parsing하여 token들을 나누는 경우의 수에 대하여 저는 크게 2가지로 잡았습니다.

command 안에 큰 따옴표" "로 묶여진 문자열이 있는 경우와 없는 경우 입니다.

이를 시작으로 command 문자열을 parsing하기 위해 문자열의 첫 글자부터 마지막 글자까지 탐색할 포인터 ptr을 선언했습니다.

첫 번째 경우 " "로 묶여진 문자열을 tokens 배열에 저장하기 위하여 buffer 2차원 배열을 만들었고, 각 행과 열의 최대 길이는 MAX_TOKEN_LEN으로 잡았습니다.

command문자열을 순차적으로 탐색하기 위해 while문으로 작성했고, 탈출 조건은 문자열의 마지막 문자 다음인 NULL문자를 만났을 때로 잡았습니다.

while문 안에서 *ptr이 가리키고 있는 문자가 whie space인 경우 그 문자를 NULL문자로 바꾸고, check 변수에 1을 넣었습니다.

check 변수 값이 1인 경우는 command 안의 단어들에 대하여 첫번째 문자에 접근할 수 있게 했습니다. 이제 여기서 첫번째 문자가 큰 따옴표냐 아니냐로 나뉘는데요.

큰 따옴표가 아닌 경우는 check 값에 0을 넣어주고, *nr_tokens번째 인자 tokens는 해당 ptr 값을 가리키게 만든 후 다음 token을 만들기 위해 *nr_tokens값을 1 늘려줍니다.

Check값에 0을 넣어줘야 하는 이유는 단어 상태의 token을 만들어야 하기 때문입니다. ptr이 순차적으로 command를 가리키는 상황에서 0이라는 상태를 따로 두지 않으면 token은 단어 단위로 parsing하지 못합니다.

첫번째 문자가 큰 따옴표인 경우는 다음 큰 따옴표를 만나기 전의 문자열을 parsing해야 하므로

앞에서 선언한 buffer 2차원 배열을 이용했습니다.

우선 buffer[j][i] 배열에 대해 i = 0을 초기화해준 후 ptr 값을 + 1 해줬습니다. i = 0의 이유는 " "로 묶인 문자열이 command 안에 여러 개인 상황을 생각했기 때문입니다.

이 후 ptr이 다음 큰 따옴표를 만나기 전까지 buffer[j][i]에 *ptr값을 넣어줬습니다.

작업이 끝난 후에 해당 문자열이 완성된 buffer[j]를 token에 넣어줬습니다. 그리고 *nr_tokens의 값 + 1해주고 j 역시 + 1해줬습니다.

이번 과제에서 고민한 부분은 최소 문자가 1개 이상인 문자열들을 token에 어떻게 저장할 것인가, 다중 " " 안의 문자열들에 대해 어떻게 처리할 것인지에 대한 것이었습니다.

그리고 해결 전략은 2차원 배열 buffer 그리고 check 상태 변수를 사용한 것이었습니다.