Course Booklet for Getting Started with Setting up an Embedded Platform

Starters Guide for Embedded Systems

By Emertxe Trainers

Version 2.2 (December 18, 2014)

Course Email: embedded.courses@emertxe.com

Contents

Preface

0.1 Course: Education goals and objectives

Linux Systems course module is a workshop which covers an overview of the Linux Operating System. By the end of the module, students are expected to have made their concepts very clear and become comfortable in Linux

The duration for this module roughly spans 6 days

Pre-requisites for this module are: BE/BTech (4th semester onwards) and Basic OS concepts and computer Usage

This booklet serves as a workbook guide, as we go along through the Advanced C course module at Emertxe Information Technologies Pvt Ltd. It will provide the enough information about all the commonly used commands in Linux along with starting steps of programming. It will provide a ground for practice and for writing your personalized notes

As part of this, it would cover: Overview of Linux OS, CLI Interface, Hierarchial File Sytem, File related commands, VI editor, Shell Scripting and makefiles

Finally, there is a complete appendix dedicated to a set of assignments well sorted and graded, which gets reviewed after every batch

Chapter 1

Over view of Linux Systems

1.1 Expectations out of this module

Notes:

1.2 Introduction

Notes:

- What are the different OSes?
  Notes:

- Are Linux and Unix same?
  Notes:

1.3 Over View of Linux Systems

Notes:

1.4 Architectures Supported

Notes:

1.5 Installing Linux on X86 System

- We can install OS in different ways
  1.CD-ROM
  2.USB
  3.N/W

  Notes:

1.6 Partition on HardDisk

- What is partition & why to make?
  Notes:

- How to make?
  1.GUI
  2.COMMAND LINE
  Notes:

- How many number of partitions we can do?
  Notes:

1.7 X86 Booting

Booting sequence of X86 system

- 1.BIOS
  Notes:

- 2.FIRST Bootloader
  Notes:

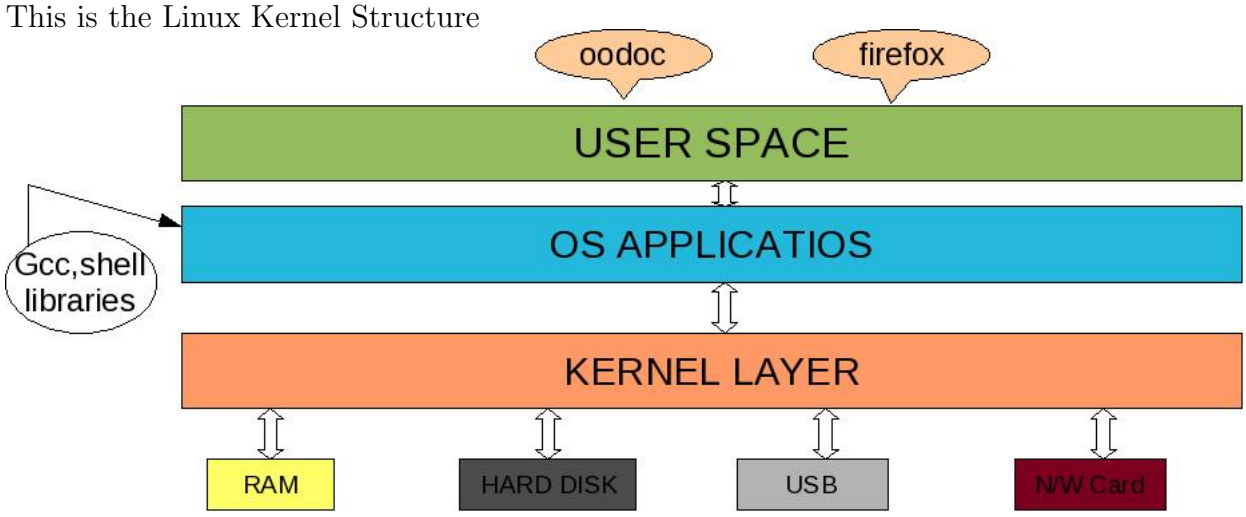
- 3.Second Bootloader
  Notes:

- 4.Kernel (OS)
  Notes:

- 5.Applications
  Notes:

1.7.1 Linux Kernel Structure

This is the Linux Kernel Structure



- Hard Ware Layer
- Kernel Layer
- OS Applications
- User Layer

Notes:

1.8 Summary

Chapter 2

Command Line Interface

- Two types
- Textual

  A CLI (command line interface) is a user interface to a computer's operating system or an application in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system, and then enters another command, and so forth. The MS-DOS Prompt application in a Windows operating system is an example of the provision of a command line interface.

  A command line interface (CLI) is a textual mode of interaction between computers and humans.

- GUI

  A graphical user interface (GUI) is a type of user interface which allows people to interact with electronic devices such as computers

  In contrast, the Graphical User Interface or GUI operate via mouse and keyboard, while CLIs only depend on commands entered via a keyboard

Notes

2.1 The Shell

- what is a shell

  The shell is nothing but the Command line user interface
  Notes

- Different types of shells

2.1.1 Login Shell

login-shell responsibility is to log you into the system. This is the shell you get when you're prompted with your uname & your password login-shell will start the non login-shell

2.1.2 Non-login Shell

Like different languages, our UNIX system provides a variety of shell types Sh — Bourne shell - basic shell Bash— Bourne again shell- Standard GNU shell & very powerfull tool for the advanced and professional user Csh — C shell - by syntax this shell resembels for C programming language Ksh — Korn shell - A superset of the Bourne shell

Notes

2.2 Bash Files

- .bash_profile
- .bashrc
- .bash_history
- .bash_logout

Notes

2.3 Environmental Variables

Notes:

2.3.1 The bash shell variables

- env
- HOME
- PATH
- PWD
- PS1
- n=10
- export n
- unset n

Notes:

2.4 Some basic shell Commands

- ls
- pwd
- cd
- man
- exit
- which

Notes:

2.5 Shell built in Commands

- cd
- break
- unset
- man
- List some shell built in commands

Notes:

Chapter 3

User Interface

A user interface is a linkage between a human and a device or system that allows the human to interact with system.

- Login prompt
- CLI
- GUI

Notes:

3.1 User specific commands

- useradd
- userdel
- su -
- finger
- passwd
- who, w
- whoami

Notes:

3.2 Remote login & Remote copy

3.2.1 ssh

- ssh username@ipaddr

Notes:

3.2.2 scp

- scp [file] username@ipaddr:[Destination]
- scp username@ipaddr:[file] [Destination]

Notes:

3.3 Misc. Useful commands

- uname
- man
- info

Notes:

3.4 Redirection

In this lesson we will explore a powerful feature used by many command line programs called input/output redirection. As we have seen, many commands such as ls print their output on the display. This does not have to be the case however. By using some special notation we can redirect the output of many commands to files, devices, and even to the input of other commands. Totally we have three types of redirections

- Standard O/P redirection Most command line programs that display their results do so by sending their results to a facility called standard output. By default, standard output directs its contents to the display. To redirect standard output to a file, the < character is used like this:

- [me@linuxbox me]$ ls > file_list.txt

- Standard O/P redirection If you want the new results to be appended to the file instead, use >> like this:
- [me@linuxbox me]$ ls >> file_list.txt

- OutPut redirection If you want to redirect the error massages to the files instead, use 2> like this:
- [me@linuxbox me]$ lls 2> file_list.txt

Notes:

3.5 Piping

A pipe is a form of redirection that is used in Linux operating systems to send the output of one program to another program for further processing. A pipe is designated in commands by the vertical bar character.

