

2016 시스템 프로그래밍
-Shell Lab 02-
- Lab 09 -

제출일자	2016.11.22
분 반	02
이 름	신종욱
학 번	2013024223

trace들을 분석하기 앞서 eval함수

```
tsh.c + (~/08/shlab-handout) - VIM
172 void eval(char *cmdline)
173 {
174     int bg;
175     char *argv[MAXARGS];
176     pid_t pid;
177     sigset_t mask;
178     bg= parseline(cmdline,argv);
179     if(!builtin_cmd(argv)){
180         sigemptyset(&mask); //mask 초기화
181         sigaddset(&mask,SIGCHLD); //SIGCHLD 시그널 추가
182         sigprocmask(SIG_BLOCK, &mask,NULL); //SIGCHLD 시그널을 블락한다
183         if ((pid=fork())==0){ //자식일 경우
184             setpgid(0,0); //자신의 프로세스 그룹을 자신의 프로세스 아이디로 바꾼다
185             sigprocmask(SIG_UNBLOCK, &mask, NULL); //SIGCHLD가 일어날 수 있도록 언블락한다
186             if(execec(argv[0],argv,enviro)<0){
187                 printf("%s: Command not found. \n",argv[0]);
188                 exit(0);
189             }
190         } else { //자식이 아닌 경우
191             if(!bg){ //프그라운드 작업일시
192                 if(addjob(jobs,pid,FG,cmdline)){ //잘 리스트 추가에 성공할 경우
193                     sigprocmask(SIG_UNBLOCK, &mask,NULL); //부모 프로세스에서 SIGCHLD가 일어날 수 있도록 언블락한다
194                     waitfg(pid, 0); //프그라운드 작업이 완료될 때까지 대기
195                 } else {kill(-pid,SIGINT); //실패할 경우 에러일으켜 종료
196             }
197         } else { //백그라운드 작업일시
198             if( addjob(jobs,pid,BG,cmdline)){ //잘 리스트 추가에 성공할 경우
199                 sigprocmask(SIG_UNBLOCK, &mask,NULL);
200                 int oid=pid2jid(pid);
201                 printf("(%d) (%d) %s",oid,pid,cmdline);
202             } else {kill(-pid,SIGINT); //실패했을 경우 에러일으켜 종료
203             }
204         }
205     }
206     return;
207 }
```

앞으로 다룰 trace들은 signal이 주된 역할은 하기 때문에 signal들을 관리해주어야 한다.
그에 맞춰 eval함수에서 명령어가 들어올 때 signal들을 초기화하고 다시 셋팅해주도록하였다.
설명은 주석으로 첨가하였다.

1.Trace08

1-1. sdriver -t 08 -s ./tsh -V 확인

```

c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/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] (3560) terminated by signal 2
tsh> quit

Reference output:
#
# trace08.txt - Send fatal SIGINT to foreground job.
#
tsh> ./myintp
Job [1] (3569) terminated by signal 2
tsh> quit

c201302423@localhost:~/08/shlab-handout$

```

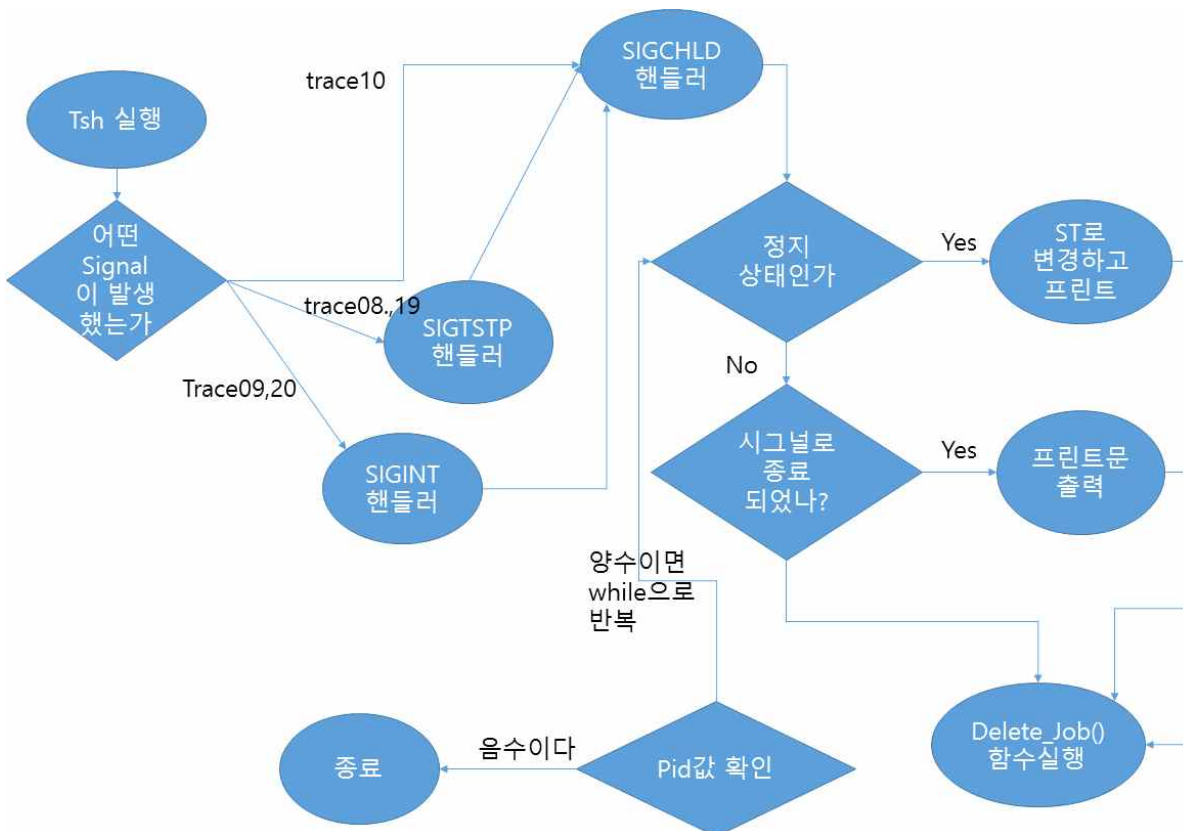
1-2 실제 tsh 실행 확인

```

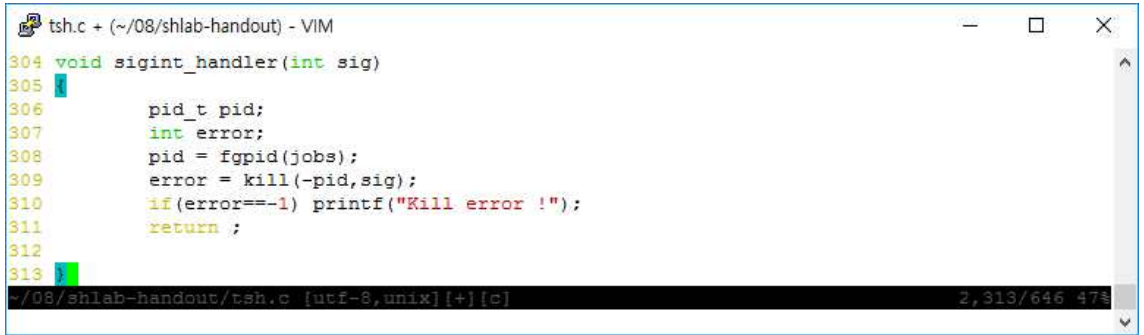
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myintp
Job [1] (4830) terminated by signal 2
eslab_tsh>

