2019 시스템 프로그래밍
- Lab 08 -

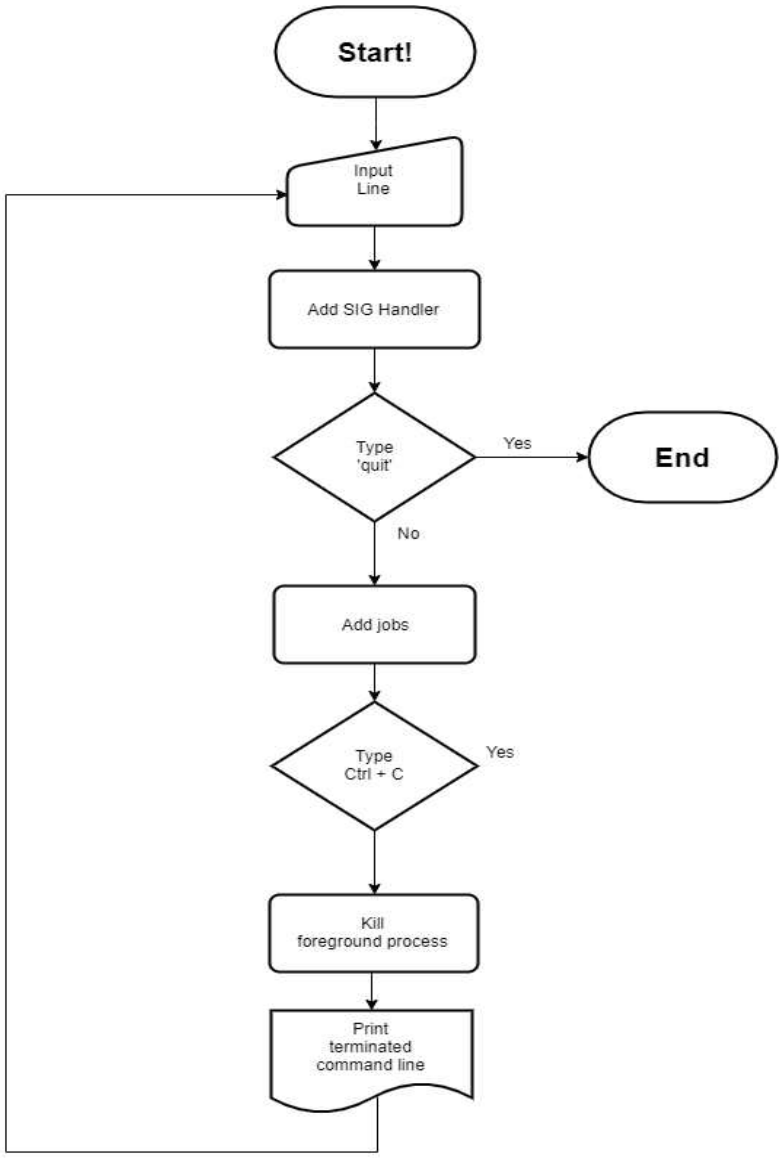
제출일자	2019.11.25
분 반	01
이 름	임준규
학 번	201602057

