Review

- What is Linux
- What is GNU/Linux
- Introduction to C Programming
- C System Environment
  - Text editor, preprocessor, compiler, linker
- C compilers in Linux
  - gcc (GNU), cc (UNIX), c99 (POSIX)
- Linux System Roadmap
  - /bin, /usr/bin, /etc, /home, /root, …

Preview

- Linux System Roadmap
  - Libraries
- What is the shell
- Simple Bash Commands
- Redirecting Input and Output
- Pipeline
- Shell Scripts

Linux System Roadmap
(Libraries)

- Libraries are collection of precompiled functions.
- Standard libraries locations :
  - /lib and
  - /usr/lib
- A library filename start with lib, then follows the part indicating what library is (m for math library, c for the C library)
- Two Types of libraries
  - .a : static library
  - .so : shared library

Linux System Roadmap
(Libraries)

- Linker knows the standard libraries locations.
- We can also instruct the linker to search a library at a specific location by passing the name of library with full path.
  - gcc –o some some.c /usr/lib/libm.a
- Shorted standard library location –lm means use libm.a static library
  - gcc –o some some.c -lm

Linux System Roadmap
(Libraries)

- We can also direct special directory by using –L flag where specific library is located.
  - gcc -o some -L/usr/openwin/lib some.c –laa

: means compile some.c with library libaa.a which is located in /usr/openwin/lib directory

Linux System Roadmap
(Static Libraries)

Static Library (call Archives)

- collection of object files in a ready to use form.
- To use a function in a Library, need include header file in your program.
- We can create and maintain our own static libraries by using **ar** (archive) program and compiling functions separately with gcc –c.

Linux System Roadmap
(How to Make Static Libraries)

- How to make your own static libraries?
  1. Make files for reusable functions.
  2. Create object codes by using commend
     - gcc –c file1.c (create file1.o)
     - gcc –c file2.c (create file2.o)
     - gcc –c file3.c (create file3.o)
  3. Create a header file (foo.h) which contains function prototypes for reusable functions. This header file must be included in the program where the reusable function is used.
  4. Create a library by using arc commend

Linux System Roadmap
(How to Make Static Libraries)

```
/* bill.c */
#include <stdio.h>

void bill (char *arg)
{
   printf("Bill: You passed %s\n ", arg);
}
```

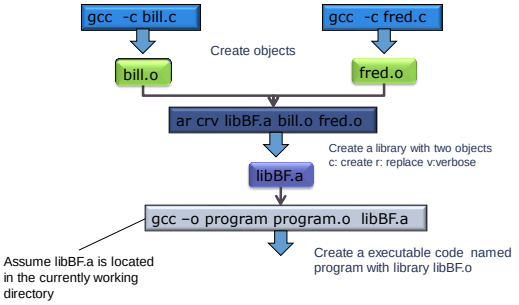
```
/*fred.c */
#include <stdio.h>

void fred (int arg)
{
    printf("Fred! You are passed: %d\n", arg);
}
```

```
/* A header file some.h
   for two function prototype
*/

void bill (char *);
void fred (int);
```

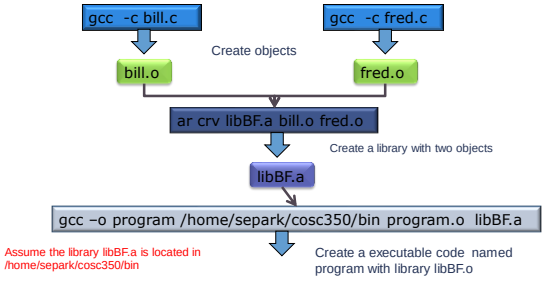

Linux System Roadmap
(How to Make Static Libraries)

```
/* program.c   which include main function*/
#include " some.h"

int main ()
{
   bill(" Hello world ");
   fred (100);
   return 0;
}
```


Linux System Roadmap
(How to Make Static Libraries)



Linux System Roadmap
(How to Make Static Libraries)



Linux System Roadmap
(Shared Libraries)

Shared Libraries

- Shared libraries might be stored in the same location as static libraries, but named with .so.
- Shared Libraries are the libraries that can be linked to any program at run-time.
- Once loaded, the shared library code can be used by any number of programs –which can save memory space.
- There are always only one copy of library in Memory.

Linux System Roadmap
(How to make Shared Libraries)

How to Create a Shared Library

- With following simple example program, we can shows how to create and use shared libraries in a program.
- We have three programs
  - shared.c :where sharable library functions are defined.
  - shared.h : function prototypes for sharable functions.
  - share_lib_ex.c : program which will use shared library functions.

