2023 시스템 프로그래밍

- Lab 02 -

제출일자	2023. 11. 07.
분 반	00
이 름	김재덕
학 번	202104340

Trace 번호 (03 ~ 06)

```
~/shlab-handout $ ./sdriver -U -t 03 -s ./tsh
Running trace03.txt...
Success: The test and reference outputs for trace03.txt matched!
Test output:
#
# trace03.txt - Run a synchronizing foreground job without any arguments.
#

Reference output:
#
# trace03.txt - Run a synchronizing foreground job without any arguments.
#
```

```
~/shlab-handout $ ./sdriver -U -t 04 -s ./tsh
Running trace04.txt...
Success: The test and reference outputs for trace04.txt matched!
Test output:
#
# trace04.txt - Run a foreground job with arguments.
#
tsh> quit

Reference output:
#
# trace04.txt - Run a foreground job with arguments.
#
tsh> quit
```

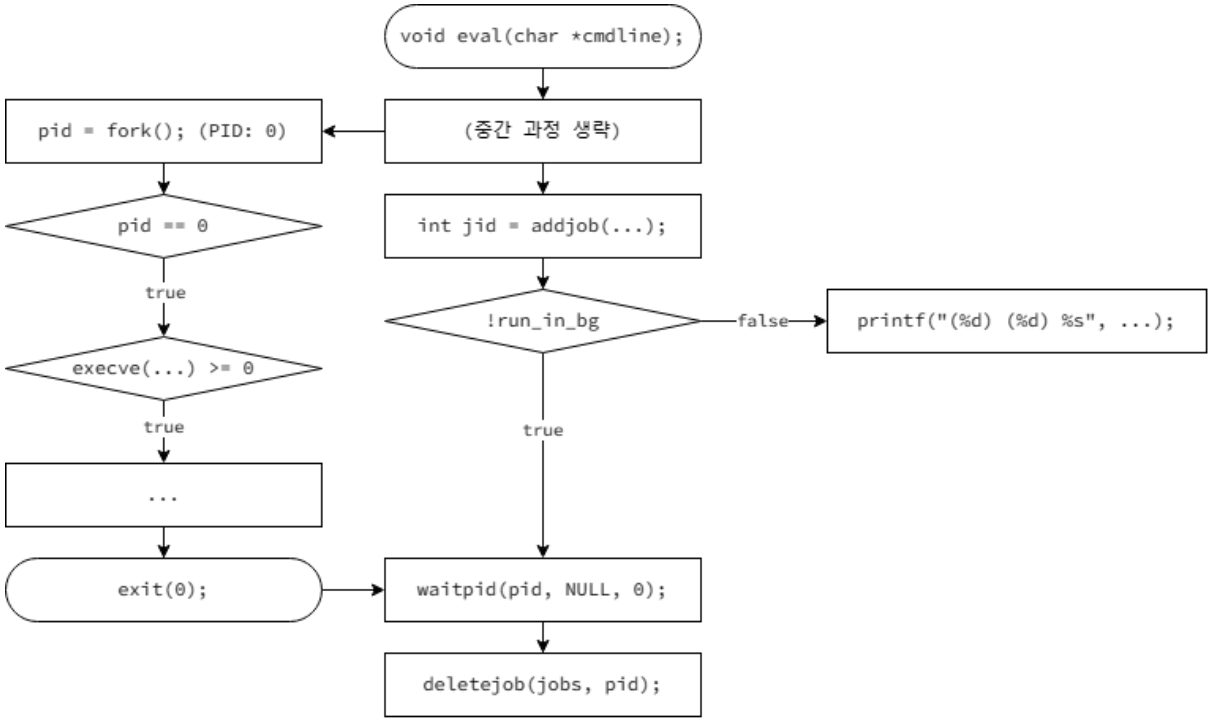
```
~/shlab-handout $ ./sdriver -U -t 05 -s ./tsh
Running trace05.txt...
Success: The test and reference outputs for trace05.txt matched!
Test output:
#
# trace05.txt - Run a background job.
#
tsh> ./myspin1 &
(1) (1175426) ./myspin1 &
tsh> quit

Reference output:
#
# trace05.txt - Run a background job.
#
tsh> ./myspin1 &
(1) (1175436) ./myspin1 &
tsh> quit
```

```
~/shlab-handout $ ./sdriver -V -t 06 -s ./tsh
Running trace06.txt...
Success: The test and reference outputs for trace06.txt matched!
Test output:
#
# trace06.txt - Run a foreground job and a background job.
#
tsh> ./myspin1 &
(1) (1175525) ./myspin1 &
tsh> ./myspin2 1

Reference output:
#
# trace06.txt - Run a foreground job and a background job.
#
tsh> ./myspin1 &
(1) (1175535) ./myspin1 &
tsh> ./myspin2 1
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```
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;

    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);
    }

    // 작업 목록에 새로운 작업을 추가한다.
    int jid = addjob(jobs, pid, ((!run_in_bg) ? FG : BG), cmdline);

    // 이 작업을 포그라운드로 실행해야 하는가?
    if (!run_in_bg) {
        // 자식 프로세스가 종료될 때까지 기다린다.
        waitpid(pid, NULL, 0);

        deletejob(jobs, pid);
    } else {
        // 작업 ID, 프로세스 ID와 명령 인수 등을 출력한다.
        printf("(%d) (%d) %s", jid, (int) pid, cmdline);
    }
}
```

