

CS 279 - Final Exam Review Suggestions - Fall 2012

last modified: 12-10-12

- This Final Exam is **comprehensive** -- so, you should use the Exam 1 and Exam 2 review suggestions (as well as Exam 1 and Exam 2) as part of your studying. You are responsible for material covered in class sessions, lab exercises, and homeworks for the entire semester; but, here's a quick overview of especially important material since Exam 2.
 - Note that there is some redundancy below (some topics are "approached" from different directions in more than one section).
- 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.
 - (Note that final exams are **not** returned, although you can come by my office after they are graded to look at yours, if you would like. I'll keep them on file for at least two years.)
- The general style of the Final Exam will be similar to Exams 1 and 2, **except** that the Final Exam will be **comprehensive**, and will deliberately encompass material from the entire semester.
 - (Thus, it would be very wise to understand anything that you missed on Exams 1 and 2; you've got those tests to study from, and you have the review suggestion sheets for Exam 1 and Exam 2, still available from the course web page under "Homeworks and Handouts".)
- Remember that UNIX/Linux are **case-sensitive**; an answer may lose points if it uses the wrong case.
- Your studying should include careful study of posted examples and notes as well as the homeworks (and posted example solutions) thus far.

a few random notes

- Chances are pretty high that a `chmod` question will be included on the Final Exam.
- Chances are also pretty high that you will get another chance to redirect standard error on the Final Exam.
- More-extensive use of `sed` than was the case on Exam 2 is now fair game; you could be asked to use selectors (both single addresses and ranges) as well as the `d` command.
- The Week 11 material on additional history-related command-line shortcuts that was not included on Exam 2 is fair game for the Final Exam, also.

intro to openSUSE and KDE

- What is openSUSE? What is KDE?
- Can KDE work only with openSUSE?

intro to bash shell script functions

- You should be able to read and write bash shell script functions; you should be able to read and write calls to bash shell script functions.
- EXPECT IT - you will have to write at least one bash shell script function, and you will have to write a call to at least one bash shell script function.
- How can a bash function have parameters/arguments? How does one call a bash function that expects arguments? Can you change the parameters within a bash function's body?
- How can a bash function kind-of "return" something? How can it "affect" what is around it?
- In a bash function, what is the difference between including an `exit` statement and including a `return` statement? What does each do?
- How might backquotes be useful in combination with a bash function?
- How can one bash shell script use functions defined in another bash shell script?
- How does scope work with bash shell script and bash functions?

intro to sftp

- What is FTP?
- What is command-line `sftp`? What can it be used for? How do you use it?
- You should be able to read and write command-line `sftp` commands.
- You should be comfortable with the `sftp` commands discussed in lecture and practiced in lab/homework.

expr and bc

- You should be familiar with the basic purpose of each of these commands, and how each can be used. You should be able to read and write basic calls to these commands.
- How can these be used to obtain the results of arithmetic operations?
- `expr`'s arguments must be made up of "words" -- what can "words" be? What must these "words" be separated by?
- Note that you can pipe an expression to `bc` in addition to using it interactively.

sleep, time, wait, nohup, nice, renice

- You should be familiar with the basic purpose of each of these commands, and how each can be used. You should be able to read and write basic calls to these commands.
- Which allows you to start up a script and try to run it at a lower priority than it would otherwise be run at? Which allows you to try to reduce the priority of a currently-running process?
- Which allows a process to simply do nothing for a given number of seconds?

- Which allows you to start up a job and have it continue to run even if you log out of the shell this job was started in?
 - In our local version of bash, what option can you use with `ps` to see that such a process is indeed still running even after its terminal has been closed?
- Which allows a process to not continue until another process has completed?
- Which allows you to obtain how long (in real time, user time, and system time) a command took to run?

bash case statement

- You should be able to read and write bash `case` statements; you should be comfortable with both its syntax and its semantics.
- Chances are pretty good that you will have to write at least one bash `case` statement.

