

Operating Systems Laboratory

Lab 1 Introduction to Linux

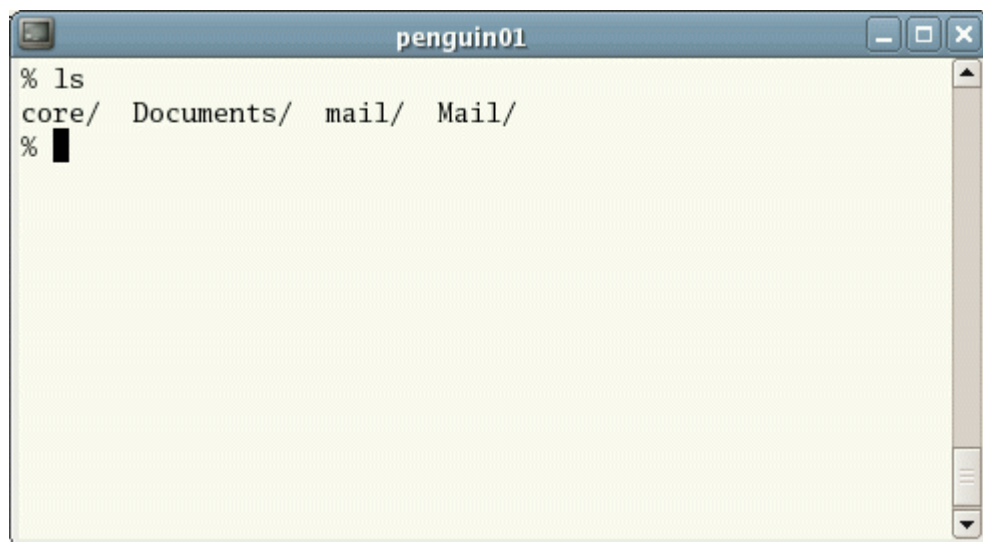
1 Listing Files and Directories

ls (list)

To find out what is in your current directory, type

```
% ls
```

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



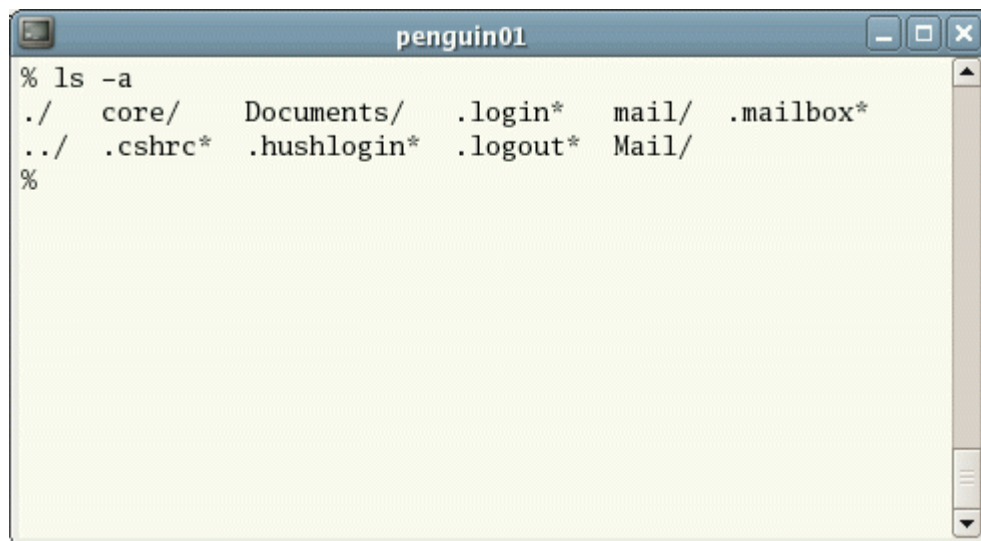
There may be no files visible in your home directory, in which case, the UNIX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account was created.

ls does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as hidden files and usually contain important program configuration information. They are hidden because you should not change them unless you are very familiar with UNIX!!!

To list all files in your home directory including those whose names begin with a dot, type

```
% ls -a
```

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

A terminal window titled "penguin01" with standard window controls (minimize, maximize, close) in the top right corner. The terminal has a light yellow background and shows the command "% ls -a" at the prompt. The output is displayed in two lines: the first line contains "./", "core/", "Documents/", ".login*", "mail/", and ".mailbox*"; the second line contains "../", ".cshrc*", ".hushlogin*", ".logout*", and "Mail/". The prompt "%" appears on a third line.

```
% ls -a
./  core/  Documents/  .login*  mail/  .mailbox*
../  .cshrc*  .hushlogin*  .logout*  Mail/
%
```

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 the course of 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
```

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

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

4. The directories . 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.

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/its/ug1/ee51vn
```

which means that **ee51vn** (your home directory) is in the sub-directory **ug1** (the group directory), which in turn is located in the **its** sub-directory, which is in the **home** sub-directory, which is in the top-level root directory called **" / "**.

6. More About Pathnames

Understanding pathnames

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

```
% ls unixstuff
```

to list the contents 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?

7. Copying Files (copies of science.txt will be distributed during the lab – place science.txt in the unixstuff folder)

cp (copy)

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

```
% cd ~/unixstuff/
```

Then at the UNIX prompt, type,

```
% cp science.txt science2.txt
```

8 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 science.bak to your backup directory.

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

```
% mv science.txt backups/.
```

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

9 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 **science.txt** file then delete it.

Inside your **unixstuff** directory, type

```
% cp science.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.

Exercise

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

10 Displaying the content 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 science.txt
```

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

less

The command less writes the contents of a file onto the screen a page at a time. Type

```
% less science.txt
```

Press the [**space-bar**] if you want to see another page, and type [**q**] if you want to quit reading. As you can see, **less** is used in preference to **cat** for long files.

head

The **head** command writes the first ten lines of a file to the screen.

First clear the screen then type

```
% head science.txt
```

Then type

```
% head -5 science.txt
```

What difference did the -5 do to the head command?

tail

The **tail** command writes the last ten lines of a file to the screen.

Clear the screen and type

```
% tail science.txt
```

Q. How can you view the last 15 lines of the file?

11. Searching the content of a file

Simple searching using less

Using **less**, you can search through a text file for a keyword (pattern). For example, to search through **science.txt** for the word '**science**', type

```
% less science.txt
```

then, still in **less**, type a forward slash [/] followed by the word to search

```
/science
```

As you can see, **less** finds and highlights the keyword. Type [n] to search for the next occurrence of the word.

grep (don't ask why it is called grep)

grep is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type

```
% grep science science.txt
```

As you can see, **grep** has printed out each line containing the word **science**.

Or has it ????

Try typing

```
% grep Science science.txt
```

The **grep** command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the **-i** option, i.e. type

```
% grep -i science science.txt
```

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for spinning top, type

```
% grep -i 'spinning top' science.txt
```

Some of the other options of **grep** are:

- v** display those lines that do NOT match
- n** precede each matching line with the line number
- c** print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time. For example, the number of lines without the words science or Science is

```
% grep -ivc science science.txt
```

wc (word count)

A handy little utility is the **wc** command, short for word count. To do a word count on **science.txt**, type

```
% wc -w science.txt
```

To find out how many lines the file has, type

```
% wc -l science.txt
```

12. 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 specifying 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 specifying 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.

12. 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

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

Appending to a file

The form >> appends standard output to a file. So to add more items to the file **list1**, type

```
% cat >> list1
```

Then type in the names of more fruit

```
peach  
grape  
orange  
^D (Control D to stop)
```

To read the contents of the file, type

```
% cat list1
```

You should now have two files. One contains six fruit, the other contains four fruit.

We will now use the `cat` command to join (concatenate) **list1** and **list2** into a new file called **biglist**. Type

```
% cat list1 list2 > biglist
```

What this is doing is reading the contents of **list1** and **list2** in turn, then outputting the text to the file **biglist**

To read the contents of the new file, type

```
% cat biglist
```

13. 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 < biglist
```

and the sorted list will be output to the screen.

To output the sorted list to a file, type,

```
% sort < biglist > slist
```

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

14. Pipes

To see who is on the system with you, type

```
% who
```

One method to get a sorted list of names is to type,

```
% who > names.txt  
% sort < names.txt
```

This is a bit slow and you have to remember to remove the temporary file called names when you have finished. What you really want to do is connect the output of the who command directly to the input of the sort command. This is exactly what pipes do. The symbol for a pipe is the vertical bar |

For example, typing

```
% who | sort
```

will give the same result as above, but quicker and cleaner.

To find out how many users are logged on, type

```
% who | wc -l
```

15. Wildcards

The * wildcard

The character ***** is called a wildcard, and will match against none 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

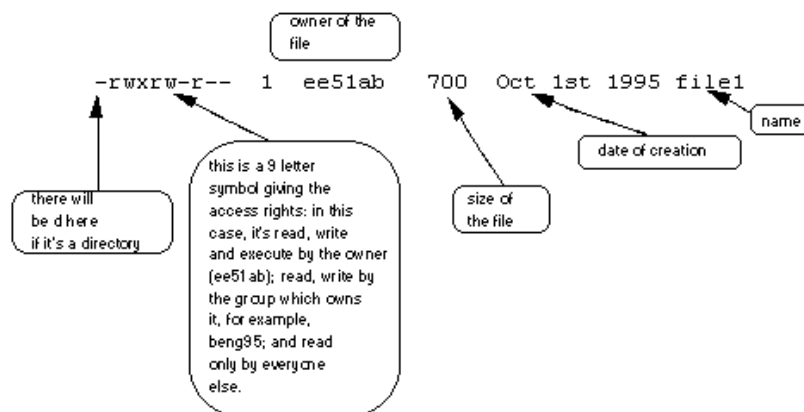
```
% ls ?list
```

16. File system security

In your **unixstuff** directory, type

