PHYS-4007/5007: Computational Physics
Course Lecture Notes
Appendix A

Dr. Donald G. Luttermoser
East Tennessee State University

Version 7.0

Abstract

These class notes are designed for use of the instructor and students of the course PHYS-4007/5007: Computational Physics I taught by Dr. Donald Luttermoser at East Tennessee State University.

Appendix A. Operating Systems

A. A Brief History of Operating Systems.

1. A simple definition of a operating system is the suite of programs that make the hardware usable. The operating system manages the CPU, disks, and I/O devices.

2. Manipulation of the operating system was typically one of the hardest aspects of learning to use computers which is why the "GUI (Graphic User Interface) mentality" has taken hold of the modern computers — the user no long needs to talk to the operating system, the GUI does it for you! (Which is actually a "pain-in-the-butt" if you need to actually talk to the operating system!)

3. We now follow the history of operating systems and programming languages that are important in the scientific community, starting in the 1940s:

a) Early computers had no operating system (*e.g.*, the IBM 604).

b) The computer was told what to do with a low-level set of commands $\Longrightarrow$ **assembly language**.

c) The IBM 604 could undertake 60 program steps before using punch cards as a backing store.

d) Often, the user had to manipulate toggle switches to input the code for the mainframe.

4. The 1950s saw the advent and rapid changes in the capability of operating systems.

a) Jobs were *batched* so that the time between jobs was minimized.

b) The user was now distanced from the mainframe and entered a program via a card reader.

c) This era saw the rapid development in higher-level programming languages such as Algol 60 and the first version of Fortran.

5. The 1960s saw the advent of multiprogramming and the concept of *time-sharing*.

a) *Multi-user* operating systems were developed which allowed users to actually log into the computer simultaneously.

b) Since scientists were the primary users of mainframes, an official programming language for the sciences was declared: Fortran IV (which was also called Fortran 66).

6. The 1970s saw numerous multi-user operating systems come on line. Two of the most successful were Unix and VMS (Virtual Memory System).

a) Unix developed by Bell labs in 1970 and further modified by the University of California at Berkeley.

b) Unix is written in the C programming language and is capable on running on a variety of different architectures.

c) VMS was released in 1978 by the Digital Equipment Corporation (DEC) to run on their VAX mainframe computers.

d) Whereas Unix commands are often cryptic, VMS commands were based on English words, and hence was considered much more "user-friendly" than Unix. As such, it was typically the preferred OS by scientists during this decade, whereas, Unix was favored by the computer "scientists."

e) The 1970s also saw a new Fortran standard come on line $\Longrightarrow$ Fortran 77. This version of Fortran was much more versatile and powerful than the earlier version of this programming language.

f) Finally, the 1970s saw the beginnings of the PC market open up through the release of the Apple IIe and IBM PC (running DOS) computers.

7. The 1980s saw an explosion of computers in science through the release of the *microcomputer* (which lasted only a few years) and the *workstation*. Both of these types of machines were scaled down versions of mainframes, with the workstations being designed to fit on a desk.

a) The most popular workstations at that time was the VAXstation, running VMS, and the Sun workstation, running Sun OS.

b) During this decade, Unix's popularity grew leaps and bounds, whereas, VMS declined.

c) Later this decade, DEC came out with their version of Unix to run on the VAX chip called *Ultrix*. Machines that had Ultrix installed on it were called DECstations.

d) Apple came out with a new 32-bit chip that they placed in a new machine called the Macintosh.

8. In the 1990s, there was a PC explosion which left the workstations in the dust.

a) Microsoft began to dominate this market with their various flavors of the Windows OS (note that Windows originally was a GUI front end that sat on top of DOS). 16-bit PCs were replaced by 32-bit PCs during this decade. PCs started becoming very fast when the Intel Pentium chip was developed.

b) DEC came out with a new 64-bit chip (the VAX chip was 32-bit) called the *Alpha* which gave rise to the *Alpha-Stations*. This chip required new versions of DEC's OSs: VMS → OpenVMS, and Ultrix → OSF/1 (which later was renamed Digital Unix in 1998, then Tru64 Unix in 2000 when Compaq bought out DEC).

c) Sun then came out with the Sparc-station (32-bit CPU) and a new OS called Solaris to run on it.

d) This decade saw the last of the mainframes, as the workstation and PC explosion made this type of computing obsolete. Because of this, and the fact that their Alpha-Stations were not as popular as their VAXstations, DEC went "belly-up" at the end of this decade.

e) Finally, this decade saw the emergence of a "free" version of Unix called Linux which was designed to run on PC-class machines.

