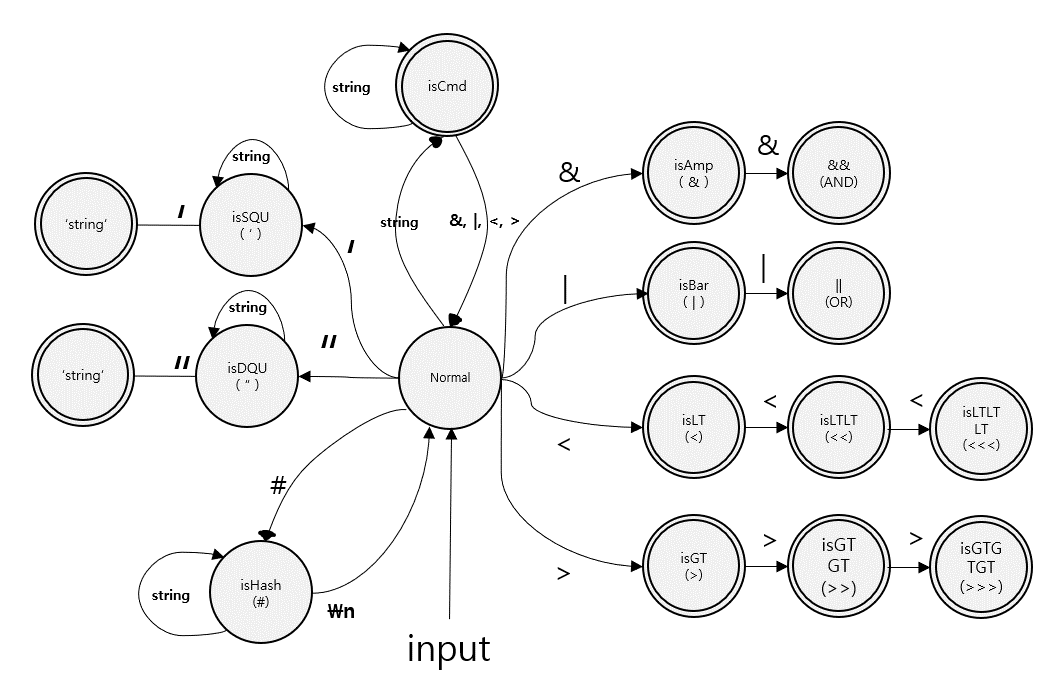
Operating System Project #1 Unix Simple Shell

2018044966 이형창

이 프로젝트는 간단한 Unix Shell을 제작하는 프로젝트입니다. 이 프로젝트는 크게 아래의 순서로 구성하였습니다.

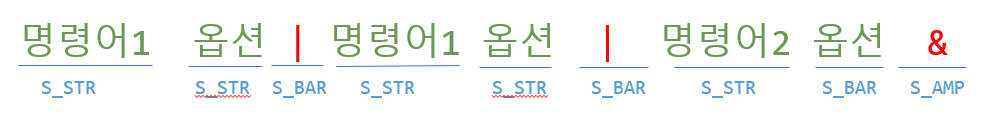
1. 사용자의 명령어를 간단하게 파싱한다.
2. 이 명령어 배열을 기준으로 Pipe를 생성한다.
3. Child 프로세스를 생성하고 Redirection처리한 후 명령어를 실행한다.
4. 해당 명령어가 Background로 돌아가야 한다면 좀비 프로세스 감시 배열에 넣고 아니라면 그냥 자식 프로세스가 종료될 때까지 기다립니다.
5. 좀비 프로세스가 존재하는지 확인하고 제거합니다.
6. 위 과정을 다시 반복합니다.
7. **파싱**

파싱은 최대한 간단하고 확장성을 생각하여 렉서와 파서로 나눴습니다. 렉서는 추후 구현을 위하여 기존 예제 명령어보다 훨씬 폭 넓게 아래와 같은 구성을 했습니다.



