCII collating sequence. The `-r` flag will sort *file* in reverse ASCII collating sequence.

The default output is the standard output (the original file is left unchanged).

1.9 xmgr/xmgrace: interactive 2D plotting package

`xmgrace` is the name of the newest version of `xmgr`. Although `xmgrace` is technically not `xmgr`, it is largely based on the last version of `xmgr`. `xmgrace` would likely be found on linux, whereas `xmgr` would be found on older versions of UNIX variants.

`xmgr` is an extensive simple-to-use 2D graphics package based on the X Window system. To invoke the package simply issue:

```
xmgr
```

Interactive commands are available using pull-down menus. `xmgr` performs regressions, spline fits, running averages, correlations, fast fourier transforms, and many other operations.

Legends, symbols, a multitude of fonts, arrow sizes, titles, subtitles can all be interactively selected and positioned. The user may plot a maximum of ten graphs with thirty data sets per graph. Also available is a batch mode for unattended plotting.

A typical scientific example of `xmgr` is shown in Figure 1.1.

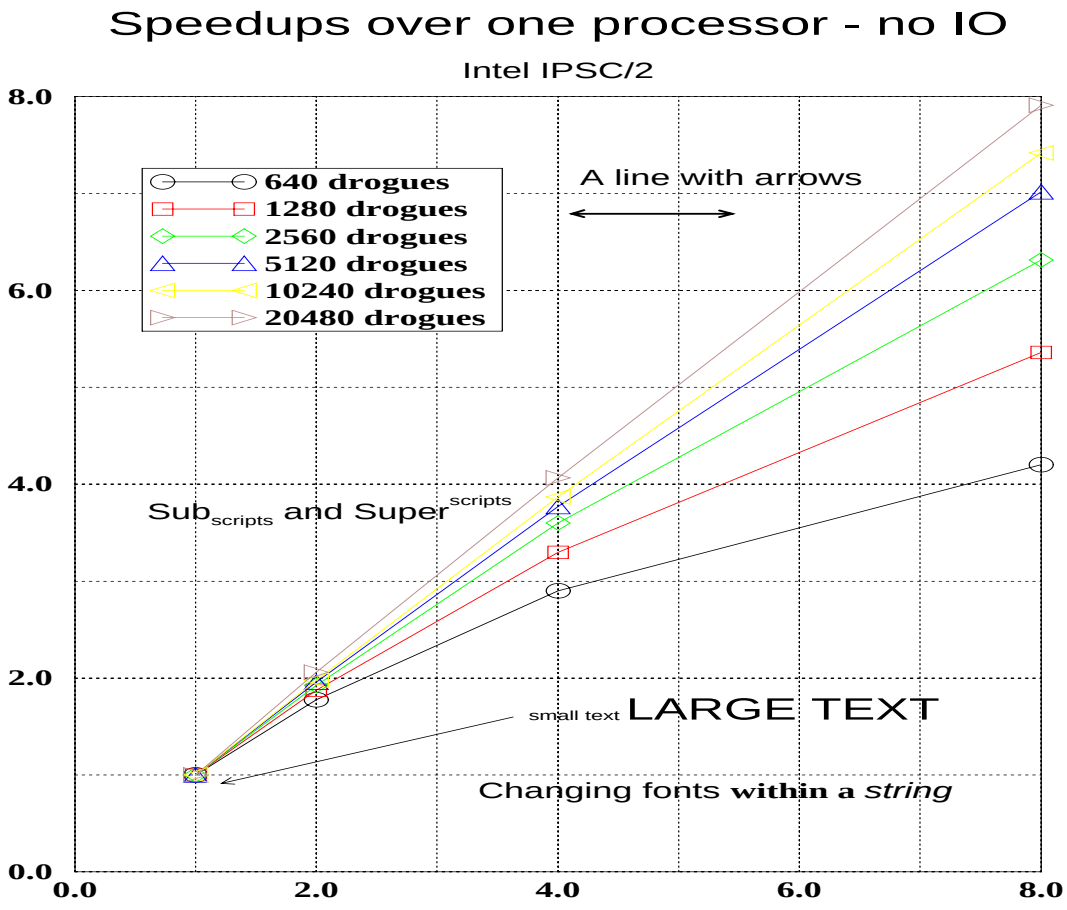


Figure 1.1: xmgr output

An 80-page PostScript manual is available as part of the package.

1.10 gnuplot: 2-D plotting program.

gnuplot is a command driven program used for plotting mathematical functions and two-dimensional data on a variety of terminal types.

If files are given, gnuplot loads each file with the load command, in the order specified. gnuplot exits after the last file is processed.

Some features include the ability to plot any number of functions, compare actual data to theoretical curves, user-defined X and Y ranges, as well as user-defined functions and constants. A number of devices can be specified for printing the plots.

gnuplot includes both a man page and a help command. A gnuplot manual is available; consult your system administrator.

To invoke the package simply type gnuplot.

1.11 xv: image viewer, editor.

xv is a program that displays/modifies/converts image files to and from numerous formats. It it GUI-based.

To invoke the package, simply type:

xv *filename*

Formats supported include .rgb, .gif and .jpeg. xv displays one image at a time in an output window. This window can be stretched, compressed, rotated, or flipped, and then saved as a new image! xv has numerous other capabilities.

1.12 a2ps: ASCII → PostScript translator.

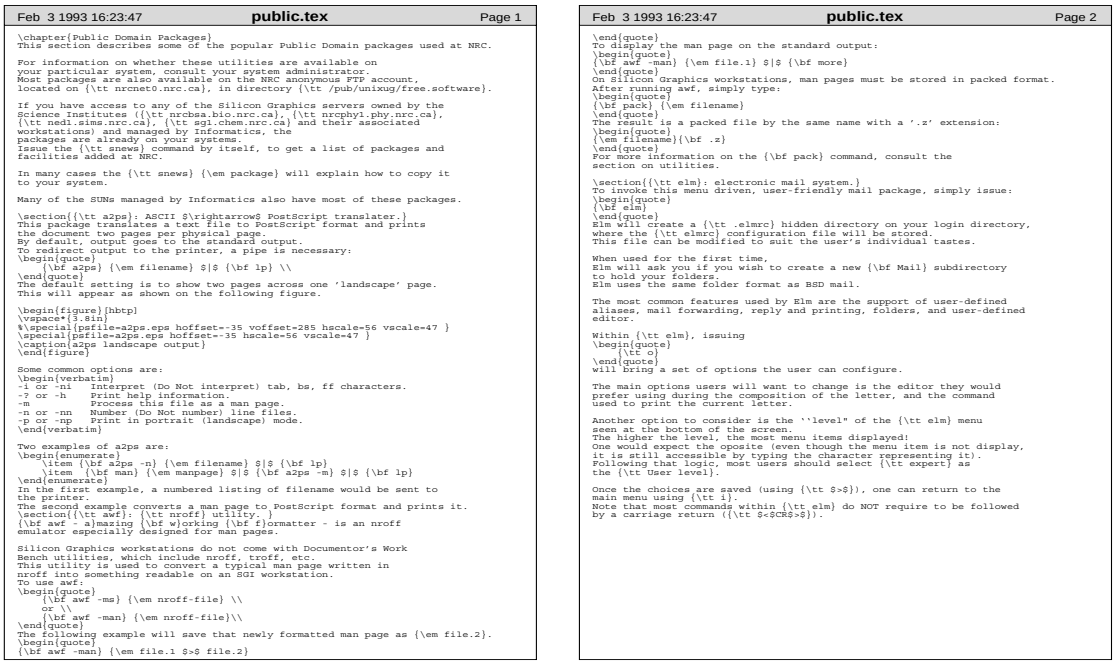
This package translates an ASCII text file to PostScript and prints the document two pages per physical page. By default, output goes to the printer, but many NRC installations have been modified so that it goes to standard output. To redirect output to the printer, a pipe is necessary:

a2ps *options filename* | lpr

Where some common options are:

```
-i or -ni   Interpret (Do Not interpret) tab, bs, ff characters.
-? or -h    Print help information.
-m          Process this file as a man page.
-n or -nn   Number (Do Not number) the lines in the files.
-p or -np   Print in portrait (landscape) mode.
```

The default setting is to display two pages across one sheet in landscape format, as shown in Figure 1.2.



Feb 3 1993 16:23:47 **public.tex** Page 1

```
\chapter{Public Domain Packages}
This section describes some of the popular Public Domain packages used at NRC.

For information on whether these utilities are available on
your particular system, consult your system administrator.
Most packages are also available on the NRC anonymous FTP account,
located on {\tt nrcnet0.nrc.ca}, in directory {\tt /pub/unixug/free.software}.

If you have access to any of the Silicon Graphics servers owned by the
Science Institutes ({\tt nrcbsa.bio.nrc.ca}, {\tt nrcphy1.phy.nrc.ca},
{\tt ned1.sims.nrc.ca}, {\tt sg1.chem.nrc.ca}) and their associated
workstations) and managed by Informatics, the
packages are already on your systems.
Issue the {\tt snews} command by itself, to get a list of packages and
facilities added at NRC.

In many cases the {\tt snews} {\em package} will explain how to copy it
to your system.

Many of the SUNs managed by Informatics also have most of these packages.

\section{{\tt a2ps}: ASCII $\rightarrow$ PostScript translater.}
This package translates a text file to PostScript format and prints
the document two pages per physical page.
By default, output goes to the standard output.
To redirect output to the printer, a pipe is necessary:
\begin{quote}
    {\bf a2ps} {\em filename} $|$ {\bf lp} \\
\end{quote}
The default setting is to show two pages across one 'landscape' page.
This will appear as shown on the following figure.

\begin{figure}[hbtp]
\vspace*{3.8in}
%\special{psfile=a2ps.eps hoffset=-35 voffset=285 hscale=56 vscale=47 }
\special{psfile=a2ps.eps hoffset=-35 hscale=56 vscale=47 }
\caption{a2ps landscape output}
\end{figure}

Some common options are:
\begin{verbatim}
-i or -ni   Interpret (Do Not interpret) tab, bs, ff characters.
-? or -h    Print help information.
-m          Process this file as a man page.
-n or -nn   Number (Do Not number) line files.
-p or -np   Print in portrait (landscape) mode.
\end{verbatim}

Two examples of a2ps are:
\begin{enumerate}
    \item {\bf a2ps -n} {\em filename} $|$ {\bf lp}
    \item  {\bf man} {\em manpage} $|$ {\bf a2ps -m} $|$ {\bf lp}
\end{enumerate}
In the first example, a numbered listing of filename would be sent to
the printer.
The second example converts a man page to PostScript format and prints it.
\section{{\tt awf}: {\tt nroff} utility. }
{\bf awf - a}mazing {\bf w}orking {\bf f}ormatter - is an nroff
emulator especially designed for man pages.

Silicon Graphics workstations do not come with Documentor's Work
Bench utilities, which include nroff, troff, etc.
This utility is used to convert a typical man page written in
nroff into something readable on an SGI workstation.
To use awf:
\begin{quote}
    {\bf awf -ms} {\em nroff-file} \\
    or \\
    {\bf awf -man} {\em nroff-file}\\
\end{quote}
The following example will save that newly formatted man page as {\em file.2}.
\begin{quote}
{\bf awf -man} {\em file.1 $>$ file.2}
```

Feb 3 1993 16:23:47 **public.tex** Page 2

```
\end{quote}
To display the man page on the standard output:
\begin{quote}
{\bf awf -man} {\em file.1} $|$ {\bf more}
\end{quote}
On Silicon Graphics workstations, man pages must be stored in packed format.
After running awf, simply type:
\begin{quote}
{\bf pack} {\em filename}
\end{quote}
The result is a packed file by the same name with a '.z' extension:
\begin{quote}
{\em filename}{\bf .z}
\end{quote}
For more information on the {\bf pack} command, consult the
section on utilities.

\section{{\tt elm}: electronic mail system.}
To invoke this menu driven, user-friendly mail package, simply issue:
\begin{quote}
{\bf elm}
\end{quote}
Elm will create a {\tt .elmrc} hidden directory on your login directory,
where the {\tt elmrc} configuration file will be stored.
This file can be modified to suit the user's individual tastes.

When used for the first time,
Elm will ask you if you wish to create a new {\bf Mail} subdirectory
to hold your folders.
Elm uses the same folder format as BSD mail.

The most common features used by Elm are the support of user-defined
aliases, mail forwarding, reply and printing, folders, and user-defined
editor.

Within {\tt elm}, issuing
\begin{quote}
    {\tt o}
\end{quote}
will bring a set of options the user can configure.

The main options users will want to change is the editor they would
prefer using during the composition of the letter, and the command
used to print the current letter.

Another option to consider is the ``level" of the {\tt elm} menu
seen at the bottom of the screen.
The higher the level, the most menu items displayed!
One would expect the oposite (even though the menu item is not display,
it is still accessible by typing the character representing it).
Following that logic, most users should select {\tt expert} as
the {\tt User level}.

Once the choices are saved (using {\tt $>$$}), one can return to the
main menu using {\tt i}.
Note that most commands within {\tt elm} do NOT require to be followed
by a carriage return ({\tt $<$CR$>$$}).
```

Figure 1.2: a2ps landscape output

Two examples of a2ps are:

1. `a2ps -n` *filename* | `lp`
2. `man` *manpage* | `a2ps -m` | `lp`

In the first example, a listing of *filename* with line numbers added would be sent to the printer. The second example converts a man page to PostScript format and prints it.

1.13 ImageMagick: convert/edit images

Imagemagick is a suite of commandline utilities to manipulate images. It may be used to scale, convert, import or even animate images.

The following sections will introduce some of the ImageMagick facilities. Many more may be found by examining it man page (each of its utilities also have their own man pages).

1.13.1 convert: convert between file formats

To convert between two image formats:

```
convert file.fmt1 file.fmt2
```

where *fmt1* and *fmt2* are respectively the source and destination file formats.

For example, to convert from a .gif to a .jpg file:

```
convert dollar.gif dollar.jpg
```

You may use convert to change the size of an image, or rotate the image. To use the dollar.gif example, and create a .rgb version, but upside down (rotated 180 degrees):

```
convert -rotate 180 dollar.gif dollar.jpg
```

There are well over 100 file formats understood by convert (and ImageMagick.

1.13.2 mogrify: convert multiple images

Instead of performing convert on many files, a more simplified procedule may be obtained with mogrify. For example, to all .gif file to .jpg files:

```
mogrify -format jpg *.gif
```

mogrify may be used to add a border to a file, to crop it (cut out a piece off the image) as well as many other operations.

1.13.3 `import`: capture a window off your screen

`import` allows you to capture a window off your screen, and convert it to the file format of your choice.

Issue

```
import file.gif
```

and notice you cursor changing to a plug (+) sign. Click in a specific window; at the bell ("beep"), `file.gif` will have been created: it is a `.gif` image of the window you clicked into!

Use `xv file.gif` to view the image.

1.14 LaTeX: Text processor.

LaTeX is a TeX macro package. TeX is a package for typesetting technical text, especially text which contains mathematical formulas.

This book was formatted using LaTeX, as were the overheads (transparencies) used during the course.

All formatting in LaTeX is done by the use of macros. By telling LaTeX what type of document you are preparing, the margins will be set automatically. A few document types include article, report, book, and letter. Some features of LaTeX include automatic hyphenation, line justification, centering, tabular aligning, and much more.

The definitive reference is "LaTeX: A Document Preparation System" by Leslie Lamport.

Research Computing Support Group does keep a copy of a basic user's guide, written in the early 1990's (but still pertinent now) at

```
http://www.sao.nrc.ca/imsb/rcsg/documents/#intro_to_latex
```

1.14.1 latex2html

`latex2html` allow the translation of LaTeXdocuments to HTML, the language used to create documents for the World Wide Web.

Although not all LaTeX commands are understood, it does a fairly good job at the translation process. It also requires a number of other public domain tools to be present for specific translations to be processed.

`latex2html` is what was used to translate the LaTeXsource used to write these notes, into the HTML version of the course notes on the web.

1.14.2 `xdvi`: DVI file previewer.

When `latex` (or `tex`) is run on a `.tex` file, the result is a `.dvi` file. `xdvi` allows the previewing of that dvi (DeVice Independent) files produced by TeX and LaTeX. To facilitate previewing, the file can be reduced in size, the margins adjusted, and the colours altered all by using a mouse and clicking buttons.

Pressing the mouse button on top of the document will cause that part of the window to be magnified. The left mouse button will cause a small part of the window to be magnified, while the middle and the right buttons will temporarily enlarge larger sections of the window.

1.14.3 `dvips`: convert from DVI to PS

Once the `.dvi` file has been produced, it is usually translated into PostScript, using

`dvips` *file* `.dvi`

The result is *file* `.ps`, which may be printed, or previewed with `ghostview`.

1.14.4 `ghostview`: X-based PostScript previewer.

The last step in working with LaTeXis typically to print or preview the document. Printing may be done with `lpr`, where as previewing is done using

`ghostview` *file* `.ps`

In reality, `ghostview` is an interface to the `ghostscript` package, which allows the previewing of PostScript documents using the X windowing system. The program allows the user to interactively select PostScript files, and then view the file using such features as plot orientation, type of media the picture is to be displayed on, and magnification factors.

Other features offered by `ghostview` are:

- Interactive selection of file to be viewed.
- Select specific pages, and either print them, or save them in a different file.
- Colour support.

1.15 `pcal`: calendar.

`pcal` is a simple calendar generation program. By default, it reads the "calendar" file in your home directory and displays those items listed for the specified month on the calendar it generates. The output – sent to the standard output – is in PostScript, so the quality is very good. Usage is:

`pcal [-e -f` *file*] [-p] [-M] [MM YY] [-a *language*]

where

`M`: print phases of the moon.
`p`: output in portrait mode instead of landscape.
MM: month in numerical format.
YY: year in numerical format.
e: do not use my "calendar" file,
`f` *file*: use *file* instead of "calendar".
`a` *language*: specifies what language to print the calendar in (ex: `fr` for French).

An example of `pcal` invoked with the command:

```
pcal -p -M 01 93 -a fr | lpr
```

is shown in Figure 1.3.

1.16 Exercises

1. Assume you are running out of disk space, and you have a few VERY large files. What can you do to regain some disk space?
2. You would like to create a backup of all your files. How would you do it?
3. You want to send a binary file to a Toronto colleague of yours. How could you do it?
4. You have received a file in the form `filename.tar.Z`. How can you read the file?
5. You receive a tape with three `tar`files on it. How do you read the second file? The third?

Janvier 1997

dimanche	lundi	mardi	mercredi	jeudi	vendredi	samedi
			1	2	3	4
5	6	7	8	9	10	11
12	13 hockey.	14	15	16 reunion ISPD/RPSO/CISTI 9:30	17	18
19	20 hockey.	21	22	23 update UNIX course Web page.	24 reunion SCSG	25
26	27 hockey.	28	29	30	31	

Decembre 1996

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Fevrier 1997

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Figure 1.3: pcal output

6. You own a Sun workstation. You receive a tape from a colleague who owns a Silicon Graphics. How will you read the `tar`file off the tape?