9. In the 2000s and beyond, the bulk of computing was done by PCs. However, up through about 2005, the physical science community did most of their number crunching on workstations. Post 2005, many scientists started switching to PCs due their fast speeds and low costs.

a) Sun came out with a new 64-bit chip and workstation called the Ultra stations, followed by the multi-processor Blade computers.

b) For workstations, the biggest sellers were Suns, Silicon Graphics, and IBM Workstations. By the end of this decade, these workstations gradually became less popular mainly due to their high cost.

c) In 2002, Hewlett Packard bought Compaq and announced that they were dropping the AlphaStation from their line meaning the death of both Tru64 Unix and OpenVMS. This is a real pity since Tru64 Unix was the most secure version of Unix on the market. (Note that OpenVMS was also a very secure operating system.)

d) In 2003, Apple came out with its 64-bit, dual processor, Power Mac G5, running Mac OS X which is a Unix-based operating system.

e) In 2004, the Athlon 64 chip by Advanced Micro Devices (AMD) and the Intel Xeon 64-bit chip have appeared in PCs. These machines run either the 64-bit version of Microsoft Windows or the 64-bit version of Linux.

f) Intel continuously comes up with faster and faster chips. Their latest CPUs are called Core and Xeon processors.

B. The Unix Operating System.

1. Since most of you are well aware of the workings on the various flavors of Microsoft's OS (*i.e.*, Windows), at this point, I will highlight the use of the Unix operating system since you will be working on the Department's Linux computers.

a) Unix/Linux is the operating system of choice for *number-crunching* 64-bit workstations and PCs in the scientific community (physics and astronomy in particular) and in technical corporations.

b) There are various *flavors* of Unix that exist on the market, each design for use by specific computer platforms.

i) Solaris was the flavor of Unix that existed on Sun Microsystems' platforms (*e.g.*, Sparc Stations and Ultra Station). Solaris superseded the earlier Sun operating system called SunOS. Sun was purchased by Oracle in 2009 and the Sun workstations have virtually disappeared from the market.

ii) Irix was the version of Unix used on Silicon Graphics' (later SGI) workstations. SGI filed for bankruptcy in 2009.

iii) HP/UX is the Unix flavor on Hewlett Packard's RISC-based workstations. With its purchase of Compaq/DEC, HP/UX had incorporated many of the security features contained in Tru64 Unix.

iv) Digital Equipment Corporation's (DEC) version of Unix was called Tru64 Unix (previously known as Digital Unix, and previous to that, OSF/1) before they went out of business. Even though HP no

longer sells AlphaStations, they do still sell Tru64 Unix for those people with still functioning AlphaStations.

v) There are at least two platform-independent versions of Unix, SCO (Santa Cruz Operating system) Unix and BSD/OS (Berkeley Systems Development Operating System) Unix.

vi) Mac OS X (and beyond) of Apple is based on the BSD version of Unix.

vii) Finally, the Linux version of Unix is designed to run on a variety of different platforms from Intel microprocessors to Sun/Ultra and Alpha chip architectures. Linux is one of the few versions of Unix that can be downloaded for free. Different companies have different names for their version of Linux:

- Red Hat two versions of Linux:
  - Enterprise Linux which costs money.
  - Fedora is Red Hat's freeware version of Linux.
- Ubuntu, which is the version of Linux on the machines in Brown Hall 264. This too is freeware.
- GNU Linux is Linux freeware developed by the people who created Emacs (*i.e.*, the GNU Project group).

2. The History of Unix.

a) Brian Kernighan and Dennis Ritchie both work at Bell Labs and were involved in the development of the Unix operating system and the C programming language.

b) In 1968, Ken Thompson of Bell Labs wrote the original version of Unix in PDP-7 (a DEC mainframe which preceded the VAX) assembly language.

c) In 1969 a language called TMG was ported to Unix — the compiler was written in assembly.

d) Using this, Thompson created B, a pared down version of BCPL. B was essentially C without types.

e) Dennis Ritchie added character, integer, and floating point types to B, anticipating floating point operations in new versions of the PDP. This was the beginning of the C programming language.

f) In 1973, the Unix microkernel was rewritten in C.

g) The K&R White book was first published in 1978. Brian Kernighan wrote the body of book, and Ritchie wrote the technical appendices.

h) The ANSI C standard was released in 1983.

