Install Linux on a Virtual Machine
Learn Basic Linux Commands

Objective:

In this first lab session, you are going to prepare the programming environment that you will need for this and future labs of the course. Also, you are going to learn how to use Linux at the command line level, by learning some of its essential commands.

Preparation:

You may have a PC loaded with Linux, MacOSX or Windows operating system. For the first two system types, skip this preparation section. If you have a Windows system, then we have prepared a readymade Linux appliance file for you to be used with a virtual machine along with other utility programs to create a computing environment suitable for your lab work. If a Linux environment was not installed already, you need to perform the following three steps before you can start this lab:

Step #1: *Download & install latest version of Oracle's Virtual Machine, VirtualBox*

a. Go to https://www.virtualbox.org/wiki/Downloads and download the latest version of *VirtualBox* and its *Extension Pack*. Make sure you get the version that is suitable for your operating system.
b. Run the downloaded Setup file and follow the installation wizard steps, or follow the installation guide at, https://www.virtualbox.org/manual/UserManual.html#installation_windows.
c. Note: You can install *VMware Workstation Player* instead, if you so prefer.

Step #2: *Install the Guest Linux Appliance:*

a. Get a copy of the *LAMP* package from the course web site. It includes the appliance (**.ova**) file.
b. Follow the instructions found in the included **readme** file to install the Linux appliance.

Step #3: *Install Windows Utility Applications:*

a. Download and install *PuTTY* from https://www.chiark.greenend.org.uk/~sgtatham/putty/releases/0.74.html.
b. Download and install *WinSCP* from https://winscp.net/eng/download.php.

Testing the Installed Environment: (2 points)

Now, that you have Linux installed on a virtual machine in your computer, you need to make sure that everything works as needed. The following procedure will guide you through a simplified typical session for developing C programs using this environment. You are going to test your installation by creating, uploading, compiling, debugging, editing and executing a small C program.

1. In your Windows (*host*) computer, open, *Oracle VM Manager* you just installed in Step #1
2. Select the Linux (*guest*) System you installed in Step #2, and make sure that the network adapter that appears in the list on the right, shows "**Host-only adapter**", then click on Start to run the Linux guest.
3. When you see the blue screen on the guest Linux system, note the IP address (e.g. 192.168.56.101), and reduce and leave both windows
4. Test network connectivity by pinging from the host to the guest in a host "cmd" screen. For example:

```
C:> ping 192.168.56.101
```

5. Copy the following C program and paste it into *notepad*, save it as "**add.c**" in your host computer.

```
#include <stdlib.h>
#include <studio.h>

void main(int argc, char *argv[]) {
    int a,b,c;

    a=atoi(argv[1]);
    b=atoi(argv[2]);
    printf("a,b = %d %d \n", argv[1], argv[2]);
    c=a+b;
    printf(" c = %d \n",c);
}
```

6. In the host computer, run the *WinSCP* application, and connect to the guest using the IP address you noted above. Use the username, "**root**" and the password "**123**"
7. If your connection was successful, you will get two file lists: On the left are files of the current directory of the host, and on the right are files of the current directory of the guest. On the right go to the directory called "**/root**", and on the left go to the directory you saved "**add.c**" in, right-click on it and choose upload
8. From within *WinSCP*, Click on the icon of *PuTTY* (fifth from top left), to get a secure command shell client to the guest system. When asked, use the username, "**root**" and the password "**123**" again. From here you can work interactively with the guest Linux system, exactly, as if you were actually there
9. The guest command prompt is "**$**" or "**%**" for normal users; "**#**" for the super user. At the prompt, Type:

```
# g++ add.c -o add
```

10. Note the error message that you get, indicating a problem with "**studio.h**" at line 2:

```
add.c:2:20: fatal error: studio.h: No such file or directory
 #include <studio.h>
                    ^
compilation terminated.
```

11. This means that you have to correct a problem at line 2! And in particular, the compiler did not find the file you specified "**studio.h**" as part of the C library. This situation can be corrected quickly by noting that there is one extra character in the name of "**stdio.h**", the standard i/o library, which needs to be removed.
12. Instead of going back to the host to correct the file there, and upload it again, we prefer to use the most powerful editor in the Linux system, called "*vim*". At the guest prompt type:

```
# vim add.c
```