Trace 번호 (08 - 12)

```
b201602057@2019sp:~/shlab-handout$ ./sdriver -V -t 08 -s ./tsh
Running trace08.txt...
Success: The test and reference outputs for trace08.txt matched!
Test output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (15909) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (15917) terminated by signal 2
tsh> quit
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```
void eval(char *cmdline)
{
        char *argv[MAXARGS];
        pid_t pid;
        int bg;
        sigset_t mask;
        bg = parseline(cmdline, argv);

        if(argv[0] == NULL)
                return;
        sigemptyset(&mask);
        sigaddset(&mask, SIGCHLD);
        sigaddset(&mask, SIGINT);
        sigaddset(&mask, SIGTSTP);
        sigprocmask(SIG_BLOCK, &mask, NULL);
        if(!builtin_cmd(argv)){
                pid = fork();
                if(pid == 0){
                        setpgid(0, 0);
                        sigprocmask(SIG_UNBLOCK, &mask, NULL);
                        if(execve(argv[0], argv, environ) < 0){
                                printf("%s : Command not found\n\n", argv[0]);
                                exit(0);
                        }
                }
                addjob(jobs, pid, (bg == 1 ? BG : FG), cmdline);
                sigprocmask(SIG_UNBLOCK, &mask, NULL);

                if(!bg){
                        while(1){
                                if(pid != fgpid(jobs))
                                        break;
                                else
                                        sleep(1);
                        }
                }
                else{
                        printf("(%d) (%d) %s", pid2jid(pid), pid, cmdline);
                }
        }
        return;
}
```

```
/*
 * sigint_handler - The kernel sends a SIGINT to the shell whenver the
 *    user types ctrl-c at the keyboard.  Catch it and send it along
 *    to the foreground job.
 */
void sigint_handler(int sig)
{
        pid_t pid = fgpid(jobs);

        if(pid != 0){
                kill(-pid, SIGINT);
        }
        return;
}
```

```
void sigchld_handler(int sig)
{
        int child_status = 0;
        pid_t pid;
        while((pid = waitpid(-1, &child_status, WNOHANG|WUNTRACED)) > 0){
                if(WIFSIGNALED(child_status)){
                        printf("Job [%d] (%d) terminated by signal %d\n", pid2jid(pid), pid, WTERMSIG(child_status));
                        deletejob(jobs,pid);
                }
                else if(WIFSTOPPED(child_status)){
                        getjobpid(jobs, pid)->state = ST;
                        printf("Job [%d] (%d) stopped by signal %d\n", pid2jid(pid), (int)pid, WSTOPSIG(child_status));
                }
                else{
                        deletejob(jobs,pid);
                }
        }
        return;
}
```

트레이스 8은 Ctrl+C를 입력 받았을 시에 sigint_handler를 통하여 process를 kill하는 동작이다.

입력을 받고, 입력을 확인하였을 때 위에서 지정한 Ctrl+C라면, sigchld로 이동하고 while이하 조건에서 조건을 확인해준 후에, 조건에 맞게 이동하여 kill할 프로그램 내용을 출력해주고 kill해준다.

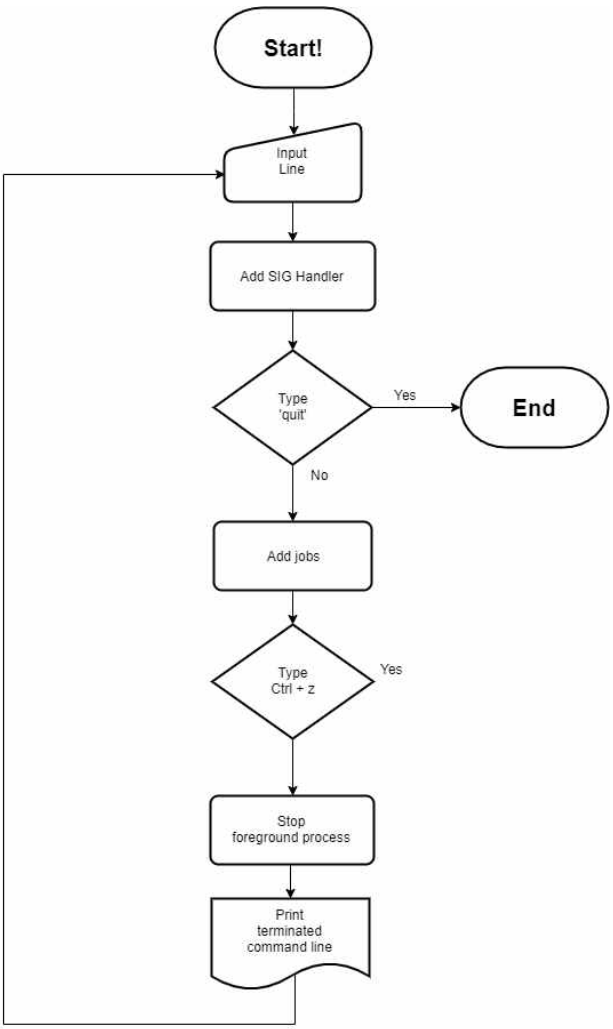
최초에 addset을 통해 핸들러들을 넣어주고, 입력을 우선 받아야 하므로 처음에는 블록 시켜둔다. 이후에 입력하고 예외처리를 마친 후에, 핸들러를 블록시켜주어 핸들러를 사용할 수 있게 만들어 주고, 핸들러에 해당하는 입력을 받았을 시에 특정 동작을 하도록 한다.

