Note: you can find this file under:

`http://www.cs.queensu.ca/~acmteam/unix.pdf`

Introduction to Unix Tutorial

In this tutorial, you will learn:

- How to manage your account (user ID, password, shell);
- Navigating through your home directory and manipulate your files;
- How to compile C, C++, and Java programs;
- How to query, send, and remove print jobs;
- Process control;
- Common Unix tools (find, diff, grep, tar, zip, etc.)

History of Unix

- Unix is a multi-user, multi-tasking system.
- Development began in 1965 and it was released in 1970.
- Designed as a programmers' environment (Unix is written in C) with a simple command line interface.

Why Unix?

- Unix is a multi-user, multi-tasking system.
- It is portable, consistent, flexible, and powerful.

Unix Variants

- Several variations exist: Solaris, Linux (Mandrake, Debian, Slackware, RedHat, Corel), HPUX, Digital Unix, and Ultrix, etc.
- All of the commands you learn here can be used on all variations of Unix.

Unix File System

- Each node is either a file or a directory.
- Each directory can contain other files and directories.
- A file or directory can be specified by its absolute path name, or its relative path name.
- An absolute path name starts with the root, `/`, and follows the branches of the file system, each separated by `/`, until you reach the desired file, e.g.:

```
/home/condron/source/xntp
```

- To see the absolute path name of the current directory, use `pwd`.

- A relative path name specifies the path relative to another, usually the current working directory that you are at. Two special directory entries exist:

`.`	the current directory
`..`	the parent of the current directory

For example, I have a file `/home/condron/source/xntp`. I am now at `/home/frank` and wish to specify the path above in a relative fashion I could use:

```
../condron/source/xntp
```

This indicates that I should first go up one directory level, then come down through the `condron` directory, followed by the source directory and then to `xntp`.

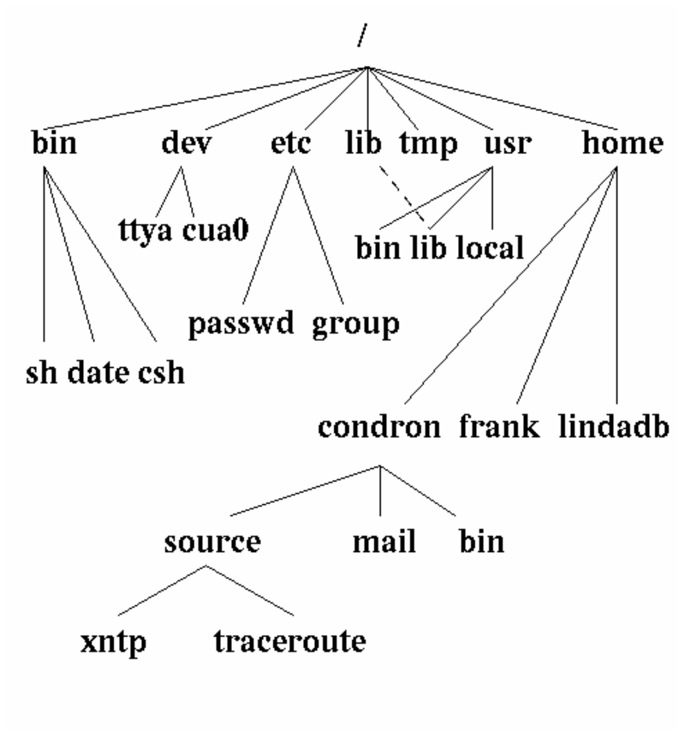

Figure 1: Unix file system example.

Basics

- The shell is where you enter commands interactively.
  - Sometimes referred to as a terminal session.
  - When you log on to a Caslab Unix machine under CDE, you can right-click the mouse on the background, under “Tools”, choose “Terminal” and a terminal session would appear.
- Manual pages (called man pages) are available online for all the commands. To see the man page of a command, use:

  `man` *command*

Changing password and shell

- To change the password, enter:

```
passwd
```

  and follow the instructions.

