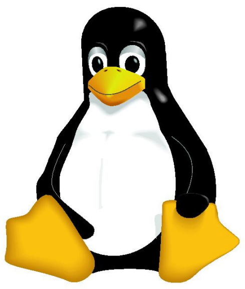


Linux Tutorial[1]

Prepared by

Mohamed Elmahdy

[1] This Linux tutorial is a modified version of the tutorial “A beginners guide to the Unix and Linux operating system” by M.Stonebank. (surrey.ac.uk) under Creative Commons licence.

Contents

UNIX Introduction

What is UNIX?

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available.

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X.

The UNIX operating system

The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The Kernel: The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the filestore and communications in response to system calls.

The shell: acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt.

Files and processes

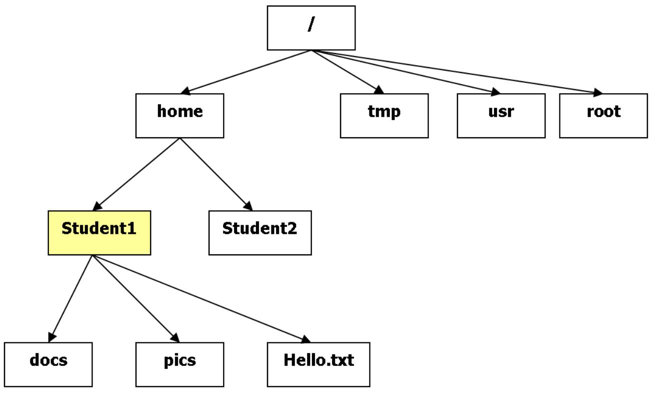
Everything in UNIX is either a file or a process.

A process is an executing program identified by a unique PID (process identifier).

A file is a collection of data. They are created by users using text editors, running compilers etc.

The Directory Structure

All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called **root** (written as a slash /)

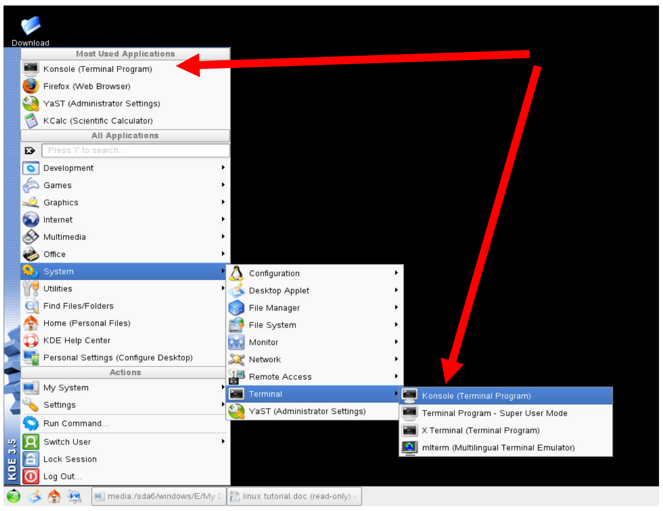


In the diagram above, we see that the home directory of the user **"Student1"** contains two sub-directories (**docs** and **pics**) and a file called **report.doc**.

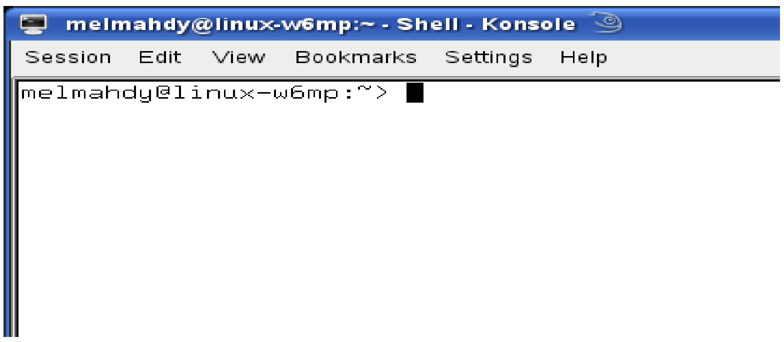
The full path to the file **report.doc** is **"/home/Student1/report.doc"**

Starting an UNIX terminal

To open an UNIX terminal window, click on the "Konsole (Terminal Program)" icon from menus as shown in the figure.



