Getting Started with ESC101 Lab

ESC101:Fundamentals of Computing
Second semester, 2000-01

Chapter 1

Getting Familiar with your Computer

Each of you will have your own PC (or personal computer) to work on in the ESc101 lab. The PC consists of the following major components. Begin by identifying each one of them.

Monitor This is the most prominent equipment on your desk and looks like a small television.

Keyboard This is the equipment that you use to type your commands etc.

Mouse This is a small rectangular piece of equipment somewhere on the right of the keyboard. Note that it has two or three large buttons on the top. Notice that as you move the mouse, a small arrow like object moves across the screen. This object is called the *pointer*.

CPU Box This is a large upright box that contains the "innards" of the computer.

Using the keyboard

Using the keyboard you can type in small letters, capital letters, digits and some special characters. To type in small letters, just press the key which is marked with the character you want to type in. To type in a capital letter, you have to press the key which is marked with the character you want to type in while pressing the `Shift` key. Then, you can release the `Shift` key. To type in a series of capital letters, you press the `Caps Lock` once. Now, whatever you type will appear in capital letters. Once, you have completed typing capital letters, you should press `Caps Lock` once again, to type small alphabets. Digit keys are marked with both digits and some special characters. For example, the key for the digit `8` is marked with

both `8` and `*`. Normally, pressing this key will print `8`. To type the special character `*` you have to press this key while pressing `Shift` key.

There is a long bar at the last row of the key board. This the key used for typing in a blank space.

There is a key marked `Ctrl` in the key board. Whenever you are told to press `Ctrl c` in this document, it means that you have to press the key marked `c` while pressing the `Ctrl` key. Then you can release both.

In this text, whatever you are going to type in will appear in *slanted* font and whatever the system displays on the screen will appear in `bold` font.

Chapter 2

Getting Started

Switch on the PC if it is not already on (by pressing the appropriate button on the CPU box). A small LED on the box indicates whether the PC is on. Note that if the display is blank, this does not mean that the PC if off. If the display is blank, you can restore it by pressing any key on the keyboard.

Caution *Never switch off the PC, even when you are finished and leaving the lab. Leave it on.*

Logging in

The display on the monitor initially displays a *login prompt*. If you had to switch on the PC, it will take some time to appear. The login prompt is a square box in the middle of the screen containing the following two lines.

```
login:
password:
```

You will be given a login name and a password by the instructor in-charge of the lab.

Type your login name and press the `Enter` key. Note that the cursor moves to the next line. Now type your password and again press the `Enter` key. Note that your password is not displayed on the screen. If you have typed the login name or the password incorrectly, you will see a message saying: `Login incorrect`. Pressing `Enter` again will restore the login prompt, so that you can try once more.

If your login name and password have been accepted as valid, you will see the screen change. There will be several button-like squares at the top or right portion of the screen and a large *window*. You will work mostly inside this window only.

Inside this window, you will find the following text displayed.

```
$
```

This is called the *shell prompt*. In your case, the shell prompt may be a little different, for example: `ccpc141:~$`. This does not matter. From now on, you can type your commands to the system at this prompt.

Changing password

Once you have logged into the system for the first time, you should change your password to whatever you feel like. The password name should have atleast 6 characters. It is better to use some digits or special characters as part of the password.

Caution Make sure though that you can remember your new password, otherwise you will need the system administrator's help in changing it again so that you can log in again. This will waste your precious lab time.

You can change your password using the command `yppasswd`. This command needs to be typed on the shell prompt. Move the pointer inside the command window and click its left button once. Now type `yppasswd` and press `Enter`.

The system will ask you for the old password. If it is correct, the system will ask you to type the new password. It will ask you to retype the new password to ensure that there are no typing errors. If both match, the password will be changed. It will take some time, for the password to get changed. So, don't try out the new password immediately. An example session is given below. The login name of the user invoking the command is assumed to be `edhan`.

```
$yppasswd
Changing YP password for edhan
OLD YP password:
New password:
Retype new password:
The YP password has been changed on agni, the master YP passwd server.
$
```

If the system fails to change the password for some reasons, it displays the message `Password Unchanged`.

