

2016 시스템 프로그래밍
- Lab 08 -

제출일자	2016.11.14
분 반	02
이 름	신종욱
학 번	2013024223

2.목차

3. 각 단계에 대한 해결방법

3-1 trace 00

- 1-sdriver 수행결과와 tsh 확인
- 2-Flow Chart
- 3-해결방법

3-2 trace 01

- 1-sdriver 수행결과와 tsh 확인
- 2-Flow Chart
- 3-해결방법

3-3 trace 02

- 1-sdriver 수행결과와 tsh 확인
- 2-Flow Chart
- 3-해결방법

3-4 trace 03

- 1-sdriver 수행결과와 tsh 확인
- 2-해결방법

3-5 trace 04

- 1-sdriver 수행결과와 tsh 확인
- 2-해결방법

3-6 trace 05

- 1-sdriver 수행결과와 tsh 확인
- 2-Flow Chart
- 3-해결방법

3-7 trace 06

- 1-sdriver 수행결과와 tsh 확인
- 2-해결방법

3-8 trace 07

- 1-sdriver 수행결과와 tsh 확인
- 2-Flow Chart
- 3-해결방법

3. 각 단계에 대한 해결방법

3-1 trace 00

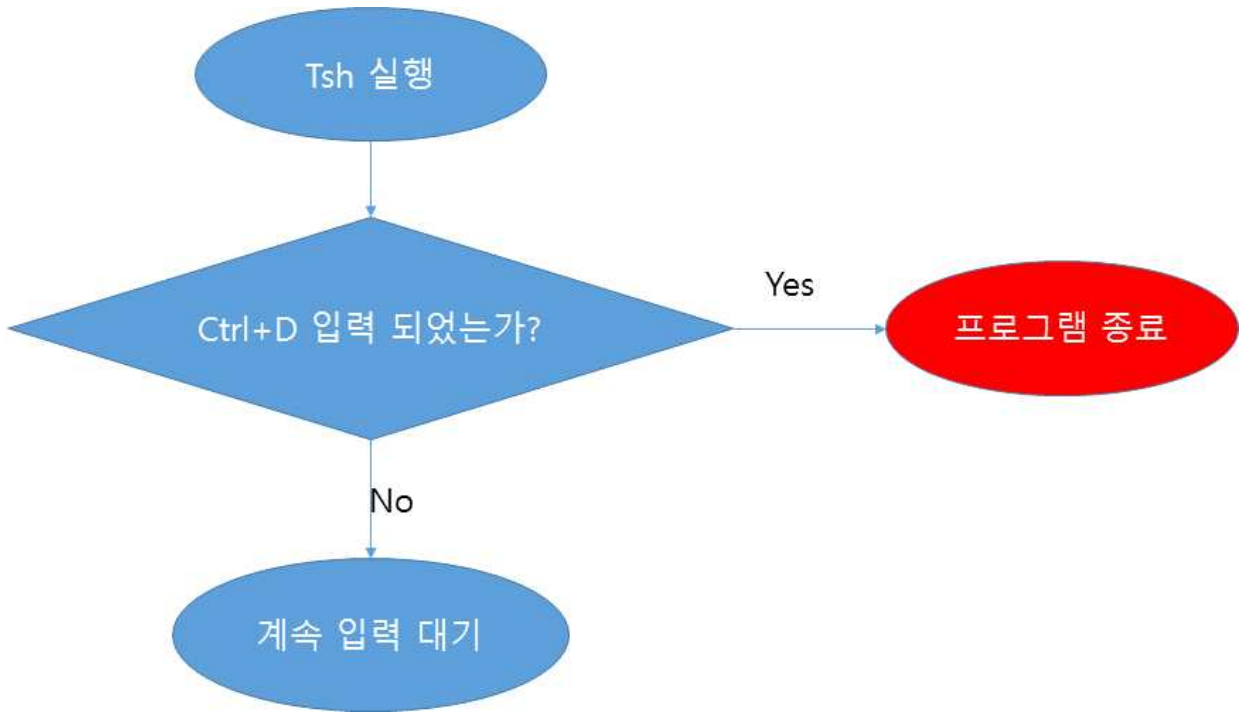
1- sdriver -V -t 00 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 00 -s ./tsh
Running trace00.txt...
Success: The test and reference outputs for trace00.txt matched!
Test output:
#
# trace00.txt - Properly terminate on EOF.
#
Reference output:
#
# trace00.txt - Properly terminate on EOF.
#
c201302423@localhost:~/08/shlab-handout$
```

실제 tsh에서 ctrl+d를 눌러 확인완료

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> c201302423@localhost:~/08/shlab-handout$
```

2-FlowChart



3-해결방법

원래 구현이 되어있습니다.

