2018 시스템 프로그래밍
- Lab 06 -

제출일자	2018.11.08
분 반	01
이 름	이규봉
학 번	201502085

Trace 번호 (00 01 02 etc)

eval , builtit_cmd 코드

```
170 void eval(char *cmdline)
171 {
172     // trace00
173     if(feof(stdin)){
174         fflush(stdout);
175         exit(0);
176     }
177
178     // trace01
179     char *argv[MAXARGS];
180     parseline(cmdline,argv);
181     builtin_cmd(argv);
182
183     // trace02, 03, 04
184     pid_t pid;
185
186     parseline(cmdline,argv);
187
188     if(!builtin_cmd(argv)){
189         if((pid=fork())==0){
190             if((execve(argv[0],argv,environ) < 0)){
191                 printf("%s: command not found\n", argv);
192                 exit(0);
193             }
194         }
195     }
196
```

```
200 int builtin_cmd(char **argv)
201 {
202     // trace01
203     char *cmd = argv[0];
204
205     if(!strcmp(cmd,"quit")){exit(0);}
206
207     return 0;
208 }
209
```

sdriver 실행 결과

```
b201502085@2018-sp:~/Shelllab/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.
#
```

trace00 test

(./sdriver -V -t 00 –s ./tsh)

```
b201502085@2018-sp:~/Shelllab/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.
#
```

trace01 test

(./sdriver -V -t 01 –s ./tsh)

```
b201502085@2018-sp:~/Shelllab/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
```

trace02 test

(./sdriver -V -t 02 –s ./tsh)

```
b201502085@2018-sp:~/Shelllab/shlab-handout$ ./sdriver -V -t 00 -s ./tshref
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.
#
```

(./sdriver -V -t 00 –s ./tshref)

```
b201502085@2018-sp:~/Shelllab/shlab-handout$ ./sdriver -V -t 01 -s ./tshref
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.
#
```

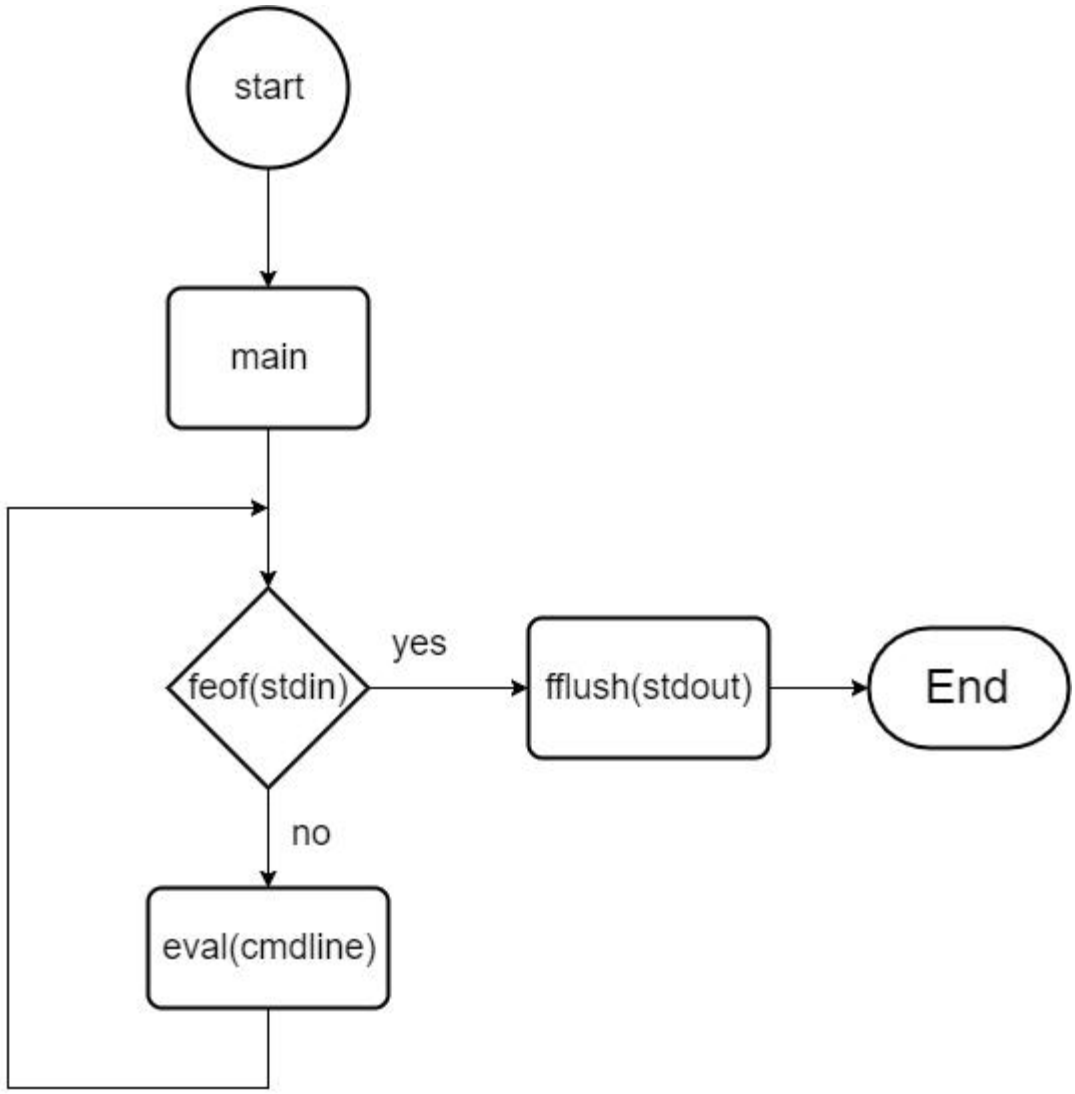
(./sdriver -V -t 01 –s ./tshref)

```
b201502085@2018-sp:~/Shelllab/shlab-handout$ ./sdriver -V -t 02 -s ./tshref
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
```