An UNIX Terminal window will then appear with a prompt, waiting for you to start entering commands.



```
melmahdy@linux-w6mp:~> 
```

UNIX Tutorial One

1.1 Listing files and directories

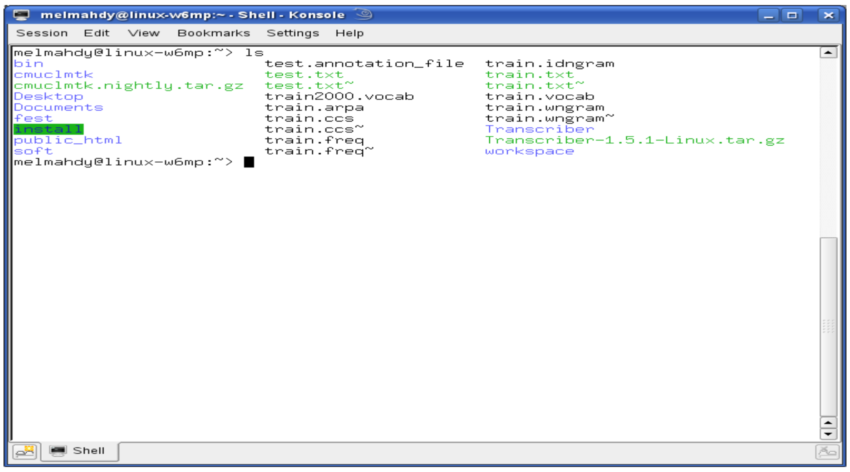
ls (list)

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, for example, **ee91ab**, and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type **ls then press the Enter button.**

```
ls
```

The **ls** command (lowercase L and lowercase S) lists the contents of your current working directory.

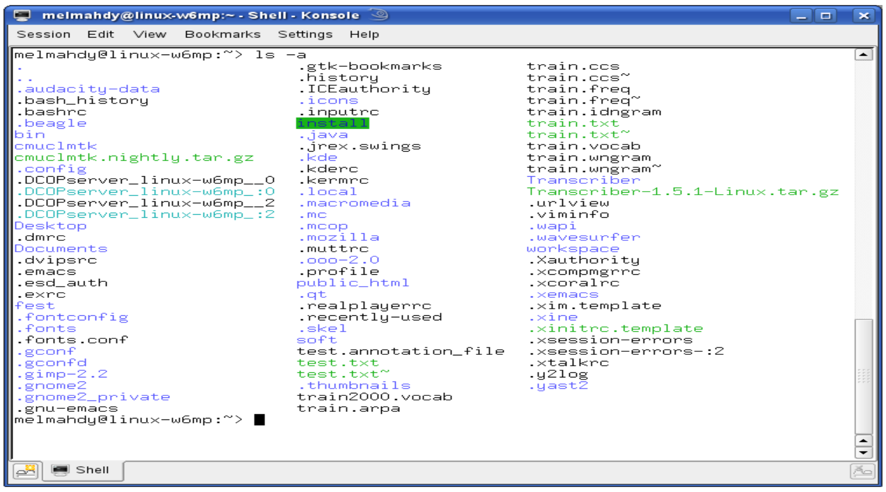

melmahdy@linux-w6mp:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

```
melmahdy@linux-w6mp:~> ls
bin                      test.annotation_file   train.idngram
cmuclmtk                 test.txt               train.txt
cmuclmtk.nightly.tar.gz  test.txt~              train.txt~
Desktop                  train2000.vocab        train.vocab
Documents                train.arpa             train.wngram
fest                     train.ccs              train.wngram~
install                  train.ccs~             Transcriber
public_html              train.freq             Transcriber-1.5.1-Linux.tar.gz
soft                     train.freq~            workspace
melmahdy@linux-w6mp:~> █
```

ls does not, in fact, show hidden files. To list all files in your home directory including hidden, type

```
ls -a
```

As you can see, **ls -a** lists files that are normally hidden.