3-2 trace 01

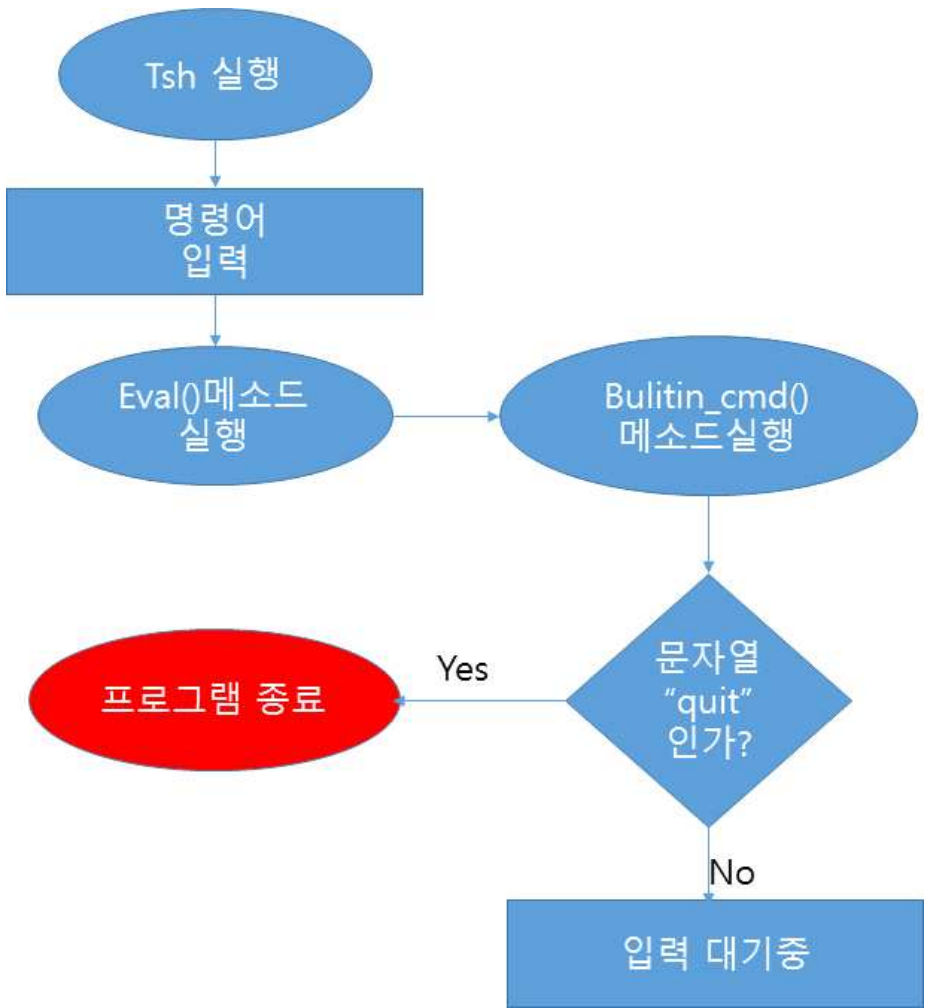
1-sdriver -V -t 01 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 01 -s ./tsh
Running trace01.txt...
Success: The test and reference outputs for trace01.txt matched!
Test output:
#
# trace01.txt - Process builtin quit command.
#
Reference output:
#
# trace01.txt - Process builtin quit command.
#
c201302423@localhost:~/08/shlab-handout$
```

tsh를 실행해서 quit를 입력하여서 종료되는지 확인하자.

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> quit
c201302423@localhost:~/08/shlab-handout$
```

2-FlowChart



3-해결방법

입력을 받을 때 만약 quit와 동일 하다면 프로그램을 종료한다고 정의하면된다.

명령어를 만들기 때문에 builtin_cmd에서 명령어를 추가해줘야하는데

문자열끼리 비교해야하기 때문에 strcmp를 써서 strcmp(cmd,"quit")를하면 만약 입력과 quit가 같으면 0을 리턴한다
if(!strcmp(cmd,"quit"))가 만족할 경우 exit(0)으로 종료한다고 하면된다.

3-3 trace 02

1-sdriver -V -t 02 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 02 -s ./tsh
Running trace02.txt...
Success: The test and reference outputs for trace02.txt matched!
Test output:
#
# trace02.txt - Run a foreground job that prints an environment variable
#
# IMPORTANT: You must pass this trace before attempting any later
# traces. In order to synchronize with your child jobs, the driver
# relies on your shell properly setting the environment.
OSTYPE=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