- The default shell at caslab is `csh`, but `tcsh` is recommended as it is more powerful and easier to use.
- This tutorial will assume that you are running `tcsh`.
- To change the shell, enter:

```
passwd -r nis -e
```

- When prompted for the new shell, enter:

```
/usr/local/bin/tcsh
```

Some Special Keys Under `tcsh`

Key	Description
Ctrl-U	Delete everything on the command-line
Ctrl-A	Move cursor to the front
Ctrl-E	Move cursor to the end
Ctrl-P	Set the current command-line to the previous command
Ctrl-N	Set the current command-line to the next command
TAB	Filename completion

File Manipulation

- Filenames:
  - You can use almost any characters for file/directory names. Each name can be up to 255 characters long.
  - Filenames are case-sensitive.
  - To avoid confusion, use only letters (A-Z, a-z), numbers (0-9), underscore "_", comma ",", and period "." for filenames.
- To see the content of a directory, use `ls`.
- Filenames start with period are "invisible". They can be viewed by using `ls -a` (meaning all).

File Manipulation (cont'd)

- Directories:

`chdir` or `cd`	change directory
`mkdir`	make a new directory
`rmdir`	remove an empty directory

- Use `rm` to remove file(s), `cp` or `mv` to copy or move files/directories:

```
cp sourceFile destinationFile
cp file1 file2 file3 destinationDirectory
```

- `rm` and `cp` can be used recursively to remove or copy directories and all their files including any subdirectories and their files by adding the switch `-r`.

```
cp -r sourceDirectory newDirectory
rm -r uselessDirectory
```

File Manipulation (cont'd)

- `.` represents current directory, `..` represents parent directory, `~` represents your home directory, `~foobar` represents the home directory of the user "foobar".
- Wildcards can be used to refer to a number of files/directories with some commons.

`?`	any single character	`ls abcde?`
`*`	any number (≥ 0) of any characters	`ls *cd*`
`[ ]`	any specific characters	`ls abcd[aeiou]f`

- Filename completion is useful when you are specifying a filename:
  - The shell will fill in the rest of the filename if you press TAB.
  - If there are more than one match, the shell will display the choices you have.

File Permissions

- Each file has its own permission. Running `ls -l` (long) will list files in long format: (this is `ls -lF`)

```
drwxr-xr-x   3 ttang  graduate   512 Aug 26 14:37 pub/
drwx--x---  10 ttang  ces       3072 Oct 19 18:06 thesis/
-rwx------   2 ttang  graduate   552 Nov 30 19:07 www*
```

- There are 3 catagories of permissions: user, group, and others.
- Each catagory has 3 modes:

Mode	Symbol	For Files	For Directories
read	`r`	readable	can be listed if accessible
write	`w`	writable	writable
execute	`x`	executable	accessible

- The first character of the permission is `d` if it is a directory.

File Permissions (cont'd)

- To change the permission, use `chmod`:

  ```
  chmod a+r somefile
  ```

  So the file `somefile` will be readable for all users:

  ```
  -rw-r--r--  3 ttang  graduate 512 Aug 26 14:37 somefile
  ```

- `chmod` can be used recursively by using the switch `-R`.
  - For example, `chmod -R og=u,og-w mynotes` will make everything under the directory `mynotes` to have the same *group* and *others* permissions as I do, except they cannot write to any of these files.

Finding Files

- To find a file with a specific name, use `find`:

  ```
  find .  -name "*ab*" -print
  ```

  where the starting point is the current directory. (This will find all files under the current directory that has “ab” somewhere in the name.)

- `find` can be used to find files using a lot of other attributes:

  ```
  find .  -mtime +2 -print
  ```

  finds files that have been modified for more than 2 days.

  ```
  find .  -atime -5 -print
  ```

  finds files that have been accessed for less than 5 days.

Viewing Text Files

- `cat` can be used to display the content of file(s):

```
cat file1 file2
```

- `head` and `tail` can be used to display the beginning or end of file(s):

```
head -10 file

tail -5 file
```

- `more` and `less` can be used to display a file page by page:

```
less file.c
```

Printing Files

- `lpr -P`*printer filename*: send the specified files to the printer.
- `lpq -P`*printer*: list all the print jobs at the printer:

```
zeus%  lpq -Pw0
Warning: w0 is down: offline
Rank    Owner   Job     File(s)                         Total Size
1st     ttang   71      foobar                          1582 bytes
2nd     root    72      barfoo                          582 bytes
```

- `lprm -P`*printer jobnumber*: remove print job from a printer:

```
lprm -Pw0 71
```

Finding Information in Files

Tools	Usage
`file`	Display file classification. `file somefile`
`grep`	Search pattern in files. `grep int file.c`
`wc`	Display number of charaters, words and lines in files. `wc somefile`
`cmp`	Compare two files. `cmp file1 file2`
`diff`	Display line-by-line differences between two files. `diff file1 file2`

