Computational Physics (Lab)

PH 331

Help with Linux, C, gnuplot and $\LaTeX$

Last update: July 29, 2013

Contents

Linux, C, gnuplot and LaTeX for the naive

This manual is only a quick reference to bare minimum on the four topics. More information can be found easily on the web. Some links:

- Linux: http://oreilly.com/linux/command-directory/
- C programming: http://www.cprogramming.com/
- gnuplot: http://www.gnuplot.info/

  All required help and info on gnuplot can be obtained on your machine itself. Type *gnuplot* in a terminal and ENTER. At the gnuplot command prompt type *help* and ENTER.
- LaTeX: http://lahelper.sourceforge.net/mini_latex_tutorial.html

Chapter 1

Some Linux

- A primary character of all UNIX-like systems is the tree structure of the directories. The Linux operating system treats everything as a file (or a directory). A directory is a collection of files, and possibly further directories. The starting of this tree is indicated by '/'. When you login you are likely to be in your home directory whose path is mostly */home/me/* (where *me* is the user name) . After you login the commands can be entered in a terminal. Some of the common commands you will need are listed in these pages.
- Further help: Once you are in a terminal, more help on any Linux command can be sought using the command *info* followed by the command. For example, to know more about the command *mv*, its usage, all options that go with it and more, just type *info mv* at the command prompt and ENTER.
- To get a list of all linux commands available in your machine, say starting with the alphabet 'a', type 'a' and enter the TAB key.

  Given below is a list of commonly used Linux commands, with some frequently used options.

1.1 Some common Commands

- ls: List files

  Purpose: To get a list of all files in a directory

  Usage: ls [options]

  Common options

  -l (long format)

  -R (recursive)

  -t (sort by time)

  eg:>ls -lt

  (Command prompt is indicated by '>')

- cd: Change Directory

  Purpose: To change to a different directory

  Usage: cd [pathname]

  Common options

   None

  eg:>cd /homework/111

- mkdir: Make Directory

  Purpose: To create a new directory

  usage: mkdir [directory name]

  e.g.: >mkdir cmpt111

  Caution: Naming conventions are important to follow

   Linux is Case sEnsItive

   Reserved characters: *.—>< (and some more)

   Spaces in names are allowed (but a pain)

   Numbers are allowed, even at the beginning

- mv: Move

  Purpose: To move a file, or a directory, from one location to another.

  usage: mv [source] [destination]

  e.g.: >mv myfile.txt cmpt111

  >mv myfile.txt cmpt111/.

  >mv myfile.txt cmpt111/myfile.txt

  >mv myfile.txt cmpt111/myfiletoo.txt (Whoops. we just changed the name of the file while shifting).

- cp: Copy

  Purpose: To make a copy of a(ny) file.

  usage: cp [source] [destination]

  e.g.: >cp myfile.txt cmpt111/

  >cp myfile.txt cmpt111/myfiletoo.txt

  >cp myfile.txt cmpt111

  >cp myfile.txt cmpt111/myfile.txt

- rm: Remove

  Puropse: To remove a file or a directory

  usage: rm [options] [filename]

  common options:

  -r or -R (recursively all files and directories in a directory)

  -f (force)

  -d (remove directories)

  e.g.: >rm -r cmpt111

  >rm -rf cmpt111

  >rm -d *

  >rm myfile.txt

  Some more cool stuff

- Editors

  Appropriate editors are needed to create any file. For the purpose of programming some of the best editors are (in the order of the authors personal preference):

  emacs (or xemacs), kate, touch, gedit, gvim all of which require graphics and vi and pico.

  Any of these editors is invoked by entering the name in the command prompt.

  Eg:>gvim (to open the editor gvim)

  :>emacs & (to open the editor emacs. The & is just to open the editor in the background, and leave the terminal free.)

  :>kate foo.c & (to open a file named foo.c using the editor kate, and put it in the background)

  Some more useful commands:

- ps: Report process status

- top: View the cpu processes

- time: Time a command or report resource usage

- Who: Show who is logged in

- Kill: Terminate a process

- Ctrl+C: Interrupt (terminate and quit) a process running in the foreground.

- chmod: Change file permissions

  To know more about any of these command get help using the command 'info' followed by the command name.

1.2 Some more cool stuff!!

- *: A * can stand for any number of unnamed characters when you call a file.
  Try :> ls a* (lists all file starting with a)
- :¿ ls *a (lists all files ending with a)
- ? A single unknown character
  e.g.: >rm m?file.*
  Also
  Command and file name completion canbe done with TAB
  Command history (try)
  Paste buffer highlight with left click and middle click
  (clipboard)
- / == home
- ../ == move up a level in the directory tree structure
- ./ == current level in the directory tree structure
- ^{C} (Control-C) == Emergency Quit
- exit == quit current login shell session
- &: To run any command in the background
  eg: > emacs & (runs the editor emacs in the background)

Chapter 2

C programming

You are suggested to go through a better programming manual available online, or a book on C programming. A bare minimum is given below.

2.1 Editing, writing a program

Programs are written using editors, such as emacs, xemacs, gedit, kate, etc.,.