Linux System Roadmap
(How to make Shared Libraries)

```c
// shared.c locate shared libary functions
#include "shared.h"
unsigned int add(unsigned int a, unsigned int b)
{
    return (a+b);
}
unsigned int subtract(unsigned int a, unsigned int b)
{
    return (a-b);
}
unsigned int mult(unsigned int a, unsigned int b)
{
    return (a*b);
}
```

```c
// shared.h function prototypes for shared libary functions
#include <stdio.h>
extern unsigned int add(unsigned int a, unsigned int b);
extern unsigned int subtract(unsigned int a, unsigned int b);
extern unsigned int mult(unsigned int a, unsigned int b);
```


Linux System Roadmap
(How to make Shared Libraries)

```c
//shared_lib_ex.c shows how to use shared libaries
#include<stdio.h>
#include"shared.h"
int main(void)
{
    unsigned int a = 7;
    unsigned int b = 4;
    unsigned int add_result, sub_result, mult_result;

    add_result = add(a,b);
    sub_result = subtract(a, b);
    mult_result = mult(a, b);

    printf("\n %u + %u = %u \n",a, b, add_result);
    printf("\n %u - %u = %u \n",a, b, sub_result);
    printf("\n %u * %u = %u \n",a, b, mult_result);
    return 0;
}
```


Linux System Roadmap
(How to make Shared Libraries)

- following two commands to create a shared library :
  - Following command compiles the code shared.c into position independent code which is required for a shared libary

    gcc –c –Wall –Werror –fPIC shared.c
  - Following command creates a shared library with name libshared.so

    gcc –shared –o libshared.so shared.o
- Following command compiles the shared_lib_ex.c code and tell gcc to link the code with shared library libshared.so. –L flag is used to tells the location of shared libraries.

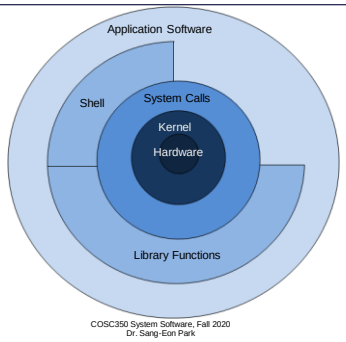
gcc –L/home/separk/Lecture/cosc350/lecture/lec1 –Wall shared_lib_ex.c –o shared_lib_ex -lshared

What is "the shell"?

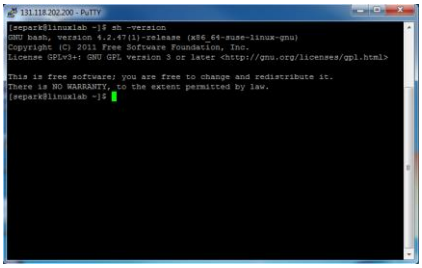
- A shell is a interface (command interpreter) between command and the Linux system (kernel).
- The shell is a program that takes your commands and gives them to the OS to perform.
- The most common Linux shell is named "Bash". The name comes from "**B**ourne **A**gain **SH**ell,"
- you can check the version of bash with

  sh -version or
  bash -version

What is "the shell"?



What is "the shell"?



What is "the shell"?

- On Linux standard shell is installed and it can be implemented by using symbolic link /bin/sh which is point to bash.

- There are several additional shell programs available on a typical Linux system: **ksh (***Korn shell*), **tcsh** (TENEX C shell) and **zsh (**The name zsh derives from Yale professor Zhong Shao)
- Users can pick the one they prefer.

What is "the shell"?

Terminal emulators

- They are programs that put a window up and let you interact with the shell.
- Most Linux distributions supply several, such as: **xterm, rxvt**, **konsole**, **kvt**, **gnome-terminal**, **nxterm**, and so on.

What is "the shell"?

- There are two primary ways to use the shell.
  - **interactively** – a user types a single command (or a short string of commands) in a terminal emulator and the result is printed out.
  - **shell scripts** – a user types anything from a few lines to an entire program into a text editor, then executes the resulting text file as a shell script

Simple Bash Commands

- **date**: display date
- **cal**: display calendar
- **whoami**: display user name
- **pwd**: display present working directory
- **cd** : change directory
- **echo** [option] string: display string on screen
- **man** [command]: display manual of the commend
- …
- http://ss64.com/bash/

Simple Bash Commands
(Listing Files)

- **ls** [-option]: display file name in the pwd (present working directory)
- **man** [commend]: display manual for the command
- **find** -name "*.jpg": list of files with .jpg suffix in current and all child directories.
  - find /home/separk -name "*.cpp" : list of file with .cpp suffix in /home/separk directory and all child directories
  - find –type d : List all the directory and sub-directory names:
  - find –type f : List all file in those sub-directories
  - find –type l : List all the links
  - find $HOME : List all files in your home directory

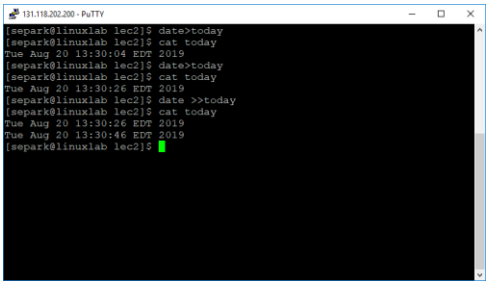
Simple Bash Commands
(Examining Files)

- The "file" command tries to identify files by examining their contents.
  file tux_small.png
- "cat" command displays the contents of a file on screen.
  cat zipcodes.txt
- "more " displays the contents of a file one screenful at a time.
  more zipcodes.txt
- "grep " print only those parts of a file you are interested in
  grep 10001 zipcodes.txt

