Penn Engineering
ONLINE LEARNING

Katie Pizziketti - TA Fall 2020

Tips for Project1 and Command Line

What to try first when your code isn't working:

- Review the starter code; do you understand what the functions are doing? Try writing out a flow chart of your program to ensure you are clear on the order of execution.
- Check the man (Linux manual) pages for the system calls you are using such as read(), write(), fork(). Are you passing in the correct arguments?
- Are your strings all null terminated? Recall that 0 and '\0' are not equivalent in a string.
- Are you *sure* that your strings are all null terminated? All of them? Walk through your program line by line; every time a string is referenced, consider whether there is a chance that it is missing the null terminating character, '\0'. Consider the memset() function.
- Consider which lines of code should be executed by the parent and which should be executed by the child. Review the fork() man page for clues on how to differentiate each process.
- Valgrind is especially helpful for identifying invalid reads/writes which typically indicate you have an off-by-one error in how you are handling one of your arrays; you are writing or reading past the end of the array. This can catch sneaky errors that might not be apparent from just running the program. Of course, Valgrind is also useful for finding memory leaks!
- If you're not sure where your segfault is happening, run gdb to try to find the line number or library function call that is causing the fault.
- Print statements are excellent for narrowing down the section of code that needs to be updated. Just take them out before submitting. Try: print the line number!
- Use the autograder! Because you get unlimited submissions until the submission deadline, there is no harm in uploading your project before you are sure it is complete, and reviewing which test cases failed for hints on what changes are needed.
- Take a break. Completely. Walk away from your code for at least 5 minutes. Sometimes you might need an entire day away from it. That's okay; budget time for a break. Do something completely different that makes you feel good about yourself. It is normal to need time to recharge when working on something new and difficult.
- Get another set of eyes: post your repository link on Piazza. Even better if you can narrow down your specific problem into a public post; your classmates may be able to help. Avoid posting large chunks of code even in private posts (TAs can see everything in your repo) but do include examples of the error message you are getting and/or any output from your program. Please indicate in your private post if you're okay with having the post made public in case your question hits a common issue other students may benefit from seeing you work through.
- Message a classmate on Slack if you are feeling down about your work. Peer support is an invaluable tool for maintaining a positive attitude and open mind. While you can't share code, simply discussing how challenging the project is can help you feel better about continuing.

Useful command line instructions:

Here are some instructions previous students in the course* recommend exploring. However, the best way to learn command line is to notice when you have the thought "I wish I could just do X action in this terminal" and then search online for "how to do X action command line linux." Note that terminal in Mac is the same as command line or cmd in Windows.

Change directory: what we call "folders" in Windows are "directories" in Linux. To navigate your VM, use the 'cd' command; it takes a single argument, which is the name of the folder to switch to. Note that when starting your VM for the first time, you will always have to use 'cd /vagrant/' or 'cd ../../vagrant' but subsequent cd commands do not need to have the slash ('/') char. Examples:

- cd project1a (enter project1a directory)
- cd .. (to go back to the previous directory, just give 2 periods as argument)

View files in current directory: The 'ls' command, short for 'list,' will list the contents of the directory you are currently in. You can use 'ls' without any arguments. Add '-l' to get details about files, such as time modified; add '-a' to see hidden files. Examples:

- ls (simple list of files, usually good enough)
- ls -l -a (see all files and details about each file, including hidden files)

Get path for current directory: the 'pwd' command is short for "print working directory;" you can use it without arguments. Example:

- pwd

Create new directory/folder: to create a new directory within the current directory use the 'mkdir' command, short for 'make directory.' This takes an argument, which is the name you want to give to the new directory. You can check if it worked by using the ls command or checking in your host OS file explorer/finder. Example:

- mkdir project2b (makes a new directory called project2b)

Create a new file: you can create a new empty file without content using the 'touch' command; example:

- touch shredder.c (creates an empty file called shredder.c)

You can create a new file with content, or update the content of an existing file (such as one you created using touch) with the 'cat' command, short for 'concatenate.' To use 'cat' to edit or create a file, use the arguments '> filename'; note that this command will not show a new shell prompt but will wait for you to enter whatever text you want the file to contain. When you are done writing the text, press ctrl+d or cmd+d to write to, save and close the new or existing file. Example:

- (to paste starter code copied to clipboard) cat > fileName.c, hit enter, paste file, hit enter, ctrl+d or cmd-d

Key git commands: after cloning your repo in project0, you only need to use 3 commands each time you want to push to git, in this order: git add, git commit, git push. Examples:

- 'git add' takes as argument the file, files, or directory you want to add. You can list multiple files or just use '.' argument to add everything in the current directory: 'git add penn-sh.c' or 'git add .' (to add everything)
- 'git status' will show you which files in the current directory have changes that have not yet been committed.
- 'git commit -m "message"' where "message" will be shown in your git repository as a descriptor for that push. Note that there is a char limit on the message; keep it short and sweet!
- 'git push' typically requires an argument to identify which branch you are pushing to; most students will use 'git push origin master' to push to the master branch of the repository.
- Note: you can check if the push went through by logging into Git and viewing your repository to see the changes that you pushed.

*Author: Katie Pizziketti

*Acknowledgements: Thanks to Melissa Amaya and Denes Martin for additional contributions!