- ls | grep *.c

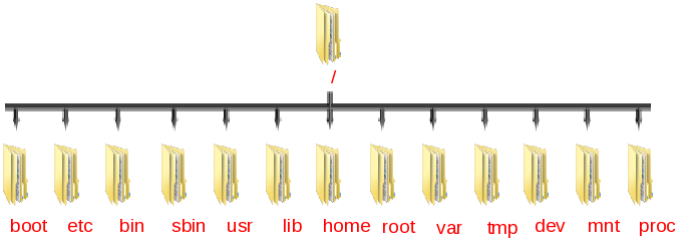
Notes:

Chapter 4

Directory & File System Structure

4.1 Directory Structure of Linux

- This is the whole structure of Linux System



- All Accesses into a Linux System are through a User
- Every thing is viewed as a file in Linux System, even a directory also

4.1.1 Linux Directory Structure Overview

One of the most noticeable differences between Linux and Windows is the directory structure Not only is the format different, but the logic of where to find things is different.
The Directory Structure in Unix & Linux are a unified Directory Structure where in all the directories are unified under the / Root file system. Irrespective of where the File System is physically mounted all the directories are arranged hierarchically under the Root file system.
Lets have a quick stroll across the different directories under the Linux Filesystem Hierarchy

- / : This is the root directory. This is where the whole tree starts.The partition where / (the root directory) will be located on a UNIX or UNIX-compatible system.
- /boot : Contains static files for the boot loader. This directory only holds the files which are needed during the boot process.
- /dev : Special or device files, which refer to physical devices.
- /sys : This contains the Kernel, Firmware and system related files.
- /bin : Contains the essential binaries for users and those utilities that are required in single user mode. Examples, include cat, ls, cp etc.
- /sbin : Like /bin, this directory holds commands needed to boot the system, but which are usually not executed by normal users.
- /lib : This directory should hold those shared libraries that are necessary to boot the system and to run the commands in the root file system.
- /home : All the user home directories are held under this directory with the exception of the root home directory which is kept under /root directory. This directory holds users files, personal settings like .profile etc.
- /media : This directory contains mount points for removable media such as CD and DVD disks or USB sticks.
- /mnt : This directory is a mount point for a temporarily mounted file system.
- /opt : A rarely used directory in Linux for Optional Software Packages. This is extensively used in UNIX OS like Sun Solaris where the software packages are installed
- /var : This directory contains files which may change in size, such as spool and log files, jobs, mail, running process, lock files etc.
- /proc : This is a mount point for the proc file system, which provides information about running processes and the kernel.
- /tmp : A temporary file system which hold temporary files which are cleared at system reboot.

- /etc : Contains configuration files which are local to the machine.The /etc/directory contain essential System configuration files /etc/hosts, and network configuration files.when you login bash reads the /etc/profile instructions.These usually set the shell variabels PATH,USER,MAIL,HOSTNAME

Notes:

4.1.2 The Linux File Systems

In Linux we have different types of file systems ext2, ext3, etc.....

- File system related shell commands
- stat :
- mount :
- find , locate :
  Notes:

4.1.3 File Related Shell Commands

Basic Shell commands

- pwd : This command print name of current working directory.
- cd : Change directory.
- ls : List information about the FILEs.
- df : df displays the amount of disk space available on the file system containing each file name argument. If no file name is given, the space available on all currently mounted file systems is shown.
- du : Summarize disk usage of each FILE, recursively for directories.
- cp : copy files and directories from SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
- mv : move (rename) files.
- rm : remove files or directories, rm removes each specified file. By default, it does not remove directories.
- mkdir : Create the DIRECTORY(ies), if they do not already exist.

