Name: ____________________________

UNIX Shells

1. What is a "shell" ?

2. Determine the "official" name for each of these

   - sh
   - jsh
   - ksh
   - csh
   - bash
   - tcsh
   - ash

3. What shell are you currently using?

4. How do you change your default shell?

5. How do you change your currently running shell (not permanently)?

UNIX Plumbing

The plumbing operators allow you to redirect output to/from a file or to another program:

- command > file - run command, send its *standard output* to file
- command < file – run command, read *file* as its *standard input*
- command 2> file – run command, send its *standard error* to file
- command1 | command2 – run command 1 and 2, send command 1's standard output to command 2's standard input
- command 2>&1 – run command, merge its standard error to its standard output

1. Use the *find* command to list all of the files in a directory

2. Use the find command as before, but pipe the output to one of the *pagination* programs (*more* or *less*) to view the output one page at a time:

3. Show the output of the *ps* command, but only look for lines that contain bash

4. Use the *git* command line utility to list all of the available branches that include the string "release-to-test"

5. The command to find lines of a file that contain the text "monty burns" (ignoring case), but then only show the 3rd, 4th, and 8th column of the text using "~" as a delimiter

Programming with *sed*

Find one of the millions of references on the *sed* tool then answer these questions (some of these are basically Hacker Rank questions, so theres some easy Hackos here). Remember, *sed* expects its input to come from standard input, and it writes its output to standard output – so you'll want to use the pipe operations:

6. Replace the text "Atlanta Falcons win" with "New England Patriots win", ignoring case and all occurrences on a line

7. Replace the text "Atlanta Falcons win" with "New England Patriots win", ignoring case and only occurrences as the start of a line.

8. Use *character types* to match two or more commas in a row and replace them with one comma

9. Use *regular expression grouping* to match a ship.edu email address (e.g. aA1234@ship.edu) and re-write it to be a "cs.ship.edu" email address, but in all lower case: aa1234@cs.ship.edu.

Programming with *awk*

Hacker Rank also has a section *awk*, which may be useful and vaguely similar:

10. Write an *awk* script that can verify when a line contains exactly 4 fields, separated by spaces (see HR: Awk #1)

11. Write an *awk* script that will compute the average of three numbers (fields 2, 3, 4) and print the first field, a colon, and the average on a line (see HR: Awk #2)

UNIX Environment Variables

Lookup the following environment variables, describe what they do and when you would need to change them:

12. PATH
13. TERM
14. LD_LIBRARY_PATH
15. DISPLAY
16. HOME
17. PS1
18. TEMP
19. EDITOR
20. PAGER
21. Write the command to set an environment variable to a new value:

22. Write the command show the value of an environment variable

23. Write the command to show all environment variables

24. Write the command to delete an environment variable

25. The "bash" shell program interprets the prompt specially, find a description of the special commands you can use in your prompt, and make yourself a cool prompt below. Show the "set" command, and then also show what it looks like when you use it:

UNIX Shell Scripting

Use "bash" to write the following as shell scripts (there are thousands of shell script refs on the internet)

26. Print all the odd numbers from 1 to 99 on the terminal

27. Print a times table from 1 to 10 on each dimension

28. Write a shell *function* that will take two numbers and determine if a is a multiple of b, then use that function to write another function to determine if a number *a* is prime, and then use a for loop to display prime numbers between 2 and 31.

29. Loop over all of the files in a directory and if the file contains the text "bob", add it to a shell variable. After processing all files, display the final list of files, all on one line. The shell script should produce no other output.

30. Consult a manual on the "test" command, and then write a bash shell function that will loop over all files in a directory whose named ends in ".c". For each C file, use the test utility to see if there is a matching ".o" file, if there is, look to see if the .c file is newer than the .o. If the .o doesn't exist or the .c file is newer, then compile the .c file into the .o file. If the compilation breaks, abort the shell script. Finally, look to see if all the .o files are newer than the name of an executable given on the command line. If the executable is not present, is not executable, or is older than any of the .o files, recompile the executable using all of the .o files in the directory. *Hint: this one might go better if you use shell variables.*

31. Perform some background research and find out the name(s) of the shell script that bash will execute when you log in, and when you log out. Modify the shell script to change your prompt to something fun (your choice), modify the path to include your own personal bin directory, and create an alias for the vi command to be edit.

UNIX Job Control

32. Show how to run a command in the background:

33. Describe how to kill a command that is currently running:

34. Describe how to suspend a command that is currently running:

35. Describe how to put a suspended command into the background:

36. Describe how to restore a suspended command to the foreground:

37. Describe how to run a command and have it continue to run after you logout: