

Name: \_\_\_\_\_

Student ID #: \_\_\_\_\_

## Lab #1: COMP 3000B (Operating Systems) January 16, 2007

In this lab, we will examine some core programs of the UNIX environment as implemented on the Debian GNU/Linux 3.1 systems in the lab. These programs provide a window into the functionality of the operating system.

Please answer all questions below. There are a total of 79 marks available.

### 1 Part A

This section is designed to be completed in the lab. You get 10% of the total marks for attempting to do part A during assigned lab hours. Please ensure that one of the Lab instructors takes your attendance.

1. [8] Have the instructor mark down that you were present and attempted part A during lab hours.

#### 1.1 The Shell

The shell or command line provides a text interface for running programs. While not as visually pleasing as a graphical interface, the shell provides a more clear representation of the functionality provided by the operating system. We will therefore use the shell for most of the work in this course.

To run a program contained in the current directory in the shell, you need to prefix the name of the command with a `./`. This `./` tells the shell that the location of the command you wish to run is the current directory. By default, the shell will not search for executable commands in the current working directory. To run most system commands, the name of the command can be typed without a path specification.

#### 1.2 Help

1. [1] When working in the shell, help is available on most programs in the system, especially those that are command line based. This system of help is available by using the `man` command. If one wanted to get help on the `echo` command, the associated command would be `man echo`. What does `man` stand for?
2. [1] The `man` command can also be used to get help on standard C functions. Briefly (in one line), what does the C function `fcntl` do?
3. [1] What does the `less` command do?

## 1.3 Shell Basics

1. [1] The `which` command can be used to figure out what directory an executable program resides in. Using this command, what directory contains the `bash` command?
2. [1] Which of the following is another shell that can be used instead of `BASH`? `mv`, `vdir`, `fgconsole`, `tcsh`, or `arch`
3. [1] The `ls` command can be used to get a listing of the files in a directory. What option is passed to `ls` to see *all* files within a directory?
4. [1] The `ps` command is used to get a list of processes which are running on the system. Start a new shell from `bash` by running `sh`. Does the new shell replace or run on top of the old `BASH` shell? You can use the `ps` command to determine if `bash` is still running.
5. [1] Because each process is started by some other process, there is a process tree structure on the system. From within your `bash` terminal, determine what process started `BASH` by using the `pstree` command.

## 1.4 Processes

Each application running on a system is assigned a unique process identifier. The `ps` command shows the process identifiers for running processes. Each process running on the system is kept separated from other processes by the operating system. This information will be useful for subsequent questions.

## 1.5 Permissions

Your permission to access a file in Unix is determined by who you are logged in as. All files on the Unix file system (including directories and other special files) have three different sets of permissions. The first set permissions denotes the allowed file operations for the owner of the file. The second set of permissions denotes the allowed file operations for a group of users. The third set of permissions denotes the allowed file operations for everyone else. A file is always owned by someone and is always associated with a group.

1. [1] The `ls` command with the `-l` option can be used to show both the permissions of a file as well as the owner and group associated with the file. Permissions are listed first, followed by the owner and the group.  
Who is the owner of the `/usr/local/bin` directory?
2. [1] What group is associated with the `/usr/local/bin` directory?
3. [1] Each user is a member of some number of groups. You can determine what groups you are part of by using the `groups` command. Based on the groups listed, would you (the “student” user in the lab) be a member of the group associated with the `/usr/local/bin` directory?

## 1.6 Environment

The environment on both Linux and Windows contains variable - value pairs which are useful to applications running on the system. In Linux, these environment variables can be printed on the command line by referring to the variable name prefixed with a \$ sign (eg: to output the value in the HELLO environment variable, one could write `echo $HELLO`).

1. [2] On the command line, run the following two sets of commands. Notice that the bash command will start a new shell separate from the shell that the HELLO environment variable was set in.

```
HELLO="Hi There"  
bash  
echo $HELLO  
exit
```

```
export HELLO="Hi There"  
bash  
echo $HELLO  
exit
```

What does the export command seem to do?

## 1.7 Dynamic Libraries

Most applications on the system do not contain all the code that they need right within the executable. Instead, dynamic libraries are loaded into the program address space when the program loads. As an example, the standard C library, which contains such functions as `printf` is loaded in at run-time.

1. [2] Using `ldd`, what dynamic library dependencies does the bash command have?
2. [2] In addition to the libraries listed as dependencies for the application bash by `ldd`, there may be other libraries that the application loads dynamically at run-time. Retrieve the process number *PID* for the bash process (using `ps`) and examine the `mmap` file located at `/proc/PID/mmap`. What other dynamic libraries have been loaded into the application while it has been running?

