VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

EMBEDDED PROGRAMMING
ECE4025 – C1

Allen Ben Philipose – 18BIS0043 – DA 1 (Q2,3,6,8,10)

I Explain x Windows basics with examples

The x windows is a common system in UNIX-like operating systems, which is used for bitmap displays as a windowing system. This evolved to be the underlying framework for GUI environments currently found in modern-day operating systems.

"x Windows" involves concepts like drawing and moving windows on the display device and I/O device interactions with peripherals like a mouse or a keyboard. User Interface is not mandated by "x" though, it is done by individual programs.

x-based environments have a huge variance in visual styling and different programs will have radically different interfaces. Originated in MIT in 1984 as part of project Athena, the x-protocol has been on version 2, (X11) since September 1987. It is currently available free open source under the MIT license

x is used for remote graphical user interface, and input device capabilities. It is architecture independent and using a network terminal has the ability to interact with the display and any type of input device.

Xlib, the protocol client library written in C programming language, containing functions for interacting with an X server, provides the basic supporting frameworks for primitive GUI environments.

Unlike most other earlier display protocols, "X" is specifically used to connect to the display over network connections rather than being directly attached to the peripheral. It features network transparency, the programs running on a computer can be implemented on another display via a network with proper authorization.

X-server typically provides graphical resources and I/O events from keyboard or mouse to the clients, meaning that the server runs on the PC while client runs anywhere in the network and interacts with the user's computer to request the rendering of graphical content and receive inputs from peripherals.

X's network is based on X command primitives. This approach allows both 2D and 3D operations by an X client application which might be running on different computer to still be fully accelerated on the X-server's display

C program application that uses xlib to bring up a simple window with a greeting:

```
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xaw/Label.h>

int main (int argc, char* argv[]) {
        XtAppContext  app_context;
        Widget  toplevel, hello;

        toplevel = XtVaAppInitialize (
                &app_context,
                "XHello",
                NULL, 0,
                &argc, argv,
                NULL, NULL);
        hello = XtVaCreateWidget (
                "Hello Sir This is Allen",
                labelWidgetClass,
                toplevel, NULL);

        XtRealizeWidget (toplevel);
        XtAppMainLoop (app_context);
}
```

System's default widget set is being used.
Output is a window opened with the message "Hello Sir This is Allen" displayed in it.

II Difference between UNIX and Linux

Linux is a clone of unix, but, even though many operating systems were modelled after Unix, none of them got the popularity of Linux, which is currently spread over desktops, servers, smartphones and even some other electric smart devices.

A detailed comparison of Linux vs UNIX

Linux	Unix
Open-source operating system and is free to use by everyone	Operating system only to be used by its copyrighters i.e closed source software
Multiple distros like Ubuntu, Fedora, Redhat, etc...	Few derived OS like IBM AIX, HP UX and Sun solaris
Used by home user, developer or even students	Developed mainly for servers, mainframes and workstations
Found in servers, PC, Smartphones to even supercomputers	Primarily in servers and workstations

Linux	Unix
Any bugs posted by users can be rectified immediately by 3rd party developers	Bugs posted by users can only be removed by OS developers
High Security, 60-100 virus listed so far	Not as secure, 85-120 virus listed so far
Only the kernel	Complete OS package
Supports more file systems	Supports lesser file systems
Default interface is called BASH (Bourne Again shell) but own interfaces developed by distros	Originally used Bourne shell and later made compatible with other GUIs
Gnome and KDE are commonly used	Gnome is commonly used
Linux is freely distributed and priced distros are also cheaper than windows	Unix copyright vendors have all the rights to price the OS and tend to be costly

Linux being an open source software product, can be developed by people all around the world, hence the sharing and collaboration improves the software in terms of quality, security, good service and sometimes even perfectly tailored for user experience.

This approach also reduces the chances of having glitches or bugs and makes the operating system much more efficient and small in size and these have benefitted the Linux community greatly.