```

2.FlowChart

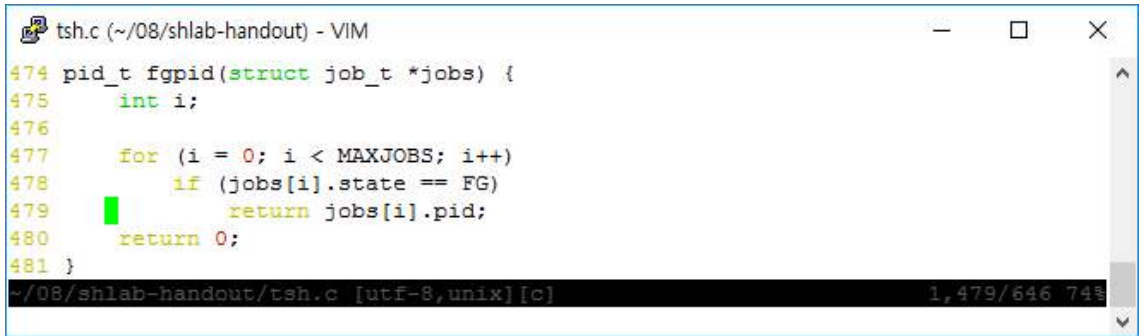


3. 해결방법



```
tsh.c + (~/.08/shlab-handout) - VIM
304 void sigint_handler(int sig)
305 {
306     pid_t pid;
307     int error;
308     pid = fgpid(jobs);
309     error = kill(-pid, sig);
310     if (error == -1) printf("Kill error !");
311     return ;
312 }
313
~/08/shlab-handout/tsh.c [utf-8,unix][+][c] 2,313/646 47%
```

ctrl+c가 입력들어왔을 경우 메인함수에서 Signal(SIGINT,sigint_handler)가 반응해서 sigint_handler가 실행되는데 foreground job을 찾아 종료시키도록 구현해야하는데



```
tsh.c (~/.08/shlab-handout) - VIM
474 pid_t fgpid(struct job_t *jobs) {
475     int i;
476
477     for (i = 0; i < MAXJOBS; i++)
478         if (jobs[i].state == FG)
479             return jobs[i].pid;
480     return 0;
481 }
~/08/shlab-handout/tsh.c [utf-8,unix][c] 1,479/646 74%
```

fgpid라는 현재 jobs 배열의 상태가 foreground인지 조사하고 해당 jobs의 pid를 리턴하는 함수가 있다. 해당 pid를 구한뒤 그 pid를 kill함수를 이용해서 종료 시키면된다.
-pid를 한이유는 뒤에있는 trace19가 그룹으로 삭제하는 구현이기 때문에 같이 구현하였다.
이후 sigchild가 실행되어야하는데 trace09에도 똑같음으로 뒤에 한번에 설명하겠습니다.

2.Trace09

1-1. sdriver -t 09 -s ./tsh -V 확인

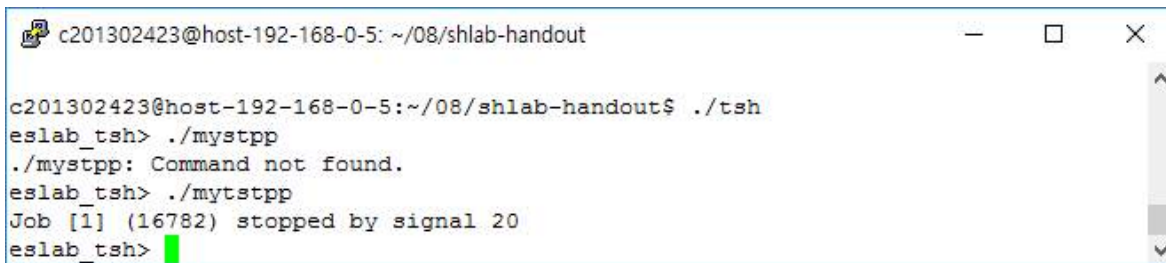


```
c201302423@localhost: ~/08/shlab-handout
Success: The test and reference outputs for trace09.txt matched!
Test output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (3608) stopped by signal 20
tsh> jobs
(1) (3608) Stopped      ./mytstpp

Reference output:
#
# trace09.txt - Send SIGTSTP to foreground job.
#
tsh> ./mytstpp
Job [1] (3616) stopped by signal 20
tsh> jobs
(1) (3616) Stopped ./mytstpp

c201302423@localhost:~/08/shlab-handout$
```

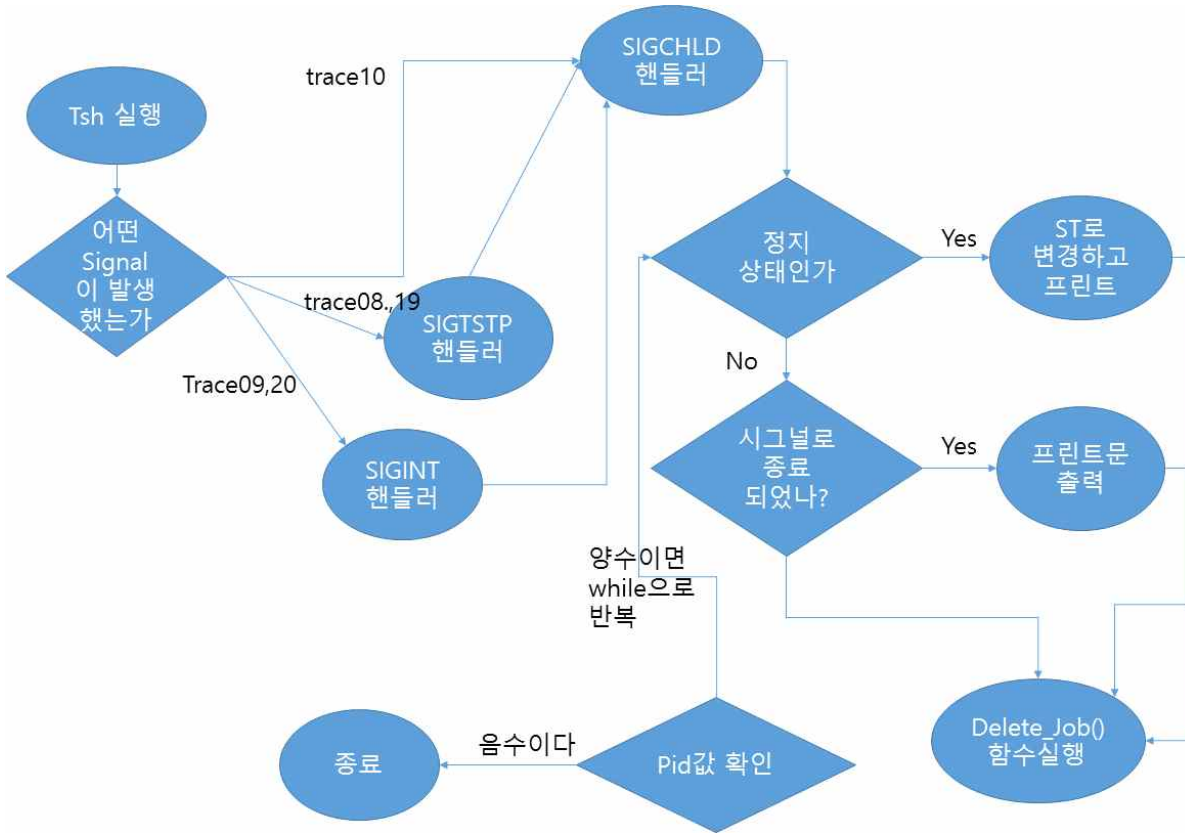
1-2 실제 tsh 실행 확인



```
c201302423@host-192-168-0-5: ~/08/shlab-handout

c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./mytstpp
./mytstpp: Command not found.
eslab_tsh> ./mytstpp
Job [1] (16782) stopped by signal 20
eslab_tsh>
```

2.FlowChart



3.해결방법

ctrl-z가 입력시

SIGTSTP 발생시 메인함수에서 Signal(SIGTSTP,sigtstp_handler)

Foreground 작업을 stop상태로 바꾸도록하여야 하는데

```

tsh.c + (~08/shlab-handout) - VIM
321 void sigtstp_handler(int sig)
322 {
323     pid_t pid;
324     int error;
325     pid = fgpuid(jobs);
326     error = kill(-pid,sig);
327     if( error==-1)
328         printf("Kill error !");
329     return;
330 }
~/08/shlab-handout/tsh.c [utf-8,unix][+][c] 2,330/646 50%
  
```

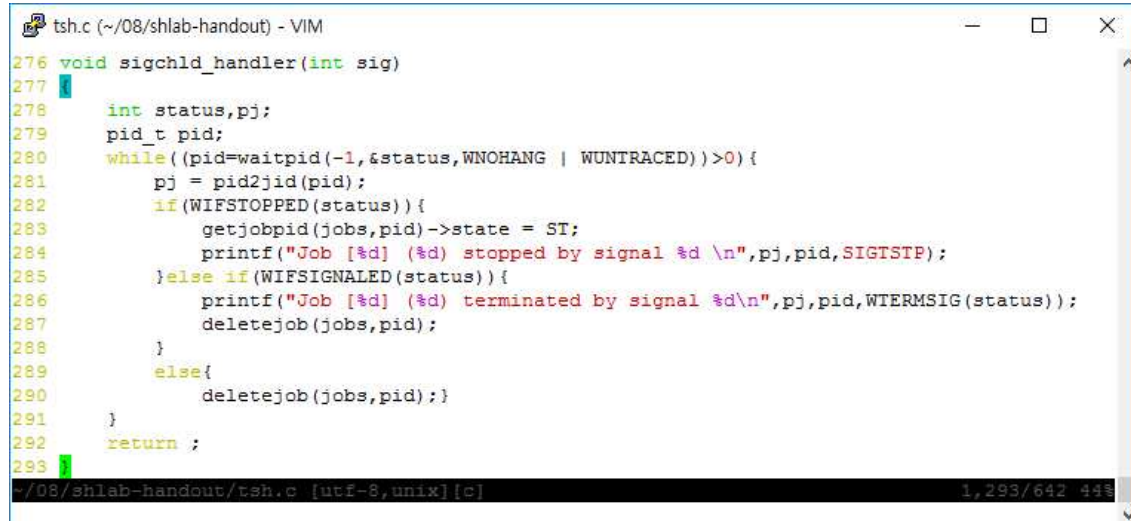
sigint랑 코드가 같은데 왜냐면 함수 실행후 다음 실행되는게

```

tsh.c (~08/shlab-handout) - VIM
122
123 /* These are the ones you will need to implement */
124 Signal(SIGINT, sigint_handler); /* ctrl-c */
125 Signal(SIGTSTP, sigtstp_handler); /* ctrl-z */
126 Signal(SIGCHLD, sigchld_handler); /* Terminated or stopped child */
127 Signal(SIGTTIN, SIG_IGN);
128 Signal(SIGTTOU, SIG_IGN);
129
130 /* This one provides a clean way to kill the shell */
131 Signal(SIGQUIT, sigquit_handler);
~/08/shlab-handout/tsh.c [utf-8,unix][c] 0,122/646 19%
  
