KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
Information and Computer Science Department
ICS 431 Operating Systems
Lab # 1
Introduction to Unix

Objectives:

This Lab is for new users of the UNIX timesharing system. The students assumed to have some prior experience with computers, but not necessarily with UNIX. The followings are the primary objectives of this lab session:

- What is Unix
- Understanding Unix File System
- Login to Unix Machine via Telnet
- Create and maintain files and directories of files.
- Be familiar with PICO to create and modify a text file

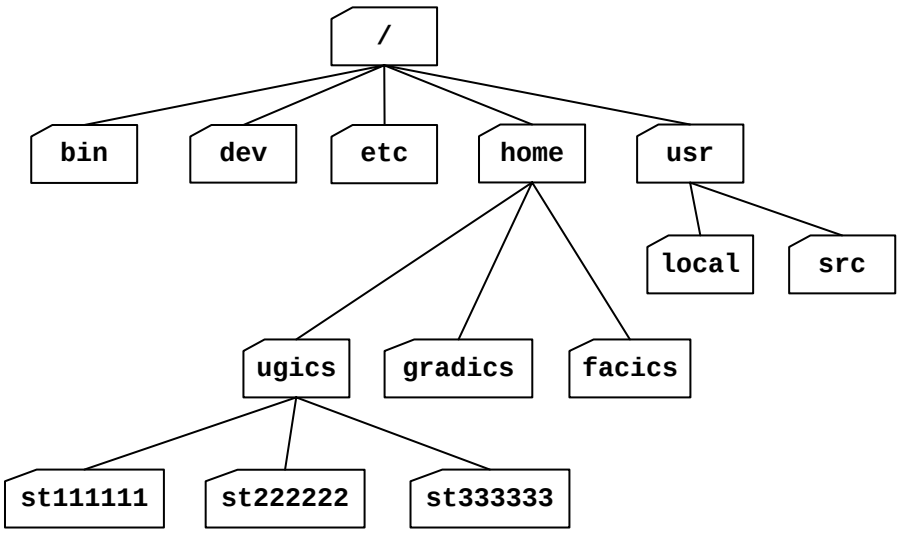
What is Unix

UNIX is a powerful operating system originally developed at AT&T Bell Labs. It is very popular among the scientific, engineering, and academic communities due to its multi-user and multi-tasking environment, flexibility and portability, electronic mail and networking capabilities, and the numerous programming, text processing and scientific utilities available. The UNIX system is mainly composed of three different parts: the **kernel**, the **file system**, and the **shell**.

1. The **kernel** is that part of the system which manages the resources of whatever computer system it lives on, to keep track of the disks, tapes, printers, terminals, communication lines and any other devices.
2. The **file system** is the organising structure for data. The file system is perhaps the most important part of the UNIX operating system. The file system goes beyond being a simple repository for data, and provides the means of organizing the layout of the data storage in complex ways.
3. The **shell** is the command interpreter. Although the shell is just a utility program, and is not properly a part of the system, it is the part that the user sees. The shell listens to your terminal and translates your requests into actions on the part of the kernel and the many utility programs

More about Unix File System

A file system is a logical organization of storage space designed to contain files in directories. The UNIX file system is quite similar to that of MS-DOS or Windows. It is organized hierarchically (inverted tree) into directories for efficient organization. However, in Windows, there are many logical trees represented by drive letters such as C:, D:, H:, etc… . In Unix, all file systems (Hard Disks, CD-ROMs, Floppy Disks, ZIP drives, network mounts, etc …) are "mounted" onto one logical tree. The top of the hierarchy is traditionally called **root** which is represented by a **/** (**slash**).



Part of file-system tree

In Unix, everything is treated as a file. A directory is a file. It is a file that contains a list of files and information belonging to those files. This would include things like who "owns" (created) the file, how long it is, and who can use it. Since a directory is simply a list of files, it can contain any file in it, including other directories.

Absolute and Relative Names

You can specify a file or directory by its **path name**. There are two ways of expressing the path name: **Full (absolute)** path name or **relative** path name. The full path name starts with the root. /, and follows the branches of the file system, each separated by /, until you reach the desired file.

