**System Programming**

**: shell Lab**

2017-18538 황선영

1. Introduction

Process control과 signaling의 기능을 가진 simple UNIX shell program을 직접 구현한다.

2. eval: main routine that parses and interprets the command line

텍스트, 스크린샷, 전화이(가) 표시된 사진

자동 생성된 설명

eval 함수는 built-in 커맨드가 아닌 다른 커맨드가 들어올 때 처리하는 함수이다. 먼저 sigemptyset 함수와 sigaddset 함수를 이용해 새로운 signal set에 SIGCHLD를 추가한다. 그리고 sigprocmask 함수를 이용하여 parent가 child에서 오는 signal을 처리할 수 있을 때까지 block을 시켜놓는다. 그 후 fork를 하여 child 프로세스를 생성한다. Child인 경우 signal을 unblock한 후 새로운 process group id를 얻는다. 그 후 새로운 프로그램을 load하고 run한다. Parent인 경우 bg일 때와 fg일 때 하는 일이 다르다. Fg인 경우 새로운 job을 add하고 block을 푼다. 그리고 fg job이 끝날 때까지 기다린다. Bg인 경우 addjob을 하고 block을 푸는 것은 fg와 같으나 log message를 print한다는 점이 다르다.

Sigemptyset, sigaddset, sigprocmask 등의 함수들을 처음 사용해봐서 그 기능을 이해하는 데 조금 어려움을 겪었다. 하지만 사용하면서 그 기능을 잘 이해하게 되었다.

3. built-in\_cmd: recognizes and interprets the built-in commands(quit, fg, bg, jobs)

앉아있는, 테이블, 전화, 화면이(가) 표시된 사진

자동 생성된 설명

이 함수는 커맨드 창에 built-in 커맨드가 들어왔을 때 그것을 처리하는 함수이다. 빌트인 커맨드는 총 네 가지가 있다. 첫번째로 quit 커맨드가 있는데, 이 커맨드가 입력되면 실행중인 shell을 종료한다. 두번째는 jobs 커맨드이다. 이 커맨드가 입력되면 멈추거나 실행중인 모든 백그라운드 job들을 list한다. 세번째와 네번째는 bg, fg 커맨드이다. 이 커맨드가 들어오면 do\_bgfg 함수를 호출하여 각 입력에 맞는 동작을 수행한다.(자세한 동작은 do\_bgfg 함수에서 설명한다.)

4. do\_bgfg: implements the bg and fg built-in commands

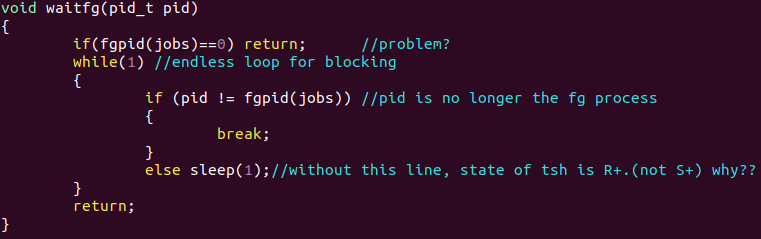
스크린샷이(가) 표시된 사진

자동 생성된 설명

do\_bgfg 함수는 built-in 커맨드인 bg와 fg를 실행하는 함수이다. 예외처리를 해주기 위해 id부분인 argv[1]이 NULL인지, pid와 jid의 형태가 맞는지 확인한다. 그 후 확인할 것은 id가 jid인지 pid인지다. id의 맨앞이 %인 경우 job id이고 myjob에 jid를 받아 유효한지 확인한다. Pid의 경우 숫자로 시작한다. 이는 앞의 예외처리에서 확인해주었기 때문에 굳이 한 번 더 확인해주지 않았다. 마찬가지로 myjob에 pid를 받고 유효한지 확인한다. 그 후 bg인지 fg인지에 따라 수행이 바뀐다. 먼저 bg인 경우 myjob의 상태를 BG으로 바꾸고 정보를 프린트했다. 그리고 그 job의 모든 그룹에 SIGCONT signal을 보낸다. Fg인 경우 myjob의 상태를 FG로 바꾸고 그 job의 모든 그룹에 SIGCONT signal을 보낸다. 그리고 myjob이 끝날 때까지 기다린다.

getjobjid, getjobpid 함수를 사용할 때 두번째 parameter에 atoi함수를 쓸 생각을 못해서 애를 먹었다. 그리고 getjobjid를 할 때 %를 제외한 숫자 부분만 받기 위해 &argv[1][1]과 같은 방법을 사용했다.