Logging out

Once you have completed all your work, you should logout from the system. For this you need to use the mouse. Move the pointer outside all windows.

Now press the leftmost button of the mouse and keep it pressed. A small box will be displayed near the pointer. On each row of the box, some text is written (This is called a *menu*). Bring the pointer (keeping the button pressed) to the last row which has `exit Fvwm95` written in it. As you do this, the row becomes highlighted. Now release the mouse button. Now release the mouse button. When you do this, the previous menu will disappear and a new one will appear in the center of the scree, with three buttons on it. Move the pointer to the one reading `Logout` and click (i.e., press and release) the left mouse button. You will now be logged out of the system and the login prompt will be displayed again.

Chapter 3

Useful Linux commands

General structure of the commands

Once you have logged in, you can use any one of a set of Linux commands. Each command has a different purpose. The structure of the commands is as follows:

- command - This is the name of the command
- options - This is a set of options or flags to the command
- arguments - These are a set of command-line arguments to the command

An example will make things clear.
`$` *ls -l esc101*
In the above example, *ls* is the name of the command, *-l* is the option for the command and *esc101* is the argument for the command. We will see what this command does a little later. The different words of a command (*ls*,*-l* and *esc101* in this example) have to be separated with a blank. The `Enter` key has to be pressed after typing the entire command.

Using on-line help

Every Linux system is supported with an on-line manual for the commands supported by the system. The command used to see the manual is `man`. For example to see the manual page for the command *ls*, type
`$`*man ls*
This will display the manual page of *ls* on your terminal page by page. You can use *spacebar* to move page by page and you can use *Enter* to move line

by line. At any point of time, you can quit by pressing *q*. The format of the man page is as follows:

- Name - gives the name and short description of the command
- Synopsis - gives the name, command line options and arguments of the command.
- Flags - gives a description of the command line options
- Description - Describes how to use the command
- Examples - gives a list of examples on how to use
- Files - describes a set of related files.
- Related info - gives the commands that are related to this.

Files and Directories

On a Linux system, information is stored in what are known as files. You can think of a file as a piece of paper on which you can write something. Each file has a name. Files are grouped into directories. A directory can have any number of files. You can think of a directory as a folder in which papers are kept. The interesting thing is that directories can contain other directories as well. To continue our analogy, this means that folders can contain papers as well as other folders.

The system has a root directory which contains all other directories and files of the system. The name of the root directory is `/` (pronounced slash). The root directory contains several directories which in turn contain other directories and so on.

To take an example, the root directory contains a directory called `users` which contains another directory called `cse` which contains another directory called `fc` which contains another directory called `deepak`. The complete path to this directory would be `/users/cse/fc/deepak`. Note that the different components of the directory name are separated by `/`.

Home directory and Current directory

Each user is given his/her own *home-directory* within which he/she can create files and directories as desired. Further, at any given point in time, a user is working in a particular directory called the current directory. When you log in the current directory is the same as your home directory. You can use the command *pwd* to print the name of the current directory. Thus, if you give this command just after logging in, the system will print the path

to your home directory. You can use the command *cd* (discussed later) to change your current directory.

Useful commands

pwd

This command will tell you what directory you are currently working in.
$*pwd*
`/user/cse/fc/deepak`
`$`

cd

You can move up and down in the file system tree to a subdirectory using the command `cd`.
$*pwd*
`/users/cse/fc/deepak`
$*cd newdir*
$*pwd*
`/users/cse/fc/deepak/newdir`
`$`
You can move up in the file system tree to the parent directory of the current directory by using the special directory name ". ." .
$*pwd*
`/users/cse/fc/deepak/newdir`
$*cd ..*
$*pwd*
`/users/cse/fc/deepak`
`$`

From any directory anywhere in the system, typing just *cd* will bring you to your home directory.

ls

The *ls* command lists the names of files present in the current working directory.
$*ls*
`new.c`
`a.out`
`core`
`$`