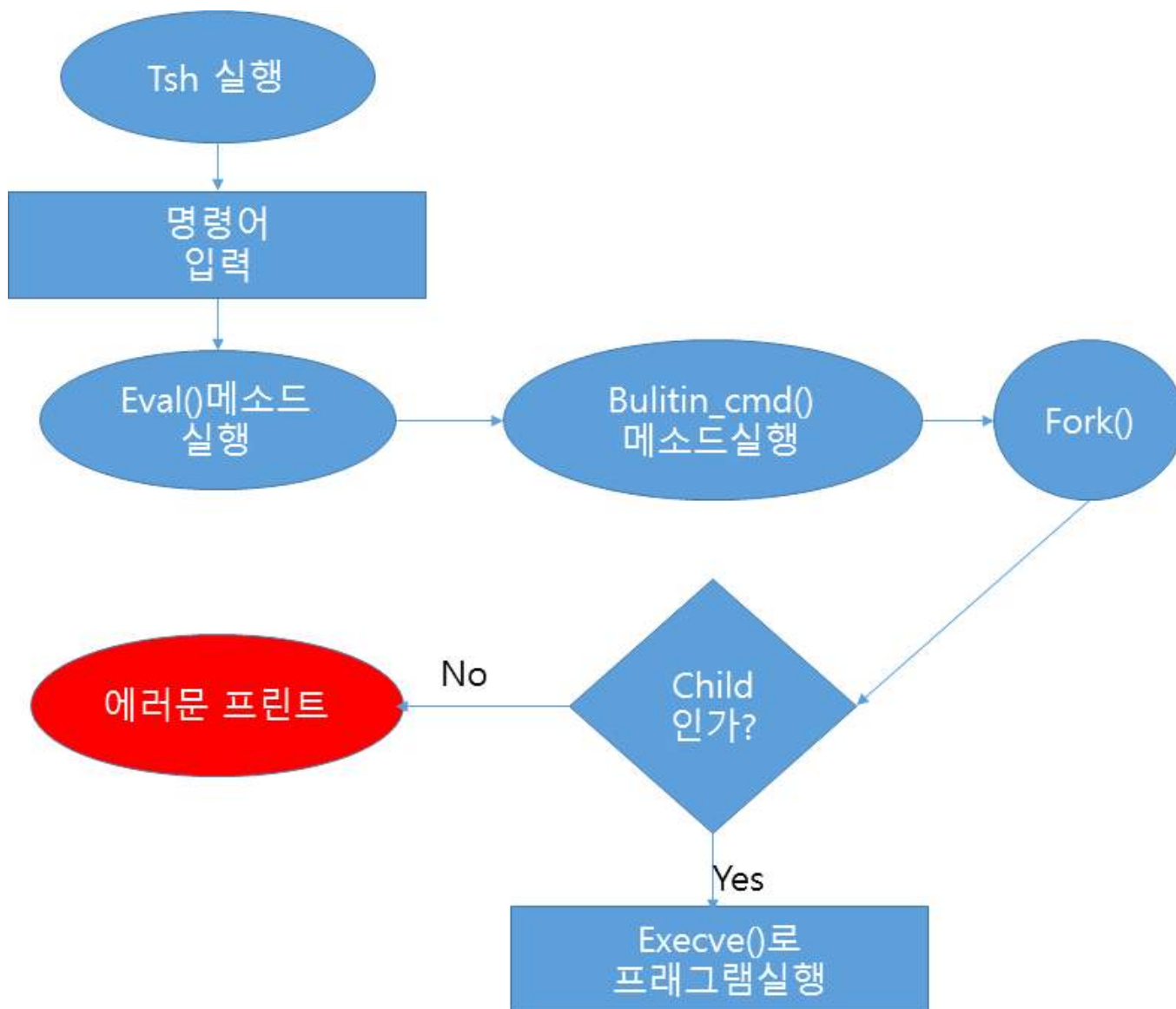
Reference output:
#
# trace02.txt - Run a foreground job that prints an environment variable
#
# IMPORTANT: You must pass this trace before attempting any later
# traces. In order to synchronize with your child jobs, the driver
# relies on your shell properly setting the environment.
OSTYPE=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

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

tsh에서 실행시

```
c201302423@localhost: ~/08/shlab-handout
eslab_tsh> c201302423@localhost:~/08/shlab-handout$
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> myenv
OSTYPE=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
eslab_tsh>
```

2-FlowChart



3-해결방법

셸에서는 프로그램을 실행하기 위해서는 일단 자식프로세서를 생성하고 자식프로세서에서 execve를 이용하여서 프로그램을 실행하면된다 그럼 일단 fork()메소드를 이용하여서 자식을 만들어준후 만약 fork의 리턴값이 0이면 즉 자식이면 execve를 이용하여서 프로그램을 실행해주면된다. 만약 execve가 리턴되어서 -1이라면 파일명 에러이다.

```

tsh.c (~08/shlab-handout) - VIM
170 void eval(char *cmdline)
171 {   int bg;
172     char *argv[MAXARGS];
173     pid_t pid;
174     bg= parseline(cmdline,argv);
175     if(!builtin_cmd(argv)){
176         if ((pid=fork())==0){execve(argv[0],argv,envron);
177             if(execve(argv[0],argv,envron)<0){
178                 printf("%s: Command not found. \n",argv[0]);
179                 exit(0);}
180     }}
~/08/shlab-handout/tsh.c [utf-8,unix][c] 1,180/545 31
  