Unix was developed by AT&T labs and currently distributed by various commercial vendors and non-profit organizations. Primary three distributions are IBM AIX, HP UX and Sun Solaris.

Apple, the tech giant, uses unix to develop their initial versions of OS x. This was later modified with multiple iterations and is now shipped as macOS along with macbooks and iMacs

III Basic Shell commands with syntax and example

$ who
allen tty2 <time>
: 'who' command displays all users who are logged into the system currently.

$ pwd
/home/allen
: Print working directory displays the directory which the user is currently located in the file system.

$ ls
Downloads, Pictures, etc.
: List command shows all the items available inside the

$ echo "hello"
Hello
: Prints the message in terminal display

$ touch "allen.txt"
: Command creates a new file "allen.txt" in the current working directory

$ mkdir "allen"
: Make directory command creates a new folder in the current working directory

$ rmdir "allen"
: Remove directory command removes the folder in the current working directory only if its empty

$ cd "allen" : Change directory command changes the directory from current working folder to the named

$ cd .. : Moves up one directory from current

$ cal : Display calendar on the terminal

$ chmod +_ "allen.txt" : Change mode command changes the modes for a file to
+w : Permission to write
+r : Permission to read
+x : Executable file

$ file Downloads Directory : Command returns and displays the file type

$ sort file : Sorts contents according to ASCII rules

$ lpr allen.txt : Sent for printing

$ head allen.txt : Prints first 10 lines

$ tail allen.txt : Prints last 10 lines

$ clear : Clean up terminal

Black ink for computer output

a) Files and Directories

cat	Displays file contents
cd	Change directory
chgrp	Change file group
chmod	Change permissions
cp	Copy file
file	Determine file type
find	Find files
grep	Regular Expression search
head	Displays first few lines
ln	Creates softlink or old name
ls	List command
mkdir	Creates new directory
more	Displays data in paginated form
mv	Moves (Renames)
pwd	Print current working directory
rm	Remove (delete)
rmdir	Deletes directory (only if empty)
tail	Prints last few lines
touch	Create file

b) Compress files and directories

compress	Compresses files
gunzip	Helps uncompress gzipped files
gzip	GNU alternative compression method
uncompress	Helps uncompress files
unzip	List, test, extract from ZIP archive
zcat	cat a compressed file
zcmp	Compares compressed files
zdiff	Compares compressed files
zmore	File perusal filter for cat viewing of compressed text

These are the commands used to manipulate files and directories.

Explanation of following commands with syntax and examples:

i) cp

This command is used for copying files and its contents to another file name

Syntax:

```
cp  file1.txt  file2.txt
```

Example:

```
linux@allen: ~$ cp -r /home/d2  /home/1/d3
```

Here, we will copy the file from /home/ directory and pasting it to another directory /home/1/ and rename d2 as d3

ii) mv

This command is used to move a file from one location to another → that is to rename the path location of the file

Syntax:

```
mv  text1.txt  text2.txt
```

Example:

```
linux@allen: ~$ mv /home/d2  /home/1/d3
```

The file will be renamed, but the original file will not exist at /home/ directory

iii) mkdir

Command used to make directory

Syntax :

mkdir foldername

Example :

~$ mkdir Allen

Allen directory will be made in the existing directory

iv) rm

Command used to delete (remove) files

Syntax :

rm temp.txt

Example :

~$ rm /Allen/cat2.txt

Cat2 file from Allen directory will be removed

v) find

command to seach system and find a file

Syntax:

find file.txt

Example:

~$ find . -name notes.txt

This will find all files in the system which is named "notes.txt"

IV I/O Handling in Linux

All commands we run essentially produces 2 outputs

- → Command result : data the program is designed to produce
- → Program status and error messages