melmahdy@linux-w6mp:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell

```
melmahdy@linux-w6mp:~> ls -a
.                          .gtk-bookmarks        train.ccs
..                         .history              train.ccs~
.audacity-data             .ICEauthority         train.freq
.bash_history              .icons                train.freq~
.bashrc                    .inputrc              train.idngram
.beagle                    install               train.txt
bin                        .java                 train.txt~
cmuclmtk                   .jrex.swings          train.vocab
cmuclmtk.nightly.tar.gz    .kde                  train.wngram
.config                    .kderc                train.wngram~
.DCOPserver_linux-w6mp__0  .kermrc               Transcriber
.DCOPserver_linux-w6mp_:0  .local                Transcriber-1.5.1-Linux.tar.gz
.DCOPserver_linux-w6mp__2  .macromedia           .urlview
.DCOPserver_linux-w6mp_:2  .mc                   .viminfo
Desktop                    .mcop                 .wapi
.dmrc                      .mozilla              .wavesurfer
Documents                  .muttrc               workspace
.dvipsrc                   .ooo-2.0              .Xauthority
.emacs                     .profile              .xcompmgrrc
.esd_auth                  public_html           .xcoralrc
.exrc                      .qt                   .xemacs
fest                       .realplayerrc         .xim.template
.fontconfig                .recently-used        .xine
.fonts                     .skel                 .xinitrc.template
.fonts.conf                soft                  .xsession-errors
.gconf                     test.annotation_file  .xsession-errors-:2
.gconfd                    test.txt              .xtalkrc
.gimp-2.2                  test.txt~             .y2log
.gnome2                    .thumbnails           .yast2
.gnome2_private            train2000.vocab
.gnu-emacs                 train.arpa
melmahdy@linux-w6mp:~> █
```

ls is an example of a command which can take options: **-a** is an example of an option. The options change the behaviour of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behaviour of the command. (See later in this tutorial)

1.2 Making Directories

mkdir (make directory)

We will now make a subdirectory in your home directory to hold the files you will be creating and using in this tutorial. To make a subdirectory called unixstuff in your current working directory type

```
mkdir unixstuff
```

To see the directory you have just created, type

```
ls
```

1.3 Changing to a different directory

cd (change directory)

The command **cd *directory*** means change the current working directory to *'directory'*. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To change to the directory you have just made, type

```
cd unixstuff
```

Type **ls** to see the contents (which should be empty)

Exercise 1a

Make another directory inside the **unixstuff** directory called **backups**

1.4 The directories . and ..

Still in the **unixstuff** directory, type

```
ls -a
```

As you can see, in the **unixstuff** directory (and in all other directories), there are two special directories called (.) and (..)

The current directory (.)

In UNIX, (.) means the current directory, so typing

```
cd .
```

NOTE: there is a space between cd and the dot

means stay where you are (the **unixstuff** directory).

This may not seem very useful at first, but using (.) as the name of the current directory will save a lot of typing, as we shall see later in the tutorial.

The parent directory (..)

(..) means the parent of the current directory, so typing

```
cd ..
```

will take you one directory up the hierarchy (back to your home directory). Try it now.

Note: typing **cd** with no argument always returns you to your home directory. This is very useful if you are lost in the file system.

1.5 Pathnames

pwd (print working directory)

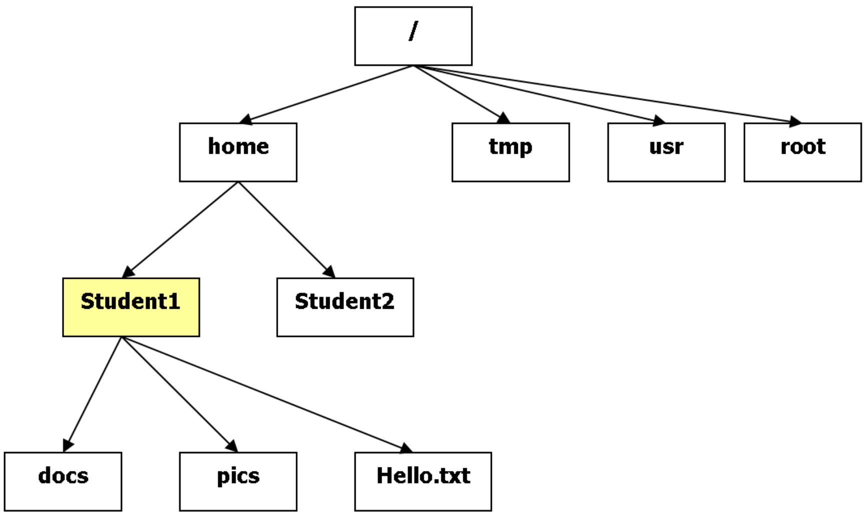
Pathnames enable you to work out where you are in relation to the whole file-system. For example, to find out the absolute pathname of your home-directory, type cd to get back to your home-directory and then type

```
pwd
```

The full pathname will look something like this -

```
/home/student1/
```

which means that **student1** (your home directory) is in the **home** sub-directory, which is in the top-level root directory called " / ".



