시스템프로그래밍 SHLAB 보고서

2015-16828 김민규

각 함수의 구현은 아래와 같이 개략적으로 설명할 수 있다.

1. eval

- 이 함수는 들어온 command line을 처리하는 중심 함수이다.

① 우선 argv를 malloc함수를 이용해 선언해준다. 길이는 이미 정의되어 있는 MAXARGS상수를 이용한다.

② parseline을 하여 cmdline을 argv에 받아온다.

③ argv를 builtin_cmd함수에 전달해 builtin command인지 알아보고, builtin command라면 builtin_cmd함수에서 작업을 완료하고 eval함수는 끝난다.

④ sigprocmask함수를 이용하여 SIGCHLD 시그널을 block 해준다

⑤ fork()함수로 자식프로세스를 만든다

⑥ 1) 자식프로세스에서는 SIGCHLD를 unblock하고, setpgid(0,0)으로 프로세스 그룹을 설정해준 뒤, execve함수로 cmdline으로 들어온 작업을 실행한다.
2) 부모프로세스에서는, 자식프로세스의 작업이 background인지 foreground인지 판별한다(pareseline의 return값으로 판별 가능하다). jobs에 그 작업을 추가하고, SIGCHLD를 unblock한다. 자식 프로세스가 부모 프로세스에서 list에 작업을 추가하기 전에 끝나서 처리되는 것을 막기 위해 signal을 block하는것이다.

⑦ 적절한 log message를 띄우고 종료한다.

2. builtin_cmd

- 이 함수는 builtin command인지 판별하고, 아니라면 0을 반환하고 맞다면 그에 맞는 행동을 한 후 1을 반환하는 함수이다.

① argv[0]이 "quit", "jobs","bg","fg" 넷 중 하나인지 살핀다. 아니라면 0을 반환한다.

② argv[0]이 "quit"이라면 exit(0)으로 프로세스를 호출한다.

③ argv[0]이 "jobs"라면 listjobs함수를 호출한다.

④ argv[0]이 “bg" 혹은 “fg”라면 do_bgfg함수를 호출한다.

⑤ 1을 반환한다.

3. do_bgfg

- 이 함수는 "bg" 또는 “fg" 커맨드가 들어왔을 때 처리하기 위한 함수이다.

① argv[1]을 살펴보고 NULL이라면 argument가 없다는 뜻이므로 적절한 에러 메시지를 출력하고 종료한다.

② argument가 pid거나 %jid꼴이 아닐 경우 적절한 에러 메시지를 출력하고 종료한다.

③ argument에 해당하는 작업을 jobs 에서 찾는다.
④ 들어온 명령어가 bg라면 jobs에서 그 작업의 상태를 ST에서 BG로 바꾸고(상태가 ST가 아니었다면 바로 종료한다) SIGCONT시그널을 보낸다.
⑤ 들어온 명령어가 fg라면 jobs에서 그 작업의 상태를 FG로 바꾸고(이미 FG였다면 바로 종료한다) SIGCONT시그널을 보낸다.

4. waitfg
- 이 함수는 자식프로세스의 pid를 인자로 받는다. 자식프로세스가 foreground job이라면 부모 프로세스로 하여금 그 job이 끝날 때까지 기다리게 해주는 함수이다.

① pid에 해당하는 job을 jobs에서 찾는다. 만약 없다면 바로 종료한다.
② pid에 해당하는 job이 존재하고, 그 job의 state가 FG라면 sleep함수를 이용하여 1초 기다린다. 이를 반복문으로 반복한다.

5. sigchld_handler
- SIGCHLD를 받았을 때 시행되는 핸들러이다.

① waitpid함수를 이용해 종료되거나 정지된 자식 프로세스의 pid를 받아온다.
② 이 pid가 jobs에 있는 pid인지 확인하고, 아니라면 바로 종료한다.
③ 할 일을 다 하고 정상적으로 종료된 자식이라면, deletejob으로 jobs에서 그 프로세스를 지운다.
④ interrupt등으로 인해 종료된 자식이라면, 적절한 메시지를 출력하고 deletejob으로 jobs에서 그 프로세스를 지운다.
⑤ stop된 자식이라면, 적절한 메시지를 출력하고 jobs에서 그 프로세스의 상태를 ST로 바꾼다.

6. sigint_handler
- SIGINT를 받았을 때 시행되는 핸들러이다.

① fgpid함수로 forward에서 시행된 프로세스의 pid를 찾는다. 없다면 바로 종료한다
② pid에 kill함수를 이용해 SIGKILL을 보낸다. 이 때 -pid를 하여서 group에 시그널을 보낸다.

7. sigint_handler
- SIGTSTP를 받았을 때 시행되는 핸들러이다.

③ fgpid함수로 forward에서 시행된 프로세스의 pid를 찾는다. 없다면 바로 종료한다
④ pid에 kill함수를 이용해 SIGTSTP을 보낸다. 이 때 -pid를 하여서 group에 시그널을 보낸다.

개인적으로 어려웠던 점

1. 처음에는 shell이 무슨 작업을 하는지에 대한 개념이 부족하여서 헤맸던 것 같다. shell이 fork하여 자식 프로세스에서 들어온 실행 명령을 처리한다는 개념을 더 잘 숙지하고 있었다면 안 헤맸을 것이다.
2. 디버깅할 때 printf를 통해 여기저기에서 궁금한 값들을 출력해 보았는데, 가끔 순서가 무시되는 듯한 출력문들이 튀어나와 당황했었다. 알고 보니 자식 프로세스와 부모 프로세스가 따로 작동하여 그런 것이었다.
3. sigchld_handler와 waitfg 두 함수에 작업을 어떻게 배분해야 할지 난감했다. 처음에는 waitfg의 반복문을 job의 state로 조건을 세울 생각을 못하고, waitpid로 반복문을 수행하려 하였으나 굉장히 꼬여서 다시 생각하다 보니 지금과 같이 구현하게 되었다.