- rmdir : Removes the DIRECTORY if itis Empty Directory.
- cat : Concatenate files and print on the standard output.
- less : Less is a filter for paging through text one screenful at a time.
- tail : Print the last 10 lines of each FILE to standard output.
- head : Print the first 10 lines of each FILE to standard output.
- touch : Will create the file, if file exits it just Update timestamp of each FILE to the current time.
- wc : Print the total number of words, characters, and tolatl number of lines in a file.
- sed : Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream

Notes:

4.1.4 File Detailed Listing

- Type ls -l on you shell, and you would see something similar to :

```
crw-rw---- 1 root      tty     4,  30 Sep 15 09:03 tty30
brw-r----- 1 root      disk    1,   3 Sep 15  2008 ram3
drwxr-xr-x 2 root      root        0 Sep 15  2008 pts
lrwxrwxrwx 1 root      root       15 Sep 15 09:03 stderr -> /proc/self/fd/2
srw-rw-rw- 1 root      root        0 Sep 15 09:04 log
prw------- 1 root      root        0 Sep 15 09:04 initctl
```

Notes:

4.1.5 Unix File Types

- We have different types of files in Linux System

Notes:

4.1.6 Unix File Permissions

- For chainging the file permissions
- chmod :
  chmod changes the file mode bits of each given file
  according to mode, which can be either a symbolic representation of changes to mak
  or an octal number representing the bit pattern for the new mode bits.
- chown :
  chown changes the user and/or group ownership of each given file

Notes:

4.1.7 File Links

A link is simply a way to refer to the contents of a file. Types of links

- 1.Hard Link
  Command to create HardLink : ln Notes:

- 2.Soft Link
  Command to create SoftLink : ln -s Soft link is also called Symbolic link Notes:

4.1.8 Advanced File Related Commands

- cut :
- split :
- cmp :
- diff :
- uniq :

Notes:

4.1.9 File Compressions

File compres'n are used to reduce the size of the file, that only regular files In particular, it will ignore symbolic links.

- gzip :
  Gzip reduces the size of the named files
  Gzip will only attempt to compress regular files

- gunzip :
  Compressed files can be restored to their original form using gzip -d or gunzip.
  If the original name saved in the compressed file is not suitable for its file system,
  a new name is constructed from the original one to make it legal.
  gunzip can currently decompress files created by gzip, zip,

Notes:

4.1.10 Archiving With tar Files

GNU tar' saves many files together into a single tape or disk archive, and can restore individual files from the archive.

- tar -cvf :
  This will create the new archive

- tar -xvf :
  Extract all files from archive.tar

- tar -tvf :
  This will list the contents from archive

- tar -rvf :
  This will append files to the end of an archive

Notes:

4.1.11 Filters

Filters are the programs, which read some input, perform the transformation on it and gives the output. Some commonly used filters are as follow.

- tail :
- sort :
- tr :
- wc :

Notes:

4.1.12 Patternmatching

Grep is pattern matching tool used to search the name input file. Basically its used for lines matching a pattern

- grep :

Notes:

Chapter 5

VIsual Editor

5.1 VI

vim is a powerful editor that has many commands. It's a screen editor allows you to see portions of your file on the reminal screen and modify characters and lines b line by simple typing at the current cursor position.

- To open a file Notes:

  vi opens a file in command mode to start mode. The power of vi comes from its 3 modes

- Escape mode (Command mode) Search mode File mode
- Editing mode Insert mode Append mode Open mode Replace mode
- Visual mode

Notes:

5.1.1 Cursor Movement

You will clearly need to move the cursor around your file. You can move the cursor in command mode. vi has many different cursor movement commands. The four basic keys appear below

- k :
- h :
- j :
- l :

Notes:

5.1.2 How to exit from a file

- :q
- :q!
- :wq
- :wq!

Notes:

5.1.3 Escape Mode (Command Mode):

In command mode, characters you perform actions like moving the cursor, cutting or copying text, or searching for some particular text.

