CS 279 - Exam 1 Review Suggestions - Fall 2014

last modified: 8-24-14

- You are responsible for material covered in class sessions and homeworks; but, here's a quick overview of especially important material.
- You are permitted to bring into the exam a single piece of paper (8.5" by 11") on which you have **handwritten** whatever you wish on one or both sides. This paper must include your name, it must be handwritten by you, and it will **not** be returned.
  - Other than this piece of paper, the exam is closed-note, closed-book, and closed-computer.
- This will be a pencil-and-paper exam, but you will be reading and writing UNIX/Linux commands, statements, and expressions in this format. There will be questions about concepts as well.
- Your studying should include careful study of posted examples and notes as well as the homeworks (and posted example solutions) thus far.

Basic UNIX/Linux commands and conventions

- Commands discussed in class, including `cat`, `cd`, `chmod`, `cp`, `echo`, `ls`, `man`, `mkdir`, `more`, `mv`, `pwd`, `rm`, `rmdir`, `sort`
  - AND the common command options that go with them (those used in class or course work)
  - what is a command you could use with these to try to access their UNIX manual pages?
  - you will likely be asked to write the command line you would use to accomplish a given task
  - you could also be asked questions about these commands
- you should understand what the `history` command does, and what it does when it is given an integer command line argument
- the only uses of `grep` that you are responsible for on this exam are the following:
  - using `grep` with a simple pattern (just letters, digits, maybe a period) and a file name, in which case it outputs the lines in that file containing that pattern
  - using `grep` with a simple pattern in a pipeline, in which case it outputs lines in the previous command's output that contain that simple pattern
- Remember that you can use the `which` command to show which version of a command you are running (to locate it in your PATH)

UNIX/Linux files and their characteristics

- The UNIX file system structure and absolute and relative paths
  - what is the name for the root directory of a UNIX file system?
  - given information about one or more directories and their contents, you should be able to give

the absolute pathname for a particular file, or the relative pathname for a file starting from a given directory;

- what are the 3 kinds of files in the UNIX file system?
- what is a pathname? what is a filename? Understand and be able to use the nicknames `~` and `.` and `..` (by themselves, and within pathnames)
- what is an "invisible" file? Why is it considered to be "invisible"? How is such a file named? How can you view it, anyway? What is the typical/traditional purpose of such a file?
- Creating, Deleting, and Modifying Files and Directories
  - how can you make a copy of a file? ...copy a collection of files into another directory?
  - how can you rename a file? ...move a collection of files into another directory?
  - what are three examples of text editor programs often available on UNIX/Linux systems?
  - what are some of the numerous ways to create a file in UNIX?
  - how can you remove a file? ...a directory? (note that `rm` without the `-r` option won't remove a directory, and that `rmdir` can only be used to remove an empty directory)
- File and Directory Permissions -- User(owner)/Group/Other(world), Read/Write/Execute
  - how can you view a regular file's permissions? how can you view a directory's permissions?
  - be able to use `chmod` to set or change a file's permissions - you should understand both octal and symbolic notation for this, although you'll usually be given the choice of which you use
- umask
  - you should know what the umask (file creation mask) is used for, and how it is used
  - you should know how to use the `umask` command to get the current umask value (in both octal and symbolic forms); what is the important thing to know about these two forms? How do they relate to each other?
  - given the permissions given to a new file by a program, and the umask value, you should be able to determine the resulting permissions for that new file
- What is an i-node? What is an i-node number? How can you view files' i-node numbers?
  - what is some of the information in an i-node?
  - is the actual data for a file stored in the i-node?
- What is a hard link? What is a soft link? What are the differences between these?
  - should be able to create both of these;
  - with either, what happens if I change one of the files involved (using `vi`, say?)
  - with each, what happens when the "original" file is deleted?
  - what are the restrictions on hard links?
- two common approaches to file limits

  - What is ULIMIT? what does the `ulimit` command do? How, then, does this approach work?
  - How does the quota approach work? How does it differ from the previous approach mentioned? What does the `quota` command do? What kinds of information does its output include?

