My Simple Shell

과목	운영체제
제출 일자	2020년 09월 20일
담당 교수	박호현 교수님
학 과	컴퓨터공학과
학 번	20172129
이 름	박예솔

1. 코드 설명

```c
#define MAXBUF 256

int main(int argc, char** argv) {

    pid_t pid;
    char *message;
    char buf[MAXBUF]; //get commands

    while(1) {
        // prompt should be “yourname$”.
        fprintf(stdout, "Yesol$ ");
        memset(buf, 0, sizeof(buf));
        fgets(buf, sizeof(buf)-1, stdin);
        buf[strlen(buf) - 1] = '\0';

        // If user inputs “exit” command, exit your shell.
        if(!strncmp(buf, "exit", strlen(buf))) {
            return -1;
        }
        // Each user command is executed by creating a child process and executing the command.
        message = ">> Command is executed by creating a child process...";
        puts(message);
        pid = fork();

        if(pid < 0) {
            perror("fork error\n");
            return -1;
        }
        else if (pid == 0) {
            message = ">> Command is executing by execlp()...\n";
            puts(message);
            execlp(buf, buf, 0);
            message = ">> Done...";
            puts(message);
            exit(0);
        }
        else {
            //The shell process must wait until the command processing is completed in the child process.
            message = ">> The shell process wait until the command processing is completed in the child process...";
            puts(message);
            wait(NULL);
        }
    }
    return 0;
}
```

우선 Myshell에서 입력 받을 명령어를 저장할 buf 문자열 배열과 명령창에 진행 상황을 알려줄 message 포인터 변수를 선언한다.

이제 While문을 이용해서 Myshell이 ‘exit’ 명령어를 받지 않는 한 계속 반복되어 실행될 수 있도록 해준다. 반복문의 시작에 fprintf를 [Yesol$ ~] 형태로 프롬프트가 실행되어 문자열을 받아오도록 한다. 이때 공백으로 문자열이 끊기는 것을 방지하기 위해 fgets()로 Command를 받아온다. 여기서 fgets()는 문자열 마지막에 개행 문자도 포함되므로 제거해주는 코드를 추가해주어야 한다. 마지막으로 받아온 문자열이 ‘esit’인지 확인하는 조건문을 넣어주고 exit이면 -1을 return해서 반복문을 빠져나가 종료한다.

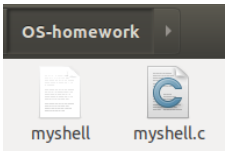
여기까지 마치고 나면 Myshell에서는 System call 명령어인 fork()를 통해 child 프로세스를 생성한다. fork의 결과로 나오는 pid 값을 통해 현재 진행 중인 부모 프로세스, 새로 만들어진 자식 프로세스 그리고 fork 에러로 분류한다. pid 값이 음수이면 fork 에러이고, 0이면 새로 생성된 child 프로세스 그 외의 경우에는 부모 프로세스의 번호이다. 따라서 fork 에러일 때는 return -1을 통해 프로세스를 종료한다. child 프로세스일 때는 execlp() 명령어를 통해 앞서 버퍼(buf[])로 입력 받은 문자열을 매개변수로 넣어준다. 받아온 문자열로 새로운 프로세스가 실행되고 해당 문자열에 해당하는 명령 프로세스를 실행시킨다. 이 과정이 정상적으로 진행되면 자체적으로 프로세스가 종료되면서 Done... 메시지가 출력되지 않고 조건문을 종

료하게 된다. 하지만 이때 입력 받은 명령어가 제대로 실행되지 않으면 execlp() 명령어가 종료되고 다시 조건문으로 돌아와서 Done이 출력된다. 마지막으로 부모 프로세스는 else 조건문의 wait()를 통해서 자식 프로세스가 종료될 때까지 잠시 정지하게 된다. 이 과정은 While문을 종료 시킬 때(exit 명령어, 키보드 인터럽트 등)까지 반복된다.

