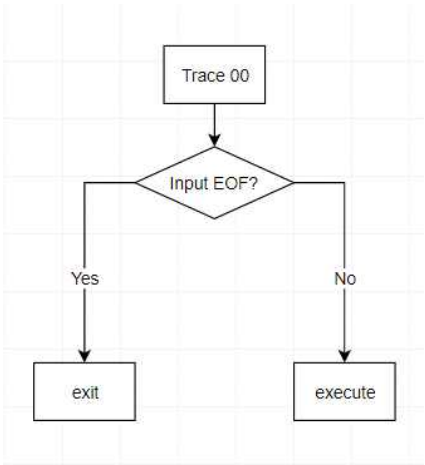
2018 시스템 프로그래밍
- Lab 06 -

제출일자	2018.
분 반	00
이 름	노효근
학 번	201502049

Trace 00

```
a201502049@2018-sp:~/shlab-handout$ ./sdriver -V -t 00 -s./tshref
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.
#

a201502049@2018-sp:~/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.
#

a201502049@2018-sp:~/shlab-handout$
```

각 trace 별 플로우 차트



trace 해결 방법 설명

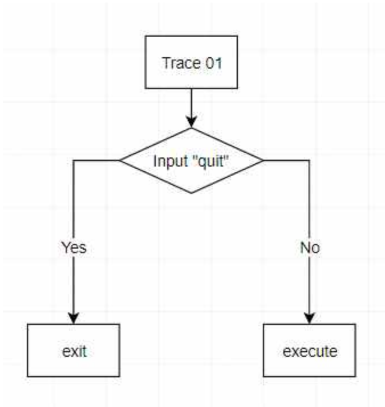
```
if (feof(stdin)) { /* End of file (ctrl-d) */
    fflush(stdout);
    fflush(stderr);
    exit(0);
}
```

- fflush로 input 값이 EOF가 들어오면 명령 shell창을 종료한다.

Trace 번호 01

```
a201502049@2018-sp:~/shlab-handout$ ./sdriver -V -t 01 -s ./tshref
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.
#

a201502049@2018-sp:~/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.
#
```

각 trace 별 플로우 차트



trace 해결 방법 설명

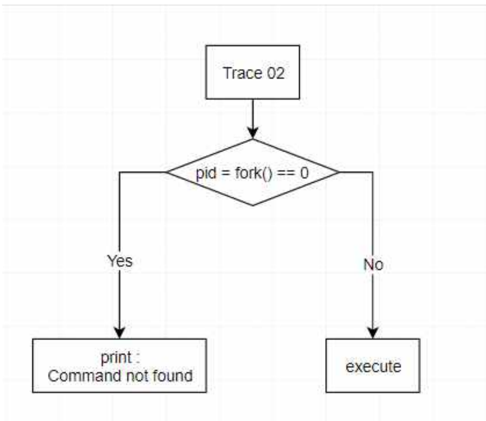
```
void eval(char *cmdline)
{
    char *argv[MAXARGS];
    // [illegible] parseline[illegible]
    parseline(cmdline,argv);
    // parsing[illegible]
    builtin_cmd(argv);
    return;
}

int builtin_cmd(char **argv)
{
    char *cmd = argv[0];
    if(!strcmp(cmd, "quit")){
        exit(0);
    }

    return 0;
}
```

- eval( )함수에서 입력 받은 명령어를 파싱한다. 파싱된 명령어를 builtin_cmd( )함수로 전달한다. 해당 명령어가 "quit"인 경우를 비교하여 쉘을 종료하도로 하게 한다.

Trace 번호 02

```
a201502049@2018-sp:~/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=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/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=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```
void eval(char *cmdline)
{
    char *argv[MAXARGS];
    pid_t pid;
    // [illegible]
    parseline(cmdline,argv);
    // [illegible]
    builtin_cmd(argv);

    if(!builtin_cmd(argv)){
        if(pid=fork()==0){
            if((execve(argv[0], argv, environ)<0)){
                printf("%s : Command not found\n", argv);
                exit(0);
            }
        }
    }
    return;
}
```

- fork() 함수를 부르면 자식 프로세스를 만들게 된다. 부모의 프로세스 pid는 0이 아니고, 자식 프로세스의 pid 는 항상 0으로 생성된다. 조건문에 따라서 자식프로세스는 해당 print 문을 실행하게 된다.