CSci 4061: Introduction to Operating Systems

Recitation 1

September 11, 2017

Unix Shell Commands

- Basic Unix Shell commands
- Unix Man pages
- Pipes and redirection
- Environment Variables
- UNIX Shells

Unix Shells

- Shell is a command Interpreter
- Interface between User and the Operating system
  - Interactive, text based interface for users to make commands/requests to the operating system to perform some operation
- Various different Shells
  - csh (C shell), and tcsh (Tenex C Shell)
  - sh (Bourne shell), bash (Bourne Again Shell)
  - korn shell
- The default shell in CSELabs is tcsh

What is my current shell?

- To find your current shell, open terminal and type:
  - echo $0
  - echo $SHELL

How do I change my current shell?

- To change your current shell, open terminal and type:
  - Type the name of the shell (eg: bash, tcsh)

Basic Shell Commands

Some Basic Shell Commands

(*We will try these commands soon*)

• `pwd,cd, mkdir,ls`	Directory related commands
• `cp, mv`	Copy/Move file
• `chmod`	Change file permission
• `rm`	Remove a file
• `more,less,cat`	Read a file
• `head, tail`	Read some beginning or end parts of a file
• `diff`	Check for differences in two files
• `wc`	Counts words, lines, characters in a file
• `find`	Search for a file
• `man`	The manual pages

  – This an important command to remember!

Try the following

- `man ls`
- `man man`

Some Basic Shell Commands

(Now try these commands)

- `pwd`
  - print current working directory
  - This should print your current home directory
- `ls`
  - List all files and directories in the current directory
- `mkdir examples`
  - Make a new directory (folder) named “examples”
- `cd examples`
  - change current directory to examples

Some Basic Shell Commands

(Now try these commands)

- `cat > testfile`

```
Hello! This is a test file.
Cntrl-D  (CONTROL+D)
```

  - This will create a file named ‘testfile”
  - cat command joins several files, but here we are taking input from the keyboard.
  - We are redirecting output to “testfile”
- `ls -l`
- `rm testfile`
  - This will delete “testfile”

Some Basic Shell Commands

(Now try these commands)

- Once again create “testfile” using cat as before.
- `mv testfile ..`
  - We moved test file to parent directory
  - Here “..” means going one level up in directory tree
- `ls -l`
  - “testfile” is not in the directory listing
- `cd ..`
  - Change working directory to one level up.
- `ls -l` You will find the testfile here.
- `rm -i` Asks you before deleting the file (Important)

Some Basic Shell Commands

(Now try these commands)

- Once again create “testfile” using cat as before.
- `cp  testfile testfile.bak`
  - We copied the testfile to testfile.bak
  - “testfile” is not in the directory listing
- `diff testfile testfile.bak`
  - diff checks differences in two
  - In this case both are the same
- `ls –l` You will find the testfile and testfile.bak here.
- `wc testfile`
  - Prints number of lines, words, and characters in testfile

Unix Man Pages and man command

man command

- Using the “man” command you can find more information about various system functions and commands as described in the Unix man pages.
- Try : man ls
  - See and try the following options

  ls –l (long listing)

  ls –s (list file sizes with file names)

  ls –d (list directories)

  ls –a (list all file including hidden files)

  ls –t (what will it do???)

man pages

- Now try typing man followed by any of these commands – cd, mkdir, chmod, ls, rm, cp, mv, pwd, cat
- Try the different options available
- Press ‘q’ to get back the prompt

- man pages for system calls and library functions (e.g. printf, fork) will give important details such as:
  - Parameters
  - Return value or error codes
  - Header files that must be included in the C program code

Unix man pages

Unix man pages are organized in different sections.

Section	*Contents*
1	*General commands, such as* ls, cp, date, chmod
2	*System calls*
3	*3C C-library functions; 3F Fortran library*
4	*Special files and devices*
5	*File format conventions*

Try

```
man –s 3C printf
man –s 2 fork
```

File permissions

- Type ls -l and it will show you the permissions of every file in the present working directory

```
rwxrw-r-- 1 user group 2364 2007-08-30 11:31 sample
```

- Permissions owner group size Date of Creation Time File Name

- Permissions are Read, Write, Execute
- Assigned to three groups of users:

(1) Owner of the file	rwx	means owner can Read, Write, Execute
(2) Group Members	rw-	means group members cab Read, Write
(3) Others	r--	mean others can only read

Changing File permissions

chmod is used to assign suitable permissions to files

- Two ways to change permissions:
  - Make direct assignments specifying octal values
  - Add/remove permission using symbolic names

Try the following

- `chmod 700 testfile`
- `ls  -l`
- `chmod g+rw testfile`
- `ls  -l`

Pipes and I/O Redirection

I/O Redirection

- Using *output redirection*, the output of a command can be sent a file instead of standard out (terminal screen).
- `ls -l > filelist` Output redirection
- Using *input redirection*, the input to a program can be obtained from a file.
- `wc < filelist` Input redirection

Pipes

- In Unix, the pipe command can be used to send to output of one command as the input to another command.
- `ls –l  | wc`
- Here the output of the command “ls –l” is given as input to the “wc” (word/line count) comand.

Try

- `ls –ls | sort -n`

Redirection and Pipes

- Try out the examples:
- `ls -l > list`
- `cat list`
- `more list`
- `less list`

- The | (pipe) operator is used to direct one command's output to a following command's input
- `head -3 example.txt | tail -1`

Environment Variables in Shell

Environment Variables

- An environment variable is a <u>global variable </u>within a shell environment
  - It is visible to all programs running within an environment
  - These variables are inherited by all child processes.
- Type env at the console to get a list of all the environment variables
- There are several system-defined environment variables.
- A program can define new variables.
- It is also possible to define <u>local variables </u>that are known only within the current shell environment

Setting Environment Variables in TCSH

- You can set the value of global environment variables by using the `setenv` command
- `setenv JHOME /home/mydir/java`
- `echo $JHOME`
- To define a local variable use `set` command
- `set name=John`
- `echo $john`
- `set lastname=Doe`
- `set fullname=$name.$lastname`
- `echo $name $lastname $fullname`
- Now start a new shell by typing tcsh and see the values of these variables.

PATH Environment Variable

- In Linux/UNIX, the names of directories that have executables are stored in an environment variable called PATH
- When you type a command, these directories are searched in the order given in the PATH string.
- `echo $PATH`
- `setenv PATH .:$PATH`
- `echo $PATH`
- This adds the present working directory to the PATH variable string

Setting Environment Variables in Bash

- A variable is declared and initialized to a value by a simple assignment statement
- `JHOME=/home/mydir/java`
- `THOME=/home/mydir/temp`
- `echo $JHOME`
- `echo $THOME`

- A variable is made global using the export command.
- `export JHOME`
- Start a new bash shell and see which variables are still defined.

Additional utility programs: find, stat

find

- Searches a directory tree for files
- Examples
  - find $HOME –name “*.c” -print
  - find $HOME -mtime -1 -exec ls -l {} \;
  - find $HOME -name "*.txt" -size 100 -print
  - find $HOME-size 0 -print
  - find .bashrc -type f
  - find $HOME -perm 711 -print

Stat

- Display file or filesystem status.

- Examples
  - cat > teste.txt

- stat -c%s teste.txt

- stat teste.txt

- Similar command in **Tcsh**

```
set file = teste.txt
set file_des = `ls -l "$file" `
@ file_size = $file_des[5]
echo $file_size
```