Lab 10 Shell Lab Report

20220665 유영준

Lab10에서는 shell에 대해 배웠다. shell은 사용자의 입력에 기반하여 프로그램을 실행하는 interactive command-line interpreter이다. shell의 기본 function에 대해 배우고, 이를 간단히 구현하는 과제가 shell lab 과제로 나왔다.

Lab11에서는 Exceptional Control Flow에 대해 배웠다. Exception이 발생할 경우 처리하는 과정에 대해 배우고, process event를 알려주는 siganl이 동작하는 방법에 대해 배웠다. 각각의 siganl에 대해 사용자는 새로운 action을 정의하여 signal hnadling을 구현할 수 있다.

Lab10과 Lab11과 관련하여 간단한 shell을 구현하는 과제가 출제되었고, 해결 방법은 다음과 같다. trace01.txt 파일부터 trace16.txt 파일을 차례대로 따라가며 과제를 해결하였다. 총 7개의 함수를 구현했고, 각 함수의 구현 방법과 실행 예시는 다음과 같다.

1. eval

- eval 함수는 main 함수에서 호출하는 기본 routine이 되는 함수로, command line을 parsing하고 해석하여 instruction을 수행한다.
- 먼저 command line을 buf에 저장한 뒤 parsing하여 argv에 저장한다. parseline 함수는 background job의 경우 1을 반환하므로 이를 지역변수 bg에 저장하여 사용하였다. 빈 입력일 경우 예외처리 해주었다.
- 만약 target program이 built-in command일 경우 builtin_cmd 함수를 호출하여 처리해 주었다. builtin_cmd 함수는 built-in command일 경우 1을, built-in command가 아닐 경우 0을 반환하도록 하였다.
- target program이 built-in command가 아닐 경우 fork 함수와 execve 함수를 호출하여 target program이 실행될 수 있도록 하였다.
- parent process가 addjob을 통해 process를 추가하기 전에 child process가 sigchld handler를 통해 reaping되는 것을 막기 위해 fork 함수를 호출하기 전에 sigprocmask를 이용하여 SIGCHLD signal을 block하였다. 그리고 나서 fork 함수를 호출하고 child process의 경우 execve로 프로그램을 실행하기 전에 해당 signal을 unblock 해주었다. parent process의 경우 addjob으로 job을 추가한 후 해당 signal을 unblock 해주었다.
- parent process에서 child process가 foreground에서 실행되는 경우, 해당 job을 FG 상태로 추가하고, waitfg 함수를 호출하여 기다리도록 하였다. background에서 실행되는 경우,

해당 job을 BG 상태로 추가하고, tshref와 실행 결과가 일치하도록 print 해주었다.

- system call인 execve와 fork 함수가 정상적으로 작동하지 않는 경우 예외처리 해주었다.

2. builtin_cmd

- builtin_cmd 함수는 built-in command를 인식하고 해석하는 함수로 command line의 첫 번째 단어, 즉 argv[0]을 지역변수 name에 저장하여 name이 built-in command에 해당할 경우 각각의 instruction을 수행하도록 하였다. string이 같은지 확인하기 위해 strcmp 함수를 사용하였다.
- name이 quit일 경우, exit 함수를 호출하여 shell을 종료하였다.
- name이 jobs일 경우, 현재 job list를 출력하였다. 이를 구현하기 위해 코드에서 제공하는 listjobs 함수를 호출하여 사용하였다.
- name이 fg 또는 bg일 경우 do_bgfg 함수를 호출하였다.
- 각 built-in command에 대해 instruction을 수행한 후 1을 return하고, built-in command가 아닌 경우 0을 return하였다.

3. do_bgfg

- do_bgfg 함수는 built-in bg and fg command를 처리해주는 함수이다.
- command line을 parsing한 argv에서 첫 번째 string, 즉 argv[0]은 fg 또는 bg이고, 두 번째 string, 즉 argv[1]는 pid 또는 jid이다. 이를 지역변수 id에 저장하여 사용하였다.
- id가 jid인 경우, SIGCONT signal을 보낸 뒤 FG 또는 BG로 재시작하였다. SIGCONT signal을 보내기 위해 kill 함수를 호출하였고, 인자로 –pid를 주어야 하므로 getjobjid 함수를 사용하여 job의 pid를 구했다. signal을 보낸 다음 job의 state를 바꾼 뒤, FG인 경우 waitfg 함수를 호출하였고, BG인 경우 형식에 맞게 출력하였다.
- id가 pid인 경우도 마찬가지로 SIGCONT signal을 보낸 뒤 FG 또는 BG로 재시작하였다. argument가 pid인지 확인하기 위해 isdigit 함수를 사용하였다.
- id argument 부분이 존재하지 않는 경우, jid 또는 pid가 유효하지 않은 경우, system call인 kill 함수가 정상적으로 작동하지 않는 경우, argument가 id의 형식이 아닌 경우 각각 tshref와 실행 결과가 일치하도록 적절한 error 문구를 출력하였다.