7. Say you own a Silicon Graphics, and you have to read a tape that was written on a Sun. How can you do it? (two ways).

8. You are writing a software package. All your source code resides in one directory. But you need to find every occurrence of all variables beginning with `molecu` in your C source code files. How would you do this?

9. You have a file containing a list of names in the format `last_name first_name`. You

want to sort that list in ascending order by last name AND by first name within the last names. Which commands would you use?

10. The only printer available to you is one which understands only the PostScript language. It does not have a PostScript interpreter. How can you send an ASCII text file to the printer to be printed?

11. Your system has an X-capable display. How would you view the content of file `tiger.ps` before sending it to the printer.

12. Convert file `tiger.ras` to `tiger.xbm`.

13. You would like to increase the size of `tiger.rgb`. Which utility could you use?

Chapter 2

Text Formatters

Text formatters differ from word processors in the sense that the text file needs to be written with formatting instructions embedded in it. It is first processed and then printed on the screen or printer to see what the document looks like. On the other hand, a word processor is usually What You See Is What You Get (WYSIWYG) style.

The standard UNIX text formatter is `nroff`, discussed in the next section. TEX, a public domain text formatter, will be discussed immediately after `nroff`.

2.1 `nroff`

This section will discuss `nroff` very briefly, introducing a handful of formatting instructions. For more information, see [7].

`nroff` is used to format `man` pages under most variants of UNIX. On these machines, `/usr/man` contains hundreds of examples of `nroff` formatted files.

Silicon Graphics' `man pages` are already formatted. They are saved in `pack`ed format. IRIX 5.3 includes `awf`, a public domain `nroff` (each vendor has to pay a royalty to the `nroff` author). The `nroff` command usually starts with a `.` (dot) as the first character of a new line, followed by one or two characters (the command itself), followed by parameters.

The following few sections introduce some simple `nroff` instructions.

2.1.1 `.po`: Page Offset

.po [+ | −] *spaces*

Right shift the page or left margin, from the left side of the page. If `+` or `-` is used, a relative

shift of *spaces* is made.

2.1.2 .ll: Line Length

.ll [+ | −] *spaces*
where *spaces* is in characters. + or - can be used to increase or decrease the line length.

2.1.3 .pl: Page Length

.pl [+ | −] *lines*

The default value is 66.

2.1.4 .pn: Page Number

.pn [+ | −] *num*
where *num* is to be the page number of the next page.

2.1.5 .pb: Page Break

.pb [*num*]

A new page will start after *num* lines.

2.1.6 .ne: Next Lines Together

.ne [*num*]
where *num* is the number of lines to keep together, on the same page.

2.1.7 .ad: Adjust

.ad [**l** | **r** | **c** | **b** | **n**]
adjusts margins, where

l : flush left, ragged right (default).

r : flush right, ragged left.

c : centres each output line.

b : right and left justification.

n : same as b.

(blank): use last mode used.

2.1.8 .sp: Spaces

.sp [*num*]
where *num* specifies a number of blank lines to be output. Default *num* is 1.

2.1.9 .ls: Line Space

.ls [*num*]
where *num* specifies the (*num* - 1) blank lines after each line. If no *num* is given, *num* is assumed to be what was written before the last change of text.

2.1.10 .ce: Centre

.ce [*num*]
centres the next *num* lines of text. The default is one.
num of 0 cancels centering.

2.1.11 .ul: Underline

.ul [*num*]
underlines the next *num* lines of text. The default is 1.
num of 0 cancels underlining.

2.2 TEX

TEX was developed by Donald Knuth. It is available free of charge, from a number of sites.

The National Research Council runs TEX and LATEX on Silicon Graphics and Sun workstations and servers.

LaTeX is a collection of TeX macros, much easier to use than TeX itself. The document you are reading was completely written in LaTeX.

SliTeX and BiBTeX are two utilities available on UNIX. The first one produces slides for overhead projectors, and the second one produces a bibliography (it was also used in this document).

The UNIX version of TeX contains other utilities such as

- `texspell`: a spelling checker.
- `texmatch`: verifies that all parentheses and brackets are "coupled".
- `xdvi`: an X-based previewer.
- `dvips`: a DVI to PostScript conversion program. It allows the inclusion of PostScript files within the document.

2.3 Exercises

1. What text formatter do you have on your system?

Chapter 3

Networking/Internet

This section is devoted to network communication between users. It will start by introducing the largest network in the world, the `Internet`.

It will then introduce a number of tools used to navigate the local network as well as the larger Internet.

3.1 The Internet

3.1.1 A Network of Networks

The Internet network is a hierarchical network of thousands of networks involving millions of host computers and tens of millions of users from over 100 countries. In March 1997, the one-millionth `domain name` was registered (from Internet World, October 1997 issue) – `nrc.ca` is considered as one registration.

ONet, BITS (Bell Internet Transit Services – formerly known as CA*net), CA*net 2 and CA*net 4 are three such networks.

NRCnet

NRC has a number of local networks. Most buildings are individually networked internally, then linked to other buidings within the same campus. The 3 Ottawa campuses and other locations across the country are in turn linked together.

Figure 3.1 shows the topology of NRC's Montreal Road network, and how it connects to the other two Ottawa campuses (Sussex Drive, and Uplands) as well as both the general Internet, and CA*net 4.

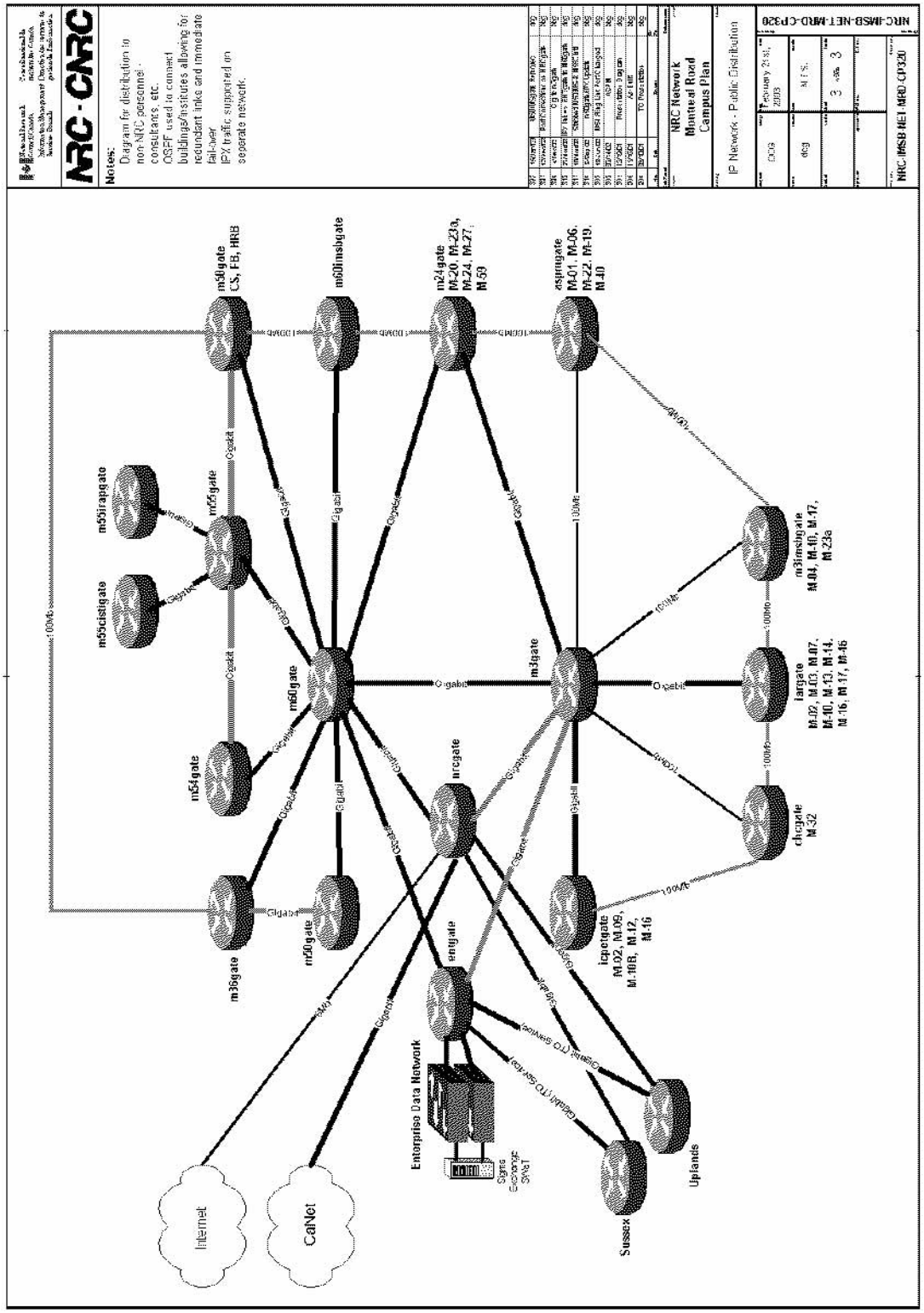


Figure 3.1: Network Connectivity of NRC/National Capital Region

Figure 3.2 shows how NRC connects all the regional Institutes together.

ONet

ONet Networking is the non-profit ISP (Internet Service Provider) mostly made out of Ontario educational and research institutions. It was the first public Ontario-based regional network (1988).

Prior to 2003, NRC's WAN was connected to ONet. For direct access to CA*net 2 and CA*net 4, ONet has a GigaPOP in Ottawa. This provides Ottawa ONet members direct access to CA*net 2/4 members.

Many Ottawa Internet sites are connected to ONet using that same router/GigaPOP.

More specific information on ONet may be found at

`http://www.onet.on.ca`

CA*net/BITS

In the early 1990s, all provinces had their own regional networks, the same way Ontario had ONet. The connections between those networks was handled by `CA*net` (regional networks are listed in Table 3.1).

On April 1, 1997, BITS (Bell Internet Transit Service, managed by Bell Advanced Communications) took over CA*net's functionality.

Soon after, a new project, CA*net 2 was officially launched (more information on CA*net 2 below).

Figure 3.3 (courtesy of the CA*net staff, 1995) shows the topology of CA*net, as it was before it became BITS.

As with ONet, more information on BITS may be found at URL

`http://www.bacits.bell.ca`

BITS is in turn connected to the American national backbone, through various connections, giving us access to international networks.

CA*net 2

The second main CANARIE (Canadian Network for the Advancement of Research, Industry and Education) initiative, CA*net 2, soon followed as the replacement Research and Development national backbone network. It was based on the ATM (Asynchronous Transfer Mode) and SONET facilities operated by Canada's Telecommunications carriers. Average

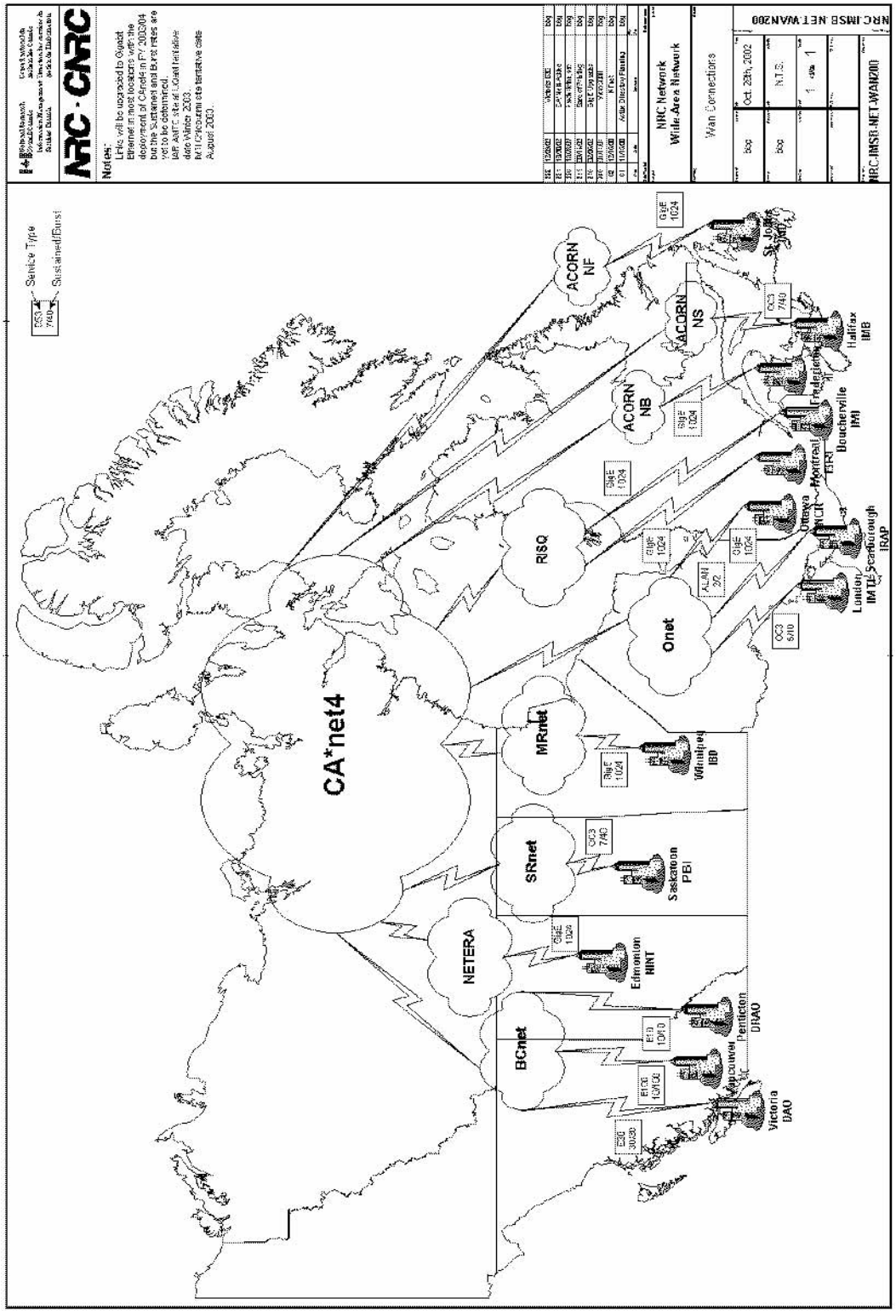


Figure 3.2: Network Connectivity of NRC/all Institutes

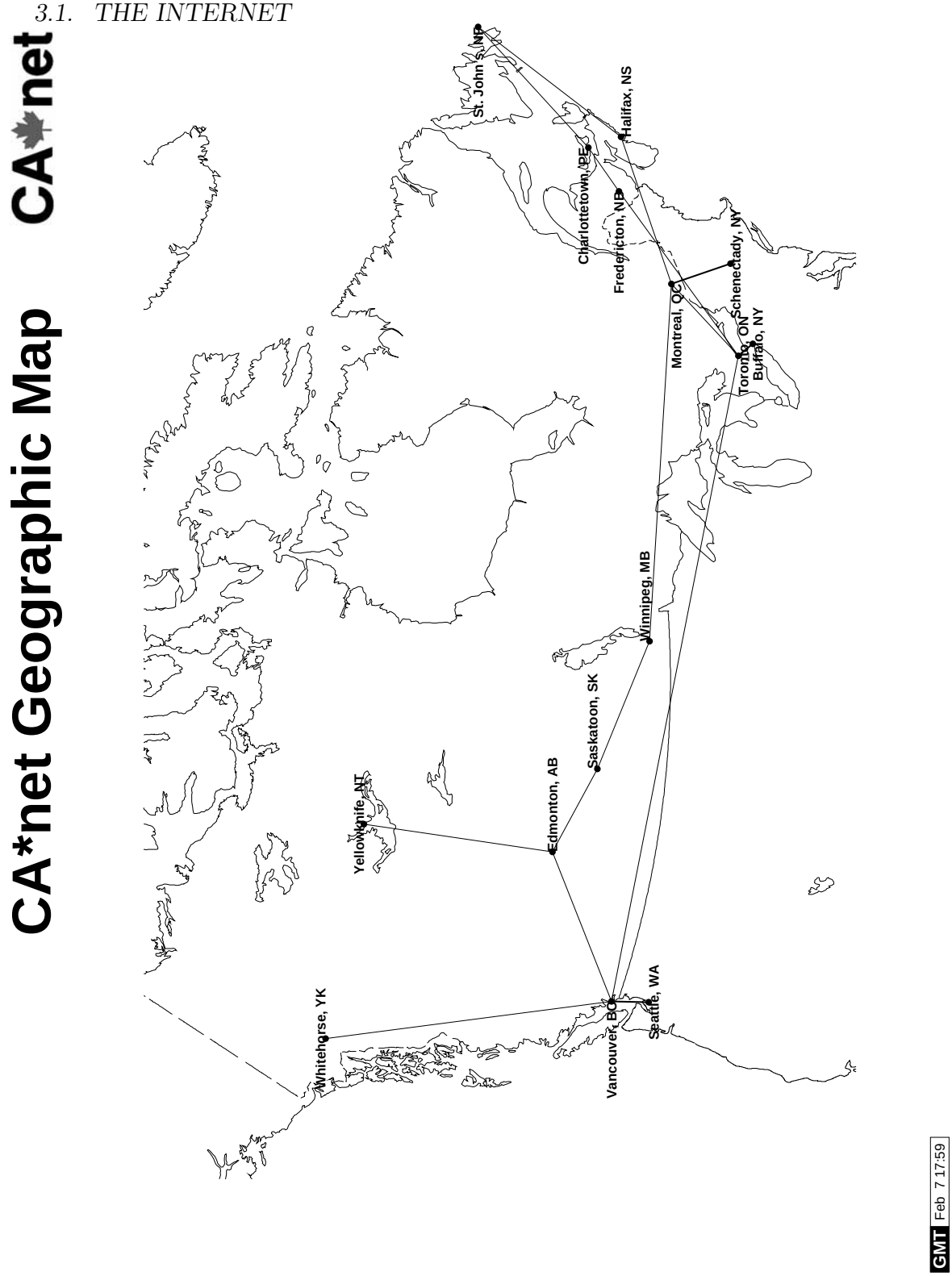


Figure 3.3: Structure of CA*net (before it became BITS)

Province/Territory	Regional Network
Yukon	YKnet
Northwest Territories	NTnet
Nunavut	???
British Columbia	BCnet
Alberta	NETERA
Saskatchewan	SRnet
Manitoba	MRnet
Ontario	ONet
Québec	RISQ
New Brunswick	ACORN NB
Nova Scotia	ACORN NS
Newfoundland	ACORN NF
Prince Edward Island	ACORN PE

Table 3.1: Canadian Regional Networks.

speeds were to be 155 Mbps, the fastest and largest national network in the world (at start-up time).

CA*net 2 is connected in each province to Regional Advanced Networks (some provinces have more than one connection to their RAN), via GigaPoPs (Gigabit Point of Presence). A list of Regional Advanced Networks is shown in Table 3.2, based on October 1997 data.