Search Mode

vi can search the entire file for a givenstring of text. A string is a sequence of characters. vi searches forward with the slash (/) key and string to search. To cancel the search, press ESC .You can search again by typing n (forward) or N (backward). Also, when vi reaches the end of the text, it continues searching from the beginning. This feature is called wrapscan Instead of (/), you may also use question (?). That would have direction reversed

5.1.4 File Mode

Changing (Replacing) Text
Notes:

5.1.5 Editing Mode

Besides insert mode, vi employs a few other input modes. They all let you enter text; the only difference's where the insertion point is. Table 5 below describes the three most common modes: append, insert,and open. Two other text input modes are change mode and replace mode, both of which youve used.The mode indicator displays the current mode.
Notes:

Deleting Text

Sometimes you will want to delete some of the text you are editing. To do so, first move the cursor so that it covers the first character of the group you want to delete, then type the desired command from the table below. Notes:

Visual Mode

Visual mode helps to visually select some text, may be seen as a sub mode of the the command mode To switch from the command mode to the visual mode type one of

- Visual Mode
  Notes:

- Visual Block Mode
  Notes:

Chapter 6

Shell Scripting

6.1 Scripting

Any collection of shell commands can be stored in a file,which is then called as shell scripting. And programming the scripts is called shell scriting.Scripts have variables and flow control statements like other programming languages. Shell script are interpreted, not compiled. The shell reads commands from the script line by line and searches for those commands on the system

6.1.1 Where to use shell scripting

- System Administration
- Development
- Daily Usuage
  Notes:

6.1.2 Why to write shell script

Notes:

6.1.3 Some Special Characters

- ~ :
- $:
- & :
- ? :
- * :
- $# :
- $:
- $0 :

6.1.4 How to invoke scripts

Example: $ vi hello.sh
andtypethefollowinginsideit:
#!/bin/bash
echo Hello World
The first line tells Linux to use the bash interpreter to run this script.
We call it hello.sh
Then make the script to executable:
$ chmod 700 hello.sh
$ ./hello.sh

6.1.5 Variables

Notes:

6.1.6 White Spaces & Line breaks

Notes:

6.2 Single & Double & Back Quotes

- Single Quotes :
  Notes:

- Double Quotes :
  Notes:

- Back Quotes :
  Notes:

6.3 Arthimetic Evaluation

- Math & Calculations
  Notes:

- Avaible Operators
  Notes:

6.4 Conditions

6.4.1 if-then Condition

The if statement chooses between alteernatives,each of which may have a complex test.The simplest form is the if-then statement.
if test
then
expression
fi

6.4.2 if-then-elif-else Condition

Multiple elif blocks can be strung together to make an elaborate set of conditional responses
if [condition_A]
then
code to run if condition_A true
elif [condition_B]
then
code to run if condition_A false and condition_B true
else
code to run if both conditions false
fi
Notes:

6.5 String Tests

- =
- !=
- -n
- -z

Notes:

6.5.1 Numeric Tests

- -eq :
- -ge :
- -le :
- -ne :
- -gt :
- -lt :

Notes:

Combining & Negating Tests

- ! :
- -a :
- -o :

Notes:

6.6 Loops

6.6.1 Flow Control - for

An alternative form of the for structure is
General syntax:
for ((EXPR1;EXPR2;EXPR3))
do
echo "Welcome $i times"
done
First the arthimetic expr EXPR1 is evaluated.EXPR2 evaluated repeattedly untill it evaluates to 0. Each time EXPR2 is evaluated to a non-zero value, statements are execauted & EXPR3 is evaluated
Notes:

6.6.2 Flow Control - while

General syntax:
while [condition]
do
code block;
done
Any valid conditional expression will work in the while loop.

The structure is a looping structure. Used to execute a set of commands while a specified condition is true. The loop terminates as soon as the condition becomes false.If condition naver becomes false, loop will never exit
Notes:

6.6.3 Flow Control - case

The case statement compares the value of the variable ($var in this case) to one or more values (value1, value2, ...). Once a match is found, the associated commands are executed and the case statement is terminated.

