# CSE410 - Advanced Programming In UNIX

# Assignment 1

**Assignment goal**

**You will get experience using the Linux bash shell, using vim, and writing a short script.**

**All the questions below are designed to help you acquire the skills needed to solve common problems.**

**Submission deadline: September 21, 2015 Midnight.**

**Documentation**

**In addition to the lecture notes, you may find many useful reference for completing this assignment.**

**http://www.makeuseof.com/tag/5-downloadable-books-to-teach-yourself-linux/**

**There is also a bash manual (http://www.gnu.org/software/bash/manual/bashref.html) available on the web**

**Command line text editor (vim or nano): http://www.openvim.com/**

## Getting ready

Download the file: hw1.tar.gz.

Extract all the files for this assignment using the following commands:

Step1: decompress the tar file. Note that question 3 below asks you to use Google to figure out the definition of a "tar file".

```
> gunzip hw1.tar.gz
```

Step2: extract the files from the tar file

```
> tar -xvf hw1.tar
```

You should now see a directory called `hw1`.

If you see it, you are ready to start the assignment.

# 1. Simple commands and customizing your environment

**Note on auto-completion**: when you type commands or file names, try to type the first few letters only and press the `TAB` key. The shell will try to automatically complete (finish typing) the file name or command for you.

## *Basic file and directory operations*

Inside directory `hw1`, you should see the following list of files:

```
animals.txt
dog.txt
file1.txt
file2.txt
hw1.sh
irun4ever
irun4ever.c
mouse.txt
very-long-file.txt
```

a) Re-organize these files into the following directory structure using **at most four bash shell statements** that can be executed **from within the hw1 directory**. Your statements must not include the `;` operator (which executes two separate statements on one line) or the `&&` or `||` operators (which join two commands together with a boolean and/or condition).

```
hw1/pb1/animals.txt
hw1/pb1/dog.txt
hw1/pb1/mouse.txt
hw1/pb1/very-long-file.txt
hw1/pb4/irun4ever.c
hw1/pb4/irun4ever
hw1/pb6/file1.txt
hw1/pb6/file2.txt
hw1/pb6/hw1.sh
```

Save your four (or fewer) statements in a file called `problem1.txt`. Indicate that this is the answer to Problem 1a.

b) In one shell statement, make a backup copy of `hw1` called `hw1-back`. This statement should be executed from your home directory. Add your statement to the file `problem1.txt`. Indicate that this is the answer to Problem 1b.

c) Change the permissions on file `hw1.sh` to make it executable. That is, typing "`./hw1.sh`" from within the `~/hw1/pb6/` directory should execute the script. Add your statement to the file problem1.txt. Indicate that this is the answer to Problem 1c.

d) In one shell statement executed from within the `~/hw1/pb1/` directory, put the first 5 lines of file `very-long-file.txt` into a new file called `very-long-file-header.txt`. Add your statement to the file `problem1.txt`. Indicate that this is the answer to Problem 1d.

## *Shell variables and customizing your environment*

In file problem1.txt, please answer the following questions:

e) What is the value of your environment variable called `HOME`? What is the meaning of this variable?

f) What is the value of your environment variable called `PATH`? What is the meaning of this variable

g) From within directory `~/hw1/pb6/` execute the commands:

```
> hw1.sh
```

```
> ./hw1.sh
```

Why does the first command fail but the second one succeed? Hint: Think back to the previous question.

h) Modify your `.bashrc` to change the `PATH` environment variable to also include the directory "`.`" (current directory).

The `.bashrc` file should be directly in your home directory: under `~/.bashrc`. If it does not exist, you need to create it. After modifying your `.bashrc`, to see the effect of your changes, you need to open a new subshell by typing `bash` on the command line or you can simply execute "`source ~/.bashrc`" (we will talk about the differences

between invoking another `bash` and using `source` in class). Remember that if you open a new subshell by typing `bash`, you later can exit that subshell by typing `exit`.

As explained in the Linux Pocket Guide, page 33 ("Tailoring Shell Behavior"), the contents of the `~/.bashrc` file are read and executed when you open a new shell. However, when you first log in, a different file, called `~/.bash_profile` is read and executed. To make sure you see the effects of your changes even in your login shell, you can put the following `.bash_profile` file in your home directory. This file simply reads and executes the commands inside your `.bashrc`.

With your change in place, both commands from question (g) above should work.

### *Combining commands*

i) What is the difference between the following two commands?

