Hanish Patel

CS 497-01

Assignment 7

CS 497 – Assignment 7

**Abstract**

Examine system calls using C and bash. Pass arguments into C and fork parent programs to bash. Wait for fork to finish before returning to parent. Execute forked programs and deallocate all ram. Stop waiting for the program to execute. Retrace system calls by using the strace command. Understand the processes and the event of executing processes during system calls. Understand how bash parses arguments and runs commands from different languages.

**Introduction**

Use the built-in gedit to create files

gedit [file]

Compile gcc with the following command

gcc -o menu menu.c

Run the c file with

./menu

Use basic commands

Ls -la [directory] to list files

Touch [dir] to make txt files

Cat [dir] to open files

Ps to see processes

Top to see more information about processes

Nano [dir] to edit a file

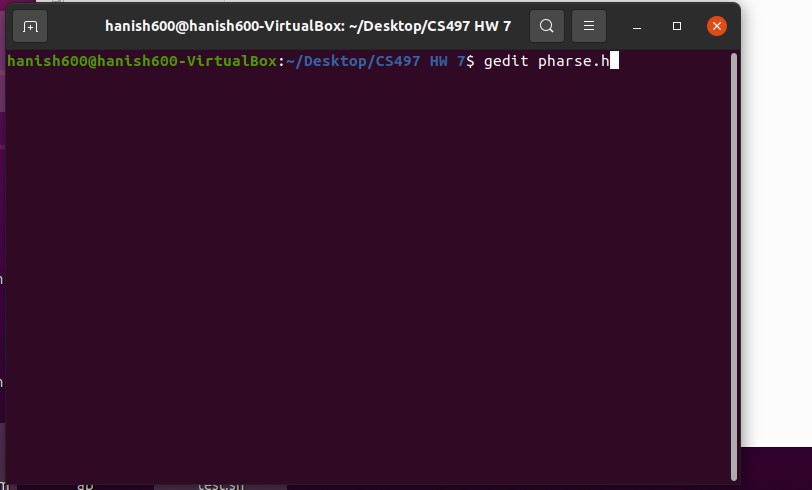
Sudo reboot to reboot the system

Exit to quit the program

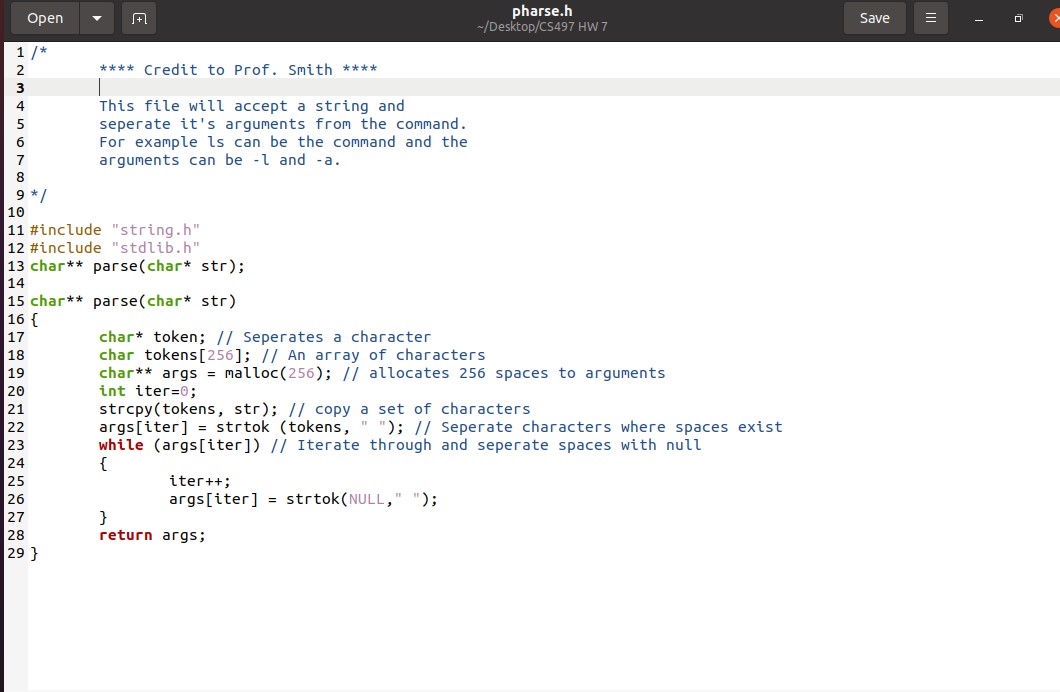
Strace to trace the program

**Summary of Results**

Start with editing files using gedit



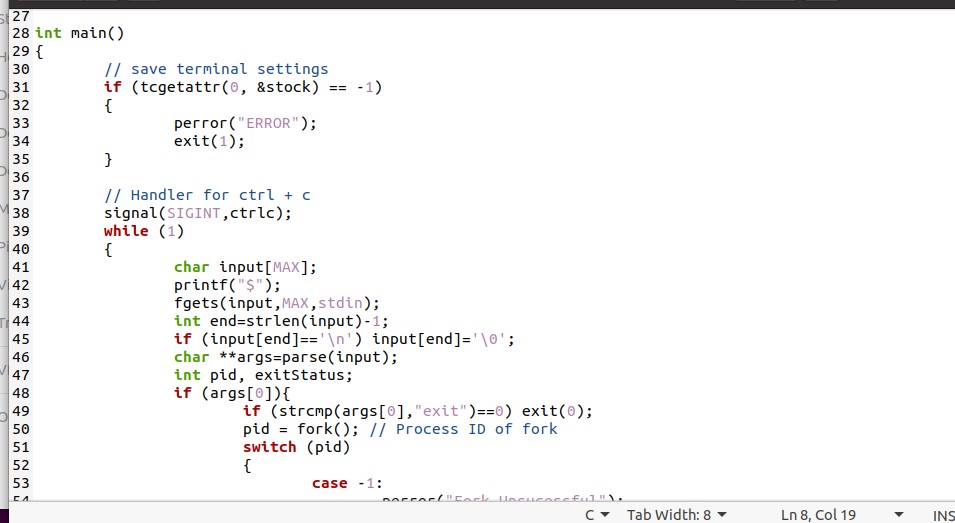
Make a parse file that reads a command, takes a token, takes an array of characters, allocates space to the arguments, copies the string into the array of characters, and identifies spaces and separates them by adding a null character at the end. Returns different strings.



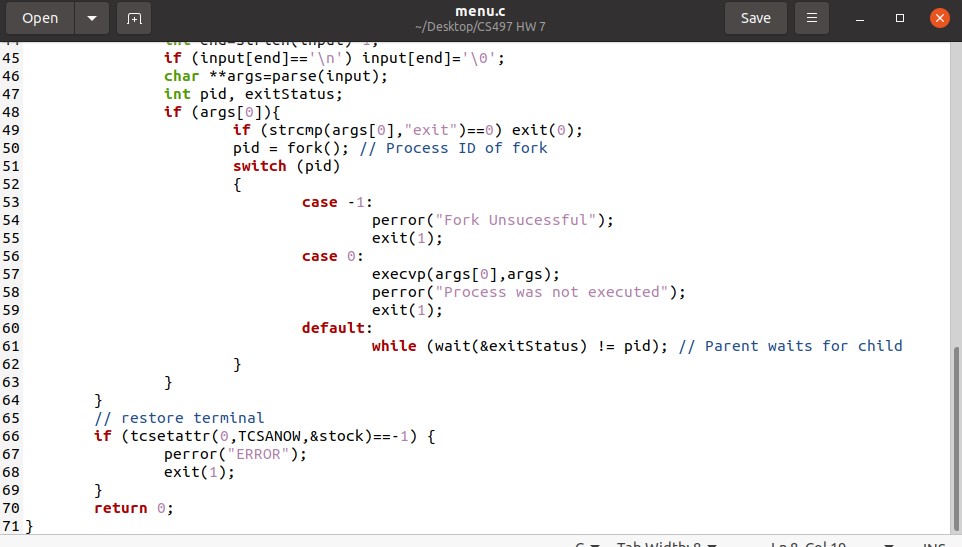
Start with the menu by importing libraries and adding a ctrl + c interrupt.



