CSC521 UNIX Lab

Commands and their arguments
Enter a command, followed by zero or more arguments. Each argument is separators by spaces. Each command is able to interpret the arguments however it sees fit.

Type the following command, describe its output:
$ cut -d: -f1 /etc/passwd

The "-d:" parameter tells the cut command to use the colon as a field separator. The "-f1" parameter tells the cut command that you want the first field.
Change the "-f1" parameter to "-f1-3", what does this give you?

Environment Variables
Your UNIX shell keeps track of your "environment." Your environment is composed of a set of variables and their values. Your environment will remain constant throughout your shell session, unless you change the shell variables. Running new programs creates a new copy of your shell's environment.

Type the following command, describe its output:
$ set

$ echo $HOME
$ cd $HOME

$ ZZZ=1234
$ echo $ZZZ
$ set

When the shell sees a word beginning with "$", it interprets this word as a shell-variable substitution. The value of that variable will be used in place of the word. For example, when setting a variable, we don't use the "$" notation, but when reading it, we do (see above).
**Compound Commands**

UNIX shells allow multiple commands to be run at the same time. They may be cooperative, or they may be independent.

The pipe "|" symbol (shift-\ on a keyboard) is used to connect the output of the first command to the input of the second command. For example, the “grep” command expects at least one argument that is the string to search for. It will output any lines that contain that string. The cut command has already been used in a previous example.

Type the following command, describe its output:

```
$ grep false /etc/passwd | cut -f1-3 -d:
```

Now:

```
$ cut -f1-3 -d: /etc/passwd | grep false
```

**Shell Redirection – UNIX Plumbing**

UNIX allows commands to be connected to each other, such that the output of one command is the input of another.

- `cmdline >` - redirects standard output from cmdline to a file, clobbering the old file, create if new
- `cmdline >>` - redirects standard output from cmdline to a file, appending the old file, create if new
- `cmdline 2>` - redirects standard error from cmdline to a file,...
- `cmdline 2>>` - redirects standard error from cmdline to a file...
- `cmdline < file` – connects standard input of cmdline to be the contents of the file (so every readline / gets will be a line from the file)
- `lhs | rhs` - connects the standard output of lhs to the standard input of rhs

**Wild Card in Command Lines**

UNIX allows the use of special characters to indicate a pattern instead of an exact match.

- `*` - match 0 or more characters
- `?` – match 1 character

Examples:

- `ls *.c` – list all files that end in “.c”
- `ls c*` - list all files that begin with “c”
- `ls c*.c` – list all files that begin with c and end with “.c”
- `ls c?.c` – list all files that begin with c, end with .c, and have exactly one character between the first c and the dot.
The UNIX shell is responsible for “globbing” the wild-cards. For example, the “ls” program does not get the argument including the “*”, instead, the shell looks at all file-names that match, and substitutes this list in the arguments for ls. So, instead, ls will have possible many arguments instead of a single wild-card.

Questions:
Consult the reference cards and the previous examples to answer the following questions:

1. Display the “HOME” environment variable

2. List all the files in your home directory

3. Create a new directory spring2017/csc521/labs/lab1/ in your home directory

4. Change directory to lab1

5. Display the calendar for the year 1066AD

6. Copy all files in the “/home” directory to your lab1/TEST folder
7. gzip every file in the directory (hint: use a wild-card)

8. Print the name of every file in the directory without using the “ls” command (hint: use wild-cards)

9. Find the pathname for a command

10. Display only the differences between two files

11. List all processes

12. Pipe the output of the previous command into “less” to display the data page by page

13. Get the process-ids (and only process ids) of every process in the system owned by you