

# UNIX Operating System

## Introduction to UNIX Operating System

- ✓ UNIX is a powerful Operating System initially developed by Ken Thompson, Dennis Ritchie at AT&T Bell laboratories in 1970.
- ✓ It is a stable, multi-user, multi-tasking system for servers, desktops and laptops. Because of these most appreciative features, it is prevalent among scientific, engineering, and academic institutions.
- ✓ UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment.
- ✓ 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. Linux in its turn is packaged in a form known as a Linux distribution.
- ✓ In UNIX, the file system is a hierarchical structure of files and directories where users can store and retrieve information using the files.

## The structure of 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.

As an illustration of the way that the shell and the kernel work together, suppose a user types **rm myfile** (which has the effect of removing the file **myfile**). The shell searches the filestore for the file containing the program **rm**, and then requests the kernel, through system calls, to execute the program **rm** on **myfile**. When the process **rm myfile** has

finished running, the shell then returns the UNIX prompt to the user, indicating that it is waiting for further commands.

## **The shell**

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 on our systems.

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

## **Files and processes**

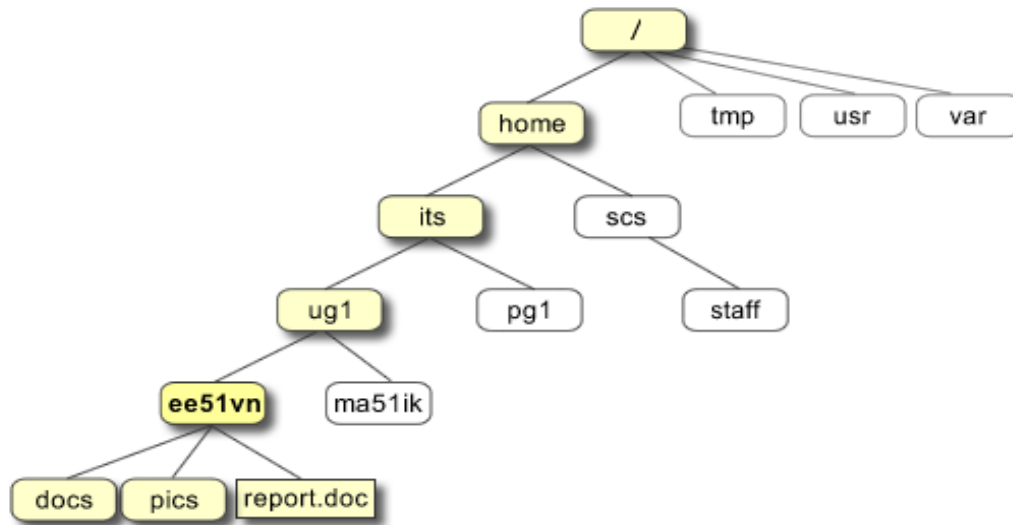
- ✓ Everything in UNIX is either a file or a process.
- ✓ A process is an executing program identified by a unique PID (process identifier).
- ✓ A file is a collection of data. They are created by users using text editors, running compilers etc.

### **Examples of files:**

- A document (report, essay etc.)
- The text of a program written in some high-level programming language
- Instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- A directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

## 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 slash / )



In the diagram above, we see that the home directory of the undergraduate student "ee51vn" contains two sub-directories (docs and pics) and a file called report.doc.

The full path to the file report.doc is **`"/home/its/ug1/ee51vn/report.doc"`**

# Basic UNIX commands:

## Listing files and directories

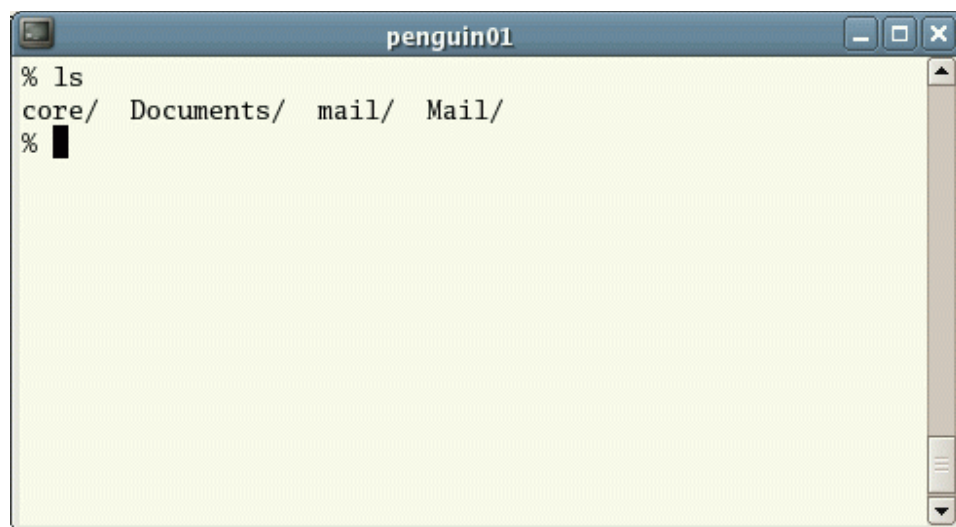
### ls (list)

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
```

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

A terminal window titled 'penguin01' with standard window controls (minimize, maximize, close). The terminal has a light yellow background and shows the command '% ls' followed by the output 'core/ Documents/ mail/ Mail/' on the next line. The prompt '%' is followed by a black cursor block.