intro to RCS

- What is RCS? What is it used for?
- How does RCS store versions? What is the trade-off it makes? Does an RCS revision group file with 10 versions of something necessarily take as much memory as storing copies of all 10 versions?
- How can you tell that a file is an RCS revision group file?
- When you check in an initial version of a file to a not-yet-existent RCS revision group file, where is it created if there is no local directory named RCS? ...if there IS a local directory named RCS?
- You should be comfortable with the RCS commands discussed in lecture and practiced in lab/homework.

intro to crontab, at, batch

- What can you use each of these commands to do? What is the difference between the capabilities of each? When would you choose to use one instead of the other two for different tasks?
- You should be familiar with the basic purpose of each of these commands, and how each can be used. You should be able to read and write basic calls to these commands.
- What is a crontab? What is allowed in a crontab?
 - What is "legal" for a crontab "comment"?
- What is a crontab entry? What is the meaning of each of a crontab entry's fields?
 - You should be able to read and write crontab entries.
 - How are crontab entry fields separated? What values are permitted in each of a crontab entry's fields? What does it mean when `*`, comma, or a dash are used in a crontab entry's field?
 - Make sure you know how to use the `crontab` command to create a crontab, how to create a crontab entry, how to edit your crontab, how to list your crontab, and how to remove your

`crontab`.

- How can you change the default editor used to edit your `crontab`? How can you find out the "full" pathname for your desired editor?
- How can you schedule a job using `at`? How can you list your jobs currently scheduled using `at`?
- How can you schedule a job using `batch`? How can you list your jobs currently scheduled using `batch`?

uptime, top, uname, df, du

- You should be familiar with the basic purpose of each of these commands, and how each can be used. You should be able to read and write basic calls to these commands.
- You should be familiar with the basic information that each of these commands provides.
- I will not ask you the define load average -- but you should have a basic idea of what it means, what command(s) you can use to obtain it, and why it is useful to occasionally check it.
 - What kind of situation might prompt you to check the load average?
- Which of these includes how long the system has been running since its last reboot/restart?
- Which of these includes much of the information from one of the others, but also includes the CPU usage of running processes currently using the highest percentage of the CPU, updated periodically as you view its output?
- Which of these, when called with its `-a` option, includes the currently-running operating system name, release, and version?
- Which of these can allow you to see the file system block usage for a file or for a directory (and all of its files/subtree)?
- Which of these allows you to display statistics about the amount of free disk space on a specified file system?

IFS and read

- What does the environment variable `IFS` contain? What does `IFS` stand for? How is it used by UNIX/Linux?
- Note that `read` can be followed by multiple variables -- what happens in this case?
 - If there is "too little" input for the number of variables, what happens?
 - If there is "too much" input for the number of variables, what happens?
- How do `IFS` and `read` work together? You should be able to read and write code involving these; you should be able to modify `IFS` for a given purpose.
- Chances are pretty high that you will be asked to identify what will be read into given variables given a code fragment including these.

date, cal, printf

- You should be familiar with the basic purpose of each of these commands, and how each can be used. You should be able to read and write basic calls to these commands.
- Which of these can be used to obtain a calendar for a given month and year?
- Which of these can be used to obtain "pieces" of the current date and time in a specified format?
 - I could provide some of this command's format descriptors, and you might be asked to read or write statements using them.
- Which of these can be used to specify formatted output in general?
 - I could provide some of this command's format descriptors, and you might be asked to read or write statements using them.

A few comments related to UNIX/Linux system administration

- Why is it a good idea, if you are a system administrator of a UNIX/Linux system (or even a UNIX-based system such as Mac OS X), to set up separate administrator and personal accounts for yourself?

sudo, useradd, passwd

- You should be familiar with the basic purpose of each of these commands, and how each can be used. You should be able to read and write basic calls to these commands.
- Which of these can be used to run a single "privileged" command (assuming that you have the ability to do so, of course)? What file, in some systems, determines who has the ability to use this command?
- Which of these can be used (in conjunction with one of the others) to create a new user account on a UNIX/Linux system?
- Which of these can be used to change one's own account password, and (in conjunction with one of the others) to set or change another user's account password?