Anaiah Quinn
CS 4375
Dr. Tosh
February 12 2025

Assignment-2A: Building your own shell

Task1:

I started by reviewing the given code. I added a bunch of comments to explain and thoroughly understand the assignments.

Next I implemented the forking of a child process for each command.

```
 return tokens;
}


int main(int argc, char* argv[]) {
        char  line[MAX_INPUT_SIZE]; // BUffer for storing user input
        char  **tokens;             // array of tokens (command + arguments)
        int i;


        while(1) {// Infinite loop to keep the shell running
                /* BEGIN: TAKING INPUT */
                bzero(line, sizeof(line));// clear input buffer
                printf("minersh$ ");// Display shell prompt
                scanf("%[^\n]", line);// read input until a newline is encountered
                getchar(); // consume the new line character left in buffer

                printf("Command entered: %s (remove this debug output later)\n", line);
                /* END: TAKING INPUT */

                line[strlen(line)] = '\n'; //terminate with new line
                tokens = tokenize(line); // tokenize the input string

      //do whatever you want with the commands, here we just print them

                for(i=0;tokens[i]!=NULL;i++){
                        printf("found token %s (remove this debug output later)\n", tokens[i]);
                        // creating new process
                        pid_t pid =fork();

                        if(pid<0){
                                printf("Fork failed");
                                perror("Fork failed");
                        }else if (pid==0){
                                printf("Child Process created successfully!\n");
                                exit(0);// Exit child process

                        }else {
                                waitpid(pid, NULL, 0); // Parent waits for child
                        }

                }

                // Freeing the allocated memory
                for(i=0;tokens[i]!=NULL;i++){
                        free(tokens[i]);
                }
                free(tokens);

        }
        return 0;
}
```

Next I executed the commands with the newly forked child process.

```c
//do whatever you want with the commands, here we just print them

        for(i=0;tokens[i]!=NULL;i++){
                printf("found token %s (remove this debug output later)\n", tokens[i]);
                // creating new process
                pid_t pid =fork();

                if(pid<0){
                        printf("Fork failed");
                        perror("Fork failed");
                }else if (pid==0){
                        //printf("Child Process created successfully!\n");
                        if(execvp(tokens[0],tokens) == -1){//execute the command
                                perror("Command failed");
                        }
                        exit(0);// Exit child process

                }else {
                        waitpid(pid, NULL, 0); // Parent waits for child
                }
```

Then I handled empty input and the exit command.

```c
bzero(line, sizeof(line));// clear input buffer
printf("minersh$ ");// Display shell prompt
scanf("%[^\n]", line);// read input until a newline is encountered
getchar(); // consume the new line character left in buffer

printf("Command entered: %s (remove this debug output later)\n", line);
/* END: TAKING INPUT */

line[strlen(line)] = '\n'; //terminate with new line
tokens = tokenize(line); // tokenize the input string

if(tokens[0] == NULL){ //ignoring empty input
        continue;
}
if(strcmp(tokens[0],"exit") == 0){ //handle exit command
        printf("Exiting shell..\n");
        free(tokens);
        break;
}
```

I then did my first round of testing.

```
[02/15/25]seed@VM:~/.../2A-shell$ ./minershell
minersh$ ls
minershell  minershell.c  notes.txt
minersh$ echo "Hello, world"
"Hello, world"
"Hello, world"
"Hello, world"
minersh$ invlaid command
Command failed: No such file or directory
Command failed: No such file or directory
minersh$ ps
    PID TTY          TIME CMD
   2366 pts/0    00:00:00 bash
   2724 pts/0    00:00:00 minershell
   2748 pts/0    00:00:00 ps
minersh$ exit
Exiting shell..
[02/15/25]seed@VM:~/.../2A-shell$
```

```
[02/15/25]seed@VM:~/.../2A-shell$ ls
minershell  minershell.c  notes.txt
[02/15/25]seed@VM:~/.../2A-shell$ echo "hello
, world"
hello, world
[02/15/25]seed@VM:~/.../2A-shell$ invalid com
mand
invalid: command not found
[02/15/25]seed@VM:~/.../2A-shell$ ps
    PID TTY          TIME CMD
   2693 pts/1    00:00:00 bash
   2746 pts/1    00:00:00 ps
[02/15/25]seed@VM:~/.../2A-shell$ wc
^C
[02/15/25]seed@VM:~/.../2A-shell$
```

