설계과제 2 개요 : SSU-Score

○ 개요

- ssu_score은 정답 파일을 기준으로 학생들이 제출한 답안 파일을 채점하는 프로그램이다.

○ 목표

- 유닉스/리눅스 컴퓨팅 환경에서 제공하는 파일 입출력, 파일속성 디렉토리에 관한 시스템 호출 함수와 라이브러리 함수를 이용한 프로그램을 작성하여 시스템 프로그램 설계 및 응용 능력을 향상시킨다.

○ 팀 구성

- 개인별 프로젝트

○ 보고서 제출 방법

- 설계과제는 "**#P설계과제번호_학번.zip**"(예. #P2_20140000_V1.zip)형태로 압축하여 classroom.google.com에 제출해야 함.
- "#P설계과제번호_학번.zip" 내 보고서인 "#P설계과제번호.hwp" 와 헤더파일 및 소스코드 등 해당 디렉토리에서 컴파일과 실행이 가능하도록 모든 파일(makefile, obj, *.c, *.h 등 컴파일하고 실행하기 위한 파일들)을 포함시켜야 함. 단. 특정한 디렉토리에서 실행해야 할 경우는 예외.
- 구현보고서인 "#P설계과제번호.hwp"에는 1. 과제개요(명세에서 주어진 개요를 그대로 쓰면 안 됨. 자기가 구현한 내용 요약) 2. 기능(구현한 기능 요약), 3. 상세설계(함수 및 모듈 구성, 순서도, 구현한 함수 프로토타입 등), 4. 실행결과 (구현한 모든 기능 및 실행 결과 캡쳐)를 반드시 포함시켜야 함.
- 과제를 기한 내 새로 제출할 경우 기존 것은 삭제하지 않고 #P설계과제번호_학번_V1.0.zip 형태로 버전 이름만 붙이면 됨. 버전 이름은 대문자 V와 함께 integer를 1부터 incremental 증가시키면서 부여하면 됨. (예. V1, V2, V3, ...) 단, 처음 제출 시는 버전 번호를 붙이지 않아도 되며 두 번째부터 V1를 붙여 제출하면 됨. 단, #P설계과제번호_학번_V1.0.zip 압축 파일 내 구현보고서에는 버전 이름을 붙이지 않아도 됨.
- 제출한 압축 파일을 풀었을 때 해당 디렉토리에서 컴파일 및 실행이 되어야 함(특정한 디렉토리에서 실행해야 할 경우는 제외). 해당 디렉토리에서 컴파일이나 실행되지 않을 경우, 기본과제 및 설계과제 제출 방법(파일명, 디렉토리면, 컴파일 시에 포함되어야 할 파일 등)을 따르지 않으면 무조건 해당 과제 배점의 50% 감점
  - ✓ 설계과제의 기준 및 각 과제별 "필수기능요건"은 각 설계 과제 명세서에 별도 설명. 설계과제명세서와 강의계획서 상 배점 기준이 다를 경우 해당 설계과제 명세서의 배점 기준이 우선 적용
  - ✓ 보고서 #P설계과제번호.hwp (15점) : 개요 1점, 기능 1점, 상세설계 10점, 실행 결과 3점
  - ✓ 소스코드 (85점) : 소스코드 주석 5점, 실행 여부 80점 (설계 요구에 따르지 않고 설계된 경우 소스코드 주석 및 실행 여부는 0점 부여. 설계 요구에 따라 설계된 경우 기능 미구현 부분을 설계명세서의 100점 기준에서 해당 기능 감점 후 이를 80점으로 환산)
- 기타 내용은 강의계획서 참고

○ 제출 기한

- 5월 3일(수) 오후 11시 59분 59초

○ 보고서 및 소스코드 배점

- 보고서는 다음과 같은 양식으로 작성(강의계획서 FAQ 참고)

```
1. 과제 개요 (1점) // 명세에 주어진 개요를 더 상세하게 작성
2. 구현 기능 (1점) // 함수 프로토타입 반드시 포함
3. 상세 설계 (10점) // 함수 기능별 흐름도(순서도) 반드시 포함
4. 실행 결과 (3점) // 테스트 프로그램의 실행결과 캡쳐 및 분석
```

- 소스코드 및 실행 여부 (85점) // 주석 (5점), 실행 여부 (80점)

