CS 240 Programming in C

Unix/Linux Terminal Basics

February 20, 2024

Schedule

1. Terminal

Terminal

What is a terminal

- The UnixLinux terminal is also known as the command line, console, or shell with which we can manually execute commands by typing in the terminal with the Linux command line.
- We can also automate commands with the use of Shell Scripts.
- It offers an interface to run programs installed on the computer.

Example

```
rm hello.c
# remove the file hello.c in the current folder
# the first word is the name of the program, you want to run
# the second word is the command line argument for the progra
```

PATH

PATH

How does the terminal find the installed program for a command entered?

- The PATH environment variable is used by the operating system to locate needed executables from the command line or Terminal window

Example

```
echo $PATH
# echo is a program to display a string on the terminal
# $ is used to access the value of a variable in the shell
# so echo $PATH prints all paths set to the PATH
```

which path?

which

There are many paths set to the PATH, but which exactly is the path for the program?

- The which command allows users to search the list of paths in the $PATH environment variable and outputs the full path of the command specified as an argument.
- The command works by locating the executable file matching the given command.

Example

```
which echo
/usr/bin/echo
```

Absolute/relative path

Aside from the paths from $PATH, an absolute or a relative path can also be used for locating the program and run.

Absolute path

- Absolute path always starts with the root directory which is '/';

```
/usr/bin/echo $PATH
```

Relative path

- Relative path starts with the current directory which is '.' or the parent directory '..'

```
./hello.out
../hello.out
```

Home folder and navigation

Home folder

- When you launch a terminal, you are in the home folder which is also represented by $\sim$
- The current folder of the terminal is also called working directory

```
# pwd short for printing working directory
allen@DESKTOP-UV2S8G7:~$ pwd
/home/allen
```

Navigation

- To change to a new working directory, use the cd command with a directory path.

```
# cd [path to directory]
allen@DESKTOP-UV2S8G7:~$ pwd
/home/allen
allen@DESKTOP-UV2S8G7:~$ cd ..
allen@DESKTOP-UV2S8G7:/home$
```

Files

ls

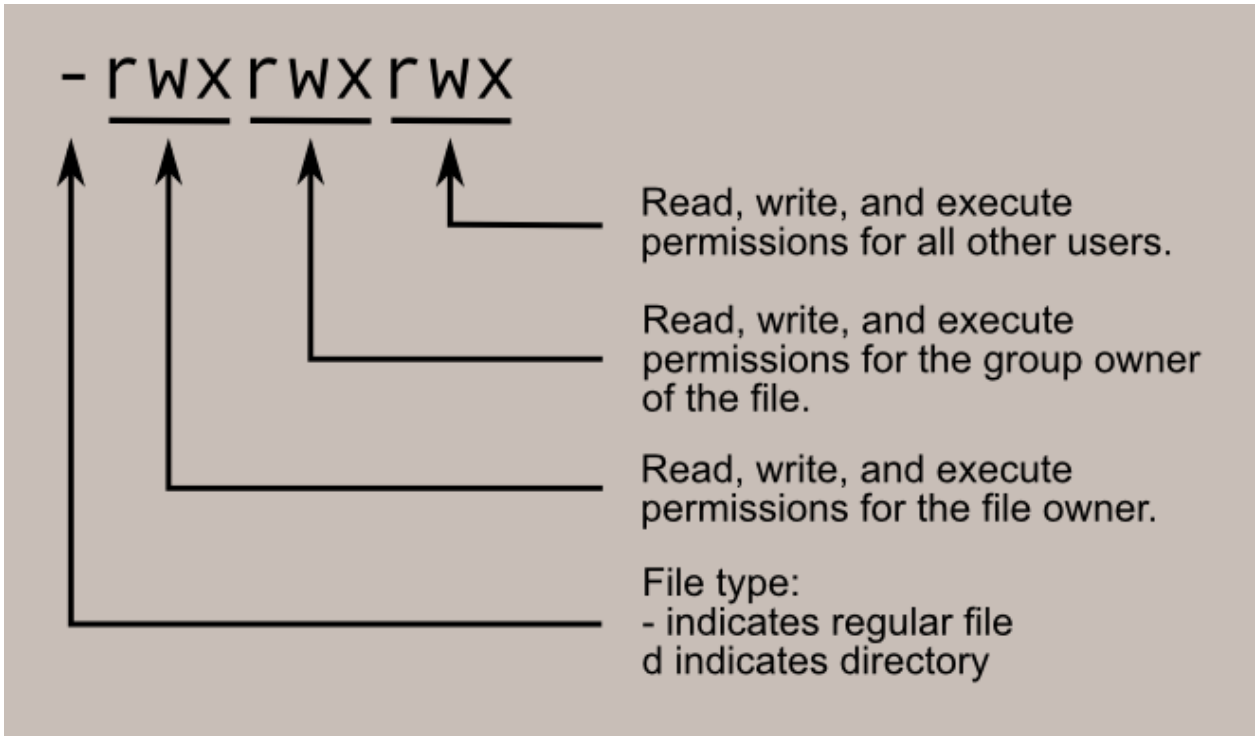
- The ls command is one of the many Linux commands that allow a user to list files or directories from the CLI.

```
allen@DESKTOP-UV2S8G7:/$ ls
bin   dev  home  lib    lib64   lost+found  mnt  proc  run
boot  etc  init  lib32  libx32  media       opt  root  sbi
```