```


자식프로세스가 종료되었거나 멈추었을 경우

Signal(SIGCHLD,sigchld_handler)가 실행되기 때문에 sigchld에서 프로세서 상태에따라 관리하면된다.



```
tsh.c (~/.08/shlab-handout) - VIM
276 void sigchld_handler(int sig)
277 {
278     int status,pj;
279     pid_t pid;
280     while((pid=waitpid(-1,&status,WNOHANG | WUNTRACED))>0){
281         pj = pid2jid(pid);
282         if(WIFSTOPPED(status)){
283             getjobpid(jobs,pid)->state = ST;
284             printf("Job [%d] (%d) stopped by signal %d \n",pj,pid,SIGTSTP);
285         }else if(WIFSIGNALED(status)){
286             printf("Job [%d] (%d) terminated by signal %d\n",pj,pid,WTERMSIG(status));
287             deletejob(jobs,pid);
288         }
289         else{
290             deletejob(jobs,pid);
291         }
292     }
293     return ;
294 }
```

wnohang|wuntraced 는 대기집합 모든자식 프로세스들이 정지하였거나 종료하였다면 리턴값 0으로 즉시 리턴되거나 자식들 중 한 개의 PID와 동일한 값으로 리턴한다.(교과서 발췌)

하는 옵션으로 기다리다가 얻은 자식 PID를 pid에 저장한다

pid2jid는 해당 pid의 jobs를 찾은뒤 jid를 리턴한다

wifstopped 자식 프로세스가 정지된 상태라면 true 리턴하는데

정지된상태면 getjobpid()로 pid를 이용해 해당 jobs[]의 인덱스주소를 찾아 그 jobs의 상태를 스탑상태로 바꾼다

그후 알맞은 프린트문을 출력한다

wifsignaled 자식프로세스가 어떤 signal에 의해 종료된 경우라면 true를 리턴하는데

여기서라면 sigint에 속한다고 볼 수 있다 jid와 pid 그리고

WTERMSIG()는 시그널의 번호를 반환하는 기능을 한다 sigint의 경우에는 항상 2이겠지만

trace10의 백그라운드 작업종료시에는 sigterm이라서 WTERMSIG()를 사용였다.

3.Trace 10

1-1 sdriver -t 10 -s ./tsh -V 확인

```

c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/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) (3640) ./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) (3652) ./myspin1 5 &
tsh> /bin/kill myspin1
kill: failed to parse argument: 'myspin1'
tsh> quit

c201302423@localhost:~/08/shlab-handout$

```

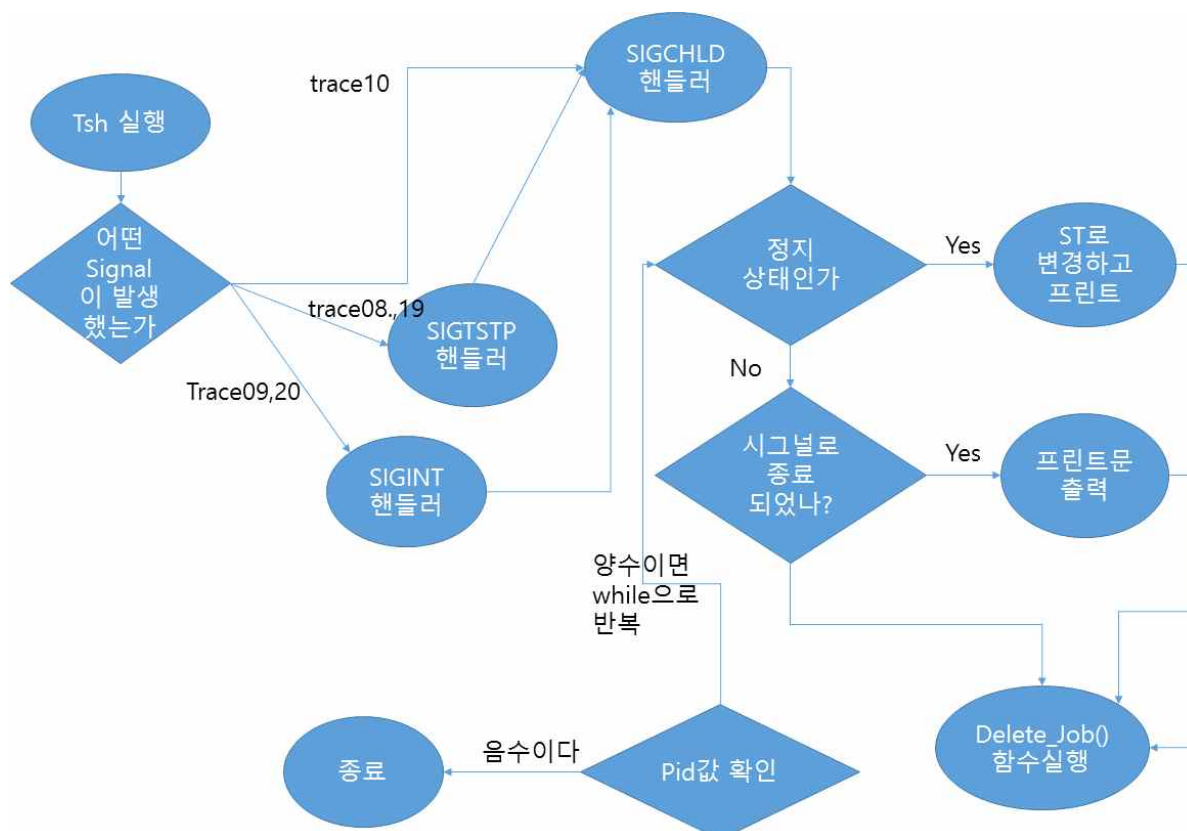
1-2 실제 tsh 실행 확인(myspin1.c의 코드가 이상해서 직접 실행해봤습니다.)

```

c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1 50 &
(1) (30859) ./myspin1 50 &
eslab_tsh> /bin/kill -15 30859
Job [1] (30859) terminated by signal 15
eslab_tsh>

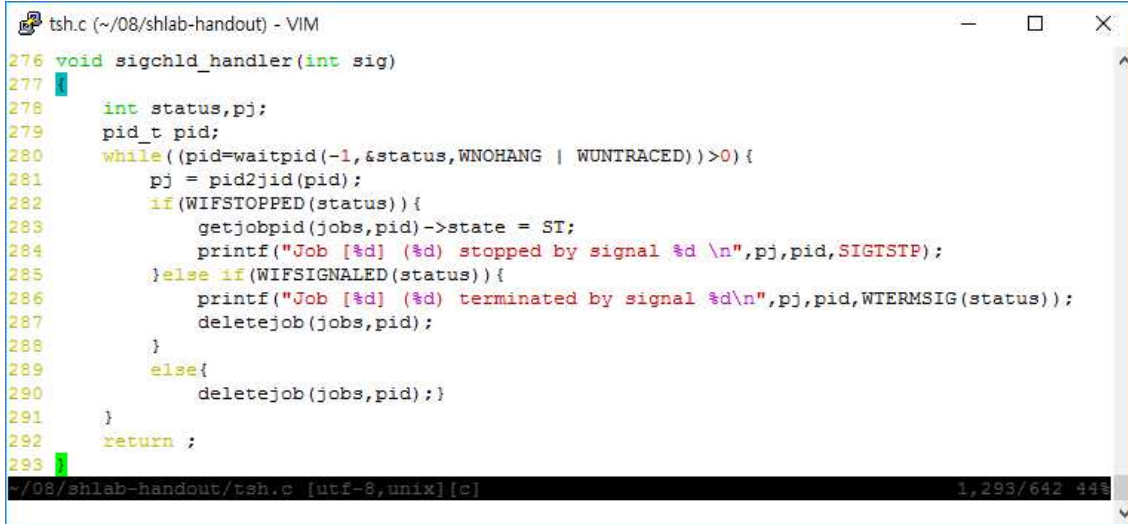
```