The *-l* option provides a long listing of the files which provides more information about each file.

```
$ls -l
-rw-r--r-- 1 deepak csefc    1024 Jul 20 10:25 new.c
-rwxr--r-- 1 deepak csefc   65536 Jul 20 10:30 a.out
-rw-r--r-- 1 deepak csefc 1000234 Jul 20 10:25 core
$
```

The string `-rw-r--r--` gives the permissions of the file new.c. This will be explained later in this document. `deepak` is the login name of the owner of the file. `csefc` is the group to which `deepak` belongs. Next field is the size of the file in characters. Thus the file `new.c` has 1024 characters in it. Next three fields give the time at which the file was last modified. The final field gives the name of the file.

Files whose name begin with the character "`.`" are treated as hidden files, i.e., they will not be seen by using the normal *ls* or *ls -l* commands. To see all files in the directory (including hidden files), you have to use the *-a* option of *ls*. For example,

```
$ls -a
.profile
.elm
new.c
a.out
$
```

File permissions

Now, coming back to the first field in the output of *ls -l* which gives the permissions of the file, there are 10 characters in this field. First character specifies the type of the file. If it is an ordinary file then it will be `-` or if it is a directory it will be `d`. Remaining 9 characters are divided into 3 subfields each having 3 characters. First subfield gives the permissions for the owner, second is for the group to which he belongs and the third is for all others. In each subfield,

- First character can be `r` or `-` depending upon whether read permission is given or not.
- Second character can be `w` or `-` depending upon whether write permission is given or not.
- First character can be `x` or `-` depending upon whether execute permission is given or not.

Therefore the string `-rwxr--r--` specifies that the file is an ordinary file. It specifies that the owner has read, write and execute permissions, the persons in the group of the owner have only read permission and all others have read permission.

chmod

You can change the permissions of your files using the command `chmod`. It has the form:
chmod mode filename
The permissions of any named file are changed according to mode. A mode has the form:
[who] op permission
who is a combination of the letters `u` (for user), `g` (for group) and `o` (for others). The letter `a` stands for `ugo`, the default if *who* is omitted. `op` can be `+` to add permission and `-` to remove permission. *permission* is any combination of the letters `r` (read), `w` (write) and `x` (execute). An example session is given below:

```
$ls -l
-rwxr-xrwx 1 deepak csefc 2048 Jul 20 10:25 junk
$chmod o-w junk
$ls -l
-rwxr-xr-x 1 deepak csefc 2048 Jul 20 10:25 junk
$
```

mkdir

You can create new subdirectories using the command `mkdir`.

```
$ls -l
-rwxr--r-- 1 deepak csefc   65536 Jul 20 10:30 a.out
-rw-r--r-- 1 deepak csefc 1000234 Jul 20 10:25 core
-rw-r--r-- 1 deepak csefc    1024 Jul 20 10:25 new.c
$mkdir newdir
$ls -l
-rwxr--r-- 1 deepak csefc   65536 Jul 20 10:30 a.out
-rw-r--r-- 1 deepak csefc 1000234 Jul 20 10:25 core
-rw-r--r-- 1 deepak csefc    1024 Jul 20 10:25 new.c
drwxr-xr-x 1 deepak csefc    1024 Jul 20 10:25 newdir
$
```

The meaning of the permissions `rwx` in case of a directory are:

- `r` - can read the contents of the directory, i.e., can see what files are present.