○ ssu_score 프로그램 기본 사항

- 출제되는 시험 문제는 **(1)빈칸 채우기 문제**와 **(2)프로그램 작성 문제** 총 두 가지 종류이며 학생들이 출제되는 시험문제에

대한 답을 작성하여 자신의 학번 디렉토리 안에 모든 문제에 대한 답안 파일을 넣어 제출함

- 빈칸 채우기 문제의 답안 파일 형태는 문제번호.txt(예. 5.txt) 이며, 프로그램 작성 문제의 답안 파일 형태는 문제번호.c (예. 10.c)임
- 단, 하나의 빈칸 채우기 문제는 여러 개의 서브 문제로 구성될 수 있으며 각 학생들은 한 문제가 여러 개의 하위 문제들로 구성될 수 있음 (예. 1-1.txt, 1-2.txt)
- 정답 파일도 학생 답안과 동일하게 하나의 디렉토리 안에 빈칸문제(문제번호.txt)와 프로그램문제(문제번호.c / 문제번호.exe / 문제번호.stdout 등)로 구성
- 출제되는 문제 수는 최대 100문제까지 허용
- ssu_score은 링크드 리스트로 파일 리스트를 관리해야 함
  - ✓ 리눅스 상에서 파일 경로의 최대 크기는 4,096 바이트이며, 파일 이름의 최대 크기는 255 바이트임
- 실행결과 및 에러 메시지를 파일에 출력 시 system()함수 사용 금지
- popen(), fork(), exec() 계열 함수 사용 금지
- 게시판 통해 일부 소스코드 직접 다운로드
- getopt() 라이브러리를 사용하여 옵션 처리 권장
- ssu_score은 foreground로만 수행되는 것으로 가정함(& background 수행은 안되는 것으로 함)

○ 설계 및 구현

ssu_score

1) Usage : ssu_score 〈STD_DIR〉 〈ANS_DIR〉 [OPTION]
- ssu_score 실행 시 출제 시험 문제에 대해 학생 답안 파일을 정답 파일과 비교하여 채점을 수행

2) 인자 설명
- 〈STD_DIR〉 : 채점 대상이 될 학생이 제출한 답안 디렉토리로 학생들 학번 서브디렉토리를 포함 (최대 100개)
- 〈ANS_DIR〉 : 채점 기준이 될 정답이 있는 디렉토리로 빈칸 채우기 문제의 경우 정답 리스트(문제번호.txt, 다수의 답이 있을 경우 콜론으로 구분), 프로그램 문제의 경우(정답 소스 프로그램(문제번호.c), 정답 실행프로그램(문제번호.exe), 정답 실행프로그래밍 실행결과(문제번호.stdout) 파일 포함

3) 실행결과
- 출제 시험 문제(빈칸 채우기 문제, 프로그램 작성 문제)에 대하여 학생이 제출한 답안과 정답을 비교하여 채점을 실시함
- 모든 채점이 끝나면(채점 결과 파일 가져오는 거 아님) 프로그램 수행시간(sec:usec) 출력 후 프로그램 종료
- 출제 시험 문제 형태 예시

○ 빈칸 채우기 문제

1) 예시
- 다음 프로그램은 dup2()를 호출하여 표준 출력 1번 파일 디스크립터를 4번으로 복사하고 4번 파일 디스크립터를 인자로 하여 write()를 호출하면 표준 출력에 쓰는 것과 같은 결과를 보여준다. 아래 실행결과를 보고 빈칸을 채우시오.

ssu_test.txt

```
Linux System Programming!
Unix System Programming!
Linux Mania
Unix Mania
```

프로그램

```
#include 〈stdio.h〉
#include 〈stdlib.h〉
#include 〈unistd.h〉
#include 〈fcntl.h〉

#define BUFFER_SIZE 1024

int main(void) {
  char buf[BUFFER_SIZE];
```

```
char *fname = "ssu_test.txt";
int fd;
int length;

if((fd = open( 1-3.txt )) < 0) {
  fprintf(stderr, "open error for %s₩n", fname);
  exit(1);
}

if( 1-1.txt ) {
  fprintf(stderr, "dup2 call failed₩n");
  exit(1);
}

while(1) {
  length = 1-2.txt ;

  if(length <= 0)
    break;

  1-4.txt ;
}

exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./a.out
Linux System Programming!
Unix System Programming!
Linux Mania
Unix Mania
```