Trace 번호 (09)

```
b2016020257@2019sp:~/shlab-handout$ ./sdriver -V -t 09 -s ./tsh
Running trace09.txt...
Success: The test and reference outputs for trace09.txt matched!
Test output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (15525) stopped by signal 20
tsh> jobs
(1) (15525) Stopped ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (15533) stopped by signal 20
tsh> jobs
(1) (15533) Stopped ./mytstpp
```

각 trace 별 플로우 차트



trace 해결 방법 설명

```
/*
 * sigtstp_handler - The kernel sends a SIGTSTP to the shell whenever
 *     the user types ctrl-z at the keyboard. Catch it and suspend the
 *     foreground job by sending it a SIGTSTP.
 */
void sigtstp_handler(int sig)
{
        pid_t pid = fgpid(jobs);
        if(pid > 0)
                kill(-pid, SIGTSTP);
        return;
}
```

```
void sigchld_handler(int sig)
{
        int child_status = 0;
        pid_t pid;
        while((pid = waitpid(-1, &child_status, WNOHANG|WUNTRACED)) > 0){
                if(WIFSIGNALED(child_status)){
                        printf("Job [%d] (%d) terminated by signal %d\n", pid2jid(pid), pid, WTERMSIG(child_status));
                        deletejob(jobs,pid);
                }
                else if(WIFSTOPPED(child_status)){
                        getjobpid(jobs, pid)->state = ST;
                        printf("Job [%d] (%d) stopped by signal %d\n", pid2jid(pid), (int)pid, WSTOPSIG(child_status));
                }
                else{
                        deletejob(jobs,pid);
                }
        }
        return;
}
```

전체적으로 위의 trace 08과 유사하나, 입력과 호출되는 시그널 핸들러가 다르다.

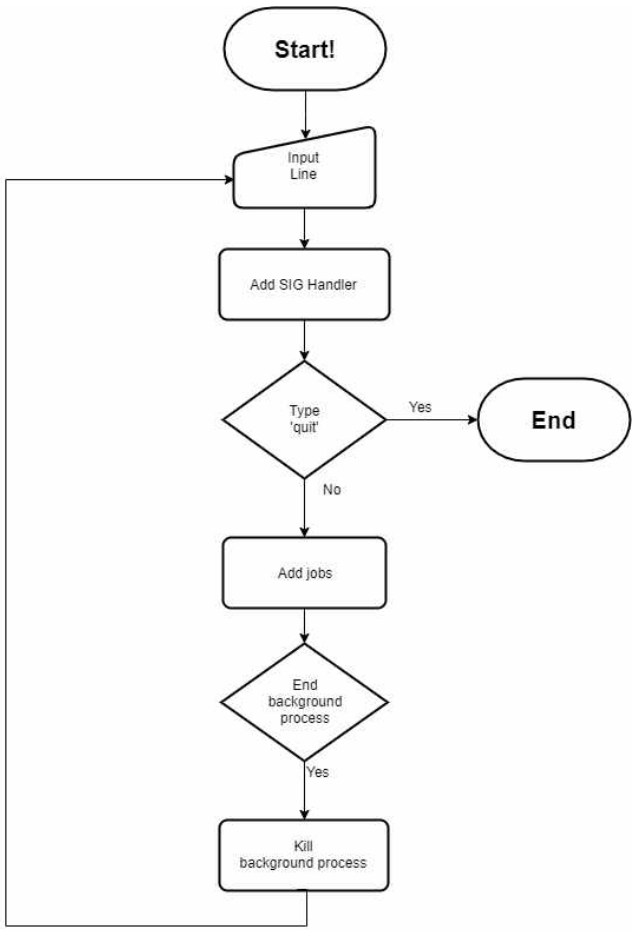
설명해 보자면, 똑같이 위의 sigchld까지 와서 while문에 들어오게 되면 두 번째 조건으로 들어가는데, else if문을 보면 처음에 getjobpid를 이용하여, 프로세스를 가져와서 상태를 ST(stop)로 만들어준다. 이후에 stop된 프로세스의 정보를 출력해준다.

