# 2019 시스템 프로그래밍

## - week06 -

| | |
|---|---|
| **제출일자** | 2019.11.10 |
| **분 반** | 01 |
| **이 름** | 임준규 |
| **학 번** | 201602057 |

**Trace 번호 (00)**

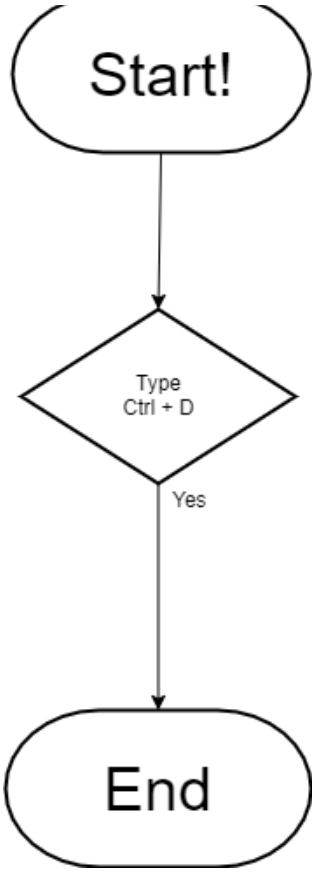
```
b201602057@2019sp:~/shlab-handout$ ./sdriver -v -t 00 -s ./tsh
Running trace00.txt...
Success: The test and reference outputs for trace00.txt matched!
Test output:
#
# trace00.txt - Properly terminate on EOF.
#

Reference output:
#
# trace00.txt - Properly terminate on EOF.
#

b201602057@2019sp:~/shlab-handout$
```

Ref와 비교

```
b201602057@2019sp:~$ cd shlab-handout
b201602057@2019sp:~/shlab-handout$ ./tsh
eslab_tsh> b201602057@2019sp:~/shlab-handout$ ls
```

Ctrl+d로 확인

각 trace 별 플로우 차트



## Trace 해결 방법 설명

```
if (feof(stdin)) { /* End of file (ctrl-d)     */
        fflush(stdout);
        fflush(stderr);
        exit(0);
}
```

이미 주어진 코드였으나, 함수의 작동을 몰라 찾아보았다.

feof(stdin)는 파일의 끝인지 확인하는 함수라고 적혀있었다.

Ctrl+d가 파일의 끝을 의미하는 입력이므로 이 조건에 해당한다.

그 아래의 fflush는 버퍼에 있는 값들을 모두 비워주고, 에러도 확인한다.

이후에 exit으로 탈출해준다.

## **Trace 번호 (01)**

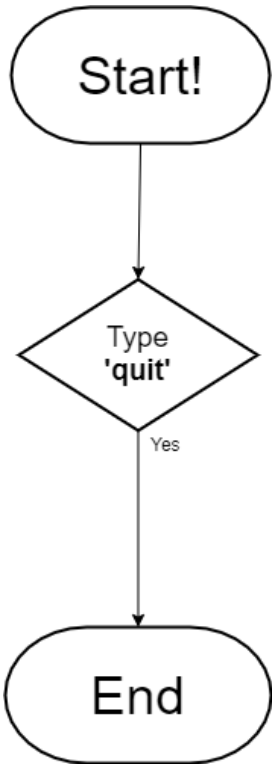
```
b201602057@2019sp:~/shlab-handout$ ./sdriver -V -t 01 -s ./tsh
Running trace01.txt...
Success: The test and reference outputs for trace01.txt matched!
Test output:
#
# trace01.txt - Process builtin quit command.
#

Reference output:
#
# trace01.txt - Process builtin quit command.
#

b201602057@2019sp:~/shlab-handout$
```

Ref와 비교

```
b201602057@2019sp:~/shlab-handout$ ./tsh
eslab_tsh> quit
b201602057@2019sp:~/shlab-handout$
```

Quit로 나갈 수 있음을 시연.

각 trace 별 플로우 차트




Trace 해결 방법 설명

```
int builtin_cmd(char **argv)
{
        char *cmd = argv[0];

        if(!strcmp(cmd, "quit")){
                exit(0);
        }
        return 0;
}
```

Strcmp는 인자로 받은 두개의 문자열이 일치하면 0을 반환하는 함수이다.

위의 조건을 해석해보면 cmd(입력값)가 quit과 일치하면 exit을 실행한다고 볼 수 있다.

```
if(!builtin_cmd(argv)){
```

Trace02를 위해 조건문으로 들어갔지만, 상관없이 argv(입력값)을 받아서, 위의 함수를 실행한다고 볼 수 있다.