3. Note that the Unix operating system is set up in a hierarchal system as shown in Figure 2.1 of your textbook.

a) The user and programmer "talks" to the *shell* (*e.g.*, the Bourne shell [sh], the Korne shell [ksh], the C shell [csh]). Note that these shells are not designed well for user interface at the keyboard (up/down/side arrows won't recall

previous commands or edit them). As such, other shells have been developed that allow keyboard editing: csh → tcsh, sh → bash.

b) The shell talks to the Unix utilities (*i.e.*, commands like 'cp' and 'ls').

c) The utilities talk to the kernel (*i.e.*, the actual operating system).

d) The kernel talks to the hardware.

C. The Structure of Unix.

1. Unix consist of two main parts

a) The **kernel** controls:

i) Computer memory.

ii) Resource allocation.

iii) Peripheral systems.

iv) Filestore organization.

v) File security and access.

b) The **shell** provides:

i) User interface or command processor.

ii) Utility programs.

c) Examples of *shell* are the **Bourne** shell (sh), **C** shell (csh), **Bourne Again** shell (bash), **Korne** shell (ksh), and the **tcsh** (tcsh).

2. Why use Unix?

a) Unix is portable.

b) Unix is a Multi-user Operating System.

c) Unix is a Multi-tasking Operating System.

d) Long running programs can be sent to **batch** which will run a process in background mode freeing up the terminal for coincidence interactive use.

e) Users have the ability to write scripts to control the running of complicated processes.

f) Unix supports the X-Window protocol. X-Windows was initially created by computer scientists from MIT to allow a windowing environment for multi-user operating systems such as Unix and VMS.

3. Getting Started in Unix.

a) Unix is designed to be user-interactive — that is the user can 'talk' to the operating system directly through *terminal* windows. **Much of the work you will do on the Linux machines will involve use of such a terminal window.**

b) The following commands are useful for checking on current users and changing your password:

passwd	sets users password.
who	allows you to see who else is logged on, showing the users on the system, their terminal ID numbers and when they logged in to the system.
whoami	tells you your own username and when you logged in.

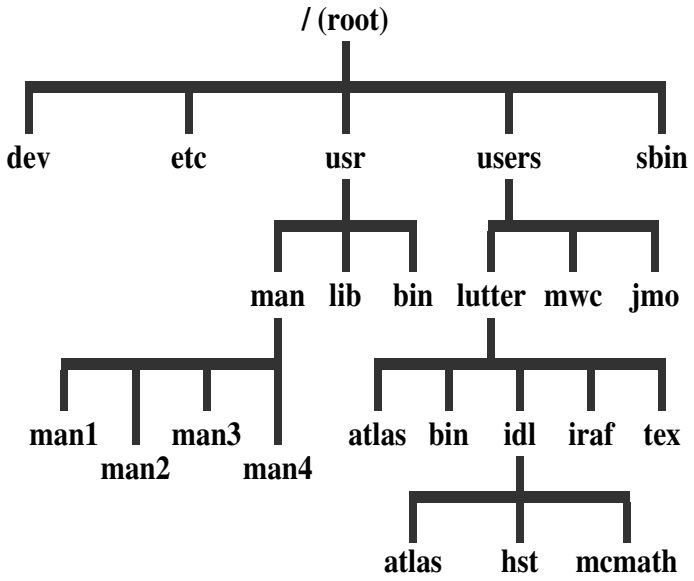

Figure A–1: Directory Tree Structure of a Unix File System

c) Getting Help:

man Displays the online manual page for a command.

d) Other useful utilities are available through 'pull-down' menus on the X-Window 'toolbar' (*e.g.*, Email, web browser, etc.).

4. Unix File System.

a) The Unix file structure is a hierarchy with the root directory '/' at the top (see Figure II-1).

b) When you log in to a Unix system you are assigned to a **login** (*home*) directory. For example: */users/lutter*.

c) The directory where you are working is your *current* directory. The *pathname* of the file gives you the exact location of the file. The *full path* is the absolute pathname relative to the root directory. For example:
/users/lutter/idl/mcmath

is the directory where I might keep my observed spectra taken with the McMath Telescope.

d) The *relative path* is the pathname relative to your current directory. For example: From the login directory */users/lutter* the relative path is *idl/mcmath*.

