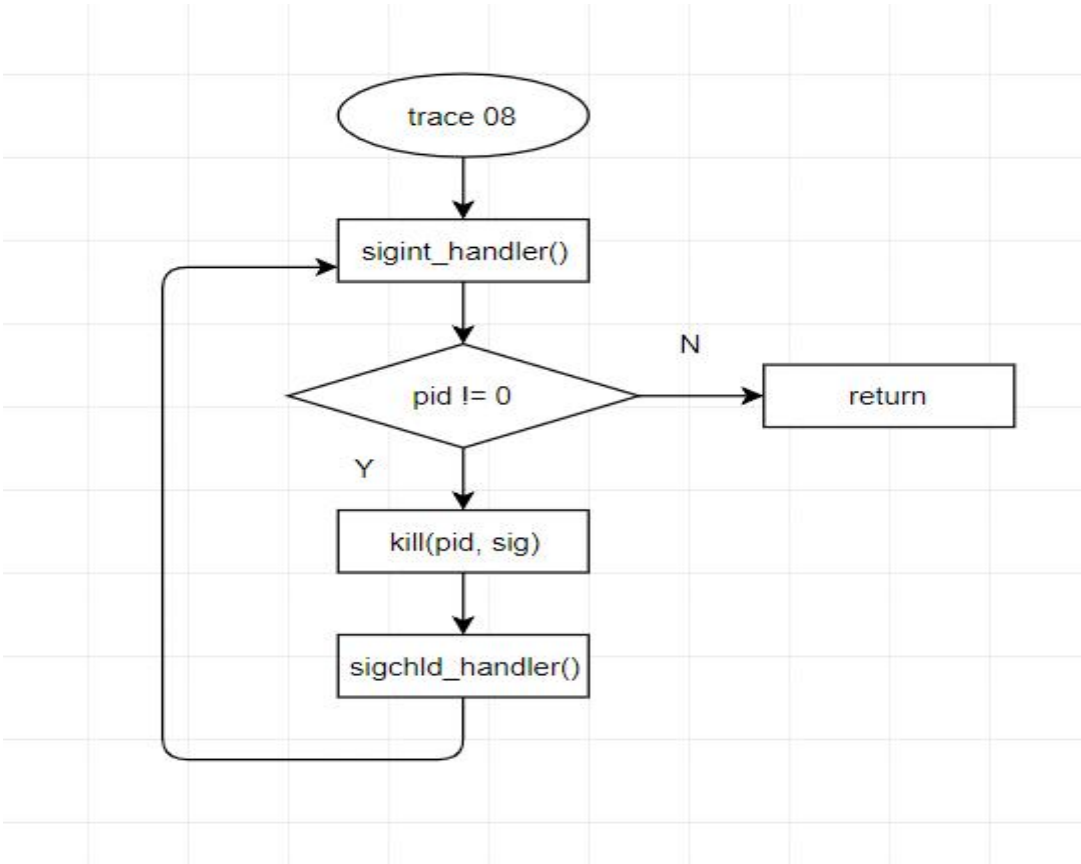
2018 시스템 프로그래밍
- Lab 08 -

제출일자	2018.11.23.
분　　반	00
이　　름	김민기
학　　번	201502023

Trace 08

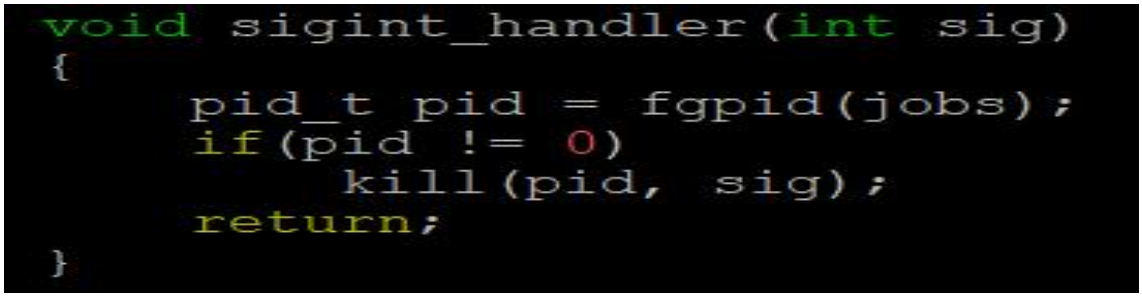
```
a201502023@2018-sp: ~/shlab-handout
./python.py
a201502023@2018-sp:~/shlab-handout$ ./sdriver -t 08 -s ./tsh -V
Running trace08.txt...
Success: The test and reference outputs for trace08.txt matched!
Test output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (4191) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (4199) terminated by signal 2
tsh> quit
```