However, a relative path name specifies the path relative to your current working directory. Relative path name are more convenient because they are shorter, but must be used with care. They never begin with / (slash). Now, we have to introduce two special directory entries:

- `.` the current directory
- `..` the parent directory

Examples: User the above diagram and assume that your current working directory (your current position in the file system) is **`ugics`**:

1. A file named "**salam.txt**" is placed inside the directory "**st111111**".
   The absolute (full) name is: `/home/ugics/st111111/salam.txt`
   The relative name is:
   `st111111/salam.txt` OR `./st111111/salam.txt`
2. A directory named "**ahmed**" is placed inside the directory "**gradics**".
   The absolute (full) name is: `/home/gradics/ahmed`
   The relative name is: `../gradics/ahmed`
3. A file named "**csh**" is placed inside the directory "**bin**".
   The absolute (full) name is: `/bin/csh`
   The relative name is: `../../bin/csh`
4. A directory named "**st123456**" is placed inside the directory "**ugics**".
   The absolute (full) name is: `/home/ugics/st123456`
   The relative name is: `st123456` OR `./st123456`

Accessing Unix System

Throughout this course, we will access Unix systems via Telnet with a telnet client such as Windows Telnet. In order to login successfully to a Unix system, you need the following pieces of information:

1. The Unix machine name. The list of machines on which you can do telnet can be found on CCSE website*. You will get the similar home directory on all these machines and similar programs.
2. User name
3. Password

Starting a new session via Windows Telnet

1. Click on Start
2. Click on Run …
3. Inside text box, type the following

   ```
   telnet solaris5
   ```
4. If you successfully connected to the Unix machine, a 'login: ' prompt will be displayed.
5. Now type your username and press "Enter"
6. Next type your password and press "Enter". Of course you will not see your password while you're typing it.
7. If you succeed, a welcome message will be displayed followed by a shell prompt.

Exercise: What is your shell prompt?

Important notes:

As the number of machines connected to the network increases, the probability of having your network transactions "sniffed" (that means viewed by unauthorized programs or users) increases. Therefore, the most suitable way to connect to Unix system is through secure shell (SSH) program. The program SSH is an access method similar to telnet. The primary difference is that the entire data stream put on the network by SSH is encrypted. For running SSH session on any Unix machine, you need to have SSH client on your PC. The most popular SSH client is PuTTY. You can get and setup PuTTY from this link.

---

* The link is http://www.ccse.kfupm.edu.sa/~waqqas/tmp/pages/services/ccse_accounts/list_of_unix_machines.php

Unix Commands:

1. Listing files and directories

ls (list)

When you first login to any Unix machine, your current working directory is your HOME directory. Your home directory has the same name as your username.

Example 1: To find out what is in your home directory, type:

```
% ls
```

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!!!

Example 2: To list all files in your home directory including those whose names begin with a dot, type:

```
% ls -a
```

The `-a` indicates an option to show ***all*** files in your home directory.

- **`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. (See later in this Lab)

Example 3: Type the following command:

```
% ls /
```

The `/` character is the directory name for the **root** directory. What you should see are the subdirectories and any files that are located in the root directory.

Exercises (1a):

1. Try putting the `l` option. What is the output?
2. Try putting the `a` option and the `l` option together (`-al`). Does the order of the parameters affect the output?
3. Write a command to display all files in the directory `/etc` ?

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 course.

Example: To make a subdirectory called **`ics431`** in your current working directory type

```
% mkdir ics431
```

Now to see the directory you have just created, type

```
% ls
```

- Please note that creating a subdirectory will not change your current position in the file system tree. Thus, after creating `ics431`, your current directory stills remain you HOME directory.

3. Changing to a different directory

cd (change directory)

If you just logged into your Unix account, you are placed ***in*** your home directory. '***in***' means your current position in the file system tree. The `cd` (change directory) command is used to change your current directory (your current position) to another directory.

Example: To change to the directory you have just made, type

```
% cd ics431
```

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

Exercises (1b):

1. Make another directory inside the `ics431` directory called "`LabOS`" then make "`LabOS`" your current working directory.
2. Now type the command:
   ```
   % cd ..
   ```
   followed by:
   ```
   % ls
   ```
   In which directory you are located now?
3. Type the following command
   ```
   % cd /etc/fs
   ```
   followed by:
   ```
   % ls
   ```
   In which directory you are located now?
4. Now enter the command:
   ```
   % cd
   ```
   In which directory you are located now?
5. Write a single Unix command to make "`LabOS`" your current working directory.

4. Pathnames

pwd (print working directory)

The `pwd` command is very useful to know your current position in the file system. It displays the full path name of your current working directory.