File descriptor numbers are used to help the shell identify the file

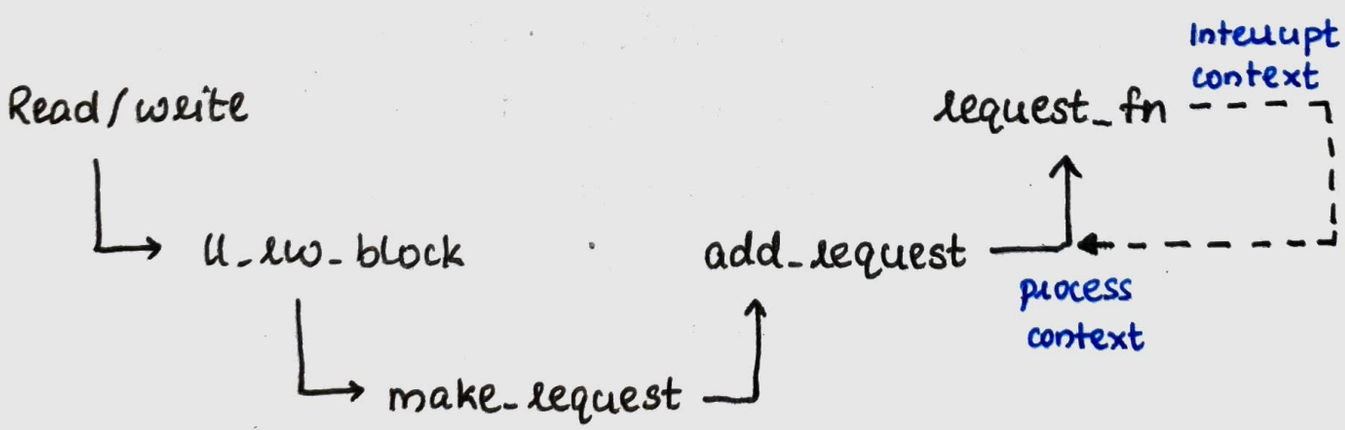
- → stdin or 0 : Connected to keyboard, programs read input from this file.
- → stdout or 1 : Attached to the screen, programs send output to this file.
- → stderr or 2 : Attached to Screen, errors are sent to this file.

Linux uses request structures to pass the I/O requests to the devices. All the block devices maintain a list of request structures. When a buffer is to be read or written, the kernel calls ll_rw_block() routine and passes it an array of pointers to buffer heads.

That routine in turn calls make_request() routine for each buffer. It first tries buffer clustering with the existing buffers in any of the request structures, present in the device queue.

A request structure consists of a list of buffers which are adjacent on the disk. This clustering is performed only for the drivers compiled in the kernel and not for loadable modules. If clustering is possible, no new request is created, otherwise a new request is taken from the global pool of structures and initialized with buffer and is passed to the add_requests() routine.

This routine applies the elevator algorithm using insertion sort based on the minor number of the devices and the block number of the buffers. If the device queue is empty, kernel calls the strategy routine i.e. the request_fn() of the driver, otherwise it is the responsibility of the driver to reinvoke it from interrupt context.



To allow accumulation of requests in the device queue, a plug is used. After accumulation, the task queue executes the unplug routine which removes the plug and calls the request_fn() to service the request.

Common operations of I/O Handling:

i) Redirect standard output to file is done by the top command.

eg. top -bn 5 > top.log

Use cat function to view the contents of top.log

ii) Redirect standard error to file is done by the ls command.

eg. ls -l /root/ 2 >> ls-error.log

iii) Redirect standard output and error to one file can be done by modifying the earlier syntax slightly

eg. ls -l /root/ > ls-error.log 2>&1

This was used to specify both the file descriptor numbers. The direct method can also be applied here

eg. ls -l /root/ & >> ls-error.log

iv) Redirect standard input to a file by using the cat command

eg. cat < domains.list

I/O redirection allows you to alter the input/output source/destination by using < and > operators.

V

Linux Shell or "Terminal"

Linux shell is an interface that allows users to execute commands and utilities in the operating systems related to Linux and some variants of unix.

