2018 시스템 프로그래밍
- Lab 08 -

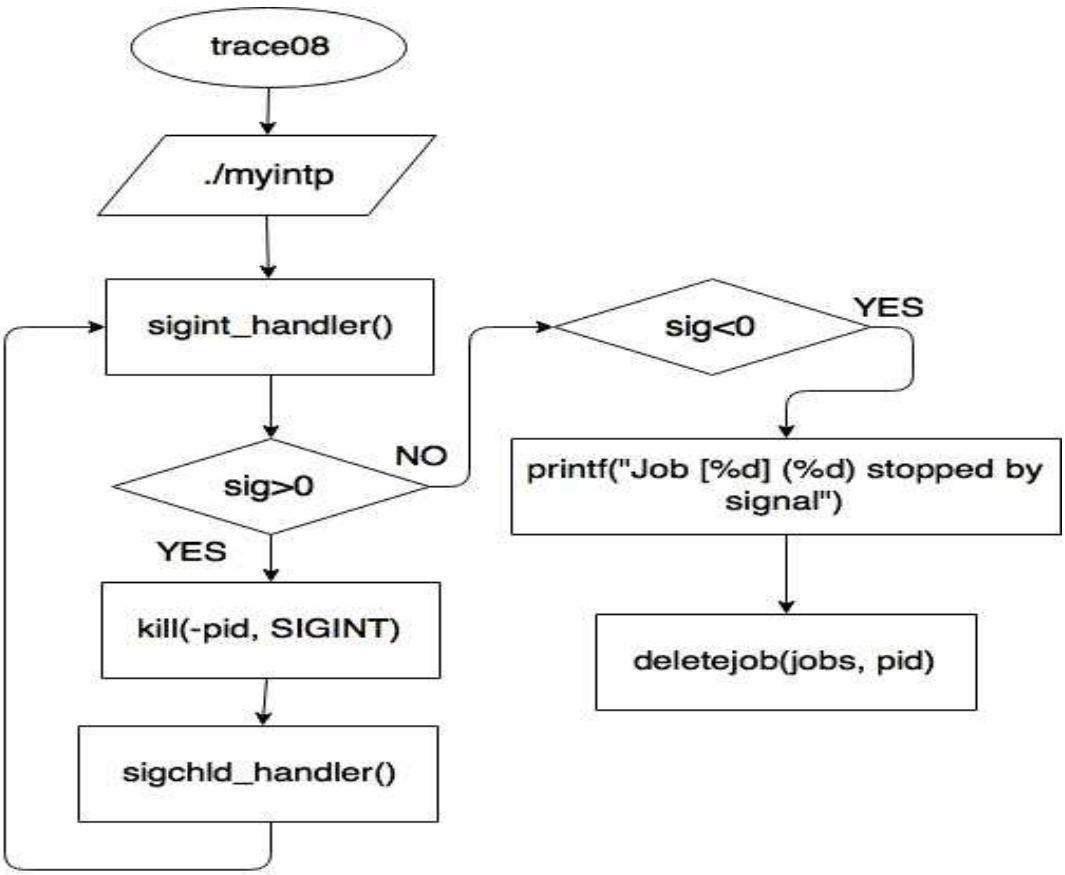
제출일자	2018.11.20
분 반	00
이 름	노효근
학 번	201502049

Trace 08

```
a201502049@2018-sp:~/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] (13436) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (13444) terminated by signal 2
tsh> quit
```

각 trace 별 플로우 차트



trace 해결 방법 설명

trace08은 SIGINT(ctrl +c) 발생 시 Foreground 작업 종료과정이다.
따라서 SIGINT 발생시, main 함수에 설치한 sigint_handler()를 호출한다.
따라서 sigint_handler()를 다음과 같이 구현하였다.

```
293 void sigint_handler(int sig)
294 {
295     pid_t pid = fgpid(jobs);
296
297     if(sig>0){
298         kill(-pid,SIGINT);
299     }
300     else if(sig<0){
301         printf("Job [%d] (%d) terminated by signal %d\n", pid2jid(pid), pid, -sig);
302         deletejob(jobs,pid);
303     }
304     return;
305 }
```

