Shell lab Report

2019-12333 정윤서

1. Explanation of implement

참고) 기본적으로 return value를 체크하기 위해 system call을 래핑한 함수들을 구현하였다.

int builtin_cmd(char** argv)

- 만약 argv[0]이 quit, bg, fg, jobs 중에 하나라면 해당 명령을 실행하고 1을 리턴한다. 그렇지 않다면 0을 리턴한다. Quit을 제외한 나머지 빌트인 명령들은 tsh.c에서 직접 구현하였으므로 해당 함수를 호출함으로써 실행한다. 이때 명령이 bg/fg인 동시에 argv[1]이 존재하지 않거나, 숫자가 아니거나, %로 시작하지도 않는다면 에러 로그를 출력하고 1을 리턴한다.

void eval(char* cmdline)

- parseline 함수를 통해 해당 job를 background에서 돌릴 것인지 foreground에서 돌릴 것인지에 대한 정보를 받아오며 동시에 버퍼에 담긴 command line을 argv에 파싱해서 담는다. 이때 argv[0]에 아무것도 담겨있지 않는다면 즉시 리턴하고, 그렇지 않은 경우엔 다시 builtin_cmd 함수를 통해 argv[0]의 빌트인 여부를 확인한다. 빌트인일 경우에는 builtin_cmd에서 바로 명령이 실행되기에 즉시 리턴하고, 빌트인이 아닌 경우에는 child를 fork해서 실행한다. 한편, child의 execve가 실패한 경우 없는 명령(같은 dir에 실행파일이 존재하지 않음)인 것이므로 "Command not found"를 출력하고 child process를 종료한다. Parent process(쉘)은 fork한 job이 background job인 경우에는 로그를 출력하고 리턴하며, foreground job인 경우 waitfg 함수를 호출한다. 한편, 쉘이 job을 job list에 추가하기 전에 job이 종료되어 커널이 sigchld 시그널을 보내게 되고, 이로 인해 쉘 입장에서 job list에 존재하지 않는 job을 delete하게 되어 버리면(아무 일도 발생 x) handler에서 다시 원래 프로세스로 돌아왔을 때 존재하지 않는 job을 리스트에 추가하려고 하게 될 것이다. 따라서 fork 이전에 sigprocmask로 sigchld를 block하고 job을 리스트에 추가한 후 블로킹을 풀어주어(child process도 fork 이후 inherit된 블로킹 복구) 문제를 해결하였다. 또한 setpgid(0,0)을 통해 fork된 프로세스를 쉘이 속한 foreground process group에서 분리함으로써 추후에 foreground job으로 보내진 sigint, sigtstp를 쉘에서 우선적으로 핸들링할 수 있도록 하였다.

void do_bgfg(char** argv)

- 우선 갖가지 에러 사항을 점검한다. Builtin_cmd 함수에서 체크한 것 외에도 argv[1]에 해당하는 job이나 process가 없는 경우 로그를 출력하고 리턴한다. 만약 bg 명령이라면 job의 state를 BG로 바꾸고 kill 함수를 통해 stop 상태일 process들(해당 job과 같은 group id)에게 SIGCONT 시그널을 전달함으로써 백그라운드 프로세스로 다시 돌아가게끔 한다.

Fg 명령이라면 마찬가지로 job의 state를 FG로 바꾸고 kill을 통해 process들에게 시그널을 전달한다. 한편, 이 경우는 foreground로 돌아가게끔 해야 하므로 waitfg 함수를 사용하여 쉘이 이 job을 기다리도록 한다.

void waitfg(pid_t pid)

- 이미 sigchld_handler가 reaping의 역할을 수행하므로 waitfg는 쉘이 foreground process를 끝날 때까지(혹은 멈출 때까지) sleep(1)을 반복하며 다른 작업을 하지 않는다.

void sigchld_handler(int sig)

- Shell의 입장에서 종료되거나 멈춘 모든 child process들에 따른 sigchld 시그널을 핸들링한다. While문으로 waitpid(-1,&status, WNOHANG)>0을 감싸서 현재 종료된 모든 child process들에 대해 루프를 돌리며 시그널로 종료된 것이라면 로그를 프린트한 뒤 job list에서 제거하고, 그렇지 않고 정상적으로 종료되었다면 바로 job list에서 제거해준다. 또 while문으로 waitpid(-1, &status, WUNTRACED | WNOHANG)>0을 감싸서 현재 정지된 모든 child process들에 대해 루프를 돌리며 state를 ST로 바꾸고 로그를 프린트해준다.

void sigint_handler(int sig)

- Ctrl-c를 통해 쉘(foreground process group의 유일한 process)에게 전달된 SIGINT를 foreground job에게 전달한다. 만약 현재 foreground job이 없다면 그대로 리턴하고, 그렇지 않다면 kill을 통해 foreground job이 속한 그룹의 모든 process들에게 SIGINT를 전달한다.

void sigtstp_handler(int sig)

- Ctrl-z을 통해 쉘에게 전달된 SIGTSTP를 foreground job에게 전달한다. Sigint handler와 동일하게 작동하며, 시그널이 전달될 job의 state을 ST로 바꾸어주는 것만 다르다.

2. What was difficult

- Trace16.txt에서는 터미널(ctrl-c, ctrl-z)이 아닌 실행되고 있는 child process에서 자기 자신에게 SIGINT, SIGTSTP 시그널을 보낸다. 이때 기존에 구현한 sigint_handler와 sigtstp_handler로는 시그널을 핸들링할 수 없어서 어려움을 겪었다. 이 문제는 child process로 직접 전달된 시그널로 인해 프로세스가 죽거나 멈춘 경우 쉘에서 이를 sigchld_handler의 waitpid를 통해 핸들링할 수 있도록 하여 해결하였다.

3. Something new and surprising

- Setpgid(0,0)을 통해 group id를 현재 pid로 설정할 수 있다는 것은 이론적으로 알고 있었는데, fork된 child process를 foreground process group에서 분리하기 위해 쓰일 수 있다는 사실을 알게 되어 새로웠다.