```


3-4 trace03

1-sdriver -V -t 03 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 03 -s ./tsh
Running trace03.txt...
Success: The test and reference outputs for trace03.txt matched!
Test output:
#
# trace03.txt - Run a synchronizing foreground job without any arguments.
#
Reference output:
#
# trace03.txt - Run a synchronizing foreground job without any arguments.
#
c201302423@localhost:~/08/shlab-handout$
```

./tsh에서 실행시

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ vi trace03.txt
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1
eslab_tsh>
```

※flowchart는 trace02와 동일하다

2-해결방법

```
trace03.txt (~/.shlab-handout) - VIM
1 #
2 # trace03.txt - Run a synchronizing foreground job without any arguments.
3 #
4 ./myspin1
5
6 WAIT
7 SIGNAL
8
9 quit
10
```

trace03은 어떠한 매개변수도없이 프로그램을 foreground job형태로 실행하여야한다.
매개변수없는 ./myspin1이 실행되는걸로 확인해보면된다

3-5 trace04

1-sdriver -V -t 04 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 04 -s ./tsh
Running trace04.txt...
Success: The test and reference outputs for trace04.txt matched!
Test output:
#
# trace04.txt - Run a foreground job with arguments.
#
tsh> quit

Reference output:
#
# trace04.txt - Run a foreground job with arguments.
#
tsh> quit

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

인자가 있을 때 foreground job형태로 잘 실행되는지 확인하는 것이다.
./tsh에서 실행시

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> /bin/echo -e tsh\076 quit
tsh> quit
eslab_tsh>
```

※flowchart는 trace02와 동일하다

2-해결방법

txt확인하기

```
trace04.txt (~/08/shlab-handout) - VIM
1 #
2 # trace04.txt - Run a foreground job with arguments.
3 #
4 /bin/echo -e tsh\076 quit
5 NEXT
6 quit
7
```

리눅스 명령어 echo는 주어진 문자열을, 문자열 사이에 포함된 공백과 줄 마지막에 개행문자를 포함하여 표준출력으로 출력하는 명령어이다.

-e옵션은 문자열에서 역슬래시(\)와 조합되는 이스케이프 문자(escape sequence)를 인용부호(")로 묶어 인식하는 옵션이다

\076은 아스키코드로 바꾸면 >로 바뀌게되어 출력이 tsh> quit로 바뀌어 출력하게된다.

이스케이프를 이용하여서도 출력이 잘되는지 확인하면 끝이다.

3-6 trace 05

1-sdriver -V -t 05 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 05 -s ./tsh
Running trace05.txt...
Success: The test and reference outputs for trace05.txt matched!
Test output:
#
# trace05.txt - Run a background job.
#
tsh> ./myspin1 &
(1) (16043) ./myspin1 &
tsh> quit

Reference output:
#
# trace05.txt - Run a background job.
#
tsh> ./myspin1 &
(1) (16051) ./myspin1 &
tsh> quit

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

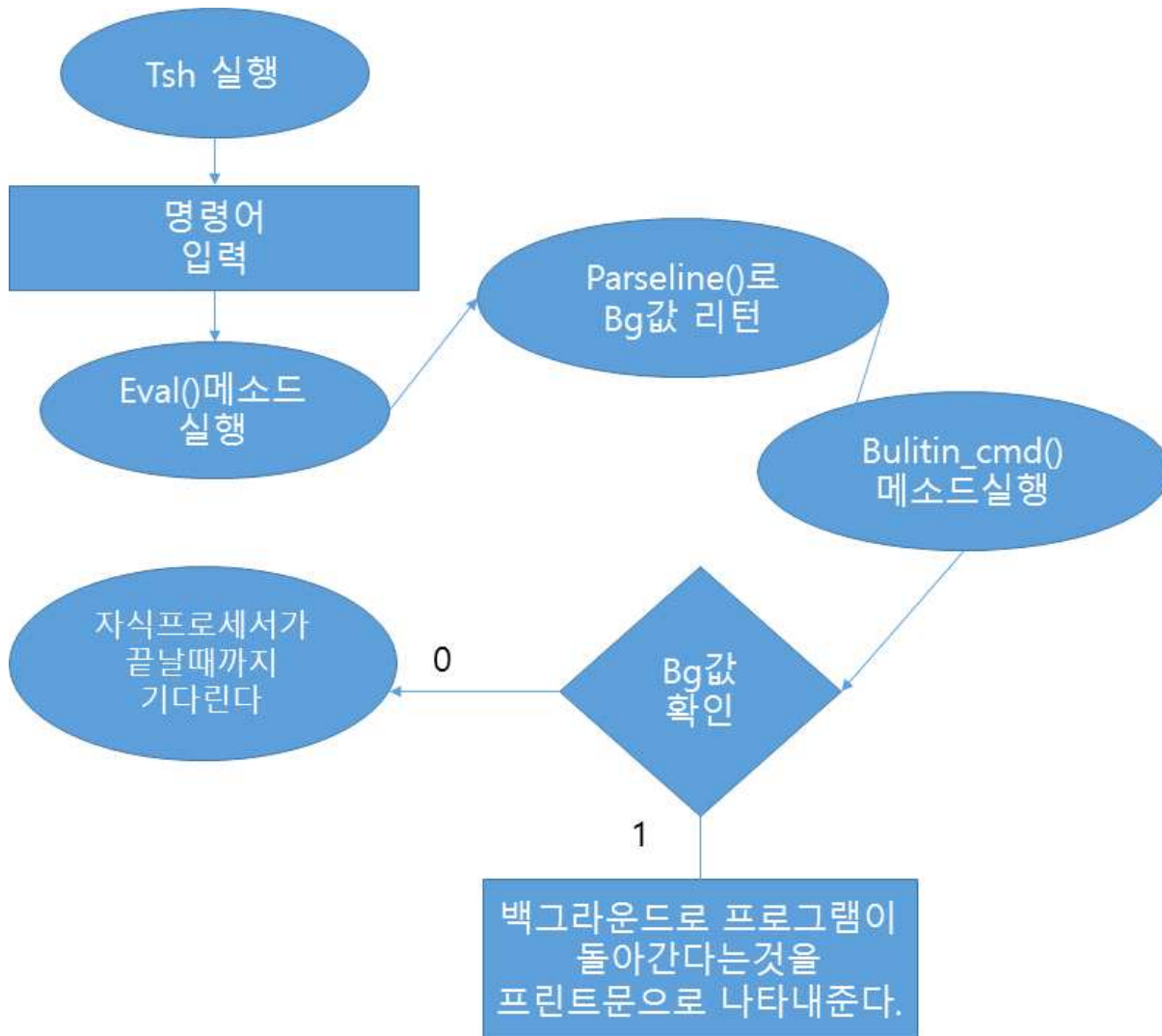
txt파일 확인