2.FlowChart



3.해결방법

Background작업이 정상 종료되면 SIGCHLD가 발생하는데 이 시그널을 받고 Background 작업을 종료해주도록 해야 하는데



```
tsh.c (~/.08/shlab-handout) - VIM
276 void sigchld_handler(int sig)
277 {
278     int status,pj;
279     pid_t pid;
280     while((pid=waitpid(-1,&status,WNOHANG | WUNTRACED))>0){
281         pj = pid2jid(pid);
282         if(WIFSTOPPED(status)){
283             getjobpid(jobs,pid)->state = ST;
284             printf("Job [%d] (%d) stopped by signal %d \n",pj,pid,SIGTSTP);
285         }else if(WIFSIGNALED(status)){
286             printf("Job [%d] (%d) terminated by signal %d\n",pj,pid,WTERMSIG(status));
287             deletejob(jobs,pid);
288         }
289         else{
290             deletejob(jobs,pid);}
291     }
292     return ;
293 }
```

앞서 설명에 있지만

wnohang|wuntraced 는 대기집합 모든자식 프로세스들이 정지하였거나 종료하였다면 리턴값 0으로 즉시 리턴되거나 자식들 중 한 개의 PID와 동일한 값으로 리턴한다. 임으로 자식이 다종료될때까지 기다리게된다 그리고 자식들을 종료 할 때 deletejob을해서 list에 반영하도록 하였다.

**myspin1에 void sigterm_handler(int signum)에 대한내용이 없어서 실행이 제대로 안되는거 같습니다.

4.Trace11

1-1. sdriver -t 11 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/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] (3853) terminated by signal 2
tsh> quit

Reference output:
#
# trace11.txt - Child sends SIGINT to itself
#
tsh> ./myints
Job [1] (3861) terminated by signal 2
tsh> quit

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myints
Job [1] (21366) terminated by signal 2
eslab_tsh>
```

2.해결방법

자식 프로세스 스스로에게 SIGINT를 전달하는 것인데 앞의 SIGINT구현을 그대로 냅두면된다.
에러 가능성이 있지만 childhandler함수에서 이미 자식들이 다 종료될때까지 부모는 기다리도록
wnohang|wuntraced를 이용하여 구현하였기에 변경없이 잘실행된다.

5.Trace12

1-1. sdriver -t 12 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
Success: The test and reference outputs for trace12.txt matched!
Test output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (3898) stopped by signal 20
tsh> jobs
(1) (3898) Stopped    ./mytstps

Reference output:
#
# trace12.txt - Child sends SIGTSTP to itself
#
tsh> ./mytstps
Job [1] (3906) stopped by signal 20
tsh> jobs
(1) (3906) Stopped    ./mytstps

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
eslab_tsh> ./mytstps
Job [1] (24583) stopped by signal 20
eslab_tsh> jobs
(1) (24583) Stopped    ./mytstps
eslab_tsh>
```

2.해결방법

자식 프로세스 스스로에게 SIGTSTP를 전달하는 것인데 앞의 SIGTSTP구현을 그대로 냅두면된다.
에러 가능성이 있지만 childhandler함수에서 이미 자식들이 다 종료혹은 정지할때까지 부모는 기다리도록 wnohang|wuntraced를 이용하여 구현하였기에 변경없이 잘실행된다.

6.Trace13

1-1. sdriver -t 13 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 13 -s ./tsh -V
Running trace13.txt...
Success: The test and reference outputs for trace13.txt matched!
Test output:
#
# trace13.txt - Forward SIGINT to foreground job only.
#
tsh> ./myspin1 5 &
(1) (3977) ./myspin1 5 &
tsh> ./myintp
Job [2] (3979) terminated by signal 2
tsh> jobs
(1) (3977) Running    ./myspin1 5 &

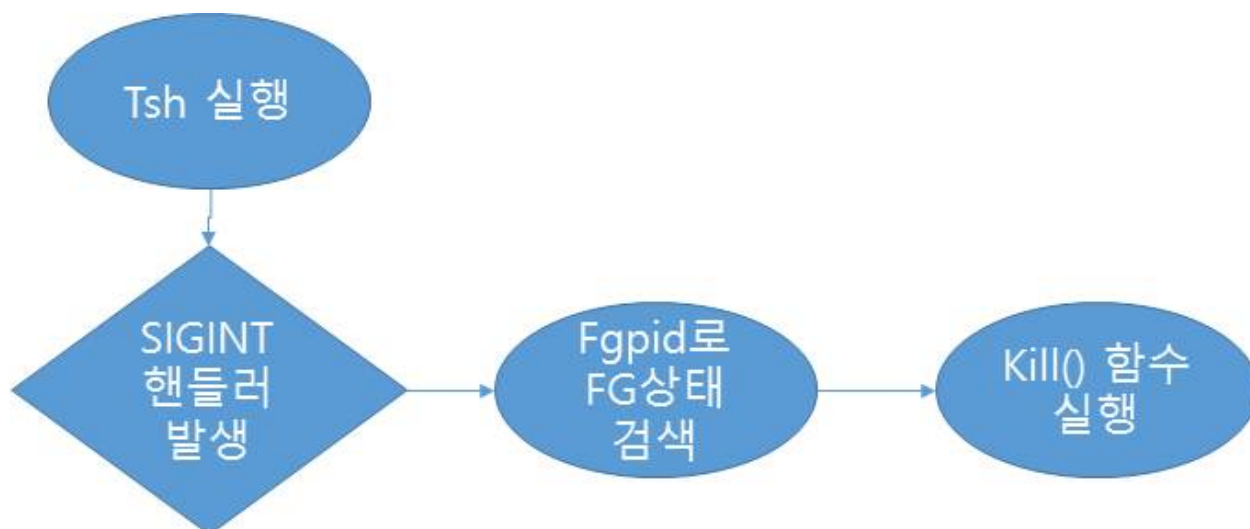
Reference output:
#
# trace13.txt - Forward SIGINT to foreground job only.
#
tsh> ./myspin1 5 &
(1) (3987) ./myspin1 5 &
tsh> ./myintp
Job [2] (3989) terminated by signal 2
tsh> jobs
(1) (3987) Running    ./myspin1 5 &

c201302423@localhost:~/08/shlab-handout$
```

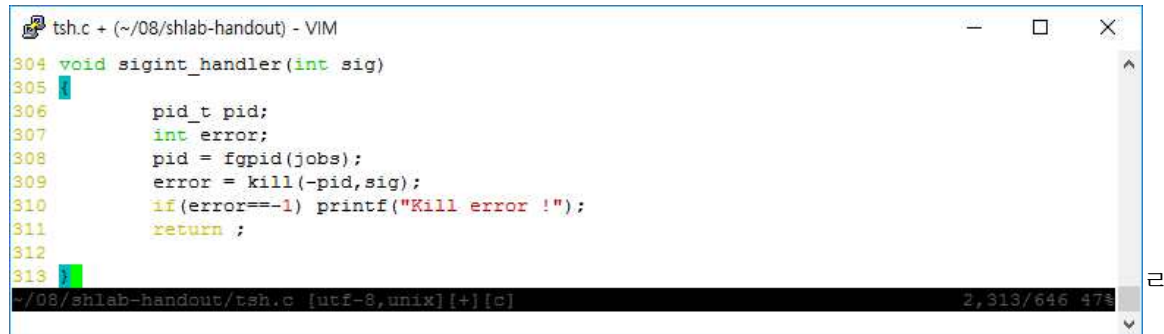
1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1 20 &
(1) (25961) ./myspin1 20 &
eslab_tsh> ./myintp
Job [2] (26005) terminated by signal 2
eslab_tsh> jobs
(1) (25961) Running    ./myspin1 20 &
eslab_tsh>
```