fgpid(jobs)를 통해 job list에 있는 foreground job의 pid를 저장, foreground job를 찾는다. sig >0 이때, kill 함수로 지정한 pid를 가지는 프로세스에 시그널 전달, SIGCHILD 시그널을 받아, sigchld_handler()를 호출한다. sig < 0인 경우 deletejob 함수를 통해 자식프로세스를 job list에서 제거한다.

```
266 void sigchld_handler(int sig)
267 {
268     int status;
269     pid_t pid;
270     struct job_t *t;
271
272     while((pid = waitpid(-1, &status, WNOHANG|WUNTRACED))>0){
273         if(WIFSTOPPED(status)){
274             sigtstp_handler(-20);
275         }
276         else if(WIFSIGNALED(status)){
277             sigint_handler(-2);
278         }
279         else if(WIFEXITED(status)){
280             deletejob(jobs,pid);
281         }
282         else
283             unix_error("waitpid error\n");
284     }
285     return;
286 }
```

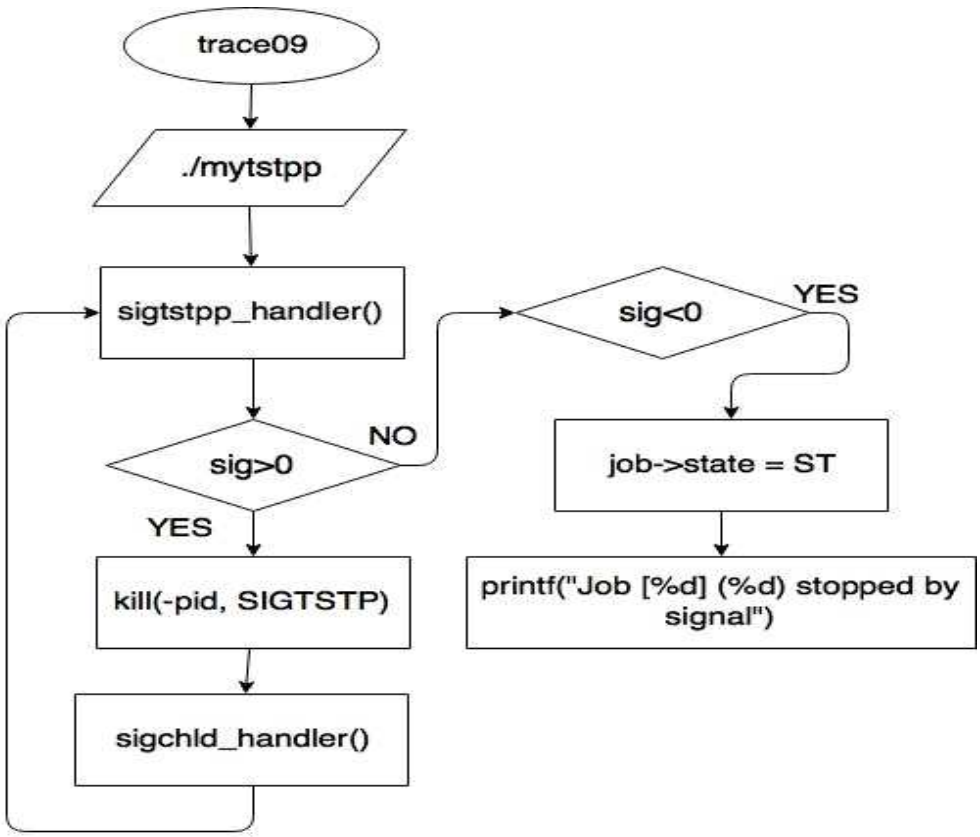
sig > 0 일 때, sigchld_handler()안 while 루프를 통해 waitpid()를 이용, 자식프로세스가 작업이 끝날 때까지 기다려준다. sigint_handler()에서 kill(-pid ,SIGINT)를 해주므로 WIFSIGNALED(status)값이 1이 되고, 이때, sigint_handler()의 시그널은 –2이가 되므로 sig < 0 부분이 실행되게 된다.

Trace 09

```
a201502049@2018-sp:~/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] (21098) stopped by signal 20
tsh> jobs
(1) (21098) Stopped    ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (21107) stopped by signal 20
tsh> jobs
(1) (21107) Stopped ./mytstpp
```

각 trace 별 플로우 차트



trace 해결 방법 설명

trace09은 SIGTSTP(ctrl +z) 발생 시 Foreground 작업을 정지하도록 한다.
따라서 SIGTSTP 발생시, main 함수에 설치한 sigtstp_handler()를 호출한다.
따라서 sigtstp_handler()를 다음과 같이 구현하였다.
SIGTSTP 발생 시, sig = 20이 되므로 sig >0 인 부분이 실행, kill함수를 통해 시그널을 커널과 부모 프로세스로 전달한다. 이후, sigtstp_handler(-20) 함수 호출, 이때, WIFSIGANALED(status) 값이 1이 되고 sigint_handler(-2)가 실행된다.
이후, sigtstp_handler(-20)를 통해 해당 job의 상태를 ST로 변경하고 sigint_handler(-2)를 통해, 자식프로세스를 종료 시킨다.

