2018 시스템 프로그래밍
- Lab 06 -

제출일자	2018.11.13
분 반	02
이 름	장 수 훈
학 번	201402414

Trace 00

./sdriver tshref

```
c201402414@2018-sp:~/shell/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.
#
```

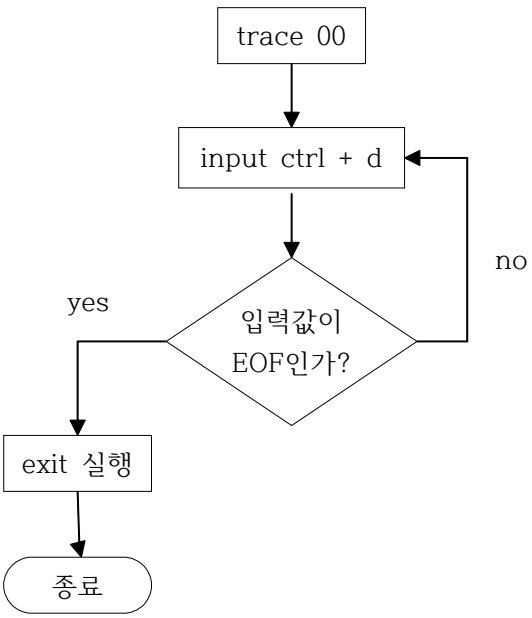
./sdriver tsh

```
c201402414@2018-sp:~/shell/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.
#
```

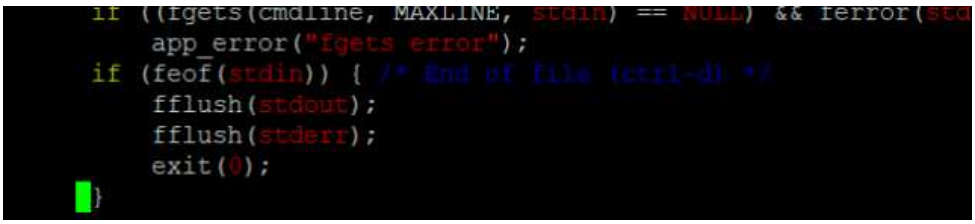
./tsh

```
c201402414@2018-sp:~/shell/shlab-handout$ ./tsh
eslab_tsh> c201402414@2018-sp:~/shell/shlab-handout$
c201402414@2018-sp:~/shell/shlab-handout$
```

각 trace 별 플로우 차트



trace 해결 방법 설명



메인함수에 이미 구현되어있다. ctrl+d입력시 exit 호출해서 종료

Trace 01

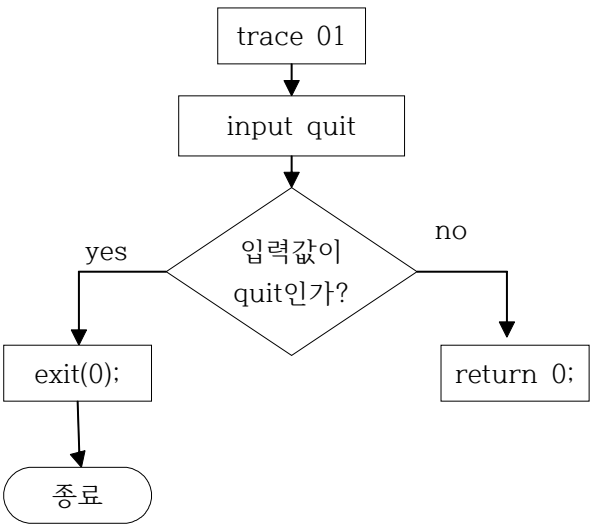
./sdriver tshref

```
c201402414@2018-sp:~/shell/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.
#
```

./sdriver tsh

```
c201402414@2018-sp:~/shell/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

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

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

    return 0;
}
```

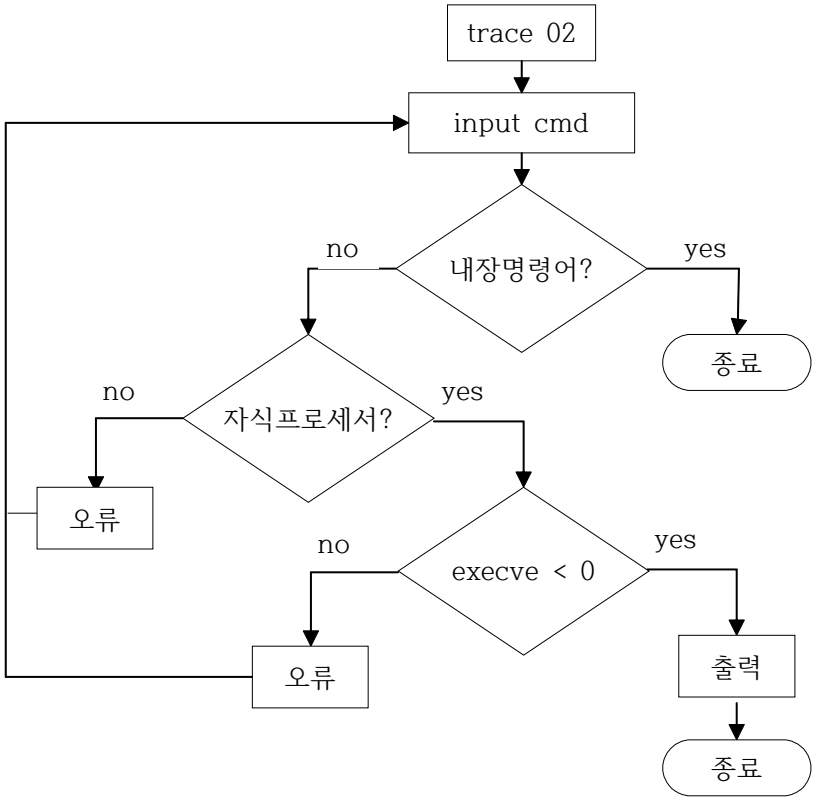
eval에서 전달된 명령어를 cmd에 저장하고 cmd의 명령어가 quit일 경우 exit(0)을 하여 종료하고 아니면 0리턴

Trace 02

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 02 -s ./tshref
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
# ce02.txt - Run a foreground job that prints an environment variable
OSTYPE=/home/sys02/c201402414/bin:/home/sys02/c201402414/.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
# ce02.txt - Run a foreground job that prints an environment variable
OSTYPE=/home/sys02/c201402414/bin:/home/sys02/c201402414/.local/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

```
c201402414@2018-sp:~/shell/shlab-handout$ ./sdriver -V -t 02 -s ./tsh
Running trace02.txt...
Oops: test and reference outputs for trace02.txt differed.

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
# ce02.txt - Run a foreground job that prints an environment variable

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
# ce02.txt - Run a foreground job that prints an environment variable
OSTYPE=/home/sys02/c201402414/bin:/home/sys02/c201402414/.local/bin:/usr/local/s
bin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```
void eval(char *cmdline)
{
    char *argv[MAXARGS];
    pid_t pid;
    //명령어를 parseline을 통해 분리
    parseline(cmdline, 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);
            }
        }
    }


    // parsing된 명령어를 전달
    //builtin_cmd(argv);

    return;
}
```

if pid = fork()를 통해 자식 프로세서인지 확인하고 자식이 0이면 실행, execve를 실행 exeve는 리턴안하고 fork는 2번리턴이 된다. 명령어를 입력하고 evel에서 판별하여 자식프로세서를 만들고 부모는 pid를 리턴하며 자신은 0을 리턴한다.