리눅스 시스템 프로그래밍

Soongil Score

숭실대학교

컴퓨터학부

20162467

민지혜

# 프로젝트 개요

Ssu\_score 은 정답 파일을 기준으로 학생들이 제출한 답안 파일을 채점하는 프로그램이다. 이 프로젝트의 목표는 유닉스/리눅스 컴퓨팅 환경에서 제공하는 파일 입출력, 파일 속성 디렉토리에 관한 시스템 호출 함수와 라이브러리 함수를 이용한 프로그램을 작성하여 시스템 프로그램 설계 및 응용 능력을 향상시키는 것이다.

# 설계

본 프로그램은 학생의 정보를 담고있는Std 구조체와 답안의 정보를 담고 있는 Q 구조체를 가지고 있다.

## Std구조체

Std 구조체 내에는 학생의 학번이 id 에 문자열 형태로 저장이 되고, 학생의 폴더의 경로를 path에 저장하고 있다. 또한 학생이 제출한 답안들이 answer에 문제의 순서대로 저장이 되어있다. 학생의 답안을 채점한 결과 각 문제를 맞췄는 지 여부가 card에 맞춘 경우 1, 틀린 경우 0으로 저장이 된다.

struct Std {

char id[10];

char path[100];

char \*answer[100];

int card[100];

};

struct Std std[22];

## Q 구조체

Q 구조체 내에는 문제의 이름이 name에 문자열 형태로 저장이 된다. 각 문제의 답을 가지고 있는 .txt또는 .stdout 파일의 경로가 path에 저장이 된다. 또한 각 문제의 점수들이 score에 float형태로 저장이 된다. 각 문제의 답안의 개수를 answer\_num에 int 형태로 저장을 하고 해당 문제의 정답들을 answer에 저장을 한다.

struct Q {

char name[30];

char path[30];

float score;

int blank\_q;

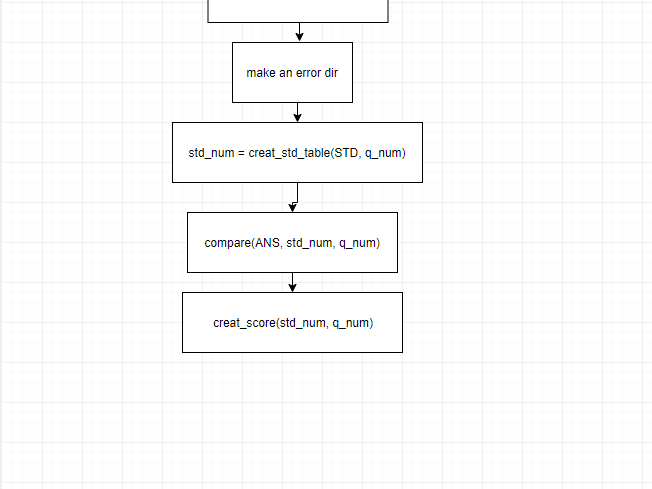
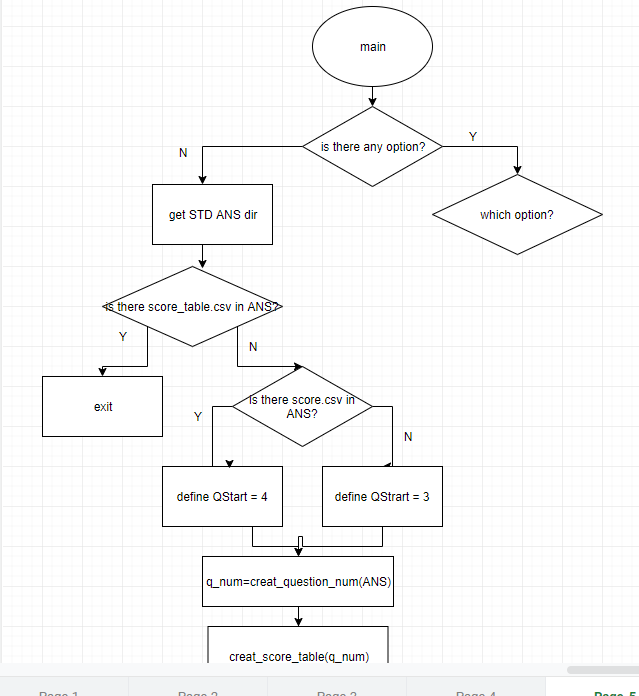
int answer\_num;

char \*answer[100];

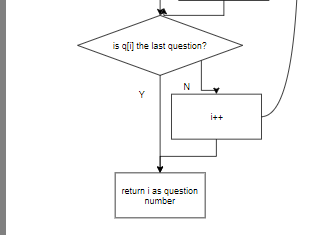
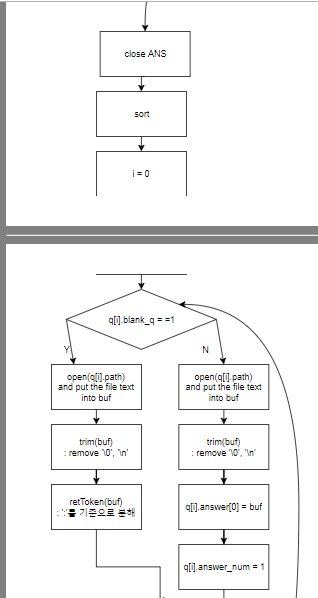
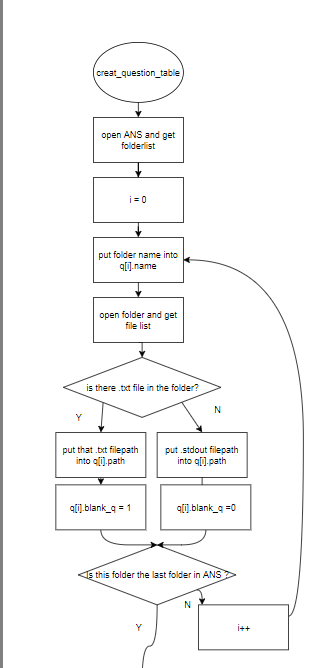
};

struct Q q[100];

## main()



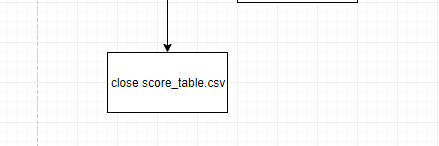
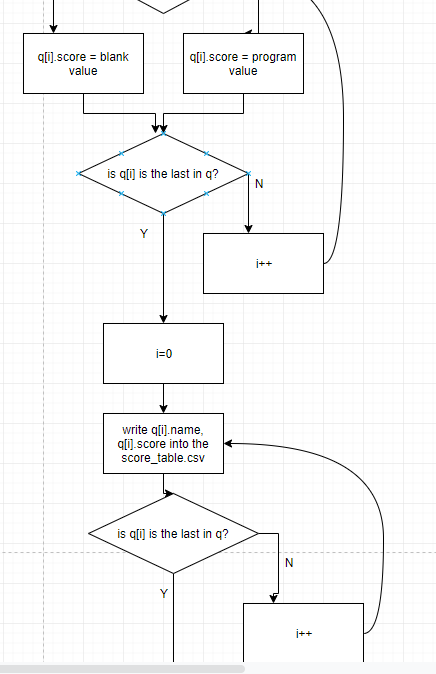
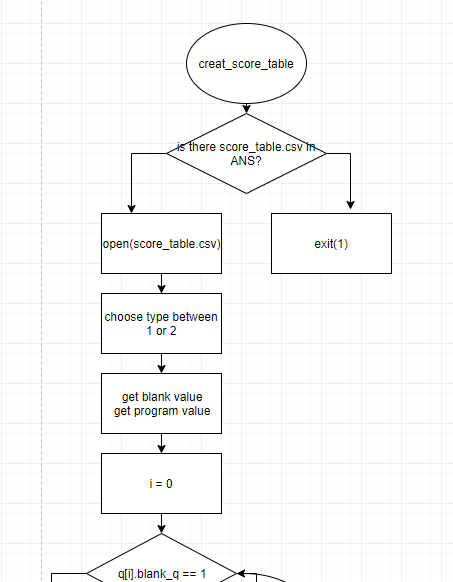
## Create\_question\_table()



## Create\_std\_table()

## cc

## Create\_score\_table()



# 구현

## void creat\_score\_table(char\* ANS,int q\_num);

ANS dir 에 score\_table.csv를 생성하기 위해 만든 함수이다.

## int creat\_question\_table(char \*ANS);

문제 목록과 문제유형, 문제정답 파일(.txt, .stdout)파일의 정보를 Q q구조체에 저장하기 위한 함수이다.

## void sort(int \*list, int count);

문제목록을 순서대로 정렬하기 위한 함수이다.

## int creat\_std\_table(char \*STD, int q\_num);

학생목록과 학생의 답을 저장하기 위한 함수이다.

## void compare(char \* STD, int std\_num, int q\_num);

학생의 답과 답지의 정답을 비교하기 위한 함수이다.

## int retToken(int i, char \*inp);

빈칸문제일 경우 정답지의 답 목록을 ‘:’로 토큰을 나누어서 Q q[i].answer[j] 에 동적할당으로 답을 저장하기 위한 함수이다. J는 해당 문제의 답지 안에 있는 답의 개수이며 j를 Q q[i].answer\_num[j] 로 해당문제의 답의 개수를 return 해준다.

## void creat\_score(char \*ANS, int std\_num, int q\_num);

ANS 디렉토리에 score.csv함수를 만들기 위한 함수이다.

## void execute\_std(int st, int a);

학생들이 작성해서 답안으로 제출한 프로그램을 컴파일 후 실행파일을 저장하기 위한 함수이다. 학생들이 제출한 프로그램을 thread를 만들어서 다른 프로세스에서 돌도록 했고 다른 프로세스에 kill함수를 올려서 5초를 기다린 후에도 프로그램이 종료가 되지 않으면 kill함수가 죽이도록 했다.

## void trim(char\* ptr);

앞 뒤 공백, 문자열 사이의 공백과 널문자를 제거하기 위한 함수이다.

## void \*t\_function(void\* multiple\_arg);

학생이 제출한 프로그램을 실행하는 함수이다.

## static void killer(void \*arg);

학생이 제출한 프로그램이 종료되지 않을 경우 죽이는 역할을 한다.

## static void \*control(void \*arg);

kill 함수에게 명령을 내려 학생이 제출한 프로그램이 죽게 한다.

## void DeleteAllFile(char\* szDir);

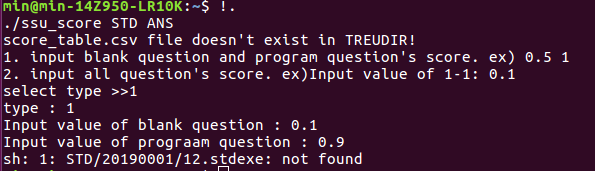
에러 디렉토리가 비어 있지 않을 경우 디렉토리를 비워준다.

## void capital(char \*ptr);

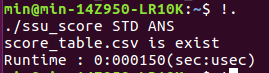
소문자를 대문자로 전환해준다.

# 테스트 및 결과

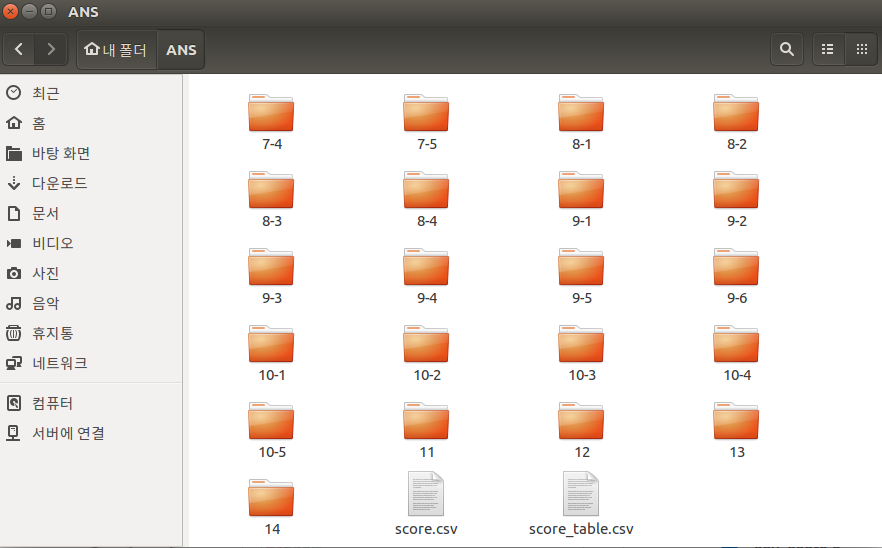
Without option



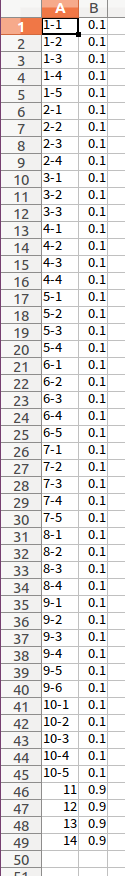
Score\_table.csv가 있을 경우

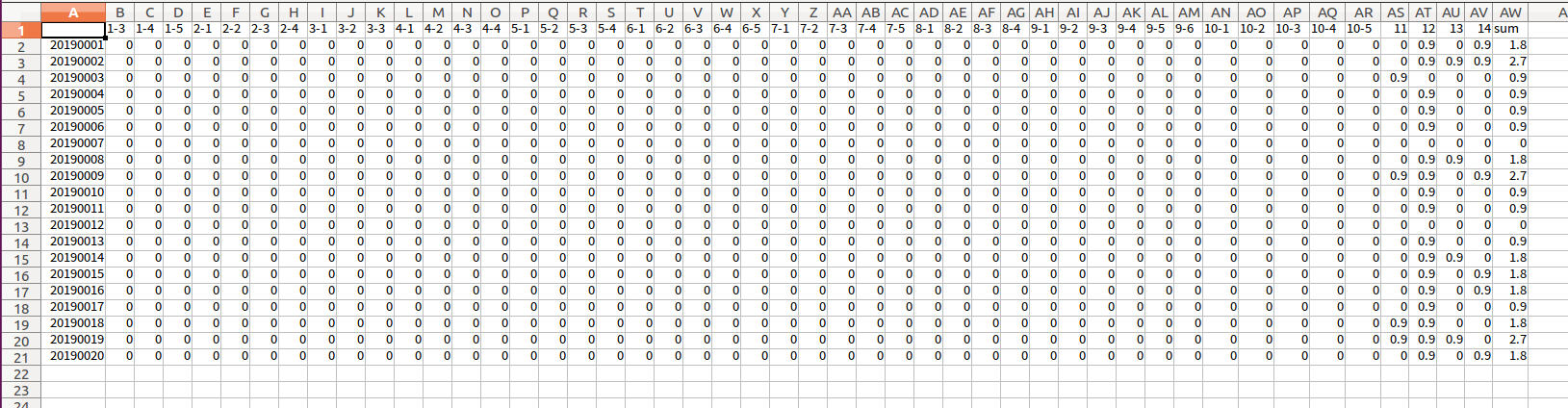


ANSdir 에 score\_table.csv와 score.csv자동 생성

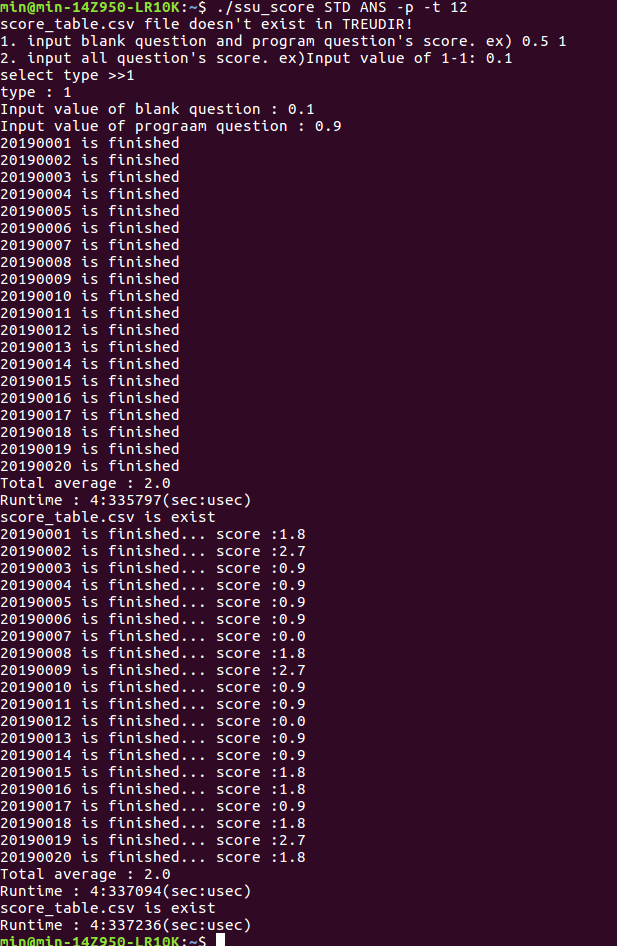


score\_table.csv

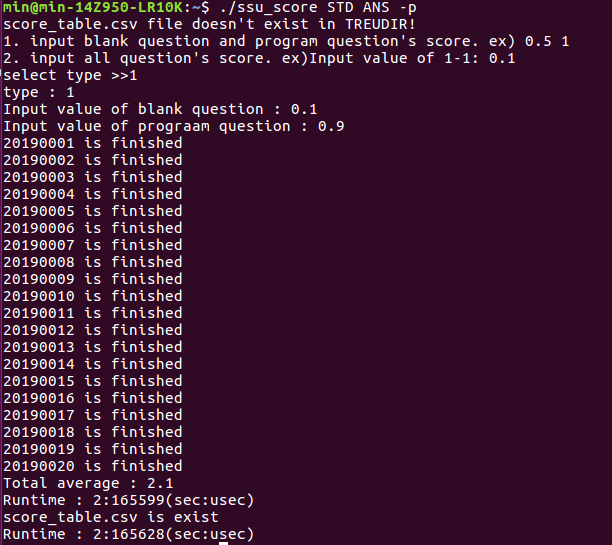


Score.csv

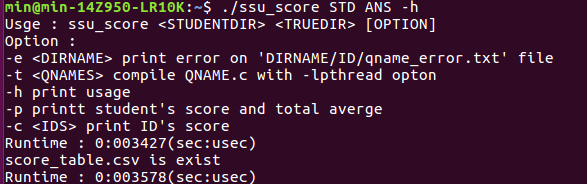
-t



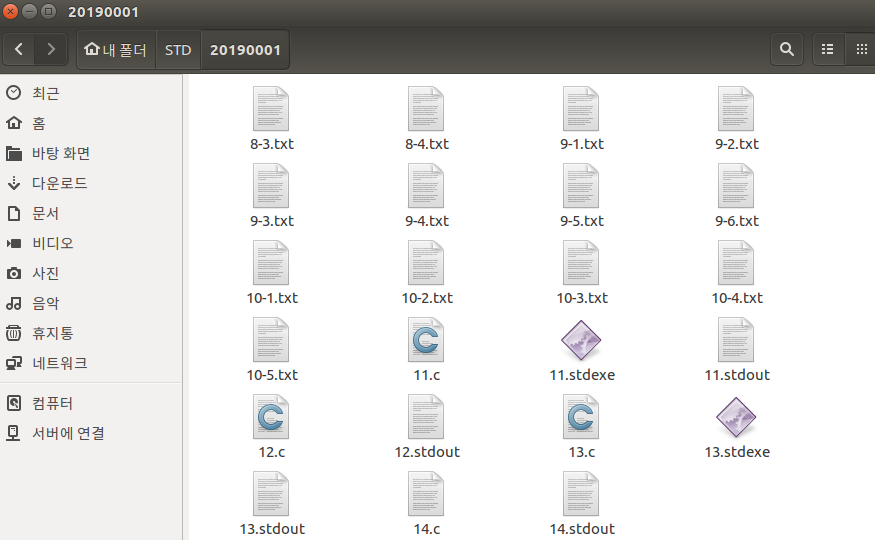
-p



-h



각 학생폴더에 실행파일 자동생성 및 실행 결과 .stdout 에 저장



# 소스코드

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <dirent.h>

#include <sys/types.h>

#include <ctype.h>

#include <time.h>

#include <pthread.h>

#include <signal.h>

#include <pthread.h>

#include <sys/stat.h>

#include <sys/time.h>

#define BUFFER\_SIZE 5000

#define STD\_NUM 100

#define MAX\_STR\_LEN 4000

#define KILL\_TIME 5

#define SECOND\_TO\_MICRO 1000000

void ssu\_runtime(struct timeval\* begin\_t, struct timeval\*end\_t) {

end\_t -> tv\_sec -= begin\_t -> tv\_sec;

if(end\_t -> tv\_usec < begin\_t -> tv\_usec) {

end\_t -> tv\_sec--;

end\_t -> tv\_usec += SECOND\_TO\_MICRO;

}

end\_t -> tv\_usec -= begin\_t -> tv\_usec;

printf("Runtime : %ld:%06ld(sec:usec)\n", end\_t -> tv\_sec, end\_t -> tv\_usec);