- `w` - can write into the directory, i.e., can create files in the directory, can delete files from the directory
- `x` - can search the directory for files, can change to that directory.

cat

This command prints the contents of file(s) on screen. For example,
$*cat new.c*
prints the contents of `new.c` on the screen. You can give more than one file name also.

more

This command is similar to `cat` except that the file contents are displayed page by page. You can quit at any time by pressing `q`. You can scroll page by page using `spacebar` and line by line by pressing `Enter`.

cp

This command is used to copy a file(s) from one location to another. The old file and new file both will remain after the operation. For example,
$*cp old.c new.c*
will create a new file `new.c` identical to `old.c`. The destination can be just a directory name in which case a new copy of the source file is created in that directory with the same name. For example,
$*cp old.c tempdir*
will create a new copy of `old.c` in the directory `tempdir`.

mv

This is same as the command `cp` except that after the operation the source file will not exist. That is, the source file(s) is(are) moved physically to the destination. For example,
$*mv old.c new.c*
will move the file `old.c` to the file `new.c` and will remove the file `old.c`. The result is that the file `old.c` is renamed to `new.c`. Now, consider the example,
$*mv old.c tempdir*
where `tempdir` is the name of a directory. This will move the file `old.c` from the current directory to the directory `tempdir`.

rm

You can delete a file(s) using the command `rm`. An example session is given below:
$*ls*
`new.c`
`a.out`
`core`
$*rm core*
$*ls*
`new.c`
`a.out`
$

rmdir

You can delete one of your subdirectories using the command `rmdir`. For this command to work, the subdirectory should have no files. If there are files, then you have to first delete them using the `rm` command.
$*rmdir newdir*
$*ls*
`new.c`
`a.out`
$
You can delete a subdirectory which is non-empty using the command *rm -r*.
$*rm -r newdir*
$*ls*
`new.c`
`a.out`
$

Chapter 4

Creating and editing files using kedit

You can create a new file and write something into it by using an editor. Linux has several editor programs. Here we will describe the one that is simplest to use: `kedit`.

You start the process of creating a new file (say new.c) by starting the editor as follows.

$*kedit new.c &*

When you type this command (followed by pressing the `Enter` key as usual) you will observe that you immediately get the command prompt back. The editor starts up in a new window of its own. You can also use this command to modify an existing file. The editor window has some menu buttons on top (such as `File`, `Edit`, etc.). Below these are some some small figures (called icons). Below this row of icons is a large blank area. This is where you will type whatever you want to go into the file.

Move the pointer to the blank area and click the left mouse button once. You are now ready to create your file. You can now type whatever you wish in this area.

Making Changes

If you make some errors while typing whatever should go into the file that you are creating or modifying, they can be corrected quite easily. Note that the text area of the kedit window has a small blinking vertical line (in addition to the pointer). This line is called the cursor. Whatever you type gets entered at the current position of the cursor. The cursor can be moved easily by using the arrow keys on the keyboard. You can also use the mouse to position the cursor whereever you want in the text area. This is achieved

by moving the pointer to where you want the cursor to be and then clicking the left mouse button. If you press the `Backspace` key, the character to the left of the cursor gets erased or deleted. If you press the `Delete` key, the character to the right of the cursor gets deleted.

Saving the File

Whatever you have typed so far or whatever changes you have made to the file so far have not yet been written to the file. They will get written to the file only once you save your work. In kedit, you do this by pressing `Ctrl-S`.

Quitting Kedit

To quit `kedit`, simply press `Ctrl-Q`. If you have not saved your work since the last change that you made, kedit will give you another chance to save it (discover for yourself how this will happen).

Kedit has several other features that can make the task of editing files easier. You can learn some of these as you go along and get more familiar with the system. What is described here is sufficient to get you started.

Chapter 5

The vi editor

`vi` is a very popular editor in the Linux and in the Unix environment. This editor originated from the work of a graduate student in the University of California, Berkeley. This editor can do many things with very few keystrokes. Here, we discuss some elementary aspects of the `vi` editor; for more advanced material you can use the help command while using the editor.

5.1 Three modes

There are three modes in which the editor works. These are:

- input mode: Every key that is depressed in this mode is entered as text, and is displayed.
- command mode: Here, keys are used as commands to act on text.
- `ex` mode: In this mode commands are entered in the last line of the screen.

A `vi` session begins by invoking the command `vi` usually with a filename. At this point, the editor is in the command mode. We go from the command mode to insert mode using any of the insert, append, replace or substitute commands. Using the escape key, the editor comes back to the command mode from the input mode.