5. File Basics and Filenames.

a) Conventions.

i) Filenames can be any length, and almost any character is valid. However, you should avoid using the following characters in filenames since these characters have special meanings to the Unix operating system:

~ $ % & ( ) [ ] ' ‘ " ? \ ; < > + - | !

ii) Unix is case sensitive.

iii) There is no division into filename and file extension, although there are certain naming conventions. For example (the '*' character below is used as a marker indicating the root name of the file in question):

*.c	C language source code
*.f	Fortran 77 source code
*.f90	Fortran 90 source code
*.pro	IDL source code
*.tex	TEX or LATEX source code

iv) Wildcards can be used:

?	represents any one character
*	represents any number of characters, including none

b) Directory Navigation:

pwd prints the working directory to the screen so that you know where you are

cd changes directory, for instance:

cd /users/lutter/idl (moves you to that directory)

cd ∼ (moves you to your login directory)

c) Listing Files:

ls lists the (unhidden) files in the current directory

d) Command switches, or options, are placed after the command and before any arguments.

ls -a lists files including hidden files (note that a *hidden* file begins with a '**.**')

ls -l lists the contents of a directory in long form

6. File Management Utilities.

a) Copying Files:

cp makes a copy of a file. The syntax of the command is: cp *sourcefile targetfile*

mv moves or renames a file. The syntax of the command is: mv *sourcefile targetfile*

i) cp makes a new copy of a file leaving the existing sourcefile unchanged.

ii) The new copy will overwrite a file of the same name in the target directory.

iii) If there is more than one sourcefile then the targetfile must be a directory.

iv) mv creates a new copy of a file without retaining the sourcefile.

v) mv is used to move files from one directory to another.

vi) mv is used to rename files.

b) Removing Files:

rm	deletes unwanted files. The syntax of the command is: rm *filename*
rm *.old	deletes a group of files ending in *.old*
rm –i *.old	Unix asks to confirm a deletion before deleting files ending in *.old*

c) Creating and Removing Directories:

mkdir	creates new subdirectories. The syntax of the command is mkdir *newdirectory*
rmdir	deletes directories. The syntax of the command is: rmdir *directory*

i) mkdir can create a subdirectory of the current working directory.

ii) mkdir alos can create directories within directories other than the current. For example:

mkdir /users/data/newdirectory

(note that you must have "write" permission to create directories and files in directories "outside" your login directory tree).

iii) To remove a directory: (1) you must be the owner; (2) the directory must be empty; (3) it must not be the current directory.

7. Other Unix Commands and Tools.

a) Displaying Contents of Files:

cat	concatenates and displays the contents of a file. The syntax of the command is: cat *filename* If the file is more than 24 lines the display will scroll off the screen.
more	displays output from file one screenful at a time. The syntax of command is: more *filename*

b) Pipes and Redirection: The shell normally expects to receive input commands from **stdin** (the keyboard). Output is normally sent to **stdout** (the screen) and any error messages to **stderr** (the screen).

|	pipe allows standard output from one command to be used as standard input for another command. For example: cat *filename* | more

The output of the cat command is *piped* through the more command.

>	redirects stdout to a real file. For example: ls -l > filelist
<	redirects stdin to a real file. For example: newfile < filelist

Redirection always sends output to a file, whereas pipe sends output to another command. There is no limit to the number of pipelines that can be set up.

c) Searching for Strings:

grep	searches for text in either a single file or group of files. The syntax of the command is: grep *"search string" filename*
grep -i	ignores the case of the search string
grep -l	lists only the names of files containing the search string

wc	counts words and lines in files. The syntax of the command is: wc *"options" filenames*
wc -c	displays only the number of characters
wc -l	displays only the number of lines
wc -w	displays only the number of words

i) grep searches one line at a time.

ii) grep looks for strings of text and does not limit itself to whole words.

iii) grep can be used with wildcards.

d) File Permissions:

i) The default setting for a new files is:

```
-rw-------
```

which means that only the user can read and write a file.

ii) The permissions column consists of ten characters. The first character denotes the type of file. For example:

-	ordinary file
d	directory
l	link

iii) The next three show access mode for the user (owner) of the file.

iv) The next three show the access for the group.

v) The last three show access for all others.

vi) The protection symbols mean:

r	**read**:	allows user to read the contents of a file
w	**write**:	allows user to modify a file
x	**execute**:	allows user to execute or run a program file

e) User Access: The different classes are defined as:

u	user (owner) of file
g	group file belongs to
o	other users
a	all users

f) Changing Access Privileges: chmod changes the access mode of files you own, to restrict or allow access. The syntax of the command is:

chmod *"class(es)" "operation(s)" filename(s)*

"operation" is one of: + - = (to add, take away or set permissions). For example:

```
chmod ug+w example.dat
```

D. Summary of Unix Commands

1. There are a large number of commands in the Unix operating system. Table A-1 lists some of the more common ones.

2. Note that commands that are listed with Unix flags (*i.e.*, letters after the initial command word) require a minus sign "-". (Note that some flavors of Unix do not require the "-" qualifier.)

a) Most processes are run in *foreground* — this means that one does not get the Unix *prompt* back until the process is complete.