Example: Now, type

```
% pwd
```

What is your current working directory?

- **Be sure your current working directory is `LabOS` before moving to next section**

5. Pico: Text Editor program

While using UNIX you will often want to create a text file and then change its content in some way. A text editor is a program that has been designed especially for this purpose. The easiest of all editors is the **pico** editor.

Example: Creating a new file named ***hello.txt***

1. At your Unix shell prompt, type `pico filename`, replacing *`filename`* with the name of the file you want to create or edit. For example, to create a file and name it ***hello.txt***, type:

   ```
   % pico hello.txt
   ```

   Pico displays a menu bar of commonly-used commands at the bottom of the screen.

2. Type the following lines:
   Hello world,
   I'm a student in KFUPM. I'm learning Unix OS
   So this is my first lab

3. To exit Pico, press **[Ctrl]**+**[x]**. Since you have mode some changes, Pico asks whether to same them. Type **y** (yes) or **n** (no). If you type **y**, Pico displays the filename. (To save the edited file under a different name, delete the filename and type a new one.) Press **[Return]**.

Exercise (1c):
Modify the above file such that the content will be as follow:

Hello world,
My name is Ahmed Abdullah
I'm a student in KFUPM. I'm learning Unix OS
So this is my first lab

6. 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.

Example: To verify your wok in previous section, you can view the content of *`hello.txt`* by typing:

```
% cat hello.txt
```

The `cat` command is useful for displaying short files of a few lines. To display longer files use **`less`** or **`more`** commands.

7. Copying Files

cp (copy)

The `cp` command allows you to copy a file from one location to another location. There are different syntaxes to use **`cp`** command as shown below:

Syntax 1: (copy a file to another file)

```
% cp file1 file2
```

where ***`file1`*** is the name of an existing file and ***`file2`*** is the name for the new copy of that file. The original file will remain unchanged and a copy will be placed in ***`file2`***. If ***`file1`*** and ***`file2`*** are not in current directory, then you have to specify its pathname.

Syntax 2: (copy a file to another directory)

```
% cp file directory
```

where ***`file`*** is the name of an existing file and ***`directory`*** is the name for the destination directory. The original file will remain unchanged and a copy will be placed in that ***`directory`***.

- **Be sure your current working directory is `LabOS` before moving to `examples`**

- **Create a new directory named "`backup`" inside your current directory**

Example1: To create a backup of **hello.txt** by copying it to a file called **salam.txt**, type:

```
% cp  hello.txt  salam.txt
```

Now to see the file you have just created, type:

```
% ls
```

Observe that you have two files in your current directory; **hello.txt** and **salam.txt**.

Example2: To copy the file named "`salam.txt`" to the directory names "**`ics431`**", type:

```
% cp  salam.txt  ..
```

Why!!? Now to see the file you have just copied, type:

```
% ls  ..
```

Example3: To copy the file named "`salam.txt`" from directory "`ics431`" to your current directory named as "salam.bak", type

```
% cp  ../salam.txt  salam.bak
```

Why!!? Now to see the file you have just copied, type:

```
% ls
```

Example4: To put a copy of "`salam.txt`" into your backup directory, type

```
% cp  salam.txt  backup
```

Why!!? Now to see the file you have just copied, type:

```
% ls backup
```

Exercise (1d): Copy the file named "**`Hello.txt`**" to your home directory? Verify your work.

8. Moving Files

mv (move)

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.

Example: We are going to move the file "**salam.bak**" to your backup directory, type

```
% mv  salam.bak  backup
```

Now what is the content of your current directory? What is the content of your backup directory?

9. Removing Files and Directories

rm (remove)

rmdir (remove directory)

To delete (remove) a file, use the **rm** command. However, to delete a directory use the **rmdir** command. There is one constraint to delete a directory; the directory must be empty.

Example 1: We know that your backup directory contains two files named "salam.txt" and "salam.bak". We are going to delete the file "salam.txt" and, as an exercise, delete the second file.

```
% rm  backup/salam.txt
```

Example 2: If you successfully delete the two files, you can delete the backup directory. To delete the backup directory, type:

```
% rmdir  backup
```

Exercises (1e):

1. Delete the file named "**salam.txt**" from directory "**ics431**"
2. create a directory named "**tempdir**" by using **mkdir** then remove it.

Exercises

Note:

Lab exercise will be given at the lab time.