```
% ls
core/ Documents/ mail/ Mail/
%
```

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 -l** lists your files in 'long format', which contains lots of useful information, e.g. the exact size of the file, who owns the file and who has

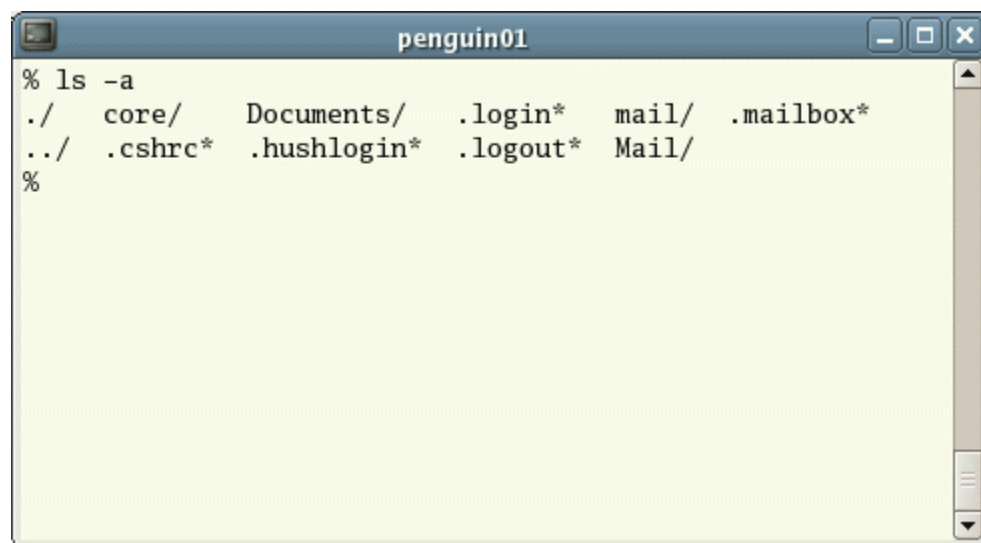
the right to look at it, and when it was last modified.

**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). The terminal shows the command '% ls -a' and its output. The output lists files and directories in two rows: './', 'core/', 'Documents/', '.login\*', 'mail/', and '.mailbox\*' on the first line; and './', '.cshrc\*', '.hushlogin\*', '.logout\*', and 'Mail/' on the second line. The prompt '%' appears at the end of the second line.

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

**ls** is an example of a command which can take options: **-a** is an example of an option. The options change the behavior of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behavior of the command.

**vi filename** is an editor that lets you create and edit a file.

**more filename** shows the first part of a file, just as much as will fit on one screen. Just hit the space bar to see more or q to quit. You can use /pattern to search for a pattern.

**mv filename1 filename2** moves a file (i.e. gives it a different name, or moves it into a different directory .

**cp filename1 filename2** copies a file **filename1 to filename2**

**rm filename** removes a file.

**rm -i filename** which will ask you for confirmation before actually deleting anything.

- **diff filename1 filename2** compares files, and shows where they differ
- **wc filename** tells you how many lines, words, and characters there are in a file.

## 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 **bca** in your current working directory type

```
% mkdir bca
```

To see the directory you have just created, type

```
% ls
```

## 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 bca
```

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

## 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 bca directory).

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

### 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).

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.

## 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/bca/bca001
```

which means that **bca001** (your home directory) is in the sub-directory **bca** (the group directory), which in turn is located in the sub-directory, which is in the **home** sub-directory, which is in the top-level root directory called **" / " ..**

## More about home directories and pathnames

### Understanding pathnames

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

```
% ls bca
```

to list the contents of your **bca** directory.

### ~ (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 ~/bca
```

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

Command	Meaning
ls	list files and directories



<b>ls -a</b>	list all files and directories
<b>mkdir</b>	make a directory
<b>cd <i>directory</i></b>	change to named directory
<b>cd</b>	change to home-directory
<b>cd ~</b>	change to home-directory
<b>cd ..</b>	change to parent directory
<b>pwd</b>	display the path of the current directory

## About other people

**w** tells you who's logged in, and what they're doing.

**who** tells you who's logged on, and where they're coming from.

## About your (electronic) self

**whoami** returns your username.

**passwd** lets you change your password,.

## Miscellaneous tools

**date** shows the current date and time.

**cal** shows a calendar of the current month. Use e.g., 'cal 10 1995' to get that for October 95, or 'cal 1995' to get the whole year.

You can find out more about these commands by looking up their manpages:

**man *commandname*** --- shows you the manual page for the command

