2023 시스템 프로그래밍

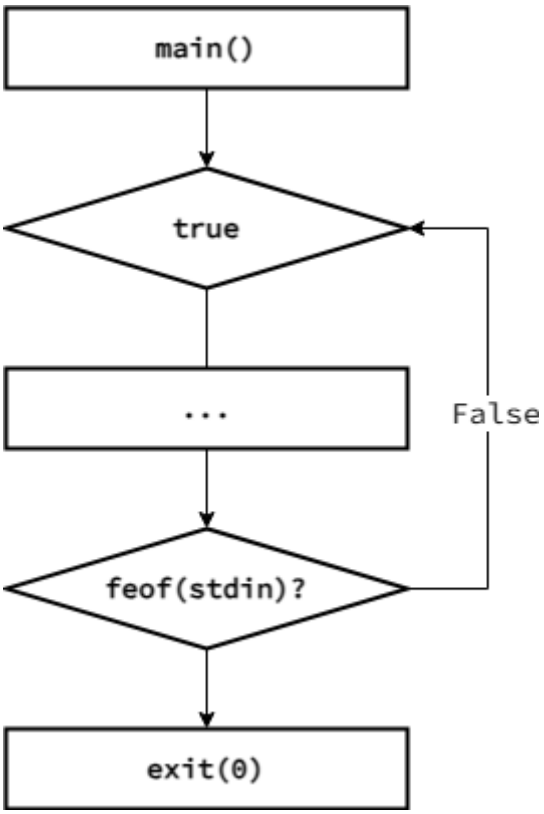
- Lab 01 -

제출일자	2023. 10. 31.
분 반	00
이 름	김재덕
학 번	202104340

Trace 번호 (00)

```
~/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.
#
```

각 trace 별 플로우 차트



trace 해결 방법 설명

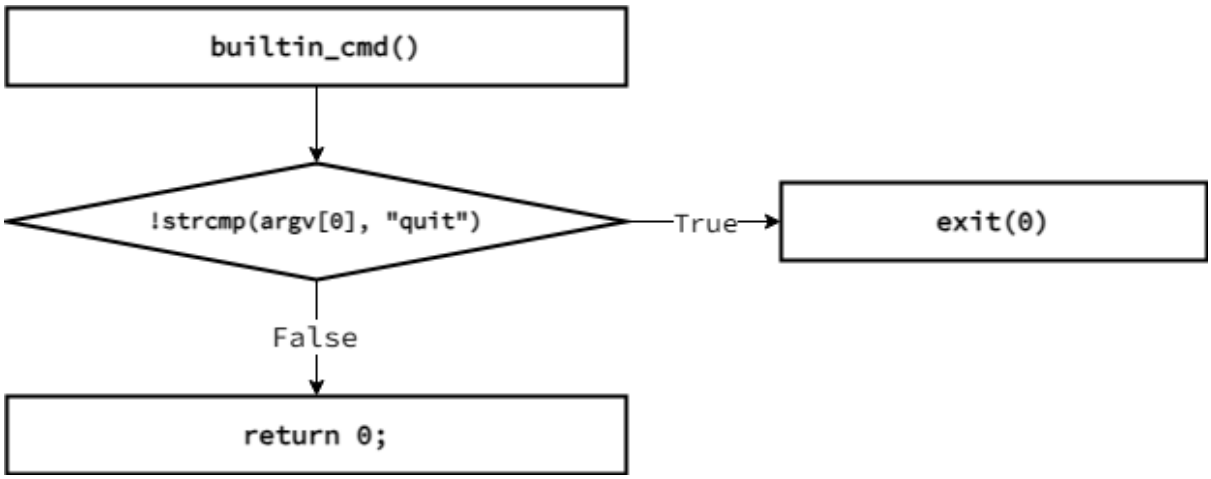
- Ctrl + D를 통해 EOF (End-Of-File)를 입력하면 셸이 종료된다.

```
/* Execute the shell's read/eval loop */
while (1) {

    /* Read command line */
    if (emit_prompt) {
        printf("%s", prompt);
        fflush(stdout);
    }
    if ((fgets(cmdline, MAXLINE, stdin) == NULL) && ferror(stdin))
        app_error("fgets error");
    if (feof(stdin)) { /* End of file (ctrl-d) */
        fflush(stdout);
        fflush(stderr);
        exit(0);
    }

    /* Evaluate the command line */
    eval(cmdline);
    fflush(stdout);
    fflush(stdout);
}
```

