Programming Assignment #1 – A Simple Shell

Introduction to Operating Systems

Prof. Li-Pin Chang

CS@NYCU

A Simple Shell

- Control flow of your simple shell:
  - Display the prompt sign “>” and take a string from user
  - Parse the string into a program name and arguments
  - Fork a child process
  - Have the child process execute the program
  - Wait until the child terminates
  - Go to the first step

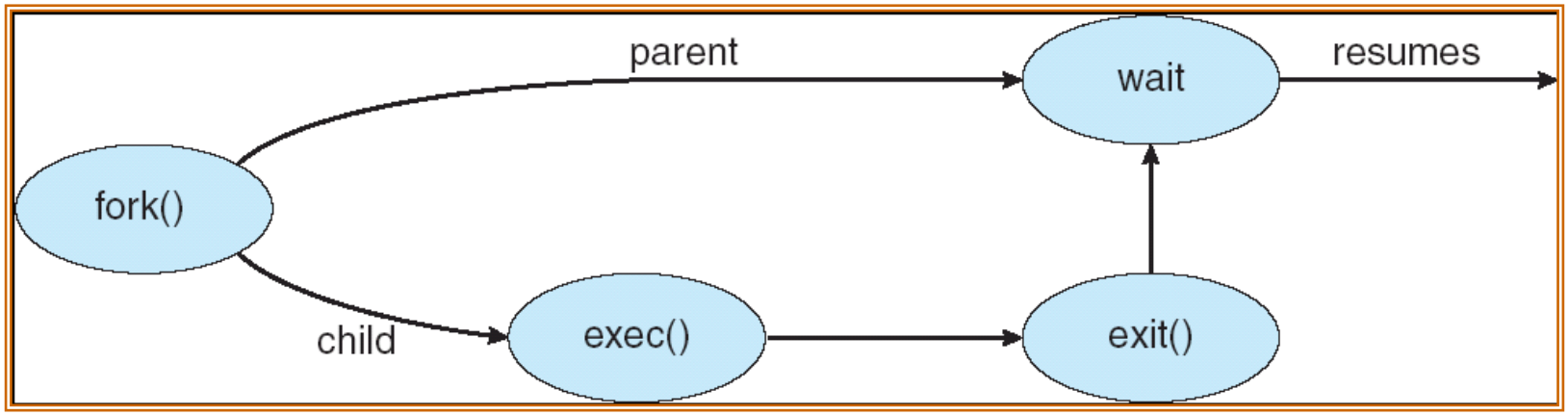
Example Output

```
justin@justin-virtual-machine:~/Desktop/SimpleShell$ ls
a.out  shell.c  shell.o  simpleshell.c  simpleshell.o  trace
justin@justin-virtual-machine:~/Desktop/SimpleShell$ ./a.out
>ls
a.out  shell.c  shell.o  simpleshell.c  simpleshell.o  trace
>/bin/ls
a.out  shell.c  shell.o  simpleshell.c  simpleshell.o  trace
>which ls
/bin/ls
>rm shell.o
>ls
a.out  shell.c  simpleshell.c  simpleshell.o  trace
>
```

Core Logic of Your Shell

```
pid_t  pid;


 …
/* fork another process */
pid = fork();
if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        exit(-1);
}
else if (pid == 0) { /* child process */
        execlp("/bin/ls", "ls", NULL);
}
else { /* parent process */
        /* parent will wait for the child to complete */
        wait (NULL);
}
…
```


parent
wait
resumes
fork()
child
exec()
exit()

Important System Calls

- fork()
  - Create a child process
  - http://man7.org/linux/man-pages/man2/fork.2.html
- exec() family
  - Have the current process execute the program specified in the pathname
  - http://man7.org/linux/man-pages/man3/exec.3.html
- wait() family
  - wait() wait the termination of anyone of the child processes
  - waitpid() waits the termination of the specified child process
  - http://man7.org/linux/man-pages/man2/waitpid.2.html

Waiting on child processes

- If a command is ended with “&”, then the shell will not wait on a child process
- For example:
  - sleep 10s
    - The prompt re-appears after 10 seconds
  - sleep 10s &
    - The prompt re-appears immediately
- A child process becomes a zombie if it is not waited by its parent process
  - How to deal with this problem?

Header of your .c or .cpp

/*

Student No.: 31415926

Student Name: John Doe

Email: xxx@yyy.zzz

SE tag: xnxcxtxuxoxsx

Statement: I am fully aware that this program is not supposed to be posted to a public server, such as a public GitHub repository or a public web page.

*/

Test Cases

- Your shell must correctly handle test cases of the following format
  - **`<program> <arg1> <arg2> <…>`**
- Test cases are like the following, but not limited to:
  - **`clear`**
  - **`ls -l`**
  - **`cp a.txt b.txt`**
  - **`cat c.txt &`**
- Do not leave zombie processes in the system!
  - Before or after your shell terminate
- The header must present in your source program

This. Is. Not. Funny.

- You got zero point if you use system()
- You don’t have to implement “ls”, “cp”, etc in our shell; they are independent programs
  - Just create a child process and have the child execute these program

Bonus!

- I/O redirection (+10 pts)
  - **`ls –l > a.txt`**
- Pipe (+10 pts)
  - **`ls | more`**
- No multiple or combined pipe/redirections
  - a | b | c
  - a < b > c
  - a | b > c
  - …

Testing OS Environment

- Ubuntu 18.04
- Install as a VM or on a physical machine