```
% ls -l (l for long listing!)
```

You will see that you now get lots of details about the contents of your directory, similar to the example below.



Each file (and directory) has associated access rights, which may be found by typing `ls -l`. Also, `ls -lg` gives additional information as to which group owns the file (beng95 in the following example):

```
-rwxrw-r-- 1 ee51ab beng95 2450 Sept29 11:52 file1
```

In the left-hand column is a 10 symbol string consisting of the symbols d, r, w, x, -, and, occasionally, s or S. If d is present, it will be at the left hand end of the string, and indicates a directory: otherwise - will be the starting symbol of the string.

The 9 remaining symbols indicate the permissions, or access rights, and are taken as three groups of 3.

- The left group of 3 gives the file permissions for the user that owns the file (or directory) (ee51ab in the above example);
- the middle group gives the permissions for the group of people to whom the file (or directory) belongs (eebeng95 in the above example);
- the rightmost group gives the permissions for all others.

The symbols r, w, etc., have slightly different meanings depending on whether they refer to a simple file or to a directory.

Access rights on files.

- r (or -), indicates read permission (or otherwise), that is, the presence or absence of permission to read and copy the file
- w (or -), indicates write permission (or otherwise), that is, the permission (or otherwise) to change a file
- x (or -), indicates execution permission (or otherwise), that is, the permission to execute a file, where appropriate

17. Changing Access Rights

chmod (changing a file mode)

Only the owner of a file can use chmod to change the permissions of a file. The options of chmod are as follows

Symbol	Meaning
--------	---------

u	user
g	group
o	other
a	all
r	read
w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

For example, to remove read write and execute permissions on the file **biglist** for the group and others, type

```
% chmod go-rwx biglist
```

This will leave the other permissions unaffected.

To give read and write permissions on the file **biglist** to all,

```
% chmod a+rw biglist
```

18. Processes and Jobs

A process is an executing program identified by a unique PID (process identifier). To see information about your processes, with their associated PID and status, type

```
% ps
```

A process may be in the foreground, in the background, or be suspended. In general the shell does not return the UNIX prompt until the current process has finished executing.

Some processes take a long time to run and hold up the terminal. Backgrounding a long process has the effect that the UNIX prompt is returned immediately, and other tasks can be carried out while the original process continues executing.

Running background processes

To background a process, type an **&** at the end of the command line. For example, the command **sleep** waits a given number of seconds before continuing. Type

```
% sleep 10
```

This will wait 10 seconds before returning the command prompt %. Until the command prompt is returned, you can do nothing except wait.

To run sleep in the background, type

```
% sleep 10 &
```

```
[1] 6259
```

The **&** runs the job in the background and returns the prompt straight away, allowing you to run other programs while waiting for that one to finish.

The first line in the above example is typed in by the user; the next line, indicating job number and PID, is returned by the machine. The user is notified of a job number (numbered from 1) enclosed in square brackets, together with a PID and is notified when a background process is finished. Backgrounding is useful for jobs which will take a long time to complete.

Backgrounding a current foreground process

At the prompt, type

```
% sleep 1000
```

You can suspend the process running in the foreground by typing **^Z**, i.e. hold down the [**Ctrl**] key and type [**z**]. Then to put it in the background, type

```
% bg
```

Note: do not background programs that require user interaction e.g. vi

Listing suspended and background processes

When a process is running, backgrounded or suspended, it will be entered onto a list along with a job number. To examine this list, type

```
% jobs
```

An example of a job list could be

```
[1] Suspended sleep 1000  
[2] Running netscape  
[3] Running matlab
```

To restart (foreground) a suspended processes, type

```
% fg %jobnumber
```

For example, to restart sleep 1000, type

```
% fg %1
```

Typing **fg** with no job number foregrounds the last suspended process.

Killing a process

kill (terminate or signal a process)

It is sometimes necessary to kill a process (for example, when an executing program is in an infinite loop)

To kill a job running in the foreground, type **^C** (control c). For example, run

```
% sleep 100  
^C
```

To kill a suspended or background process, type

```
% kill %jobnumber
```

For example, run

```
% sleep 100 &  
% jobs
```

If it is job number 4, type

```
% kill %4
```

To check whether this has worked, examine the job list again to see if the process has been removed.

ps (process status)

Alternatively, processes can be killed by finding their process numbers (PIDs) and using `kill PID_number`

```
% sleep 1000 &  
% ps  
  
PID TT S TIME COMMAND  
20077 pts/5 S 0:05 sleep 1000  
21563 pts/5 T 0:00 netscape  
21873 pts/5 S 0:25 nedit
```

To kill off the process **sleep 1000**, type

```
% kill 20077
```

and then type **ps** again to see if it has been removed from the list.

If a process refuses to be killed, uses the **-9** option, i.e. type

```
% kill -9 20077
```

Note: It is not possible to kill off other users' processes !!!