[static int parse_command 함수 분석]

이 함수는 입력 받은 command를 parsing해주는 함수이다. 공백문자가 들어온 경우 그 공백문자를 기준으로 parsing 되고, '"'문자가 들어오는 경우에는 ""안에 공백문자가 있더라도 하나의 묶음으로 판단해주는 함수이다.

<Input & output&declare>

*command 는 들어온 전체 문자 배열, *nr_tokens 는 토큰의 개수, *tokens[]는 토큰 배열들을 입력값으로 받았다. 함수가 종료되는 마지막 시점에서 return 0; 을 선언해주었다.

quatationCnt 는*nr_tokens 의 개수를 세기 위하여 사용하였고, 출력할 tokens에 값을 집어넣기 위하여 필요한 index의 기능으로도 구현되도록 하였다. 'char*tempTokens' 은 parsing 한 부분을 임시 저장하기 위한 것으로 선언하였다. 하나의 토큰이 되는 원소들을 받아 잠시 저장한 후, 공백이 나오면 한 덩어리를 tokens에 넘겨주기 위하여 사용하였다. i는 command 배열의 index로 사용하였다. j는 tempToken 배열의 index로 사용하였다.

<코드해석 : parsing 구현 방법>

command 배열에서 공백문자가 들어올 때까지 코드를 반복하기 위하여 while(command[i]!=0)을 사용하였다.

만약 command[i]에서 공백이 들어오는 경우, tokens에 이전까지 나온 묶음을 넣어주도록 하였다. tempToken배열이 비어있다면 아무런 실행을 하지 않고, tempToken이 비어있지 않은 경우, 즉 tempToken의 인덱스가 0이 아닌 경우(j!=0), quatation번째 tokens 배열에 tempTokens를 대입하였다. 다음 token은 quotation+1번째에 넣어주기 위하여 quotation = quotation+1을 하고 tempTokens의 index인 j를 0으로 초기화하고, tempTokens를 초기화하였다.

command[i]에서 공백이 들어오지 않은 경우, "문자가 들어온 경우와 "문자가 들어오지 않은 경우로 나누었다. '"'문자가 들어온 경우, command의 index인 i를 1만큼 증가시키고 다음 '"' 가 들어올 때까지 tempTokens에 "command" 값을 넣어주었다. '"'문자가 들어오지 않은 경우에는 tempTokens에 command 값을 넣어주었다.

*nr_tokens(토큰의 개수)는 토큰의 개수 겸 token의 인덱스로 사용하기 위해 선언한 quatationCnt의 값을 넣어주었다. 마지막에는 return 0을 통하여 함수를 종료시켰다.

<코드를 만들면서 느낀점>

사실 처음에 공백문자를 기준으로 파싱한다고 했을 때, <string.h>안에 있는 strtok()함수를 사용하면 될 것 같다고 생각하였는데, 이 함수를 사용하지 않고 구현하는 방법을 어떻게 생각해야되나 고민이 많았다. isspace()함수를 사용하라고 하였을 때, if문을 사용하여 공백문자의 유무를 나눠야 겠다고 생각하였다. command값이 없을 때 까지 반복하기 위하여 while문의 조건으로 command에 '\0' 이 아닌 경우 계속 반복하는 반복문을 사용해야겠다는 생각을 하였다. 이는 문제에 접근하기 위한 큰 그림을 그려 접근할 수 있게 해주었던 것 같다.

parse_command() 함수를 만드는 과정을 통해 미리 구현되어 있는 함수를 새롭게 구현하기 위하여 어떤 방법이 있을지 생각할 수 있었던 것 같다.