1 A Very Short Linux Manual

Assume you have a computer that runs (any flavor of) Linux[1]. The programs we would like to be able to run in this course are FORM, LaTeXand whatever C programs we make. The first thing is that we need to be able to create directories and move to them. When you log in, you are usually in what is called your home directory. You create new directories with the command

mkdir nameofnewdirectory

This command is to be typed in in a so-called terminal window. There are also ways by which to make new directories with the mouse, but we will need the terminal window anyway. Usually you find the terminal program with the mouse in the menu under System. You can open any number of terminal windows. The ones that you keep open when you log out will open again automatically the next time you log in. With the mouse you select in which terminal window you want to issue commands. You can see which files and/or subdirectories you have with the ls command. There are several useful varieties of this command:

- ls Just list what files and directories there are
- ls -l Give more details in the listing.
- ls -la Also show the hidden files (names starting with a period)
- ls -ltr Order the files such that they are in order of last access.
- ls name Give an ls of the named directory or file.

There are more options. One can obtain information about any command with the man command e.g.:

man ls

To change directory, use the cd command as in

cd nameofsubdirectory

To go to the parent directory use the command

cd ..

and to go to your home directory simply type

cd

Unlike windows the directory separator is the / symbol as in bin/form

For the later stages of the course we will need the FORM program. There is no hurry to do this immediately, but sooner or later you will have to do this. If this is your first time with Linux and you have to install FORM, please go to your home directory and type the ls command. This should show you whether there is a subdirectory called bin. If not, please create this directory with the command

mkdir bin

[1]Many of the commands below also work under Cygwin and on Apple systems if you open a terminal window. On many Apple systems the C compiler has not been installed though, and maybe also not the TeX system. These then have to be installed first. They should be in the AppStore.

Next you are advised to start up FireFox, Safari or any other web browser and connect with github.com. Now you have to options:

First the lazy one: go to github.com/FormLink/form-binaries and select the proper binary for your computer. Download it, if possible directly into the bin directory. Next there is an important step to make. Go back to the terminal. If the form file is in your home directory you move it to the bin directory. This can be done with

cd
mv form bin
cd bin

The important step is now to make the file executable. If you type the command

ls -l form

you may find something that starts with

-rw-r–r–

In that case you and others can read the file and you can write to it, but you cannot execute it. Type the command

chmod 755 form

and after the next 'ls -l form' you will see

-rwxr-xr-x

indicating that the execution flag has been set. And as usual you can find more information about the chmod command by typing

man chmod

The second option is the better one. You start up your web browser and go to the page github.com/vermaseren/form. This is where the Form sources reside. Click on the green "Clone or download" button and download the ZIP file.

Next go again to the github.com/vermaseren/form page and click on "Find file". This gives you a list of files and click on the file "INSTALL". Next follow the instructions there.

Making the file this way gives you also the manual in .pdf format. The online version of the manual is at www.nikhef.nl/∼form/maindir/documentation/reference/online/online.html

If you compile the sources on your own computer you have the best guarantee that things will run properly.

The next thing you have to do is a bit trickier. You will need a text editor. The Linux system has a variety of them. The two main editors are based on either emacs or vi(m). If you are familiar with one of those you will have an easy time. There are also the products of the 'office' family, but those are usually not suitable for preparing programs. There is also an editor (with explanations of how to install it) in the FORM distribution. It is called stedi. Below I will assume that you are capable of running an editor of your choice. I will call it 'editor' in the commands and we will not discuss what commands you should type exactly inside the editor; we will only tell you what has to be done.

Now go to your home directory and type

editor .cshrc

Look whether there is a line that says something like

```
set path = ( . ~/bin $path )
```

If not, add such a line. This will make the shell program in the terminal find the FORM executable from any directory that you are working in. It means that you need only one copy of FORM.
The above .cshrc file works properly if the shell program that is run inside your terminal windows is for instance the tcshell (tcsh). With the bash shell you may have to edit the .bashrc file. There the syntax is slightly different. You have to add the line

```
export PATH=$PATH:~/bin:.
```

