2018년 2학기

컴퓨터 SW시스템 개론

CSED211

Lab Report #10

담당교수 : 김종

학번 : 20160074

학과 : 화학공학과

이름 : 고진민

POVIS ID : eric9709

--------------------------------------------------------------------------------------------------------------

<명예서약>

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

--------------------------------------------------------------------------------------------------------------

1. **구동 환경**

* 학교 프로그래밍 서버

1. **함수별 설명**

* void eval(char \*cmdline)

이 함수가 주된 처리를 하는 함수라 생각하면 된다. Eval 함수는 사용자가 프로그램을 실행한 후 입력한 command line을 분석하는 함수이다. 만약 built-in command를 입력받은 경우는 대응하는 작업을 즉시 실행하고, 그렇지 않은 경우는 child process를 fork한 뒤 child에서 작업을 수행한다. foreground에서 작업이 수행 중이면 terminate할때까지 기다린 후 return한다.

먼저 변수를 선언한 뒤, parseline 함수를 통해 background에서 수행하는 명령어(&)가 있는지 확인한다. parseline 함수를 사용하면 자동으로 cmdline으로 입력된 값을 argv로 바꿔준다. 이후 builtin\_cmd 를 if 조건문에서 호출해 builtin command인 경우는 바로 작업을 수행하고 아니면 if문 내부로 진입한다. If문 내부에서는 먼저 SIGCHLD에 대한 mask를 생성해 해당 신호를 block한다. 이후 fork를 진행해 child process를 생성하고, 이 반환값을 pid로 저장해 child process를 구분할 수 있게 한다. if문으로 pid 변수를 확인해 0인 경우(=child인 경우)에는 mask를 unblock하고 setpgid를 이용해 group id를 child의 pid로 바꾼다. 이후 execvp를 이용해 입력한 command가 올바른지 확인한다. Parent인 경우는 fg와 bg인 경우로 다시 나눈다. 기본적으로 두 작업 모두 addjob 함수를 이용해 list에 넣고, mask를 unblock한다. Fg인 경우는 이후 waitfg를 이용해 fg에서 termination 되는 것을 기다리고, bg인 경우는 해당하는 데이터를 출력한다.

* int builtin\_cmd(char \*\*argv)

이 함수는 built-in command를 입력받은 경우 해당하는 작업 수행 후 1을 return, 아닌 경우는 0을 return한다.

quit을 입력받은 경우는 바로 프로그램을 종료한다. jobs를 입력받은 경우는 listjobs 함수를 호출해 job list를 출력하고 1을 반환한다. &을 입력받은 경우는 bg에서 시행하는 것이므로 1을 반환한다. Bg나fg를 입력받은 경우는 do\_bgfg 함수를 호출하고 1을 반환한다. 이상의 경우가 아니면 built-in command가 아니므로 0을 반환한다.

* void do\_bgfg(char \*\*argv)

이 함수는 built-in command로 bg나 fg를 입력받은 경우, 더 세부적인 작업을 수행하도록 하는 함수이다. 먼저 입력값이 올바른지 판단한 후 올바르다면 해당하는 작업을 수행한다.

먼저 변수를 선언한다. argv[0]은 bg나 fg이므로, 이후 입력된 부분은 argv[1]에 있을 것이다. 이 값이 NULL인 경우는 pid나 jid가 입력되지 않았다는 의미이다. 만약 fg/bg 이후 입력값의 첫 번째 문자가 숫자가 아니면서 jid를 의미하는 %도 아닌 경우는 입력값 자체가 올바르지 않은 경우이다. 단어가 숫자인지 판단하는 방법은 함수 isdigit을 사용했다. 만약 fg/bg 이후 입력값의 첫 번째 문자가 %라면, 뒤에 오는 단어가 숫자라고 생각한다면 jid를 입력받은 것이다. atoi 함수를 사용해 jid를 입력받고, getjobjid 함수로 jid에 해당하는 작업을 찾는다. 만약 없다면, 오류 메시지를 출력하고 return한다. 위에서 말한 경우가 모두 아닌 경우라면 pid를 입력받은 경우이므로 jid와 거의 같지만 pid로 대응되는 작업을 바꿔서 수행한다. 이 작업들은 fg/bg 관계없이 공통으로 진행하는 작업이고, fg와 bg에 대해 다르게 작업을 수행하는 것은 다음과 같다. 먼저 fg의 경우는 job의 state를 FG로 변경한 후 SIGCONT를 전체 group에 보내 멈췄던 작업을 재시작하고 waitfg 함수를 호출해 fg가 아닌 상황이 될 때까지 루프를 돌린다. bg의 경우는 먼저 메시지를 출력한 후 state를 BG로 바꾼 후, SIGCONT를 보내 멈췄던 작업을 background에서 돌린다.

* void waitfg(pid\_t pid)

이 함수는 pid가 foreground에서 돌아가지 않을 때까지 붙잡아 두는 역할을 한다. pid를 입력받으면 getjobpid 함수를 통해서 대응하는 job을 확인하고 이 job이 NULL이 아닌 경우, while 문 내에서 pid가 fgpid를 통해 받은 fg에서의 pid와 일치하는지 확인해 달라지기 전까지 무한루프를 돌리고 달라지면 루프를 나가게 한다.

* void sigint\_handler(int sig) / void sigtstp\_handler(int sig)

이 두 함수는 각각 SIGINT와 SIGSTP 신호를 포착했을 때 실행되는 함수이다. 이 함수들 에서는 포착한 신호를 foreground에서 돌아가고 있는 작업이 있다면 그 작업의 그룹 전체에 신호를 보낸다. 이 때문에 pid라는 변수에 fgpid 함수를 이용해 foreground의 pid를 입력받고, NULL이 아니라면 해당 pid를 가진 그룹 전체에 signal을 보낸다.

* void sigchld\_handler(int sig)

이 함수는 SIGCHLD 신호를 포착했을 때 실행된다. SIGCHLD는 child process가 멈추거나 종료된 경우에 발생하는 신호이기 때문에 해당하는 경우에 대해 나눠 처리를 해주어야 한다. 먼저 일반적인 종료의 경우는 매크로 WIFEXTED로 확인, 이 경우는 해당하는 pid의 작업을 deletejob 함수로 작업에서 제거한다. 만약 다른 이유에 의해 terminated된 경우는 WIFSIGNALED로 확인하고 어떤 signal에 의해 중지됐는지 출력한 후 같은 방법으로 job list에서 제거한다. Stop 된 경우는 WIFSTOPPED로 확인 가능하며 이 경우 해당 작업의 state를 ST로 변경 후 메시지를 출력한다.