Exercise 1b

Use the commands **cd**, **ls** and **pwd** to explore the file system.

(Remember, if you get lost, type **cd** by itself to return to your home-directory)

1.6 More about home directories and pathnames

Understanding pathnames

First type cd to get back to your home-directory, then type

```
ls unixstuff
```

to list the conents of your unixstuff directory.

Now type

```
ls backups
```

You will get a message like this -

backups: No such file or directory

The reason is, **backups** is not in your current working directory. To use a command on a file (or directory) not in the current working directory (the directory you are currently in), you must either **cd** to the correct directory, or specify its full pathname. To list the contents of your backups directory, you must type

```
ls unixstuff/backups
```

~ (your home directory)

Home directories can also be referred to by the tilde ~ character. It can be used to specify paths starting at your home directory. So typing

```
ls ~/unixstuff
```

will list the contents of your unixstuff directory, no matter where you currently are in the file system.

What do you think

```
ls ~
```

would list?

What do you think

```
ls ~/..
```

would list?

Using the “Tab” key for auto-completion

When you press on the “Tab” key from the keyboard, the shell try to automatically complete command names and file names.

Now, change directory to your home by typing

```
cd
```

Now type **ls unix** and then press on the “Tab” button. what happened ?

Summary

Command	Meaning
ls	list files and directories
ls -a	list all files and directories
mkdir	make a directory
cd *directory*	change to named directory
cd	change to home-directory
cd ~	change to home-directory
cd ..	change to parent directory
pwd	display the path of the current directory

UNIX Tutorial Two

2.1 Copying Files

cp (copy)

cp *file1 file2* is the command which makes a copy of **file1** in the current working directory and calls it **file2**

What we are going to do now, is to take a file stored in an open access area of the file system, and use the **cp** command to copy it to your unixstuff directory.

First, **cd** to your **unixstuff** directory.

```
cd ~/unixstuff
```

Then at the UNIX prompt, type,

```
cp /NetLab/hello.txt .
```

Note: Don't forget the dot **.** at the end. Remember, in UNIX, the dot means the current directory.

The above command means copy the file hello**.txt** to the current directory, keeping the name the same.

Exercise 2a

Create a backup of your **hello.txt** file by copying it to a file called **hello.bak**

2.2 Copying folders

cp folder1 folder2 -r is the command which makes a copy of ***folder1*** and past it to ***folder2, -r*** is needed for recursive copy (copy all the files in the folder)

Now we will take a copy of the **unixstuff** folder to **unixstuff_copy**, type:

```
cp ~/unixstuff ~/unixstuff_copy -r
```

2.3 Moving files

mv (move)

mv *file1 file2* moves (or renames) **file1** to **file2**

To move a file from one place to another, use the **mv** command. This has the effect of moving rather than copying the file, so you end up with only one file rather than two.

It can also be used to rename a file, by moving the file to the same directory, but giving it a different name.

We are now going to move the file **hello.bak** to your backup directory.

First, change directories to your **unixstuff** directory (can you remember how?). Then, inside the **unixstuff** directory, type

```
mv hello.bak backups/.
```

Type **ls** and **ls backups** to see if it has worked.

Note: to move an entire folder with all the sub-folders and files, you have to use the **-r** option as used in copying folder.

2.4 Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the rm command. As an example, we are going to create a copy of the **hello.txt** file then delete it.

Inside your **unixstuff** directory, type

```
cp hello.txt tempfile.txt
ls
rm tempfile.txt
ls
```

You can use the **rmdir** command to remove a directory (make sure it is empty first). Try to remove the **backups** directory. You will not be able to since UNIX will not let you remove a non-empty directory.

Note: to remove an entire folder with all the sub-folders and files, you have to use the **-r** option as used in copying folder.

Exercise 2b

Create a directory called **tempstuff** using mkdir , then remove it using the rmdir command.

2.5 Displaying the contents of a file on the screen

clear (clear screen)

Before you start the next section, you may like to clear the terminal window of the previous commands so the output of the following commands can be clearly understood.

At the prompt, type

```
clear
```

This will clear all text and leave you with the prompt at the top of the window.

cat (concatenate)

The command cat can be used to display the contents of a file on the screen. Type:

```
cat hello.txt
```

As you can see, the file is longer than than the size of the window, so it scrolls past making it unreadable.