```
trace05.txt (~/08/shlab-handout) - VIM
1 #
2 # trace05.txt - Run a background job.
3 #
4 /bin/echo -e tsh\076 ./myspin1 \046
5 NEXT
6 ./myspin1 &
7 NEXT
8
9 WAIT
10 SIGNAL
11
12 /bin/echo -e tsh\076 quit
13 NEXT
~/08/shlab-handout/trace05.txt [utf-8,unix][text] 1,1/14 Top
```

./tsh에서 직접 실행

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> .myspin1 &
.myspin1: Command not found.
(1) (12812) .myspin1 &
eslab_tsh> quit
c201302423@localhost:~/08/shlab-handout$
```

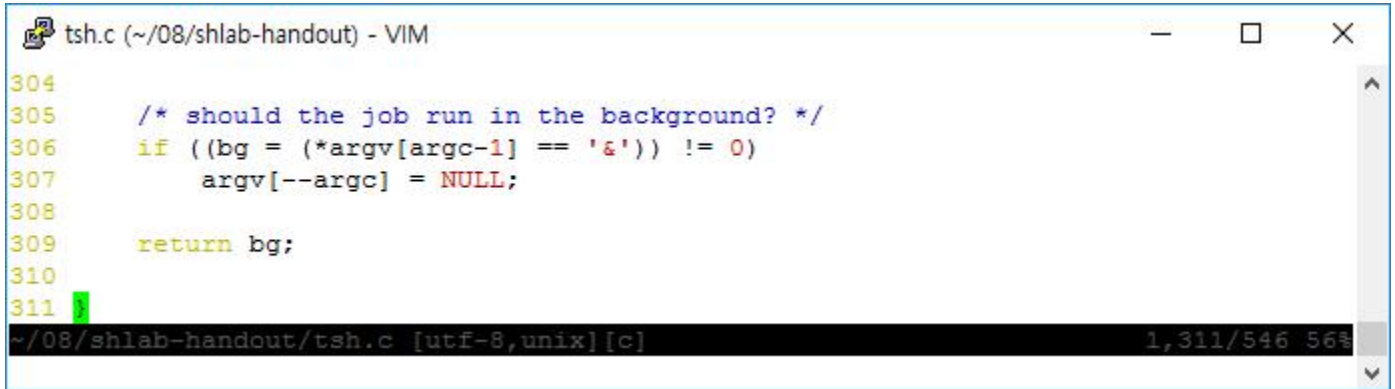
2-FlowChart



3-해결방법

trace5는 프로그램을 background job 형태로 할때도 잘되는지 확인하는 것이다.