13. This will display the contents of the specified file, and allow you to edit it quickly and easily. Use the arrow keys on your keyboard to Move the cursor to character 13 in line 2, at the letter "**u**"
14. Or, you can also jump there directly by typing: "**j12l**". Note that the last character is small 'L'
15. Now, you need to delete the extra character, save the changes, and exit. Type: "**x:wq**". You will learn more about *vim* later.
16. Recompile again. This time two errors are displayed. These are related to the syntax differences between C and C++. Since our program is written in C, we should have used the *gcc* compiler instead of *g++*
17. Recompile again using:

```
# gcc add.c -o add
```

18. This time there will be no errors reported. To test run the program by adding numbers 1076 and 764, type:

```
# ./add 1076 764
```

If everything went right and you get 1840, then congratulations, your development environment is ready.

Linux Command Tutorial: (2 points)

Now it is time to learn more about Linux commands. You should gain good essential knowledge of the basic commands in Linux, to be able to work comfortably in this environment. Every system engineer or administrator will find it easier and more powerful to manage his systems using CLI rather than GUI.

First, it is safer to work as a normal user rather than a super user. Working as a normal user protects your system and its setup from being changed accidently! At the guest's super user command prompt, type:

```
# adduser <your_prefered_userID>                    // e.g. adduser ahmad
```

The system will respond by the following:

```
Adding user `ahmad' ...
Adding new group `ahmad' (1000) ...
Adding new user `ahmad' (1000) with group `ahmad' ...
Creating home directory `/home/ahmad' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
```

Enter your desired password. You may be asked to re-enter it again for confirmation, do it. Then, you will be asked to enter some more information. You can skip by pressing the Enter key. Press Y when finished.

```
passwd: password updated successfully
Changing the user information for ahmad
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n]  Y
```

Now, you can disconnect as a super user, and re-connect with *WinSCP* and *PuTTY* as a normal user. Use the username and password you specified above. Your home directory will be: **/home/YourID**.

Go to item #6 titled "*Unix Tutorial for Beginners*" in the references page linked from the main page of this course and do tutorials one and two there. Apply the examples on your tested environment.

Required Submittals:

This lab is worth 4 points. To be eligible for the full credit of this lab, you are required to show that you did the work specified above, as follows:

1. Take screen shots of the steps you did to run the above C program.
2. Do the exercises in both tutorials and take screen shots of the results.
3. Submit all the screen shots you took, in one .pdf file.

Further Work:

As an extra work, to get even more familiar with Linux commands, try exercising on the commands listed in the next section.

Linux Basic Command Summary:

man command
Display manual pages for command.
If you do not know the exact name of command, issue the command *man* -k info to get a list of all commands dealing with the subject info.
eg. *man* -k editor will list all available editors.

exit
Closes an open shell or logs the user out of the computer.

more file
Display a file one screen at a time.
This command is often used with a pipe to display the output of another command one screen at a time.
Hit the space bar to display the next screen;
Type "q" to quit the display.

rm file-list
Remove (delete) file-list.

mv source dest
Move or rename one or more files.
dest may be a new file name or a directory. Be careful not to clobber useful files.

cat file-list
Join or display files.
This command can concatenate files (eg. cat file1 file2 > file3) or list files to the screen (eg. cat file).

grep word file-list
Write out lines in files in file-list that contain the given word.

wc file
Output a count of the lines, words and characters in the file. Options -c, -w, -l lets you output just one of these.

cd directory
Change to another working directory.

pwd
Display the current working directory.

mkdir directory
Create one or more directories.

rmdir directory
Delete an empty directory.

ls [options] [file-list]
Display information about one or more files.
-a also display hidden files (which begin with ".")
-l display several columns of information about each file.

cp source dest
Copy one or more files. dest may be destination files or a directory.

chmod options file
Change file permissions on file.
For example: chmod u+x myshell.

rsh machine command
Execute command on another machine
e.g., rsh engg ls

telnet host or *rlogin* host
Log into remote host computer.

nice [options] [command-line]
Change the priority of a command.
Example:
nice +4 prog_name > name.lst &

top
Display currently active processes.

lp file-list
Print file-list.