}

struct Std {

char id[10];

char path[100];

char \*answer[100];

int card[100];

};

struct Q {

char name[30];

char path[30];

float score;

int blank\_q;//빈칸문제이면 1 아니면 0

int answer\_num;

char \*answer[100];

};

typedef struct MultipleArg {

char \*tmp;

char \*filename;

char \*error;

}MultipleArg;

char \*error\_dir = "./err";

struct Q q[100];

struct Std std[22];

int QStart = 4;

void creat\_score\_table(char\* ANS,int q\_num);

int creat\_question\_table(char \*ANS);

void sort(int \*list, int count);

int creat\_std\_table(char \*STD, int q\_num);

void compare(char \* STD, int std\_num, int q\_num);

int retToken(int i, char \*inp);

void creat\_score(char \*ANS, int std\_num, int q\_num);

void execute\_std(int st, int a);

void trim(char\* ptr); //문자열 좌우 공백 모두 삭제

void \*t\_function(void\* multiple\_arg);

static void killer(void \*arg);

static void \*control(void \*arg);

void DeleteAllFile(char\* szDir);

void capital(char \*ptr);

int main(int argc, char \* argv[]){

int fd\_t, fd1, fd2;

int q\_num, std\_num;

int i, j;

struct timeval begin\_t, end\_t;

gettimeofday(&begin\_t, NULL);

if(argc < 3) {

fprintf(stderr, "usage : %s <file>\n", argv[0]);

exit(1);