4. waitfg

- waitfg 함수는 foreground process가 멈추거나 종료될 때까지 기다리도록 하는 함수이다.
- 인자로 받은 pid의 job이 존재하고, pid가 0이 아닌지 확인하여 해당 pid가 유효한 pid인지 확인하였다. 유효한 pid의 경우 sleep 함수를 이용한 busy loop을 만들어 기다리도록 하였다.

5. sigchld_handler

- sigchld_handler 함수는 SIGCHILD signal을 catch하는 함수이다.
- waitpid를 호출하는 while문을 돌면서 종료되거나 중지된 child process를 handling하였다. waitpid 함수의 인자로 WNOHANG와 WUNTRACED를 사용하여 process가 정지된 상태에도 waitpid 함수가 반환되도록 하였다.
- Terminated by signal인 경우 tshref와 실행 결과가 일치하도록 적절한 문구를 출력한 뒤 deletejob 함수를 호출하여 해당 job을 제거하였다.
- Stopped by signal인 경우 tshref와 실행 결과가 일치하도록 적절한 문구를 출력한 뒤 해당 job의 state를 ST로 변경하였다.
- Exited의 경우 다른 문구를 출력하지 않고 해당 job을 제거하였다.

6. sigint_handler

- sigint_handler 함수는 SIGINT (ctrl-c) signal을 catch하는 함수이다.
- 유효한 foreground job에 대해 kill 함수를 호출하여 해당 signal을 catch한다.
- system call인 kill 함수가 정상적으로 작동하지 않는 경우 예외처리 해주었다.

7. sigtstp_handler

- sigtstp_handler 함수는 SIGTSTP (ctrl-z) signal을 catch하는 함수이다.
- sigint_handler와 같은 방식으로 동작한다.
- system call인 kill 함수가 정상적으로 작동하지 않는 경우 예외처리 해주었다.

8. 실행 예시

- 각 trace 파일의 실행 예시는 다음과 같다.

- trace01.txt

```
[colin31472@programming2 ~]$ make test01
./sdriver.pl -t trace01.txt -s ./tsh -a "-p"
#
# trace01.txt - Properly terminate on EOF.
#
```

- trace02.txt

```
[colin31472@programming2 ~]$ make test02
./sdriver.pl -t trace02.txt -s ./tsh -a "-p"
#
# trace02.txt - Process builtin quit command.
#
```

- trace03.txt

```
[colin31472@programming2 ~]$ make test03
./sdriver.pl -t trace03.txt -s ./tsh -a "-p"
#
# trace03.txt - Run a foreground job.
#
tsh> quit
```

- trace04.txt

```
[colin31472@programming2 ~]$ make test04
./sdriver.pl -t trace04.txt -s ./tsh -a "-p"
#
# trace04.txt - Run a background job.
#
tsh> ./myspin 1 &
[1] (14808) ./myspin 1 &
```

- trace05.txt

```
[colin31472@programming2 ~]$ make test05
./sdriver.pl -t trace05.txt -s ./tsh -a "-p"
#
# trace05.txt - Process jobs builtin command.
#
tsh> ./myspin 2 &
[1] (14834) ./myspin 2 &
tsh> ./myspin 3 &
[2] (14836) ./myspin 3 &
tsh> jobs
[1] (14834) Running ./myspin 2 &
[2] (14836) Running ./myspin 3 &
```

- trace06.txt

```
[colin31472@programming2 ~]$ make test06
./sdriver.pl -t trace06.txt -s ./tsh -a "-p"
#
# trace06.txt - Forward SIGINT to foreground job.
#
tsh> ./myspin 4
Job [1] (15245) terminated by signal 2
```