parseline 함수를 이용해서 background로 실행했는지 확인 할 수있는데 parseline은 문자열을 스페이스 단위로 잘라서 저장하는 함수인데 만약 맨마지막 문자가 &일 경우에는 1을 리턴 아니면 0을 리턴한다
parseline함수의 한 부분이다.



```
tsh.c (~/.shlab-handout) - VIM
304
305 /* should the job run in the background? */
306 if ((bg = (*argv[argc-1] == '&')) != 0)
307     argv[--argc] = NULL;
308
309 return bg;
310
311
~/08/shlab-handout/tsh.c [utf-8,unix][c] 1,311/546 56%
```

다음은 완성한 코드이다



```
tsh.c (~/.shlab-handout) - VIM
170 void eval(char *cmdline)
171 {
172     int bg;
173     char *argv[MAXARGS];
174     pid_t pid;
175     bg= parseline(cmdline,argv);
176     if(!builtin_cmd(argv)){
177         if ((pid=fork())==0){execve(argv[0],argv,enviro);
178             if(execve(argv[0],argv,enviro)<0){
179                 printf("%s: Command not found. \n",argv[0]);
180                 exit(0);}}
181     if(!bg){
182         waitpid(pid,NULL,0);
183     }
184     else {
185         addjob(jobs,pid,BG,cmdline);
186         int oid=pid2jid(pid);
187         printf("(%d) (%d) %s",oid,pid,cmdline);
188     }
189 }
190
191
192 return;
193 }
~/08/shlab-handout/tsh.c [utf-8,unix][c] 1,170/546 32%
```

bg가 만약 0이라면 foreground로 실행 될경우고 이럴경우는 자식의 프로세스가 종료될 때 까지 기다려야 한다

waitpid함수를 써서 pid의값은 자식의 값일테니 waitpid(pid,null,0)을 사용한다

그 외는 background 작업일테니 addjob을해주고 tshref와 같은 printf 형식으로 printf문을 추가하면된다.

여기에서 pid2jid는 프로세서 id를 작업 id로 맵핑하는 것이다.

3-7 trace 06

1-sdriver -V -t 06 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 06 -s ./tsh
Running trace06.txt...
Success: The test and reference outputs for trace06.txt matched!
Test output:
#
# trace06.txt - Run a foreground job and a background job.
#
tsh> ./myspin1 &
(1) (16528) ./myspin1 &
tsh> ./myspin2 1

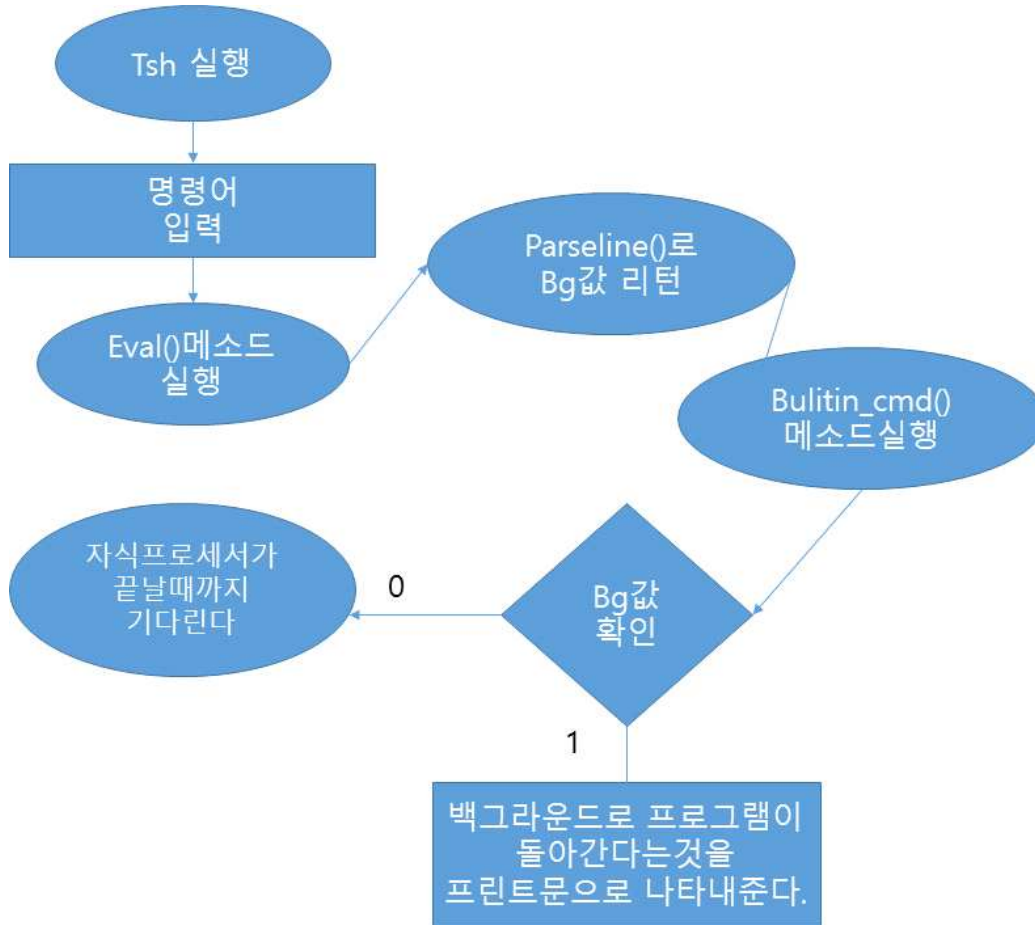
Reference output:
#
# trace06.txt - Run a foreground job and a background job.
#
tsh> ./myspin1 &
(1) (16540) ./myspin1 &
tsh> ./myspin2 1

