2019-2 학기 CSE4009(시스템프로그래밍): 서지원 교수님

Assignment 3 Tiny Shell 구현 Report: 2018008904 이성진

이번 과제는 기존에 수업 시간에 배운 process control과 signal에 관련된 내용을 바탕으로 간단한 Unix shell을 구현하는 것이다. 구현된 shell은 여러 job들을 control할 수 있는 구조를 갖추고 있어야 하며, background와 foreground를 구분하여 프로세스를 실행할 수 있어야 한다.

과제 skeleton에서는 shell을 구현하기 위한 여러 보조 함수들과 main함수들이 이미 정의되어 있는데, 요구 사항을 만족시키기 위해서 비어있는 함수들인 **eval**, **builtin_cmd**, 그리고 **waitfg**를 다음과 같이 정의했다.

1. `void eval(char *cmdline)`

함수 **eval**은 main에서 입력 받은 한 줄의 command를 해석하고 실행하는 기능을 담당하는 함수이다. Parameter로 입력된 command 문자열이 사용된다.

이 함수가 호출될 경우, 입력된 명령어가 built-in command(quit, jobs, bg, fg)에 해당될 경우 즉시 해당 작업을 실행하고 return한다. 만약 그렇지 않을 경우, 새로운 child process를 folk하여 해당 child process가 요청된 명령을 실행하도록 한다. 만약 이때 command line의 맨 끝에 '&' 문자가 존재할 경우, 요청된 작업을 background에서 실행하도록 하며, 그렇지 않을 경우 foreground에서 실행하도록 해야 한다.

실제 함수 내부에서는 우선 cmdline을 제공된 parseline함수를 활용하여 argv에 파싱하고 background인지 foreground인지 여부를 확인하여 bg 변수에 저장하게 된다. 만약 argv[0]이 NULL값을 갖는 경우, 즉 empty line이 입력된 경우에는 아무 작업도 진행하지 않고 return한다. 이후 builtin_cmd(argv)의 return value를 확인하는 것을 통해 built-in command가 요청되었는지 확인한다. 만약 return value가 true(!=0)일 경우, builtin_cmd를 호출할 때 이미 해당 작업이 실행이 완료된 상태이기 때문에 추가 작업 없이 함수를 종료한다.

만약 요청된 command가 built-in command가 아닐 경우, 새로운 child process를 folk한다. 이후 folk로 인해 생성된 child process는 execve함수를 이용하여 입력된 명령을 실행하게 되며, 오류가 발생하면 오류 메시지를 출력하고 exit(1)을 호출하게 된다. Parent process는 addjob 함수를 호출하여 job list에 현재 요청된 작업을 추가하게 된다. 이 과정에서 우선 fork가 호출되기 전에 empty signal set을 생성한 후 SIGCHILD signal을 set에 추가하고, sigprocmask를 호출하여 SIGCHILD signal을 block해야 한다. 이후 folk가 호출되면 child는 새로운 프로그램을 실행하기 전 sigprocmask를 호출하여 SIGCHLD signal을 unblock해주게 되고, parent는 addjob이 호출된 후에 sigprocmask를 이용하여 signal을 unblock하게 된다. 이를 통해

parent가 addjob에 새로운 job을 추가하기 전에 SIGCHLD signal이 전달되어 child가 reap되고 job list에서 제거되는 것을 막을 수 있게 된다.

Child process를 실행하고 job list에 추가한 후에는, parseline의 return value를 저장해놓은 bg가 0이 아닐 경우 foreground에서 실행되는 job이므로 child process가 종료될 때까지 기다리도록 waitfg를 호출한다. 만약 bg가 0일 경우 background에서 실행되는 job이므로 wait을 진행하지 않고 child process의 정보를 출력한 후 함수를 종료한다.

2. int builtin_cmd(char **argv)

함수 **builtin_cmd**는 eval 실행 도중 입력된 command가 built-in command인지 확인하고, 그럴 경우 즉시 실행하는 기능을 수행한다. Parameter로 command line을 파싱한 argv를 이용하는데, 함수가 호출되면 첫 번째 argument인 argv[0]을 strcmp함수를 이용하여 "jobs", "quit", "bg", 그리고 "fg"중 일치하는 것이 있는지 확인한다.

만약 argv[0]이 위의 built-in command중 하나와 일치할 경우, 해당 command에 해당하는 미리 정의되어 있던 함수를 호출한다. "jobs"인 경우에는 현재 실행중인 job들을 출력해주는 함수인 listjobs(jobs), "quit"인 경우에는 shell 자체가 종료되도록 exit(0), "bg" 또는 "fg"일 경우 argv를 parameter로 받아 bg 또는 fg에 맞는 적절한 작업을 수행해주는 do_bgfg(argv)를 호출한 후 true(1)을 return하여 이후 eval 함수가 추가 작업을 진행하지 않고 종료되도록 한다.

argv[0]이 위의 built-in command 중 어떤 것과도 일치하지 않는 경우, 요청된 명령은 built-in command가 아니므로 builtin_cmd는 아무 작업도 진행하지 않게 되고, 이후 eval에서 child process를 folk하여 실행시킬 수 있도록 false(0)을 return하게 된다.

3. void waitfg(pid_t pid)

함수 **waitfg**는 eval 함수 실행 도중 child process가 foreground에서 실행하도록 요청되어 parent process가 해당 child가 더 이상 foreground process가 아닐 때까지 기다리게 하는 기능을 한다. 우선 getjobpid 함수를 활용하여 child process의 pid를 바탕으로 실행중인 job의 정보가 담겨 있는 구조체 job을 확보한다. 만약 job이 NULL일 경우, 즉 현재 해당 pid를 가지는 프로세스가 실행 중이 아닐 경우 곧바로 return한다. 만약 그렇지 않은 경우, 해당 job의 상태가 FG 상태인지 확인하여 FG 상태가 아닐 때까지 반복해서 sleep(1)을 호출하여 기다리도록 한다. 만약 child job이 FG 상태가 아닐 경우, 즉 해당 job이 BG(background에서 실행) 또는 ST(실행 중 정지) 상태가 되는 경우에는 곧바로 return하여 waitfg가 종료된다.