To start writing a program, open an editor, say emacs, enter the command

```
\> emacs & (to open the editor emacs in the background)
\> emacs foo.c & (to open a new or existing file 'foo.c' in the background using emacs)
```

Some sample program is given below (text appearing with /* and */, or after //, are comments that are ignored by the machine):

Program 1:

```
/*-----------------------------------------------------------------------------*/
/*This program illustrated basic C syntax. Read carefully and then try to compile*/
/*-----------------------------------------------------------------------------*/


#include<stdio.h>/*This has to be included to enable standard input and output*/
main()
{
  int i;/*This is how to declare an integer*/
  float x,y;/*This is how to declare a floating point*/

  x=1.;/*x is assigned a value 1.*/
  printf("Give y\n");/*To print a statement*/
  scanf("%f",&y);/*To scan/take input from you*/
  /*Here is a for loop*/
  for(i=0;i<10;i++){
    x=x+y;/*x is increased by y*/

    if(i%2==0)/*Here is a if conditional checks if i is even*/
      printf("x+y is %f\n",x);/*Here is how to print output*/
  }
}
```

Program2:

```
//This prog will calculate the recipocal of a number:
#include<stdio.h>
main()
{
float n,rec;
printf("Enter the value of n ");
scanf("%f",&n);
rec=1.0/n;
printf("\nThe recipocal of %f is %f\n",n,rec);
}
```

Program3:

```
//This prog will calculate the factorial of a number:
#include<stdio.h>
main()
{
int n,fact,i;
fact=1;
printf("Enter the value of n ");
scanf("%d",&n);
for(i=1;i<=n;i++)
fact=fact*i;
printf("\nThe factorial of %d is %d\n",n,fact);
}
```

Program4:

```
//This prog will find the no of odd nos from the given set of numbers:
#include<stdio.h>
main()
{
```

```
int x,n,count,i;
count=0;
printf("Enter the total no of elements in the set of nos ");
scanf("%d",&n);
printf("Start entering the nos ");
for(i=1;i<=n;i++)
{
scanf("%d",&x);
count=count+x%2;
}
printf("\nThe total nos of odd nos in the set is %d\n",count);
}
```

2.2 Compiling

C-is a 'high' level language, as opposed to 'low' level languages that machines can understand - strings of '1's and '0's. Programs written in a high level language, such as C, has to be then converted to machine readable (*executable*) form - a process known as *compiling*.

To compile a program file 'foo.c' using the gnome C compiler (gcc), and write the output binary file (an *executable*), say as 'foo',

```
\> gcc -o foo foo.c -lm
```

To compile with the debugger option:

```
\> gcc -g -o foo foo.c -lm
```

2.3 Running an executable program

To run a executable program, say 'foo', enter

```
\> ./foo
```

The './' is to indicate that the executable file is in the directory from where the program is being run. Else, the complete path from the current directory should be specified.

Chapter 3

Gnuplot

Gnuplot is a simple graphics based plotting tool under gnu-gpl that is widely used. To open gnuplot, enter the command gnuplot

```
\>gnuplot
```

This gets you the gnuplot command prompt in the terminal itself. Further help on Gnuplot, all topics, and subtopics, can be obtained on the machine itself. At the gnuplot command prompt type 'help' and ENTER.

For example,

```
\> pl 'out.dat' using 2:3 w l
```

for a *2-D plot* from a data file *'out.dat'*, *using* data in column '2' of the file along the 'x'-axis, and data in column '3' along the 'y'-axis, *with*(w) a *line*(l).

```
\> spl 'out.dat' using 2:3:1 w d
```

for a 3-D plot from a data file 'out.dat', using data in columns '2, 3' and '1' of the file along the 'x, y' and 'z'-axis respectively, with dots(d).

Upon execution of the command, the figure is displayed on the monitor. To get the output as a EPS file, instead of being displayed on the monitor, enter these two lines, followed by the plot command:

```
\> set term postcript eps enhanced color solid defaultplex "Times-Italic" 20
\> set out 'foo.eps'
```

A new eps file 'foo.eps' will be created containing the figure.

In the subtopic 'set' under 'help', the following topics may will be usefull: 'xlabel', 'ylabel', 'title', 'xrange', 'yrange'.

Chapter 4

LaTeX

LaTeXis a excellent tool for preparing scientific documents. To edit/write a file using emacs (or gedit, etc.,), enter

```
\> emacs labreport.tex &
```

This opens a new or existing file 'labreport.tex' in emacs in the background. Here is a template file for the labreport you will be submitting. A sample report can be found here. After editing the latex file, a pdf document has to be generated. First, latex file is compled to generate the dvi file:

```
\> latex labreport.tex
```

This generates a dvi file called 'labreport.dvi'. This can be viewed as

```
\> xdvi labreport.dvi
```

The pdf file can be generated from the dvi file as follows:

```
\> dvipdf labreport.dvi
```

This creates a pdf file 'labreport.pdf'. 'xpdf' is a simple pdf viewer

```
\>xpdf labreport.pdf &}
```

Other common viewers are acroread, gv.

Alternately, if you do not have any figures to be inserted, pdf file can be directly generated from the latex source file:

```
\> pdflatex labreport.tex
```

This directly generates the pdf file 'labreport.pdf'.

A more detailed manual on LaTeX can be found here.