Table A–1: Common Unix Commands

Unix Command	**Description**
cat *file*	concatenate and display *file* contents
cd *dirname*	Change to directory *dirname*
cd ..	Go back to the parent directory of the current directory
cd ∼	Go back to *home* (*i.e.*, login) directory
chmod *ijk file*	Change the read-write-execute protection of a file
cp *file1 file2*	Copy file *file1* to file *file2*
df	Give information about the mounted disks
emacs *file*	Edit file *file* with the emacs editor (the shortcut command puts the emacs editor into a pretty screen with big fonts)
grep *str file*	Search file(s) for pattern
head *file*	Displays beginning of *file* (default is ten lines)
logout	Log off of the terminal session
lpr *file*	Print the file *file* on the laser printer
ls	Give listing of a directory
ls –l	Give a long listing of a directory
ls –a	Give listing of a directory including hidden files
man *command*	Display information (man pages) of the Unix command *command*
mkdir *dirname*	make directory *dirname*
more *file*	Type the contents of file *file* to the screen one page at a time
mv *file1 file2*	Move or rename file *file1* to *file2*
passwd	Change your password to a new password
printenv *var*	Display the value of the environment variable *var*
ps	Display current process status
ps –af	(just gives information on the user's processes).
pwd	Print working directory
rm *file*	Delete file *file*
rmdir *dirname*	remove directory *dirname*, if empty
source *file*	Run a Unix shell script called *file*
tail *file*	Displays end of *file* (default is ten lines)
who	Displays names and other information about users on system

Table A–1: (continued)

Unix Command	Description
latex *file*	Run the LaTeX file *file* through the LaTeX word processor
dvipdf *file*	Convert a dvi file *file* to a PDF file (one that can be printed on the laser printer)
dvips *file*	Convert a dvi file *file* to a postscript file (one that can be printed on the laser printer)
xdvi *file*	Preview a *dvi* file on an X-Windows terminal
idl idlde	Start IDL in command line mode Start IDL in GUI mode

i) For typical Unix commands and processes, this is the way to go since these commands and processes take a fraction of a second to run.

ii) However, for long running processes (editing a file for instance), it is often wise to put a process in *background* mode. To do this, just include an *ampersand* sign at the end of a command (*e.g.*, emacs myfile.txt &).

iii) The job will then run in background mode and the Unix prompt will immediately come back to the screen awaiting further input.

b) You can check on background jobs by entering the jobs command.

i) To bring a background job back to foreground mode, either enter fg to bring the last entered background job to foreground or enter fg *PID*, where *PID* is the process ID number which can be obtained with the ps command described above.

ii) If you ever need to stop a job while it is in background, enter kill –9 *PID*. (**Be very careful with this command!**)

c) Your can also *suspend* a foreground job by entering <Ctrl>–z (*i.e.*, pressing down on the *control* (Ctrl) key when you hit the *z* key — note that this doesn't work from inside an emacs process, <Ctrl>–z will save and exit the file you are editing).

3. Finally, we have only scratched the surface of Unix in this section of the notes. This, however, should be enough to enable you to work on the Linux machines.

E. The Emacs Editor.

1. Emacs is a user-friendly editor that exists on most Unix workstations.

a) Although emacs is not shipped with Unix itself, it is freely available to the Internet community.

b) It was written (and still evolving) by the *GNU Project*, a group of computer scientists who don't like the large amounts of money that vendors are charging for software — they write software and give it out for free!

2. Unix's standard editor is vi and it is very user-unfriendly — instead, use emacs!

3. To edit a file in the emacs editor, move to the subdirectory containing the file and enter emacs *filename*, where *filename* is the name of the file to be edited.

4. The most recent versions of emacs brings up a nice GUI widget. The buttons and pull-down menus are obvious in their operation, as such, we won't describe the keyboard commands that one could also use inside of emacs.

F. Additional Information about *Linux*.

1. Most of you are not used to talking to the operating system by issuing commands at the system prompt in a terminal window (though you should practice getting used to it).

2. Many Unix operating systems have a front-end windowing system based on the X Window protocol called CDE (Common Desktop Environment).

3. Many Linux operating systems (*e.g.*, Red Hat) use the gnome front-end which is also based on X-Windows.

4. As such, you can following the same protocol that you do on a Window's machine by *double clicking* on icons and through the use of pop-up menus from the "toolbar":

a) In Red Hat Linux, the toolbar is located horizontally at the bottom of the main console screen.

b) In Ubuntu Linux, the toolbar is located vertically on the left side of the main console screen.