운영체제 과제 myshadv 보고서

201720720 조민재

이번 과제는 저번 과제의 연장선으로, 저번에 구현한 mysh에 몇 가지 기능을 추가한 과제였다.

제일 먼저 한 것은 Background processing이었다. 만약 커맨드에 &가 있는지 플래그를 세워서 검사하고, 만약 있다면 fork를 하되, wait를 불러주지 않았다. 이렇게 하면 백그라운드에 계속 돌고 있고, 다른 커맨드를 받아 실행할 수 있게 된다. 다만 wait를 하지 않으면 백그라운드로 보낸 프로세스가 좀비 프로세스가 되는 현상이 발생하여, 부모가 이를 무시하는 SIG_IGN를 사용했더니 좀비 프로세스가 되지 않고 잘 동작하였다. Fg를 통해 foreground로 불러오는 것은 간단하게 wait를 통해 구현할 수 있었다.

다음으로 signal handling을 구현했는데, 맨 처음 kill 시그널은, 만약 커맨드에 kill이 입력되었다면 해당 프로세스를 죽이는 kill 시그널을 보내는 식으로 구현하였다. 컨트롤 C와 컨트롤 Z를 막는 것은 역시 SIG_IGN를 사용하여, 시그널 발생시에도 종료되거나 멈추지 않도록 하였다. 문제는 zombie alert였는데, 좀비 프로세스가 있다는 신호를 받고 프로그램이 바로 종료되기 때문이었다. 그러나 우리가 구현해야 하는 것은 '종료 시'에 zombie alert를 발생시켜야 했으므로, 이 방법으로 구현하면 안되었다. 그래서 sigaction을 사용하려 하였으나, 이것은 makefile을 gnu로 바꿀 때에만 오류가 없고, 주어진 c99에서는 컴파일 오류가 발생하였다. Makefile을 수정해도 되는지에 대한 것이 확실하지 않아서, 결국 좀비가 있으면 zombie alert를 보내고 wait를 불러주는 방식으로 구현하였다. Signal handlers 항목이라 시그널로 구현하려 했으나 도무지 다른 방법이 생각나지 않아 어쩔 수 없이 이렇게 구현하게 되었다.

마지막으로 파이프는 구현하지 못했다. Uds를 통해 구현하라고 하였는데, 이는 프로세스들끼리 파일을 통한 통신이기 때문에 파일 디스크립터를 생성하여 파일을 넘겨줘야 한다고 생각했다. 그러나 우리가 해야 하는 것은 커맨드를 실행한 후, 즉 exec를 실행한 결과를 파일 디스크립터에 넘겨줘야 하는데 그것을 어떻게 구현할지 생각이 나지 않았다. 그냥 실행 파일의 결과라면 "./mysh"이라고 넣어주면 되는데, exec를 통해 실행한 것은 어떻게 넣어주어야 하는지 고민해 보고 찾아보았지만 알 수 없었다.

Path resolution은 저번 과제에서 이미 구현하였다.

이번 과제에서는 여러 시그널에 대한 동작과, 그것을 처리하는 함수에 대해서 배우게 되었고, 백그라운드에서 어떻게 프로세스가 동작을 하는지 알게 되었다. 파이프 또한 어떤 개념이고, 어떻게 구현되는 것인지는 조금 이해했으나, 결국 구현하지 못하여 아쉬움이 많이 남았다.

다음 과제에서는 좀더 검색하면 자료가 많이 나오는 것을 내주셨으면 좋겠다. 파이프 함수를 쓰지 않고 구현하는 것은 아무리 찾아보아도 예시가 없어, 이해하는데 꽤 오랜 시간이 걸렸기 때문이다. 그리고 과제 기한도 조금 늘려 주셨으면 좋겠다.