(./sdriver -V -t 02 –s ./tshref)

각 trace 별 플로우 차트, Trace 해결 방법 설명



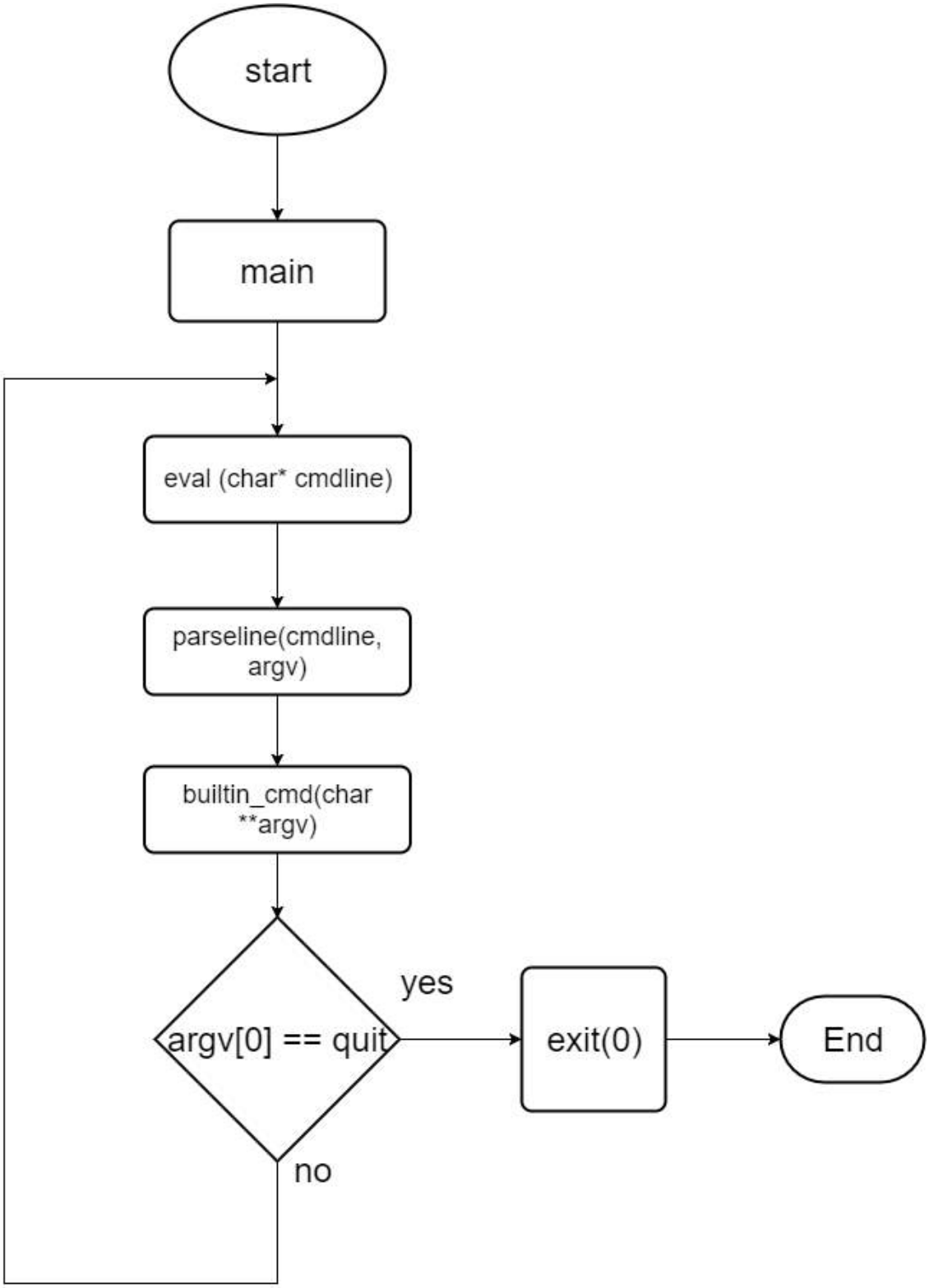
trace00

위는 trace00의 flow chart입니다. 쉘을 시작하면 h,v,p의 인자를 따로 주지 않는다면 main의 while문을 돌기 시작하게 됩니다. (Signal에 관한 코드들에 대한 내용은 생략하였습니다)

while문에선 기본적으로 cmdline을 읽어들여, eval 함수를 실행시키는 역할을 하며, 만약 stdin이 EOF라면, fflush(stdout)을 실행시킨 후 Exit(0)을 호출해 프로그램을 종료시킵니다.