}

int flag\_e =0, flag\_p=0, flag\_t =0, flag\_h=0, flag\_c=0;

int c;

char \*std\_ids[5];

char \*q\_names[5];

int c\_num;

int t\_num;

char \*dir[2];

//옵션을 걸었을 경우

while((c = getopt(argc, argv, "e:pt:c:h")) != -1) {

switch(c) {

case 'e' :

flag\_e = 1;

error\_dir = optarg;

mkdir(error\_dir, 0777);

break;

case 'p' : //채점을 진행하면서 각 학생의 점수 출력 및 전체 평균 출력

flag\_p = 1;

break;

case 't' :

flag\_t = 1;

i=0;

q\_names[i] = optarg;

i++;

for(i, optind; i<5 && optind < argc && \*argv[optind] != '-'; i++, optind++)

q\_names[i] = argv[optind];

t\_num = i;

break;

case 'c' :

flag\_c = 1;

i=0;

std\_ids[i] = optarg;

i++;

for(i, optind; i<5 && optind<argc && \*argv[optind]!= '-'; i++, optind++)

std\_ids[i] = argv[optind];

c\_num = i;

break;

case 'h' :

flag\_h =1;

break;

case '?' :

if(optopt =='c'||optopt == 'e' || optopt =='t') {

printf("option requires value\n");

break;

}

printf("Unknown flag : %c", optopt);

break;

}

}

i=0;

while(optind < argc) {

dir[i] = argv[optind];

i++;

optind++;

}

if(flag\_e) {

q\_num = creat\_question\_table(dir[1]);

creat\_score\_table(dir[1],q\_num);

std\_num = creat\_std\_table(dir[0], q\_num);

printf("grading student's test papers..\n");

compare(dir[0], std\_num, q\_num);

creat\_score(dir[1], std\_num, q\_num);

gettimeofday(&end\_t, NULL);

ssu\_runtime(&begin\_t, &end\_t);

}

if(flag\_p) {

int nResult, Result;

char tmp[30];

double total = 0;

double sum=0;

int fd;

int i;

char buf[BUFFER\_SIZE];

char temp[30];

strcpy(tmp, dir[1]);

strcat(tmp, "/score\_table.csv");

strcpy(temp, dir[1]);

strcat(temp, "/score.csv");

if(Result =access(temp, 0) == 0) {

QStart = 6;

}

else if(Result == -1) {

QStart = 4;

}

if((nResult = access(tmp, 0))==0){

fd = open(tmp,O\_RDONLY);

q\_num = creat\_question\_table(dir[1]);

lseek(fd, sizeof(",0.90"), SEEK\_SET);

for(i=QStart; i<q\_num; i++) {

lseek(fd, sizeof(q[i].name)+1, SEEK\_CUR);

read(fd, buf, 5);

q[i].score = atoi(buf);

}

close(fd);

std\_num = creat\_std\_table(dir[0], q\_num);

compare(dir[0], std\_num, q\_num);

creat\_score(dir[1], std\_num, q\_num);

}

else if(nResult == -1) {

q\_num = creat\_question\_table(dir[1]);

creat\_score\_table(dir[1], q\_num);

std\_num = creat\_std\_table(dir[0], q\_num);

compare(dir[0], std\_num, q\_num);

creat\_score(dir[1], std\_num, q\_num);

}

for(j=0; j<std\_num; j++) {

sum = 0;

for(i=0; i<q\_num; i++) {

if(std[j].card[i] == 1){

sum += q[i].score;

total += sum;

}

}

}

printf("Total average : %.1f\n", total/std\_num);

gettimeofday(&end\_t, NULL);

ssu\_runtime(&begin\_t, &end\_t);

}

if(flag\_t) {

int nResult, Result;

int k;

char tmp[30];

double total = 0;

double sum=0;

strcpy(tmp, dir[1]);

strcat(tmp, "/score\_table.csv");

char temp[30];

strcpy(temp, dir[1]);

strcat(temp, "/score.csv");

if(Result=access(temp, 0) == 0) {

QStart = 6;

}

else if (Result == -1){

QStart = 4;

}

if((nResult = access(tmp, 0))==0){

printf("score\_table.csv is exist\n");

}

else if(nResult == -1) {

q\_num = creat\_question\_table(dir[1]);

creat\_score\_table(dir[1], q\_num);

std\_num = creat\_std\_table(dir[0], q\_num);

compare(dir[0], std\_num, q\_num);

creat\_score(dir[1],std\_num, q\_num);

}

for(j=0; j<std\_num; j++) {

sum = 0;

for(i=0; i<q\_num; i++) {

for(k=0; k<t\_num; k++) {

if(std[j].card[i] == 1&&q\_names[k]){

sum += q[i].score;

total += sum;

}

}

}

printf("%s is finished... score :%.1f\n",std[j].id, sum);

}

printf("Total average : %.1f\n", total/std\_num);

gettimeofday(&end\_t, NULL);

ssu\_runtime(&begin\_t, &end\_t);

}

if(flag\_h) {printf("Usge : ssu\_score <STUDENTDIR> <TRUEDIR> [OPTION]\n"

"Option :\n"

"-e <DIRNAME> print error on 'DIRNAME/ID/qname\_error.txt' file\n"

"-t <QNAMES> compile QNAME.c with -lpthread opton\n"

"-h print usage\n"

"-p printt student's score and total averge\n"

"-c <IDS> print ID's score\n");

gettimeofday(&end\_t, NULL);

ssu\_runtime(&begin\_t, &end\_t);

