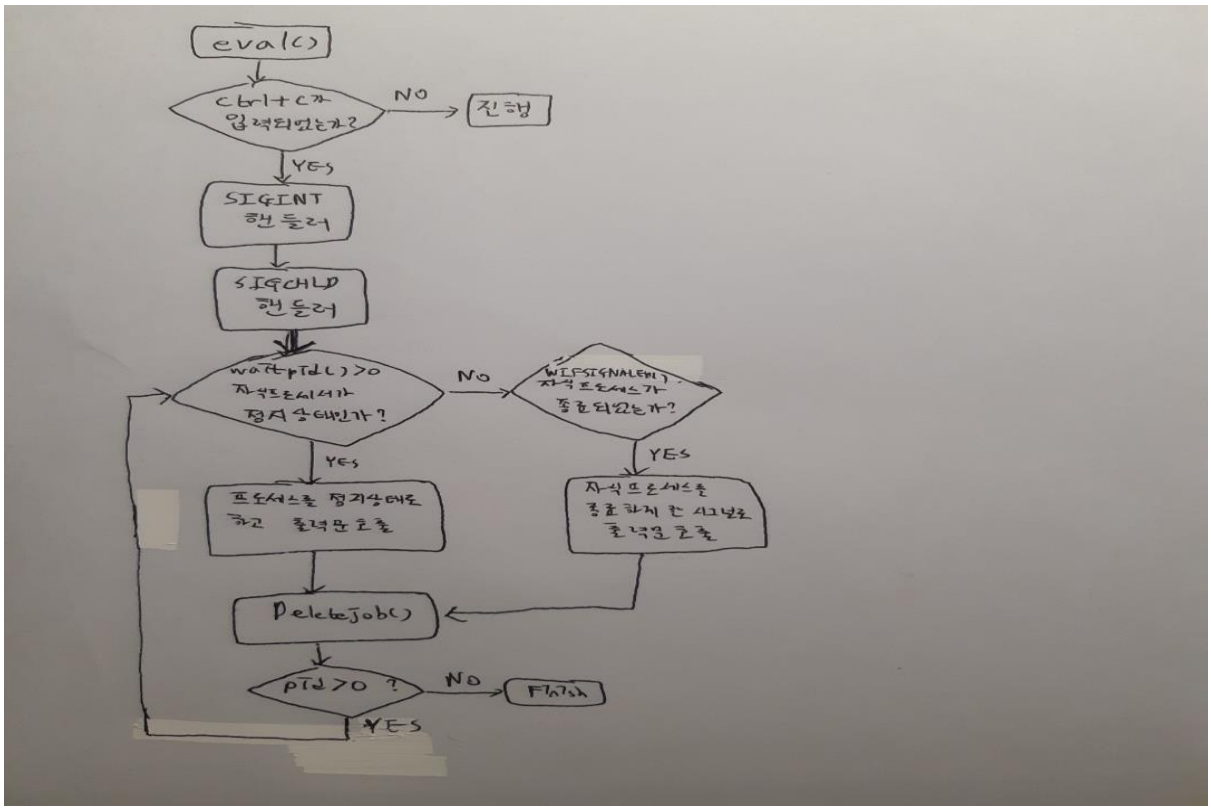
2018 시스템 프로그래밍

- Lab 08 -

제출일자	2018.11.22
분 반	02
이 름	진승언
학 번	201404377

실습 8 [화면 캡처]

```
c2014043770@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] (3463) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (3471) terminated by signal 2
tsh> quit

c2014043770@2018-sp:~/shlab-handout$
```



실습 8 [과정 설명]