2) 채점 방법

- 학생들이 제출한 답안 디렉토리의 txt 파일들과 정답 파일의 txt 파일을 비교하면서 채점하고 맞으면 주어진 정답 파일에서 주어진 점수를 부여하고 틀리면 0점 부여
- 포인터 배열, 캐스팅 연산처럼 다수의 답이 정답이 될 경우 정답파일에 '콜론(:)'으로 구분하며, 학생들이 작성한 답안 파일을 비교할 때 다수의 답을 모두 정답으로 채점 -> 정답 파일에 새로운 답을 추가 및 삭제를 통해 학생들의 답안을 채점

〈예〉 정답 파일	〈예〉 학생 답안 파일
creat(fname, CREAT_MODE):creat(fname, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH):creat(fname, 0644)	creat(fname, CREAT_MODE)
	creat(fname, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)
	creat(fname, 0644)

- 연산자 사용으로 인해 다수의 답이 정답이 되는 경우 자동으로 동일한 연산식으로 자동으로 채점
- 단, 피연산자들의 순서가 바뀌었을 때 결과 값이 달라지는 문제는 지원하지 않음
- 연산자 우선순위을 기반으로 한 괄호도 정답으로 처리
- 문자열 간 공백은 구분하지 않음 -> 컴파일 시 에러가 나지 않는 경우 공백은 정답으로 처리

〈예〉 정답 파일	〈예〉 학생 답안 파일
value < STOP1 || value > STOP2	value < STOP1 || value > STOP2
	STOP1 > value || value > STOP2

	(value 〈 STOP1) || (STOP2 〈 value)
	(STOP1 〉 value) || (STOP2 〈 value)
	value 〉 STOP2 || value 〈 STOP1
	value 〉 STOP2 || STOP1 〉 value
	(value 〈 STOP1) || (STOP2 〈 value)
	(STOP1 〉 value) || (STOP2 〈 value)

〈예〉 정답 파일	〈예〉 학생 답안 파일
value 〈= 3 && 6 〉= value	6 〉= value && value 〈= 3
	3 〉= value && value 〈= 6
(fd1=open(filename,O_RDWR|O_APPEND, 0644))==-1	-1==(fd1=open(filename,O_RDWR|O_APPEND,0644))
	(fd1=open(filename,O_APPEND|O_RDWR,0644))==-1

〈예〉 정답 파일	〈예〉 학생 답안 파일
creat(filename, S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH)	open(filename, O_WRONLY|O_CREAT|O_TRUNC, 0666)
(fd1=open(filename,O_RDWR|O_APPEND, 0644))==-1	-1==(fd1=open(filename,O_RDWR|O_APPEND,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH)

○ 프로그램 문제

1) 예시

- 다음은 부모 프로세스에서 입력 받은 메시지를 2개의 자식 프로세스를 통해 화면에 출력하는 프로그램이다. 동일한 실행결과가 나오도록 〈조건〉에 알맞게 소스코드를 작성하시오.

〈 조 건 〉

1. 부모와 자식 프로세스간의 메시지를 전달하기 위해 공유메모리를 생성
2. key와 id값을 생성한 후 전역변수 data에 공유메모리 생성(key 생성을 위한 인자들은 임의로 설정)
3. 공유 메모리의 마지막 1byte는 부모와 자식 프로세스 간의 동기화를 위해 사용
4. 부모 프로세스에서 입력 받은 메시지의 크기는 127byte를 넘지 않는 것으로 가정

부모 프로세스

1. 자식 프로세스 2개 생성
2. "parent : " 출력 후 메시지를 입력받음
3. 입력 받은 메시지를 공유메모리에 저장 후 자식 프로세스들에게 SIGUSR1을 전달
4. 입력 받은 메시지가 "exit"일 경우 SIGKILL을 자식 프로세스들에게 전달 후 종료
5. 두 개의 자식 프로세스에서 부모 프로세스가 전달한 메시지를 출력하기 까지 공유메모리의 마지막 1byte를 계속 검사하면서 대기
   ※ 두 개의 자식 프로세스는 공유메모리 마지막 1byte에 메시지를 출력했다는 표시를 함 (표시 및 검사 방법은 자유)
6. [2."parent : " 출력 후 메시지를 입력받음] 반복

자식 프로세스

1. SIGUSR1 시그널에 대한 핸들러를 등록 한 후 sleep(1)을 호출하는 무한루프
2. 부모 프로세스로부터 받은 메시지를 화면에 출력 후 공유 메모리 data의 가장 마지막 1byte에 출력 했다는 표시

void ssu_signal_handler1(int signo)