Summary

Command	Meaning
cp *file1 file2*	copy file1 and call it file2
mv *file1 file2*	move or rename file1 to file2
rm *file*	remove a file
rmdir *directory*	remove a directory
cat *file*	display a file

UNIX Tutorial Three

3.1 Redirection

Most processes initiated by UNIX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

We have already seen one use of the cat command to write the contents of a file to the screen.

Now type cat without specifing a file to read

```
cat
```

Then type a few words on the keyboard and press the [**Return**] key.

Finally hold the [**Ctrl**] key down and press [**d**] (written as **^D** for short) to end the input.

What has happened?

If you run the cat command without specifing a file to read, it reads the standard input (the keyboard), and on receiving the 'end of file' (**^D**), copies it to the standard output (the screen).

In UNIX, we can redirect both the input and the output of commands.

3.2 Redirecting the Output

We use the > symbol to redirect the output of a command. For example, to create a file called **list1** containing a list of fruit, type

```
cat > list1
```

Then type in the names of some fruit. Press [**Return**] after each one.

```
pear
banana
apple
^D {this means press [Ctrl] and [d] to stop}
```

What happens is the cat command reads the standard input (the keyboard) and the > redirects the output, which normally goes to the screen, into a file called **list1**

To read the contents of the file, type

```
cat list1
```

Exercise 3a

Using the above method, create another file called **list2** containing the following fruit: orange, plum, mango, grapefruit. Read the contents of **list2**

3.3 Redirecting the Input

We use the **<** symbol to redirect the input of a command.

The command sort alphabetically or numerically sorts a list. Type

```
sort
```

Then type in the names of some animals. Press [Return] after each one.

```
dog
cat
bird
ape
^D (control d to stop)
```

The output will be

```
ape
bird
cat
dog
```

Using < you can redirect the input to come from a file rather than the keyboard. For example, to sort the list of fruit, type

```
sort < list1
```

and the sorted list will be output to the screen.

To output the sorted list to a file, type,

```
sort < list1 > slist
```

Use cat to read the contents of the file **slist**

Summary

Command	Meaning
command > *file*	redirect standard output to a file
command < *file*	redirect standard input from a file
sort	sort data

UNIX Tutorial Four

4.1 Wildcards

The * wildcard

The character ***** is called a wildcard, and will match against n**one or more character(s)** in a file (or directory) name. For example, in your **unixstuff** directory, type

```
ls list*
```

This will list all files in the current directory starting with **list....**

Try typing

```
ls *list
```

This will list all files in the current directory ending with **....list**

The ? wildcard

The character ? will match exactly one character.
So **?ouse** will match files like **house** and **mouse**, but not **grouse**.
Try typing

```
ls ?list

ls list?
```

4.2 Filename conventions

We should note here that a directory is merely a special type of file. So the rules and conventions for naming files apply also to directories.

In naming files, characters with special meanings such as **/ * & %** , should be avoided. Also, avoid using spaces within names. The safest way to name a file is to use only alphanumeric characters, that is, letters and numbers, together with _ (underscore) and . (dot).

Good filenames	Bad filenames
project.txt	project
my_big_program.c	my big program.c
fred_dave.doc	fred & dave.doc

File names conventionally start with a lower-case letter, and may end with a dot followed by a group of letters indicating the contents of the file. For example, all files consisting of C code may be named with the ending **.c**, for example, **prog1.c** . Then in order to list all files containing C code in your home directory, you need only type ls *.c in that directory.

4.3 Getting Help

On-line Manuals

There are on-line manuals which gives information about most commands. The manual pages tell you which options a particular command can take, and how each option modifies the behavior of the command. Type **man *command*** to read the manual page for a particular command.

For example, to find out more about the **cp** (copy) command, type

```
man cp
```

Alternatively

```
whatis wc
```

gives a one-line description of the command, but omits any information about options etc.

Apropos

When you are not sure of the exact name of a command,

```
apropos keyword
```

will give you the commands with keyword in their manual page header. For example, try typing

```
apropos copy
```

Summary

Command	Meaning
*	match any number of characters
?	match one character
man *command*	read the online manual page for a command
whatis *command*	brief description of a command
apropos *keyword*	match commands with keyword in their man pages

UNIX Tutorial Five

Other useful UNIX commands

gzip

This reduces the size of a file, thus freeing valuable disk space. For example, type

```
ls -l hello.txt
```

and note the size of the file using ls -l . Then to compress **hello.txt**, type

```
gzip hello.txt
```

This will compress the file and place it in a file called **hello.txt.gz**

To see the change in size, type ls -l again.