Linux uses BASH (Bourne Again Shell) is a command line language interpreter for the GNU operating system. The concept is similar to the windows command line or windows powershell, but linux shells are a lot more powerful because it also functions as a scripting language as well as take advantage of the operating system benefits with its readily available numerous tools.

Shell can also be described as an environment in which we can run our commands, programs as well as shell scripts.

Shell prompt:

The computer, when ready for taking input, would prompt the user with the symbol $. If this is issued by the shell, it implies that the user can enter the input.

When the shell gives the prompt, user must enter the command and hit the 'enter' key to let the machine know it can read the entered statement. It determines the command you want to be executed by looking at the first word of the input.

Types of shells:

i) Bourne Shell: '$' will be default prompt
   - → Bourne shell (sh)
   - → Korn shell (ksh)
   - → Bourne Again Shell (BASH)
   - → POSIX Shell (sh)

ii) C shell: '%' will be default prompt
   - → C shell (csh)
   - → TENEX/TOPS (tcsh)

Graphical shells provide means of manipulating programs based on Graphical user interface by allowing for operations such as opening, moving, closing and resizing windows using peripherals like a mouse.

Usually shells are interactive, they accept inputs and execute immediately. For executing a series of commands routinely, we can write them in a file and execute them in shell. These program files are called shell scripts. This prevents the repetition of continuous input commands and avoids waiting time of the user after each command.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

EMBEDDED PROGRAMMING
ECE4025 – C1

Allen Ben Philipose – 18BIS0043 – Task 4 (Q1,4,5,7,9)

I Explain Command Prompt basic command

Command prompt is used in a text based or "command line" interface such as UNIX Linux or DOS. It is a symbol or series of characters at the beginning of a line that indicates that the system is ready to take input from the user. It prompts the user for a command.

The command prompt is often preceded by the current directory of the system the user is working with.

The default prompt in DOS would be the line C:\ which indicates the user is working at the root level of the main C drive.

The default prompt in UNIX would be the line ~ allen $ where "allen" is the name of the current user. The tilde (~) indicates that the current directory is the user's home folder.

A user can type commands in the command prompt such as cd / which means change directory to root folder. The "cd" command allows the user to browse through different directories of files on a hard disk or network.

There are dozens of other commands that a user can type which can be used to list, move, delete, copy files, run programs, or perform other operations. While "cd" command is the same both in DOS and UNIX, many other commands have different syntax.

Since a command requires a specific input, knowing syntax is very essential to use CLI - command line interface

GUI - Graphical User Interface has replaced CLI as the primary input medium which eliminates the requirement of learning programming syntax by an average user, but having experience with CLI would help in achieving special and quick operations which would be tougher or impossible to do in Graphical user Interface.

cmd command	Desciption
calls	calls a batch file from another one
cd	change directory
cls	clear screen
cmd	start command prompt
color	change console color
date	show/set date
dir	list directory content
echo	text output
exit	Exits command prompt
find	find files
hostname	display hostname
pause	pauses execution of batch file
runas	starts program as another user
shutdown	shutdown computer
sort	sort screen output
start	start an own video
taskkill	terminate process
tasklist	display applications
time	display/edit system time
timeout	wait only time
title	set title from input to prompt
ver	display OS version
w32tm	setting time synchronization

II Navigating file system, finding files, working with folders and text editing

Print working directory (pwd)

The command prints the current directory, telling where you are currently located in the filesystem. This command always prints out the absolute filepath from the root drive.

Absolute path is the full path from root directory (/) and it is best practice to use absolute paths when its used inside scripts.

`/usr/bin/ls`

Relative path is relative to the current working directory and can be displayed using dots

`../../usr/bin/ls`

Change directory (cd)

The command lets you change to a different directory. The default working directory is the home directory.

`/home/allen`

Absolute and relative paths can be better understood with the cd command. Similar to pwd, dots are used to represent relative paths to a particular directory.