- 첫 번째 자식 프로세스에서 SIGUSR1 시그널에 대한 핸들러로 사용하는 함수
- "child1 : 부모로부터 받은 메시지"를 출력

void ssu_signal_handler2(int signo)

- 두 번째 자식 프로세스에서 SIGUSR1 시그널에 대한 핸들러로 사용하는 함수
- “child2 : 부모로부터 받은 메시지”를 출력

33.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHM_MEM_SIZ 128
#define BUF_SIZ 128
void ssu_signal_handler1(int signo);
void ssu_signal_handler2(int signo);
char *data;

int main(void) {
  key_t key;
  pid_t pid[2];
  char buf[BUF_SIZ];
  int shm_id;
  int n;
}

void ssu_signal_handler1(int signo){
}

void ssu_signal_handler2(int signo){
}
```

실행결과

```
oslab@oslab:~/lsp_fin$ ./final_번호
parent : hello world
child1 : hello world
child2 : hello world
parent : lsp final exam
child1 : lsp final exam
child2 : lsp final exam
parent : su go
child2 : su go
child1 : su go
parent : exit
oslab@localhost:~/lsp_fin$
```

2) 채점 방법

- 학생이 작성해서 답안으로 제출한 프로그램을 컴파일 후 실행파일은 “./STD/학번/문제번호.stdexe” 이름으로 저장 그리고 정답 프로그램의 실행 파일은 “./ANS/문제번호.exe” 이름으로 생성
- 각 문제에 대한 정답 프로그램인 “./ANS/문제번호.c”의 실행파일인 ”./ANS/문제번호.exe”를 실행시켜 실행결과를 “./ANS/문제번호.stdout”으로 저장
- 학생이 작성해서 답안으로 제출한 프로그램의 실행파일인 “./STD/학번/문제번호.stdexe“을 자동으로 실행시키고 실행결과를 “./STD/학번/문제번호.stdout“ 자동으로 저장

- “./ANS/문제번호.stdout”를 학번 디렉토리를 순회하면서 “./STD/학번/문제번호.stdout”과 비교하면서 결과만 우선 채점
- 학생들이 제출한 프로그램의 실행 결과 채점 시 실행결과의 대소문자와 공백을 구분하지 않음
- 학생들이 제출한 프로그램의 실행 결과 채점 시 warning이 발생한 문제는 -0.1점 (#define으로 정의 변경이 용이하게) 처리, error가 하나라도 있는 경우 발생한 문제는 0점 (#define으로 정의 - 변경이 용이하게) 처리
- 학생들이 제출한 프로그램의 실행이 5초 이상의 시간이 걸리는 프로그램은 0점(#define으로 정의 - 변경이 용이하게) 처리함
- 학생들이 제출한 프로그램의 실행결과 파일의 크기에 상관없이 채점 가능해야 함

○ 점수 테이블 생성
- 점수 테이블 파일은 “./ANS/score_table.csv” 이름으로 생성해야 하며, 문제번호와 점수를 저장
- 점수 테이블 파일이 “./ANS/”에 존재해야 하며, 존재하지 않은 경우 “./ANS/score_table.csv” 이름으로 점수 테이블 파일을 새로 생성하고, 사용자가 문제번호와 점수를 설정하고 이를 저장할 수 있어야 함

실행 예. 점수 테이블 파일 생성

```
oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET
score_table.csv file doesn't exist in TREUDIR!
1. input blank question and program question's score. ex) 0.5 1
2. input all question's score. ex) Input value of 1-1: 0.1
select type >> 1
Input value of blank question : 0.8
Input value of program question : 1.2

oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET
score_table.csv file doesn't exist in TREUDIR!
1. input blank question and program question's score. ex) 0.5 1
2. input all question's score. ex) Input value of 1-1: 0.1
select type >> 2
Input of 1-1.txt: 0.3
Input of 1-2.txt: 0.5
Input of 1-3.txt: 0.3
Input of 1-4.txt: 0.4
Input of 1-5.txt: 0.2
```

예. 점수 테이블 파일 형식

	A	B
1	1-1.txt	0.1
2	1-2.txt	0.1
3	1-3.txt	0.5
4	2-1.txt	0.5
5	2-2.txt	0.5
6	3.c	1
7	4.c	1
8	5.c	1

- 점수 테이블 파일(score_table.csv)가 〈ANS_DIR〉에 존재하지 않을 경우 “score_table.csv file doesn’t exist in “〈ANS_DIR〉”!” 출력 후 사용자의 입력에 따라 점수 테이블 파일 생성 및 저장
  - ✓ 1 입력 시 빈칸 채우기 문제와 프로그램 문제, 두 종류의 점수만 설정
  - ✓ 2 입력 시 각 문제마다 점수를 설정

실행 예. (옵션 없는 경우)

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET
score_table.csv file doesn’t exist in “/home/oslab/P2/TRUESET”!
1. input blank question and program question’s score. ex) 0.5 1
2. input all question’s score. ex) Input value of 1-1: 0.1
select type 〉〉 1
Input value of blank question : 0.5
Input value of program question : 1
```

```
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

○ 채점 결과 테이블 생성

- 학생들의 성적 테이블은 ssu_score 프로그램이 종료하면 " ./ANS/score.csv" 이름으로 자동으로 생성되어야 하며, (1)학번 (2) 문제번호당 채점 점수 (3) 총점을 저장하여 전체 학생들의 각 문제당 점수와 전체 점수를 알 수 있음

예. 채점 결과 파일 형식

	A	B	C	D	E	F	G	H	I	J
1		1-1.txt	1-2.txt	1-3.txt	2-1.txt	2-2.txt	3.c	4.c	5.c	sum
2	20190001	0.1	0.1	0	0	0.5	1	0	0	1.7
3	20190002	0	0.1	0	0	0.5	1	0	0	1.6
4	20190003	0	0.1	0.5	0.5	0	1	1	0	3.1
5	20190004	0	0.1	0.5	0.5	0	1	1	1	4.1
6	20190005	0	0.1	0	0	0	1	0	0	1.1
7	20190006	0.1	0.1	0	0.5	0.5	1	0	1	3.2
8	20190007	0.1	0	0	0	0.5	0	1	1	2.6
9	20190008	0	0.1	0.5	0.5	0.5	0	1	1	3.6
10	20190009	0.1	0	0.5	0.5	0	1	1	0	3.1

4) 예외처리(스스로 판단하여 추가 구현, 필수 구현은 아님)

- 존재하지 않는 디렉토리 혹은 파일을 대상으로 할 경우 에러 처리 후 프로그램 종료
- 모든 학생의 채점을 완료 후 "result saved.."와 함께 채점 결과 파일 저장 경로를 출력하고 프로그램 종료
- 점수 출력은 소수점 둘째자리까지 출력 등

옵션 1. -n

1) Usage : -n 〈CSVFILENAME〉

- 채점 결과 테이블 생성 시 기본적으로 생성되는 score.csv가 아닌 〈CSVFILENAME〉 파일명으로 테이블 생성

2) 인자 설명

- 첫 번째 인자 〈CSVFILENAME〉은 상대경로와 절대경로 모두 입력 가능해야 함

3) 실행 결과

실행 예. -n 옵션

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -n new_score.csv
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/new_score.csv)
Runtime: 10:125432(sec:usec)
```

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)
- 첫 번째 인자의 경로가 .csv파일이 아닐 경우 에러 처리 후 프로그램 종료
- 첫 번째 인자의 경로가 이미 있는 파일일 경우 덮어씌움

옵션 2. -m

1) Usage : -m
- 점수 테이블 파일 내의 특정 문제 번호의 배점 수정

2) 실행 결과

실행 예. -m 옵션

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -m
Input question's number to modify >> 3-1
Current score : 0.2
New score : 0.1
Input question's number to modify >> no
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

3) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)
- 점수 테이블 파일이 존재하지 않을 경우 에러 처리 후 프로그램 종료

옵션 3. -c

1) Usage : -c [STUDENTIDS ...]
- 인자로 입력받은 학생들의 점수 출력

2) 인자 설명
- 첫 번째 인자 [STUDENTIDS ...]는 가변인자로써 여러 개 입력 가능
- 첫 번째 인자 생략 시 모든 학생에 대하여 결과 출력