Trace 번호 (10)

```
b201602057@2019sp:~/shlab-handout$ ./sdriver -V -t 10 -s ./tsh
Running trace10.txt...
Success: The test and reference outputs for trace10.txt matched!
Test output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (17918) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

Reference output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (17929) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit
```

각 trace 별 플로우 차트



trace 해결 방법 설명

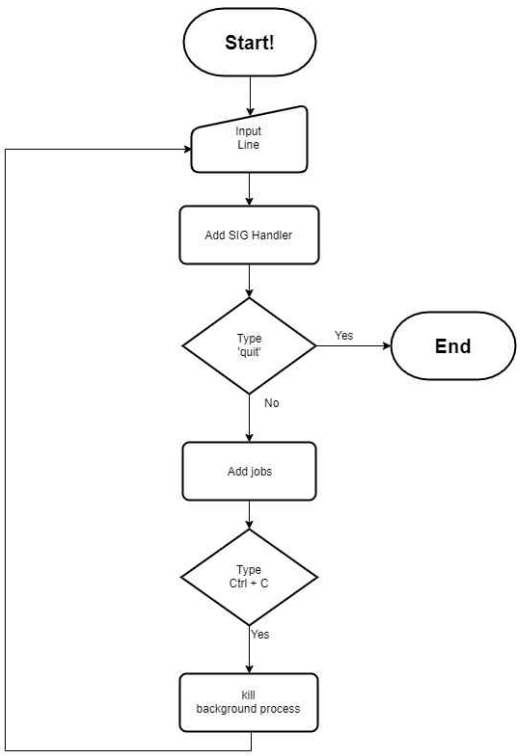
자동으로 구현되었습니다. 시그널을 받아 종료하는 것이므로 앞에서 한 구현으로 사용이 가능하며, 이후 프린트는 getjobpid를 이용하여 업데이트하고 프린트해줍니다.

Trace 번호 (11)

```
b201602057@2019sp:~/shlab-handout$ ./sdriver -V -t 11 -s ./tsh
Running trace11.txt...
Success: The test and reference outputs for trace11.txt matched!
Test output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (18449) terminated by signal 2
tsh> quit

Reference output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (18457) terminated by signal 2
tsh> quit
```

각 trace 별 플로우 차트



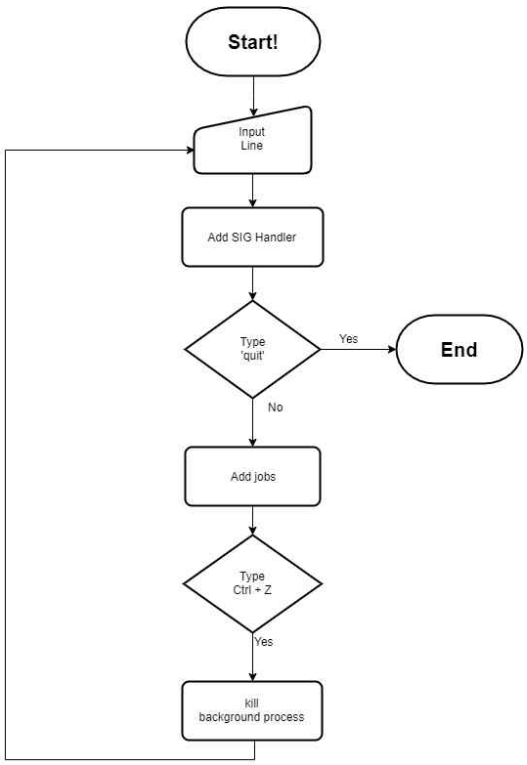
trace 해결 방법 설명
자식프로세스가 호출되면서 같은 함수가 다시 호출되는데, 거기서 sigint를 받아서 그 함수의 기능인 kill을 수행하기 때문이다.

Trace 번호 (12)

```
b201602057@2019sp:~/shlab-handout$ ./sdriver -V -t 12 -s ./tsh
Running trace12.txt...
Success: The test and reference outputs for trace12.txt matched!
Test output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (18498) stopped by signal 20
tsh> jobs
(1) (18498) Stopped ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (18506) stopped by signal 20
tsh> jobs
(1) (18506) Stopped ./mytstps
```

각 trace 별 플로우 차트



trace 해결 방법 설명

12과 똑같이 호출한 후에 sigtstp를 호출하여 stop시킨다.