Shell Lab Report

-2019-17577 경제학부 정지후-

1. Please explain how to implement each function.

1.1 Eval

Assignment pdf 파일에 나온 순서대로 구현했다. 첫째로, parseline함수를 이용해 argument를 parse하고 맨 마지막 character가 &인지를 체크하여 background job인지 foreground job인지 확인한다. 다음으로, 첫번째 argument가 builtin cmd에 해당하는지(bg,fg,quit,jobs)인지 확인하기 위해 builtin_cmd 함수를 이용한다. 만약, builtin_cmd 가 아니라면, block signal한 이후 fork해서 child process와 parent process를 만든다. Child process에서는 unblock the signal하고, new process group id를 배정한다. 이는 foreground process group에는 우리 shell 1나의 프로세스만 존재하도록 한다. 이후, execve함수를 call해서 프로그램을 실행시킨다.
Parent process에서는 프로세스를 addjob으로 넣고, unblock the signal한다. Background job일 경우, 적절한 log message를 print한다.

1.2 builtin_cmd

builtin_cmd는 argument의 첫 번째 인자가 builtin command인지 확인한다. 만약, 첫 번째 인자가 quit이면, exit을 이용해 즉시 종료한다. 만약, 첫 번째 인자가 “fg” 또는 “bg”이면 do_bgfg 함수를 이용해 실행시키고 1을 return한다. 만약, 첫번째 인자가 “jobs”이면, listjobs함수를 이용해 처리하고 1을 return한다. 위의 command가 아니라면, 0을 return한다.

1.3 do_bgfg

먼저, 첫번째 argument가 fg인지 bg인지를 check한다. 만약 fg,bg 이후 argument가 없다면, “bg(fg) command requires PID or %jobid argument” 를 print한다. 만약, fg,bg 이후 argument가 %+숫자라면, jobid를 이용한 것이고, 그렇지 않으면 pid를 이용한 것이다.

주어진 숫자를 이용해, getjobjid 또는 getjobpid 함수를 이용해 해당하는 job을 찾는다. SIGCONT 시그널을 해당 pid그룹에 보내고, 첫 번째 argument가 무엇인지에 따라, 해당 state를 바꾸어준다. Bg라면, message 출력하고, fg라면 waitfg함수를 이용한다.

1.4 waitfg

lab assignment guide에 따라 while과 sleep을 이용해 foreground job을 기다리게 했다.

1.5 sigchld_handler

Lab assignment guide에 따라 waitpid에 다음과 같은 옵션을 주었다. waitpid(-1, &status, WNOHANG |WUNTRACED)이가 -1이 되지 않을 때까지 while loop을 돌렸다. 만약, signal에 의해 process가 terminate되면 해당 log message와 함께 delete job을 하여 reap한다. 정상적으로 terminate된 것도 마찬가지로 reap하고, signal 에 의해 stop된 것은 state를 ST로 바꾸어 준다.

1.6 sigint_handler

Current process의 pid를 fgpid함수를 이용해 구한 뒤에, kill함수의 argument에 해당 pid를 음수값으로 넣어줌으로써 foreground group에 signal을 send한다.

1.7 sigstp_handler

Current process의 pid를 fgpid함수를 이용해 구한 뒤에, kill함수의 argument에 해당 pid를 음수값으로 넣어줌으로써 foreground group에 signal을 send한다.

2. What was difficult.

WNOHANG , WUNTRACED 등의 option 등이 잘 이해가지 않아 시간이 조금 오래 걸렸다. 또, waitfg에서 처음에는 wait함수를 사용했었는데, sleep을 사용해야 한다는 것이 살짝 어려웠다. 처음 eval함수는 교재에 있는 내용을 토대로 구성할 수 있어 비교적 간단했으나, 이후 추가적인 작업이 어려웠다. 특히, sigrocmask와 같이 signal을 block하고 unblock하는 것의 의미를 파악하는 것도 어려운 점이었다.

3. Something new and surprising.

모든 system call의 return값을 하나하나 다 체크해야 하는 것이 까다로웠으나 꼭 필요함을 느꼈다. Background job과 foreground job을 처리하는 과정도 흥미로웠다. 교과서에서는 Builtin command 2개만 사용했는데, 이번 과제에서는 4개를 사용하는 것을 통해, builtin command를 마음대로 조정할 수 있다는 것도 직접 깨닫게 되었다.

4. And so on.

직접 shell을 만들 수 있다는 사실 자체가 신기한 경험이었다. Signal에 대한 이해를 높일 수 있는 좋은 경험이었다. Signal handler를 이용해 signal을 직접 다루는 것이 흥미로웠다.