3) 실행 결과
- 각 학생의 점수 출력 후 학생들 점수에 대한 평균을 "Total average : 〈점수 평균〉" 형태로 출력

실행 예. -c 옵션

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -c 20190002 20190003 20190005
grading student's test papers..
20190001 is finished..
20190002 is finished.. score : 73.50
20190003 is finished.. score : 72.00
20190004 is finished..
20190005 is finished.. score : 67.50
20190006 is finished..
20190007 is finished..
Total average : 71.00
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)
- 첫 번째 인자 입력 시 가변인자 개수는 5개로 제한. 그 이상의 가변인자 입력 시 "Maximum Number of Argument Exceeded. :: [STUDENTIDS ...]" 출력 후 수행에는 반영하지 않음

- 첫 번째 인자로 입력받은 학생이 〈STD_DIR〉에 없는 경우 에러 처리 후 프로그램 종료

옵션 4. -p

1) Usage : -p [STUDENTIDS ...]
- 인자로 입력받은 학생들의 틀린 문제 번호(점수) 출력

2) 인자 설명
- 첫 번째 인자 [STUDENTIDS ...]는 가변인자로써 여러 개 입력 가능함
- 첫 번째 인자 생략 시 모든 학생에 대하여 결과 출력
- 틀린 문제 출력은 각 학생의 채점이 끝난 이후 수행되며 링크드 리스트로 틀린 문제 목록을 관리해야 함

3) 실행 결과

실행 예. -p 옵션

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -p 20190002 20190003 20190005
grading student's test papers..
20190001 is finished..
20190002 is finished.. wrong problem : 1-3(3), 2-2(2), 2-3(2), 3-2(1.5), 3-3(3), 5-1(4), 8(6), 10(5)
20190003 is finished.. wrong problem : 1-3(3), 2-2(2), 2-3(2), 3-1(3), 3-3(3), 5-1(4), 8(6), 10(5)
20190004 is finished..
20190005 is finished.. wrong problem : 1-3(3), 2-2(2), 2-3(2), 3-1(3), 3-2(1.5), 3-3(3), 5-1(4), 6-2(4), 8(6), 10(5)
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)
- 첫 번째 인자 입력 시 가변인자 개수는 5개로 제한. 그 이상의 가변인자 입력 시 "Maximum Number of Argument Exceeded. :: [STUDENTIDS ...]" 출력 후 수행에는 반영하지 않음
- 첫 번째 인자로 입력받은 학생이 〈STD_DIR〉에 없는 경우 에러 처리 후 프로그램 종료

5) 주의할 점
- -p 옵션을 -c 옵션과 함께 사용할 경우 점수 먼저 출력하고 틀린 문제 번호(점수)를 출력함. [STUDENTIDS ...] 가변인자 그룹은 -c 옵션 혹은 -p 옵션 뒤에 한번만 나와야 함. 만약 가변인자 그룹이 두 번 나올 경우 에러 처리 후 프로그램 종료(-t 옵션의 〈QNAMES ...〉 가변인자 그룹은 별개로 취급) 등

실행 예. -p 옵션 -c 옵션 동시 사용

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -c -p 20190002 20190003 20190005
grading student's test papers..
20190001 is finished..
20190002 is finished.. score : 73.50, wrong problem : 1-3(3), 2-2(2), 2-3(2), 3-2(1.5), 3-3(3), 5-1(4), 8(6), 10(5)
20190003 is finished.. score : 72.00, wrong problem : 1-3(3), 2-2(2), 2-3(2), 3-1(3), 3-3(3), 5-1(4), 8(6), 10(5)
20190004 is finished..
20190005 is finished.. score : 67.50, wrong problem : 1-3(3), 2-2(2), 2-3(2), 3-1(3), 3-2(1.5), 3-3(3), 5-1(4), 6-2(4), 8(6), 10(5)
20190006 is finished..
20190007 is finished..
Total average : 71.00
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

옵션 5. -t

1) Usage : -t [QNAMES ...]

- [QNAMES ...]을 문제 번호로 하는 문제는 컴파일 시 -lpthread 옵션 추가

2) 인자 설명

- 첫 번째 인자 [QNAMES ...]는 가변인자로써 여러 개 입력 가능함
- 첫 번째 인자 생략 시 모든 문제 번호에 대하여 컴파일 실행

3) 실행 결과

실행 예. -t 옵션 적용 안 한 경우

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -c
grading student's test papers..
20190001 is finished.. score : 70.50
20190002 is finished.. score : 73.50
20190003 is finished.. score : 72.00
20190004 is finished.. score : 70.50
20190005 is finished.. score : 67.50
20190006 is finished.. score : 73.50
20190007 is finished.. score : 73.50
Total average : 71.57
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