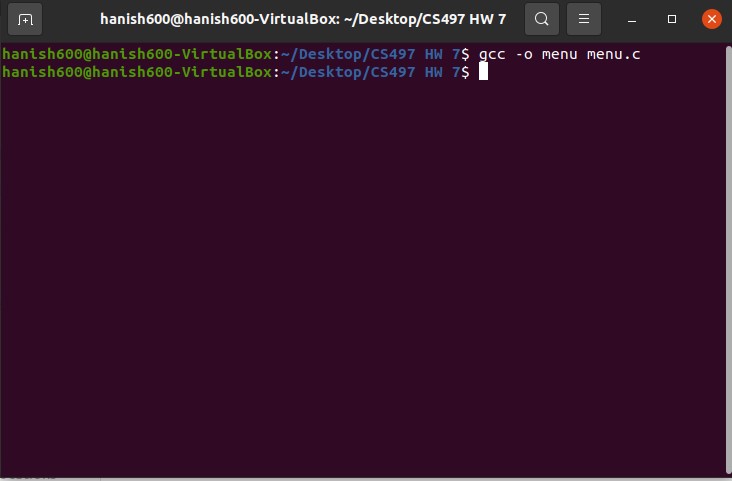
Set a terminal attribute, and parse through commands. Take a pid and exitstatus and fork the child into the new pid.



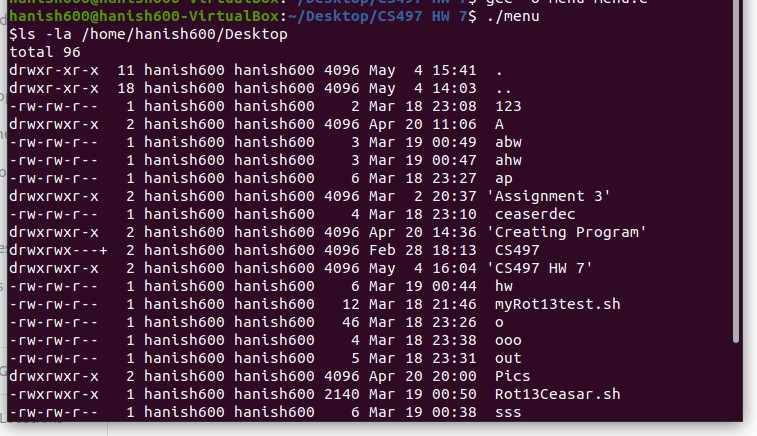
Create conditions for successful or failure of the execution of the forked processes. Restore the attribute.



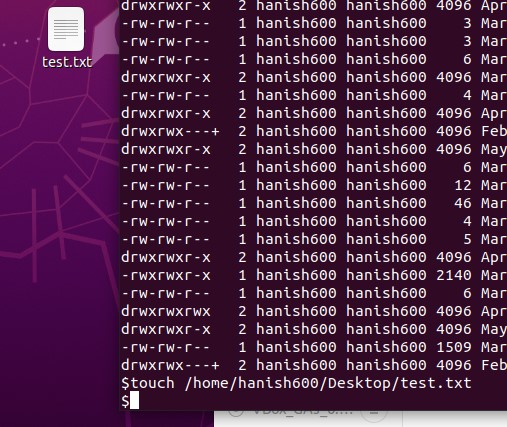
Use gcc to create an executable.



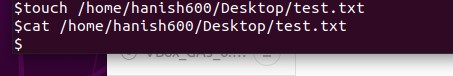
Start with the ls -la command.



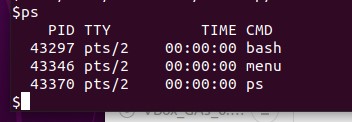
Create a txt file with touch.