Simple commands such as ls and ps worked no problem. The problem is that I seem to be making a child process for every token, not every command, resulting in repeated outputs. Problem code:

```
        //do whatever you want with the commands, here we just print them

                for(i=0;tokens[i]!=NULL;i++){
                        //printf("found token %s (remove this debug output later)\n", token
;[i]);
                        // creating new process
                        pid_t pid =fork();

                        if(pid<0){
                                printf("Fork failed");
                                perror("Fork failed");
                        }else if (pid==0){
                                //printf("Child Process created successfully!\n");
                                if(execvp(tokens[0],tokens) == -1){//execute the command
                                        perror("Command failed");
                                }
                                exit(0);// Exit child process

                        }else {
                                waitpid(pid, NULL, 0); // Parent waits for child
                        }

                }

                                                                        79,3-17       89%
```

I moved the fork outside of the for loop and this fixed the problem. I've included more test cases below

Ls,ls -a, ls -l, ps , echo, pwd	Cat, sleep, wc,

```
[02/15/25]seed@VM:~/.../2A-shell$ ./minershell
minersh$ ls
minershell  minershell.c  notes.txt
minersh$ echo "hello, world"
"hello, world"
minersh$ invalid command
Command failed: No such file or directory
minersh$ ps
    PID TTY          TIME CMD
   2366 pts/0    00:00:00 bash
   2860 pts/0    00:00:00 vim
   2914 pts/0    00:00:00 minershell
   2918 pts/0    00:00:00 ps
minersh$ ls -a
.  ..  minershell  minershell.c  notes.txt
minersh$ ls -l
total 28
-rwxrwxr-x 1 seed seed 17360 Feb 15 16:13 minershell
-rw-rw-r-- 1 seed seed  3057 Feb 15 16:13 minershell.c
-rw-rw-r-- 1 seed seed    67 Feb 13 13:11 notes.txt
minersh$ pwd
/home/seed/OS2025/2A-shell
```

```
minersh$ ls
minershell  minershell.c  notes.txt  testfile.txt
minersh$ cat testfile.txt
This is a test file.
minersh$ wc testfile.txt
 1  5 21 testfile.txt
minersh$ sleep 5
minersh$ ps aux | grep minersh
error: garbage option

Usage:
 ps [options]

 Try 'ps --help <simple|list|output|threads|misc|all>'
  or 'ps --help <s|l|o|t|m|a>'
 for additional help text.

For more details see ps(1).
minersh$ exit
Exiting shell..
[02/15/25]seed@VM:~/.../2A-shell$
```

Task 2

I started by detecting all “cd” commands entered. Instead of forking a child process, I used “chdir” to change directories. I also checked that the cd command included a directory

```
if(strcmp(tokens[0],"exit") == 0){ //handle exit command
        printf("Exiting shell..\n");
        free(tokens);
        break;
}
//detect cd command
if(strcmp(tokens[0], "cd") ==0){
        if(tokens[1] == NULL){ // if no directory provided
                printf("Shell: Incorrect command");
        }
        else{
                if(chdir(tokens[1]) !=0){ // change directory
                        perror("Shell"); // if change directory fails
                }
        }
        free(tokens);
        continue;// skip to next prompt
}
```

I’ve included a screenshot of the change directory working.

```
minersh$ mkdir testdirec
minersh$ ls
minershell  minershell.c  notes.txt  testdirec  testfile.txt
minersh$ cd testdir
Shell: No such file or directory
minersh$ ls
minershell  minershell.c  notes.txt  testdirec  testfile.txt
minersh$ cd testdirec
minersh$ ls
minersh$ pwd
/home/seed/OS2025/2A-shell/testdirec
```

The test cases pictured below also show “cd ..” as functional and the appropriate handling of invalid cd commands.

```
[02/15/25]seed@VM:~/.../2A-shell$ ./minershell
minersh$ cd
Shell: Incorrect command
minersh$ cd testdirec
minersh$ pwd
/home/seed/OS2025/2A-shell/testdirec
minersh$ cd ..
minersh$ pwd
/home/seed/OS2025/2A-shell
minersh$ cd test
Shell: No such file or directory
minersh$ pwd
/home/seed/OS2025/2A-shell
minersh$ exit
Exiting shell..
[02/15/25]seed@VM:~/.../2A-shell$
```

References:

I referenced chapter from the book, a geek for geeks article

- *Linux Man Pages (https://linux.die.net/man/1/intro)*
- *Geeks for Geeks* “Making your own Linux Shell in C”: https://www.geeksforgeeks.org/making-linux-shell-c/
- *minershell.c* (code template)