Introduction to Unix shell & system calls in C

Week 02 – Lab 2

Link: https://goo.gl/F35fD5

What is Unix shell?

- Command-line interface to the operating systems
- Many implementations: bash shell, original Unix shell, Bourne shell, ksh, csh etc.
- Bash shell is the most used and default shell for Linux

General things about shell

- $ is a prompt sign (sometimes %)
- First word you type - a program to run (command)
- Commands can have **arguments**
- Arguments that control the operation are called **flags** and are proceeded with a dash
- Commands can be executed consequently, even combined together (output of one program can be an input of another)

Arguments and flags

- $*cp src dest* - invokes cp program with two arguments. Makes a copy of *src* file and saves it to *dest*
- *$head -20 file* - invokes head program. Prints first 20 lines of *file*. *-20* here is a flag, *file* is an argument

Wildcards

- To make it easier to specify filenames and search patterns, shell accepts **magic characters** or **wildcards**
  - * (asterisk) - matches all character
  - ? (question mark) - matches one character
  - [abcde] (list of chars in square brackets) - matches any of the characters inside the brackets
- Examples:
  - ls *.c - list all the files with .c extension
  - ls [ape]* - list all the files that start with ‘a’ or ‘p’ or ‘e’

Stdin, stdout, stderr

- All user input goes to a file called stdin (standard input). Programs launched from shell have read access to this file, i.e. have access to user input
- All output of programs have write access to stdout and stderr files, stdout for normal output of program and stderr for errors. Writes to stdout or stderr go to the screen

Redirecting input/output

- It is possible to redirect standard input or standard output
  - > (greater then) redirects standard output
  - < (less then) redirects standard input
- Example:
  - *$ls* > *list.txt* - saves list of files in the current directory to list.txt file
  - *$head -20* < *list.txt* - displays the first 20 rows of list.txt file

Pipelines

- It is possible to make an output of the first program an input of the second one
- "|" (pipe symbol) is used to link multiple programs into a pipeline
- Examples:
    - *$sort <file.txt* | *head* –30 - display the first 30 lines of sorted file.txt

Man

Almost each shell command has a supporting documentation manual

- $*command* --help
- man *command*

- *Example:*
  - *$man cp*
  - *$cp --help*

Exercise 1

- Display the number of files in /usr/bin directory that start with *s*. Hint: *$wc -l* command counts the number of lines in the input
- Display last 3 lines of sorted /usr/share/tmp/dummy.txt file
- Display number of processes containing “getty” in their names currently running in the system. Hint: use $*grep* <*pattern*> to search the input for a pattern and $*ps ax* to display all running processes

Shell scripts

- It is possible to put a list of shell commands in a file and then start a shell with this file as standard input. Files containing shell commands are called **shell scripts**
- “;” is used to separate one program from another in a script
- “&” is used to send a program into background and execute the next command
- Example:

```
$ echo "ls /usr/bin | wc -l; echo "Done"" > script.sh
$ sh script.sh
```

Exercise 2

- Write a shell command that will save the current time in hh:mm:ss format to the file called time.txt
- Write and execute a shell script that will output “Time to wake up” in 10 seconds after script launch

System calls in C

- A system call is a request for service that a program makes of the kernel. The service is generally something that only the kernel has the privilege to do, such as doing I/O. Programmers don’t normally need to be concerned with system calls because there are functions in the GNU C Library to do virtually everything that system calls do. These functions work by making system calls themselves

System calls in C cont

- Example of explicit system call

```
#include <unistd.h>

int main()
{
    char data[128];

    if(read(0, data, 128) < 0)
     write(2, "An error occurred in the read.\n", 31);

    return 0;
}
```

- read is a system call used to read data into a buffer.

Important! Save history to a file

- $mkdir week2
- $cd week2
- $history > history.txt

Extra shell exercise

- Write a C program that executes “ls -la” shell command and prints the output to the user using printf(). Compile and run the program. Name the file ex1.c

Extra C exercises:

- http://c.learncodethehardway.org/book/
- Ex 8, 9, 15, 16