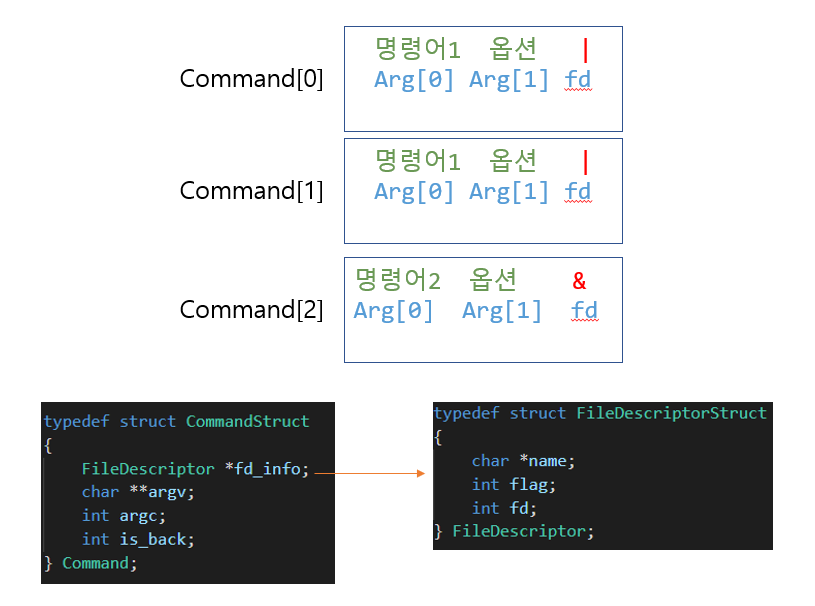
Lexer.c 구성도

Token \*lexer() 함수에서 토큰의 간단한 의미와 문자열을 반환 받습니다. 예제 명령어 중 가장 긴 “명령어1+옵션 | 명령어2+옵션 or 명령어1+옵션 | 명령어2+옵션 &”을 기준으로 생각해보자면 처음엔 string이 들어오기 때문에 isCmd에서 S\_STR을 반환 받습니다. 옵션도 string이 들어오기 때문에 앞과 동일하게 S\_STR를 반환 받습니다. 그러나 그 다음에 ‘|’를 받으면 isBar가 되어 ‘||’를 구분하는데 이 예제에서는 ‘|’ 다음에 명령어라는 string을 받았기에 S\_BAR를 반환 받습니다.



Lexer 반환 예시

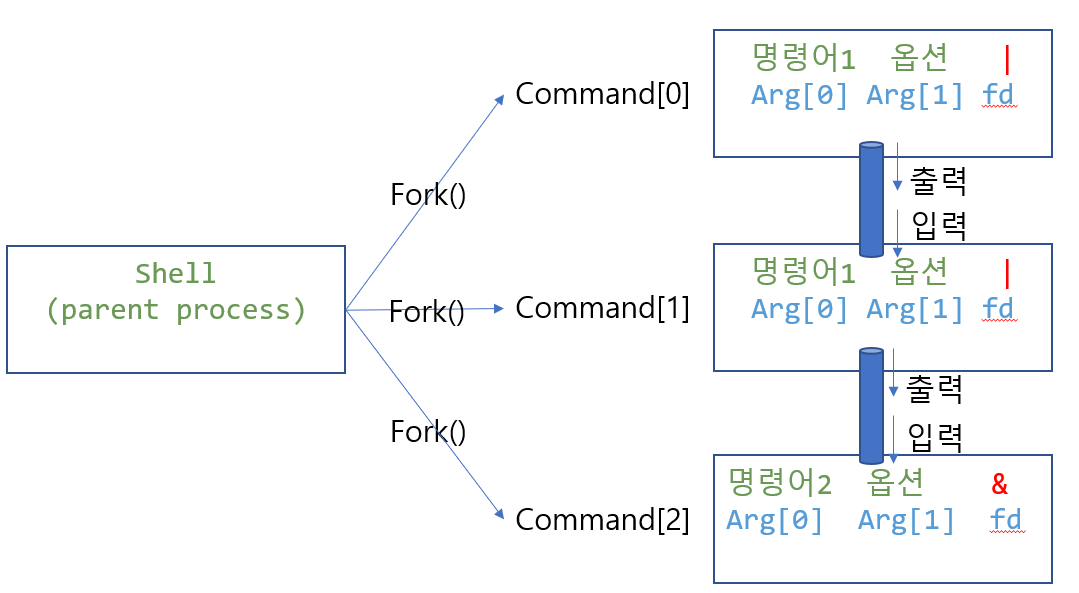
lexer에서 나온 토큰들을 void run\_parse(char \*inputBuffer, Command \*층, int cmd\_size)에서 가 각 커맨드의 File Descriptor에 대한 의미를 분석합니다. Pipe(|), Background(&) 기준으로 명령어를 구분 짓습니다. 그 안 명령어 하나에 Redirection이 존재하면 open 명령어에 필요한 파일명과 flag를 File Descriptor구조체에 작성하여 저장합니다.



Parser 반환의 예시

1. Pipe 생성 및 Child 프로세스 Redirection 처리

파싱을 통해 잘 만들어진 명령어 배열을 받아 명령어 개수만큼 Pipe를 생성합니다. Parent 프로세스는 fork()를 통해 생성된 Pipe를 각 명령어의 관계끼리 연결해줍니다. 이 과정에서 redirection이 존재 할경우 거기에 맞는 file descriptor을 연결해줍니다. 그 다음 기존에 shell에 존재하는 built-in 명령어 인지 확인합니다. 만약 그 명령어가 built-in 명령어 일경우 실행하고 그게 아니라면 execvp()를 실행합니다. 만약 Background(&)가 있다면 부모는 기다리지 않고 다음 명령의 실행을 받는 상태로 갑니다. 이때, Background로 실행 되는 프로세스는 Linked List로 저장되어 Parent process의 routine이 끝날때마다 좀비 프로세스인지 순차적으로 검사하고 좀비 프로세스가 되었을 경우에 목록에서 잘 반환 한후 목록에서 제거시킵니다.



Pipe의 구성도

1. 컴파일 과정

컴파일은 Makefile에 모두 명세되어있다. make clean으로 파일들을 지우고 make로 실행 파일을 만든다 실행파일명은 main이다.