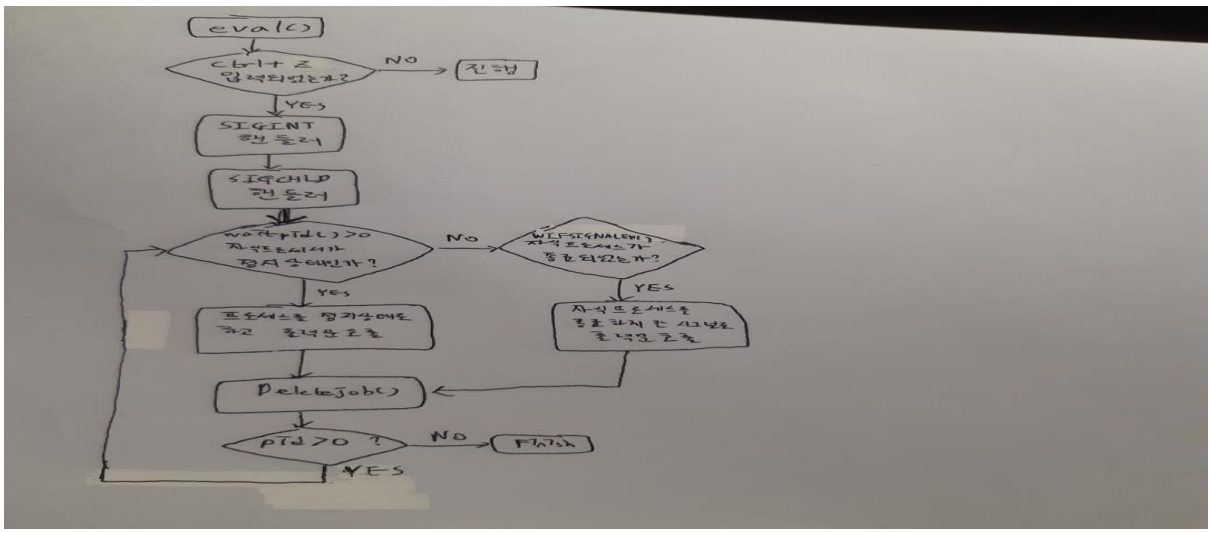
```
170 void eval(char *cmdline)
171 {
172     char *argv[MAXARGS];
173     pid_t pid;
174     int bg = parseline(cmdline, argv);
175
176     sigset_t mask;
177     sigemptyset(&mask);// 시그널 셋 초기화
178     sigaddset(&mask, SIGCHLD);//시그널 추가
179     sigaddset(&mask, SIGINT);//시그널 추가
180
181     if(!builtin_cmd(argv)){
182         sigprocmask(SIG_BLOCK, &mask, NULL);//시그널 블록
183         if((pid=fork())  == 0){
184             setpgid(0,0);
185             sigprocmask(SIG_UNBLOCK, &mask, NULL); //시그널 언블록
186             if(execve(argv[0], argv, environ)<0){
187                 printf("%s : Command not found\n", argv[0]);
188                 exit(0);
189             }
190         }
191             addjob(jobs, pid, (bg == 1?BG:FG), cmdline);
192             sigprocmask(SIG_UNBLOCK, &mask, NULL);  //시그널 언블록
193
194             if(!bg){
195                 waitfg(pid, 1);
196             }
197             else{
198                 printf("(%d) (%d) %s", pid2jid(pid), pid, cmdline);
199             }
200         }
201     return;
202 }
```

```
256 void sigchld_handler(int sig)
257 {
258
259     pid_t pid;
260     int child_status = 0;
261
262
263     while( (pid = waitpid(-1, &child_status, WNOHANG|WUNTRACED)) > 0){//자식프로세스 종료를 대기
264
265         if(WIFSIGNALED(child_status)){ //자식프로세스 종료를 체크
266             printf("Job [%d] (%d) terminated by signal %d\n", pid2jid(pid), pid, WTERMSIG(child_status));
267             deletejob(jobs, pid); //자식프로세스를 목록에서 제거
268         }
269         else{
270             deletejob(jobs, pid);
271         }
272     }
273     return;
274
275 }
276
277 /*
278  * sigint_handler - The kernel sends a SIGINT to the shell whenver the
279  *    user types ctrl-c at the keyboard.  Catch it and send it along
280  *    to the foreground job.
281  */
282 void sigint_handler(int sig)
283 {
284     pid_t pid;
285     pid = fgpid(jobs); //포어그라운드 프로세스의 pid저장
286      if(pid != 0){
287         kill(-pid, sig);//프로세스에 시그널 전달
288      }
289     return;
290
291 }
```