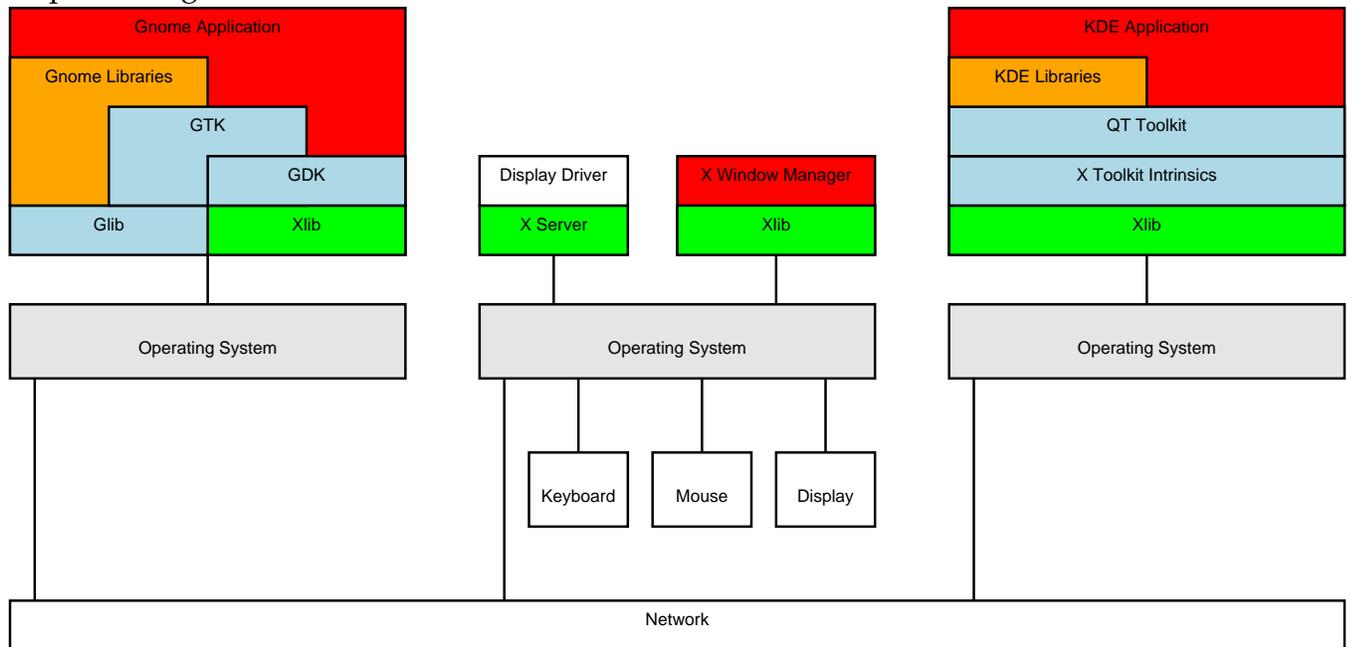
## 1.8 X

Below is an architectural overview of the structure of X Window System, the graphical display environment for Linux and other UNIX systems. Microsoft Windows has a different architecture, as does Apple's MacOS X. Under Linux, core X functionality is provided by those elements coloured green. Additional graphical libraries are coloured blue and orange. Applications are coloured red<sup>1</sup>. The X server, which handles the video card, is not contained in the kernel but

---

<sup>1</sup>Image adapted from <http://cnlart.web.cern.ch/cnlart/2000/001/kde-gnome2>

instead runs as a separate process on the system. This is in stark contrast to Windows, where video processing is done within the kernel.



The X window system, which takes care of controlling the video hardware, is separate from the window manager, which arranges the behaviour and look of windows on the screen. Under Linux, there are two major desktop environments built on top of the X window system, KDE and Gnome. Each of these have their own associated libraries and functionality.

1. [2] By examining the dynamic dependencies of the `xterm` command, determine whether it is a KDE application, Gnome application, or X application which does not depend on either Gnome or KDE.
2. [2] Is `metacity` a Gnome application, a KDE application, or neither?

## 2 Part B

This section is to be completed in your own time. Your Linux SCS computing account may be useful in solving some of the problems.

### 2.1 Processes

1. [2] What is the difference between the `fork` and `exec` function in a Unix environment?
2. [2] What is a Zombie process?
3. [8] Give an example C program which creates a zombie process. Note that `BASH` by default will collect and destroy zombie processes and so you will need to avoid `bash` destroying the zombie process during debugging. This can be done by delaying the parent exit (using `sleep` is one good way to do this).
4. [4] Perform the modifications to your program above to avoid creating a zombie process. List the new program.

## 2.2 Permissions

1. [3] Permissions on Unix are grouped into three basic file operations. What are these file operations?
2. [1] What does it mean to have execute permission on a directory?
3. [4] What are the 6 basic file permissions within Windows?
4. [1] What is the difference between the write and modify file permission in Windows?
5. [1] Because all files are stored in a directory, under Unix permission to delete, rename, and move files is determined by the users access rights on the directory the file is contained in. What attribute on the directory prevents those who can modify a directory from deleting files (hint: help on the `chmod` command may prove useful).
6. [4] In Windows, a file can be associated with more than one group and each group can have different access permissions. On Unix, new groups can be created by the system administrator which are supersets of other groups. Given this, is it possible to develop an access permission scenario which would be impossible to implement in Unix but possible to implement in Windows? If yes, give an example. If no, explain why.

## 2.3 Dynamic Libraries

1. [6] Dynamic libraries allow one copy of executable code to be used by many different processes on the system, without requiring that multiple copies of the code be stored on disk in different files. What are some problems that can arise when different programs use the same common DLLs (hint: "DLL Hell")?

## 2.4 Environment

1. [1] What does the `PATH` environment variable do?
2. [2] What environment variable tells X applications where to find the X server which it should communicate with to display the output?

## 2.5 X

1. [3] The architecture of X was given above, but the applications were shown running on other computers and connected over a network. Draw the architectural diagram if the applications were run on the same computer as the X server (which is the common scenario).
2. [2] There are two major desktop environments for Linux, Gnome and KDE. Each of these environments has it's own associated libraries which implement core functionality required for all applications. You can, however, run Gnome applications under KDE and KDE applications under Gnome. What feature of KDE and Gnome makes this possible?
3. [5] Draw a architectural diagram of how the graphical interface is laid out in Windows. Be sure to cite where you obtained the architectural information.