}

if(flag\_c) {

int nResult;

int k;

char tmp[30];

double total = 0;

double sum=0;

strcpy(tmp, dir[1]);

strcat(tmp, "/score\_table.csv");

if((nResult = access(tmp, 0))==0){

printf("score\_table.csv is exist\n");

}

char temp[30];

strcpy(temp, dir[1]);

strcat(temp, "/score.csv");

if(access(temp, 0) == 0) {

QStart = 4;

}

else if(nResult == -1) {

q\_num = creat\_question\_table(dir[1]);

creat\_score\_table(dir[1],q\_num);

std\_num = creat\_std\_table(dir[0], q\_num);

compare(dir[0], std\_num, q\_num);

creat\_score(dir[1],std\_num, q\_num);

}

for(k=0; k<c\_num; k++) {

sum = 0;

for(j=0; j<std\_num; j++) {

if(strcmp(std\_ids[k],std[j].id)==0){

for(i=0; i<q\_num; i++) {

if(std[j].card[i] == 1){

sum += q[i].score;

}

}

printf("%s's score : %.1f\n", std\_ids[k], sum);

break;

}

}

}gettimeofday(&end\_t, NULL);

ssu\_runtime(&begin\_t, &end\_t);

}

/\*score\_table이 존재\*/

int nResult;

char tmp[30];

strcpy(tmp, dir[1]);

strcat(tmp, "/score\_table.csv");

if((nResult = access(tmp, 0))==0){ //존재

printf("score\_table.csv is exist\n");

}

else if(nResult == -1){

char temp[30];

strcpy(temp, dir[1]);

strcat(temp, "/score.csv");

if(access(temp, 0) == 0) { //존재

QStart = 4;

}

q\_num = creat\_question\_table(dir[1]);

creat\_score\_table(dir[1],q\_num);

mkdir(error\_dir, 0777);

std\_num = creat\_std\_table(dir[0], q\_num);

printf("grading student's test papers..\n");

compare(dir[0], std\_num, q\_num);

creat\_score(dir[1], std\_num, q\_num);

}

gettimeofday(&end\_t, NULL);

ssu\_runtime(&begin\_t, &end\_t);

}

int creat\_question\_table(char \*ANS) {

char buf[BUFFER\_SIZE];

int fsize;

int k;

char folderlist[50];

int list[100];

struct dirent \*\*namelist;

struct dirent \*file = NULL;

int fd\_q;

int count;

int i, c;

char \*temp;

DIR \*f;

char \*ext;

if((count = scandir(ANS, &namelist, NULL, alphasort)) == -1){

fprintf(stderr, "%s directory scan Error\n","./ANS/");

exit(1);

}

//open folder and put file list in q

for(c=0; c < count-2; c++){

i = c+2;

strcpy(folderlist, ANS);

strcat(folderlist, "/");

strcat(folderlist, namelist[i]->d\_name); // ./ANS/2-2

strcat(folderlist, "/"); // ./ANS/2-2/

f = opendir(folderlist);

strcat(folderlist, namelist[i]->d\_name); // ./ANS/2-2/2-2

if(f != NULL){ //forder 열기

while((file = readdir(f)) != NULL) //file 이름명 q에 넣기, 문제유형 파악

{

ext = strrchr(file->d\_name, '.');

if(ext == NULL) {

continue;

}

if(strcmp(ext, ".txt") == 0) {

strcpy(q[c].path, folderlist); // ./ANS/1-1/1-1

strcat(q[c].path, ".txt");

strcpy(q[c].name, namelist[i]->d\_name); //1-1

q[c].blank\_q = 1;

break;

}

if(strcmp(ext, ".stdout") == 0) {

strcpy(q[c].path, folderlist);// ./ANS/11/11

strcat(q[c].path, ".stdout"); // ./ANS/11/11.stdout

strcpy(q[c].name, namelist[i]->d\_name);

q[c].blank\_q = 0;

break;

}

}

}

closedir(f);

temp = namelist[i]->d\_name; // temp = 1-2

temp = strtok(temp, "-"); //temp = 1

list[c] = atoi(temp);

}

for(i=0; i<count; i++){

free(namelist[i]);

}

free(namelist);

//정렬

sort(list, count);

for(c=QStart; c<count; c++) {

for(k=0; k<BUFFER\_SIZE; k++){

buf[k] = '\0';

}

if(q[c].blank\_q == 1) {

if((fd\_q = open(q[c].path, O\_RDONLY)) < 0) {

fprintf(stderr,"200 open error :%s\n", q[c].path);

exit(1);

}

fsize = lseek(fd\_q, 0, SEEK\_END);

lseek(fd\_q, 0, SEEK\_SET);

read(fd\_q, buf, fsize);

//retToken()로 q[i].answer 입력받은 스트링을 전달

trim(buf);

q[c].answer\_num = retToken(c, buf);

close(fd\_q);

}

else if(q[c].blank\_q == 0){

if((fd\_q = open(q[c].path ,O\_RDONLY|O\_CREAT|O\_TRUNC)) < 0) {

fprintf(stderr, "218 open error %s\n", q[c].path);

exit(1);

}

fsize = lseek(fd\_q, 0, SEEK\_END);

lseek(fd\_q, 0, SEEK\_SET);

read(fd\_q, buf, fsize);

trim(buf);

capital(buf);

q[c].answer[0] = (char \*)malloc(sizeof(char) \* (fsize+1));

trim(buf);

strcpy(q[c].answer[0], buf);

q[c].answer\_num = 1;

close(fd\_q);

}

}

int j;

// printf("%d\n", c);

return c;