Figure 3.4 (courtesy of CANARIE Inc.) shows the topology of CA*net 2, as of October 1997. As mentionned above, this topology is constantly changing.

CA*net 2 would connect to the US "vBNS" (very high-speed Broadband Network Service) and "Internet 2" initiatives. CA*net 2's European connection was done via the Halifax GigaPoP.

NRC was connected to CA*net 2 via an ONet GigaPoP located in building M-60 of the Ottawa Montreal Road Campus. The Canadian Bioinfomatics Resource (CBR) based in the Institute for Marine Biology in Halifax, was the driving force behind that connection.

CA*net 2 would be replaced by CA*net 3.

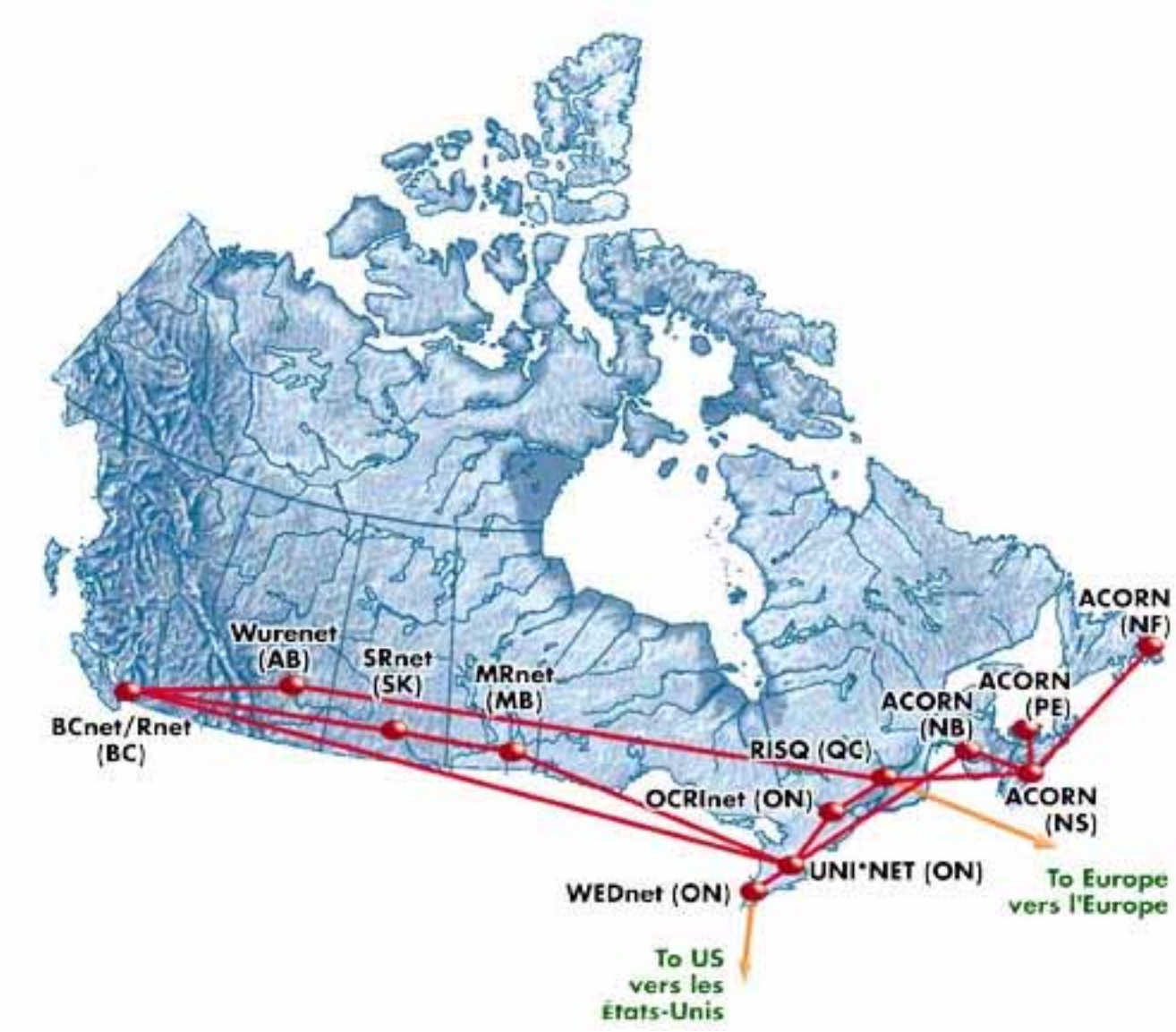


Figure 3.4: Structure of CA*net 2

Province	Regional Advanced Network
British Columbia	BCnet/Rnet
Alberta	Wurenet
Saskatchewan	SRnet
Manitoba	MRnet
Ontario	LargeNet
Ontario	ONet
Ontario	UNI*NET
Ontario	WEDnet
Québec	RISQ
New Brunswick	ACORN
Nova Scotia	ACORN
Newfoundland	ACORN
Prince Edward Island	ACORN

Table 3.2: CA*net 2 Connectivity (October 1997).

CA*net 3

In February 1998, CA*net 3 was announced. This fibre optic network was the first of its kind in the world, as it was developped first and foremost as a fibre optic/data based network. It could deliver up to 40-gigabit capability (that is 250 times faster than CA*net 2). Imagine: the 2.5 hour movie Titanic downloaded in .5 seconds.

CA*net 3 was the world's first national optical network. Instead of using 1 light within a strand of the fibre optic cable, it uses 32 lights of different colours, thus providing for a much larger bandwidth. This is known as Dense Wavelength Division Multiplexed (DWDM) technology.

CA*net 3, as was the case when CA*net 2 was first developed, was only available to researchers at universities, and government laboratories engaged in research and applications development relating to high-performance networks.

The March 1999 CA*net 3 (proposed) topology is presented in 3.5.

Access points to CA*net 3 are shown in 3.3.

More up to date information may be accessed at

`http://www.canarie.ca`

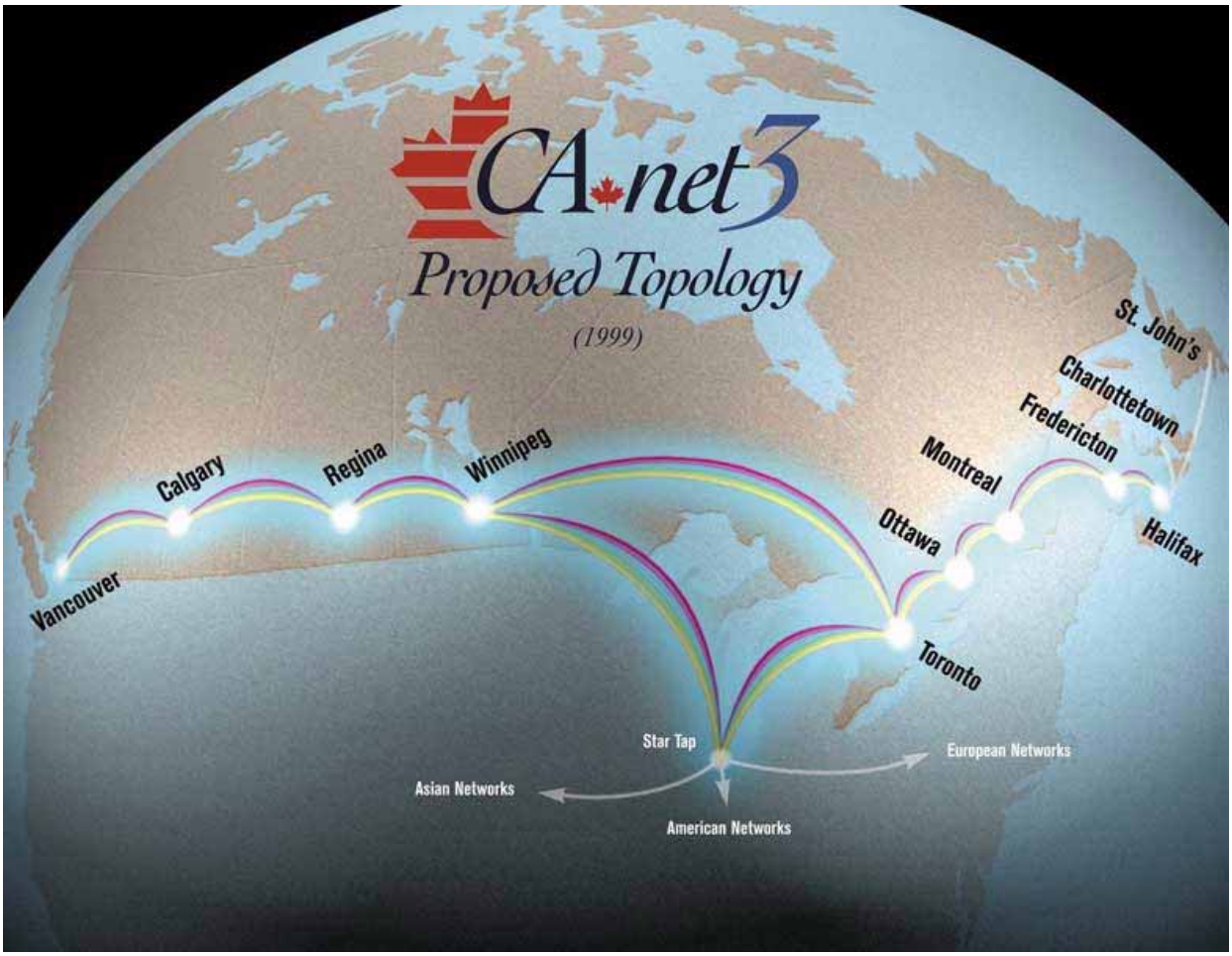


Figure 3.5: Topology of CA*net 3

Province	gigaPOP location	backbone node
British Columbia	BCnet	Vancouver
Alberta	WURCnet	Calgary
Saskatchewan	SRnet	Regina
Manitoba	MRnet	Winnipeg
Ontario	ONet	Toronto
Ontario	ARDNOC	Ottawa
Ontario	NRC	Ottawa
Ontario	CRC	Ottawa
Québec	RISQ	Montreal
New Brunswick	UNB	Fredericton
Nova Scotia	DAL	Halifax
Nova Scotia	UPEI	Halifax
Nova Scotia	Memorial	Halifax
US	Chicago	

Table 3.3: CA*net 3 Connectivity.

`http://www.canet3.net` (this will bring you to the CA*net 4 page!)

CA*net 4

As has happened with CA*net 2 and CA*net 3, a newer, faster, better, more flexible network followed.

During 2002, CA*net 3 was therefore replaced with CA*net 4. As with its predecessors, CA*net 4 connects the provinces research networks using state-of-the-art network technology. Initial connections to the sites is at OC-192 (10Gbps).

A Map of CA*net 4 is shown in 3.6.

Summary

In summary, smaller networks (NRC's) are connected to bigger networks (ONet) which in turn are connected to still bigger networks (BITS, CA*net 2, CA*net 4), inter-connected with other similar networks (NSFnet, Internet 2).

Each `node` on the network is assigned a name (and, transparently, a numeric IP address).

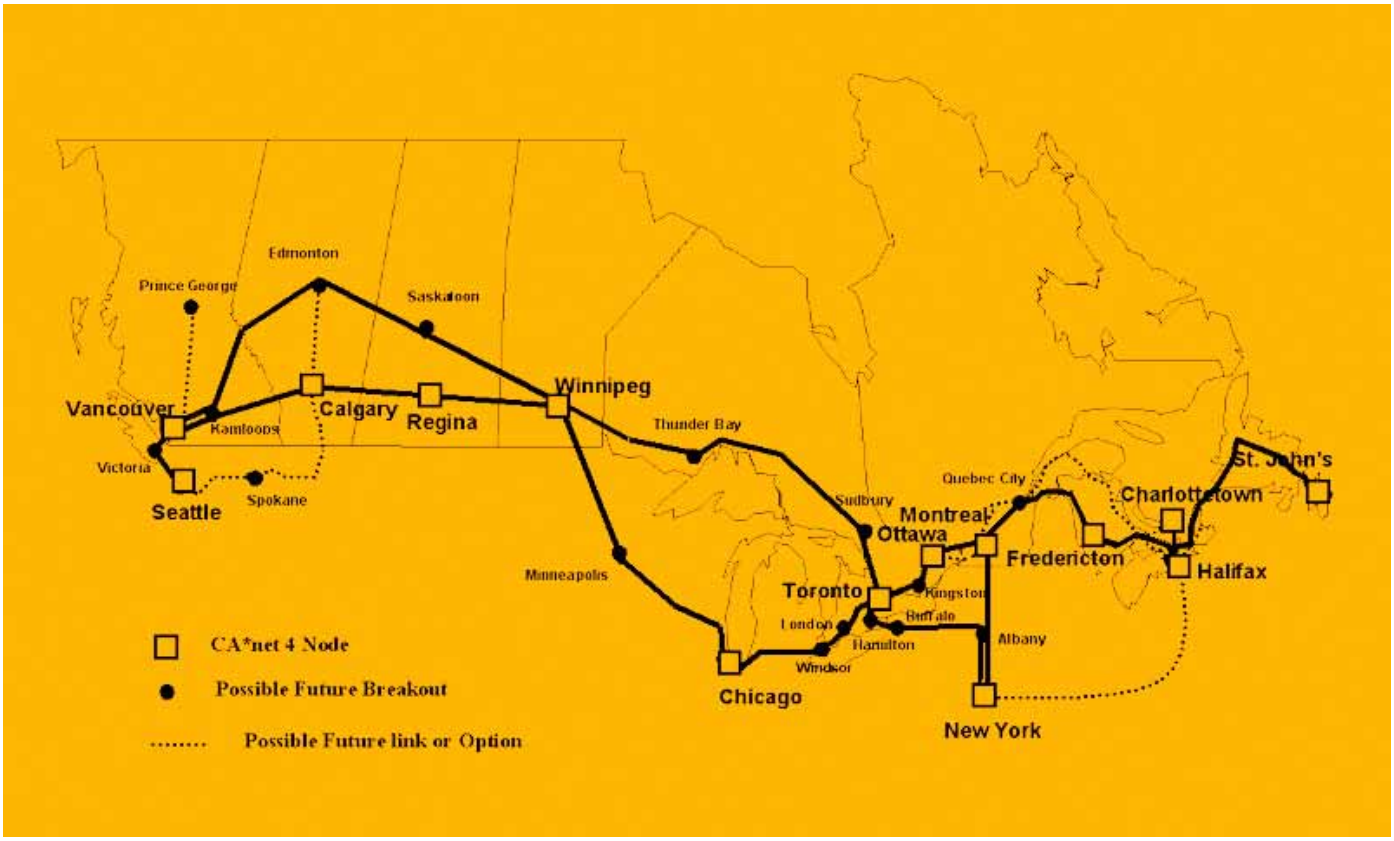


Figure 3.6: Topology of CA*net 4

Tools to access the information distributed on the Internet are gaining in popularity. Traditional tools like `gopher`, `USENET News Readers` have been replaced with web browsers such as `Mozilla`, `Netscape` and `Internet Explorer`. The goal of these tools is to provide assistance to users in searching and collecting the wide range of information available on the Internet.

3.1.2 Cost

Each member of the network has to pay their share of the cost of the lines. NRC's cost for the 2000/2001 fiscal year for connecting to the Internet, BITS, CA*net 2 and CA*net 3 was $263,000.

NRC's allowed bandwidth on CA*net 2/3 is currently (March 2001) set to 5 Mbps with bursts (if the bandwidth is available) of up to 12 Mbps. The non CA*net 2/3 brandwidth available to NRC is 6 Mpbs with 12 Mpbs bursts.

3.1.3 Basic Internet Protocols

IP: Internet Protocol

The Internet set of networks are all based on `IP`, the `Internet Protocol`. "The Internet Protocol (IP) takes care of addressing, or making sure that the routers know what to do with your data when it arrives." [5, p. 25]

Data is transmitted in a series of small chunks, called *packets*, each approximately 1200 characters in length.

The header of each *packet* includes the destination address of the system to receive the packet. The address of that system is the *IP address*.

All IP addresses consist of four fields of numbers ieach one less than 256.

Any packet containing IP addresses in the form

`132.246.`*xxx.xxx*

are for systems owned by NRC (actually, it is more precise to say that they are destined for systems behind the main NRC National Capital Region router). The routers in the network use this number to decide where to forward each packet.

Some addresses, however, are used for internal networks only. Three sets of addresses were set aside for that specific purpose:

- 10.0.0.0 – 10.255.255.255

- 172.16.0.0 – 172.16.255.255
- 192.168.0.0 – 192.168.255.255

Those sets of addresses should never be routed through the Internet.

TCP: Transmission Control Protocol

The Transmission Control Protocol describes how those IP packets are to be transmitted. The TCP software ensures all packets are received and that all packets are placed in the proper order. It then "extracts the data, and puts it in the proper order." [5, p. 27]

UDP: User Datagram Protocol

The User Datagram Protocol is typically used by applications requiring small amounts of data transfer (less than one packet at one time). UDP may be viewed as "TCP's small brother" since it handles one packet at a time instead of a string of them.

3.1.4 DNS: Domain Name Service

The Domain Name Service is also know as `name server`. It translates IP addresses to system names and system names to IP addresses.

The DNS is responsible for translating

```
nickel.sao.nrc.ca
```

which humans understand, to the address

```
132.246.10.3
```

which routers (the equipment responsible to route the packets through the network) and computers understand.

NRC's main name server (and thus, DNS) in Ottawa is

```
dns1.nrc.ca
```

3.1.5 Node/Machine Names

By convention, the name of systems follow the style

machine_name.division.organization.country (outside USA)
or
machine_name.division.organization.org_type (inside USA)

where

- *machine_name* is the personal name of the system.
- *division* is the division, institute or group which owns that workstation.
- *organization* is the name of the company which owns the computer.
- *country* is a two letter country code, such as `ca` for Canada, `fr` for France, `uk` for United Kingdom, etc.

  The official list of two letter accronyms may be found at `http://www.iana.org/cctld/cctld-whois.htm`.
- *org_type* is the type or organisation, as shown in Table 3.4.
- In November 2000, ICANN (the Internet Corporation for Assigned Names and Numbers) approved a list of seven (6) new Top Level Domains. Shown in 3.5, those new Top Level Domain names were not expected to be in use before the Fall of 2001. In fact (Winter 2003), they have yet to be noticed by the casual web surfer.

An example is

```
nickel.sao.nrc.ca
```

where `nickel` is the *machine_name* owned by the group/branches `sao` (Science Affairs Office – now called Research Program Support Group) of the `nrc` (National Research Council) , in `ca` (Canada).

org_type	Description
arpa	Old Style Arpanet
com	Commercial Organisation (mostly US)
edu	US Educational
gov	US Government
int	International
mil	US Military
nato	Nato field
net	Major Network gateway, ISP
org	Non-profit Organisation (mostly US)

Table 3.4: Main Domain Names (non-country).