trace 08 플로우 차트

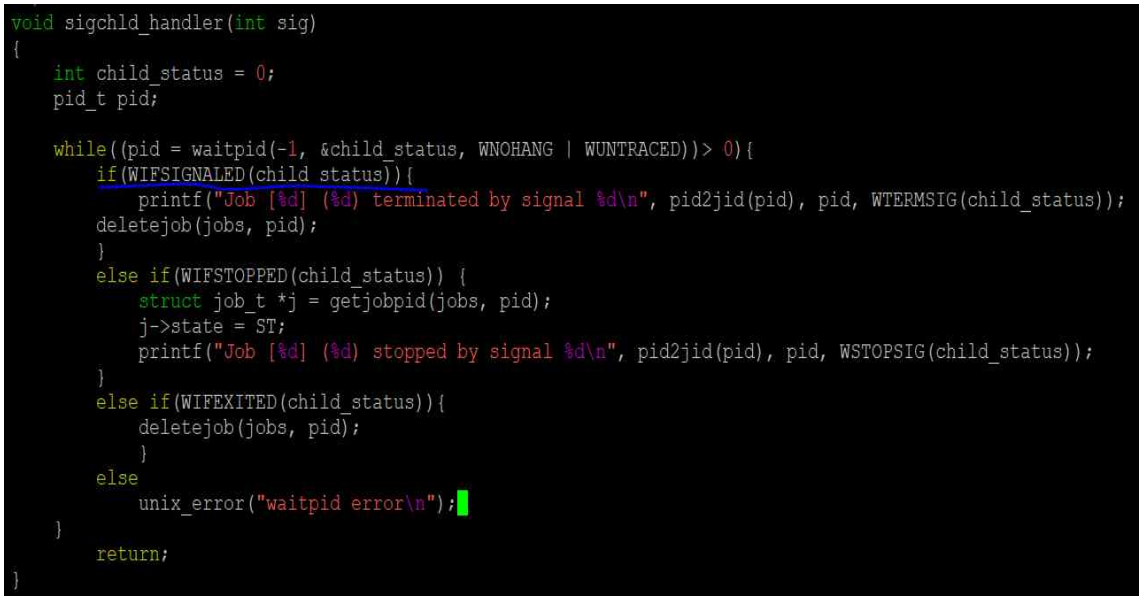


trace 08 해결 방법 설명

fgpid(jobs)를 통해 포그라운드 프로세스의 pid을 저장한다. (pid != 0)인 경우에 kill 함수를 통해서 지정한 pid를 가지는 프로세스에 시그널을 전달하고 sigchld 시그널을 받아서 sigchld_handler()를 호출한다. sigchld_handler()에서 while문으로 자식프로세스이 종료를 대기한다. WIFSIGNALED(child_status)로 자식프로세스 종료를 체크하여 자식프로세스가 종료되면 참을 반환되어 해당 프린트문을 출력하고 deletejob()을 통해 자식프로세스를 목록에서 제거한다.



```c
void sigint_handler(int sig)
{
        pid_t pid = fgpid(jobs);
        if(pid != 0)
            kill(pid, sig);
        return;
}
```

