Exam 1 Review

# Notes From Instructor

## Unix in General

UNIX History and Philosophy (Linux Essentials: Chap 1-2)

The Shell and common commands (Linux Essentials: Chap 6-7) and Videos of Dr. Jey.

Understand different types of shells

Man Pages and Command Options (Linux Essentials: Chap 8)

File Systems and ownership/permissions (Linux Essentials Chap 15)

## Editors

**for BOTH vi and emacs**

Must know how to:

Create, open, edit, save, exit

## Shells

**Environment variables (AP in Unix: Chap 7.9):**

what they are, how to print them, how to set them, how to query a value

**I/O Redirection (Linux Essentials: Chap 10):**

stdout and stderr

append vs overwrite

## Scripts

**(Linux Essentials: Chap 12)**

what they are , how to create them, how to call them, how to change interpreter, how to make them executable

## Compiling

**Know the Stages of compiling**

Know how to get output from each stage using g++ flags

Know WHY you might want output at each stage

Know WHY you want to support multiple file projects

## Misc

Know the system() call

Zip/Tarballs

commands to create

commands to unpack

WinSCP

Putty

Screen and the –L option

## Program 1 Objectives

Designed to get your started with

Unix and compiling Programs

Multi file compiling that calls linker separately

stdout/stdin/stderr (AP in Unix: Chap 3.2/5.3)

Understanding shells

Shell Scripts that execute tasks

Shell Scripts that redirect I/O (Linux Essentials: Chap 10)

File Permissions on shells

argc/argv intro

## Program 2 Objectives

Designed to get your started with

Unix commands (copying and untarring)

Third Party Libraries

Download programs from reputable internet sites

untarring and installing

Compiler option flags needed to use

Command line parsing

argc / argv

TCLAP

C++ Standard Template Library

what are templates

Why do we need them

How are they used

C++ map class

What it is

How to use it

# PowerPoint Review

## Unix in General

UNIX History and Philosophy (Linux Essentials: Chap 1-2)

The Shell and common commands (Linux Essentials: Chap 6-7) and Videos of Dr. Jey.

Understand different types of shells

Man Pages and Command Options (Linux Essentials: Chap 8)

File Systems and ownership/permissions (Linux Essentials Chap 15)

### What is Unix

1. Family of Computer OS
2. Multitasking
3. concurrent tasks, not parallel
4. multiuser
5. many users can log into Software
6. Derived from AT&T original by Ken Thompson
7. History

### Unix History

1. Driver
   1. Needed to re-wrtie multix
   2. 'emasculated Multics' called Unics
   3. Final name Unix
2. Business Development
   1. AT&T -> Unix Systems Laboratories (USL) -> Novell -> X/Open -> The Open Group
3. Today
   1. UNIX ® takes the form of the worldwide Single UNIX Specification integrating:
      1. XPG4
      2. define all aspects of the operating system, programming languages and protocols which compliant systems should have.
   2. IEEE POSIX (Portable Operating System Interface)
      1. POSIX defines the [application programming interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API), along with command line [shells](http://en.wikipedia.org/wiki/Unix_shell) and utility interfaces, for software compatibility with variants of [Unix](http://en.wikipedia.org/wiki/Unix) and other operating systems
   3. ISO C
   4. Programming Language

### Unix Components

1. **Kernel:** The kernel is the
   1. master control program of the operating system
      1. Handles:
      2. memory management
      3. system calls
      4. low-level functions common to most programs
      5. drivers for controlling hardware.
2. **Shell**
   1. The [shell](https://kb.iu.edu/d/agvf) is an interactive program
   2. provides an interface between the user and the kernel
   3. The shell interprets commands entered by the user or supplied by a shell script
   4. passes commands to the kernel for execution.
   5. Shells available for use on Unix and Unix-like systems include
      1. sh (the Bourne shell)
      2. bash (the Bourne-again shell)
      3. csh (the C shell)
      4. tcsh (the TENEX C shell)
      5. ksh (the Korn shell)
      6. zsh (the Z shell)
3. **File system**
   1. Unix and Unix-like operating systems employ a hierarchical directory structure
   2. (i.e., inverted tree)
   3. with the root directory (/) at the top.