I/O Redirection

>	Dump the output to a file	`cat f1 f2 > f12`
<	Use a specific file as input	`cat < inputfile`
|	Use the output of the previous command as the input of the next command	`grep pattern file | less`

Process Control

- Everything so far is running on the foreground.
- For program that will run very long, it can be put in the background:

```
find . -name "*a*" -print > files_with_a &
find . -name "*e*" -print > files_with_e &
```

- Then when you enter `jobs`:

```
[1] + Running  find . -name *a* > files_with_a
[2] - Running  find . -name *b* > files_with_b
```

- To put a current job on the foreground, use `fg %`n;
  to put a current job on the background, use `bg %`n;
  to terminate a current job, use `kill %`n.

Process Control (cont'd)

- To find out about the process on the CPU, use `ps`.
- There are two versions of `ps`, `/usr/bin/ps` and `/usr/ucb/ps`
  - For `/usr/bin/ps`, using `ps -ef` will show all running processes;
  - For `/usr/ucb/ps`, use `ps auxw` instead.
- To terminate a job on the CPU, use `kill process_id`;
  if that does not work, use `kill -9 process_id`.

Remote Login

- The best way to login remotely is to use `ssh`.
  - `telnet` works but it is not secure.
- To login to the machine `zeus.caslab.queensu.ca` as user `3abcd` in Caslab, enter this under the shell:

```
ssh -l 3abcd zeus.caslab.queensu.ca
```

Editing Text Files

- The standard editors of Unix are `vi` and `emacs`.
  - They can be hard to use for beginners
- Other editors are available in Caslab:
  - `jed` works inside a terminal session (e.g. under `ssh`)
  - `nedit` has a full graphical user interface but works only under X Window (Unix's window system)

Compiling Programs

- To compile a C program (e.g. `somefile.c`), use `gcc`:

```
gcc -o somefile somefile.c
```

  - If it compiles fine, an executable file named `somefile` will be created, otherwise there will be error messages.
  - To run the executable file, enter `./somefile`
- To compile a C++ program, use `g++`:

```
g++ -o somefile somefile.cc
```

- To compile a Java program, use `javac`:

```
javac somefile.java
```

  and class file(s) will be generated. To run the main class file:

```
java somefile
```

Common Utilities

- `tar` is a packaging utility that can take files and directories and store them as one big file, or extract existing tar files.
- To extract an existing tar file:

```
tar xvf file.tar
```

- To create a tar file:

```
tar cvf newfile.tar files directories ...
```

- To see the content of a tar file:

```
tar tf file.tar
```

Common Utilities (cont'd)

- `gzip` and `gunzip` are utilities for compressing and uncompressing a file. A file compressed by `gzip` will have a file extension of `.gz`.
- To compresss a file, use:

  ```
  gzip filename
  ```

  then the file `filename` would become `filename.gz`.
- To uncompresss a file `filename.gz`, use:

  ```
  gunzip filename.gz
  ```

  then the file `filename.gz` would become `filename`.

Common Utilities (cont'd)

- `zip` and `unzip` compress and uncompress files similar to Winzip, pkzip, and pkunzip in DOS/Windows.
- `zip -r myzipfile.zip firstdir seconddir` makes a zip file called `myzipfile.zip` which stores all files under the directories `firstdir` and `seconddir`.
- `unzip myzipfile.zip` extracts the content of the zip file to the current directory.
- `unzip -v myzipfile.zip` views the content of the zip file.

Common Utilities (cont'd)

- `ftp`/`sftp` can be used to transfer files between system.
- Say I want to get a file from a Caslab Unix machine, and the file is located at `cisc271/myfile.m`, then I can do `ftp` (available in MS-DOS also) from elsewhere:

  `sftp 3abcd@zeus.caslab.queensu.ca` (*secure*)

  `ftp zeus.caslab.queensu.ca` (*not secure*)

  enter your login and password, then a prompt

  ```
  ftp>
  ```

  will show up. At the prompt, you can enter:

  ```
  ftp> cd cisc271
  ftp> get myfile.m
  ```

This will put `myfile.m` in your current directory.

Common Utilities (cont'd)

- To transfer from the local machine to the remote machine, use `put` at the ftp prompt.
- At the end you can enter `quit` to quit.

CASLAB Files

- In CASLAB, you do not need to transfer files from your NT account to the Unix account.
- Under Unix, your NT files are stored under the directory `~/.NTfiles`.