실행 예. -t 옵션 적용 한 경우

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -c -t 12 15
grading student's test papers..
20190001 is finished.. score : 78.50
20190002 is finished.. score : 81.50
20190003 is finished.. score : 80.00
20190004 is finished.. score : 70.50
20190005 is finished.. score : 67.50
20190006 is finished.. score : 81.50
20190007 is finished.. score : 73.50
Total average : 76.14
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

실행 예. 가변인자 최대 개수를 초과한 경우의 출력

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -c 20190001 20190002 20190003 20190004
20190005 20190006 20190007 -t 11 12 13 14 15 16
Maximum Number of Argument Exceeded. :: 20190006 20190007
Maximum Number of Argument Exceeded. :: 16
grading student's test papers..
20190001 is finished.. score : 73.50
20190002 is finished.. score : 76.50
20190003 is finished.. score : 75.00
20190004 is finished.. score : 70.50
20190005 is finished.. score : 67.50
20190006 is finished..
20190007 is finished..
Total average : 72.60
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)

- 첫 번째 인자 입력 시 없는 문제 번호일 경우 에러 처리 후 프로그램 종료
- 첫 번째 인자 입력 시 가변인자 개수는 5개로 제한. 그 이상의 가변인자 입력 시 "Maximum Number of Argument Exceeded. :: [QNAMES ...]" 출력 후 수행에는 반영하지 않음 등

옵션 6. -s

1) Usage : -s 〈CATEGORY〉 〈1|-1〉
- 채점결과 파일을 〈CATEGORY〉를 기준으로 링크드 리스트를 사용하여 정렬한 후 저장함

2) 인자 설명
- 첫 번째 인자 〈CATEGORY〉는 학번(stdid), 점수(score) 입력 가능
- 두 번째 인자 〈1|-1〉은 1 입력 시 오름차순, -1 입력시 내림차순으로 정렬

3) 실행 결과
- 첫 번째 인자로 score 입력 시 총 점수를 기준으로 정렬함

실행 예. -s 옵션 (학번 기준 오름차순 정렬)

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -s stdid 1
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

채점 결과 csv(/home/oslab/P2/TRUESET/score.csv) 예시

	A	B	C	D	E	F	G	H	I	J
1		1-1.txt	1-2.txt	1-3.txt	2-1.txt	2-2.txt	3.c	4.c	5.c	sum
2	20190001	0.1	0.1	0	0	0.5	1	0	0	1.7
3	20190002	0	0.1	0	0	0.5	1	0	0	1.6
4	20190003	0	0.1	0.5	0.5	0	1	1	0	3.1
5	20190004	0	0.1	0.5	0.5	0	1	1	1	4.1
6	20190005	0	0.1	0	0	0	1	0	0	1.1
7	20190006	0.1	0.1	0	0.5	0.5	1	0	1	3.2
8	20190007	0.1	0	0	0	0.5	0	1	1	2.6

실행 예. -s 옵션 (점수 기준 내림차순 정렬)

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -s score -1
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/TRUESET/score.csv)
Runtime: 10:125432(sec:usec)
```

채점 결과 csv(/home/oslab/P2/TRUESET/score.csv) 예시

	A	B	C	D	E	F	G	H	I	J
1		1-1.txt	1-2.txt	1-3.txt	2-1.txt	2-2.txt	3.c	4.c	5.c	sum
2	20190004	0	0.1	0.5	0.5	0	1	1	1	4.1
3	20190006	0.1	0.1	0	0.5	0.5	1	0	1	3.2
4	20190003	0	0.1	0.5	0.5	0	1	1	0	3.1
5	20190007	0.1	0	0	0	0.5	0	1	1	2.6
6	20190001	0.1	0.1	0	0	0.5	1	0	0	1.7
7	20190002	0	0.1	0	0	0.5	1	0	0	1.6
8	20190005	0	0.1	0	0	0	1	0	0	1.1

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)
- 첫 번째 인자로 stdid나 score 외에 다른 입력이 들어오면 에러 처리 후 프로그램 종료
- 두 번째 인자로 1 혹은 -1 외에 다른 입력이 들어오면 에러 처리 후 프로그램 종료 등

옵션 7. -e

1) Usage : -e 〈DIRNAME〉
- 〈DIRNAME〉/학번/문제번호_error.txt에 에러 메시지가 출력

2) 인자 설명
- 첫 번째 인자 〈DIRNAME〉는 상대경로와 절대경로 모두 입력 가능해야 함

3) 실행 결과
- 〈DIRNAME〉 폴더가 없을 경우 생성하고 각 학생들에 대한 서브디렉토리를 생성함
- 서브디렉토리 안에는 에러가 난 문제들에 대해 txt파일로 에러 메시지를 관리함

실행 예. -e 옵션