- `parseline()`를 이용해 사용자로부터 입력받은 명령 줄을 공백 문자를 기준으로 적절하게 끊고, `argv` 문자열 배열에 저장한다. **이때 `parseline()`의 반환값이 `0`이라면 `!run_in_bg`의 반환값은 1이므로, 포그라운드 (foreground)로 작업을 실행해야 한다.**

- 포그라운드로 작업을 실행한다는 것은 곧 `fork()`와 `execve()`로 생성한 자식 프로세스가 종료될 때까지 기다려야 한다는 것을 뜻하므로, POSIX 시스템 콜[1]인 `waitpid()`를 이용해 자식 프로세스 종료에 대한 상태 정보를 받을 때까지 현재 쓰레드 실행을 중단한다. ("The `wait()` function shall cause the calling thread to become blocked until status information generated by child process termination is made available to the thread...") 마지막으로, 자식 프로세스의 실행이 끝나면, `deletejob()`을 이용해 작업 목록에서 현재 작업을 제거한다.

[1] https://pubs.opengroup.org/onlinepubs/9699919799/functions/wait.html

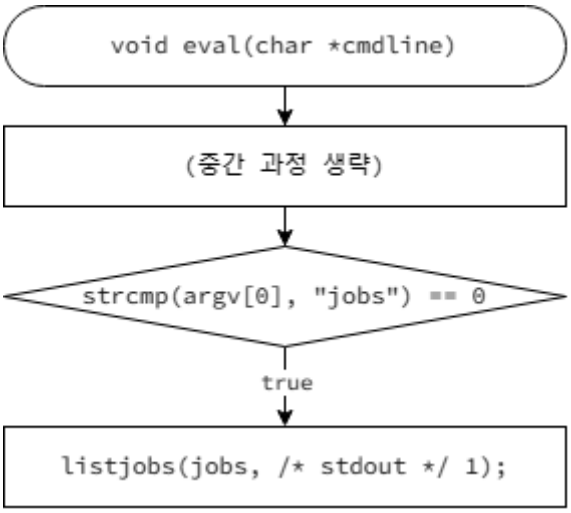
- **!run_in_bg의 반환값이 0이라면, 자식 프로세스가 종료될 때까지 따로 기다리지 않고,** 작업 ID와 프로세스 ID, 그리고 명령 줄을 출력한다.

Trace 번호 (07)

```
~/shlab-handout $ ./sdriver -V -t 07 -s ./tsh
Running trace07.txt...
Success: The test and reference outputs for trace07.txt matched!
Test output:
#
# trace07.txt - Use the jobs builtin command.
#
tsh> ./myspin1 10 &
(1) (1175614) ./myspin1 10 &
tsh> ./myspin2 10 &
(1) (1175616) ./myspin2 10 &
tsh> jobs
(1) (1175614) Running    ./myspin1 10 &
(2) (1175616) Running    ./myspin2 10 &

Reference output:
#
# trace07.txt - Use the jobs builtin command.
#
tsh> ./myspin1 10 &
(1) (1175625) ./myspin1 10 &
tsh> ./myspin2 10 &
(2) (1175628) ./myspin2 10 &
tsh> jobs
(1) (1175625) Running ./myspin1 10 &
(2) (1175628) Running ./myspin2 10 &
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```c
int builtin_cmd(char **argv)
{
    if (!strcmp(argv[0], "exit") || !strcmp(argv[0], "quit")) {
        // 셸 프로세스를 종료한다.
        exit(0);

        // return 1;
    } else if (!strcmp(argv[0], "jobs")) {
        /*
         * NOTE: On program startup, the integer file descriptors associated with
         * the streams `stdin`, `stdout`, and `stderr` are 0, 1, and 2,
         * respectively.
         */

        // 표준 출력 스트림을 이용해, 작업 목록을 출력한다.
        listjobs(jobs, 1);

        return 1;
    }

    return 0;
}
```

- `listjobs()` 함수는 `output_fd`가 가리키는 표준 입력 스트림 (`stdin`)이나 표준 출력 스트림 (`stdout`) 등의 파일 디스크립터 (file descriptor)[2]로 작업 목록을 출력하는 기능을 수행한다. GNU/Linux 계열 운영 체제에서 표준 출력 스트림을 가리키는 파일 디스크립터는 1이므로, `output_fd`의 값으로 `1`을 넣어준다.

[2] https://pubs.opengroup.org/onlinepubs/9699919799/functions/write.html