<SPRING-OS-ASSIGNMENT 2 보고서>

소프트웨어학과 201420997 최지원

이번 과제는 저번에 구현했던 mysh에서 백그라운드 실행, 파이프 구현, 신호처리, 프로세스 생성을 구현하였다.

제일 먼저 구현한 것은 Signal Handler였다. 과제에서 요구한 ^C 와 ^Z를 처리이다. ^C는 프로세스를 종료시키는 것이고, ^Z는 프로세스를 중단시키는 기능을 한다. 원래는 mysh에서 ^C를 누르면 종료되지만 이번 과제에서는 그 종료를 막도록 요구하였다. 사실 백그라운드 작업을 같이 했을 경우 ^C를 누르면 백그라운드는 종료되어서는 안되지만, 그것을 해결하지 못하였다. 제대로 구현하였다면 백그라운드 실행한 상태에서 ^C 와 ^Z을 입력하면 아무 일도 일어나지 않을 것이고, fg를 누르고 난 후 ^C 와 ^Z를 누르면 해당 프로세스의 종료 또는 중단이 발생할 것이다.

두번째로 구현한 것은 Process Creation이였다. 이는 fork()함수를 이용하여 생성하였는데 과제에서는 어디서 fork를 해야하는지 명시되어있지 않았다. 그래서 임의로 background를 실행할 때 생성하도록 코드를 작성하였다. 또한 파이프를 구현할 때 각 소켓에 대해서도 fork를 사용하였다. 사실 백그라운드에 대한 fork()는 프로세스를 생성하는 것이었고, 소켓에 대한 fork는 스레드를 생성하는 목적이었다. 이 fork함수는 return value로 자식의 pid_num를 반환하고 이를 이용하여 부모 프로세스와 자식 프로세스를 구분할 수 있었다.

세번째로 구현한 것은 백그라운드 실행이었다. 쉘에서 뒤에 '&'를 붙여주면 그에 대한 프로세스를 백그라운드에서 실행시키도록 하게 해주는 것이다. 사실 실제로 백그라운드에서 돌아가는 것이 아닌 그렇게 보이도록 하는 것이다. 그래서 백그라운드 프로세스를 실행시켜도 pwd나 cd를 입력하면 정상 실행이 된다. 또한 fg를 입력하면 백그라운드 프로세스가 포어그라운드로 나오게 되고 이 때 pwd나 cd를 입력하면 아무 일도 일어나지 않는다. 오직 ^C와 ^Z만 작동할 뿐이다. 이는 본인의 test 실행파일에 pwd나 cd에 대한 구현이 없기 때문이다. 이 test 실행 파일은 5초에 한번씩 "background"라는 구문을 출력시키도록 하였다.

마지막으로 구현한 것은 PIPE를 통한 IPC통신이였다. PIPE는 두 프로세스간 통신을 도와주게 되는데 만약 a | b 를 입력하면 a 의 표준 출력이 b의 표준 입력에 연결되도록 한다. 과제에서는 이를 도메인 소켓을 이용하여 구현하도록 요구하였다. 소켓에 대한 코드는 참고 URL에서 따왔고 파이프는 인터넷에 공유되고 있는 코드를 참고하였다. | 연산자를 쓰게 되면 서버 소켓과 클라이언트 소켓이 생성된다. 소켓을 생성할 때에는 멀티 스레드를 이용하도록 과제에서 요구하였다. 그래서 pthread를 이용하여 스레드를 생성하도록 코드를 작성하였다. Dup는 파이프의 입력 식별자를 표준 입력으로 복사한다. dup2는 close작업과 식별자 복사 작업을 한번에 하게 된다. 이를 이용하여 두 소켓을 연결하였고, IPC가 되는 것처럼 보일 수 있었다.

이번 과제에서 파이프가 가장 어려운 부분임이 분명하지만, 이를 구현한 비슷한 코드들이 많기에 어느정도 구현은 한 것 같다. 사실 제대로 구현한 것인지는 잘 모른다. 그러나 가장 쉽다고 생각한 신호처리가 더 어려웠던 것 같다. 백그라운드 프로세스를 종료시켜서는 안되는데 이를 구현하지 못한 점이 너무 아쉬웠다. 이번 과제를 통해서 제대로 알게 된 것은 백그라운드 실행에 대한 개념이었다. 처음에는 fg를 입력하면 foreground에 있던 프로세스와 background 프로세스의 위치가 바뀌는 것이라 생각하였는데 fg를 입력할 수 있다는 자체가 foreground에서 실행중인 프로세스가 없다는 것도 알았다.

이번 과제도 정말 어려웠다. 다음 과제가 벌써 두려울 뿐이다.