2.Flow Chart



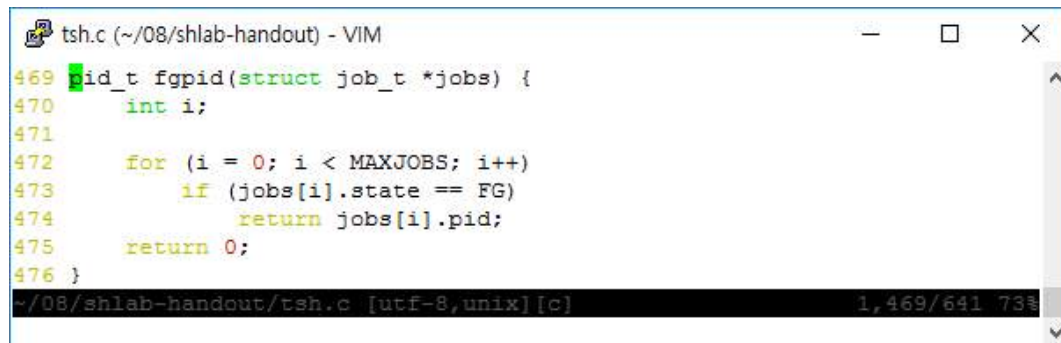
3. 해결방법



```
tsh.c + (~/.08/shlab-handout) - VIM
304 void sigint_handler(int sig)
305 {
306     pid_t pid;
307     int error;
308     pid = fgpid(jobs);
309     error = kill(-pid, sig);
310     if (error == -1) printf("Kill error !");
311     return ;
312 }
313
```

~/08/shlab-handout/tsh.c [utf-8,unix][+][c] 2,313/646 47%

오직 포그라운드에서만 시그널처리가 되도록하려면 FG인 프로세스를 찾아서 kill함수를 실행해주면된다



```
tsh.c (~/.08/shlab-handout) - VIM
469 pid_t fgpid(struct job_t *jobs) {
470     int i;
471
472     for (i = 0; i < MAXJOBS; i++)
473         if (jobs[i].state == FG)
474             return jobs[i].pid;
475     return 0;
476 }
```

~/08/shlab-handout/tsh.c [utf-8,unix][c] 1,469/641 73%

FG인 프로세스를 찾는 함수가 바로 fgpid라고 구현되어있다.
들어오 jobs배열에서 상태가 FG인걸 찾으면 jobs의 pid값을 리턴한다.

7.Trace14

1-1. sdriver -t 14 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 14 -s ./tsh -V
Running trace14.txt...
Success: The test and reference outputs for trace14.txt matched!
Test output:
#
# trace14.txt - Forward SIGTSTP to foreground job only.
#
tsh> ./myspin1 10 &
(1) (5457) ./myspin1 10 &
tsh> ./mytstpp
Job [2] (5468) stopped by signal 20
tsh> jobs
(1) (5457) Running      ./myspin1 10 &
(2) (5468) Stopped      ./mytstpp

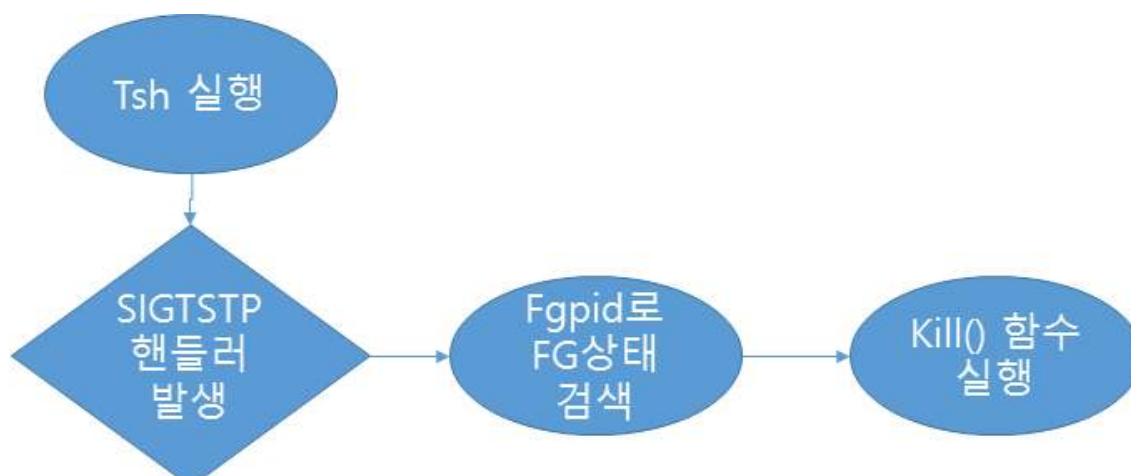
Reference output:
#
# trace14.txt - Forward SIGTSTP to foreground job only.
#
tsh> ./myspin1 10 &
(1) (5477) ./myspin1 10 &
tsh> ./mytstpp
Job [2] (5479) stopped by signal 20
tsh> jobs
(1) (5477) Running ./myspin1 10 &
(2) (5479) Stopped ./mytstpp

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1 20 &
(1) (28978) ./myspin1 20 &
eslab_tsh> ./mytstpp
Job [2] (29035) stopped by signal 20
eslab_tsh> jobs
(1) (28978) Running      ./myspin1 20 &
(2) (29035) Stopped      ./mytstpp
eslab_tsh>
```

2.Flow Chart



3. 해결방법



```
tsh.c + (~/.08/shlab-handout) - VIM
321 void sigtstp_handler(int sig)
322 {
323     pid_t pid;
324     int error;
325     pid = fgpid(jobs);
326     error = kill(-pid, sig);
327     if (error == -1)
328         printf("Kill error !");
329     return;
330 }
```

~/08/shlab-handout/tsh.c [utf-8,unix][+][c] 2,330/646 50%

trace 13과 동일하게 FG인 상태를 검색한후에 리턴되는 pid를 실행해주면된다.

8.Trace15

1-1. sdriver -t 15 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 15 -s ./tsh -V
Running trace15.txt...
Success: The test and reference outputs for trace15.txt matched!
Test output:
#
# trace15.txt - Process bg builtin command (one job)
#
tsh> ./mytstpp
Job [1] (5501) stopped by signal 20
tsh> bg %1
[1] (5501) ./mytstpp
tsh> jobs
(1) (5501) Running ./mytstpp

Reference output:
#
# trace15.txt - Process bg builtin command (one job)
#
tsh> ./mytstpp
Job [1] (5510) stopped by signal 20
tsh> bg %1
[1] (5510) ./mytstpp
tsh> jobs
(1) (5510) Running ./mytstpp

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./mytstpp
Job [1] (3249) stopped by signal 20
eslab_tsh> bg %1
[1] (3249) ./mytstpp
eslab_tsh> jobs
(1) (3249) Running ./mytstpp
eslab_tsh>
```

2.FlowChart



3.해결방법

```
tsh.c + (~/08/shlab-handout) - VIM
221     if(!strcmp(cmd, "bg")){
222         if(argv[1][0] == '%'){
223             if(argv[1][1] == '1'){
224                 jid = findST(jobs);
225                 getjobpid(jobs,jid)->state=BG;
226                 kill(getjobpid(jobs,jid)->pid,SIGCONT);
227             }
228             else if(argv[1][1]=='2'){
229                 jid = findST(jobs);
230                 getjobpid(jobs,jid)->state=BG;
231                 kill(getjobpid(jobs,jid)->pid,SIGCONT);
232             }
233         }
234         printf("[%d] (%d) ./mytstpp\n", pid2jid(jid), jid);
235         return 1;
236     }
```

빌트인 명령어를 구현하는 것으로 bg %1이 들어왔을 경우 정지상태인 프로세스를 검색하고 해당 프로세스를 background 형태로 재실행하여야하는데 정지상태인 프로세스를 검색하는 함수를 구현하였다.

```
tsh.c (~/08/shlab-handout) - VIM
477 pid_t findST(struct job_t *jobs){
478     int i;
479     for(i=0; i<MAXJOBS;i++)
480         if(jobs[i].state == ST)
481             return jobs[i].pid;
482     return 0;
483 }
```

findST로 함수로 jobs중에 stop상태를 검색하여서 pid를 리턴하여 jid로 저장한뒤
getjobpid함수는 jobs배열과 pid가 들어오면 해당 pid의 jobs배열의 인덱스의 주소를 리턴한다
해당 jobs인덱스의 상태를 BG로 바꿔준다.
그리고 해당 프로세스를 다시 실행시키기 위해 kill함수를 써서 해당 pid와 sigcont를 넣어 정지상태인 프로세스를 재실행한다

9.Trace16

1-1. sdriver -t 16 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 16 -s ./tsh -V
Running trace16.txt...
Success: The test and reference outputs for trace16.txt matched!
Test output:
#
# trace16.txt - Process bg builtin command (two jobs)
#
tsh> ./myspin1 10 &
(1) (5571) ./myspin1 10 &
tsh> ./mytstpp
Job [2] (5575) stopped by signal 20
tsh> jobs
(1) (5571) Running      ./myspin1 10 &
(2) (5575) Stopped      ./mytstpp
tsh> bg %2
[2] (5575) ./mytstpp
tsh> jobs
(1) (5571) Running      ./myspin1 10 &
(2) (5575) Running      ./mytstpp

