**Project 2**

**리눅스 프로그래밍 004분반**

**컴퓨터공학과 12181611 박재민**

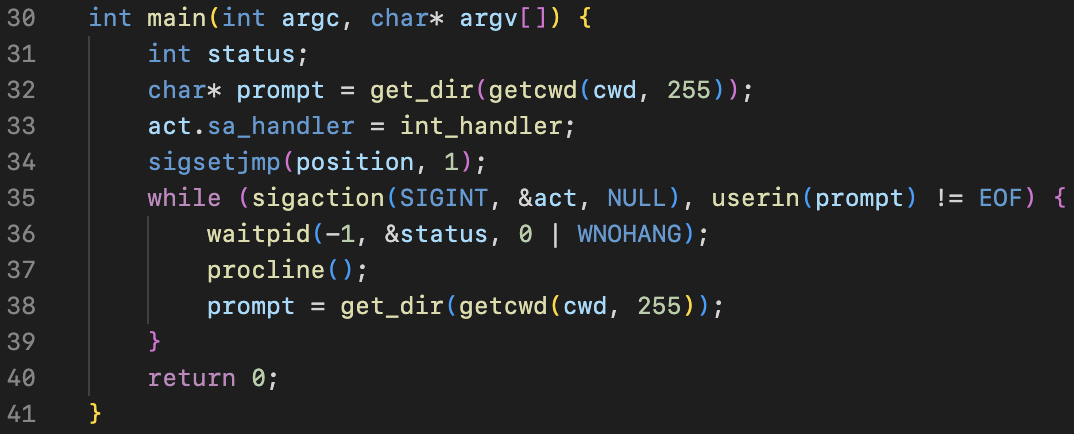
project1에서 가독성을 위해 코드를 일부 수정했지만 기능은 정상 작동

1. **&의 문제점 해결**

project1에서 이미 해결했기 때문에 그때 코드를 조금 수정해 사용

main에서 한 커맨드가 끝나고 while문을 다시 시작할때 자식 프로세스를 wait한다 이때 block되는걸 방지하기 위해 WNOHANG option을 사용한다.

main.c



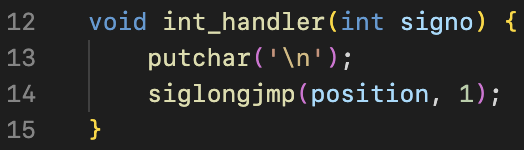
텍스트이(가) 표시된 사진

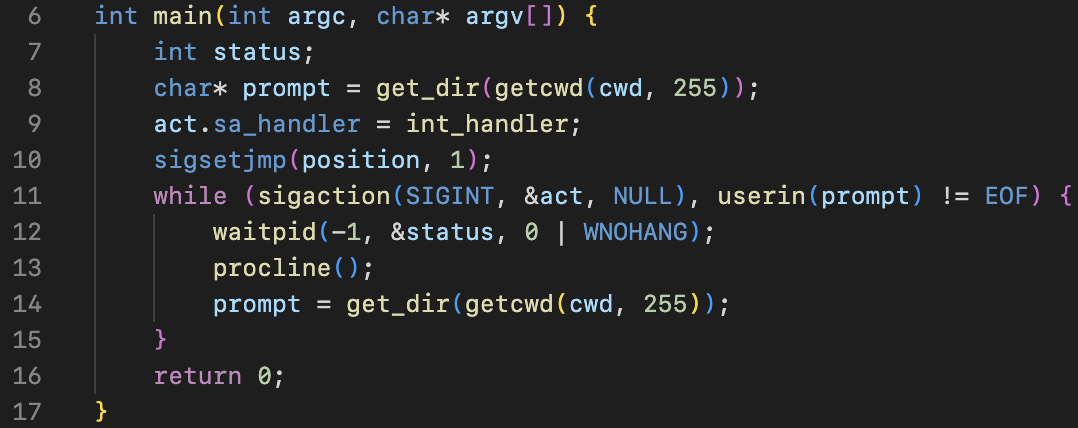
자동 생성된 설명

10초간 잠들었다 종료하는 sleep 프로그램을 백그라운드로 실행한 다음, 10초후에 ps로 확인해 봤을 때 sleep프로세스가 없는것을 볼 수 있다.