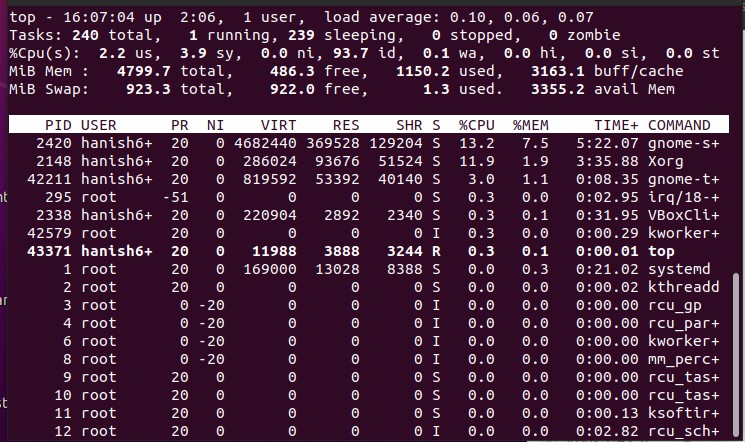
Open the newly created file with cat. Notice, there is no content in this file.



See running processes with ps.

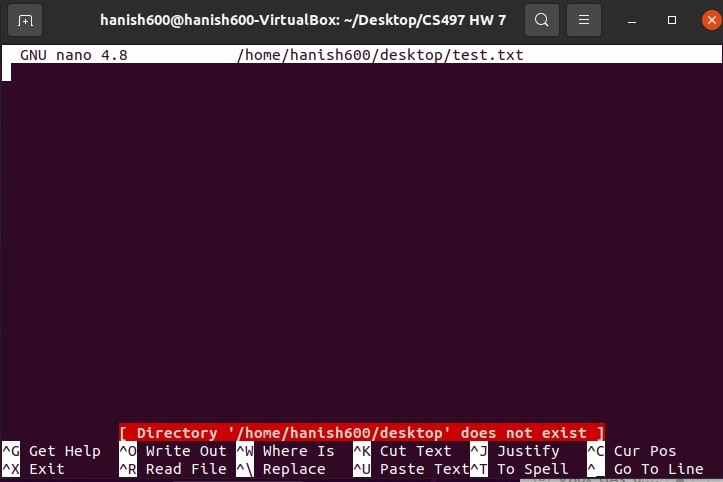


See more information using top command.



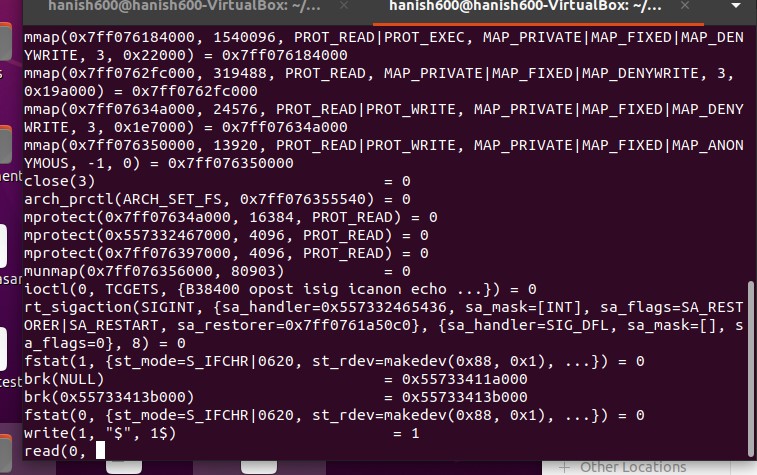
Edit a text file.