Reference output:
#
# trace16.txt - Process bg builtin command (two jobs)
#
tsh> ./myspin1 10 &
(1) (5585) ./myspin1 10 &
tsh> ./mytstpp
Job [2] (5587) stopped by signal 20
tsh> jobs
(1) (5585) Running ./myspin1 10 &
(2) (5587) Stopped ./mytstpp
tsh> bg %2
[2] (5587) ./mytstpp
tsh> jobs
(1) (5585) Running ./myspin1 10 &
(2) (5587) Running ./mytstpp
c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1 20 &
(1) (7529) ./myspin1 20 &
eslab_tsh> ./mytstpp
Job [2] (7530) stopped by signal 20
eslab_tsh> jobs
(1) (7529) Running      ./myspin1 20 &
(2) (7530) Stopped      ./mytstpp
eslab_tsh> bg %2
[2] (7530) ./mytstpp
eslab_tsh> jobs
(1) (7529) Running      ./myspin1 20 &
(2) (7530) Running      ./mytstpp
eslab_tsh>
```

2. FlowChart



3. 해결방법

```
tsh.c + (~/.shlab-handout) - VIM
221     if(!strcmp(cmd, "bg")){
222         if(argv[1][0] == '%'){
223             if(argv[1][1] == '1'){
224                 jid = findST(jobs);
225                 getjobpid(jobs, jid) -> state = BG;
226                 kill(getjobpid(jobs, jid) -> pid, SIGCONT);
227             }
228             else if(argv[1][1] == '2'){
229                 jid = findST(jobs);
230                 getjobpid(jobs, jid) -> state = BG;
231                 kill(getjobpid(jobs, jid) -> pid, SIGCONT);
232             }
233         }
234         printf("[%d] (%d) ./mytstpp\n", pid2jid(jid), jid);
235         return 1;
236     }
```

앞서 구현한 함수와 동일하게 구현 하는데 bg %2가 추가로 인식하도록 하면 된다

10.Trace17

1-1. sdriver -t 17 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 17 -s ./tsh -V
Running trace17.txt...
Success: The test and reference outputs for trace17.txt matched!
Test output:
#
# trace17.txt - Process fg builtin command (one job)
#
tsh> ./mytstps
Job [1] (5668) stopped by signal 20
tsh> jobs
(1) (5668) Stopped      ./mytstps
tsh> fg %1

Reference output:
#
# trace17.txt - Process fg builtin command (one job)
#
tsh> ./mytstps
Job [1] (5677) stopped by signal 20
tsh> jobs
(1) (5677) Stopped ./mytstps
tsh> fg %1

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./mytstps
Job [1] (7717) stopped by signal 20
eslab_tsh> jobs
(1) (7717) Stopped      ./mytstps
eslab_tsh> fg %1
eslab_tsh> jobs
eslab_tsh>
```

2.FlowChart

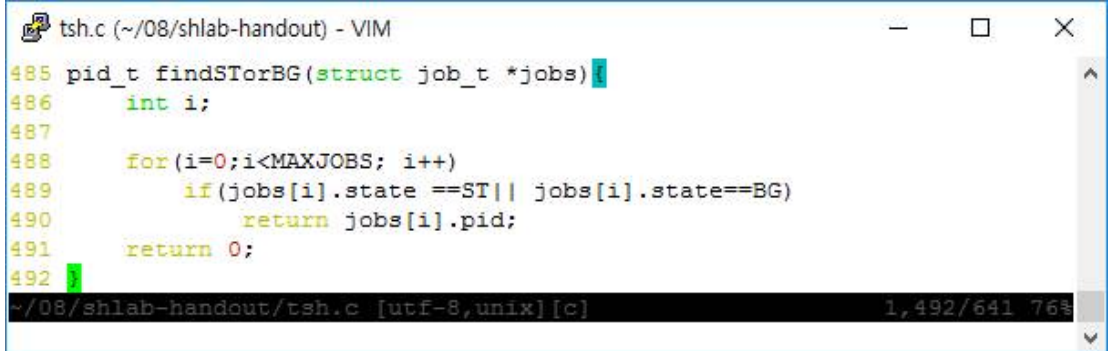


3.해결방법



```
tsh.c + (~/.shlab-handout) - VIM
236     }
237     if(!strcmp(cmd, "fg")){
238         if(argv[1][0] == '%'){
239             if(argv[1][1] == '1'){
240                 jid = findSTorBG(jobs);
241                 getjobpid(jobs, jid)->state = FG;
242                 kill( getjobpid(jobs, jid)->pid, SIGCONT);}
243             else if(argv[1][1] == '2'){
244                 jid = findSTorBG(jobs);
245                 getjobpid(jobs, jid)->state = FG;
246                 kill( getjobpid(jobs, jid)->pid, SIGCONT);}
247             }
248         return 1;}
249     return 0;
250 }
251
```

빌트인 명령어를 구현하는 것으로 fg %1이 들어왔을 경우 정지상태거나 Background형태인 프로세스를 검색하고 해당 프로세스를 Foreground 형태로 재실행하여야하는데



```
tsh.c (~/.shlab-handout) - VIM
485 pid_t findSTorBG(struct job_t *jobs){
486     int i;
487
488     for(i=0; i<MAXJOBS; i++)
489         if(jobs[i].state == ST || jobs[i].state == BG)
490             return jobs[i].pid;
491     return 0;
492 }
```

findSTorBG로 함수로 jobs중에 stop상태이거나Background실행중인 프로세서를 검색하여서 pid를 리턴하여 jid로 저장한뒤 getjobpid함수를 이용하여 해당 pid의 jobs배열의 인덱스의 주소를 구한뒤에 해당 jobs인덱스의 상태를 FG로 바꿔준다.
그리고 해당 프로세서를 다시 실행시키기 위해 kill함수를 써서 해당 pid와 sigcont를 넣어 정지상태인 프로세스를 재실행한다.

11.Trace18

1-1. sdriver -t 18 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 18 -s ./tsh -V
Running trace18.txt...
Success: The test and reference outputs for trace18.txt matched!
Test output:
#
# trace18.txt - Process fg builtin command (two jobs)
#
tsh> ./myspin1 10 &
(1) (5731) ./myspin1 10 &
tsh> ./mytstps
Job [2] (5733) stopped by signal 20
tsh> jobs
(1) (5731) Running      ./myspin1 10 &
(2) (5733) Stopped      ./mytstps
tsh> fg %2

Reference output:
#
# trace18.txt - Process fg builtin command (two jobs)
#
tsh> ./myspin1 10 &
(1) (5743) ./myspin1 10 &
tsh> ./mytstps
Job [2] (5745) stopped by signal 20
tsh> jobs
(1) (5743) Running      ./myspin1 10 &
(2) (5745) Stopped      ./mytstps
tsh> fg %2

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> myspin1 20 &
(1) (8175) myspin1 20 &
eslab_tsh> ./mytstps
Job [2] (8207) stopped by signal 20
eslab_tsh> jobs
(1) (8175) Running      myspin1 20 &
(2) (8207) Stopped      ./mytstps
eslab_tsh> fg %2
eslab_tsh> jobs
(1) (8175) Foreground myspin1 20 &
(2) (8207) Stopped      ./mytstps
eslab_tsh>
```

2.FlowChart



3. 해결방법