}

int retToken(int i, char \*buf) {

int j =0;

char \*ptr = strtok(buf, ":");

int len;

while(ptr != NULL) {

len = strlen(ptr) + 1;

q[i].answer[j] = (char \*)malloc(sizeof(char) \* len);

strcpy(q[i].answer[j], ptr);

ptr = strtok(NULL,":");

j++;

}

return j;

}

//sort

void sort(int \*list, int count) {

char tem\_str[30];

char tem\_p[30];

int tem, idx, temp\_b;

int i, j, key;

for(i=1; i<count; i++) {

key = list[i];

strcpy(tem\_str, q[i].name);

strcpy(tem\_p, q[i].path);

temp\_b = q[i].blank\_q;

for(j=i-1; j>=0; j--) {

if(list[j]>key){

list[j+1] = list[j];

strcpy(q[j+1].name , q[j].name);

strcpy(q[j+1].path, q[j].path);

q[j+1].blank\_q = q[j].blank\_q;

}else

break;

}

list[j+1] = key;

strcpy(q[j+1].name, tem\_str);

strcpy(q[j+1].path, tem\_p);

q[j+1].blank\_q = temp\_b;

}

}

void compare(char \*STD, int std\_num, int q\_num) {

int i,j, k;

int fd;

int fsize;

char filepath[30];

char forderpath[30];

char temp[30];

char filename[30];

char buf[BUFFER\_SIZE];

for(j=0; j<std\_num; j++){

for(i=QStart; i<q\_num; i++){

for(k=0; k<BUFFER\_SIZE; k++)

buf[k] = '\0';

if(!q[i].blank\_q) {

strcpy(temp, std[j].path);

strcat(temp, q[i].name);

strcat(temp, ".stdout");

if((fd = open(temp, O\_RDONLY)) < 0) {

fprintf(stderr, "file open error %s %d\n", temp, q[i].blank\_q);

exit(1);

}

fsize = lseek(fd, 0, SEEK\_END);

if(fsize > 1024)

break;

lseek(fd, 0, SEEK\_SET);

read(fd, buf, fsize);

trim(buf);

capital(buf);

for(k=0; k<q[i].answer\_num; k++) {

if(strcmp(buf, q[i].answer[k])==0){

// printf("%s vs\n%s\n\n", std[j].answer[i], q[i].answer[0]);

std[j].card[i] = 1;

//printf("정답입니다.\n");

break;

}

}

if(k==q[i].answer\_num) {

//printf("%s vs\n%s\n\n", std[j].answer[i], q[i].answer[0]);

//printf("%s : %s 틀렸습니다.\n", std[j].id, q[i].name);

std[j].card[i] = 0;

}

}

else if(q[i].blank\_q) {

for(k=0; k<q[i].answer\_num; k++) {

if(strcmp(buf, q[i].answer[k])==0){

std[j].card[i] = 1;

break;

}

}

if(k==q[i].answer\_num) {

}

}

}printf("%s is finished\n",std[j].id);

}

}

void execute\_std(int st, int a) {

pthread\_t p\_thread[2];

MultipleArg \*multiple\_arg;

multiple\_arg = (MultipleArg \*)malloc(sizeof(MultipleArg));

int thr\_id;

int status;

char temp[30];

char file[30];

char error[30];

int i = 1;

strcpy(temp, std[st].path); // ./STD/2019000/

strcat(temp, q[a].name);

strcat(temp, ".stdexe"); // ./STD/20190000/11.stdexe

multiple\_arg->tmp = temp;

strcpy(file, std[st].path);

strcat(file, q[a].name);

strcat(file, ".stdout"); // ./STD/20190000/11.stdout

multiple\_arg->filename = file;

strcpy(error, error\_dir); // ./err

strcat(error, "/");

strcat(error, std[st].id); // ./error/20190003/

mkdir(error, 0777);

strcat(error, "/");

strcat(error, q[a].name); // ./error/20190003/11

strcat(error, "\_error.txt"); // ./error/20190003/11.\_error.txt

multiple\_arg->error = error;

thr\_id = pthread\_create(&p\_thread[0], NULL, t\_function, (void\*)multiple\_arg);

if(thr\_id) {

perror("pthread\_creat()");

return;

}

thr\_id = pthread\_create(&p\_thread[1], NULL, control, &p\_thread[0]);

if(thr\_id) {

perror("pthread\_creat()");

return;

}

sleep(KILL\_TIME);

pthread\_cancel(p\_thread[1]);

pthread\_join(p\_thread[0],(void\*\*)&status);

pthread\_join(p\_thread[1], (void\*\*)&status);

free(multiple\_arg);

return;

}

static void killer(void \*arg) {

pthread\_t \*tids = arg;

pthread\_cancel(tids[0]);

}

static void \*control(void \*arg) {

pthread\_cleanup\_push(killer, arg);

do {

pthread\_testcancel();

} while(1);

pthread\_cleanup\_pop(1);

return (void\*)(NULL);

}

void\* t\_function(void\* multiple\_arg) {

int fd, fd\_e;

int fsize;

MultipleArg \*my\_multiple\_arg = (MultipleArg\*)multiple\_arg;

pid\_t pid; // process id

pthread\_t tid; //thread id

// void\* 로 type casting하셔서 넣으시면 됩니다.

clock\_t start, end;

pid = getpid();

tid = pthread\_self();

if((fd = open(my\_multiple\_arg->filename, O\_WRONLY|O\_CREAT|O\_TRUNC, 0777)) < 0) {

fd\_e = open(my\_multiple\_arg->error, O\_WRONLY|O\_CREAT|O\_TRUNC, 0777);

dup2(fd\_e, 2);

fprintf(stderr, "480 open error %s\n", my\_multiple\_arg->filename);

close(2);

close(fd\_e);

exit(1);