Unix Command Quick Reference:

command	*argument*	*action*
Logging In and Out, Using the Command Line		
login		exit and re-login
passwd		change your password
exit		logout and exit from the system
ctrl-a		move to the beginning of the command line
ctrl-e		move to the end of the command line
ctrl-b		move backward on command line (left arrow)
ctrl-f		move forward on command line (right arrow)
ctrl-d		delete one character to right of the cursor (delete key)
ctrl-k		delete all of the current line to the right of the cursor
ctrl-p		recall the previous command in the command history
ctrl-n		recall the next command in the command history
Navigating in Directories and Manipulating Files		
ls		list contents of the current directory
	-a	list *all* contents (including hidden files and directories)
	-l	list contents with full file details (permissions, owner…)
	-F	list all contents with helpful marks / directory, * executable file, @ symbolic link
	-R	list all contents of current directory and its subdirectories
	-t	list files, sorted by timestamp rather than filename
mv	*oldname newname*	rename or move a file or directory
cp	*name copyname*	copy a file
	-r *name copyname*	copy a directory and its contents
rm	*name*	delete a file
	-r *name*	delete current directory and all its contents
grep	*pattern file*	powerful utility for searching each line in a file to find those that match a pattern — 'man grep' for details
find	*/path* –name *file*	find *file* under *path* (can use wildcards)
partial filename	esc esc	complete a filename from a partial filename
more	*file*	view file one page at a time; use / while in more to search, b to go back a page, q to quit
compress	*file*	convert a file to a compressed version (will end in .Z)
uncompress	*file*	decompress a .Z file to its original state
chmod	ugo+-rwx *file/dir*	modify file permissions
mkdir	*directory*	make a new directory
rmdir	*directory*	delete a directory (must be empty)
cd	*/path*	change the current directory to a specific directory
	..	go 'up' one directory level
	~	move to your home directory
	-	move to the last directory visited

command	*argument*	*action*
Job/Process, Print and System Management		
ctrl-z		suspend a job — return to it with fg, place it in the background with bg
ctrl-c		exit the foreground job or program
jobs		view current jobs under your username
fg	%*n*	resume job *n* in the foreground
any command	&	run a command in the background; return to it with fg
nohup	*any command*	run a command that can keep running after you logout
ps	-ef | grep *user*	view current running processes; including process ids
kill	*processid* -9 *processid*	kill a process cleanly and politely kill a process forcefully *(use only if necessary)*
enscript	*file* -2d *file* -DDuplex:true *file*	print *file* to the default printer print *file* 2 pages per sheet, landscape orientation print *file* double-sided
lpq		check print queue for jobs
lprm	*jobid* or *user*	remove print jobs
sasclean		automatically clean out your SAS work files
Useful tools		
man	*command* -k *keyword*	read the online technical help for *command* find Unix commands related to your keyword
date		display current time and date
cal	 *month year*	display current month display specified month and year
who		show who is currently logged on
pico	*filename (optional)*	basic and user-friendly file editor
emacs	*filename (optional)*	powerful, intermediate–level file editor
pine		Unix e-mail
biff	*y* or *n* (on or off)	Unix e-mail notification — alerts you to a new message
Pico Commands *(also displayed at the bottom of the Pico screen)*		
ctrl-g		help
ctrl-o	*file*	save as (defaults to current filename)
ctrl-x		exit (will prompt to save if necessary)
ctrl-w	*search term*	search for text (defaults to last search term)
ctrl- ^ (ctrl-shift-6)		begin marking a text block to cut or copy
ctrl-y, ctrl-v		go one page up or one page down, respectively
ctrl-a, ctrl-e		go to beginning or end of current line, respectively
ctrl-k		cut current line or selected text
ctrl-u		paste most–recently cut text
ctrl-t		spell check file
ctrl-r	*file*	insert an existing file into current file
ctrl-j		clean up odd line lengths in current paragraph (note: a "paragraph" is text surrounded by two blank lines)