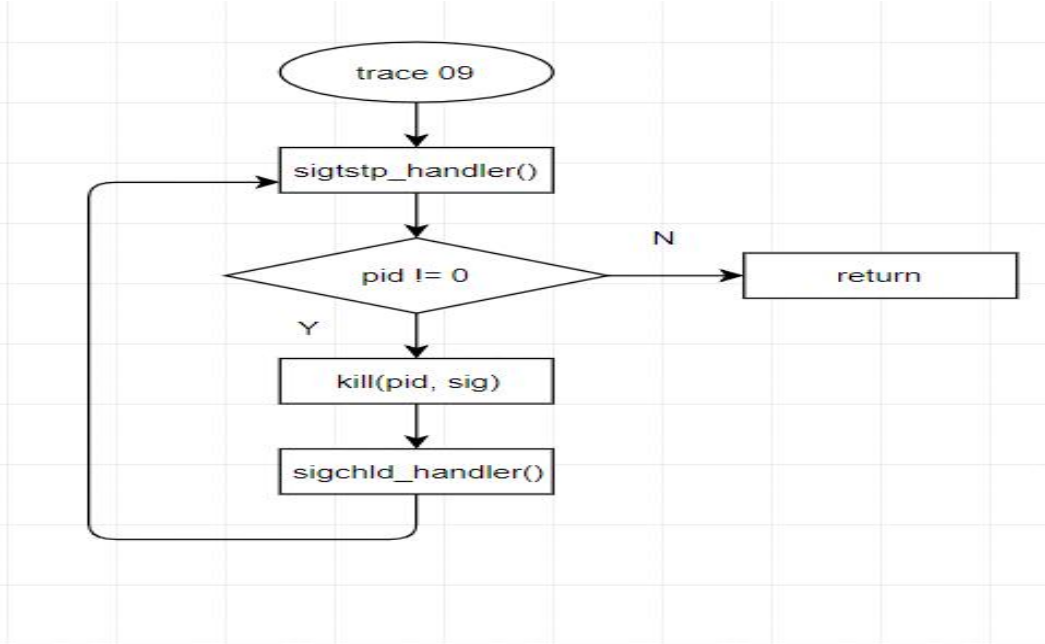


```c
void sigchld_handler(int sig)
{
    int child_status = 0;
    pid_t pid;

    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)) {
            struct job_t *j = getjobpid(jobs, pid);
            j->state = ST;
            printf("Job [%d] (%d) stopped by signal %d\n", pid2jid(pid), pid, WSTOPSIG(child_status));
        }
        else if(WIFEXITED(child_status)){
            deletejob(jobs, pid);
            }
        else
            unix_error("waitpid error\n");
    }
        return;
}
```

Trace 09

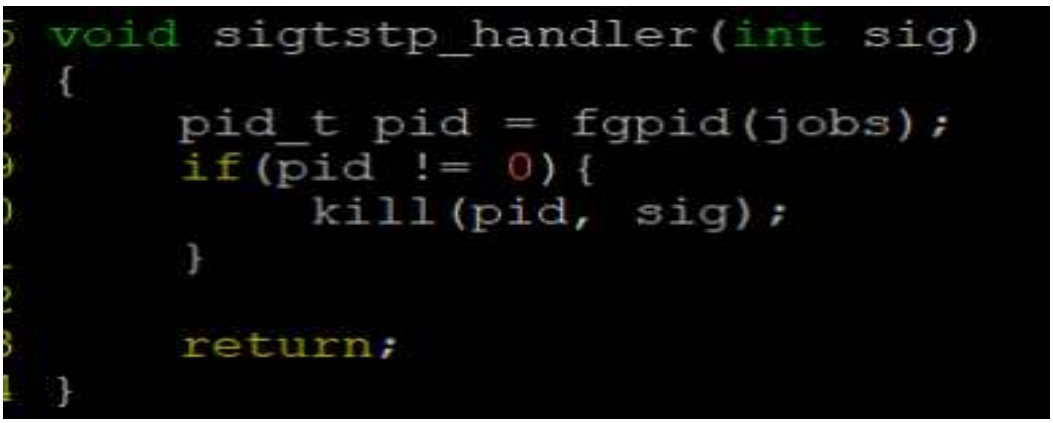
```
a201502023@2018-sp: ~/shlab-handout

a201502023@2018-sp:~/shlab-handout$ ./sdriver -t 09 -s ./tsh -V
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] (4564) stopped by signal 20
tsh> jobs
(1) (4564) Stopped    ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (4574) stopped by signal 20
tsh> jobs
(1) (4574) Stopped ./mytstpp

a201502023@2018-sp:~/shlab-handout$
```