Redirecting Output

- Normally the output from commands is printed on the screen by default.
- But using the symbol "**>**", you can redirect the output to a file.
- The symbol "**>**", to replace the content of any existing file having the name.
- To append new data to an existing file, use "**>>**" instead:

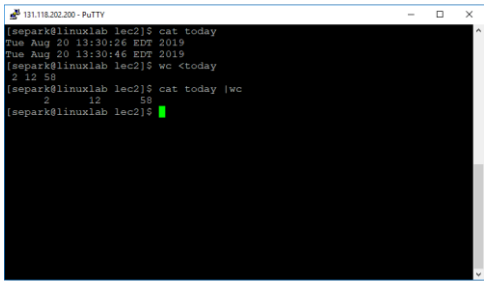
Redirecting Output



Redirecting Input

- The input defaults to the keyboard, the output defaults to the screen.
- To redirect the output to a file, use ">" or ">>" as shown previous note.
- To make the contents of a *file* serve as the input to a command, use "<":
- To make the output of a *command* serve as the input of another command, use "|" (pipe).
- wc command display three information
  - # of line , # of string, # of characters

Redirecting Input



Pipes

- *We can use a pipe "|" to* ***make a output of one command serve as a input to another command****.*
- This idea can be used to create a combination of commands to accomplish something no single command can do.

Pipes

ls –l >list.txt

sort list.txt > list.out

- Instead of previous command we can get a same result by using pipe

  ls –l |sort >list.out

- There is no limit to the permissible number of connected processes with pipes

Pipes

- View the contents of the /etc directory.

  ls -al /etc
- **less** allows you to view information one page (or screen) at a time.

  ls -al /etc | **less**
- We can view the result with sorted order of file name.

  ls -al /etc |**sort –k 9,9** | **less**
  - //Sort by 9th string

Pipes

A file **student.txt** file for students list taking COSC350, including student id, last name, first name, age, phone number

1234 Smith Chistine 27 410-980-2222
0235 Park Sangeon 35 410-230-1023
8989 Kim David 23 410-111-0009
7878 Smith Emily 34 410-898-0009
6756 Chab Jessie 24 410-786-0345
1345 Clark Joshua 27 410-897-2345
1243 Close James 22 410-887-2345
7845 Smith Jason 23 410-772-3678
3412 Davis Andrew 21 410-876-2347
4567 Figiel Charles 27 410-777-8877
1345 Smith John 26 410-666-8888

Pipes

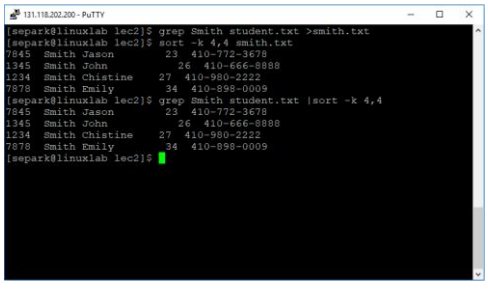
- If you wanted to find all the Smith and sort them by age, and save in a file "Smith.txt", you need sequence of commands without pipes.

  grep Smith student.txt > smith.txt
  sort –k 4,4 smith.txt

- But with pipes, we can get a same result.

  grep Smith student.txt |**sort** –k 4,4

Pipes



Shell Script

- A shell script is a plain-text file that contains shell commands.
- It can be executed by typing its name into a shell, or by placing its name in another shell script.
- To be executable, a shell script file must meet some conditions:
  - Need provide where bash sell program (interpreter) is located
  - Need change shell script mode to executable

Shell Script

- The file must have a special first line that names an appropriate command processor.
  - #!/bin/sh
- If this example doesn't work, you will need to find out where your Bash shell executable is located and substitute that location in the above example. Here is one way to find out:
  - **whereis** sh
- The file must be made executable by changing its permission bits. An example:
  - **chmod** +x (shell script filename)

Shell Script

- A shell script file may optionally have an identifying suffix, like ".sh". This only helps the user remember which files are which.
- One normally executes a shell script this way:
  - ./scriptname.sh

Shell Script

chmod: (Change Mode)

- Each file or directory has permission code called MODE.
- **chmod** changes the permissions of each given file or directory according to MODE.
- The MODE can be either an octal number representing the bit pattern for the new permissions or a symbolic representation of changes to make.

Shell Script

chmod: (Change Mode)

- Octal number representation
  - Three bits is needed to save octal number.
  - 000, 001, 010, 011, 100, 101, 110, 111
  - Each file mode composed with three digit octal number.
  - 777, 577 or,….
  - We can change a file mode by chmod with octal number representation

    chmod 755 filename

Shell Script

chmod: (Change Mode)

- Symbolic representation
  - Combination of letter 'a' (all), 'u' (user), 'g' (group) and 'o' (other) controls which **users**' access to the file will be changed:

chmod a+rwx filename
chmod a-rwx filename
chmod g+x filename

The Shell Programming

```
#!/bin/sh
# first.sh
# This file looks through all the files in the current
# directory for the string "main", and then print the
# name of those files to the standard output.

for file in *
do
       if grep -q main $file
       then
              echo $file
       fi
done
exit 0
```