```
tsh.c + (~/.08/shlab-handout) - VIM
236     }
237     if(!strcmp(cmd, "fg")){
238         if(argv[1][0] == '%'){
239             if(argv[1][1] == '1'){
240                 jid = findSTorBG(jobs);
241                 getjobpid(jobs, jid)->state = FG;
242                 kill( getjobpid(jobs, jid)->pid, SIGCONT);}
243             else if(argv[1][1] == '2'){
244                 jid = findSTorBG(jobs);
245                 getjobpid(jobs, jid)->state = FG;
246                 kill( getjobpid(jobs, jid)->pid, SIGCONT);}
247             }
248         return 1;}
249     return 0;
250 }
251
```

앞선 구현과 같이 작성하되 fg %2를 인식할수 있도록하여야한다.

12.Trace19

1-1. sdriver -t 19 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 19 -s ./tsh -V
Running trace19.txt...
Success: The test and reference outputs for trace19.txt matched!
Test output:
#
# trace19.txt - Forward SIGINT to every process in foreground process group
#
tsh> ./mysplit 10
Job [1] (5768) terminated by signal 2
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplit'

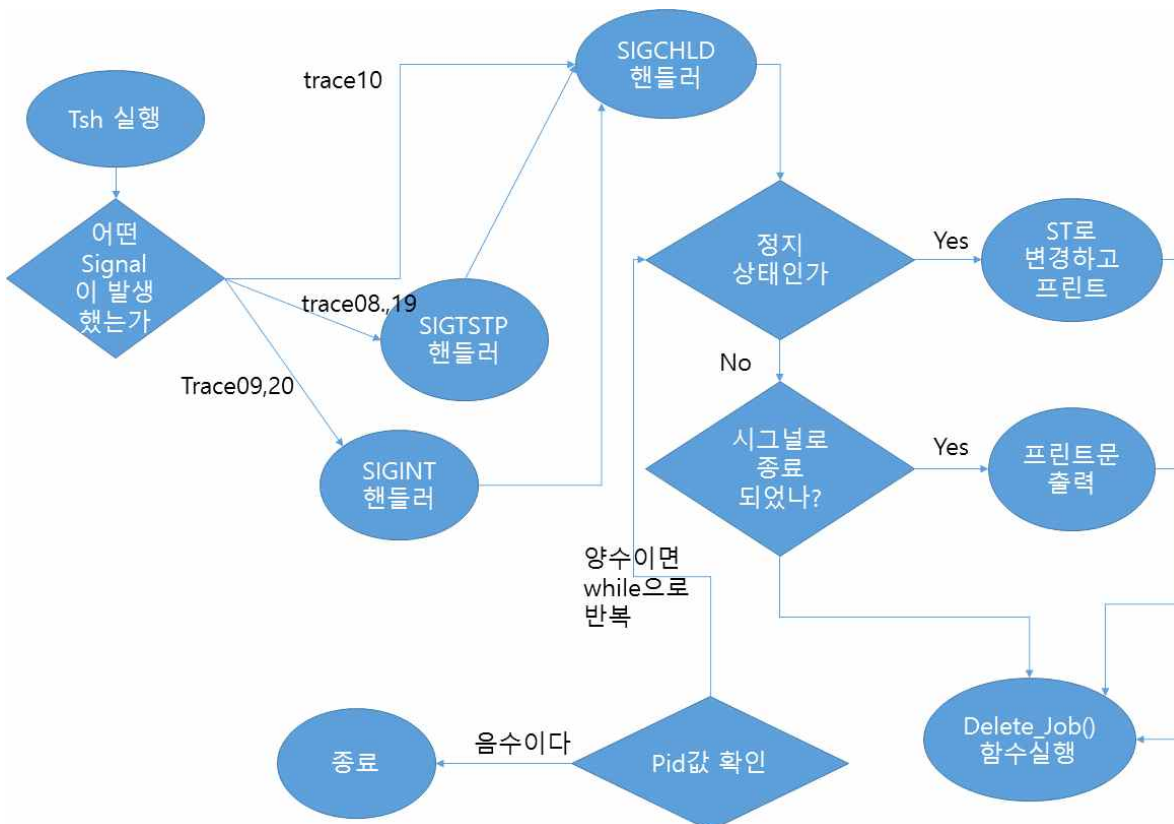
Reference output:
#
# trace19.txt - Forward SIGINT to every process in foreground process group
#
tsh> ./mysplit 10
Job [1] (5782) terminated by signal 2
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplit'

c201302423@localhost:~/08/shlab-handout$
```

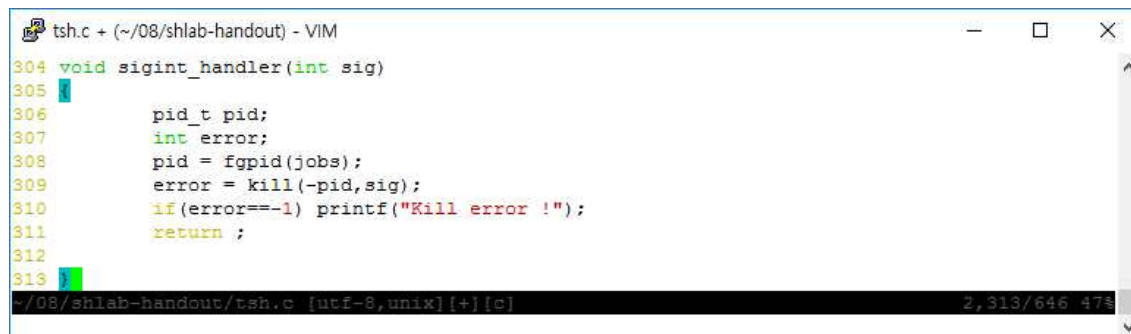
1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./mysplit 10
eslab_tsh> ./mysplit 10
^CJob [1] (11447) terminated by signal 2
eslab_tsh>
```

2.FlowChart



3. 해결방법



```
tsh.c + (~/08/shlab-handout) - VIM
304 void sigint_handler(int sig)
305 {
306     pid_t pid;
307     int error;
308     pid = fgpuid(jobs);
309     error = kill(-pid, sig);
310     if (error == -1) printf("Kill error !");
311     return ;
312 }
313
```

~/08/shlab-handout/tsh.c [utf-8,unix][+][c] 2,313/646 47%

앞서 trace08를 구현할 때 미리 -pid를 하여서 그룹프로세서를 sigint하도록하였다
여기서 kill에서 첫 번째 인자가 양수가 들어올 경우 해당 pid에 signal을 실행시키고
음수일 경우 해당 pid의 그룹내에 signal을 실행시킨다.

13.Trace20

1-1. sdriver -t 20 -s ./tsh -V 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 20 -s ./tsh -V
Running trace20.txt...
Success: The test and reference outputs for trace20.txt matched!
Test output:
#
# trace20.txt - Forward SIGTSTP to every process in foreground process group
#
tsh> ./mysplit 10
Job [1] (5878) stopped by signal 20
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplit | /usr/bin
/expand | /usr/bin/colrm 1 15 | /usr/bin/colrm 2 11'
T ./mysplit 10
T ./mysplit 10

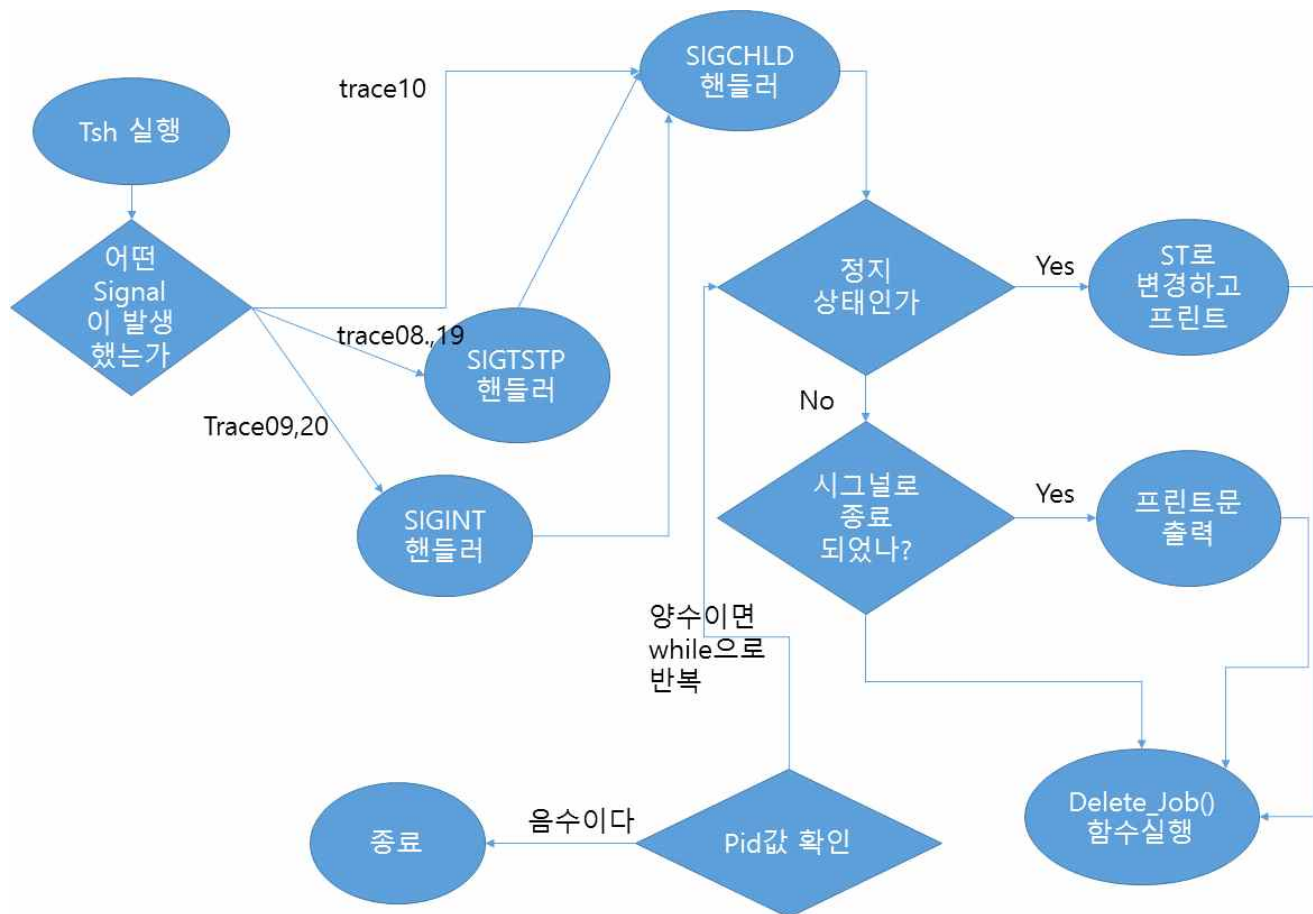
Reference output:
#
# trace20.txt - Forward SIGTSTP to every process in foreground process group
#
tsh> ./mysplit 10
Job [1] (5894) stopped by signal 20
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplit | /usr/bin
/expand | /usr/bin/colrm 1 15 | /usr/bin/colrm 2 11'
T ./mysplit 10
T ./mysplit 10

c201302423@localhost:~/08/shlab-handout$
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./mysplit 10
^ZJob [1] (11847) stopped by signal 20
eslab_tsh>
```