- trace07.txt

```
[colin31472@programming2 ~]$ make test07
./sdriver.pl -t trace07.txt -s ./tsh -a "-p"
#
# trace07.txt - Forward SIGINT only to foreground job.
#
tsh> ./myspin 4 &
[1] (15283) ./myspin 4 &
tsh> ./myspin 5
Job [2] (15286) terminated by signal 2
tsh> jobs
[1] (15283) Running ./myspin 4 &
```

- trace08.txt

```
[colin31472@programming2 ~]$ make test08
./sdriver.pl -t trace08.txt -s ./tsh -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (15344) ./myspin 4 &
tsh> ./myspin 5
Job [2] (15346) stopped by signal 20
tsh> jobs
[1] (15344) Running ./myspin 4 &
[2] (15346) Stopped ./myspin 5
```

- trace09.txt

```
[colin31472@programming2 ~]$ make test09
./sdriver.pl -t trace09.txt -s ./tsh -a "-p"
#
# trace09.txt - Process bg builtin command
#
tsh> ./myspin 4 &
[1] (15376) ./myspin 4 &
tsh> ./myspin 5
Job [2] (15378) stopped by signal 20
tsh> jobs
[1] (15376) Running ./myspin 4 &
[2] (15378) Stopped ./myspin 5
tsh> bg %2
[2] (15378) ./myspin 5
tsh> jobs
[1] (15376) Running ./myspin 4 &
[2] (15378) Running ./myspin 5
```

- trace10.txt

```
[colin31472@programming2 ~]$ make test10
./sdriver.pl -t trace10.txt -s ./tsh -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (15445) ./myspin 4 &
tsh> fg %1
Job [1] (15445) stopped by signal 20
tsh> jobs
[1] (15445) Stopped ./myspin 4 &
tsh> fg %1
tsh> jobs
```

- trace11.txt, trace12.txt, trace13.txt

(생략)

- trace14.txt

```
[colin31472@programming2 ~]$ make test14
./sdriver.pl -t trace14.txt -s ./tsh -a "-p"
#
# trace14.txt - Simple error handling
#
tsh> ./bogus
./bogus: Command not found
tsh> ./myspin 4 &
[1] (15594) ./myspin 4 &
tsh> fg
fg command requires PID or %jobid argument
tsh> bg
bg command requires PID or %jobid argument
tsh> fg a
fg: argument must be a PID or %jobid
tsh> bg a
bg: argument must be a PID or %jobid
tsh> fg 9999999
(9999999): No such process
tsh> bg 9999999
(9999999): No such process
tsh> fg %2
%2: No such job
tsh> fg %1
Job [1] (15594) stopped by signal 20
tsh> bg %2
%2: No such job
tsh> bg %1
[1] (15594) ./myspin 4 &
tsh> jobs
[1] (15594) Running ./myspin 4 &
```

- trace15.txt

```
[colin31472@programming2 ~]$ make test15
./sdriver.pl -t trace15.txt -s ./tsh -a "-p"
#
# trace15.txt - Putting it all together
#
tsh> ./bogus
./bogus: Command not found
tsh> ./myspin 10
Job [1] (15644) terminated by signal 2
tsh> ./myspin 3 &
[1] (15648) ./myspin 3 &
tsh> ./myspin 4 &
[2] (15650) ./myspin 4 &
tsh> jobs
[1] (15648) Running ./myspin 3 &
[2] (15650) Running ./myspin 4 &
tsh> fg %1
Job [1] (15648) stopped by signal 20
tsh> jobs
[1] (15648) Stopped ./myspin 3 &
[2] (15650) Running ./myspin 4 &
tsh> bg %3
%3: No such job
tsh> bg %1
[1] (15648) ./myspin 3 &
tsh> jobs
[1] (15648) Running ./myspin 3 &
[2] (15650) Running ./myspin 4 &
tsh> fg %1
tsh> quit
```

- trace16.txt

```
[colin31472@programming2 ~]$ make test16
./sdriver.pl -t trace16.txt -s ./tsh -a "-p"
#
# trace16.txt - Tests whether the shell can handle SIGTSTP and SIGINT
#     signals that come from other processes instead of the terminal.
#
tsh> ./mystop 2
Job [1] (15704) stopped by signal 20
tsh> jobs
[1] (15704) Stopped ./mystop 2
tsh> ./myint 2
Job [2] (15709) terminated by signal 2
```