5. waitfg: waits for a foreground job to complete

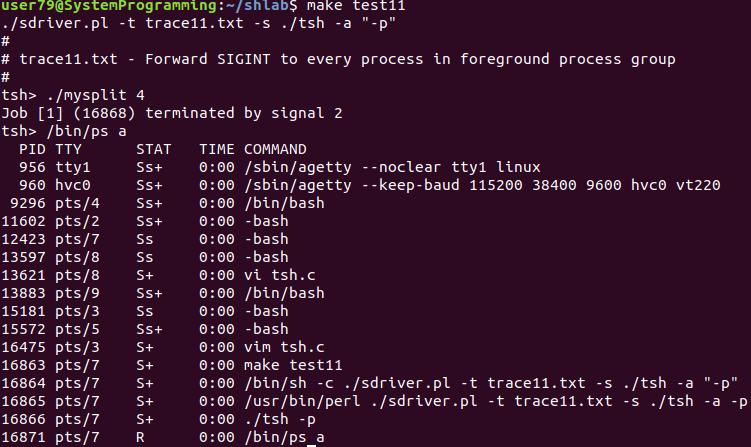


이 함수는 foreground process가 끝날 때까지 blocking하는 함수이다. 무한 루프 안에서 만약 parameter로 받은 pid가 fg가 아닐 경우 무한 루프를 break하고, fg인 경우 sleep하며 blocking해준다. 이 때 else 이후 아무 것도 쓰지 않았을 때 다음과 같은 문제가 발생했다.

전화, 테이블이(가) 표시된 사진

자동 생성된 설명

tshref에서 실행할 때와 달리 커맨드 ./tsh -p의 stat이 R+(running)으로 뜬다. 원래 S+(sleeping)이 떠야 정상이다. 이는 else 이후 아무것도 쓰지 않아서 계속 무한 루프가 돌기 때문에 발생하는 문제이다. 그래서 sleep(1);을 else 이후에 추가해주었다. 그랬더니 다음과 같이 정상적인 결과가 나왔다.



6. sigchld\_handler : catches SIGCHILD signals

스크린샷이(가) 표시된 사진

자동 생성된 설명

waitpid 함수에서 parameter인 pid는 상태를 알아보고자 하는 job의 pid이다. WNOHANG은 좀비프로세스의 경우에도 바로 리턴하고, WUNTRACED 종료되었거나 멈춘 프로세스로부터 상태를 얻을 수 있다. 리턴 값인 pid은 함수가 error를 일으킬 경우 -1이고 어떤 child도 이용되지 않았다면 0이고 정상적으로 함수가 끝났을 때만 child process의 pid이다. 고로 정상적으로 함수가 끝난 경우만 while문이 실행된다. 첫번째 if문에서는 exit하는 경우로 해당 pid의 job을 삭제한다. 두번째 if문에서는 job을 멈추는 경우다. 그래서 해당 job의 state를 ST(stop)으로 바꿔주고 어떤 signal number에 의해 멈췄는지 출력한다. 세번째 if문에서는 어떤 signal number에 의해 종료되는지 출력 후 해당 job을 삭제한다.

이 함수를 구현할 땐 waitpid함수의 parameter를 이해하는데 시간이 소요되었다.

7. sigint\_handler: catches SIGINT(ctrl-c) signals

그리기이(가) 표시된 사진

자동 생성된 설명

먼저 jobs의 pid가 유효한지(0이 아닌지) 확인한다. 받은 sig를 모든 foreground process group에 보내기 위해 kill의 첫번째 parameter에 pid 대신 -pid를 쓴다. 이 때 sig는 SIGINT이다.

8. sigtstp\_handler: catches SIGTSTP(ctrl-z) signals

그리기이(가) 표시된 사진

자동 생성된 설명

먼저 jobs의 pid가 유효한지(0이 아닌지) 확인한다. 받은 sig를 모든 foreground process group에 보내기 위해 kill의 첫번째 parameter에 pid 대신 -pid를 쓴다. 이 때 sig는 SIGTSTP이다.

9. Conclusion

이번 lab을 통하여 어떻게 process를 control하고 signal을 handling하는지 간단하게나마 알 수 있었다.