trace 08은 SIGINT 발생 시, Foreground 작업을 종료시키는 것을 구현하는거다. 여기서 SIGINT는 ctrl + c가 입력되었을 때인데 이 때 foreground작업를 kill 해주면 된다. SIGINT가 발생하면, main() 함수에 등록어 있는 sigint_handler() 함수가 자동으로 호출되는데 이 함수에서 forefround job를 종료시키도록 구현했다. waitpid함수로 자식프로세스가 종료 될 때 까지 위와 같이 반복을 한다. 이 때 WNOHANG | WUNTRACED 옵션을 통해서 대기 집합 모든 자식 프로세스들이 정지하였거나 종료했다면 0을 반환시키거나 자식들 중 하나의 PID와 동일한 값을 반환하도록 했다. 자식프로세스가 종료했다면 종료됐다는 출력문과 함께 자식프로세스를 목록에서 제거해준다. 만약 안했다면 그냥 deletejob만 해주면 됐다.

실습 9 [화면 캡처]

```
c201404377@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] (3493) stopped by signal 20
tsh> jobs
(1) (3493) Stopped    ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (3501) stopped by signal 20
tsh> jobs
(1) (3501) Stopped ./mytstpp

c201404377@2018-sp:~/shlab-handout$
```



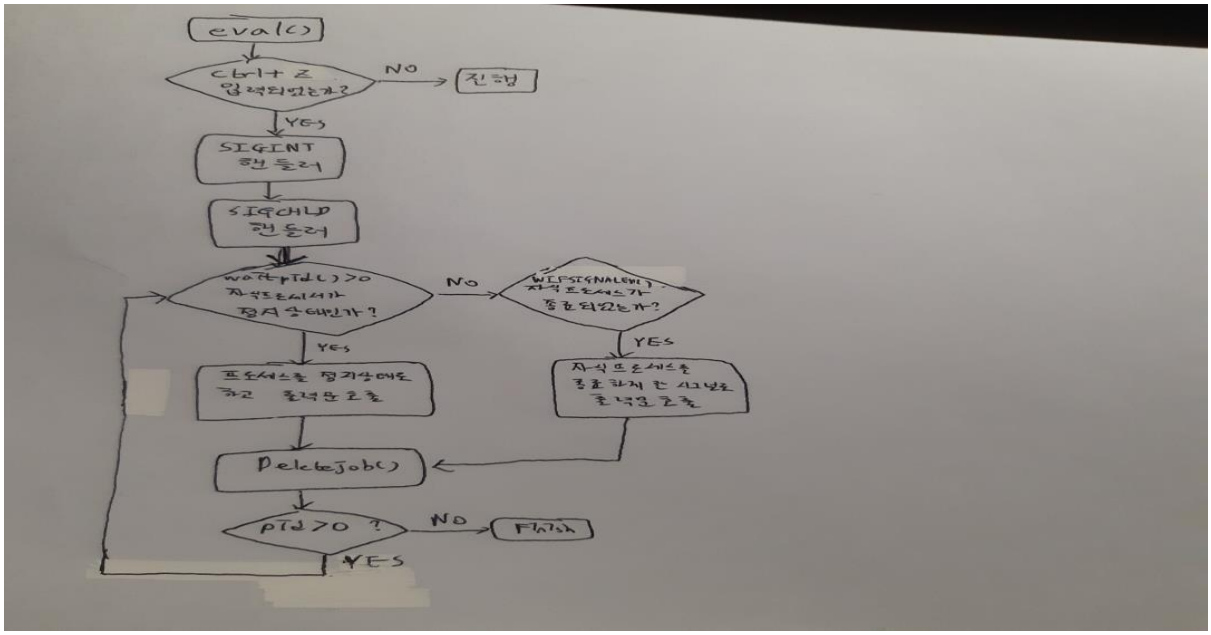
실습 9 [과정 설명]

```
293 /*
294  * sigtstp_handler - The kernel sends a SIGTSTP to the shell whenever
295  *     the user types ctrl-z at the keyboard. Catch it and suspend the
296  *     foreground job by sending it a SIGTSTP.
297  */
298 void sigtstp_handler(int sig)
299 {
300     pid_t pid;
301     pid = fgpid(jobs); //포어그라운드 프로세스의 pid저장
302     if(pid != 0){
303         kill(-pid, sig);//프로세스에 시그널 전달
304     }
305     return;
306
307 }
```

