CSED211: Lab 10 (due Dec. 11)

손량(20220323)

Last compiled on: Monday 11th December, 2023, 13:31

1 개요

수업 시간에 배운 child process management, signal 등을 활용하여 UNIX 시스템에서 다른 프로그램을 실행하고, 실행 상태를 관리하기 위한 프로그램인 shell을 작성하여 본다.

2 프로그램의 작성

과제에서 주어진 `tsh.c`에 기본적인 기능과 함수는 이미 구현되어 있기 때문에, 요구 사항을 구현하기 위해서는 다음 함수들만 구현하면 되었다.

1. `eval`
2. `builtin_cmd`
3. `do_bgfg`
4. `waitfg`
5. `sigchld_handler`
6. `sigtstp_handler`
7. `sigint_handler`

2.1 `eval` – Running Command Line

`eval` 함수에서는 사용자가 입력한 command line이 `cmdline` argument로 주어질 때, 이를 파싱한 다음 명령어에 맞게 동작을 수행하는 함수이다. 과제에서 이미 주어진 `parseline` 함수를 이용하여 commmand line을 파싱할 수 있었고, 만약 빈 줄이 주어지는 경우에는 더 이상 실행하지 않고 종료하여 다시 prompt가 출력되게 하였다. 이후, `builtin_cmd` 함수를 사용하여 built-in command가 주어진 상황에서는 명령어를 수행하고 종료한다. 이외의 경우에는 외부 프로그램을 실행해야 한다.

외부 프로그램을 실행하는 경우 `eval` 함수에서는 `fork` 함수를 사용하여 child process를 생성한다. Child process의 control flow에서는 `setpgid` 함수를 사용해 자신이 leader가 되는 process group을 생성하고, `execve` 함수를 사용하여 사용자가 입력한 command line에 따라 프로그램을 실행한다. 한편 parent process의 control flow에서는 `jobs` 배열에 `addjob` 함수를 사용해 사용자가 실행한 프로세스에 해당하는 job을 실행하고, 만약 foreground job인 경우에는 `waitfg` 함수를 활용, 프로그램이 종료하거나 사용자가 control-c 혹은 control-z 키를 눌러 시그널을 보낼 때까지 기다리게 한다.

Child process와 parent process의 분기가 나누어지는 부분에서, job list에 추가하기 전에 child process가 종료하는 등의 race condition이 발생할 가능성이 있다. 이를 방지하기 위하여 `sigprocmask` 함수를 사용해 job list에 대한 synchronization을 할 수 있도록 job list를 변경하는 동안은 `SIGCHLD` signal을 block하였다.

2.2 `builtin_cmd` – Built-in Command

`builtin_cmd` 함수에서는 `quit`, `jobs`, `bg`, `fg`와 같은 `tsh`의 built-in command를 처리한다. `quit` 명령어의 경우에는 별다른 처리 없이 `exit` 함수를 사용해 shell을 종료한다.

`jobs` 명령어의 경우 미리 구현된 `listjobs` 함수를 활용하여 현재 job list를 출력한다.

`bg` 혹은 `fg` 명령어의 경우 `do_bgfg` 함수를 사용하여 background job 혹은 foreground job으로 stopped 상태인 job을 실행하도록 하였다.

2.3 `do_bgfg` – Handling `bg` and `fg` Command

`do_bgfg` 함수에서는 주어진 stopped job을 background 혹은 foreground 상태로 실행한다. 우선 command line argument로 주어진 job 번호를 파싱해야 하는데, 이것에는 `strtol` 함수를 사용하였다. `strtol` 함수의 `endptr` 인자와 `errno`를 활용하면 주어진 문자열을 숫자로 변환하지 못하는 경우도 알아낼 수 있기 때문에, 미리 문자열을 검사하는 코드를 작성할 필요가 없다는 장점이 있다.

Command line argument를 파싱한 후에는 주어진 job id 혹은 PID에 해당하는 job의 정보를 job list에서 가져오고, job의 process group에 `SIGCONT` signal을 `kill` 함수로 보내 프로세스가 다시 실행되도록 만들었다. Foreground job을 실행하는 경우에는 `waitfg` 함수로 프로세스가 종료하거나 사용자가 시그널을 보낼 때까지 기다리도록 하였다.

2.4 `waitfg` – Wait for Foreground Process

`waitfg` 함수에서는 주어진 PID를 가지는 프로세스가 종료하거나 시그널을 받을 때까지 기다린다. Writeup에서 제시된 대로, foreground job이 실행 중이라면 `sleep`을 반복적으로 실행하여 busy wait 하도록 구현하였다.

2.5 `sigchld_handler` – Handling `SIGCHLD`

Child process가 종료하거나 시그널을 통해 실행을 종단한다면 parent process는 `SIGCHLD` signal을 받는다. `sigchld_handler` 함수는 이 `SIGCHLD` signal을 처리한다. 이 signal handler는 단순히 `signal` 함수로 등록되어 있기 때문에, handler에서는 child process가 종료하거나 실행을 중단했다는 사실만 알 수 있고, 어떤 process인지는 알 수 없기 때문에 `waitpid` 함수의 `pid` argument에 -1을 넣어 자식 프로세스 중 아무거나 reaping하는 과정을 반복하도록 만들었다. Reaping할 프로세스가 없는 경우 기다리는 대신 반복문을 종료하기 위해 `WNOHANG` flag를 사용하였고, child process가 실행을 중단하는 경우도 처리하기 위해 `WUNTRACED` flag도 사용하였다. 대상이 되는 child process가 존재할 경우, 프로세스가 종료했을 때는 job list에서 프로세스를 삭제하면서, 필요한 경우 종료 메시지를 출력하도록 했다. 프로세스가 중단된 경우에는 job list에서 프로세스의 상태를 바꾸도록 처리하였다.

2.6 `sigtstp_handler` – Handling `SIGTSTP`

`sigtstp_handler`는 사용자가 터미널에서 control-z 키를 눌렀을 때 발생하는 `SIGTSTP` signal을 처리한다. 만약 foreground job이 실행 중이지 않다면 프롬프트에서 control-z를 누른 것이므로 단순히 무시하도록 하였고, foreground job이 실행 중인 경우에는 `kill` 함수를 통해 해당되는 job의 process group에 `SIGTSTP` signal을 보내도록 구현했다.

2.7 `sigint_handler` – Handling `SIGINT`

`sigint_handler`는 사용자가 터미널에서 control-c 키를 눌렀을 때 발생하는 `SIGINT` signal을 처리한다. 이 함수의 구현은 `sigchld_handler`와 비슷하게 foreground job이 실행 중이라면 `kill` 함수를 통해 해당 process group에 `SIGINT` signal을 보내도록 하였다.

3 결론 및 제언

이로써 간단한 shell을 구현할 수 있었고, `make testXX` 테스트를 모두 통과함을 확인할 수 있었다. 이미 주어진 함수들을 최대한 활용하기 위해 이 구현에서는 할 수 없었지만, 기회가 된다면 `waitfg` 함수의 busy waiting을 개선하는 것도 좋을 것이다. 많은 shell에서 채택하고 있는 것처럼 `sigsuspend` 함수 등을 활용해 flag가 세팅될 때까지만 wait하거나, per-process timer 등을 활용하는 방법 등이 있을 것이다.