**ECS 40** **Written Homework (30 points)** **Spring 2016**

Due: April 13th **in lecture by 9:50**.

1. (8 points, 2 points each) Write a UNIX command line that uses **find** to
   a) move all files in the computer that were modified within the last 24 hours to the *today* directory under your parent directory. You may assume that you have root privileges for this question.
   find / -type f -mtime -1 -exec mv {} ../today \;
   b) locate all files in your account named *a.out* or *core* and remove them interactively. You cannot assume that you are in your home directory.
   find ~ \(-name "a.out" -o -name "core" \) -exec rm -i {} \;
   c) locate all files larger than 5 MB in your ecs40 directory tree and print just their name and size, e.g.

```
10421760 ecs40/cusp/aside.exe
6768349 ecs40/cusp/CUSP.out
```

   find ecs40 -type f -size +5M -print "%k %p\n"
   d) change all directory permissions in your current directory tree so that only you can access and use them.
   find . -type d -exec chmod go-rwx {} \;
2. (1 point) Write a UNIX command to select the second to last line in a file named hw.txt.
   sed '1d' hw.txt
3. (1 point) What is the difference between a *wild card* and a *regular expression*?
   1. Wild card is used for system command line (globbing, the name of the file). Regular expression is for the pattern of the content of files. 2. Regex pattern must be enclosed in brackets, whereas wildcards do not have to. 3. Period in regular expression has meaning of any character, wild cards use a question mark instead.
4. (1 point) Why aren't the following commands equivalent? *grep "^[^a-z]" foo* and *grep –v "^[a-z]" foo*
   If a foo contains blank lines. grep -v ^[a-z] foo will print out blank lines, but grep "^[^a-z]" foo will not.
5. (1 point) How do you print to the screen all nonblank lines that don't begin with #, /*, or // in a file named hw.txt?
   cat hw.txt | grep -v '^\s*$' | grep -v ^# | grep -v '^\s*/[/*]'
6. (2 points) Write a UNIX command line to open the vi editor with each C program file (ending with .c or .h) in the ecs40 directory tree that contain the words **#include <stdio.h>** Note that there may be any number of spaces on either side of the '<' and '>'.
   find ecs40 -type f \( -name '*.c' -o -name '*.h' \) -exec grep -l '#include\s*<\s*stdio.h\s*>\s*' {} \; -exec vi {} \;
7. (1 point) Write a UNIX command line to create an archive file of the contents of your p1, p2, and p3 directories. The name of the archive file should be Pfiles.gz, and it should be able to be decompressed using gzip –d. As the files are stored you want to see their names printed to the screen.
   tar czvf PFiles.gz p1 p2 p3
8. (2 points) You want to be able to type "LL" in your current C Shell command prompt and want "ls –l" to be actually executed. How would you go about achieving that?
   alias LL 'ls -l'
9. (2 points) The C `puts(s)` function writes its const char* parameter to the screen, and appends a newline character. It is more efficient than using `printf("%s\n", s)`. Write a UNIX command line that will read foo.c, and produce foo2.c that has calls to `printf` properly replaced with calls to `puts` where appropriate.
   sed -r 's/printf\("%s\\n",\s*/puts\(/g' foo.c > foo2.c
10. (1 point) Assume that the number of compiling errors when you type `make` is huge. You decide that you want to store them in a file so you can look through them more easily. What C Shell UNIX command line would you type to store the errors in the file named errors.txt?
    (make > stderr) > & errors.txt

More questions on back!

11. (5 points) The ruptime utility displays the host status of local machines. An excerpt from running it is below. Write a UNIX command line that will start a secure shell session (though it will be waiting for your password) on the CSIF "pc" lab computer that has the lowest load average. CSIF "pc" lab computers all start with "pc" and then are followed by a number. If more than one of the CSIF "pc" computers has the lowest load average, then you may open any one of those. Hint: look at the man pages of ruptime.

ssh `ruptime -l | grep "^pc[0-9]*\s*up" | tail -1 | awk '{print $1}'`

```
[cs40a@pc15 ~]$ ruptime | head
coecstest2  down 170+18:54
iceman      down 183+21:44
lect1       down 183+21:42
pcupstairs  down  70+14:17
pc1           up      2:43,     1 user,   load 0.00, 0.02, 0.05
pc10          up      2:49,     1 user,   load 0.07, 0.05, 0.05
pc11        down  43+12:32
pc12          up      2:46,     1 user,   load 0.00, 0.02, 0.05
pc13          up      2:40,     1 user,   load 0.01, 0.02, 0.05
pc14          up      2:49,     1 user,   load 0.01, 0.02, 0.05
[cs40a@pc15 ~]$
```

12. (2 points) Given the following facts, where endless.sh is running for a very long time in the current shell session. Describe the steps you would take within the currently running shell session to terminate endless.sh. Remember, you are not allowed to start another shell session to solve this.

```
[ssdavis@lect1 p1]$ cat endless.sh
#! /bin/bash
trap INT TERM
find / -name "what?" &> /dev/null
[ssdavis@lect1 p1]$ endless.sh
```

Step1:
Ctrl + Z
Step 2:
use ps commad to display the runing shell. Read it PID
setp 3
Kill -KILL <PID>
Then the program will be killed

13. (3 points) Write a UNIX command line that prints out information about the five root processes that have used the most CPU time.

top -u root | head -12 | tail -5