File Permissions

ls -l

```
[~] > ls -l
total 12
-rw-r--r-- 1 allen allen    0 Feb 14 13:00 hello.txt
drwxr-xr-x 3 allen allen 4096 Feb 10 10:42 projects
```



Create Folder and Text File

Folder creation

```
[~] > mkdir cs240
[~] > cd cs240/
[cs240] >
```

Text file creation

```
[cs240] > touch main.c
[cs240] > nano main.c
[cs240] > cat main.c
#include <stdio.h>
int main(){
        printf("Hello world!");
}
[cs240] >
```

Compilation Example

Main compiling stages
source code $\rightarrow$ preprocessor $\rightarrow$ compiler $\rightarrow$ assembler $\rightarrow$ object code $\rightarrow$ linker $\rightarrow$ executable

```
gcc -E main.c  // print out the preprocessed code
gcc -S main.c  // produce the assembly code main.s
gcc -C main.c  // produce object code main.o (lib file)
gcc -g main.c  // produce executable for debugging

gcc -o main main.c // give a name to the output file
gcc -Wall main.c //  enables all the warnings in GCC
gcc main.c -lm // will link the standard math library

gcc -l  // is linking a library (we will come back to this)
```

Compiling to an executable

gcc -o [name of executable]

- If a C source file contains a main function, then that source file can be compiled into an executable.
- You will see the [x] executable on the attributes

```
[cs240] > gcc -o main.out main.c
[cs240] > ls -la
total 28
drwxr-xr-x  2 allen allen  4096 Feb 15 17:20 .
drwxr-x--- 11 allen allen  4096 Feb 15 17:12 ..
-rw-r--r--  1 allen allen    58 Feb 15 17:19 main.c
-rwxr-xr-x  1 allen allen 15968 Feb 15 17:20 main.out
[cs240] > ./main.out
Hello world![cs240] >
```

stdin, stdout

stdin, stdout

- By default, when a terminal gets launched, there are three file descriptors created they are stdin, stdout, and stderr.
- stdin by default is wired to keyboard input
- stdout by default is wired to terminal console.
- There are c functions like getchar to read from stdin.
- stdin, stdout can be redirected with redirection operations.
- Each standard IO has an EOF at the end, for stdin from the keyboard there are different shortcuts for the EOF for different OS

Redirection

I/O redirection

You can manipulate and change the default behavior of these three basic file descriptors by leveraging redirection.

Redirection with >

- command > file: Sends standard output to <file>

Append with »

- command » file: Appends standard output to a file

Redirect with <

getchar, scanf

- There are some functions in C can get characters and strings from standard input like getchar and scanf.
- Standard input has to have an special ending character which is called EOF (contant integer literal with value of -1)
- By default it will take the keyboard as the stdin device
- There is a keyboard EOF.
- But we can also redirect the stdin from a file by <

```
[cs240] ./a.out < ../data/ints.txt
7
```

Keyboard EOF

In Windows, Control+Z is the typical keyboard shortcut to mean "end of file", in Linux and Unix it's typically Control+D. But on Windows, it may not work.

Pipe

Piping

Piping involves passing output from one command as input to another.

```
[cs240] ./a.out < ../data/ints.txt
7
[cs240] cat ../data/ints.txt | ./a.out
7
```