PA2 보고서

시스템프로그래밍실습

2018312280 이상수

Executable:

첫 fork 에서 setpgid(pid,1)로 현존 1그룹으로 바꾸었고 부모와 달랐습니다.

Ls, man, grep, sort, awk, bc 는 execvp 로 구현하였습니다.

Head, tail, cat, cp, mv, rm, pwd 는 realpath로

(Make를 실행 후 make로 생성된 head, tail등 각 바이너리 파일이 원래 경로에 그대로 존재한다는 가정하에 ./pa2 를 실행하면 pa2에서 main함수에서 위 바이너리파일들의 경로를 가져와서 전역변수로 저장을 합니다. 만약 make를 실행 후 바이너리 파일들의 경로를 바꿔 버리시면 제 코드가 의도대로 실행이 안될 확률이 높습니다.)

각 파일 실행파일의 경로를 구해서 저장해두고, 필요시 호출하여 execv 로 실행시켰습니다.

각 파일은 main하나안에서 argv로 인수를 받는 방식으로 다 구성하였습니다.

Builtin_cmd는 minishell.c 의 main 내에 구현하여 거기서 처리하도록 만들었습니다.

Main을 지나서 일어나는 대부분의 동작은 builtin을 제외하고 첫 fork를 eval에서 한 다음 실행됩니다. 최초의 fork 후 grouppid를 최초 fork 된 자식의 pid에 맞춰주어 맨 첫 minishell의 pid와 다르게 하였습니다.

Pipe 와 redirection은 저희가 배운대로 dup2를 활용하였고, 특정 문자(<,>,>>)를 기준으로 각 문자를 나눠 명령을 수행하거나 파일을 읽었습니다.

특히 pipe는 일단 받은 문자에 | 이 포함되어 있는지 판단하고, 문자를 |을 기준으로 여러 string으로 나눴습니다. 처음 string 에는 < 가있는지 판단하여 redirection을 해주었고, 그 다음 for 문으로 나눈 문자중 첫,마지막을 제외한 나머지 개수만큼 fork 를 해주어 문장들을 처리해주고, 마지막 문장은 > 나 >>가있는지 판단후 redirection을 해주었습니다.

Sigint, sigtstp는 int는 sit_ign를 하였고, stp는 main에서 받아서 받으면 wait을 실행해 child가 나오면 그 child의 그룹에 sigkill을 날렸습니다.

Background는 waitpid(WNOHANG | WUNTRADEP) 를 활용하여 만들었고,

이렇게 했을 때 출력값이 background의 출력값과 minishell의 출력이 순서가 바뀌어 다음 커맨드 입력에 mini>가 위로가 있는 경우가 있었습니다.

예) ls &

Mini> ~ ~ ~ directory files

(input next command)