After adding the line to the file, you should save the file and leave the editor.
Next you can make a subdirectory in your homedirectory that you can use for this course e.g.:

cd
mkdir course

and you can pick up the other files and put them in this directory. I will keep adding files to the dirctory in the FORM distribution. Hence you are advised to check regularly whether there is something new.
Sometimes you find a so-called tar file. The word tar stands for tape archive. Of course we don't have tapes anymore but the name has survived. It is a file that can contain whole directories and lots of other files and its purpose is to make file transfers easier. It takes much less effort to transfer one file in this way than to try to transfer thousands of small files spread over many directories. It also makes it easier to compress the files with the gzip program. These files can be recognized by the extensions .tar for the tar file, .gz for a gzipped file and .tar.gz or .tgz for a tar file that has been gzipped. As an example let us take the file kinc.tar.gz which you can pick up from the site. You are to put this file in the subdirectory course and then go to this directory with the command

cd course

Then you can do either of two things. The first is

gunzip kinc.tar.gz
tar -xf kinc.tar

or in one command

tar -xzf kinc.tar.gz

In the first case we first uncompress the file, recreating the .tar file and then we unpack its contents. The -xf tells the tar command that you want to extract (x) the contents of the file (f) of which the name follows. In the second command the extra character z says that the name of the file refers to a gzipped file and tar can do everything in one step.
The gunzipping and untarring of the kinc.tar.gz file will create a subdirectory kinc with a number of files in it. You can enter this directory with the command

cd kinc

For the moment we will worry only about the first few examples. You can compile the first example with the command

cc test1.c ranf.c -o test1 -lm

This command calls the C compiler (called cc or gcc) and tells it to translate the files test1.c and ranf.c. It then has to produce an executable by the name

test1 (-o indicates the name of the output file. If it is omitted the default name will be a.out) and the -lm means that the math library should be included (-l stands for library and the library that will be searched for in the system is the file libm.so). The math library is needed when we use functions like the square root etc.
The program can now be executed with the command

    test1

or

    ./test1

Similarly the second example can be compiled with

    cc test2.c ranf.c -o test2 -lm

The third example involves the files main1.c, vegas.c , inplot.c, boundaries.c, ranf.c, fun1.c, iipow.c and ipow.c. This is a bit much to type all the time. Moreover we don't want to recompile all routines each time we change someting in only one of them. For this all unix systems are equipped with a facility named 'make'. In a file named the makefile we tell the system about what files belong to our project and how they should be translated. In the case of this example we have done that in the file make1. Then all we have to do is type

    make -f make1

and the system looks which files will have to be translated. By looking at the dates of the source files and the object files (the translated files) it can see which source files have been changed since the previous compilation.
If you have done all this and there were no problems you should have a file prog1 which contains the executable file of this example which is the first example run of the Vegas Monte Carlo integration program. You execute it with the command

    prog1

For prog2 (the second example) we become a little bit fancier. Again, you can make it with

    make -f make2

but for the execution this program needs an argument as in

    prog2 pict2.tex

We want here the name of a future file that has the extension .tex because this file will contain LaTeXcode. If everything has gone well, this code can be processed with the commands

    latex pict2
    dvips pict2 -o
    okular pict2.ps

The last command starts up the postscript viewer and will show you some of the histograms that were made during the running of the Monte Carlo integration. The first command lets the TeXsystem (using the LaTeXpackage) translate the .tex code into a device independent file with the extension .dvi which in its turn can be translated into a postscript (.ps) file with the dvips command in which the -o option at the end tells that the output file should be called pict2.ps. okular is the postscript viewer. If you prefer to have a .pdf file you can use the command

    ps2pdf pict2.ps

and you will get a file pict2.pdf which you should be able to show on the screen with the command

    okular pict2.pdf

On Apple systems there is no okular program and you have to use the Preview.app command as in

    open -a /Applications/Preview.app/Contents/MacOs/Preview pict2.pdf

One may consider it rather user-unfriendly to have to type this in all the time. Hence what you could do is add a line to .cshrc or .bashrc file which reads

    alias okular open -a /Applications/Preview.app/Contents/MacOs/Preview

With this line active (after the next time you open a new terminal window), you only have to type

    okular pict2.pdf

again, because the okular command is now automatically replaced by that whole open command.

2 Mail

Sometimes there is a spam filter active that may send e-mail that comes from 'suspected' sites to the spam. One way to prevent this is to whitelist a given sender. Any modern mailreader should have an option somewhere to 'manage' the whitelist. Make sure that as long as you are not finished with the course the address t68@nikhef.nl is in the whitelist, unless you have no problems receiving mail from me.