Gebze Technical University
Department of Computer Engineering - CSE 344 System Programming
Spring 2018-19, HW #3 (Due April 5th @ 23:55)
NO LATE SUBMISSIONS

The objective is to develop your own shell named **gtushell** (The name of the executable must be this! We will run it with the command ./gtushell). Your shell must support the following commands which you will code:
- **lsf**; which will list file type (R for regular file, S for non-regular(special) file), access rights (int the form of rwxr-xr-x, just like actual ls), file size(bytes) and file name of all files (not directories) in the present working directory. It doesn't take any argument. Does not list any directory.
- **pwd**; which will print the path of present working directory.
- **cd**; which will change the present working directory to the location provided as argument.
- **help**; which will print the list of supported commands.
- **cat**; which will print on standard output the contents of the file provided to it as argument or from standard input. The file can be on any directory on the system. (example, cat file.txt)(example2, pwd | cat)
- **wc**; which will print on standard output the number of lines in the file provided to it as argument or the string coming from standard input until EOF character (example, Input: wc file.txt Output:55) (example2, Input: lsf | wc Output: 5 (there are 5 files in current directory so output of lsf has 5 lines)).
- **bunedu;** your HW#1. This time, the file path argument of bunedu can also come from standard input.
- **exit**; which will exit the shell

pwd, lsf, cat, wc and **du** commands will be separate executables (each has its own main function), we will call them ***utilities***. They will be called with **fork+exec** method from your shell. Your makefile will compile all of them including the shell to create an executable of each.

Your shell must have the following features:
- When the user types !n , the n-th previous command will run. For example, lets assume the commands ran so far are [pwd, cd arg, lsf, pwd], when the user types !2, lsf will run again (cd arg for !3 etc.). The command must run with the same arguments.
- The vertical bar character '|' must enable constructing **a pipe** from **[wc, bunedu, lsf, cat** or **pwd]** to **[wc, cat** or **bunedu]**. This will redirect output of the first command to input of second command (example, cat file.txt | bunedu -z, this will call bunedu -z on the path written inside file.txt). Do not make a seperate if statement for each case, implement the redirection by using pipe system calls in a general form! You can test if a command is utility or not.
- must **support redirecting** standard input and output of commands all ***utilities*** to files through the '<' and '>' characters.
- Your each command will have a single pipe | or a single redirection operator >< or no operators. So you don't need to think about cases like a | b > c (**maybe** we will add that feature in future homeworks). Do not think complicated cases like this.
- in case of SIGTERM, your shell must exit properly (e.g. by printing a message on screen).

CONTINUE TO NEXT PAGE...

Rules

a) You must implement all the commands supported by your shell; **you cannot use the "system()" system call.**

b) Your program must handle eventual errors gracefully according to the POSIX traditions.

c) Ask your questions in the Moodle forum "HW3 Questions" by opening a new topic.

d) There shouldn't be any zombie or orphan processes when the execution is finished. Also there shouldn't be any memory leak. Test all of these with valgrind.

e) Homework format is same as previous.

f) Keep in mind that your homework will be tested by our script. Do your homeworks in the way that is wanted, that will be enough.

Good luck.