Trace 번호 (01)

```
~/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

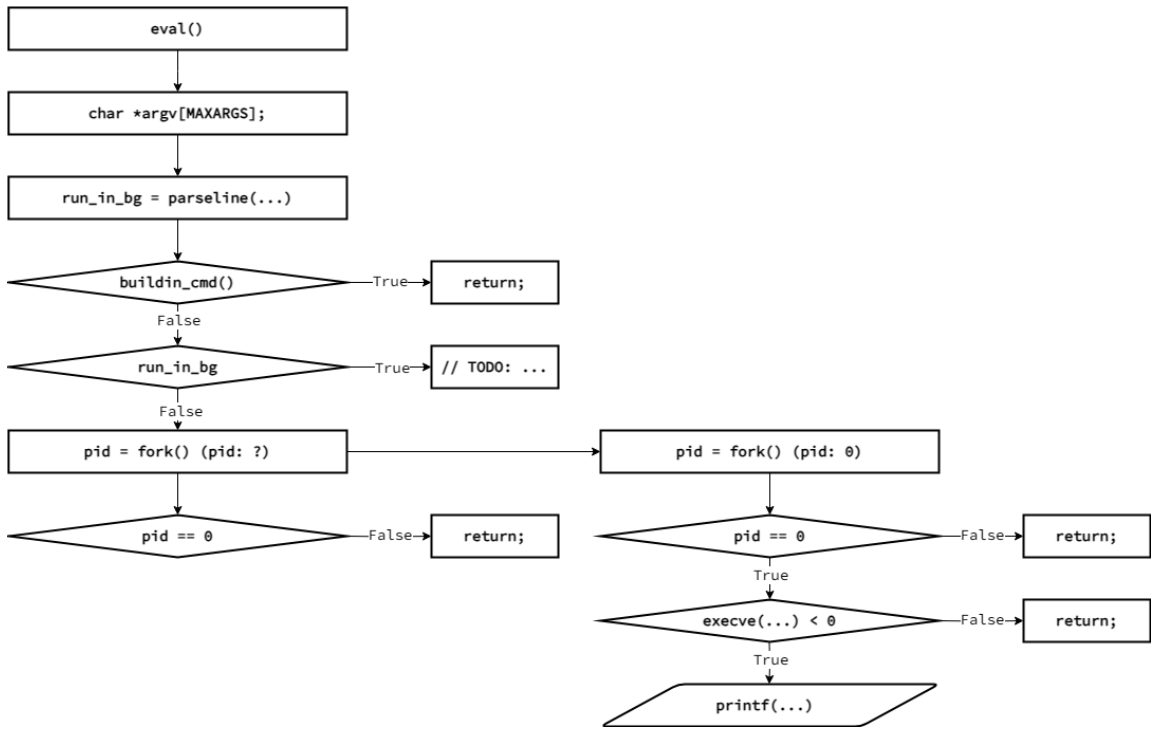
- eval()에서 builtin_cmd() 함수를 호출하여, 빌트-인 명령어를 처리한다.

```
int builtin_cmd(char **argv)
{
    if (!strcmp(argv[0], "exit") || !strcmp(argv[0], "quit")) {
        exit(0);

        // return 1;
    }

    return 0;
}
```

Trace 번호 (02)

```
~/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:/usr/local/games:/snap/bin

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:/usr/local/games:/snap/bin
```

각 trace 별 플로우 차트



trace 해결 방법 설명

- fork()로 현재 실행 중인 프로세스를 복제하고, execve()로 현재 실행 중인 프로세스의 코드 영역 (code segment)과 데이터 영역 (data segment)을 주어진 경로에 위치한 실행 파일의 코드 영역과 데이터 영역으로 덮어쓴다.

```
void eval(char *cmdline)
{
    // 명령어 인자 배열을 선언한다.
    char *argv[MAXARGS];

    // `parseline()`의 반환값이 1이라면, 이 작업을
    // 백그라운드에서 실행한다.
    int run_in_bg = parseline(cmdline, argv);

    // 먼저 빌트-인 명령어인지 확인한다.
    if (builtin_cmd(argv)) return;

    // 이 작업을 포그라운드로 실행해야 하는가?
    if (!run_in_bg) {
        pid_t pid;

        // `fork()` 수행 후, 자식 프로세스인지 확인한다.
        if ((pid = fork()) == 0) {
            // NOTE: https://pubs.opengroup.org/onlinepubs/007904875/functions/exec.html
            if (execve(argv[0], argv, environ) < 0)
                printf("%s: command not found\n", argv[0]), exit(0);
        }
    } else {
        // TODO: ...
    }
}
```