**Trace 번호 (02)**

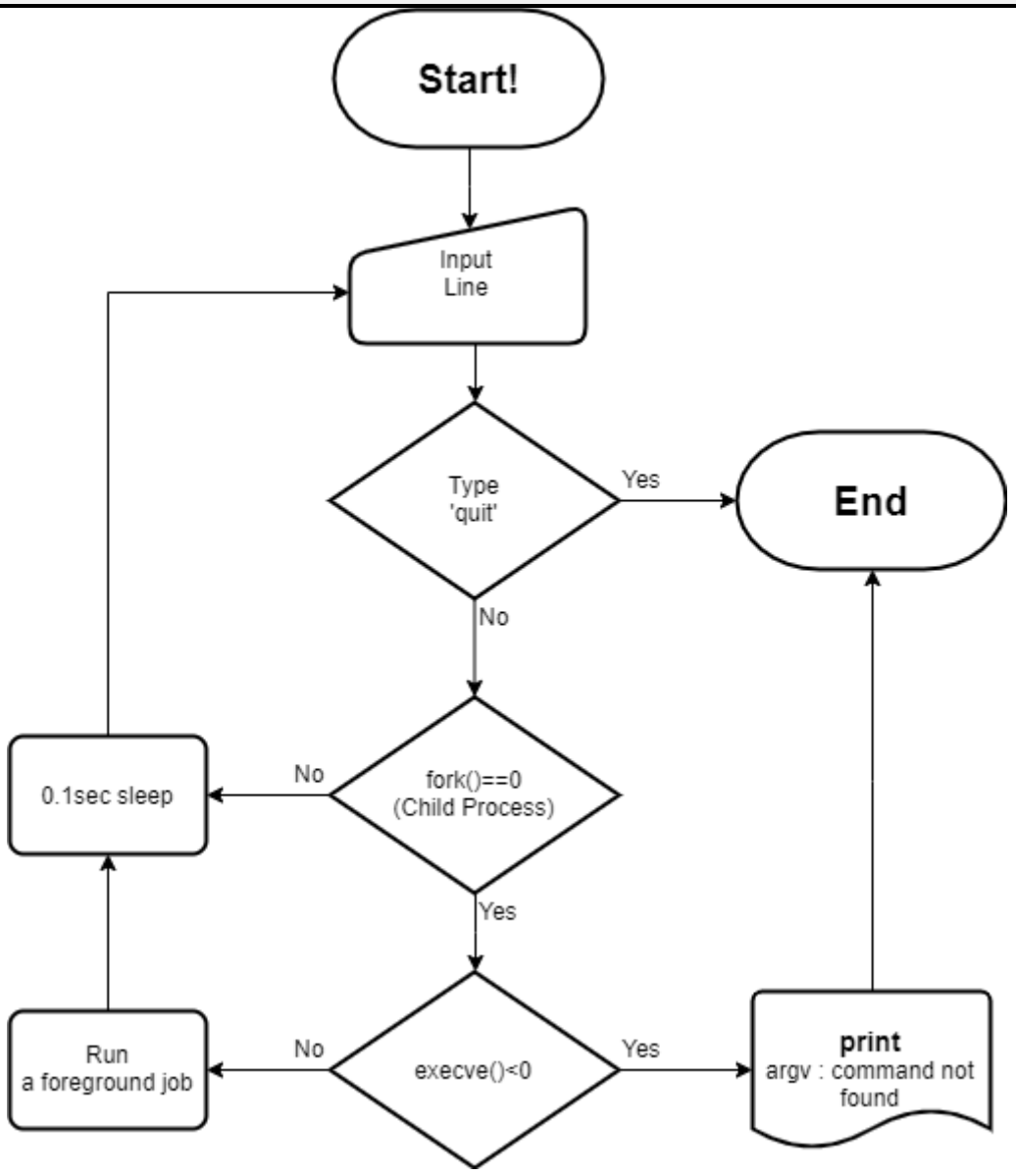
```
b201602057@2019sp:~/shlab-handout$ ./sdriver -V -t 02 -s ./tsh
Running trace02.txt...
Success: The test and reference outputs for trace02.txt matched!
Test output:
#
# trace02.txt - Run a foreground job that prints an environment variable
#
# IMPORTANT: You must pass this trace before attempting any later
# traces.  In order to synchronize with your child jobs, the driver
# relies on your shell properly setting the environment.
OSTYPE=/home/sys01/b201602057/bin:/home/sys01/b201602057/.local/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

Reference output:
#
# trace02.txt - Run a foreground job that prints an environment variable
#
# IMPORTANT: You must pass this trace before attempting any later
# traces.  In order to synchronize with your child jobs, the driver
# relies on your shell properly setting the environment.
OSTYPE=/home/sys01/b201602057/bin:/home/sys01/b201602057/.local/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

b201602057@2019sp:~/shlab-handout$
```

Ref와 비교하여 일치함.

```
b201602057@2019sp:~/shlab-handout$ ./tsh
eslab_tsh> ./myenv
OSTYPE=/home/sys01/b201602057/bin:/home/sys01/b201602057/.local/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
eslab_tsh> quit
b201602057@2019sp:~/shlab-handout$
```

Myenv를 foreground의 형태로 실행함을 확인.

각 trace 별 플로우 차트



Trace 해결 방법 설명

```
void eval(char *cmdline)
{
        char *argv[MAXARGS];

        parseline(cmdline, argv);
        if(!builtin_cmd(argv)){
                if(!fork()){
                        if(execve(argv[0], argv, environ) < 0){
                                printf("%s : notfound\n\n", *argv);
                                exit(0);
                        }
                }
                usleep(100000);
        }
        return;
}
```

비슷하게 입력을 받고 시작한다. 첫번째 if문에서는 입력 값이 해당함수에서 0을 반환 받으면 분기 안으로 들어갈 수 있다. 0을 반환 받지 못하는 경우는 trace01의 조건처럼 quit를 입력받아 종료한다는 경우밖에 없다.

그 다음은 fork인데 fork는 child일 때 0을 반환하므로 child를 사용할 우리는 !fork()해주어 fork가 0일 때를 조건으로 주었다.

그 다음은 execve값이 0보다 작다는 조건인데, 이 경우는 찾아보니 -1이 나올 경우가 있는데 실패했을 경우이다. 우리는 실패했을 때를 예외처리를 해주어야 하므로, if문으로 상황을 주고, 실패했을 때 종료하도록 한다.

그 이후는 0.1초간 쉬고 return한다.