**2. SIGINT 처리**

2-1. SIGINT가 들어왔을 때 smallsh가 닫히지 않도록 처리





SIGINT가 들어왔을 때 줄바꿈만 하고 다시 prompt를 띄우는 줄로 돌아가는 int\_handler를 만들고 sigaction을 통해 등록한다

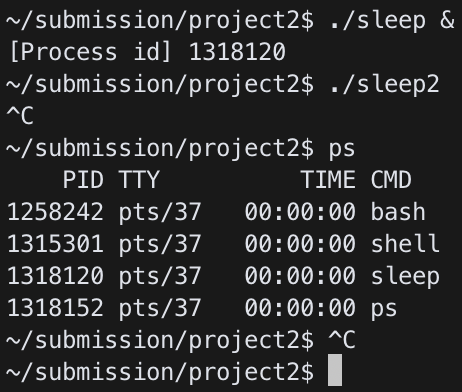
2-2. foreground 프로세스만 종료되도록 처리

smallsh.c의 runcommand 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

프로세스가 background인 경우 프로세스가 리더인 새로운 그룹을 생성하게 해 SIGINT를 받지 않게 했고 foreground인 경우 핸들러에 시그널 기본동작을 하게 만드는 SIG\_DFL을 설정하고 등록해 SIGINT를 받았을 때 종료되게 한다.



10초동안 sleep한 후 종료되는 sleep, sleep2을 만들고 sleep은 background로, sleep2는 foreground로 실행한다

이후 ctrl + C를 눌러 SIGINT를 보내 foreground 프로세스를 종료한 후 ps를 확인해 보면 sleep 프로세스는 잘 작동하는 것을 볼 수 있다

더불어 smallsh도 ctrl + C를 했을 때 줄바뀜 후 정상작동하는걸 볼 수 있다.

**3. |(pipe) 처리**

struct tok\_types

텍스트이(가) 표시된 사진

자동 생성된 설명

gettok함수에서 redirection, pipe 유무를 체크하기 위해 사용

type1은 기존 타입들(ARG, EOF, …) type2는 redirection이나 pipe용으로 사용

smallsh.c의 gettok 함수

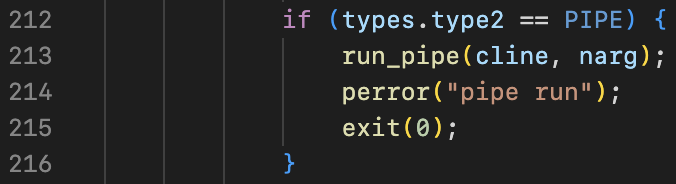
텍스트이(가) 표시된 사진

자동 생성된 설명

pipe 처리를 위해 명령어 토큰을 나눌 때 명령어 중 |가 존재하는지 확인하고 존재한다면 type2를 PIPE로 변경

break가 없기 때문에 default로 내려가게 되고 |도 토큰 배열에 들어가게 된다.