}

if(dup2(fd, 1) != 1) {

fprintf(stderr,"dup2 fail\n");

exit(1);

}

if(system(my\_multiple\_arg->tmp)!=0) { //if "temp"'s running time is longer than 5 sec, kill "temp"

fd\_e = open(my\_multiple\_arg->error, O\_WRONLY|O\_CREAT|O\_TRUNC, 0777);

dup2(fd\_e, 2);

fprintf(stderr, "system fail!! %s\n", my\_multiple\_arg->tmp);

close(1);

close(fd);

close(2);

close(fd\_e);

return (void\*)(NULL);

}

else {

close(1);

close(fd);

}

return (void\*)(NULL);

}

void creat\_score\_table(char \*ANS,int q\_num) {

FILE \*fp;

int i,j,n;

float b\_value, p\_value;

char tmp[30];

strcpy(tmp, ANS);

strcat(tmp, "/score\_table.csv");

//score\_table 생성

printf("score\_table.csv file doesn't exist in TREUDIR!\n1. input blank question and program question's score. ex) 0.5 1\n2. input all question's score. ex)Input value of 1-1: 0.1\nselect type >>");

if((fp = fopen(tmp,"w+")) == NULL) {

fprintf(stderr, "fopen error for %s\n", tmp);

exit(1);

}

int type;

scanf("%d", &type);

printf("type : %d\n", type);

//value 입력하기

switch(type) {

case 1 :

printf("Input value of blank question : ");

scanf("%f", &b\_value);

printf("Input value of prograam question : ");

scanf("%f", &p\_value);

for(i=3; i<q\_num; i++) {

//빈칸문제

if(q[i].blank\_q) {

q[i].score = b\_value;

fprintf(fp, "%s,%.2f\n", q[i].name,b\_value);

}

//프로그래밍 문제

else if(!q[i].blank\_q) {

q[i].score = p\_value;

fprintf(fp, "%s,%.2f\n", q[i].name,p\_value);

}

}

break;

case 2 :

//value 입력받기

for(i=0; i<=q\_num; i++) {

printf("Input of %s: ", q[i].name);

getchar();

scanf("%f", &q[i].score);

getchar();

}

for(i=QStart; i<=q\_num; i++)

fprintf(fp, "%s,%.2f\n", q[i].name, q[i].score);

break;

}

fclose(fp);

return;

}

void creat\_score(char \*ANS, int std\_num, int q\_num){

FILE \*fp;

int i, j;

float sum = 0;

float temp;

char tmp[30];

strcpy(tmp, ANS);

strcat(tmp, "/score.csv");

if((fp = fopen(tmp, "w+")) == NULL) {

fprintf(stderr,"fopen error for %s\n", "./ANS/score.csv");

exit(1);

}

fprintf(fp, " ,");

for(i=QStart; i<q\_num; i++) {

fprintf(fp, "%s,",q[i].name);

}

fprintf(fp, "%s\n", "sum");

for(j=0; j<std\_num; j++) {

sum=0;

fprintf(fp, "%s,", std[j].id);

for(i=QStart; i<q\_num; i++) {

if(std[j].card[i] == 1){

sum += q[i].score;

fprintf(fp, "%.2f,", q[i].score);

}

else

fprintf(fp, "0.00,");

}

fprintf(fp, "%.2f\n", sum);

}

fclose(fp);

}

int creat\_std\_table(char \*STD, int q\_num){

//STD 학번넣기 std struct에 넣기

struct dirent \*\*stdlist;

struct dirent \*\*filelist;

int count\_s, count\_f;

int i, j, k, s, fd;

char buf[BUFFER\_SIZE];

int length;

char temp[30];

char err[30];

if((count\_s = scandir(STD, &stdlist, NULL, alphasort)) == -1){

fprintf(stderr, "%s directory scan Error\n", "./STD/");

exit(1);

}

for(s=0; s<count\_s-2; s++) {

i = s+2;

strcat(std[s].id, stdlist[i]->d\_name);

//printf("stdid : %s\n", std[s].id);

for(j=QStart; j<q\_num; j++) {

strcpy(std[s].path, STD);

strcat(std[s].path, "/");

strcat(std[s].path, std[s].id); //./STD/20162467

strcat(std[s].path, "/"); // ./STD/20162467/

if(q[j].blank\_q == 1) {//빈칸문제일 경우

strcpy(temp, std[s].path);

strcat(temp, q[j].name); // ./STD/20162467/1-1

strcat(temp, ".txt");

if((fd = open(temp, O\_RDONLY|O\_CREAT)) < 0) {

fprintf(stderr,"open error for %s\n", std[s].path);

exit(1);

}

length = lseek(fd, 0, SEEK\_END);

lseek(fd, 0, SEEK\_SET);

read(fd, buf, length);

std[s].answer[j] = (char \*)malloc(length+1);

trim(buf);

strcpy(std[s].answer[j],buf);

for(k=0; k<BUFFER\_SIZE; k++){

buf[k] = '\0';

}

close(fd);

}

else if(q[j].blank\_q == 0){//프로그래밍 문제일 경우

execute\_std(s,j);

continue;

}

}

// printf("280\n");

}

// printf("282\n");

/\* for(i=0; i<count\_s; i++){

free(stdlist[i]);

}\*/

free(stdlist);

return s;

}

void trim(char\* s) {

char t[MAX\_STR\_LEN];

char \*end;

for(; \*s!= '\0'; s++){

if(\*s == '\n'){

strcpy(s, s+1);

s--;

}

if(\*s == '\0'){

strcpy(s, s+1);

s--;

}

}

}

void capital(char \*s) {

int i ;

for(i=0; s[i] == '\0'; i++) {

if(s[i]>='a' && s[i]<= 'z')

s[i] = s[i]-32;

}

}