2.FlowChart



3.해결방법

```

tsh.c + (~/08/shlab-handout) - VIM
321 void sigtstp_handler(int sig)
322 {
323     pid_t pid;
324     int error;
325     pid = fgpid(jobs);
326     error = kill(-pid, sig);
327     if( error==-1)
328         printf("Kill error !");
329     return;
330 }
~/08/shlab-handout/tsh.c [utf-8,unix][+][c] 2,330/646 50%
  
```

앞서 trace09를 구현할 때 미리 -pid를 하여서 그룹프로세서를 sigtstp하도록하였다
여기서 kill에서 첫 번째 인자가 양수가 들어올 경우 해당 pid에 signal을 실행시키고
음수일 경우 해당 pid의 그룹내에 signal을 실행시킨다.

14.Trace21

1-1. sdriver -t 21 -s ./tsh -V 확인

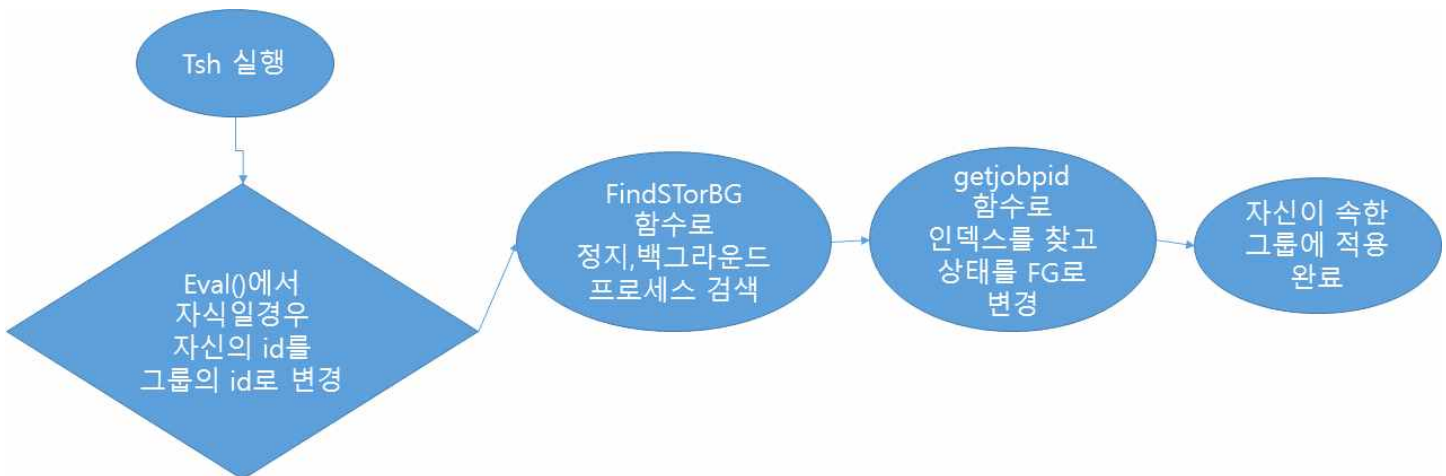
```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -t 21 -s ./tsh -V
Running trace21.txt...
Success: The test and reference outputs for trace21.txt matched!
Test output:
#
# trace21.txt - Restart every stopped process in process group
#
tsh> ./mysplitp
Job [1] (5973) stopped by signal 20
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplitp | /usr/bin/expand | /usr/bin/colrm 1 15 | /usr/bin/colrm 2 11'
T ./mysplitp
T ./mysplitp
tsh> fg %1
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplitp'

Reference output:
#
# trace21.txt - Restart every stopped process in process group
#
tsh> ./mysplitp
Job [1] (6001) stopped by signal 20
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplitp | /usr/bin/expand | /usr/bin/colrm 1 15 | /usr/bin/colrm 2 11'
T ./mysplitp
T ./mysplitp
tsh> fg %1
tsh> /bin/sh -c '/bin/ps ha | /bin/fgrep -v grep | /bin/fgrep mysplitp'
```

1-2 실제 tsh 실행 확인

```
c201302423@host-192-168-0-5: ~/08/shlab-handout
c201302423@host-192-168-0-5:~/08/shlab-handout$ ./tsh
eslab_tsh> ./mysplitp
Job [1] (13526) stopped by signal 20
eslab_tsh> jobs
(1) (13526) Stopped      ./mysplitp
eslab_tsh> fg %1
eslab_tsh> jobs
eslab_tsh>
```

2.FlowChart



3.해결방법

```
tsh.c + (~/.08/shlab-handout) - VIM
236     }
237     if(!strcmp(cmd,"fg")){
238         if(argv[1][0] == '%'){
239             if(argv[1][1] == '1'){
240                 jid = findSTorBG(jobs);
241                 getjobpid(jobs,jid)->state = FG;
242                 kill( getjobpid(jobs,jid)->pid,SIGCONT);}
243             else if(argv[1][1] == '2'){
244                 jid = findSTorBG(jobs);
245                 getjobpid(jobs,jid)->state = FG;
246                 kill( getjobpid(jobs,jid)->pid, SIGCONT);}
247             }
248         return 1;}
249     return 0;
250 }
251
```

앞서 구현한 fg와 bg에 명령어 마지막에
kill(getjobpid(jobs,jid)->pid,SIGCONT);라는 kill함수가있는데
구한 프로세서의 pid SIGCONT 시그널을 보내 정지된 프로세서들을 다시 실행하라는 Kill함수이다.

```
tsh.c (~/.08/shlab-handout) - VIM
179     sigemptyset(&mask);; //mask 초기화
180     sigaddset(&mask,SIGCHLD);; //SIGCHLD 시그널 추가
181     sigprocmask(SIG_BLOCK, &mask,NULL);; //SIGCHLD 시그널을 블락한다
182     if ((pid=fork())==0){; //자식일 경우
183         setpgid(0,0);; // 자신의 프로세스 그룹을 자신의 프로세스 아이디로 바꾼다
184         sigprocmask(SIG_UNBLOCK, &mask, NULL);; //SIGCHLD가 일어날 때 언블락한다
185         if(execve(argv[0],argv,enviro)<0){
186             printf("%s: Command not found. \n",argv[0]);
187             exit(0);}
188     }else {; //자식이 아닌 경우
189         if(!bg){; //프로그라운드 작업일시
190             { if(addjob(jobs,pid,FG,cmdline)){; //잡리스트 추가에 성공할 경우
191                 sigprocmask(SIG_UNBLOCK, &mask, NULL);; //부모 프로세서에서 SIGCHLD가 일어날 수 있
192                 용으로 언블락한다
193                 waitfg(pid, 0);; //프로그라운드 작업이 완료될 때까지 대기
194                 else{kill(-pid,SIGINT);; //실패할 경우 여러일므로 종료
195                 }
196             }else {; //백그라운드 작업일시
197                 if( addjob(jobs,pid,BG,cmdline)){; //잡리스트 추가에 성공할 경우
198                     sigprocmask(SIG_UNBLOCK, &mask, NULL);
199                 }
200             }
201         }
202     }
203 }
```

eval 함수에서 자식프로세서일 경우 pid를 그룹프로세서 id로 바꾸기위해
setpgid(0,0)을 해서 바꿨다. 그럼 그룹내 pid로 인식하기 때문에 조건에 만족하였다.