`$ cd ..` Represents the directory, one level above the current working directory

`$ cd /home` Absolute path to home directory
`$ cd /` Absolute path to root directory

When the working directory is deeply nested inside the filesystem, the `(~)` is used to immediately return to the home directory. The `(-)` has a similar functionality and the command takes user to the previous folder/ working directory.

Nano editor in Linux system is used to edit files and can be accessed by the nano command, in linux terminal

- → `esc` normal mode
- → `i` insert mode
- → `q!` exit without saving
- → `wq!` save and exit
- → `w` save

`cat allen.txt` command can be typed in Linux terminal to read and print the contents of a file. This is the most commonly used method but there are other alternatives in bash script.

You can read a file line by line from command line using the code

```
while read line; do command; done <input.file
```

Here, while loop will reach each line of the file and store the content of the line in the variable $line which will be printed later.

Using Bash script we can construct this code,

```
#!/bin/bash
file = 'allen.txt'
i = 1
while read line; do

echo "Line No. $i: $line"
i = $((i+1))
done < $file
```

to print each line of the file seperately

III Different compression and archiving tools used in Linux

Compression: Way of reducing the size of a file on a disk using different algorithms and mathematical calculations. Files are formatted in certain ways that make their general structure somewhat predictable, even if their content varies. Furthermore, content itself is often repeated

→ Lossy conversion: Information lost to diminish Quality for saving space

→ Lossless conversion: No data lost

Archiving: Generally implies backing it up and saving it to a secure location, often in a compressed format. Data from servers was often backed up onto tape archives, which are magnetic tapes used to store sequential data and to do that efficiently, "tar" program was created. This helped to address and manipulate many files in the filesystem with intact permissions and metadata. It helps to conveniently distribute, store, back up and manipulate groups of related file.

Most of compression and archive tools can be operated via command line interface in Linux. Short commands can quickly compress data files to save storage space and bandwidth when sent over a network.

→ gzip :

GNU zip is one of the most commonly used compression methods. This tool plays a major role in web development, which is based on the deflate algorithm. Today, the application program in C can be used for extracting and packaging files in Linux, windows and MacOS

gzip builds 32,000 bytes data blocks which makes it feel obsolete in modern compression programs.

```
allen@linux:~$  gzip  allen/desktop/test.odt
```

Fast compression process, standard popular web server software but small block size and low compression ratio

→ bzip2 :

Loss free and high quality compression files with 3 layered compression method

- → Burrow's wheeler transformation (900000 bytes)
- → Move-to-front transformation
- → Huffman coding

With the help of bzip2 recover, partially damaged archives can be atleast extracted and unpacked. This along with strong compression rates benefitted users but the process was very slow.

→ p7zip :

Uses Lempel-Ziv-Markov Algorithm (LZMA) which is a further development on deflate algorithm. It has password protection and an optional encryption using AES-256. It has an excellent ratio of compression and duration but very high system requirements

→ lzop:

Lempel-ziv-oberhumer algorithm which offered very quick compression with high portability but lower compression speeds.

→ tar : Archives only

tar -cf archive.tar test.txt test2.txt

Example

IV Types of files and file management in Linux

How file management is done in Linux ?

→ Tree like structures of directories also called as folders
→ Most operations are performed on files
→ Command Line Interface gives a greater control during file management actions

7 operations on files in linux :

→ Files listing
→ Creating files
→ Displaying file contents
→ Copying a file
→ Moving a file
→ Renaming a file
→ Deleting a file

3 Types of files in Linux :

→ Regular files: Text, binary, images, etc... Such files can be created using the touch command, being majority of files in Linux/Unix Systems
→ Directories: Files that store a list of file names and other information related to those files. Refered to in windows as folders.
/ → Root directory at base of system
/home/ → User's home directory

→ Special files: Represents a real physical peripheral device such as a printer which is used for I/O operations.