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

./tsh에서 직접실행시

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1 &
(1) (7776) ./myspin1 &
eslab_tsh> ./myspin2 1
eslab_tsh>
```

2-FlowChart



3-해결방법

trace06은 백그라운드와 포그라운드 둘다 잘돌아가는지 확인하는 과정이다 이미 앞단계에서 포그라운드와 백그라운드를 다 구현했음으로 자동적으로 조건을 만족할 것이다.

3-8 trace 07

1-sdriver -V -t 07 -s ./tsh 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./sdriver -V -t 07 -s ./tsh
Running trace07.txt...
Success: The test and reference outputs for trace07.txt matched!
Test output:
#
# trace07.txt - Use the jobs builtin command.
#
tsh> ./myspin1 10 &
(1) (16954) ./myspin1 10 &
tsh> ./myspin2 10 &
(2) (16956) ./myspin2 10 &
tsh> jobs
(1) (16954) Running      ./myspin1 10 &
(2) (16956) Running      ./myspin2 10 &

Reference output:
#
# trace07.txt - Use the jobs builtin command.
#
tsh> ./myspin1 10 &
(1) (16964) ./myspin1 10 &
tsh> ./myspin2 10 &
(2) (16966) ./myspin2 10 &
tsh> jobs
(1) (16964) Running      ./myspin1 10 &
(2) (16966) Running      ./myspin2 10 &

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

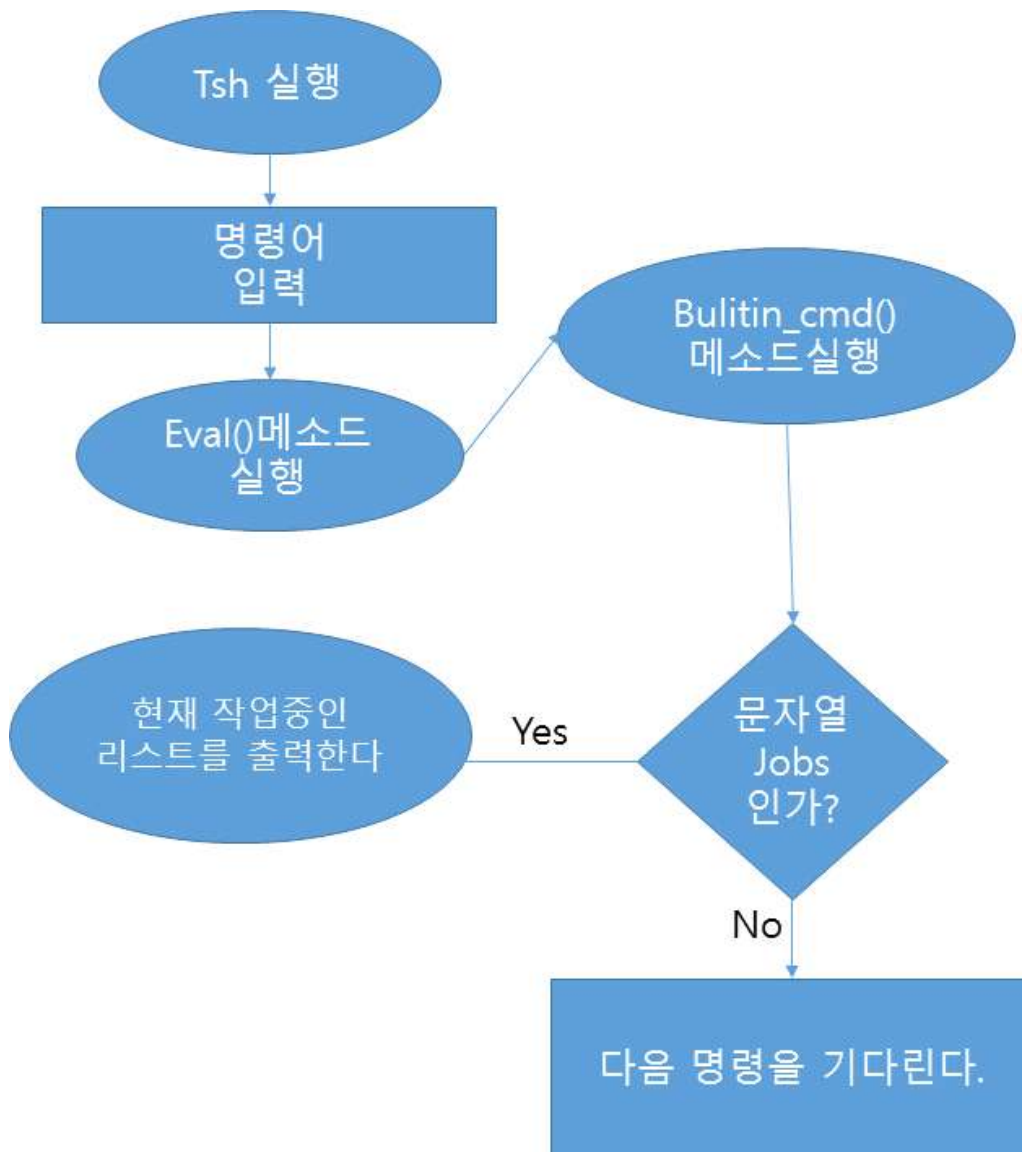
txt문 확인

