첨부파일 설명

2016314827.c: convert.c 안의 convert 함수를 구현하였다.

convert.h: 2016314827.c 안의 함수들에 대한 헤더파일이다. convert 함수뿐만이 아니라 다른 함수도 들어있어서 함수들을 몇 개 추가하였다.

string_sw.c, string_sw.h: strcpy, strcat, strrchr 등의 함수를 사용하기 위해 넣은 라이브러리 파일들이다.

Makefile: 2016314827.c, string_sw.c, main.c를 한번에 컴파일 해준다. Makefile 안의 파일이름도 수정해놓았으므로 별도의 수정이 필요없다.

PA 1 코드 설명

#13 ~ #19: 필요한 라이브러리들을 추가했다. 특히, strcpy, strcmp, strcat 등의 함수를 활용했기 때문에 string_sw.h도 추가해주었다.

#25 ~ #58: read할 파일을 open하고, write할 파일을 open하는 과정이 담긴 코드이고, 과제 안내서에 적인 4가지의 오류 모두 에러코드에 따른 swtich 문을 통해 구현하였다.

#61: 주어진 파일이 unix 파일인지 dos 파일인지 알아내는 과정이 담긴 코드이다. fileType에는 U, D, N이 있는데 이는 getFileType 함수에서 설명할 것이다.

#65 ~ #75: file type을 알아내고, 해당하는 타입에 따라 각기 다른 변환 함수에 넣어준다. unix파일이면 dos파일로(case U), dos 파일이면 unix 파일로(case D), 그 무엇도 아니면 case N으로 넣어준다. 각 변환 함수는 파일의 크기를 리턴해준다.

#77 ~ #92: 파일 이름들과 파일 크기를 stdout해주고, free와 close를 해주는 부분이다. 라이브러리 함수를 쓰면 안되므로 write함수를 썼다.

getFileType 함수: \n과 \r\n의 개수를 각각 세서 비교하는 로직이다. 1byte 단위로 문자를 읽어들여서 \n인지 \r\n인지 판별한다. 판별 방식은, \n을 발견했을 때 바로 앞의 문자가 \r이면 dosCount를 1 증가시키고, \r이 아니면 unixCount를 증가시키는 방식이다. 이 카운트 들에 따라서 무슨 file 타입인지 리턴해준다.

만약 \n과 \r\n 수가 같으면, 이 파일은 unix포맷으로 변환해야하므로 이 파일은 dos 파일이라고 리턴해준다. 만약 파일 안에 new line을 발견하지 못하면, 어떤 파일 타입인지 판별할 수가 없으므로 none이라는 뜻인 N을 리턴한다.

isSourceCode 함수: 주어진 인풋 파일이 .c, .h, .java 파일인지 아닌지 검사해주는 함수이다. 이 때 확장자는 맨 뒤에 찍힌 점을 기준으로 결정되므로 앞에서부터 탐색하지않고 뒤에서부터 찾아주는 함수인 strrchr을 활용했다. 만약 .c, .h, .java 중 하나에 해당하면 1을, 아니라면 0을 리턴한다.

unix2dos 함수: unix 포맷을 dos 포맷으로 변환해주는 함수이다. 이때 추가점수 부분을 구현하기 위해 spaceCount라는 변수를 추가했다. 1byte씩 읽으면서 각 case에 해당하는 행동을 switch 문으로 구현했다.

dos포맷으로 바꿀때는 tab 하나는 4개의 space와 같다. 따라서 space나 tab과 같은 공백문자는 불필요한 공백문자일 경우 지우고 다시 쓰는 경우를 피하기 위하여, 불필요하지 않다는 것이 확정될 경우에만 써주고, 아직 확정이 되지 않을 때엔 spaceCount에다 추가해주었다.

만약 \r을 발견했을 때, 이게 뒤에 \n이 따라오는 \r인지, 아니면 그냥 단순히 \r이라는 문자일 뿐인지 모르므로, 두 가지 경우로 나누었다. 만약 \r\n일 경우, 그냥 \r\n을 그대로 써주고, spaceCount를 0으로 초기화해준다(불필요한 공백문자 제거).

만약 그냥 단순한 \r일 뿐이라면 a, b, c와 같은 평범한 문자와 다를 바가 없기 때문에 default case로 이어서 진행된다. default case에서는 space를 써도 된다는 사항이 확정이 되는 상황이므로, 그 동안 누적해왔던 space를 모두 써주고, 버퍼로 읽어들인 문자를 써준다.

버퍼에 \n이 담긴 상황은 unix new line을 발견한 상황이다. \r\n인지 판별하지 않아도 되는 이유는 이미 앞에 case \r에서 모두 고려되었기 때문이다. 따라서 \r\n으로 바꿔주기만 하면 된다.

dos2unix 함수: 로직은 unix2dos 함수와 거의 같다. 다만, 한 가지 큰 차이점이 있는데, unix 파일에는 tab과 space가 섞여 들어갈 수 있다는 점이다. 따라서 단순히 counting만 해준다면 tab과 space의 순서를 고려하지 못한다. 따라서 white space를

저장하는 버퍼를 두어서 순서대로 tab과 space를 담을 수 있도록 했다. 만약 소스코드일 경우 8 space == 1 tab라는 점도 물론 고려했다. 이는 #219 ~ #238에 구현이 되어 있다.

copy 함수: 만약 인풋파일에 new line character가 없을 경우 unix 파일인지, dos 파일인지 구분하지 못한다. 따라서 이 경우엔 output파일에 그냥 그대로 복사해 넣었다.