**How programs are launched, and arguments are passed**

프로그램이 실행되는 방식을 알려면, 우선적으로 mash.c 함수를 살펴봐야 한다. 여기서 포인터 배열을 선언해서 문자열을 공백 기준으로 토큰처럼 넣을 수 있게 해주고, ret 변수를 선언해서 run\_command 함수의 결과값을 저장하게 해준다. 그 다음 stdin 입출력장치에서 문자열 한 줄을 가져온다. 이를 parsing 함수에 넣어 입력 받은 문자열을 자르고 각 단어를 tokens 포인터 배열에 strdup 해준다. Nr\_tokens 는 토큰의 개수를 의미한다. Tokens, nr\_tokens 을 run\_command 함수의 argument로 하에 run\_command함수가 실행된다. Run command의 리턴 값은 ret 변수에 저장된다. 리턴 값에 따라 다시 run command 함수가 실행되지 말지 결정된다.

**Approach to implementing aliasing feature**

Alias 부분을 보면 총 두가지 유형이 있다. Alias 만 입력했을 때, alias와 뒤에 단어들이 더 올 때이다. 그래서 token[0]이 alias 일 때에 대해 nr\_token을 기준으로 1개일때, 1개 초과일때로 경우를 조건문을 통해 구분했다. 우선 Alias을 구현하기 위해서는 pa0처럼 리스트헤드로 링크드 리스트를 만들어야 한다. 리스트에 들어갈 구조체에는 키워드를 가리킬 문자 포인터, 그 뒤 문자열을 가리킬 문자 포인터가 들어가야 한다. 설명을 위해 key, sentence로 설명하겠다. Alias 단어 하나를 입력 받으면 list\_for\_each\_entry 로 순회하면서 stderr 출력장치에 key, sentence을 프린트 해주면 된다. 그게 아니라 alias 뒤에 여러 단어가 온다면 list\_add\_tail을 하여 링크드 리스트에 연결시켜주면 된다. 여기서 주의할 점은 tokens[1]은 key에 strdup하고, 뒤에 있는 여러 단어들은 strcat을 이용하여 하나의 문자열로 만들어준다. 그리고 이 문자열을 sentence에 strdup해준다.

이것들이 alias을 tokens[0]으로 입력 받았을 때 실행되는 하나의 과정이다. 허나 alias는 tokens[0]이 alias가 아니더라도 항상 신경 써줘야 한다. 이 말인 즉, 모든 입력 문자열(tokens)에 대해 alias로 변경시켜줘야 하는 단어들이 있으면 변경해줘야 한다는 뜻이다. 그러므로 그 어떤 입력을 tokens로 받아도 run\_command함수가 실행되면 가장 우선적으로 모든 단어를 검사해서 바꿔줄 단어가 있으면 바꿔줘야 한다. 만약 바꿔줄 단어가 있으면, 해당 단어는 sentence로 바꿔줘야 한다. 이는 strdup을 하면 된다. 하지만 이러면 문제가 발생한다. 왜냐면 token[x]에 하나의 단어가 들어가야 하는데, 문장이 들어갔기 때문이다. 그러므로 새롭게 변경된 tokens 들을 다시 파싱 해줘야 한다. 그래야 정상적으로 돌아간다.

**Approach to implementing pipe feature[how many forks I used & for what]**

우선 이를 설명하기 앞서 과연 몇개의 프로세스가 필요한지에 대해 먼저 알아야 한다. 우선 pipe을 실행하고 프롬프트 창이 떠야 한다. 즉 셸 프로세스가 진행이 되어야 한다. 그러므로 셸 프로세스 하나가 끝까지 있어야 하고, pipe(이 과제에서는 두개)을 구현하기 위한 프로세스가 두개가 필요하니 총 3개의 프로세스가 필요하다. 보통 fork을 하면 2의 배수로 생기기 때문에 fork을 1번 하고, 자식 프로세스에서만 fork을 한 번 더 한다. 이렇게 하면 (부모-자식(부모)-자식) 관계로 프로세스가 총 3개가 된다. 자식 프로세스에서 한번 더 fork한 이유는 두 프로세스가 파이프를 통해 소통하기 위해서는 같은 디스크립트 테이블이 있어야 하는데, fork 하면 그대로 디스크립트 테이블이 복사되기 때문이다. 최초의 셸 프로세스를 제외한 나머지 두개의 프로세스는 셸 execvp에 성공해서 다른 내장 프로세스가 될 수 있고(ls, pwd, ,,,,) 아니면 실패해서 셸 프로세스로 남을 수도 있다. 그리고 두번째 fork하기 전에 pipe()함수를 통해 파이프를 만들어준다. 이 과제에서는 특별히 입력을 쓰는 것이 아닌, testcase에 있는 문자열들을 알아서 가져가기 때문에, pipefd[0], pipefd[1]을 닫고, dup함수를 활용해서 STDOUT\_FILENO, STDIN\_FILENO와 파이프를 연결해주었다. 그 다음 execvp을 실행해주면 된다. 만약 두개 다 성공하게 된다면 하나의 셸 프로세스, 두개의 파일들이 실행된다. 만약 execvp에 실패하게 된다면 -EINVAL을 리턴 해 줘야한다. Return -EINVAL을 하는 이유는 이것이 구체적인 오류 상황을 처리하기 때문이다.

**Lessons learned**

코드가 PASS을 통과하고 보고서를 작성하면서 깨닫은 사실인데, 모든 토큰에 대해 alias을 검사하여 기존의 토큰을 변환해주고, 재 파싱 하는 것을 run command 맨 위에서 해줘야 한다. 테스트 케이스에서는 없지만 cd 부분에 alias에 해당하는 단어들이 있다면 틀렸을 것 같다.

그리고 파이프 부분에서 return -EINVAL을 하지 않고, return -1과 return 0을 해줘야 하는 상황을 어떻게 구분하는지 굉장히 오랫동안 고민했는데, return -EINVAL을 하니 시스템이 알아서 판단해준다는 것을 알았다. 하지만 아직 완벽하게 이해한 것은 아니어서 좀 더 공부해봐야 할 것 같다. 그리고 다름 함수 파일들도 자세히 살펴봐야 할 것 같다고 느꼈다.