### Unix Linux Philosophy

1. Relationships of Programs over the Programs themselves
2. Many small commands
3. Written to be simple and chained
4. The chained commands can produce new behaviors when combined

### Linux

1. Os designed like Unix’s kernel
2. Most common name for any “Unix like” OS that uses Linux as their kernel
3. GNU is Linux, but not Unix
4. GNU is Not Unix, recursive acronym

### Shell Scripts

1. Files that contain shell commands
2. Able to execute the shell file and the shell will interpret the commands in the script file
3. Like dos (.bat)
4. You can access the shell through terminal prompt, also called Xterm
5. Also remotely over a secure connection with SSH

#### Different Types of Shells

1. Shebang
   1. #! interpreter [optional arguments]
2. sh (the Bourne shell)
   1. #!/bin/sh
3. bash (the Bourne-again shell)
   1. #!/bin//bash
4. csh (the C shell)
   1. #!/bin/csh
5. tcsh (the TENEX C shell)
   1. #!/bin/tcsh
6. ksh (the Korn shell)
   1. #!/bin/ksh
7. zsh (the Z shell)
   1. #!/bin/zsh

#### Common commands

1. Man
   1. Man “command”
   2. Bring up manual page of command
2. ls
   1. List files
3. cd
   1. Set current directory
4. pwd
   1. Print working directory
5. Mkdir /”directory name”
   1. Make directory
6. Cat
   1. Concatenate files to standard output
7. more / less
8. tail
   1. Print last lines of file to standard output
9. Grep
   1. Print lines matching a pattern
10. Echo
    1. Echo strings to standard output
11. Cp
    1. Copy source to destination
12. Diff
    1. Compare files line by line
13. Wc
    1. Word count
14. ln
    1. Make links between files
15. Chmod
    1. Change file mode bits
    2. chmod -R a+rwx /home/013/a/am/aml140830/Exam1/
    3. chmod -R a+rwx “directory”
    4. Recursivley set all files to a certain permission level
    5. #!/bin/bash
    6. Chmod -R a+rwx /home/013/a/am/aml140830/Exam

#### Man Pages

Shows on-line reference for the manual page of a command

Syntax: man <command name>

[] <- Indicates items are optional

- <- Short name for option

-- <- Long name for option

… <- indicates zero or more items

* + Examples
  + Defaults
  + Exit Codes
  + Environment Variable Info
  + Files that may be utilized
  + A “*SEE ALSO*” section that is useful

#### File Systems

Ownership /permissions

(d)(rwx)(r-x)(r-x)

total 137

drwxr-xr-x 2 root root 4096 Jul 13 2014 bin/

drwxr-xr-x 4 root root 1024 May 8 2014 boot/

drwxr-xr-x 14 root root 4140 Jan 13 16:00 dev/

drwxr-xr-x 118 root root 4096 Jan 13 16:00 etc/

drwxr-xr-x 3 root root 4096 Sep 25 2012 home/

lrwxrwxrwx 1 root root 33 Sep 25 2012 initrd.img -> /boot/initrd.img-3.2.0-29-generic

drwxr-xr-x 19 root root 4096 May 27 2014 lib/

drwxr-xr-x 2 root root 4096 Feb 2 2014 lib64/

drwx------ 2 root root 16384 Sep 25 2012 lost+found/

## Editors

for BOTH vi and emacs

Must know how to:

Create, open, edit, save, exit

### VI editor

vi – modal (command mode and insert mode)

#### Presentations

i/a go to insert mode

esc go to command mode

dw delete word

dd delete line

5dd delete 5 lines

d$ delete to end of line

/<something> search for <something>

!shellcmd <- execute command in shell

:q quit

:q! force quit

:w write

:wq write and then quit

#### My Notes

Vi filename

Opens editor with file

:! “command to run”

Runs command in current buffer

### Emacs

#### Presentations

Arrow keys move cursor

^f – forward one character

^b – back one character

^a – beginning of line

^e – end of line

^n – down one line

^p – up one line

^d – delete character

^s – search

^x^f – new file

^x^c – exit

Meta-> - end of file

Meta-< - Beginning of file

Meta-x revert <- revert changes

#### My Notes

^x^s – save

## Shells

### Environment variables (AP in Unix: Chap 7.9):

1. What they are
   1. A named object that contains data used by one or more applications
2. How to read
   1. ***printenv***
      1. Prints all as name and value pairs
   2. ***printenv [name]***
      1. Prints specific value only
3. How to set them
   1. ***bash***
      1. PATH= $PATH:$HOME/bin
      2. export PATH
      3. PATH= $PATH:$HOME/bin; export PATH
   2. ***csh***
      1. setenv PATH  $PATH:$HOME/bin
4. How to create them
5. How to call them
6. How to change interpreter
7. How to make them executable

### I/O Redirection (Linux Essentials: Chap 10):

#### Steam objects

1. Input
   1. keyboard
   2. stdin
2. output
   1. terminal screen
   2. stdout
3. error or diagnostic messages
   1. terminal screen
   2. stderr

#### Stream Redirection

Allows us to change where *stdin*, *stdout*, *stderr* get or put their data.