```
trace07.txt (~/.shlab-handout) - VIM
1 #
2 # trace07.txt - Use the jobs builtin command.
3 #
4 /bin/echo -e tsh\076 ./myspin1 10 \046
5 NEXT
6 ./myspin1 10 &
7 NEXT
8
9 /bin/echo -e tsh\076 ./myspin2 10 \046
10 NEXT
11 ./myspin2 10 &
12 NEXT
13
14 WAIT
15 WAIT
16
17 /bin/echo -e tsh\076 jobs
18 NEXT
19 jobs
20 NEXT
21
~/08/shlab-handout/trace07.txt [utf-8,unix][text] 1,1/25 Top
```


직접실행하여서 확인

```
c201302423@localhost: ~/08/shlab-handout
c201302423@localhost:~/08/shlab-handout$ ./tsh
eslab_tsh> ./myspin1 10 &
(1) (1602) ./myspin1 10 &
eslab_tsh> ./myspin2 10 &
(2) (1702) ./myspin2 10 &
eslab_tsh> jobs
(1) (1602) Running    ./myspin1 10 &
(2) (1702) Running    ./myspin2 10 &
eslab_tsh>
```

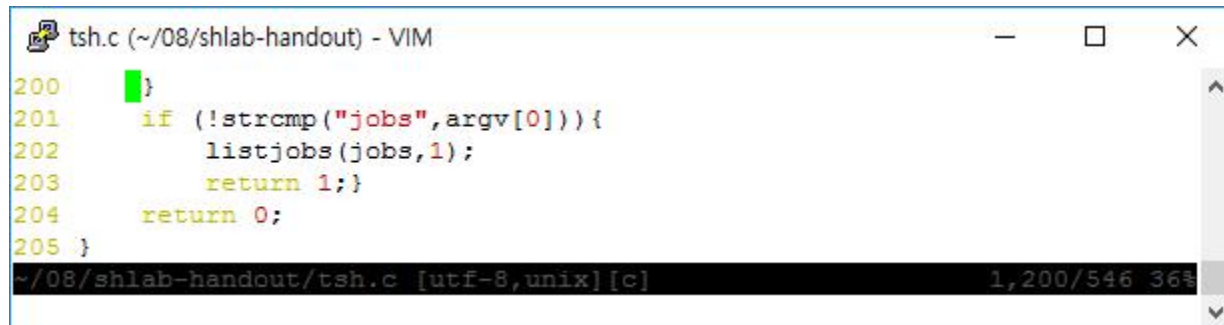
2-Flow chart



3-해결방법

일단 builtin 명령어를 구현해야함으로 builtin_cmd 메소드에서 jobs라는 문자열을 받았을 때 현재 작업중인 리스트를 출력하면된다.

작업리스트를 출력하는데에는 listjobs를 쓰면 되기때문에



```
tsh.c (~/08/shlab-handout) - VIM
200 }
201 if (!strcmp("jobs", argv[0])){
202     listjobs(jobs, 1);
203     return 1;
204 }
205 return 0;
~/08/shlab-handout/tsh.c [utf-8,unix][c] 1,200/546 36%
```

builtin_cmd메소드에 코드를 추가하면된다