COP6611: Operating Systems **Name:** *Dr. Ankur Mali*
Assignment 2: Implement basic shell in Unix and add system calls in xv6
09/30/2022

Academic Honesty

All work submitted in this course must be your own. Cheating and plagiarism will not be tolerated. If you have any questions about a specific case, please ask me and your TA's. We will be checking for this!

Assignment Notes

General Notes

- Please read the assignment instructions carefully, including what files to include/submit.
- Don't assume any limitations unless they are explicitly stated.
- Developing optimized solutions is good, but I won't deduct any points if your code is not optimized
- Please ensure the functionalities requested in these assignments are included, you can add extra but cannot remove any.
- When in doubt regarding what needs to be done, ask. Another plausible option is test it in the real UNIX or ubuntu operating system. Unix is your friend, use it wisely. Does your code behave the same way? If not debug. Remember to use man "command_name" for reference.
- Test your solutions, make sure they work. Attach your screenshots, showing your steps, your console and other details.

Rubric

Here are some of the basic things our script will test, but these are not the only things we will test. Don't hard code any solution. Ensure that you test your program completely and provide detailed information concerning steps followed/adopted to ensure your code is bug free. This assignment will add 6.25 to your total grade point. To make it easy I have allocated 625 points which will be converted into 6.25 after your submission.

Total: 625 points = 6.25 Grade point

1 Implementing Shell command on your Unix system and comparing against xv6 – Total points 500

Important notes:

- For this assignment you are free to use any C compiler and test your code in your Unix machine
- Given we are working outside of xv6, you are free to use your favorite libraries such as string.h, stdio.h, stdlib.h etc.
- Remember there are several differences between xv6 and Unix kernel. If you simply copy paste content from xv6, many commands may not work and vice-versa is also true.

In this assignment, you will implement pieces of the UNIX shell and get some familiarity with a few UNIX library calls along with the UNIX process model. By the end of the assignment, you will have implemented a shell that can run any series of complex pipelines of commands. **Remember to follow the basic procedure and implement what we have asked for. Making your code complex, might lead to issues while debugging.** I will provide a text file that contains list of majority of english words. While designing your shell ensure that it has some capability to handle some complex processes such as:

$ cat OS6611_words.txt — grep ca — sed s/ca/bird/ > sample.txt

The following command shown above takes a document labelled OS6611_words.txt (I have include this word file with this assignment). Then selects the word containing "ca" and replaces that with "bird". In others words words like "cat", "can", "can't" will be replaced to "birdt", "birdn" , "birdn't". This output is then copied to a file sample.txt. **Important points to remember:** If sample.txt does not exist, you should create one. However if file exists then you should append these values. Other error handlings are also important.

Implement basic commands using your shell

- (5 points) Create a visualization for your shell, you have flexibility to use any pattern. This will make your shell unique compared to others. In others words whenever i run your shell in my console, I should see some pattern like a star, triangle with digits, letters etc.
- (20 points) Assuming you have completed your xv6 Assignment 1. In this case assignment you will be implementing the 3 basic commands (sort, uniq, head) in standard C and will test them in your system. **What are the differences between xv6 and your shell implementation. Provide a detailed report**
- (80 points) Next you will add other functionality, such as Cat, tail, grep, sed, ls, mkdir. For all these commands default option should work. Error handling is important, for instance while using mkdir if directory exists, abort the process and provide error to the user. **However, in this assignment we provide flexibility to include additional functionality. For instance one can extend ls with -a or -l option.**

Create a detailed report, showing how your implementation differs compared to Unix and xv6. Analysis should include average time to execute any command and other notable details.

- All your programs should have following functionalities. Failure to add these functionality will lead to **zero** points. In worst case scenario you should have minimum 1 functionality for all your processes.
  - (120 points:) **Command Execution** exec command in Linux is used to execute a command from the bash itself. This command does not create a new process it just replaces the bash with the command to be executed. If the exec command is successful, it does not return to the calling process. More information concerning working of this process look at "man exec". For instance output of:
    $ ls
    $ exec ls
    should be similar, but later one shouldn't create any new process. There are many variations for "exec", do man exec and choose one that is most suitable for your task.
  - (60 points:) **I/O Redirection** Part of this functionality is already completed in your assignment 1. You will extend your shell to handle input and output redirection. Programs will be expecting their input on standard input and write to standard output, so you will have to open the file and then replace standard input or output with that file. For instance look at following command:
    $ ls > demo.txt
    $ sort > sortdemo.txt
    $ tail > taildemo.txt
    The output of these standard command should be copied into some file. **Remember:** If file exists output should be appended or else create new file.
  - (215 points) **Pipes:** The final functionality is to add the ability to pipe the output of one command into the input of another. This will help in executing complex process example shared earlier. For this part it is important to look into fork, pipe, wait, close and other related process.

2 Implementing tail on xv6 – Total points 125

It is the complementary of Head command. The Tail command, as the name implies, prints the last N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name. If no filename is provided, tail should read from standard input.You should also be able to invoke it without a file, and have it read from standard input.

Important:(If the first argument does not start with -), the number of lines to be printed should default to 10)

Rubric for Part-2

- If tail does not work (-65)

- If any debug statements (printf) are getting printed along side output (-5)
- If program does not terminate successfully (-5)
- tail ”filename” does not work. Even if this partially works but doesn’t show default settings = 10 lines (-25)
- tail -n ”some_number” ”filename” does not work (-25)

Submission Instructions

File should be submitted in zip format, with 2 folders **Shell for Unix, Tail for xv6**.

- For unix shell: You should provide the C file with detailed report. Provide screenshot and explain in your own words logic you developed to solve complete/solve this assignment and approach that helped in successfully implementing each sub-processes.
- **For xv6**
  - 1.Screenshots after executing ”make clean”
  - 2. Screenshots after executing ”make”
  - 3. Screenshots after executing ”make qemu”
  - 4. Output of all conditions provided in your rubric
  - 5. All c scripts and modified Makefile