Used to execaute statements based on specific values.Often used in place of an if statement if thee are a large number of conditions. value used can be an expression
Each set of statements must be ended by a pair of semicolon's. *) is used to accept any not matched with list of values. General syntax:
case "$var" in
value1)
commands;
;;
value2)
commands;
;;
)
commands;
;;
esac
Notes:

6.6.4 Functions

Writing functions can greatly simplify a program. If a chunk of code is used multiple times in different parts of a script, the code can be enclosed within a function and run using only the function name
General syntax:
function name()
exit 0

function name
exit 0

name ()
exit 0

Notes:

6.6.5 Arrays

Having an array of variables is of no use unless you can use those values somehow. This recipe shows a few methods for looping through the values of an array in the bash shell.Given the array definition:
names=(Jennifer Tonya Anna Sadie)
Notes:

6.6.6 Command Line Arguments

shell script can accept command-line arguments & options just like other Linux commands.Within your shell script, you can refer to these arguments as $1,$2,$3,.. & so on.
Refer to the examples in command directory
Then the command line arguments are execauted like
eg: ./no_arg.sh Emertxe Technologies Jayamahal
Notes:

Chapter 7

Assignments

7.1 List of Assignments

(Id) / Date	Assignment Topic
( ) _____	Write a basic script for printing all the files related information in present working directory(eg: size,permessions,& etc...)
( )	Read n and generate 1 1 2 1 2 3 1 2 3 4
( )	Read n and generate 1 2 3 4 5 6 7 8 9 10
( ) _____	Write a script for scp & ssh
( ) _____	Write a script for addition of two numbers for real numbers also
( ) _____	write a script for Arthimetic calculator using command line arguments
( ) _____	Write a script to compare larger integer values from a n number of arguments using command line arguments
( ) _____	Write a script to print the given number in reverse order eg: i/p=1234, o/p=4321
( ) _____	Write a script to delete a empty lines from a file
( )	Write a script to perform arthimatic operation on given number depending on the operator Eg:- i/p=1234+ then o/p=10

(Id) / Date	Assignment Topic
( ) _____	Read n and generate fibonacci nos. ≤ n
( ) _____	write a script to print the lenght of each and every string using arrays

( )	write a script to print chess board this two will print
	`Black [echo -e -n "\033[40m"] ],`
	`white [echo -e -n "\033[47m"] ]`

( ) _____	write a script to sort a given number in assending or decending order
( )	Write a script to print the following information 1. currently logged usre 2. your shell directory 3. home directory 4. os name & version 5. current working directory 6. number of users logged in 7. show all available shells in your system 8. hard disk information 9. cpu information 10. memory information 11. file system information 12. currently running process
( ) _____	Write a script to rename a file/directory replaced by lower/upper case letters
( ) _____	Write a script to rename current working directory with given name
( ) _____	Given album name and corresponding directory this scripts renames them properly by inserting index numbers. For example given file numbers (DSN001.jpg, DSN002.jpg as linux_class_photos_001.jpg, linux_class_photos_002.jpg etc.)
( ) _____	Read n and print the greatest fibonacci no ≤ n
( ) _____	Write script to print contents of file from given line number to next given number of lines. For e.g. If we called this script as script.sh and run as ./script 5 5 myfile, Here print contains of 'myfile' file from line number 5 to next 5 line of that file.
( ) _____	Display the longest and shortest usernames on the system (usernames are in the first field in /etc/passwd).
( )	Write a script to delete all the .swp files found in your system.