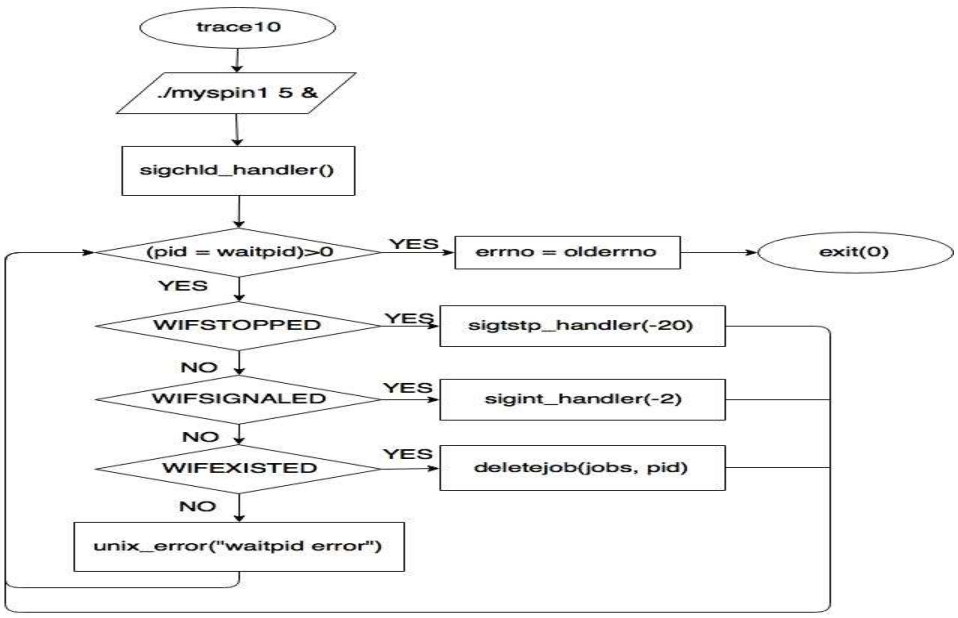
```
312 void sigtstp_handler(int sig)
313 {
314     pid_t pid = fgpid(jobs);
315     struct job_t *job = getjobpid(jobs,pid);
316
317     if(sig>0){
318         kill(-pid,SIGTSTP);
319     }
320     else if(sig<0){
321         job->state = ST;
322         printf("Job [%d] (%d) stopped by signal %d\n",pid2jid(pid), pid, -sig);
323     }
324     return;
325 }
```

Trace 10

```
a2015020490@2018-sp:~/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) (21853) ./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) (21863) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit
```

각 trace 별 플로우 차트



trace 해결 방법 설명

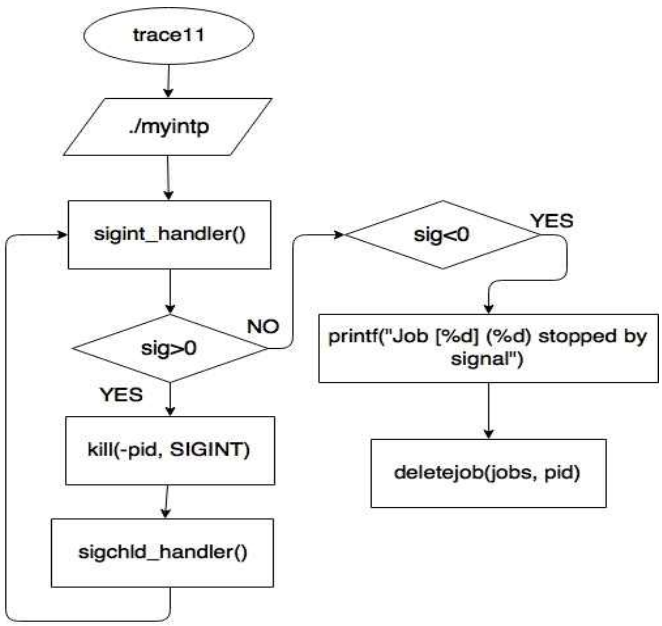
trace10은 Background 작업을 정상 종료 처리 구현이다.
따라서 정상종료 시, SIGCHLD 발생, main 함수에 설치한 sigchld_handler()를 호출한다.
WIFEXITED(status)는 정상적으로 자식프로세스가 종료 되었을 시, 0이 아닌 값을 리턴해야하므로, waitpid를 통해 자식프로세스가 종료될 때까지 기다리고, 자식프로세스 종료시, deletejob을 통해 자식을 제거한다.