< - Uses file as stdin

>, >> - Creates or appends stdout to file

2> - Creates or appends stderr to file

append vs overwrite

#### Pipes

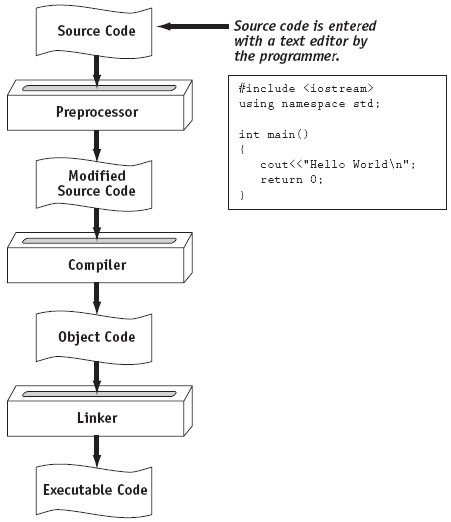
the standard output from one program is redirected as the standard input to another program

## Scripts

(Linux Essentials: Chap 12)

## Compiling

### Stages of compiling



### Each stage using g++ flags

-E

Stop after preprocessor

Output preprocessed source code

-c

Stop after Compiler, don’t link

Output object file

-o

lookup

g++ file1.cc

No flags set, so preprocess, compile and link to a.out

### Know WHY you might want output at each stage

Preprocessor

1. Edits the file so it might be textually different
2. Example would be replacing #define constants with their numerical values

**Before**  **After**

# define MYVAL 10

Int main (){

Return MYVAL;

}

# define MYVAL 10

Int main (){

Return 10;

}

Compiler

1. Must have all object code across project before linking
2. Build systems need 1 object file at a time

Linker

1. Create final executable file

### Know WHY you want to support multiple file projects

#### Object Oriented Design

Best practice uses multiple files in large project

## Misc

### System() call

### Zip

### Tarballs

1. Group of files collected together as one
2. Typically not compressed
3. Used to backup Linux systems

Create

1. Change directory so that you can see the folder you want to turn into a tarball
2. The “Program1.tar.gz” is your output file
3. The “Program1” (second arg) is the source folder
4. tar cfvz Program1.tar.gz Program1

-c, --create

create a new archive

-f, --file=ARCHIVE

use archive file or device ARCHIVE

-v, --verbose

verbosely list files processed

-z, --gzip

filter the archive through gzip

All the same output

1. tar cfvz Program1.tar.gz Program1
2. tar -c -f Program1.tar.gz -v -z Program1
3. tar --create --file=Program1.tar.gz --verbose --gzip Program1

Unpack

1. tar zxfv tclap-1.2.1.tar.gz

### WinSCP

### Putty

Screen and the –L option

## Program 1 Objectives

Designed to get your started with

Unix and compiling Programs

Multi file compiling that calls linker separately

stdout/stdin/stderr (AP in Unix: Chap 3.2/5.3)

Understanding shells

Shell Scripts that execute tasks

Shell Scripts that redirect I/O (Linux Essentials: Chap 10)

File Permissions on shells

argc/argv intro

## Program 2 Objectives

Designed to get your started with

Unix commands (copying and untarring)

Third Party Libraries

Download programs from reputable internet sites

untarring and installing

Compiler option flags needed to use

Command line parsing

argc / argv

TCLAP

C++ Standard Template Library

what are templates

Why do we need them

How are they used

C++ map class

What it is

How to use it

### Header File

#include <tclap/CmdLine.h>

### Program

Program2.cc and parsecmdline.cc

#include “program2.h”

### Shell Script

G++ -I ~/Program2/include/ -c ~/Program2/program2.cc

G++ -I ~/Program2/include/ -c ~/Program2/parsecmdline.cc