To expand the file, use the gunzip command.

```
gunzip hello.txt.gz
```

file

file classifies the named files according to the type of data they contain, for example ascii (text), pictures, compressed data, etc.. To report on all files in your home directory, type

```
file *
```

UNIX Tutorial Six

Useful programs

Ksnapshot

Ksnapshot is a useful tool used to capture the screen and save it as an image file. From the shell type:

```
ksnapshot &
```

Exercise 6a

Try to capture the entire screen and save it as an image file on your desktop.

Try to capture only the shell window and save it as an image file.

OpenOffice Writer

Openoffice writer is very similar to Microsoft word (but it is for free). From the shell type **oowriter** and play around. Try to include the captured images from the previous exercise in the openoffice writer document and save the document on your desktop.

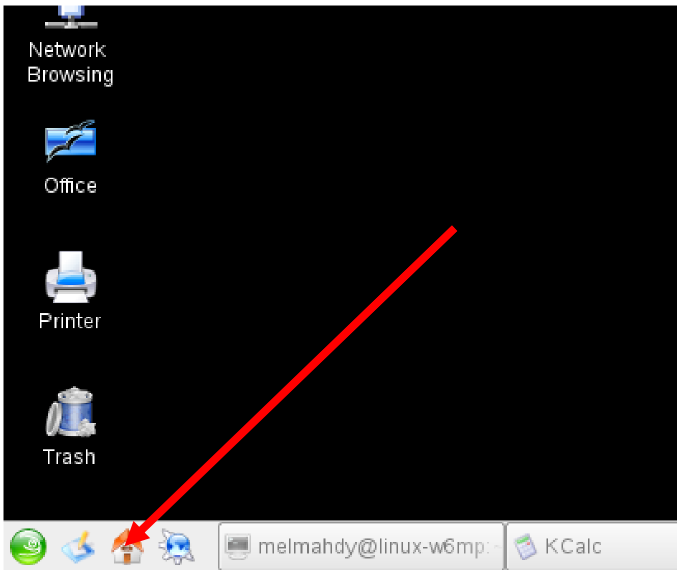
```
oowriter &
```

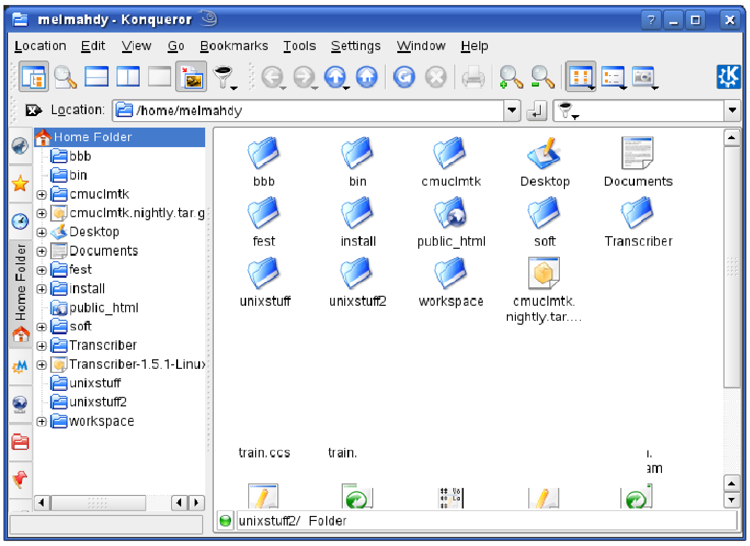
Kwrite

Kwrite is a very useful text editor. From the shell type **kwrite** and play around. Try to copy the text shown on the shell window and paste it in kwrite. In Unix, you can just select the text that you would like to copy with your mouse, and to paste it, just click on the middle mouse button.

Exploring your PC using Konqueror

Click on the "Home" icon from the KDE bar as shown on the following image. Konqueror will be launched in your home folder. Try to explore your PC using Konqueror, and try to open your files.





Recommended Readings

1. "Ubuntu Pocket Guide and Reference", Keir Thomas, http://www.ubuntupocketguide.com/download_main.html

.