One operation on files: Creating files

For creating a file, we can use the touch command. It will create and open a new blank file if the filename doesn't exist. If existing, the old file will not be affected.

Example:

```
linux@allen:   $ touch cat2.txt
linux@allen:   $ ls
               cat2.txt
```

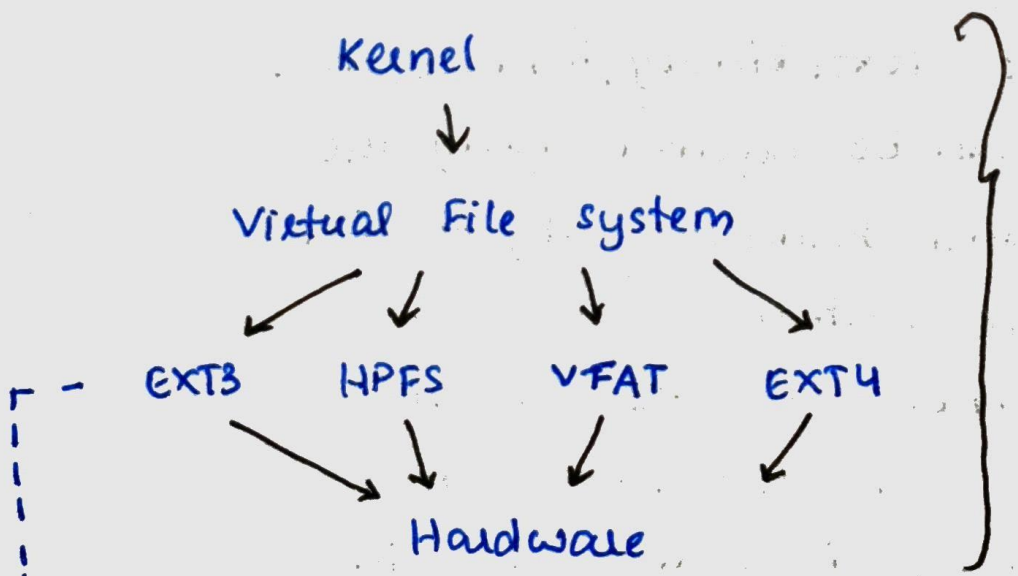
ls → function/command for listing all elements inside a given directory.

Different file system in Linux :

Structured collection of files on a disk drive or partition. A general purpose computer system needs to store data systematically to access files later.

Linux file system contains :

- Root Directory
- A specific data storage format (EXT3, EXT4, XFS, etc...)
- A partition for logical volume having a particular file system



Structure of Linux different file systems

- Different file systems

V File - locking in Linux programming

File locking is a mutual exclusion mechanism to ensure a file can be read/written by multiple processes in a safe way. This mechanism restricts access to a file among multiple processes and it allows only one process to access the file in a specific time, thus avoiding the interceding update problem. This also prevents reading of the file while it is being modified or deleted

Most operating systems support the concept of record locking, which means that individual records within any given file may be locked, thereby increasing the number of concurrent update processes. Database maintenance uses file locking, whereby it can serialize access to the entire physical file underlying a database.

Although this does prevent any other processes from accessing the file, it can be more efficient than to individually lock a large number of regions in the file by removing the overhead of acquiring and releasing each lock.

There are 2 types of file locking

i) Advisory Locking:

Not an enforced locking scheme, as it will work only if the participating processes are co-operating by explicitly acquired looks. Otherwise Advisory locks will be ignored if a process is not aware of locks at all.

ii) Mandatory Locking:

Mandatory Locking does not require any co-operation between the participating processes. Once a mandatory lock is activated on a file, the operating system prevents other processes from reading or writing the file.

- → All locks support blocking and non-blocking operations
- → Locks are allowed only on files and not directories
- → Locks are automatically removed when the process exits or terminates.
- → It's guaranteed that if a lock is acquired, the process acquiring the lock is still alive

These are few of the characteristics of File locking.