trace 09 플로우 차트



trace 09 해결 방법 설명

fgpid(jobs)를 통해 포그라운드 프로세스의 pid을 저장한다. (pid != 0)인 경우에 kill 함수를 통해서 지정한 pid를 가지는 프로세스에 시그널을 전달하고 sigchld 시그널을 받아서 sigchld_handler()를 호출한다. sigchld_handler()에서 while문으로 자식프로세스이 종료를 대기한다. WIFSTOPPED(child_status)로 자식프로세스의 정지 상태를 파악하여 자식프로세스가 정지되어 있으면 참이 반환되고 job의 상태를 stop으로 바꿔준다. 그리고 해당 프린트문을 출력하고 deletejob()을 통해 자식프로세스를 목록에서 제거한다.



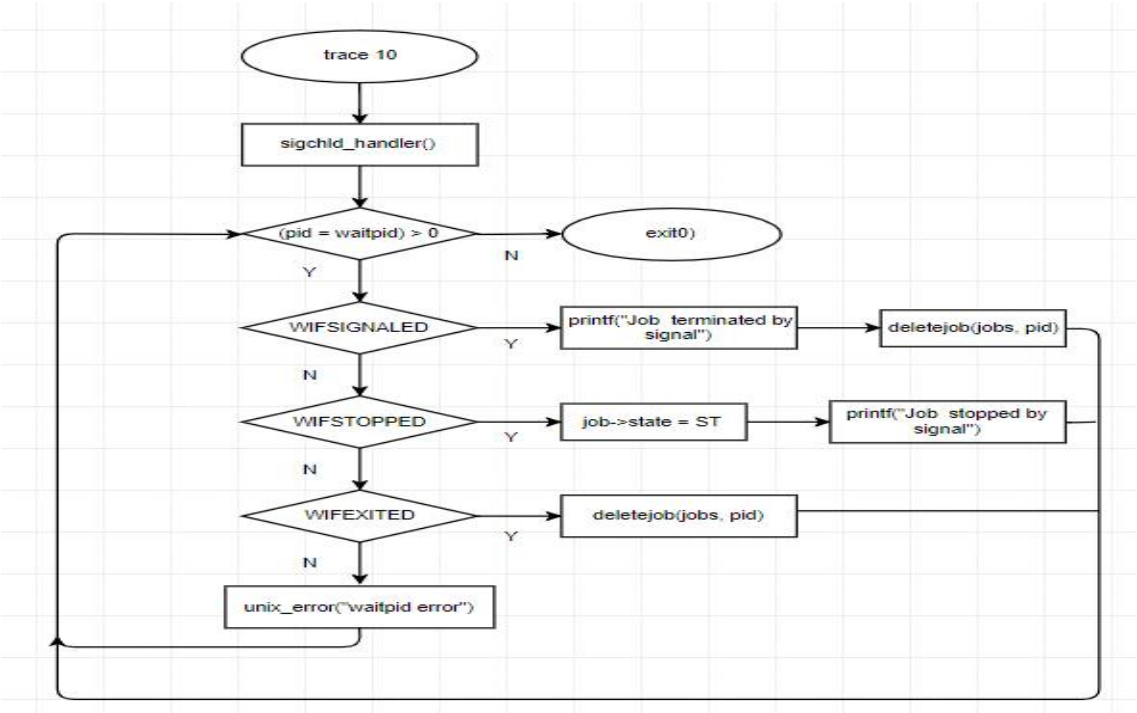
```c
void sigtstp_handler(int sig)
{
    pid_t pid = fgpid(jobs);
    if(pid != 0){
        kill(pid, sig);
    }

    return;
}
```

```c
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)) {
            struct job_t *j = getjobpid(jobs, pid);
            j->state = ST;
            printf("Job [%d] (%d) stopped by signal %d\n", pid2jid(pid), pid, WSTOPSIG(child_status));
        }
        else if(WIFEXITED(child_status)){
            deletejob(jobs, pid);
            }
        else
            unix_error("waitpid error\n");
    }
        return;
}
```