- `mkdir 'whoami'`
- `mkdir` `` `whoami` ``
    - Hint: watch-out for the direction of the quotes.

j) What is the difference between the following two commands?

- `ls ~/hw1/pb1/*.txt | grep dog`
- `grep dog` `` `ls ~/hw1/pb1/*.txt` ``
    - Hint: watch-out for the direction of the quotes.

Remember to add your answers to all above questions to the file problem1.txt.

## 2. Using manpages

The command `man` displays an online manual page or *manpage* for a given program.

Use the command `man` to see the documentation for `wc`. The documentation should include information about `wc`'s options. Use `wc` with

at least one possible option. In a file called "`problem2.txt`" write the exact command that you used (including all options and arguments). Give a one sentence explanation of `wc`and the option(s) that you used.

Note: another way to get information about the options of a program is to invoke the program with the option `--help`. For example: `wc --help`.

# 3. Using Google

Google is a great source of information that you must learn to use.

Using Google, find a short definition for a "tar file".

In a file called " `problem3.txt`", write the definition that you found along with the URL of the page where you found the information.

# 4 - Processes

Launch the program `irun4ever`. This C program executes an infinite loop that prints a message every second.

Suspend the program by typing `^Z`

Examine the list of all processes that you are running:

```
> ps ux
```

Find the process id (pid) of `irun4ever` using one of the commands below:

```
> ps ux | grep irun4ever
```

```
> ps -C irun4ever
```

Find the process id of the currently executing instance of `irun4ever`. Note that the process id will change every time you launch `irun4ever`. This is the reason that you need to look it up using `ps`.

Kill `irun4ever` using the command `kill`.

Check that the program has been killed by typing "`fg`" to resume the suspended job by putting it back into the foreground.

In a file called `problem4.txt`, write the exact command that you used to kill the process.

# 5 - Aliases

Create an alias for the command `rm` such that executing `rm` prompts the user for a confirmation before actually removing the file. With your alias, typing:

```
> rm mouse.txt
```

Should trigger the following prompt:

```
> rm: remove regular file `mouse.txt`?
```

Hint: Try "`man rm`" or try using Google to find the appropriate option for the `rm` program.

Put your solution in a file called `myalias`. Running `source myalias` should successfully add your alias to the shell.

# 6 - Short script

We provided you a short bash script called `hw1.sh`, that takes one argument as input, prints it, and exits.

Try to execute `hw1.sh`. Make sure you have the execute permission on the file.

Based on `hw1.sh`, create a script called `merge.sh`. The script should take 3 arguments (let's call them f1, f2, and f3). The high-level idea is to merge the content of files f2 and f3 and write the output to file f1. More specifically, the script should work as follows:

- It treats all arguments as filenames.
- If a user specifies fewer than three arguments, the script prints an appropriate error message to `stderr` and exits with return code 1.

    - Hint: look at what we did for the case of fewer than one comand line argument.
- If a file (or directory) named f1 exists, it prints an error message to `stderr` and exits with return code 1.
- If either f2 or f3 does not exist, it prints an error message into file "`errors.txt`" and exits with return code 1.
- It concatenates f2 and f3, sorts the result, eliminates duplicates, and writes the final output into file f1.
- It prints a message that includes the name of f1 *in quotes* and the content of the first and the last line in f1.
- Your script may not produce any files other than f1 or "`errors.txt`".

Example. Executing your script on the provided files, file1.txt and file2.txt, as follows:

```
> merge.sh dst.txt file1.txt file2.txt
```

should produce the following output:

```
> First line in "dst.txt" is a

> Last line in "dst.txt" is z
```

And the final content of `dst.txt` should be:

```
a
b
c
d
e
f
g
h
w
z
```

# 7 - Extra credit

Modify your `.bashrc` so that your command line prompt looks as follows:

```
servername:path_to_current_directory username$
```

For example, if your username is `happyjoe` and you are in directory `~/hw1/` on `attu1`, your prompt should look like:

```
attu1:~/hw1 happyjoe$
```

# Submission

You should turn in the following files:

- problem1.txt
- problem2.txt
- problem3.txt
- problem4.txt
- myalias
- merge.sh
- Your .bashrc

**In Piazza Submission process:**

**Type: note**

**To: instructors**

**Folder: assignment N (N is your assignment number)**

**Summary: Your name and ID, Assignment N**

**Details: small description and attachment in zip**