CS 279 - Final Exam Review Suggestions - Fall 2022

last modified: 2022-12-06

Final Exam BONUS Opportunity

- You can receive (a maximum) ***5 POINTS BONUS*** on the Final Exam if you do the following:
  - Make a **hand-written** Final Exam study sheet (a single sheet of paper, no larger than 8.5" by 11", on which you have hand-written as much as you would like on one or both sides)
  - Submit a photo or scan of it saved as a .pdf, .png, .jpg, .gif, or .tiff to Canvas **by 8:00 am on Wednesday, December 14** such that I can read at least some significant CS 279 post-Exam-2 material on it.
  - Please let me know if you have any questions about this, and I hope it helps you in reviewing course concepts more effectively before the Final Exam.
  - You are **encouraged** to have this **at hand** as you are taking the Final Exam.

Final Exam Set-up

- You will take the Final Exam in TA 011 at **8:00 am** on **Wednesday, December 14**.
  - You are expected to work **individually** on the exam -- it is not acceptable during the exam to discuss anything on the exam with anyone else.
  - You may have your Final Exam study sheet on hand during the exam. Otherwise, the exam is closed-note, closed-book, and closed-computer/closed-electronic-devices.
  - I expect there will be some multiple-choice questions, and the rest will be short- to medium-answer questions.
- Your studying should include careful study of posted examples and notes thus far.
- The Final Exam is CUMULATIVE!
  - if it was fair game for Exam 1 or Exam 2, it is fair game for the Final Exam;
  - Thus, using the posted review suggestions for Exam 1 and Exam 2 in your studying for the Final Exam would be a **GOOD** idea.
    - Note that these are still available on the public course website's "Homeworks and Handouts" page, http://nrs-projects.humboldt.edu/~st10/f22cs279/279handouts.php
- You are responsible for material covered in class sessions and homeworks.
  - This review handout is intended to be a quick overview of especially important material since Exam 2.
  - TIP: It is **perfectly fine** to retake/read over the short-answer questions in Canvas from course Homeworks as you are studying for the Final Exam!

These are set up for unlimited retakes, and only keep the highest score, so you will not hurt your grade by doing so!

- Remember that Linux/UNIX commands are **case sensitive** - for example, `Ls` is not the same command type as `ls`. You are expected to use the correct case in your answers.
- You are also expected to follow CS 279 course style guidelines in your answers.

the `find` command

- you should be able to read, write, and understand how to use the `find` command
- When might this command be useful? How does it differ from, say, `ls` or `grep`?
- How do you specify where the `find` command should start searching?
- criteria you should be comfortable with include `-name`, `-print`, `-type`
- Remember that you can use file globbing wildcards to represent the files you want to look for, but you need to escape or quote them so they won't be expanded too soon.
- If you start searching from, say, the root directory, what might be convenient to add to the end of a `find` command so you don't see error messages from trying to search through directories to do not have permissions to search?

`tar`, `gzip`, `gunzip`

- 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.
- When might you want to use these commands?
- What does `tar` stand for?
- How can you use `tar` to package a directory of files into a single tar file?
- How can you list the contents of a tar file?
- How can you extract all of the files form a tar file?
- How can you use `gzip` and `gunzip` to compress and uncompress a file? How can you find out the percentage by which a file was reduced?

the `sed` command

- you should be able to read, write, and understand how to use the `sed` command
- When might this command be useful? What does it allow one to do?
- Should be able to write patterns to make substitutions for the first instance of something in each line of a file; should be able to write patterns to make substitutions for all instances of something in each line of a file
- Should be able to write selectors that specify which lines in a file to edit

- Should be able to write a `sed` command to delete lines from a file

more options for searching through history

- how can you specify to redo the most-previous command?
- how can you specify to redo the most-previous command that starts with a specified string?
- how can you specify to redo the most-previous command that includes a specified string?
- how can you specify to redo the command with a particular history number?
- how can you specify to redo the command a specified number of commands ago?
- how can you redo a previous command but with the first instance of a specified pattern replaced? ...but with all instances of a specified pattern replaced?

file links and related topics

- 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 symbolic/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 `nano`, say?)
  - with each, what happens when the "original" file is deleted?
  - what are the restrictions on hard links?

intro to Bash functions

- You should be able to read and write Bash functions; you should be able to read and write calls to Bash functions.
- EXPECT IT - you will have to write at least Bash function, and you will have to write a call to at least one Bash 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 can the caller obtain the value from a called function's `return` statement?
- 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 scripts and Bash functions?

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 reasonably good that you will have to write at least one Bash `case` statement.

`uptime`, `top`, `who`, `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?
- Which of these can let you know who else is currently logged in?

`head`, `tail`, `date`, `cal`

- 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 let you see just some of the lines at the beginning of a file? How can you specify how many?

- Which of these can let you see just some of the lines at the end of a file? How can you specify how many?

`sleep`, `time`, `wait`, `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 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? Interestingly, to where is this command's output actually sent?

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`?

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?