SIGSTP(키보드의 'ctrl+z'입력)이 되면 foreground 작업을 stop시키는 핸들러이다. trace08의 구현방법과 마찬가지로, sigstp 핸들러를 똑같이 구현해주면 되었다. 플로우 차트는 crtl+z 인것 말고는 동일하므로 똑같이 작성해주었다

실습 10 [화면 캡처]

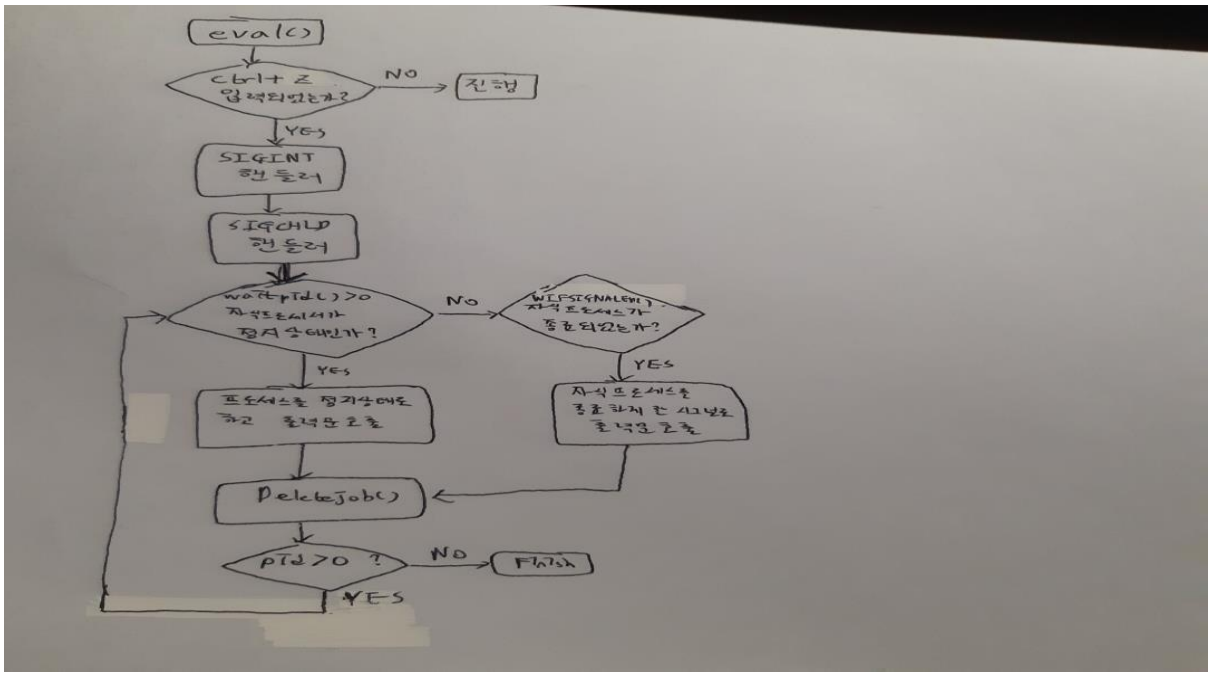
```
c201404377@2018-sp:~/shlab-handout$ ./sdriver -t 10 -s ./tsh -V
Running trace10.txt...
Success: The test and reference outputs for trace10.txt matched!
Test output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (2727) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

Reference output:
#
# trace10.txt - Send fatal SIGTERM (15) to a background job.
#
tsh> ./myspin1 5 &
(1) (2738) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

c201404377@2018-sp:~/shlab-handout$
```