UNIX processes and jobs and job control

- what is the real user id of a process? the effective user id of a process? what are the significance of these? (also real group id and effective group id)
- know the three file descriptors associated with each process -- standard input, standard output, and standard error
  - by default, what is standard input set to come from? ...what is standard output set to go to? ...what is standard error set to go to?
  - how can these each be redirected in the command line?
- what does `ps x` output? what does `jobs` output? What are some of the differences between what these output?
  - which would you use to see the background jobs for the current shell?
  - which would you use to see all of the current processes you own, regardless of the shell they were started in?
  - which contains process ids?
- Foreground and Background Execution
- Starting, Listing, and Killing Jobs; should be able to kill processes you own as well
- should be familiar with several ways to indicate a background job that you would like to now execute in the foreground; should be familiar with job identifiers for background jobs
- should be able to describe and write commands for starting a process in the background, putting it in the background from the foreground, and changing the status of a `Stopped` job to `Running`
- job - a collection of one or more processes [in the current shell]
  - should understand: foreground process, background process, running background process, stopped background process, current job, previous job
  - significance of &, +, -

intro to the bash shell

- What is the significance of the `PATH` environment variable? What is it used for?
- What is Command History? You should know at least one way to rerun previous commands;
- Control Characters -- ^C, ^D, ^Z, ^U
  - ^D - one of the ways you can log out of a UNIX/Linux shell session
  - ^C - tries to interrupt the foreground process (and if it does, the process is no more...)

  - ^Z - tries to suspend the foregound process (and if it does, the process is Stopped in the background, able to be resumed)
  - ^U - lets you cancel the line you're typing at the shell -- ("erases" the command in process, puts you back right at the prompt with an empty command line)
- Input and Output Redirection, Appending and Redirecting standard error - should understand what these mean, and how to do them
- Piping (|)
  - should understand what this is, how to do it, why it is useful, and should be able to read and write commands using this
  - what is a filter? What are the benefits of filters? Note that the `sort` command is an example of a filter (it accepts standard input, and outputs the lines given to it in sorted order); `grep` is also an example of a filter, because it can accept standard input, and can, for example, output only those lines containing the given pattern
- Shell Metacharacters -- `*`, `?`, `[ ]`, etc.
  - example possibility: given a pattern and several file names, you would say for each whether they would be matched or not
  - or, given a directory's contents and a pattern, and you would list which would be output by an `ls` command using that pattern
  - or, given a pattern, you might need to give an example of a filename that it would match, and one that it would not match;
  - and of course you might have to give a pattern that would match specified files
- every process (including your current shell) has a collection of environment variables associated with it, that can be used or set by that process;
  - know how to use the `env` command to see those currently in effect
  - know how to view the value of a given environment variable using `echo` and `$`
  - be aware that a child process has its environment variables set to be a *copy* of its parent process's environment variables
    - (so, it can use its copy of these, and can change them, but when it is complete its environment variables disappear and any changes thus disappear, too)
    - you should know how you can change the value of an environment variable
  - environment variables you should be familiar with at this point: `SHELL` (contains your login shell), `$0` (your currently-running shell), `TERM` (your terminal type), `HOME` (your home directory), `PATH` (the search path -- list of directories to be searched -- for commands)
- should understand, and be able to read and write, the 3 kinds of "quotation" discussed
  - what happens when a \ precedes a character? (two general possibilities) What will be echoed when `echo`'s argument includes this?

- what are the meanings of characters within a double-quoted string? What will be echoed when `echo`'s argument is such a string?
- what are the meanings of characters within a single-quoted string? What will be echoed when `echo`'s argument is such a string?

• should be able to write simple `bash` shell scripts;

- what should be the first line, to ensure that the commands are run using the `bash` shell?
- how do you write comments in `bash`?
- what permissions does a shell script need to be able to execute it? how can you execute a given shell script?
- how can you specify that the shell script be run as part of the current shell, instead of being run in a new little subshell (`source`)? What are some possible implications of running a shell script using the `source` compared to executing it by just giving its name?

intro to vi and emacs

• for BOTH `vi` and `emacs`:

- ...you should know how to create a new file or open an existing file for editing
- ...know what you need to do (if anything) to start entering or editing text,
- ...how to save your work, and
- ...how to exit

• know what `vi`'s two modes are, and how to switch between them

• know how to open up multiple screens in `emacs`, and how to move your cursor between them

concepts and history and miscellany

• should know what UNIX and Linux are...!

• should know how to ssh to nrs-labs

• what is POSIX? Why is it significant (what is its potential benefit?)

• what does a shell do/what is it used for? what are some common UNIX shells?

• what is the special user name for the superuser?

• what is the UNIX kernel, and what are its basic duties? What is a daemon?

• know that UNIX signals have a numerical code and a name

• 3 possible outcomes of a process: success, failure, abnormal termination

- can tell which via its exit status: 0 - success, < 128 - normal "failure" exit status, >=128 - abnormal termination (e.g., because of an interrupt or other signal) -- often 128 + the signal code number