3.1.6 `NIC`, `ICANN`: Internet Management

If every system on the network must have a unique IP address, who is ultimately assigning those addresses to institution?

The answer this question is the Internet `Network Information Center`, also known as the `InterNIC`.

They may be contacted at

```
www.internic.net
```

Some of the `InterNIC`'s previous responsabilities have been transferred to `ICANN` (The

org_type	Description
aero	Air-transport industry
biz	Businesses
coop	Cooperatives
info	Unrestricted use
museum	Museums
name	Registration by individuals

Table 3.5: New Top Domain Names (2001)

Internet Corporation for Assigned Names and Numbers). `ICANN`'s responsibilities include IP address space and domain name service (DNS) management (including root DNSs).

`ICANN` information is at

`www.icann.net`

3.2 Finding Users and Communicating with them

There is no central place on the network where one could look to find out the electronic mail address for a specific person. It would be like asking if there was one place in the world where you could get the phone number (or address) of any individual.

The Internet is a network of networks. Individual networks may have a database of their own users and their addresses. Some do not.

There is no standard by which one can get (or be) registered. There is also the "privacy issues" to consider (many would argue the telephone directories are no different than email addresses. The main differences is that the phone companies have had 80 years to get together and collaborate. There are also a lot fewer telephone companies than there are Internet Service Providers + Organisations).

There are a number of ways, within UNIX/Linux, one can use to try to find out information about users. The most often used are the `finger` and `rusers` commands. But because of security issues, many systems disable the use of those services.

On the systems maintained by the Research Computing Support Group, the `phone` command is available. That command is mapped to

`whois -h gold.sao.nrc.ca` *person_name*

One may also use Web Browsers to find information about a specific person. Use a search engine, and enter the person's name in quotes. Results may amaze you!

3.2.1 `finger`: Point Finger To

finger [*user*[@*node*]]

where *user* is the logon name of the user whose information is to be displayed, and *node* is that user's machine name.

With no arguments, `finger` displays all users presently logged on, who they are, and the time each of them logged on.

When information concerning a specific user is sought, the information displayed is more detailed: the logon name, and the real name are displayed , the `home` directory of the user is displayed, as well as the logon shell, and the last logon time.

user does not need to be logged on for `finger` to work.

```
prompt> finger cantin

Login name: cantin                      In real life: Claude Cantin
Office:  2025 Sx, 993-0822
Directory: /usr2/people/cantin           Shell: /bin/tcsh
On since Jul 22 08:49:29 on ttyq0       2 hours 59 minutes Idle Time
On since Jul 22 08:49:34 on ttyq1       2 hours 33 minutes Idle Time
On since Jul 22 10:11:35 on ttyq4 from dp1.phy.nrc.ca
On since Jul 22 09:11:02 on ttyq2       1 hour 53 minutes Idle Time
On since Jul 22 11:00:38 on ttyq5 from dp1.phy.nrc.ca
48 seconds Idle Time
Plan:

Progresser, progresser, progresser...
```

In the above example, `cantin` is presently logged on. His real name is Claude Cantin. His home directory is `/usr2/people/cantin`. He is logged on to his account five times, two of which are from a remote location (`dp1.phy.nrc.ca`)

Furthermore, he has a file named `.plan` in his home directory, whose content is

> Progresser, progresser, progresser...

If he did not have the `.plan` file, the string

> No Plan.

would have been displayed instead.

In the next example, Ratilal Haria, on *node* `nrcnet0`, last logged on Friday December 8th, at 10:52, from the node whose I.P. (Internet Protocol) address is 132.246.20.10. That node was using a windowing interface (we know this because it was logged on to `ttyp0` on that machine – not the console).

```
prompt> finger ratilal@nrcnet0
```

```
[nrcnet0.nrc.ca]
Login name: ratilal                 In real life: Ratilal Haria
Directory: /home/nrcnet0/ratilal    Shell: /bin/csh
Last login Fri Dec  8 10:52 on ttyp0 from 132.246.20.10
No Plan.
```

It should also be pointed out that some systems may not allow local users to be `finger`ed by remote systems. In that that case, and error message would be displayed:

```
prompt> finger cantin@cansnd.cisti.nrc.ca
finger:  cansnd.cisti.nrc.ca:  connection refused
```

3.2.2 `rusers`: Remote Users

To find out if other users on the same network are logged on,

rusers [*node...*]

`rusers` by itself simply gives all the logged on users on the given network. If *node* is specified, then only that node(s) is checked.

3.2.3 `talk`: Talk To

If user_a wants to talk to user_b and both are logged on, he/she may want to use the `talk` command:

talk *user_b*[@*node*]

When `talk` is invoked, the screen of the originator is split into two equal parts (a horizontal line of hyphens separates the two parts). *user_b* receives a request to answer:

```
Message from Talk_Daemon@neon at 9:59 ...
talk:  connection requested by smith@neon.sao.nrc.ca.
talk:  respond with:  talk smith@nickel.sao.nrc.ca
```

Once *user_b*, in this case `smith@neon`, responds to `cantin@nickel`, his screen will be split in two: the top portion of the screen always displays the local user's messages, and the bottom portion the remote user's message. Characters are displayed as they are typed.

To close the connection, press <CTL-C>.

As with `finger`, many systems do not allow non-local systems to use the `talk` facility on the local system.

3.2.4 `write`: Write To

For line mode terminals,

write *user_b*

is used. As with `talk`, `write` is used to communicate between logged-on users. BUT unlike `talk`, `write` is not "conversational".

When `write` is invoked, the user types in the message to be sent: every time a carriage return is typed, that line is sent to the remote user. <CTL-D> is used to get out of line mode.

3.3 Connecting to other Systems

`Traditionally, UNIX/Linux systems have used telnet to connect to remote systems. But if someone were to use a` *network sniffer* `(a program which allows the reading of all information flowing through the network), not only could they eavesdrop on your connection, but they could see your login AND password as you type them!`
`telnet connections are therefore considered` *non-secure*.

One way to avoid this is to ensure all data exchanged is encrypted. This is what SSH – Secure SHell – does: the initial connection between two system allows for the transfer of information, including public keys. A session ID is calculated by both parties. That session ID is then used as part of every piece of information exchanged, to ensure the connection is encrypted, and thus considered much more *secure*.

`ssh` is quickly becomming the prefered method for connecting to remote systems.

`ssh` is first introduced (because it is considered *secure*), even though it is still not found on all UNIX/Linux installations. `telnet`, on the other hand, is found on all UNIX/Linux platforms, as well as on the MS Windows 95/98/2000/NT systems.

3.4 SSH: Connecting Securely to other Systems

SSH itself is a protocol. Older NRC/RCSG supported systems use mostly version 1.5 of the protocol (version 1.2.`xx` of the `ssh` tool). All newer systems (from year 2000+) use openSSH, which supports both versions 1 and 2 of the protocol.

Version 2 of the SSH protocol is not compatible with version 1. Although it is considered more secure, and may provide more flexibility, not many public domain client and server packages are available for it.

The following commands are tools based on the SSH protocol to ensure encrypted sessions between systems.

3.4.1 `slogin`, `ssh`: Secure `telnet`

ssh [**-l** *login*] *node*
or
ssh [*login***@**]*node*

allows user *login* to initiate a session with system *node*.

`slogin` is another name for the `ssh` command. Both work the same way – only the name differs.

If there is not a proper `.rhosts` or `.shosts` file entry, a password will be required. That password will NOT be sent in plain text. It will be sent in encryption form.

The default encryption method uses RSA encryption and decryption, using the public/private key scheme: during the initial handshake between the two systems, they exchange their public keys (encryption keys). When any message is sent to the other party, it is first encrypted using that party's public key. When the message is received, it is read using the host's own private (or decryption) key.

The private key is NEVER given to anyone. Only the public key is.

The public key encrypts messages, while the private key decrypts them. Even is the message is intercepted by a third party, they cannot decrypt it because they have no access to the private key of the destination host.

The session is therefore *secure*.

Most RCSG supported SGI/IRIX NRC systems configured to use SSH have had their `telnet` and `rlogin` commands modified so that they use the SSH protocol if the remote host will accept it.

Escape sequences

During your SSH session, you may want to come back to your original shell, without breaking your connection. A series of escape sequences allow you to perform that function:

Supported escape sequences:
~. - terminate connection

~^Z - suspend ssh
~# - list forwarded connections
~& - background ssh (when waiting for connections to terminate)
~? - this message
~~ - send the escape character by typing it twice
(Note that escapes are only recognized immediately after a newline.)

For example, to suspend a current SSH session:

```
~^Z
```

(that's *tilde*, followed by CTRL-Z). To resume execution of the session:

```
fg
```

If, for some reason, there are network problems, the current SSH session "hangs", and you want to disconnect from it:

```
~.
```

3.4.2 `ssh-keygen`: password-less SSH login

SSH is often used to login from one system to another without requiring passwords.

A number of methods may be used for that to work properly, one of which is to setup a `.rhosts` file (permission 600) with its content being the name of the remote system you trust, followed by the username your trust:

```
nickel.sao.nrc.ca cantin
```

would mean you trust user `cantin` from `nickel.sao.nrc.ca` to connect to your account, without requiring a password.

But for that to work, SSH itself must be configured to trust `.rhosts` files (which it does not for most OpenSSH installations – but we do on most systems RCSG maintains), and the private/public key pair of each system must be properly set in the system-wide `ssh_known_hosts` public key file.

This, of course, requires help from the local systems administrator.

The second method does not require any help from the systems administrator. And it does not require modifications to the `.rhosts` file. Instead, it requires you generate your own personal set of private/public pair.

`ssh-keygen` is used to generate that key pair for you. Here is a session where your own personal private/public key pair is created:

```
cantin@sodium:~> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cantin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cantin/.ssh/id_rsa.
Your public key has been saved in /home/cantin/.ssh/id_rsa.pub.
The key fingerprint is:
f6:61:a8:27:35:cf:4c:6d:13:22:70:cf:4c:c8:a0:23 cantin@sodium
```

The command `ssh-keygen -t rsa` initiated the creation of the key pair.

No passphrase was entered (`Enter` key was pressed instead).

The private key was saved in `.ssh/id_rsa`. This file is read-only and only for you. No one else must see the content of that file, as it is used to decrypt all correspondence encrypted with the public key.

The public key is save in `.ssh/id_rsa.pub`.

In this case, the content of file `id_rsa.pub` is

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEArkwv9X8eTVK4F7pMlSt45pWoiakFkZMw
G9BjydOJPGHORFNAy1QqIWBGWv7vS5K2tr+EEO+F8WL2Y/jK4ZkUoQgoi+n7DWQVOHsR
ijcS3LvtO+5ONp4yjXYWJKh29JL6GHcp8o7+YKEyVUMB2CSDOP99eF9g5Q0d+1U2WVdB
WQM= cantin@sodium
```

It is one line in length.

Its content is then copied in file `.ssh/authorized_keys` of the system you wish to SSH to without being prompted for a password.

The example shown here generated keys on `sodium` by user `cantin`. If the public key generated, file `.ssh/id_rsa.pub`, was copied to your account, file `.ssh/authorized_keys` on `nickel.sao.nrc.ca`, then user `cantin@sodium` is allowed to SSH into your own account on `nickel.sao.nrc.ca` without the use of a password.

To summarize, a personal private/public key pair is generated using the `ssh-keygen` command. The public key is then copied onto a remote systems' `.ssh/authorized_keys` file. And you can now SSH to the remote systems's account without the use of a password.

3.4.3 `telnet`: Connecting to Remote Systems – (*non-secure*)

telnet *node*

allows a connection to *node*, from any other machine connected to the Internet. *node* itself will prompt for a logon name, then the password.

Logging on remotely allows users to use the UNIX/Linux commands, but not the Graphics User Interface of the remote machine.

Logging out of the remote machine will close the connection between the host and the remote system.

You cannot use `telnet` to connect to any of the systems mananged by the Research Computing Support Group: it was completely disabled in 2000.

3.4.4 `rlogin`: Bypassing Password – (*non-secure*)

rlogin *node*

allows a connection to *node* without the use of a password.

For this to be true, two things must be properly set.

1. You must have an account (of the same name) on the remote system.
2. File `.rhosts` on the remote system must contain the name of the local system, followed by your account name.

`rlogin` is essentially the same as a `telnet` but assumes the same account name, and looks for the `.rhosts` file on the remote machine. You can therefore not use `rlogin` to connect to any of the Research Computing Support Group's managed systems.

3.5 Logging on to NRC from home

3.5.1 Cisco Server/M-60 dial-in access

NRC no longer maintains its M-60 dial-in access. That service was discontinued on April 1, 2001.

NRC contracted a local ISP, EduNET to replace that service.

3.5.2 EduNET dial-in server

This service replaces the "M-60/Cisco Server Dial-up facility".

Since the summer 2000, NRC personnel may get an account with `EduNET`, a (Ottawa) local Internet Service Provider (ISP). NRC has an agreement with that ISP, which allows

approved NRC personnel to have an account with them. Once connected to the EduNET ISP, the NRC employee becomes part of the NRC internal network.

This allows them to access NRC-only services such as CISTI services and the NRC/zone web pages. It also allows users to connect to internal NRC UNIX/Linux systems.

There is a minimum monthly cost for each account. That cost is paid by the individual institutes (in some intitutes, by the individual groups). EduNET registration information is available either from your institute corporate office, or from your computing support personnel.

3.6 Connecting to UNIX/Linux from MS Windows-based Systems

Although these course notes are for UNIX/Linux, many people use PCs running a Microsoft-based operating system such as Windows 95/98/2000/NT to access their UNIX/Linux servers. Traditionally they have relied on "stock" programs like `telnet` and `ftp` to access their systems. They have also used tools like Eudora or Outlook to read their UNIX mail.

Since the spring of 2001, all communication done with UNIX/Linux must be done through a secure channel. Between UNIX/Linux systems, that secure channel is created when using `ssh` and `scp`.

The Research Computing Support Group (RCSG) has put together a series of tools people can install on their PCs, to access the UNIX/Linux systems both within NRC, and from the NRC dial-up access.

The tools covered include `putty` and `WinSCP`:

- `putty` is a SSH-based telnet-like client. It allows for secure communication between Windows and UNIX/linux, much the same way `ssh` does on the UNIX/linux platforms.

  It has a wide range of configuration for fonts, colours, behaviour. If you run X on your PC, `putty` allows the tunnelling of X applications (option must be enabled within `putty`).

  Its basic installation requirement is the download of one executable `.exe` file, but the full package includes command and batch capable utilities.

- `WinSCP` is the UNIX/linux equivalent of `scp`. Graphical-based, it allows for the safe/encrypted transfer of files to/from Windows and UNIX/linux platforms.

More details about those tools, as well as downloadeable modules may be found at

`http://www.nrc.ca/imsb/rcsg/ras/ssh-clients.html`

That web page also explains how GUI-based PC FTP tools, how mail tools like Eudora and Outlook, may be safely used, through the secure channel created by SSH.

3.7 Transferring Files Between Systems

3.7.1 `scp`: Secure Copy

scp *filename(s) node:filename*
or
scp *node:filename(s) filename*
or
scp *filename(s) user@node:filename*
or
scp *user@node:filename(s) filename*

where *node* is the name of the remote system, and *user* is the login name to use on the remote system.

`scp` allows the copy of file(s) between systems. The files are transferred in encryption form.

If proper `.rhosts` or `.shosts` files exist, no password will be required. If they do not exist, or the proper entry does not exist, a password will be required.

The encryption method is the same as with `ssh`.

3.7.2 WinSCP: Windows Secure Copy

Although the purpose of these notes are for linux/UNIX users, many people require moving files to/from the PC/Windows environment from/to linux/UNIX. The recommended MS Windows package to perform that operation is `WinSCP`, an open source package, available primarily at

`http://sourceforge.net/projects/winscp/`

but also made available at NRC on our web site at

`http://www.sao.nrc.ca/imsb/rcsg/ras/ssh-clients.html`

3.7.3 FTP: File Transfer Protocol – (*non-secure*)

`This is the traditional way of transferring files with UNIX/Linux. Since it involves using plain-text login/password combinations, it is considered non-secure. Many systems, including most systems managed by the Research Computing Support Group at NRC, do not allow connections to their FTP servers using that tool.`

`The exception to that is the` *anonymous* *FTP* `servers the Research Computing Services Group maintains.`

ftp *node*

A connection to *node* will take place, and a logon name will be required, as well as a password.

Within the `ftp` tool, a restricted number of commands are available:

?: Help

?

will display the list of available commands.

? *command*

will display a short message explaining what *command* does.

`put/send`: Transfer to Other

put [*local_file* [*remote_file*]]

If invoked by itself, `put` will prompt for *local_file*, then for *remote_file*. If only one parameter is used, *remote_file* will be assumed to be the same as *local_file*.

`mput`: Multiple Transfer to Other

Used the same way as `put`, this command also accepts metacharacters. It will prompt for every file to be transferred, unless the `-i` flag was used on the `ftp` command line, or the `prompt` command is issued.

`mput` is very useful when moving a number of files.

`get/recv`: Transfer From Other

The syntax for `get` is the same as for `put`:

get [*remote_file* [*local_file*]]

If invoked by itself, `get` will prompt for *remote_file*, then for *local_file*. If only one parameter is used, *local_file* will be assumed to be the same as *remote_file*.

If *local_file* is set to `-`, it will be taken as standard output (i.e. the screen). For example,

get README -

will display the content of `README` on the screen.

get *remote_file* "| *command*"

will run the local *command* using *remote_file* as input.

Example:

get README "| more"

`mget`: Multiple Transfer From Other

Used the same way as `get`, this command can use metacharacters, but will prompt the user before every file transfer unless the `-i` flag was used with the `ftp` command, or the `prompt` command is issued.

As with `mput`, this command is very useful when moving a group of files.

`bin`: Binary

bin

is used when transferring binary files. Files containing binary code include compiled programs, commands, tar files (*file*.`tar`), compressed files.

The response to this command is

Type set to I.

`as`: **ASCII**

as

is used when transferring ASCII files (text, source code, etc.).

The response to this command is

Type set to A.

`cd` : **Change Directory**

cd *directory*

is used to change directory in the remote machine.

`ls` : **Listing**

ls [**-lsa**]

is used to produce a listing of the files on the remote machine. The meaning of the flags is the same as for UNIX's `ls`.

If the form

ls *argument filename*

where *argument* may be either a list of options (such as `-l`) or a filename(s) on the remote site, the output from the command would be put in *filename* on the local system.

Another form,

ls *remote_file* "| *command*"

would pipe the output of the remote `ls` directive into the local *command*.

A typical *command* may be `more` or `grep`, such as

ls * "| more"

`quit`:

quit

exits `ftp`.

3.7.4 `.netrc`: FTP in batch

File

`.netrc`

in your $HOME directory allows file transfers in batch mode. Permission on that file should be `600` (`rw-------`); it will likely not work otherwise.

Each record has the format:

`machine` *machine* `login` *login* `password` *passwd*

where each of *machine*, *login* and *passwd* refer to a system name with the login and password for that account on the machine.

Typically, this is used to do anonymous FTP transfers, so an entry could look like:

```
machine nrcnet0 login anonymous passwd cantin@nrc.ca
```

FTP commands to be used would then be put in some file, and the command to be executed would be similar to

ftp *machine* < *filename*

where the content of *filename* could look like

```
bin
cd /pub
get README
quit
```

3.7.5 sftp: secure FTP

There are two incompatible versions of the `sftp` package:

- SSHv2 companion program.
- an independant client/server version

Systems with SSHv2 (ex: openSSH) use the SSH-compatible package. The older IRIX systems at NRC use the other version (if it was installed).

Both are used in a very similar way.

`sftp [`*user*`@]`*node*

`sftp` allows the transfer of file, using interactive commands very similar to `ftp`. The entire `sftp` session is encoded, as are the login/password tokens.

The basic commands are the same as those of the `ftp` command: `cd`, `get`/`mget`, `put`/`mput`, `help`, `quit` are all used the same way as their `ftp` counterpart.

`sftp` ignores `.netrc`.

3.7.6 Anonymous FTP

An `anonymous FTP` account is a special account for which no password is necessary. The login id is either "`ftp`" or "`anonymous`" and the password is typically your E-mail address.

`anonymous FTP` accounts are typically used to provide unrestricted access to freely available files on various sites on the Internet.

Public Domain packages are usually put in such accounts (although many sites now put them within their web server area, for added access control/security).

3.7.7 `archie`/`xarchie`: accessing FTP databases.

`archie` is no longer used, but kept here for historical reasons.

There are thousands of anonymous FTP systems world-wide. They all contain different files and programs.

Prior to the web and its browsers, the best way to find out which application was located where, `archie` was the tool of choice. It was created circa 1990.

The `Archie` database was constructed bit by bit over the period of one month. During that month, each night, `Archie` visited 1/30 (approximatly) of the anonymous FTP sites registered with it, and collected all file/program names stored at each and every location.

`archie` was then used to interrogate the `Archie` database to find the location of a specific program. `xarchie` is a graphical version of the tool.

Although `archie`/`xarchie` is still in use, web searches through search engines is currently a more efficient measure in finding tools on the Internet.

3.8 The Network News

The `Network News`, also know as `NetNews`, is the largest Bulletin Board Service in existence. It may be considered as the "first-generation bulletin boards".

3.8.1 NetNews terminology

`category`: top level name of the organisational structure of newsgroups.
`newsgroup`: is a special interest group.
`article`: is a post to a specific `newsgroup`.
`thread`: a set of articles with the same subject.
`newsfeed`: workstation used to propagate the news to `newsservers`.
`newsserver`: a workstation storing news articles.
`newsreader`: a program used to read the `NetNews` off the `newsserver`.

3.8.2 newsgroup Organisation

The `newsgroups` are organised in a hierarchical tree, each level separated by a dot. The top level is called a `category`.

The [original] `USENET News` consisted of 7 categories/groups (based on [5, p. 154]). Table 3.6 displays those categories.

Category	Description
comp	computer science related.
news	network news related.
rec	hobbies; recreational activities; arts.
sci	scientific research and applications.
soc	social issues; politics.
talk	forum for debate on controversial topics.
misc	other topics.

Table 3.6: USENET News Categories.

But since any `newsgroup` may be created locally and be distributed to all other `news servers`, a number of other categories are now accessible from most sites.

The most important other `categories` are (based on [5, p. 155-6]) shown in Table 3.7.

Any `newsserver` may chose which groups are allowed on the system. NRC has only a subset of the `Network News` groups on the main `newsserver`, `news.nrc.ca`, in addition to many other categories, some of which are seen in Table 3.8.

Category	Description
alt	alternate groups that don't quite fit anywhere.
bionet	for biologists.
bit	BITNET listserv discussion groups.
biz	for business; one of the only categories where advertisement and marketing are allowed.
de	different groups, in the German language.
fj	different groups, in Japanese.
hepnet	High Energy Physics.
ieee	related to the Institute of Electrical and Electronic Engineers.
info	mailing lists on wide variety of topics.
gnu	Free Software Foundation and GNU project.
k12	for teachers and students of kindergarten to grade 12.
relcom	groups originating in the former USSR.
u3b	AT&T 3B computer series.
vmsnet	Digital VAX/VMS and DECnet groups.

Table 3.7: Other important Network News Categories.

3.8.3 FAQ: Frequently Asked Questions

Most `newsgroups` maintain a `FAQ`, regularly posted to the group.

A `FAQ` is a list of `F`requently `A`sked `Q`uestions (with answers) typically asked in a specific `newsgroup`.

The easiest way of accessing the news FAQs is to use the web at URL

```
http://www.faqs.org
```

That site contains FAQs from a large percentage of the USENET newgroups.

3.8.4 Newsreaders

`newsreaders` are programs used to access and read the news. They usually fetch the news from the `newsserver`.

All `newsreaders` are capable of performing at least the following tasks:

Category	Description
nrc	NRC related; eg: `nrc.test`, `nrc.events`.
uw	University of Waterloo; eg: `uw.unix`, `uw.outers`.
ut	University of Toronto; eg `ut.chinese`; `ut.dcs.graphics`.
ott	Ottawa related; eg: `ott.general`, `ott.forsale`.
tor	Toronto specific; eg: `tor.news`, `tor.test`.
ont	Ontario-wide; eg: `ont.jobs`, `ont.uucp`.
can	Canadian groups; eg: `can.newprod`, `can.domain`.

Table 3.8: More Local Network News Categories.

- Read an article.
- Post a `follow-up` article (i.e. post an article within the same thread/subject).
- Reply to the author of the article.
- Save an article.
- Print an article.
- Mail an article.
- Subscribe/Unsubscribe to newsgroup(s).

`newsreaders` use the configuration file

```
.newsrc
```

to find out which newsgroups you subscribe to. It is created the first time the `newsreader` is invoked.

That file may be viewed/edited with any text editor to find out which `newsgroups` are available to be read on that system.

`tin`: threaded Internet newsreader.

`tin` is a user-friendly menu-driven interface to the News. It is said to be `threaded` because it keeps all articles from the same thread together (a more typical newsreader lists the articles in the order they came in).

To invoke it, issue

```
tin
```

When `tin` is started for the first time, it will display a screenful of helpful advice. Pressing any key will then subscribe the user to all newsgroups distributed by the newsserver.

At this point, it is better to `unsubscribe` to all groups by pressing

U

`tin` will then ask for a `regex` (regular expression) to enter. Enter

```
*
```

to specify "all newsgroups".

You are now unsubscribed from all groups. You may then scan the group names using the arrow keys and issue

s

to subscribe to any single group you wish to.

Whenever `tin` is started, a new newsgroup may be added by pressing

g

(`goto` newsgroup) and enter the newsgroup's name. That newsgroup will be automatically subscribed to and its name added to the `.newsrc` file.

Next time you start `tin`, only groups to which you subscribe will be displayed.

There are three main levels:

1. `newsgroups`: this level shows you the newsgroups for which you are currently subscribed. Some possible operations are "`s/u`" (subscribe/unsubscribe) and "`g`" (goto a specific newsgroup).

   The numbers on the left side of the group names tell you how many unread articles are in each of the newsgroups.

2. `threads`: this level shows the list of threads in a newsgroup. The plus (+) sign on the left shows you which one have unread articles.

3. `articles`: you are now looking at the postings to the newsgroup. You may now reply, mail, print, etc.

The TAB key automatically displays (if at the articles level) the next unread article. If not at the articles level, it invokes the next sublevel.

q

will bring back the next higher level, until tin is exited.

One last option worthy of mention is

y

This toggle option (used at the newsgroups level) brings in all unsubscribed newsgroups, allowing you to scan them at your leisure. Issuing the option another time will "yank" out all unsubscribed groups.

There are numerous other options within tin. ‘‘h" within tin will display two help screens of commands to use.

Once in tin, the screen layout will appear as in Figure 3.7.

xrn: X-based news reader.

xrn is a package used to read the News, using the X window system as its interface. ASCII terminals cannot use this package; tin should be used with such terminals.

xrn is user friendly. The mouse is used to click on boxes to read, forward, reply, print, save, delete, and post articles. Customization instructions are found in the man page.

3.9 gopher: go fer it!

gopher is "a lookup tool that lets you prowl through the Internet by selecting resources from menus" [5, p. 233]. The Gopher system is a set of interconnected ASCII based menus. It was developed at the University of Minnesota. To cycle through a menu list, use the arrow keys. To select a menu, simply press the ENTER key.

Using gopher (also the name of a program to access the Gopher system), users can access files, copy them, print them, mail them. They can also FTP programs to their local machine, or access databases, phone books.

Gopher was developed in the late 80s early 90s. Many view it as "the first generation web browser".

It has since been superseeded by web browsers like Netscape and Internet Explorer.

```
Group Selection (nrcnet0.nrc.ca  15)

   1    54  comp.sys.hp                Discussion about Hewlett-Packard
   2    11  ott.general
   3    24  comp.sys.sgi               Silicon Graphics's Iris workstations
   4    45  comp.sys.sun.admin         Sun system administration issues and
   5    11  comp.sys.sun.apps          Software applications for Sun comput
   6    31  comp.sys.sun.hardware      Sun Microsystems hardware.
   7        sci.aeronautics.airliners
   8     6  comp.sys.sgi.admin
   9        comp.sys.sgi.announce
  10     7  comp.sys.sgi.apps
  11     1  comp.sys.sgi.bugs
  12    11  comp.sys.sgi.graphics
  13     6  comp.sys.sgi.hardware
  14     8  comp.sys.sgi.misc
  15        alt.gopher                 Discussion of the gopher information

       <n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
    g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unread,
      s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out

*** End of Groups ***
```

Figure 3.7: `tin` layout, `newsgroup` level.

3.9.1 Veronica: Searching Gophers

Veronica is to Gopher what Archie is to anonymous FTP. Veronica keeps a collection of all known/registered Gopher menus. People can then search the database to gather a collection of menu items pertaining to (hopefully) a specific domain.

Those menu items are automatically put into a gopher-like menu.

As with archie, veronica is now part of Internet history.

3.10 WWW: World Wide Web

3.10.1 What is it?

The World Wide Web is one of the newer information gathering services on the Internet (early to mid 90s). It is also the most successful tool, by far, used to access information. It is based on the concept of hypertext, first developed at CERN (Centre Européen de Recherche Nucléaire), the European particle physics lab in Geneva, Switzerland..

A hypertext is based on the concept that a word, phrase, or object is actually a link to another object. The hyperlink object is usually either underlined, highlighted, or of a different colour.

The link could represent many types of objects. When the link is activated (by clicking the mouse of a graphical browser, or pressing the ENTER key on an ASCII browser), a number of different scenarios may happen.

Table 3.9 displays the different actions the browser may take, depending on the type of link.

It is assumed the example browser being used is graphical in nature (eg: netscape).

The language used to write WWW documents is HTML (HyperText Markup Language).

http (hypertext transport protocol) is the protocol used to access/service HTML documents.

3.10.2 netscape, internet explorer, lynx, mozilla: WWW browsers

There are two styles of browsers. One is graphical, the other is ASCII.

The ASCII browser was developed at the University of Kansas and is called lynx. It shares many of the same properties as its graphical counterparts, Netscape and Internet Explorer.

linked object	action or program used
Local text file	File is displayed.
Remote text file	File downloaded, then displayed.
Remote `HTML` file	File downloaded, then interpreted.
Remote PostScript file	File downloaded; `ghostview` invoked.
Remote `GIF` file	File downloaded; `xv` invoked.
Remote audio file	File downloaded; `sfplay` invoked.
Remote video file	File downloaded; `mpeg_play` invoked.
Remote `DVI` file	File downloaded; `xdvi` invoked.
Telnet session to *host*	`xterm` invoked with `telnet` *host*.
Gopher site	Gopher site contacted; `gopher` menu started.
FTP site	FTP connection done; file may be downloaded.

Table 3.9: Some actions taken by WWW browsers.

The first graphical browser, `xmosaic`, was developed, as a research project, at the National Center for Supercomputing Applications (NCSA) at the University of Illinois. Version 1.0 of the tool was released in April 1993. After a few versions of the browser, it was decided the project had exceeded its goal, so it was terminated.

The last full version ever released was 2.6 (July 1995) and the last beta version released was 2.7b5 (July 1996).

`Netscape` was developed at Netscape Communications Inc. (formerly Mosaic Communications, Inc), a private software company, founded by Marc Andreesen (the original `xmosaic` developer) and Jim Clark (one of the original Silicon Graphics founders).

Netscape Communications Inc. was purchased by AOL (America On Line) in 1999.

Figure 3.8 is an example of what `netscape` looks like.

Browsers, ASCII-based or graphical in nature, offer much of the same functionalities.

Some of the functionalities include

- starting a new browser window.
- send/receive mail.
- access web servers.
- access news servers.
- save documents as HTML, text or PS format.
- print documents.

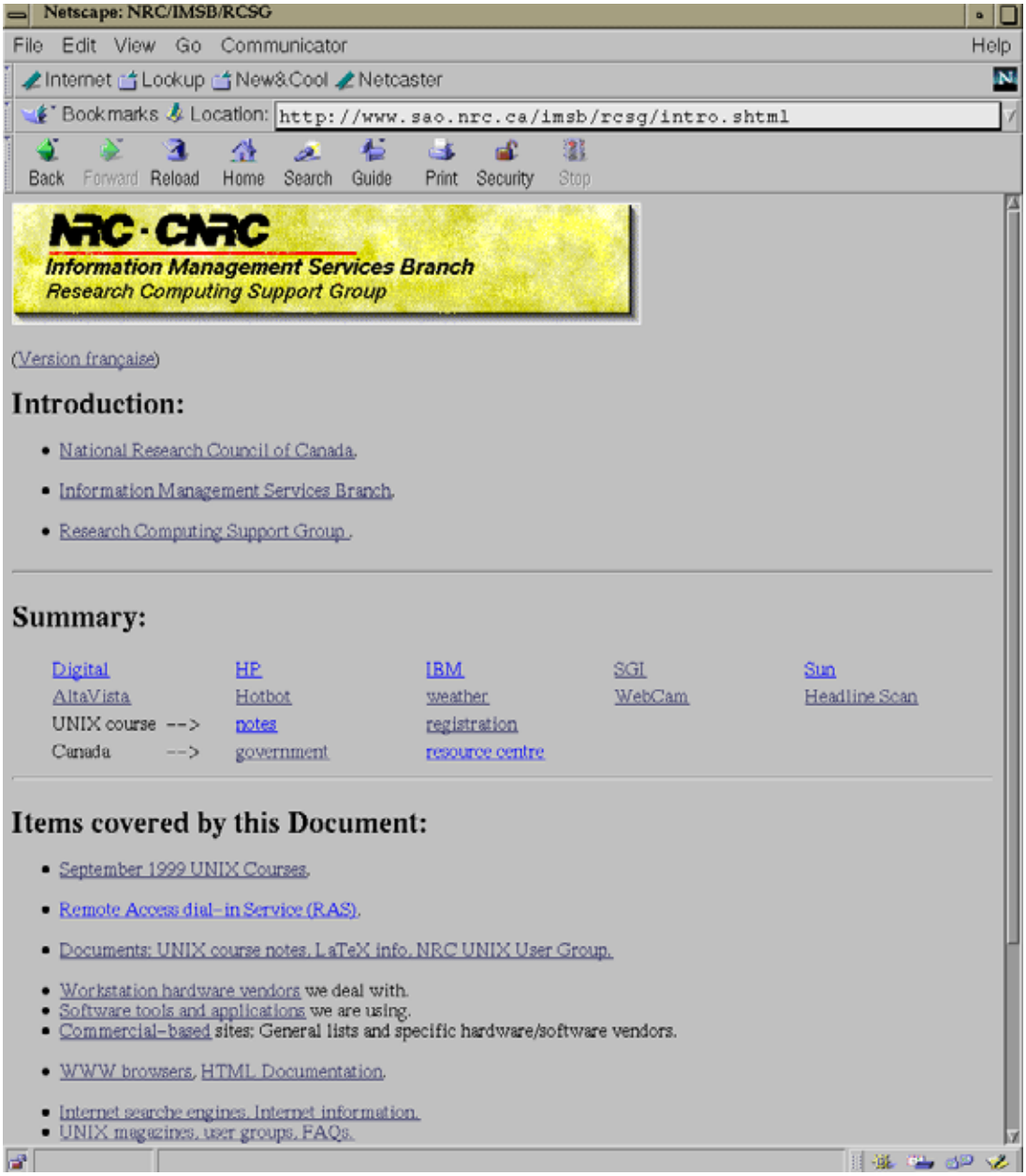


Figure 3.8: RPSO/Research Computing Support Group home page using `netscape`

- find a strings within documents.
- web searches.
- look at the HTML source for the current document.
- navigate through recently accessed documents.
- tag (bookmark) specific documents; manage that list of tags.

3.10.3 URL: Uniform Resource Locator

The action done on the `hypertext link` depends on both the type of `link` it is (the `URL`), and the file name (its extension) itself.

The general syntax of a `URL` is

URL-type`://`*system*`/`*path*

where

- *URL-type* is defined in table 3.10.
- *system* is the name of the system on the network (may also include additional information such as account name, password, and/or port number of server)
- *path* is the path name of the document or directory to be accessed.

An example would be

`http://www.nrc.ca/imsb/rcsg/unix_course`

3.10.4 Miscellaneous Comments on WWW browsers

Here are a number of comments on the use of browsers, but also directed towards people writting HTML documents.

- People don't realize how much information is transferred using these tools. An image file may easily be 1 MB in size. Over a slow or busy link, it may take a long time to transfer.

  Also, each icon has to be downloaded individually; that also adds to the download time.

  In writing the document, use small images; if the user wants to see a more detailed version of it, allow the user to click on it for a high resolution display.

URL type	Description
ftp	allows connection to anonymous FTP servers
http	accesses web server with HTML documents
gopher	accesses gopher servers
telnet	opens telnet session
file	opens local HTML file
mailto	opens builtin mail tool to send mail
news	connects to news server

Table 3.10: URL types and descriptions.

- When documents are read with `lynx` or `netscape` with graphics turned off, information normally given by the images is omitted. That information should be made available using substitute text (HTML has the constructs to do that; unfortunately too many people don't use them in created web pages).

  In `netscape`, unselect `Auto Load Images` (set on by default) in the `options` menu. It will reduce the network bandwidth and the time required to load the documents.

- Large (or many) pictures use up a lot of bandwidth. As a result, it takes a long time to display the document.

  A good balance between graphics and text should be maintained.

- There are converters available to translate from some file formats (such as `WordPerfect` or LaTeX documents) to HTML. Some are better than others.

  In fact, these course notes are available on