실습 10 [과정 설명]

```
258 void sigchld_handler(int sig)
259 {
260
261     pid_t pid;
262     int child_status = 0;
263
264
265     while( (pid = waitpid(-1, &child_status, WNOHANG|WUNTRACED)) > 0){//자식프로세스 종료를 대기
266
267         if(WIFSIGNALED(child_status)){ //자식프로세스 종료를 체크
268             printf("Job [%d] (%d) terminated by signal %d\n", pid2jid(pid), pid, WTERMSIG(child_status));
269             deletejob(jobs, pid); //자식프로세스를 목록에서 제거
270         }
271         else if(WIFSTOPPED(child_status)){
272             printf("Job [%d] (%d) stopped by signal %d\n", pid2jid(pid),     pid, WSTOPSIG(child_status));
273             jobs[pid2jid(pid)-1].state =ST;
274         }
275         else{ //WIFEXITED(status)
276             deletejob(jobs, pid);
277         }
278     }
279     return;
280
```

```
170 void eval(char *cmdline)
171 {
172     char *argv[MAXARGS];
173     pid_t pid;
174     int bg = parseline(cmdline, argv);
175
176     sigset_t mask;
177     sigemptyset(&mask);// 시그널 셋 초기화
178     sigaddset(&mask, SIGCHLD);//시그널 추가
179     sigaddset(&mask, SIGTSTP); //trace 09
180     sigaddset(&mask, SIGINT);//시그널 추가
181
182
183     if(!builtin_cmd(argv)){
184         sigprocmask(SIG_BLOCK, &mask, NULL);//시그널 블록
185         if((pid=fork())  == 0){
186             setpgid(0,0);
187             sigprocmask(SIG_UNBLOCK, &mask, NULL); //시그널 언블록
188             if(execve(argv[0], argv, environ)<0){
189                 printf("%s : Command not found\n", argv[0]);
190                 exit(0);
191             }
192         }
193             addjob(jobs, pid, (bg == 1?BG:FG), cmdline);
194             sigprocmask(SIG_UNBLOCK, &mask, NULL);  //시그널 언블록
195
196             if(!bg){
197                 waitfg(pid, 1);
198             }
199             else{
200                 printf("(%d) (%d) %s", pid2jid(pid), pid, cmdline);
201             }
202         }
203     return;
204 }
```



trace10은 Background작업 정상 종료를 처리하는 것을 구현하는 것이었다. Background 작업이 정상 종료되면 SIGCHLD가 발생하는데, 이 시그널을 받고 Background 작업을 종료 처리해준다. 자식 프로세스가 정상 종료되고, joblist에 그것을 반영하기 위해서 WIFSTOPPED과 WIFSIGNALED 함수를 이용하여 job 정보를 업데이트 하였다. 또한 무슨 작업에 의해 종료되었느냐에 따라 출력문도 정답과 같은 출력문을 작성하였다.

실습 11~12 [화면 캡처]

```
c2014043770@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] (11970) terminated by signal 2
tsh> quit

Reference output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (11978) terminated by signal 2
tsh> quit
```

```
c2014043770@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] (12010) stopped by signal 20
tsh> jobs
(1) (12010) Stopped    ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (12018) stopped by signal 20
tsh> jobs
(1) (12018) Stopped ./mytstps

c2014043770@2018-sp:~/shlab-handout$
```

실습 11~12 [과정 설명]

trace 11, 12는 자식 프로세스 스스로에게 SIGINT와 SIGSTP를 전송되고 처리하는 것이다. 자식 프로세스가 커널을 통해 SIGINT, SIGSTP를 보내고 부모 프로세스는 SIGINT, SIGSTP 핸들러로 자식프로세스에 kill명령어를 수행한다. –pid로 pid그룹 내 전체에 시그널을 보내게 되므로 trace 11, 12모두 자동으로 통과 된다