(Id) / Date	Assignment Topic
( )	Write a script to delete all .swp files from given directory.
( )	Write a script for generating random 8-character passwords including alpha numeric characters.
( )	Write a script that takes any number of directories as command-line arguments and then lists the contents of each of these directories.
( )	Write script called sayHello, put this script into your startup file called .bash_profile, the script should run as soon as you logon to system, and it print any one of the following messages depending on system time. • "Good Morning" (9 AM 1 PM) • "Good Afternoon" (2 PM 5 PM) • "Good Evening" (5PM 9 PM) • Good Night (9 PM 9 AM)
( )	Shell script to convert string lower to upper and upper to lower • Enter any string in lowercase : cRiCkEt • Lower to upper converted o/p : CRICKET • Upper to lower converted o/p : cricket
( )	The script should output each directory name prior to listing the files within that directory.
( )	Use an address range negation to print only the fifth line of your /etc/passwd file. Hint: Use the delete editing command.
( )	Use pipes or redirection to create an infinite feedback loop, where the final output becomes the input again to the command line. Be sure to stop this command before it fills your hard disk. (If you are having trouble, look at the documentation for the tail command.)
( )	Write a short script that outputs its process number and waits, allowing you to view the contents of the Linux /proc in another shell window. That is, this script should remain at rest long enough that you can view the contents of the process-specific directory for that process.
( )	Use a recursive function to print each argument passed to the function, regardless of how many arguments are passed. You are allowed to echo only the first positional argument (echo $1).

(Id) / Date	Assignment Topic
( ) _____	Write a script to determine whether a given file system or mount point is mounted, and output the amount of free space on the file system. If it is mounted. If the file system is not mounted, the script should output an error.
( ) _____	Write a script to show the output same as ls command. Directory name should pass using command line arguments.
( ) _____	Write a script to locks down file permissions for a particular directory.(Remove all permissions for groups and others)
( ) _____	Display the names of any filesystem which have less than 10% free space available (see the df command for this information)
( ) _____	Count the number of users with user IDs between 500 and 10000 on the system.
( ) _____	For each directory in the $PATH, display the number of executable files in that directory.
( ) _____	Write a script to search a user present in your system.
( ) _____	Write a script to replace 20% lines in a C file randomly and replace it with the pattern <—DEL—>.

Appendix A

Assignment Guidelines

The following highlights common deficiencies which lead to loss of marks in Programming assignments. Review this sheet before turningin each Assignementt to make sure that the it is complete in all respects.

A.1 Quality of the Source Code

A.1.1 Variable Names

- Use variable names with a clear meaning in the context of the program whenever possible.

A.1.2 Indentation and Format

- Include adequate white-space in the program to improve readability. Insert blank lines to group sections of code. Use indentation to improve readability of control flow. Avoid confusing use of opening/closing braces.

A.1.3 Internal Comments

- Main program comments should describe overall purpose of the program. You should have a comment at the beginning of each source file describing what that file contains/does. Function comments should describe their purpose and other pertinent information, if any.

- Compound statements (control flow) should be commented. Finally, see that commenting is not overdone and redundant.

A.1.4 Modularity in Design

- Avoid accomplishing too many tasks in one function; use a separate module (Split your code into multiple logical functions). Also, avoid too many lines of code in a single module; create more modules. Design should facilitate individual module testing. Use automatic/local variables instead of external variables whenever possible. Use separate header files and implementation files for unrelated functions.

A.2 Program Performance

A.2.1 Correctness of Output

- Ensure that all outputs are correct. Incorrect outputs can lead to substantial loss in grade

A.2.2 Ease of Use

- The program should facilitate repeated use when used interactively and should allow easy exit. Requests for interactive input from the user should be clear. Incorrect user inputs should be captured and explained. Outputs should be well-formatted.

Appendix B

Grading of Programming Assignments

- Total points per assignment = 10
- Points for timely/early submission = 1
- The source code is out of 3 points. The distribution of points is as follows:
  - (a) The existence of the code itself (1 pts)
  - (b) Proper indentation of the code and comments (1 pts)
  - (c) Proper naming of the functions, variables + Modularity + (1 pts)
- You get 4 points if the program does exactly what it is supposed to do.
- Two (2) points are reserved for the ease of use, the type of user interface, the ability to handle various user input errors, or any extra features that your program might have.