  ```
  http://www.nrc.ca/imsb/rcsg/documents
  ```

  in HTML format (translated from the LaTeX source using `latex2html`).

- If possible, always test new documents with several systems and browsers (on PC, MAC, UNIX colour terminal, UNIX b&w terminal, X terminal; using `xmosaic`, `netscape`, `lynx`) to ensure that it is readable from all.

  If a document looks shabby, it may be because it was written on a different platform than you are using to read it.

- Try to access a default local home page. The `home page` is the startup document.

  Most of the SGI Science Institute servers in Ottawa access the IMSB/RCSG home page (see Figure 3.8).

  Any user may write his/her own home page and access it using

  netscape *file.html*

  where *file.html* is local, and written in `HTML`.

- If the server you are using is running an `http server`, you can make your own documents available to anyone with a browser by putting your document in directory

  `$HOME/public_html/`*file.html*

  where *file.html* is the document you would like others to access.

  People could then access your file by using URL

  http://*server_name*/~*login*/*file.html*

  where *server_name* is the name of the UNIX workstation or server running the `http` server, *login* is your `logon` account, and *file.html* is the document opene to the public.

  Some institutes are restricting the use of the `public_html` directory to people within NRC only. Outside users have to access to it.

- When writing documents, make use of the directory structure; every document should probably have its own directory along with the images it needs.

- When writing documents, make use of `relative` path names as much as possible. Relying on `absolute` file names is asking for disaster.

  One never knows when the top level directory will be moved somewhere else!

- Writers of `HTML` documents: plan ahead. Know what the general look of the documents will be; how it will be presented.

  Planning ahead will also incorporate ideas/concepts addressed during the few points above. It will also make it much easier for the next person taking over to understand not only the relationships between the documents, but also *where* the source for all documents is (including images, etc.).

- There are many HTML editors available, both in the public and commercial domains. As with other software, you often get what you pay for. You may also find very good public domain editors, both for UNIX and DOS.

  Popular work processing packages also often offer HTML output for documents.

- Again, for developers: ensure each and every directory has an `index.html` file. That way, users accessing the `http` server will never see the layout of your directory structure.

  It also increases the security of the system by not allowing them to see how the information is stored.

3.11 Navigating the Web

Finding information on the Web is like looking for a needle in a hay stack. Fortunately, there are starting points one can use. Tools are also available to help perform searches for specific data.

The following two sections address these two points.

3.11.1 Where to Start

More and more technical journals now publish `URL`s. More and more organisations have a home page. Fortunately, the format of the addresses is relatively consistent: more than often, the name looks like

`www.`*organisation*`.`*org_type*
or
`www.`*organisation*`.`*country_code*

For example

```
www.nrc.ca
```

represents NRC's own home page.

Most high technology organisations have a home page.

A few examples include `www.sgi.com`, `www.nt.com`, `www.ams.org`.

But where does one start when one wants to explore what is out there? The easiest way is to use someone else's work (we all do for driving; that's what maps are).

That "someone else's work" is a list of lists (also known as `portals`), or a home page filled with interesting starting points. That's probably all you need to get started!

One of the more popular portal is

`http://www.yahoo.com`
or
`http://www.yahoo.ca`

Governments are increasingly creating portals for their own services.

`http://www.canada.gc.ca`

is an entry point for the federal government services.

3.11.2 Search Engines

Portals also include search engines. They allow for searches (either within the portal, or on the web in general) of specific items.

Search engines access databases constructed "and updated by programs called 'robots' or 'spiders' that continually 'travel' the Net finding documents. As they encounter documents, they record information about each document, which is then used to update the database of documents." [2, p. 70].

The databases are built over time, as the "robots" and "spider" travel the Internet. Some date the entries and "drop them" after a period of time (not all do!). As a result some of the information gathered may be out of date.

But this is not a reason not to use theses services.

The complexity of the search engine, and of the interface, varies from tool to tool. But the output from the searches, in all cases, results in a list of `URLs` that match the search string.

There are a number of good search engines. Some of them include

`http://www.google.com`
`http://www.altavista.com`
`http://www.yahoo.com`
`http://www.hotbot.com`

3.12 Exercises

1. What is the identifying name of the UNIX system you are using?
2. You have an account on a remote machine. How can you access it? Do it (if you don't have an account on a remote machine, assume the remote machine is the one you are now using).
3. On that same remote system, you have a file you wish to transfer to yourself. How would you do it?

Chapter 4

Special Files

When a user logs on to UNIX, a series of configuration files are executed. The order in which the files are executed differs slightly between workstations of different vendors.

For people using `bash` in the SuSE linux evironment, various system and personal files are executed. The content of the files are executed in the following order:

`/etc/profile`
`/etc/profile.local` (if it exists)
`/etc/localenv` (NRC specific)
`/etc/bash.bashrc`
`/etc/bash.bashrc.local` (if it exists)
`.bashrc`
`.alias`
`.profile`

For the C and Tenex shell in IRIX, `/etc/cshrc` and `/etc/.login` are first executed, followed `$HOME/.cshrc`, followed by `$HOME/.login`.

Many utilities, such as mail programs and newsreaders also look at special files at startup. This chapter examines a few of these files.

4.1 `.login` and `.cshrc`: C Shell Login Files

Both those files are executed upon logging into the system. But `.cshrc` is the only one executed when a C Shell script is executed. For that reason, the contents of both files may differ.

4.1.1 Environment vs Local Variables

Environment variables are defined by the user, either interactively or in a startup file such as `.login` or `.profile`. They are typically used by specific applications. Environment variables are automatically transferred to a new shell when one is created.

Examples of such variables include `HOME`, `MANPATH`, `TERM`, `PRINTER` and `DISPLAY`.

In general, `environment variables` are defined in upper case characters.

Local variables, on the other hand do not get automatically transferred to a new shell when it is created, because they are used by the shell itself, not the applications. That is why those variables need to be initialised every time a shell is created.

Examples of C shell *local variables* include `prompt`, `history` and `savehist`. All new variables in `bash` are *local* unless they are *exported*, at which point they become *environment variables* (more details in the *.profile* section.

In general, `local variables` are defined in lower case.

4.2 `.profile`: Bourne, Korn and Bash Shell Autoexec

If the user logs on through Bash, Bourne or the Korn Shell, the `.profile` file in his/her home directory is automatically executed.

That file contains commands to be executed upon logging in. Such commands are the setup of the `PATH` variable, and other environment variables (keyboard type, the prompt, etc.).

Some users like to know who is logged on when they get into UNIX. `who` could be added at the end of the file.

By default, all new variables are considered *local*, which means that if a new shell is invoked, the current *local* variable values will not be carried over to the new shell.

To get the *local* variable to become an *environment* variable, it needs to be *exported*. This is done with the `export` command, in one of two ways:

```
PRINTER="myprinter"
export PRINTER
```

or

```
export PRINTER="myprinter"
```

If the `export` command is not used, the value of the variable will stay *local*, which means than even if a shell script is run, the value of that variable will not be transfered to within that shell script. It will stay *local* to the shell.

But if the `export` command is used (in one of the two ways shown above), then the value of `PRINTER` will be transfered to any shell invoked, or to within shell script that are run from the current shell.

The command

```
env
```

will list the currently defined variables.

Not only are *local* and *environment* variables defined in `.profile`, any commands required to run at startup can be put in this file. Often, `aliases` are included (they may also be included in the *.alias* file.

4.3 .bashrc: Bash Shell Startup

This file is specific to the `bash` shell. It is executed every time a new bash shell is created.

Both environment and local variables would be defined in that file, the exact same way they would be idefined in `.profile`.

4.4 .login

`.login` is executed at login time by the C and Tenex shells. Environment variables would be defined in this file.

A few examples include

```
# set default directory search for help man pages.  setenv MANPATH
/usr/local/catman:/usr/catman
# set default BSD printer (lpr command)
setenv PRINTER myprinter
# set default System V printer (lp command)
setenv LPDEST myprinter
# set my editor of choice
setenv EDITOR my_editor
```

4.5 .alias

In SuSE linux, `.alias` is executed automatically at login time, before `.profile` and `.bashrc`. Any aliases defined in that file become permanent as they are automatically redefined at each

login, and everytime a new shell is invoked.

4.6 .cshrc

This file is executed every time a new C (or Tenex) shell is started. The contents of the file should not generate any output; it should simply define local variables, or aliases.

A few example of statements found in such a file would be

```
# define some aliases. alias ls "ls -C"
alias bye logout
# add to the path definition.
set path = ($path $HOME/myprogs)
# redefine my prompt; the %! represent the history number
set prompt = "%! my_prompt> "
# set the number of commands remembered by current shell.
set history = 40
# set the length of history remembered between sessions.
set savehist = 40
```

4.7 .bash_logout and .logout: Leaving linux/UNIX

When logging out of a linux system running the BASH shell, file `.bash_logout` is executed. This file is usually very short, and contains commands users want to execute upon leaving the account. The most popular is the `clear` command, which clears the screen.

Other shells typically use `.logout` instead of `.bash_logout`.

4.8 .bash_history: Content of History in bash

`.bash_history` file contains the last commands used with the `bash` shell. The size of the file depends on what size `HISTFILESIZE` is set to in the startup file.

The file is usually updated at logout time.

4.9 .history: Content of History

The `.history` file contains the last *n* commands used. The size of the file depends on what size `history` is set to in the startup file.

The file is usually updated at logout time.

4.10 .mailrc: BSD Mail Customisation File

This file contains instructions that affect how the BSD `mail` command works. First, `/usr/lib/Mail.rc` is executed and then, if it exists, `$HOME/.mailrc` is executed.

Here are some of the entries found in a typical `.mailrc` or `/usr/lib/Mail.rc` file:

- `alias` *nick_name real_address*: this allows the user to use *nick_name* instead of *real_address* to send mail.
- `set ask`: asks the user for a `Subject` when sending mail.
- `set append`: appends new mail files at the end of folders, including `mbox`.
- `set askcc`: enables the `Cc:` prompt when about to send the message.
- `set folder`=*dir*: folders will all be within directory *dir*. Usually this directory is `mail`.
- `set record`=*pathname*: all outgoing messages are saved into this folder.
- `set VISUAL=/usr/bin/vi`: name of the editor to use when ~v is used at composition time to select edit mode. This example uses `/usr/bin/vi`. It could use any available editor.

Consult your system manuals for more examples.

4.11 .elm/elmrc

This file is the equivalent of `.mailrc` but for the elm electronic mail system. Modifying that file with an editor allows a whole range of customisation, except for the setting up of aliases. Aliases are set up within `elm` itself.

If the file does not exist, invoke `elm` and call the options menu (press "`o`"). Make a change, and save it. The `elmrc` file will be created.

The file is self-explanatory. Samples taken from `elmrc` include:

```
# what editor to use ("none" means simulate Berkeley Mail)
editor = vi
# The full user name for outbound mail
fullname = Claude P. Cantin

# local ".signature" file to append to appropriate messages...
localsignature = .lsignature

# remote ".signature" file to append to appropriate messages...
remotesignature = .rsignature

# automatically copy message being replied to into buffer?
autocopy = ON
# save a copy of all outbound messages?
copy = ON
```

4.12 .pinerc

This is the `pine` mail interface configuration file.

A sample (also the default) may be found at

```
/usr/local/lib/pine.conf
```

4.13 .newsrc

When a Network News newsreader is invoked, it first looks at that file to find out which newsgroups have been subscribed to, and which articles have not been read.

The general format of the file, for every record, is:

newsgroup name
a colon (:) or an exclamation mark (!)
numbers

If the second field is a colon, this newgroup is subscribed to. Alternatively, an exclamation mark means that newsgroup is not subscribed to.

The numbers after that are used by the newsreader as information as to which articles have been read (the newsserver keeps track of the articles using an index number; there is

a direct relationship between the index numbers of the articles and the numbers seen in the `.newsrc` file).

4.14 .tin/tinrc

`tinrc` contains commands for the `tin` newsreader. That file is read when `tin` is started.

The commands in that file are used to configure the news reader. It may also be used to tell `tin` which command to use when sending an article to be printed, or which editor to invoke when composing an article to be posted.

4.15 calendar

The `calendar` file—usually in the `home` directory—is used as a reminder file. Each entry in the file is one line long, and starts with a date in the form `May 21` or `5/21` followed by some message. An asterisk (∗) represents all months (in the case `* 21`, every 21st), or every day (`May *`).

When the program `calendar` is started, UNIX looks into the current directory for a file called `calendar`, and reads it. If any of the dates represent the current date, or the day following (excluding weekends), it prints the message line on the console.

It is common to see the command

calendar

in the `.login` or `.profile` file.

Information from this file may be referenced by the `pcal` command for possible inclusion to its output.

4.16 Exercises

1. When logging into your system, you want a greeting message. How would you do it?
2. Change your system prompt to say the history number, followed by your name, followed by a comma, followed by the directory you are currently using, followed by a question mark. It should look similar to `25 cantin, /home/nrccsb2/cantin` ?.
3. In your `.login` file, write code that would display how many users are logged onto the system, who they are, and when they logged on.

4. Add an `alias` to your `.cshrc` file that would interpret the `lls` command as `ls -l`.

5. Add an `alias` to your `.cshrc` file that would interpret `hl` as going to your `home` directory, and performing a listing.

6. Increase your history to 100 items, but save only the last 50 when you log out.

7. Upon logging out of UNIX, you want a message to be sent to a file called `.logout.time` in your home directory. That logout message should be similar to `cantin logged out on Wed Aug 22 at 17:22 1990.` (*NOTE*: the seconds and the timezone are NOT included).

8. Set up a file called `calendar` in your home directory with fake appointements you could have. Set it up so that it gets executed every time you log into your system.

Chapter 5

Compilers

For UNIX vendor workstation (SGI, SUN, IBM, HP) compilers are usually optional software packages. They are not included with the Operating System.

But linux systems come with the set of GNU compilers: C and C++ (`gcc`) and Fortran (`g77`) are the major ones.

This chapter briefly describes how to compile C and FORTRAN programs on most workstations. Other systems are similar, but differences do occur.

On most RISC-based workstations, such as the Sun SPARCstation series, the Silicon Graphics 4D series, the Digital Equipment AXP series and the IBM RS/6000 family, there is no floating point option for the compilers, since all workstations include a floating point unit.

5.1 C Compiler

5.1.1 `cc`/`gcc`: Invoking the C Compiler

The compiler is invoked with the `cc` command, for commercial C compilers, and with `gcc` for the freely available GNU compiler.

Basic flags are the same.

Commercial UNIX systems require the purchasing of a C Compiler (unless `gcc` is installed by the owner), while linux does come with `gcc`.

Hence, to compile `prog.c`, one uses:

gcc prog.c

A new program, called `a.out`, will appear in the directory: `a.out` is the compiled program.

In general, the command is

gcc [**-o** *out_file*] [**-O**[*level*]] [**-g**] [**-c**] *source_file* [**-l***lib_name*...]

On the IBM RS/6000, the C compiler is invoked using `xlc` instead of `cc`.

5.2 FORTRAN Compiler

FORTRAN compilers usually do not come bundled with the workstation. They need to be purchased separately, from either the workstation vendor, or from a third party.

5.2.1 `f77/g77`: Invoking FORTRAN

The FORTRAN compiler is usually called `f77`. On the IBM RS/6000, it is invoked by issuing `xlf`. Some linux distributions include GNU's `g77`.

In its simplest command form, program `prog.f` can be compiled by:

g77 prog.f

and the resulting `a.out` file will be its executable counterpart.

In general, the command is

g77 [**-o** *out_file*] [**-O**[*level*]] [**-g**] [**-c**] *source_file* [**-l***lib_name*...]

5.3 Common Flags

Most of the compiler flags are common to a number of compilers. The list of flags shown here is common to most C and Fortran compilers.

There are many options on all compilers. Be sure to read the man page for both the compiler (`f77`, `cc` or the equivalent) and the link editor, commonly called `linker` (`ld`).

5.3.1 `-o` *out_file*: Output File

This option produces an executable file called *output_file*. To produce an executable called `prog` from the above example,

gcc -o prog prog.c

5.3.2 -O[*level*]: Optimisation

The **-O**[*level*] flag indicates the optimisation level of the compiled program. A level of 1 gives the least optimisation, whereas a level of 3 (some compilers go higher; others, such as the SGI IRIX 5.2 version only goes as high as 2) gives the most optimisation.

NOTES:

- Certain types of program, such as device drivers, should never be compiled with an optimisation of more than level 2.
- Optimisation re-arranges the original source code. It is strongly recommended the program be first compiled with NO optimisation, then run with test files. The program could then be compiled with higher optimisation, and the results compared.

  This would ensure that the compiler did not introduce errors as a result of re-arranging the code.

On some systems **-O** is equivalent to **-O1**; on others, it is equivalent to **-O2**. Check the local compiler man page.

5.3.3 `-g`: Debugger Flag

When this flag is used, extra code related to the debugger is added. Do not use with **-O**.

5.3.4 `-c`: Suppress Linking

This flag suppresses linking. *object_file*`.o` will then be produced instead of an executable output file. This object file could then be linked with other routines/packages.

5.3.5 `-l`: Link to Library

In many instances, a program needs to be linked to a specific library, or to many libraries. A standard library is represented by a file of the form `/usr/lib/lib`*name*`.a` where

- `lib`*name*`.a` is the full library name.
- *name* is the name of the library included on the command line with the `-l` flag.

With `shared library` systems, like the one with SGI IRIX 5.2, the libraries are also represented by the library name appended with `so`, for *shared object*.

For example, to access the math library (file `/usr/lib/libm.a`, or `/usr/lib/libm.so`), the flag

-lm

would be used at the end of the command line (for either library file). The SGI Graphics Library (file `/usr/lib/libgl_s.a`) would be accessed with

-lgl_s

When many libraries are needed, a separate flag is used for each of the libraries. If both the math and the gl library need to be accessed,

-lgl_s -lm

would be added at the end of the command line.

5.4 make

`make` is a UNIX utility designed to assist the programmer in maintaining different packages. When invoked, `make` automatically looks at a file called `makefile`, `Makefile`, or `MAKEFILE`, which contains information concerning the dependencies of all files in the package.

Consider a package composed of ten or fifteen C or FORTRAN files. If only one source file is changed, EVERY source file need not be recompiled. If a *makefile* is used, rebuilding the entire package may mean recompiling only one or two source files (only the necessary ones) and relinking.

The following example is a `makefile` to compile a package, called `mypack`. `mypack` is built using four routines: `mypack.f`, `getfiles.f`, `crunch.f`, and `dotables.f`.