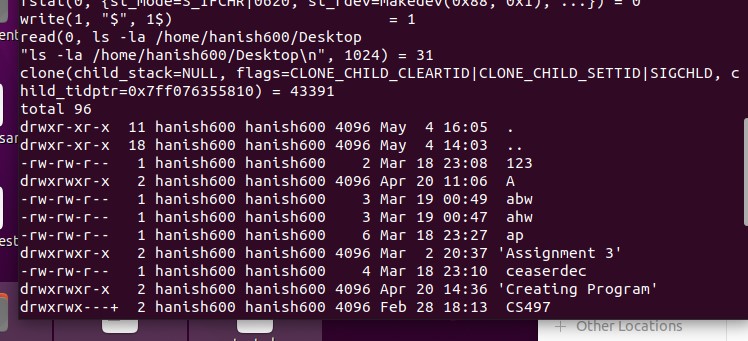
Notice, desktop is case sensitive.



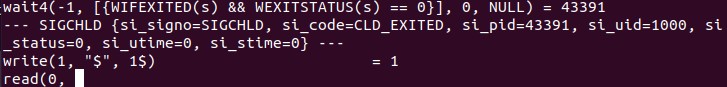
Check the commands with strace.

Start with ls -la.



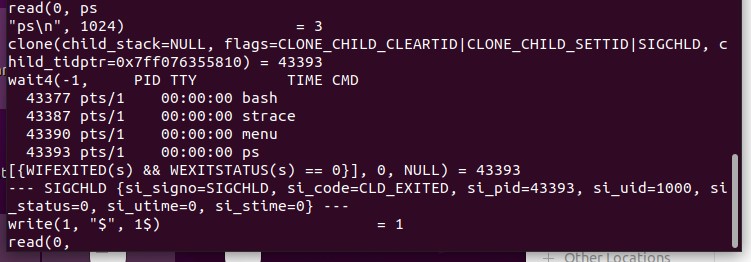


As we can see it has cloned a child.



We notice the parent is waiting for the child to return exit 0.

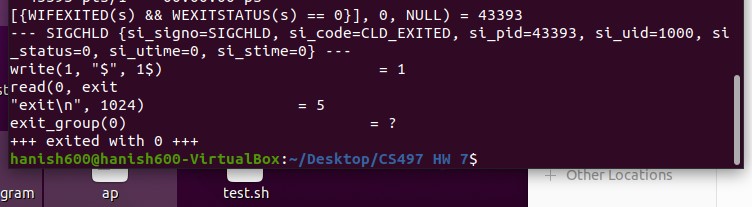
We use the ps command next.



We again notice the fork clones a child and the parent waits.

We see ps is process id 43393 and this is returned to some variable.

Finally, we run a simply exit command.



We see it has exited without any waiting being required.

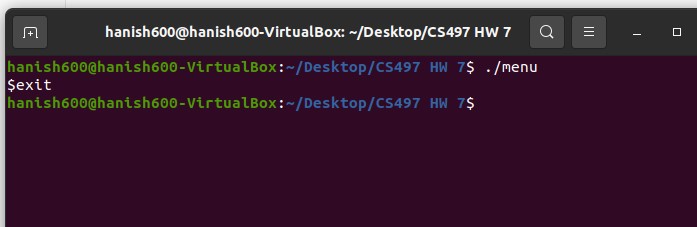
We now use sudo reboot without the strace. We also test our interrupt.



Now going back to sudo reboot we get the VM to reboot.



Finally, we test that exit does not return errors.



**Conclusion**

We saw how processes can be forked, executed, destroyed, and how we can track each process with strace. We learned to use C to make system calls such as Linux’s very own ls or ps or top. We also saw we can add arguments. This was done using a C program and by creating clone copies. We saw how a program like bash can take a string, parse it, and execute commands using that program. Like for example, if we take python, or C, or Java, or some other language and we create a text file, we saw how bash could clone that command and execute it while the main program waits.

I was impressed that the bash can simply clone a command with it’s arguments and the code for it in a program such as C does not have to be very complicated to understand. We also saw that each process has its own pid and that pid value can be part of the cloned copy. Cloned copies can be useful when we want to run system calls, but we don’t want to write them directly in bash. For example, maybe we want to use the cat program, but we don’t want to create a shell script, we want to instead create a C script. We are able to still use the cat program thorough forking and execvp. We do this a lot when we write in a programming language.