`ex` mode is entered when : is keyed-in in the command mode, and the transition to the command mode from the `ex` mode is made on the enter key.

5.2 Some command mode commands

Commands to navigate:
h (or backspace) moves the cursor left by one character
l (or spacebar) moves the cursor right by one character
k moves the cursor one line up
j moves the cursor one line down
b moves cursor to the beginning of the current word
e moves cursor to the end of the current word
0 moves cursor to the beginning of the current line
$ moves cursor to the end of the current line
w moves the cursor to the beginning of the next word
Ctrl-f scrolls full page forward
Ctrl-b scrolls full page backward
G moves the cursor to the last line of the file
nG moves the cursor to the n th line of the file

Commands to delete:
x deletes the character at which the cursor is there
dd deletes the current line
(Both of these commands can take a numerical parameter preceding the command– for example 2dd will delete two lines, the current and the next line).

Some insertion commands:
i insert text starting at the left of the current cursor position. Whatever is typed gets inserted till the escape key is keyed in.
a insert text starting from the right of the current cursor position.
o open a new line below the current line
O open a new line above the current line

Commands for pattern searching:
/*pat* seraches forward for pattern *pat*
?*pat* seraches backward for pattern *pat*
n repeats search in the direction of previous search

5.3 Some ex mode commands

:w save the file by writing it into disc, continue editing
:wq write the file and quit editing
:q! quit editing without saving the changes made in the current edit session
:r *filename* read in the file named *filename* below the current line
:n1,n2d delete all the lines from line number n1 to n2, both n1 and n2 inclusive
:n1,n2 w *filename* write lines from n1 to n2 (both inclusive) as a file called *filename*

Chapter 6

Compiling and Executing Programs

C programs are compiled using the command `cc`. It expects that the file name has the extension `.c` . The default name for the output file created is `a.out` .

```
$cc myfile.c
$ls
myfile.c
a.out
$
```

However, you can give a name for the output executable file using the option `-o` of the command `cc`.

```
$cc myfile.c -o myfile
$ls
myfile.c
myfile
$
```

To execute the program just type the name of the executable file(say *a.out*) and press *Enter*.

If there is some logic error in your program, it may get caught in an infinite loop. You wait for some time. If it is not terminating, press *CTRL c* to get out of it. There may be some run-time errors, which cause your program to abort in between its execution. This will create a memory image of your program in a file `core`. It will display the message `memory fault - core dumped`. Do not bother about this. This means that there is some error in your program. You will later see how to use this `core` file to know what actually has happened during execution.

At times, instead of getting the output on the screen, you may want to store it in a file. For this, you have to use the shell indirection operator `>`. You have to specify an output file name also. An example will make things clear here:

$*a.out* `>` *outfile*
$*ls*
`new.c`
`a.out`
`outfile`
$
Here, the output will be stored in a new file `outfile`.

Sometimes, you may want to give input from a file, instead of giving from a keyboard. For this, you have to use the shell indirection operator `<`. You have to specify the file name from which to take the input also.
$*a.out* `<` *infile*
Here, the input is taken from the file `infile` and output is displayed on the screen. You can use both `<` and `>` together also.

Chapter 7

Netscape

Netscape is a browser used to access information available on the world wide web. You will have to use this program to access your assignment problems for the laboratory sessions.

Starting Netscape

Remember the set of large buttons on the top or left of your display. One of these is the button for Netscape. To start Netscape, just move the pointer to this button and click the left mouse button.

Netscape will create its own window.

Using Netscape

Any piece of information available on the web has a *URL*. To access this information, type the URL in the "location box" in the Netscape window and press `Enter`. The URL for the information on this course is: `http://www.iitk.ac.in/esc101`

Type this URL in the location box of Netscape and press `Enter`. You will see the *home page* for the course appearing in the main window of Netscape. On this page, you will find several pieces of information about this course. You will also find that some lines are underlined. These are called *links* to other documents. Bringing the pointer to one of these underlined texts and clicking the left mouse button causes Netscape to display the other document.

Play around with Netscape till you are comfortable with it.