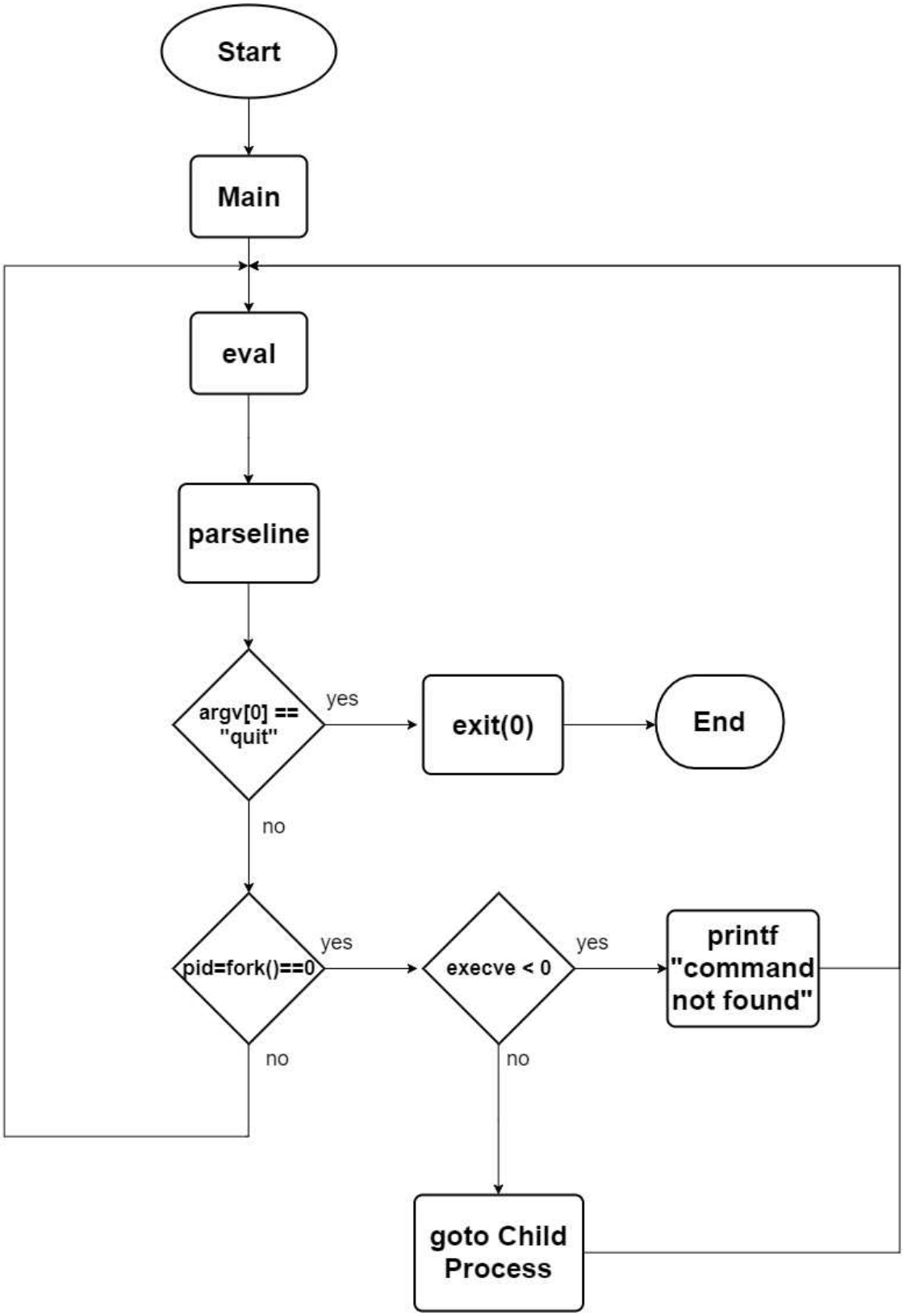
만약 feof(stdin)이 False라면, eval 함수를 실행한 후, 다시 main의 while문으로 돌아오게 됩니다. (eval 함수에서 exit 문이 호출되지 않은 경우)

trace00의 경우, main에 관련된 코드가 이미 작성되어 있어 위처럼 순서도를 표현하여 보았습니다.



trace01

위는 trace01의 flow chart입니다. trace01의 경우, main에서 eval을 호출하면, eval에서 입력된 cmdline들을 parseline 함수를 통해 parse하여, argv 배열로 만듭니다. 후에 builtin_cmd란 함수로 만든 argv 배열을 넘겨주면, argv[0], 즉 커맨드로 입력한 것이 quit인 지를 검사합니다. 만약 입력된 값이 quit이 맞다면, exit(0)를 호출하여 프로그램을 종료합니다.



trace02

위는 trace02를 flow chart로 표현하여 본 것입니다. main에서 호출한 eval은 parseline으로 cmdline을 argv 배열에 저장하고 이를 builtin_cmd에 넘겨줍니다. builtin_cmd에선 argv[0]이 quit이라면 쉘을 종료하고, quit이 아니라면 pid에 fork()를 대입하고, pid가 0인지를 검사하여, 0이라면, 즉 자식프로세스라면 execve를 호출합니다

execve는 argv[0]이 유효한 실행파일인지 검사하여, 유효하지 않은 경우엔 0을 리턴하여 'command not found' 란 문자열이 콘솔에 출력되도록 합니다. 유효한 실행파일이라면 입력받은 파일을 Foreground 형태로 실행시키며, 자식 프로세스가 실행중일 땐, 콘솔에 입력을 받을 수 없습니다. 자식 프로세스가 종료되고 난 후엔, 다시 쉘로 돌아가 다시 입력을 받을 수 있게 합니다.