Trace 10

```
a201502023@2018-sp: ~/shlab-handout
a201502023@2018-sp:~/shlab-handout$ ./sdriver -t 10 -s ./tsh -V
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) (4605) ./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) (4615) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

a201502023@2018-sp:~/shlab-handout$
```

trace 10 플로우 차트



trace 10 해결 방법 설명

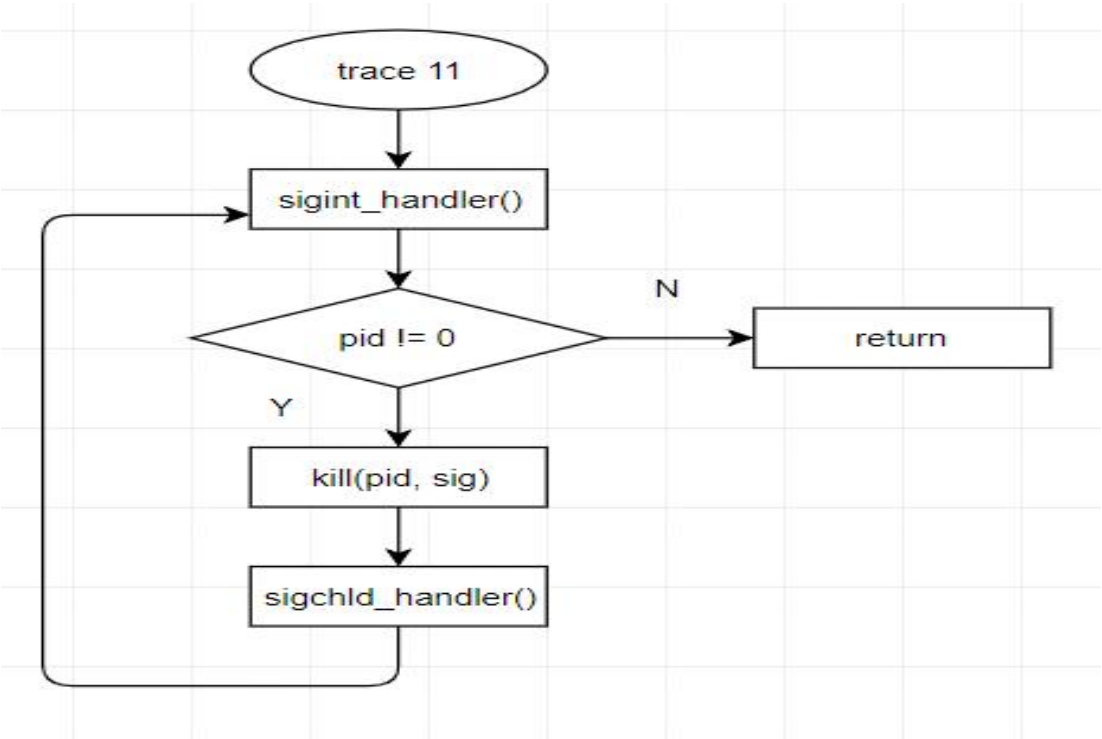
fgpid(jobs)를 통해 포그라운드 프로세스의 pid을 저장한다. (pid != 0)인 경우에 kill 함수를 통해서 지정한 pid를 가지는 프로세스에 시그널을 전달하고 sigchld 시그널을 받아서 sigchld_handler()를 호출한다. sigchld_handler()에서 while문으로 자식프로세스이 종료를 대기한다. WIFEXITED(child_status)로 백그라운드 작업을 정상 종료를 확인한다. 정상 종료 시 자식이 정상적으로 종료되었다면 0이 아닌 값을 리턴하므로, 자식프로세스가 종료될 때 까지 기다린 후에 자식프로세스가 종료되면 deletejob()을 통해 자식을 제거한다.

```
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)) {
            struct job_t *j = getjobpid(jobs, pid);
            j->state = ST;
            printf("Job [%d] (%d) stopped by signal %d\n", pid2jid(pid), pid, WSTOPSIG(child_status));
        }
        else if(WIFEXITED(child_status)){
            deletejob(jobs, pid);
            }
        else
            unix_error("waitpid error\n");
    }
        return;
}
```