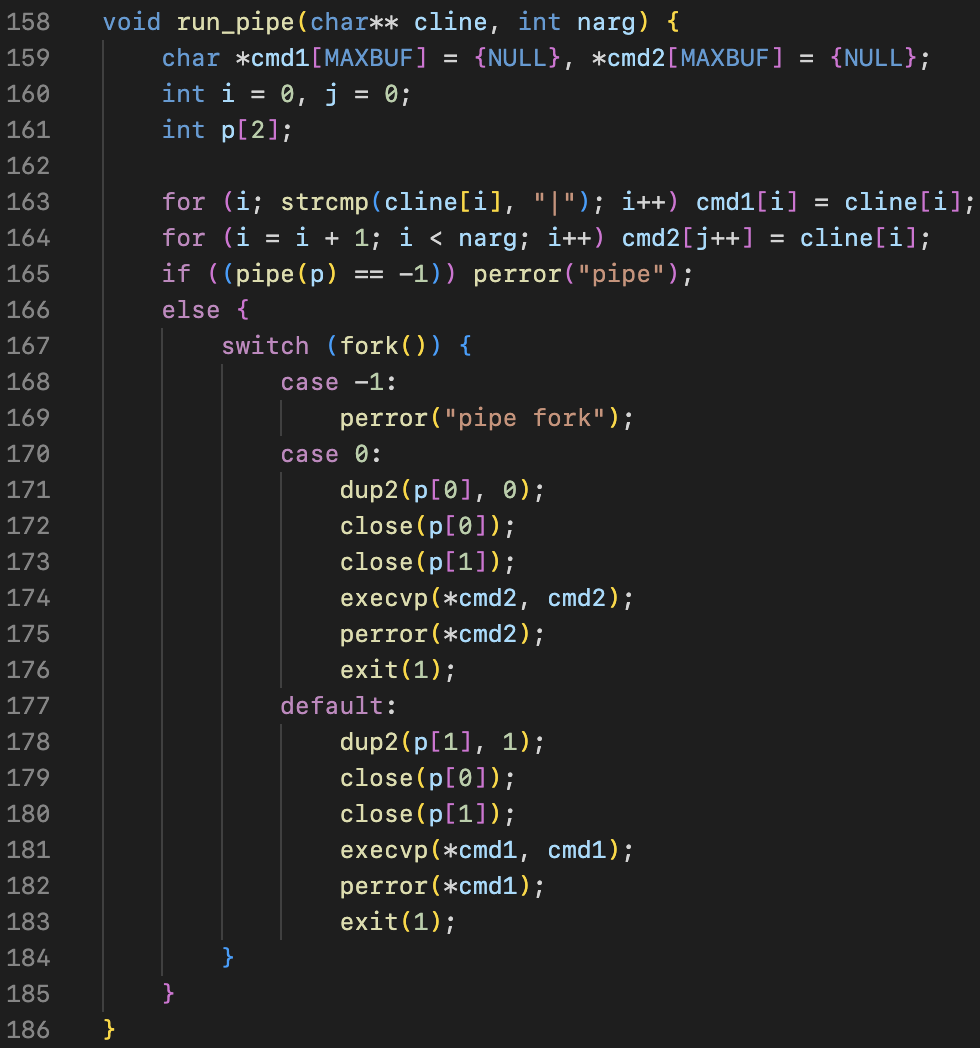
smallsh.c의 runcommand 함수



types.type2가 PIPE라면 run\_pipe 함수로 이동

run\_pipe 함수는 return이 없기 때문에 return되는 경우 에러 출력

smallsh.c의 run\_pipe 함수



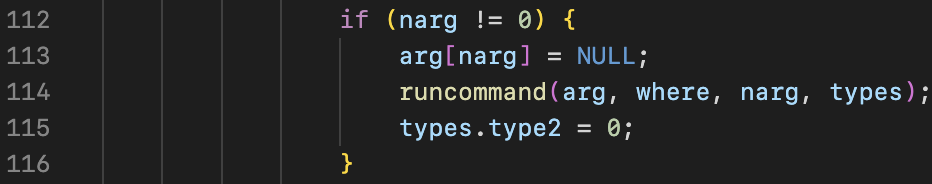
for문을 돌며 | 직전까지 명령어는 cmd1에 | 이후 명령어는 cmd2에 저장

그리고 pipe를 생성후 fork

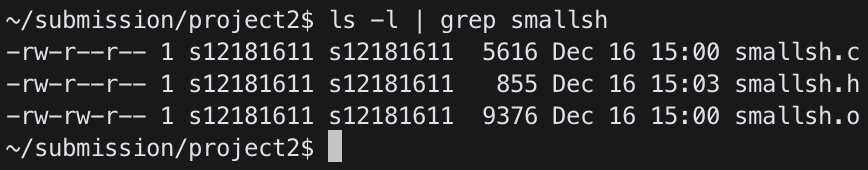
부모 프로세스는 pipe out을 stdout으로 사용하고 자식 프로세스는 pipe in을 stdin으로 사용

부모와 자식 프로세스 모두 pipe를 닫은 후 부모 프로세스는 cmd1, 자식 프로세스는 cmd2를 실행

smallsh.c의 procline



다음 명령어가 정상적으로 처리되도록 runcommand가 실행된 후 types.type2를 0으로 변경



ls -l의 출력값 중에서 smallsh가 포함된 라인만 출력된 것을 확인 가능