```
OBJS=   mypack.o    getfiles.o  \
        crunch.o  dotables.o
FFLAGS=    -O2
all:    mypack
mypack:   $(OBJS)
        f77 -o mypack $(FFLAGS) $(OBJS)
mypack.o: mypack.f
        f77 -O mypack.f
```

```
clean:
        rm *.o mypack
```

NOTE: the makefile is VERY sensitive to spaces. TABs must be used to indent. There shall not be trailing blanks.

The first two lines of the `makefile` assign to a variable called `OBJS` the names of all object files. The backslash is a line continuation character. The second and any subsequent continuation line MUST start with a tab character.

The third line defines `FFLAGS` to be **-O2**, which is the optimisation flag that will be used when invoking the compiler.

The fourth line starts with the `all` label. By default, `make` will look at that label, and see that it depends on `mypack`. It will then go to the `mypack` label, and see that it depends on the four object files (`$OBJS`). But to create those four object files, the command used should be `g77 -o mypack $(FFLAGS) $(OBJS)` (line 6).

BUT the object files have not yet been created. So, by default `make` will use `f77 -O2 -c getfiles.f` to create `getfiles.o`, and similarly for `crunch.o` and `dotables.o`, BUT will use `g77 -O -c mypack.f` to produce `mypack.o` (from line eight).

For all routines to be recompiled, two commands would need to be issued:

make clean
make

The above example is a typical short `makefile`. If the program to be compiled does not require special flags and is only one file,

make *filename*

would compile and link it (note that there are NO extensions to the file). The result would be put into file `filename`.

Public domain packages usually come in source code only. Those packages usually contain a `makefile`. It is often sufficient to type **make** to compile the entire package.

More about `make` can be found in [3, p. 139-145], [7, p. 478-480], and your system manuals.

5.5 Tools

This section will list a few tools useful when developing packages. Most of them are only available on the SGI IRIX platform.

Its only aim is to mention the tools. For more information on them, look at their respective man pages.

5.5.1 `fsplit`: Split Fortran Program

`fsplit` *prog.f*

will scan *prog.f* and create a separate file for each subroutine found in the file.

5.5.2 `pixie`: Add Profiling Code to Program

`pixie` *prog*

By running the executable file *prog* through `pixie` two new files (namely *prog*`.pixie` and *prog*`.Addrs`) will be produced. These new files contain the code of the original program in addition to new code pertaining to the execution of every program block/unit.

Running

prog`.pixie`

will produce a new file: *prog*`.Counts`.

`prof` is then used to analyze that data.

5.5.3 `prof`: Profile Analyzer

There are two ways to use `prof`:

`cc -p -o` *myprog myprog.c*
myprog (generates `mon.out`)
`prof` *myprog* `mon.out`

OR

`cc -o` *myprog myprog.c*
`pixie -o` *myprog*`.pixie` *myprog* (generates *myprog*`.Addrs`)
`myprog.pixie` (generates *myprog*`.Counts`)
`prof -pixie` *myprog* (reads *myprog*`.Addrs` *myprog*`.Counts`)

In both cases, the results are sent to Standard Output.

5.5.4 `cb`: C Beautifier

Causes the source code program to be indented in a consistent manner.

5.5.5 `lint`: C Program Syntax Checker

`lint` *prog.c*
OR
`cc -wlint` *prog.c* (on IRIX 5.x)

will scrutinize `prog.c` for syntax errors and wasteful constructs. `lint` is very picky about how a C program is written. Among other things, it ensures that

- Any variable created is used.
- Every program has an exit point.
- Any declared function is used.

5.6 Exercises

1. Write a short Fortran or C program to print `Hi!` on the display.
2. Compile that program with optimisation level 2. The executable should be called `hello`.
3. Why do most `makefiles` begin with an upper case `M` ("Makefile")?
4. Remove hello. Write a simple `Makefile` that could be used to compile your program, but also used to remove the executable. Try running the `Makefile` to compile the program, test it, then run the `Makefile` to remove the executable.
5. Use `make` without any `Makefile` to compile the program.

Chapter 6

UNIX Batch Systems

The UNIX operating system has a number of ways to process a number of programs for the same user, simultaneously. Programs can be run in the `background`, they can be run by `cron`, they can be run at a specific time using `at`, they can be run in the background without interruption on logout, or they can simply be run in the foreground, as an interactive job.

This chapter describes ways of running programs in some kind of `batch` process.

6.1 `&`: Background

Programs can be sent to the `background` to be processed, while the user keeps doing interactive work in the `foreground`. Sending a job to the background is done by ending the command line with an ampersand (&).

When a job is sent to the `background`, the prompt immediately reappears, allowing the user to continue work in the foreground. At completion of the background job, a message will be displayed, announcing termination of the process.

NOTE: before leaving UNIX, all jobs running in the background mode should be allowed to complete; if the terminal used was connected to a serial port of the UNIX system and background jobs were still running, attempting to leave UNIX would potentially hang the port and stop other users from using it!!

If jobs are to be allowed to continue running upon exiting the interactive UNIX session, then the `batch` command should have been used instead of the `&` function.

6.2 `nohup`: No Interruption

nohup *command* [> *outfile*] **&**

Even when leaving UNIX, *command* will keep running. It is recommended that output redirection be used in order to capture the output.

When using `nohup`, it is necessary to put the job in the `background` using &.

6.3 `at`: Execute at Specific Time

The `at` command allows the user to specify exactly when a specific job is to be run. The output of `at` is sent by mail to the user who initiated the job.

The usage is

at *time* [*date*] [+ *increment*] [< *script*]

where *script* is a C shell or a Bourne shell script.

If *script* is not given as a parameter, `at` will prompt the user for the name(s) of the script(s) to be run.

time can be given as 1, 2, or 4 digits. A 1- or 2-digit time would indicate the hour for *script* to be run (using a 24 hour clock). A 4-digit time indicates *hour*:*minutes* (even if the colon is not present), again on a 24-hour clock. `pm` or `am` may be appended to use a 12-hour clock. `noon` and `midnight` are recognized as such.

If omitted, *date* is considered to be today. If it is used, it is read as a month name, followed by a day number, or name. The name may be fully spelled out or abbreviated to three characters.

The two literals `today` and `tomorrow` are accepted for days.

If used, *increment* is a number, followed by one of the keywords `minutes`, `hours`, `days`, `weeks`, `months`, or `years` (singular form of the word is also accepted). The script will be executed that interval of time following *date*..

The following examples illustrate the use of `at`:

at 4 < *script*
at 17:00 < *script*
at + **4 days** < *script*
at 3 pm < *script*
at 3:00 september 24 < *script*

In each case, the system will reply with a job number.

6.3.1 -l: List Jobs

at -l [*job...*]

where *job...* is a job number(s).

This will print a list of jobs in the queue. If the *job...* option is not used, all jobs queued to `at` for that user will be displayed:

`prompt>` **at -l**

```
3079 a    Sat Jan 13 16:43:00 1990
```

6.3.2 at -r (atrm on linux): Remove Jobs

at -r *job...* (UNIX)
or
atrm *job...* (linux)

where *job...* is the job number of job(s) to be removed from the `at` queue.

Using `at` this way will remove *job...* from the queue.

`prompt>` **at -r 3079**
or
`prompt>` **atrm 3079**

removes the queued job number 3079.

6.4 batch: Batch Processor

batch `[-csm]` [< *script*]

where *script* is a C shell or a Bourne shell script.

-c (C shell): will run *script* as a C shell script.

-s (standard shell): will run *script* as a Bourne shell script (default).

-m (mail): will send mail to the user, even on successful termination.

If *script* is not given as a parameter, `batch` will prompt the user for the name(s) of the script(s) to be run.

Jobs queued by `batch` will be executed as soon as the system load level permits.

As with `at`, the output of `batch` jobs will be mailed to the user.

`batch` and `at now` are equivalent.

6.5 `NQS`: Network Queueing System

Some of the Science Silicon Graphics servers have the `at` and `batch` commands disabled. Instead, they must use the `qsub` command.

On those systems, issue

```
help nqs
or
xhelp nqs
```

to find out how to use NQS, how to submit jobs, how to keep track of running jobs, which queues are available, and how to remove jobs from the different queues.

6.6 `cron`: Job Scheduler

The `crontab` utility is mostly used to run jobs at the same time every day, every month, or every year. Every minute, the system checks a system file, called `crontab`, to see which (if any) processes/jobs need to be started.

There are four ways to manipulate the `crontab` file:

crontab [*filename*]
crontab -l [*username*]
crontab -r [*username*]

NOTE: Be careful in using the first method: if a `crontab` file already exists, it will be overwritten!! (To avoid the file being overwritten, escape using <CTL-c>, NOT <CTL-d>).

Every line in the `crontab` file contains six fields, each separated by a space. A field may contain more than one value, each separated by a comma. An $*$ means "all values".

The fields are:

1. `minute`: values are 0 to 59.

2. `hour`: legal values are 0 to 23.

3. `date of the month`: 1 to 31.

4. `month`: 1 to 12 (Jan - Dec).

5. `day of the week`: 0 to 6 (Sun - Sat).

6. `Bourne shell command`.

Here is an example of what a `crontab` file would look like:

```
0 0 * * * /usr/bin/new.messages
0,15,30,45 * * * * /usr2/people/cantin/check.mail
0 0 1 1 * /usr2/people/cantin/new.year.message
```

The first line on the above example runs the command `new.messages` at 0 minutes, 0 hours (midnight), every day, every month (i.e., every day at midnight).

The second line runs `/usr2/people/cantin/check.mail` every fifteen minutes.

Finally, the third entry runs `new.year.message` the first of January, at midnight.

6.6.1 `crontab` [*filename*]

If *filename* is used, the contents of *filename* will replace the old schedule.

If no *filename* is typed, the contents of the *new* file will be accepted through the standard input (usually the keyboard). *NOTE: This will overwrite any previous contents of crontab.*

6.6.2 `crontab -l` [*username*]: List

List the job schedule for a user (or yourself if *username* is not used).

6.6.3 `crontab -r` [*username*]: Remove

This option removes the job schedule for *username*.

6.7 Exercises

1. You are logged in with a VT100 ASCII terminal, and you have a job to run that you KNOW will take ten minutes to execute. How can you run this job at the same priority as interactive jobs, but still read and respond to your mail?

2. Using that same job, I want you to run it at 3:00 AM. How will you do that?

3. Use the same job again. Suppose you want to run the job during the day, but you don't want to interfere with other users' interactive work. How could you do it? What is the preferred way?

4. Suppose you have a series of ten jobs you would like to run. They all produce output. You are also interested in finding out how much CPU time and system time those jobs took. You do not want to affect interactive users when those jobs are running. How will you do it?

5. You have a program that needs to be run at 2:30 AM every Wednesday. How can you do that?

6. Send two jobs to be run tomorrow afternoon, at 1:00 PM and 3:00 PM respectively. Suppose you now change your mind: you now want to run only ONE job at 2:00 PM tomorrow. Take the necessary steps to meet that goal.

Chapter 7

X-based Graphical User Interfaces

A Graphical User Interface, GUI, is synonymous with a window system. Examples of GUIs are Microsoft Windows for DOS, OpenWindows from Sun, IndigoMagic from Silicon Graphics and VUE from Hewlett Packard.

These GUIs usually run on top of UNIX, providing the user with tools to use UNIX to its full potential. These tools are easy to use, and are usually driven by a mouse, a "point-and-click" device.

The UNIX world has two GUIs emerging as standards: Open Look and Motif. Both of these run on top of the X Window System.

7.1 The X Window System

The X Window System is a set of guidelines and libraries developed by the X Window Consortium from the Massachusetts Institute of Technology, during the mid to late 80's. This package, now a standard, is in the public domain, and the most commonly used version is now at Version 11, Release 6 (June 1996).

This set of libraries dictates how a window should be opened, what its characteristics may be, and how it will be displayed on the screen. But it does not say what the window will LOOK like.

The X Window System also includes a set of `widgets`, or basic structures to construct different types of windows. It also contains a library of functions to help build windows and define what each window will do, how it will react to different stimuli, etc.

Over all, the system is based on the `notifier` concept: a set of functions are performed (window created, filled with text, or buttons, or slides, etc.); the Window System is then gov-

erned by the `notifier`, or the master. The `notifier` invokes different functions, according to what mouse button was pushed, or what was typed on the keyboard, etc.

The power behind the X Window System resides in the fact that an application running on system A may be displayed on system B (the two systems may also be of different architectures).

Figure 7.1 is an example of a Motif-based GUI. It is of Silicon Graphics' IndigoMagic, running IRIX 5.2 on an Indy workstation.

X is based on the client-server approach: the client runs the application while the server displays it on the local display (note that X uses a different definition for "client" and "server"; the system running the application is termed the "client" while the system displaying the application is the "server").

Most utilities based on the X window system are very easy to customize. They can usually be resized, moved, put on top of other windows, and pushed behind other windows interactively. Their appearance can also be altered significantly with the help of *resources*, through the use of the `.Xdefaults` file, described in one of the sections below.

Some utilities/programs based on the X Window system include a terminal emulator (`xterm`), window managers (`4Dwm`, `twm`), a news reader (`xrn`), access control programs (`xhost`, `xauth`), a load monitor (`xload`), a clock (`xclock`), a calculator (`xcalc`), utilities for listing information about fonts, windows, and displays (`xlsfonts`, `xfontsel`, `xlswins`, `xlsclients`, `xdpinfo`, `xprop`), screen image manipulation utilities (`xwd`, `xwud`, `xpr`, `xmag`), 2-D graphics programs (`xmgr`, `gnuplot`), a public domain retrieval system (`xarchie`), a PostScript file previewer (`ghostview`), and many, many more utilities, many of which are in the public domain.

7.1.1 X Windows and X Terminals

An X Terminal is a "dumb terminal" running the X Window System. The X Terminal must have a bitmap display. A bitmap display is a display which is made out of "pixels", or dots. A typical UNIX workstation has a display screen of approximately 1,000 by 1,000 pixels (SGI has 1280 x 1024 on most, SUN has 1150 x 980).

The bitmap display allows a large amount of detail to be shown. It permits a user to logon to other machines running X applications and hence use the terminal as a virtual console of the remote system.

An alternative to X Windows is a PC running a TCP/IP package, X Windows software, and equipped with an ethernet card. Using the TCP/IP software, the PC is given an IP address, and is registered within the name server as an X Terminal. From that point on, the PC is used the same way as an X Terminal is used.

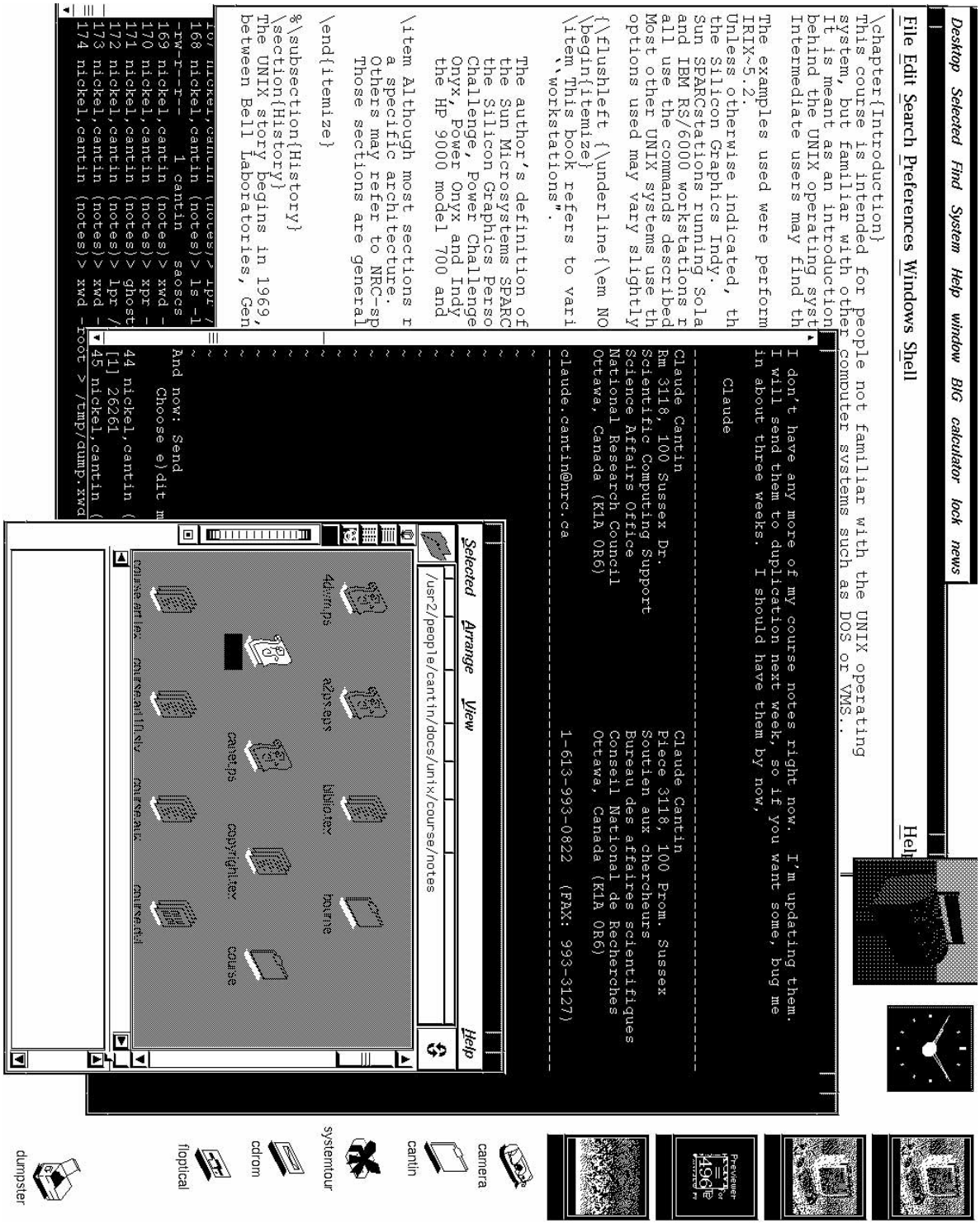


Figure 7.1: Example of a Motif-based Graphical User Interface

7.1.2 Window Managers

A `window manager` is a program which manages the different utilities used within the X window system. It allows the interactive creation of other utilities and the moving of windows, to name just two of its functions.

In short, the window manager manages the resources of the local display.

Examples of window managers are `mwm` ("Motif"), `olwm` (Open Look) and `4Dwm` (by Silicon Graphics).

Window managers are started at login time.

7.1.3 Using X

The main advantage of the X Window System is to allow a user to run a job on a remote system, but use the local terminal as the display unit. The local display unit must be able to display X applications.

The "local display unit" may be one of:

- A workstation.
- An X Terminal.
- A PC running X emulation software.

On the remote system, ensure the environment variable `DISPLAY` is defined as

```
setenv DISPLAY local_host:0                    (csh)
DISPLAY=local_host:0; export $DISPLAY          (sh)
```

where *local_host* is the full name of the local display unit.

In some cases, the local display unit must allow the remote host to use it to display the application. This is done by using the `xhost` command on the local system, such as

```
xhost remote_host
```

where *remote_host* is the name of the system where the application is actually running.

7.1.4 `.Xdefaults` file

Upon launching an X application, the user's `.Xdefaults` file is scanned for directives to use when displaying the application. Each line in the `.Xdefaults` file takes the form:

appname`*`*subname*`...: value`

where *appname* is, by convention, the name of the application to which it refers (with the first character in upper case), and *subname* refers to resources or characteristics of the application.

For example,

```
XTerm*geometry:  40x80+0+0
```

in the `.Xdefaults` file would provide an `xterm` terminal emulator window with 40 rows and 80 columns at the upper left corner of the screen.