Trace 11

```
a2015020049@2018-sp:~/shlab-handout$ ./sdriver -V -t 11 -s ./tsh
Running tracell.txt...
Success: The test and reference outputs for tracell.txt matched!
Test output:
#
# tracell.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (22036) terminated by signal 2
tsh> quit

Reference output:
#
# tracell.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (22044) terminated by signal 2
tsh> quit
```

각 trace 별 플로우 차트



trace 해결 방법 설명

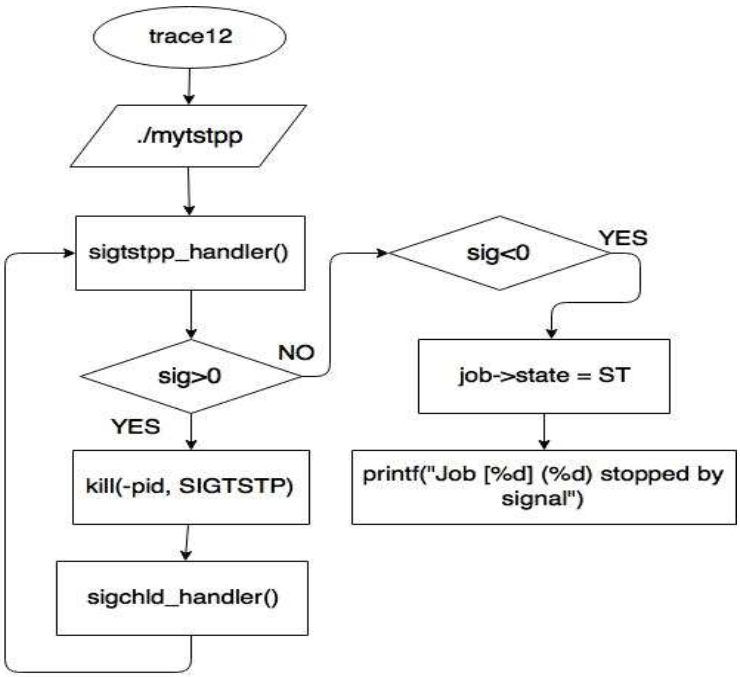
trace11은 자식 프로세스 스스로에게 SIGINT 전송되고 처리하는 단계이다.
스스로에게 SIGINT 시그널을 전달 시, 정상적으로 처리되도록 구현.
부모를 거치지 않고 시그널이 전송되므로 sigint_handler()의 sig 값에 2가 들어올 수 없으므로, SIGCHLD 발생 시, sigchld_handler() 함수 호출이 되고, 이때 WIFSIGNALED(status)값이 활성화 되어, sigint_handler()에 –2 값이 들어오므로 정상적으로 deletejob이 실행된다.

Trace 12

```
a2015020490@2018-sp:~/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] (22545) stopped by signal 20
tsh> jobs
(1) (22545) Stopped    ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (22553) stopped by signal 20
tsh> jobs
(1) (22553) Stopped ./mytstps
```

각 trace 별 플로우 차트



trace 해결 방법 설명

trace12는 자식 프로세스 스스로에게 SIGTSTP 전송되고 처리하는 단계이다.
스스로에게 SIGTSTP를 전달하였을 때, 정상적으로 SIGTSTP에 대한 처리가 되도록 구현.
부모를 거치지 않고 시그널이 전송되므로 sigint_handler()의 sig 값에 20가 들어올 수 없으므로, SIGCHLD 발생 시, sigchld_handler() 함수 호출이 되고, 이때 WIFEXITED(status)값이 활성화 되어, sigtstp_handler()에 –20 값이 들어오므로 정상적으로 job의 상태가 ST가 된다.