Trace 11

```
a201502023@2018-sp: ~/shlab-handout
tsh> quit

a201502023@2018-sp:~/shlab-handout$ ./sdriver -t 11 -s ./tsh -V
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] (4652) terminated by signal 2
tsh> quit

Reference output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (4660) terminated by signal 2
tsh> quit

a201502023@2018-sp:~/shlab-handout$
```

trace 11 플로우 차트



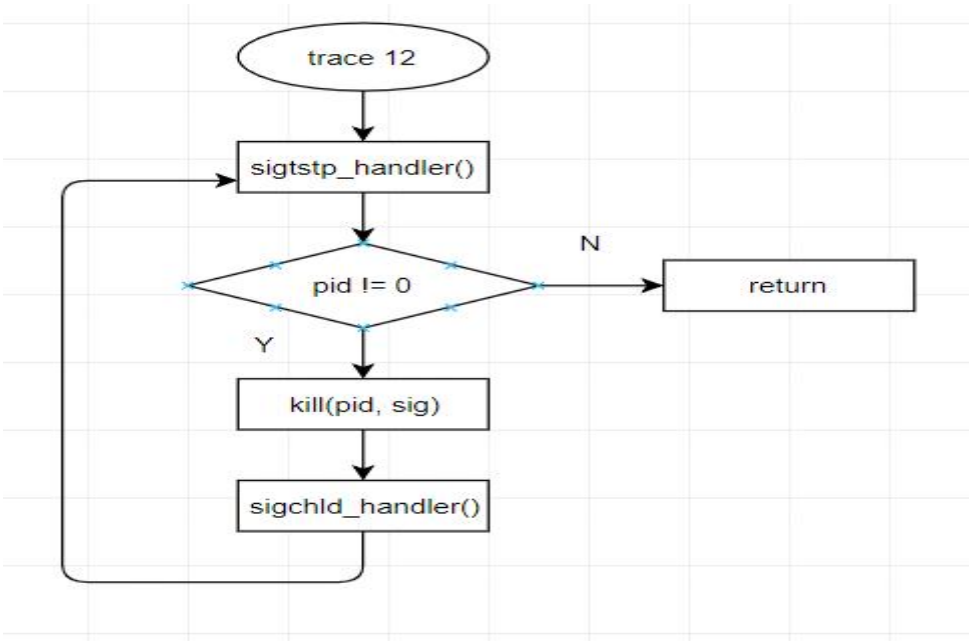
trace 11 해결 방법 설명

자식 프로세스 스스로에게 SIGNIT가 전송되고 처리되는 단계이다. 자신의 pid로 SIGINT를 전달하였을 때, 부모를 거치지 않고 시그널이 전송되어 SIGCHLD가 발생하면 sigchld_handler 함수가 호출되어 WIFSIGNALED(child_status)값이 활성화되므로 deletejob() 함수가 실행된다.

Trace 12

```
a201502023@2018-sp: ~/shlab-handout
a201502023@2018-sp:~/shlab-handout$ ./sdriver -t 12 -s ./tsh -V
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] (4696) stopped by signal 20
tsh> jobs
(1) (4696) Stopped    ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (4704) stopped by signal 20
tsh> jobs
(1) (4704) Stopped ./mytstps

a201502023@2018-sp:~/shlab-handout$
```

trace 12 플로우 차트



trace 12 해결 방법 설명

자식 프로세스 스스로에게 SIGTSTP가 전송되고 처리되는 단계이다. 자신의 pid로 SIGTSTP를 전달하였을 때, 부모를 거치지 않고 시그널이 전송되어 SIGCHLD가 발생하면 sigchld_handler 함수가 호출되어 WIFSTOPPED(child_status)값이 활성화되므로 job의 상태가 ST가 되고 deletejob()함수가 실행된다.