```
XTerm*foreground:  black
XTerm*background:  white
```

would ensure that the `xterm` terminal emulator would have black characters on a white background.

A utility to help verify the integrity of the `.Xdefaults` file is

`xrdb`: X resource database

7.1.5 `xterm`: a typical utility

Typical flags used when using an X terminal emulator are discussed below. The flags may be used either on the command line of `xterm`, or as an entry in the `.Xdefaults` file. Many such flags may also be used by other X applications.

In the examples shown below, the resource is specified on the command line (the first line) or from the `.Xdefaults` file (the second line).

```
xterm -sb
or
XTerm*scrollbar:  true
```

puts a scrollbar to the left of the window. Text scrolling past the top of the window can be recalled by positioning the mouse cursor on the scrollbar and, while holding down the middle button of the mouse, moving the bar up and down.

`xterm -geometry` *W*`x`*H*`+`*XX*`+`*YY*
or
`XTerm*geometry:` *W*`x`*H*`+`*XX*`+`*YY*

where *W* and *H* are the column and row values of the terminal emulator. Normally they are measured in pixels; terminal emulators are an exception to this rule.

XX and *YY* are the X and Y distance, in pixels, from the upper left corner of the screen to the upper left corner of the `xterm` window.

`xterm -fn` *font*
or
`XTerm*font:` *font*

is the font used for the characters within the window.

`xterm -bg` *colour*
or
`XTerm*background:` *colour*

is the colour of the background pane or panel of the window.

`xterm -fg` *colour*
or
`XTerm*foreground:` *colour*

is the colour used for the characters.

In addition to the above, people running an X Window package on their PC while using the window manager on the UNIX workstation should use the `-ls` option on the `xterm` terminal emulation command. This will ensure their `.login` file is read (if using the `C` or `Tenex` shell).

7.1.6 Fonts

Fonts are by default, found in directories `/usr/lib/X11/fonts/{misc,75dpi,100dpi}`. Commands directly related to fonts are:

- `mkfontdir`: creates font database from newly acquired fonts.

- `xset fp rehash`: re-write font database into font path; generally used immediately after `mkfontdir`.

- `xset fp=`*path*: sets font path to directories in `path`. It overrides any other previously defined font paths.

- `xlsfonts`: lists available fonts.

- `xfd`: X Font Descriptor. It displays all the characters in the font specified.

Font names may be quite long. Wildcards may be used in their names, but should be within quotes because of the special meaning of metacharacters. This is especially true when specifying them on the command line.

7.1.7 Colours

Most X applications recognize common colour names, such as `black`, `white`, `green`, `blue`, `yellow`, etc.

An application helping you choose colours may be available on your system.

`colorview` on the Silicon Graphics is such a tool.

7.1.8 Mouse Functionality

Typically, pointing the right mouse button on the border and clicking will display a menu.

Left mouse click on a border brings that window on top of the others.

Middle mouse click (hold down) on a border allows drag and drop of that window.

Highlighting text by dragging the mouse while holding the left button down will put the highlighted text in a buffer. Clicking on the middle button will copy the content of the buffer onto the application window wherever the cursor is located.

If part of the text is already highlighted, bring the cursor to where you would like the highlight to end, and press the right mouse button (if you are using SGI's `xwsh`, SHIFT-left button will do the same).

A double click on the left button will highlight a word; three clicks a line.

The methods outlined above are VERY convenient when copying and pasting from one window to another, or from within the same window.

7.1.9 xdpr: X Window Dump to Printer

This is a combination of the

`xwd` (X Window dump)
and
`xpr` (X Window dump print)

programs.

It allows a user to dump an X Window directly to a printer.

Typically, it would be used as:

```
xdpr -device ps
```

7.1.10 xdm: X Display Manager

This facility is used to manage X Terminals and graphics consoles. A person with an X terminal may use `xdm` to login directly to a system (typically, one has to open a window on the X terminal, `telnet` to a remote system, login, then start a window manager. `xdm` allows the user to skip all those steps.)

7.2 Exercises

All exercises assume the use of an X-capable display.

1. A friend tells you he is running `Motif` but he does not have any of the `X Window` software. Do you believe him? Why, or why not?
2. You would like to start an `xterm`, 80 columns wide by 24 lines. Which command would you use?
3. Modify your `.Xdefaults` file so that the `xterm` command (no flags) creates a window 80 columns wide by 24 lines.
4. Create an xterm window with black characters and a white background.
5. Which fonts are available to you? Which command can you use to display a list of those fonts?
6. Start an `xclock` 100 x 100 pixels in size, at the top right corner of your display.

Chapter 8

Solutions to Exercises

This section represents possible solutions to the exercises in the course material.

The subsection numbers represent the section covered, and the solution number corresponds to the question number within that section.

Solutions to new chapters of the book will start on a new page.

8.1 Utilities

1. The first thing to do when disk space is getting scarce is to see if you really `NEED` those files. If you don't need all the files, you can simply delete them.

   Or if you don't need them now, but know you will need them in a few weeks, put them on tape.

   Or, you could leave them on disk, but the problem is that some of the files are fairly `LARGE`. Using the `compress` utility will compress those files by 50% or more, thus giving you the disk space you are looking for.

2. The most widely used command to create a backup of your own directory is `tar`. To put that file on tape, say */dev/tape*, issue the following commands:

   ```
   cd
   cd ..
   tar -cvf /dev/tape ./your_logon
   ```

   If the `tarfile` is to be put on disk, replace */dev/tape* by a file name. You could then transfer that file onto another disk, or system (or, it is possible to directly create the

file in your account on a remote system by using `tar -cvf` *remote_system:file_name* `./`*your_logon*, if you have the required permissions).

3. Assume your colleague is running a system similar to yours. To send the binary file to a remote system, a number of methods could be used:

   - You could copy the file onto a tape and send the tape to him/her.
   - If you know his/her password, you could `ftp` the file directly into his/her account.
   - There may be an *anonymous ftp* site you could write to, and he/she could read from.
   - You could use `uuencode` to encode that file into ASCII characters, and send that file over to him via `mail`. Upon receiving the file, `uudecode` could be used to decode it into its original binary form.

4. The file name is `filename.tar.Z`. It ends with `.Z`, therefore is in compressed format. Once the `.Z` is removed, it ends with `.tar`. This suggests it is a tarfile. Here are the steps required to "read" it:

   ```
   uncompress filename.tar.Z
   tar -xvof filename.tar
   ```

   NOTE: it is recommended you do a listing of the tarfile first (`tar -tvf`) to make sure that none of your files will get overwritten.

5. The tape received has three tarfiles on it. The first can be read using

   `tar -xvf` */dev/tape*

   The second one could be read using the `mt` command (magnetic tape) with the no rewind option on the tape device, and the `fsf 1` (forward space one file) option.

   ```
   mt -t /dev/nrtape fsf 1
   tar -xvof /dev/tape
   ```

   In this case, the tape device for most Suns will be `/dev/[n]rst0`; for SGIs, it will most likely be `/dev/[nr]tape`.

   The third file would be read using the same sequence as for the second file, but using `fsf 2` instead of `fsf 1`.

6. The tape was written on an SGI. You own a Sun. `dd` will be needed to read that tape:

   ```
   dd if=/dev/rst8 conv=swab,noerror,sync | tar xvf -
   ```

   NOTE that the tape device may be `/dev/rst0` (for 60 MB tape drives).

   Be aware of tape incompatabilities: a 150 MB drive can read 60 MB tapes, but the 60 MB drives cannot read 150 MB tapes.

7. This is the same as the previous question, but the systems are interchanged.

   SGI systems have a driver to read (and write) tapes directed to Suns. That driver is `/dev/tapens`. The `ns` stands for "no swap".

   So, to read the Sun tape, either of the two following commands could be used.

   ```
   dd if=/dev/tape conv=swab,noerror | tar xvof -
   tar -xvof /dev/tapens
   ```

8. This looks to be a good place for `grep`, the global regular expression parser. First `cd` to the specific directory where the source code is, and issue

   ```
   grep "molecu*" *.c
   ```

9. This problem is misleading (i.e. it looks difficult). However, one simple command will do the job:

   ```
   sort names
   ```

   where `names` is the file of names.

10. The questions assume the `lpr` command will not translate the ASCII file for you.

    What you need is an ASCII to PostScript filter, such as `a2ps` or `lwf`. The following two commands will each print to the printer the ASCII text file

    ```
    a2ps ascii.file | lpr
    lwf ascii.file | lpr
    ```

    but note that `a2ps` will, by default, generate two pages per physical page of output.

11. The file ends in `.ps`. It is a PostScript file, and can be viewed by issuing:

    ```
    ghostscript tiger.ps
    ```

12. To convert `tiger.ras` to `tiger.xbm`,

    ```
    convert tiger.ras tiger.xbm
    ```

    will be sufficient.

13. `xv` allows not only the viewing of files, but also conversions and modifications. The new file can then be saved.

8.2 Text Formatters

1. On Sun workstations, `nroff` is bundled with the software. Some systems also have TEX and LATEX.

   Silicon Graphics systems do NOT come with the Documentor's Workbench (nroff, troff, etc.), but some systems, like the three Power Series in the M60 machine room and in M12, have TEX and LATEX installed.

8.3 Networking/Internet

1. The name of the system you are using is usually found by issuing the command

   ```
   hostname
   ```

2. To logon remotely, `telnet` is used. To logon remotely to `machine.div.com.ca`,

   ```
   telnet machine.div.com.ca
   ```

   is used.

3. You have a file on `machine.div.com.ca` and want to transfer it to the system you are now using. That file is in directory `docs/unix` and is called `intro.tex`.

   Here is the sequence of commands you could use:

   ```
   ftp machine.div.com.ca
   system asks for login and password
   cd docs/unix
   get intro.tex
   system now transfers file
   quit
   ```

   The file is now in your account.

   Be wary of VAXes running VMS versions lower than 5.0 and the `CMU` TCP/IP package. They will hang the `ftp` utility if the first command issued is a `cd`. With those systems, issuing `ls` as the first command will avoid that problem.

8.4 Special Files

1. When logging into a system, the first of your files to be executed is `.cshrc`, followed by `.login`.

   But everytime you start a C shell, `.cshrc` is again executed. For that reason, a greeting message would be placed in `.login`, since that file is only executed once at logon time (or, in SGI systems, everytime a new `wsh` is opened).

2. First open `.cshrc`, then add/or modify the line beginning with the `set prompt` command:

   ```
   set prompt="!$user, $cwd ? "
   ```

   This will include your home directory when you first login.

   BUT what happens when you change directories???? The directory name does NOT change!!!

   This is a toughy! To get the directory name to change within the prompt when you change directories, an `alias` has to be created:

   ```
   alias cd 'cd \!*; set prompt="! $user, `echo $cwd` ? "'
   ```

   Note the three different type of quotes used. They are essential! (' and " are found on the same key on the right side of the keyboard, whereas ` is typically found on the upper left side.)

3. The simplest way is to add the following code:

   ```
   echo " "
   echo "There are `who | wc -l` users logged on to the system:"
   echo " "
   who
   echo " "
   ```

4. An alias for `ls -l` could be added with

   ```
   alias lls 'ls -l'
   ```

5. Similar to the last question, an alias is added:

```
alias hl 'cd; ls'
```

6. One variable is needed to set the history list to be 100 items. Another variable is needed to save the last 50 items between sessions. The changes are made in the `.cshrc` file for any system.

```
set history=100
set savehist=50
```

7. Add the following in `.logout`:

```
echo "logged out on `date '+%a %h %d at %H:%M'`"
> $HOME/.logout
```

(note: all on one line.)

Again, notice the several types of quotes used.

To get the format of the `date` command, look it up in the Reference Manual, or issue the `man` command.

8. The file *must* be named `calendar`, and *must* be in your home directory. The content of it should look like

```
Jan 1:  Happy New Year.
Jan 28:  Meeting with the head of the group.
Feb 14:  Valentines Day.  Got to buy flowers!!!
Feb 29:  Leap year!
Mar 3:  Meet John Doe for Lunch.
Mar 4:  Meeting with Director General.
```

The first item on each line must be the date. The date can be expressed in several formats. Issue `man calendar` or look it up.

To start the program, you could add the following lines in your `.login` file:

```
echo "************************************
echo " "
calendar
echo " "
```

```
        echo "**************************************
```

8.5 Compilers

1. A short C program would look like:

```
main() {
printf("hi!\n");
}
```

   A short FORTRAN program would look like:

```
program hello
print *, "hi!"
stop
end
```

   You could use default flags on the compiler by issuing

```
make hello
```

   to compile the program.

2. To compile with optimization 2 on file `hello`,

```
cc -O2 -o hello hello.c
```

   How would you do it for the FORTRAN program? Try it.

3. Most makefiles begin with an `M` rather than an `m` because the letter `M` comes before any lower case letters. Hence `Makefile` is always closer to the beginning of a directory, and the `make` utility finds it quickly.

4. A simple `Makefile` to compile your program could be

```
hello: hello.c
        cc -O2 -o hello hello.c
clean:
        rm hello
```

   A more expandable `Makefile` would look like

```
OBJS= hello.o
CFLAGS= -O2
all: hello
hello: $(OBJS)
        cc -o hello $(OBJS)
hello.o:        hello.c
clean:
        rm *.o hello
```

WORD OF CAUTION: `make` is VERY picky: it *requires* tab spaces at the beginning of a line which has no labels.

5. See the solution of the first question for this solution. If `-O2` is still desired, the variable `CFLAGS` can be set within the shell:

```
setenv CFLAGS -O2
```

At which point `make hello` could be issued.

8.6 UNIX Batch Systems

1. To send a job, say `myprog`, into the background using the same priority level as interactive jobs, use

   ```
   myprog > myprog.output &
   ```

   at which point, any other commands could be issued.

2. Depending which system you are working on, there are a number of ways this can be done.

   First, one should create a file, say `batch.job`, whose content is the job to be run. In our case, `myprog` will be the only entry in that file.

   ```
   at 3:00 < batch.job
   ```

   On any system, the same can be accomplished using

   ```
   at 3:00 <CR>
   myprog <CR>
   <CTL-D>
   ```

3. The preferred way is to use the `batch` command (we assume that the job(s) to be run are in `batch.job`).

   ```
   batch < batch.job
   ```

   Or, similar to `at`,

   ```
   batch <CR>
   myprog <CR>
   <CTL-D>
   ```

   `batch` will start the job whenever system load permits, therefore not interfering with interactive jobs.

4. To run a series of jobs, all producing output, `batch` could again be used. The simplest way would be to create a file (`batch.job`) with entries similar to

```
echo "This is job1."
/bin/time job1
echo "This is job2."
/bin/time job2
```
and so on...

Then start the process using

```
batch < batch.job
```

This would start the jobs, time them, and output the result into the mail message sent to you when all jobs are finished. The echo message is to remind you which job's output follows.

Try `/bin/time` on a fairly short program and see how it reacts.

5. To run a program periodically, the `crontab` utility is used. To see if you have any scheduled cron jobs, issue

```
crontab -l
crontab -l > crontab.file
```

`crontab.file` could then be removed.

The first line displays your schedule on the screen; the second redirects it into a file.

Add

`30 2 * * 3` *job.name*

where *job.name*'s path is from your home directory. Then, to activate the cron jobs,

```
crontab crontab.file
```

6. Suppose the two jobs were `job1` and `job2`. What follows are the steps taken to start running those jobs, then the steps taken to delete them from the run queue, and restart `job1`.

```
at 1:00PM tomorrow <CR>
job1 <CR>
CTL-D
at 3:00PM tomorrow <CR>
```

```
job2 <CR>
CTL-D
at -l
at -r job #1
at -r job #2
at 2:00PM tomorrow <CR>
job1 <CR>
```

8.7 X based Graphical User Interfaces

1. `Motif` is actually written using the `X Window` software. So if your friend is telling you they don't have any of X, then he is lying.

2. The command

   ```
   xterm -geometry 80x24 &
   ```

   will open an `xterm` window. That window will include a shell for you to issue UNIX commands.

   The ampersand (&) is recommended so you can still use the parent window to issue commands.

   If an error message similar to

   ```
   Could not open display.
   ```

   was displayed, your `DISPLAY` environment variable is not set. Issue

   `setenv DISPLAY` *localhost*`:0` (C shell)
   or
   `DISPLAY=`*localhost*`:0; export DISPLAY`

   where *localhost* is the name of the system you're using. That command will tell the application (`xterm`) which system to use to display.

3. Add the line

   ```
   XTerm*geometry:  80x24
   ```

   Then, from the command line, simply issue

   ```
   xterm &
   ```

4. The command

   ```
   xterm -g black -gb white &
   ```

will open a white `xterm` with black characters.

To create the same effect, the lines

```
XTerm*foreground:  white
XTerm*background:  black
```

could be added to the `.Xdefaults` file, so that every `xterm` command issued will contain the same attributes.

5. The command

```
xlsfonts | more
```

will list fonts available to you during this session.

6. An `xclock` application can be started by

```
xclock -geometry 100x100+0-0 &
```

or the line

```
Xclock*geometry:  100x100+0-0
```

could be added to the `.Xdefaults` file.

Chapter 9

Bibliography

Bibliography

[1] Stephen R. Bourne. *The UNIX System V environment*. Addison-Wesley Publishing Company. Don Mills, Ontario. 1987.

[2] D. Dougherty, R. Koman, and P. Ferguson. *The Mosaic Handbook for the X Window System*. O'Reilly & Associates, Inc. Sebastopol, California. 1994.

[3] E. Foxley. *UNIX for Super-Users*. Addison-Wesley Publishing Company. Don Mills, Ontario. 1985.

[4] Æleen Frisch. *Essential System Administration*. O'Reilly & Associates, Inc. Sebastopol, California. 1992.

[5] Ed Krol. *The Whole INTERNET User's Guide & Catalogue*. O'Reilly & Associates, Inc. Sebastopol, California. 1994.

[6] Jerry Peek, Tim O'Reilly, and Mike Loukides. *UNIX Power Tools*. O'Reilly & Associates, Inc. Sebastopol, California. 1993.

[7] H. McGilton and R. Morgan. *Introducing the UNIX SYSTEM*. McGraw-Hill Software Series for Computer Professionals. Toronto. 1983.

[8] R. Thomas, and R. Farrow. *UNIX Administration Guide for System V*. Prentice Hall. Englewood Cliffs, New Jersey. 1989.

[9] Silicon Graphics Inc. *IRIS-4D User's Guide*, man pages.

[10] Sun Microsystems. *SunOS 4.0, 4.1 Reference Manuals*.

[11] SuSE Linux LTD. *SuSE Linux 7.3 Reference Manual*.

[12] UNIX International. *The UNIX Operating System: A Commercial Success Story*. Nov 1, 1989. Parsippany, NJ.

[13] http://www.canarie.ca; November 1997 version.