2. 실행 화면

① 작성한 소스 코드 컴파일/실행 & 컴파일로 생성된 myshell 문서

```
yesol@yesol-virtual-machine:~/문서/OS-homework$ gcc myshell.c -o myshell
yesol@yesol-virtual-machine:~/문서/OS-homework$ ./myshell
Yesol$
```



② ps 명령어를 통해 현재 실행중인 프로세스 확인

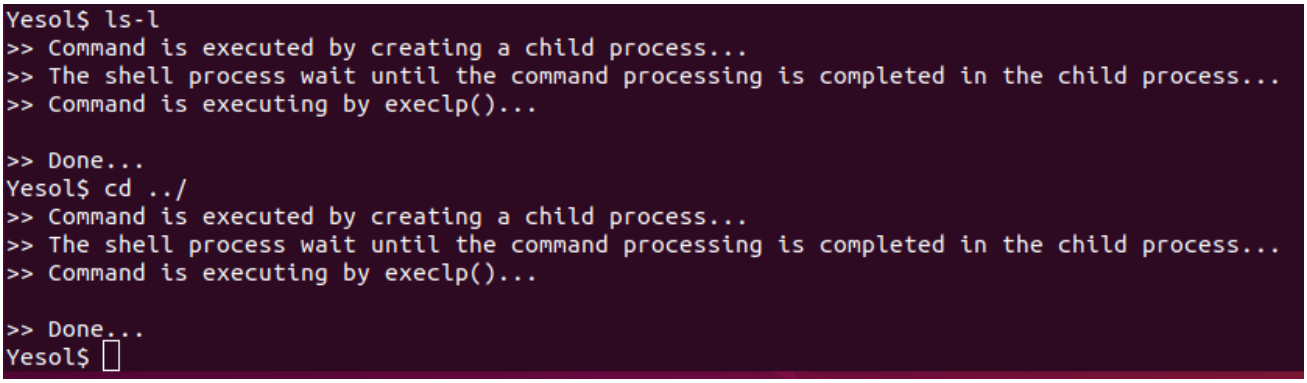
```
Yesol$ ps
>> Command is executed by creating a child process...
>> The shell process wait until the command processing is completed in the child process...
>> Command is executing by execlp()...

   PID TTY          TIME CMD
 26169 pts/0    00:00:00 bash
 26183 pts/0    00:00:00 myshell
 26187 pts/0    00:00:00 ps
```

③ 그 밖에 ls, pwd, ping 등 명령어를 수행 했을 때 출력 화면

```
Yesol$ ls
>> Command is executed by creating a child process...
>> The shell process wait until the command processing is completed in the child process...
>> Command is executing by execlp()...

fork1  fork1.c  myshell  myshell.c
Yesol$ pwd
>> Command is executed by creating a child process...
>> The shell process wait until the command processing is completed in the child process...
>> Command is executing by execlp()...

/home/yesol/문서/OS-homework
Yesol$ ping
>> Command is executed by creating a child process...
>> The shell process wait until the command processing is completed in the child process...
>> Command is executing by execlp()...

Usage: ping [-aAbBdDfhLnOqrRUvV64] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
Usage: ping -6 [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
             [-l preload] [-m mark] [-M pmtudisc_option]
             [-N nodeinfo_option] [-p pattern] [-Q tclass] [-s packetsize]
             [-S sndbuf] [-t ttl] [-T timestamp_option] [-w deadline]
             [-W timeout] destination
Yesol$
```

위의 결과를 보면 Myshell 프로세스를 실행 중에 명령어를 입력 받으면 제일 먼저 자식 프로세스를 생성하고 부모 프로세스를 wait 시키고 난 후, 생성된 자식 프로세스에서 execlp()를 통해 입력 받은 명령어에 해당하는 프로세스로 교체하는 것을 알 수 있다.

④ ls -l, cd ../ 명령어



ls -l, cd ../ 과 같은 명령어는 실행되지 않는 것을 확인하였다. 자식 프로세스는 생성되고 명령어대로 프로세스가 교체는 되나 실제로 입력 받은 명령어가 실행되지 않고 조건문으로 돌아와 "Done"이 출력되는 것을 확인할 수 있다.

⑤ exit 명령어를 입력하여 Myshell 프로세스를 종료시킨다.