```
oslab@oslab:~/P2$ ./ssu_score STUDENT TRUESET -e error
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
result saved.. (/home/oslab/P2/TRUESET/score.csv)
error saved.. (/home/oslab/P2/error/)
Runtime: 10:125432(sec:usec)
oslab@oslab:~/P2$ ls error/20190005/
11_error.txt 13_error.txt 14_error.txt
oslab@oslab:~/P2$ cat error/20190005/11_error.txt
/usr/lib/gcc/i686-linux-gnu/5/../../../i386-linux-gnu/crt1.o: In function `_start':
(.text+0x18): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)
- 첫 번째 인자의 경로가 잘못되었을 경우 에러처리 후 프로그램 종료
- 에러 파일 생성 시 오류 발생 시 에러처리 후 프로그램 종료

5) 주의할 점
- 같은 에러 디렉토리에 대해서 덮어씌울 경우, 안에 내용을 모두 삭제하고 다시 에러 파일을 만듦(이전 결과 초기화)

옵션 8. -h

1) Usage : -h
- 사용법 출력

2) 인자 설명
- -h 옵션은 〈STD_DIR〉, 〈ANS_DIR〉 없이 사용 가능

3) 실행 결과

실행 예. -h 옵션

```
oslab@oslab:~/P2$ ./ssu_score -h
Usage : ssu_score <STD_DIR> <ANS_DIR> [OPTION]
Option :
 -n <CSVFILENAME>
 -m
 -c [STUDENTIDS ...]
 -p [STUDENTIDS ...]
 -t [QNAMES ...]
 -s <CATEGORY> <1|-1>
 -e <DIRNAME>
 -h
```

4) 예외 처리(스스로 판단하여 추가 구현, 필수 구현은 아님)

- 다른 옵션과 동시에 사용할 경우 사용법을 출력 후 ssu_score 프로그램 종료

○ 과제 구현에 필요한 함수 (필수 아님)

- 1. getopt() : 프로그램 실행 시 입력한 인자를 처리하는 라이브러리 함수

```
#include <unistd.h>
int getopt(int argc, char * const argv[], const char *optstring); //_POSIX_C_SOURCE

#include <getopt.h>
int getopt_long(int argc, char * const argv[], const char *optstring, const struct option *longopts, int *longindex);
//_GNU_SOURCE
```

- 2. scandir : 디렉토리에 존재하는 파일 및 디렉토리 전체 목록 조회하는 라이브러리 함수

```
#include <dirent.h>
int scandir(const char *dirp, struct dirent ***namelist, int (*filter)(const struct dirent *), int (*compar)(const struct
dirent **, const struct dirent **));

-1 : 오류가 발생, 상세한 오류 내용은 errno에 설정
0 이상 : 정상적으로 처리, namelist에 저장된 struct dirent *의 개수가 return
```

- 3. realpath : 상대경로를 절대경로로 변환하는 라이브러리 함수

```
#include <stdlib.h>
char *realpath(const char *path, char *resolved_path);

NULL : 오류가 발생, 상세한 오류 내용은 errno 전역변수에 설정
NULL이 아닌 경우 : resolved_path가 NULL이 아니면, resolved_path를 return,
resolved_path가 NULL이면, malloc(3)으로 할당하여 real path를 저장한 후에 return
```

○ make와 Makefile

- make : 프로젝트 관리 유틸리티
  - ✓ 파일에 대한 반복 명령어를 자동화하고 수정된 소스 파일만 체크하여 재컴파일 후 종속된 부분만 재링크함
  - ✓ Makefile(규칙을 기술한 파일)에 기술된 대로 컴파일 명령 또는 쉘 명령을 순차적으로 실행함
- Makefile의 구성
  - ✓ Macro(매크로) : 자주 사용되는 문자열 또는 변수 정의 (컴파일러, 링크 옵션, 플래그 등)
  - ✓ Target(타겟) : 생성할 파일
  - ✓ Dependency(종속 항목) : 타겟을 만들기 위해 필요한 파일의 목록
  - ✓ Command(명령) : 타겟을 만들기 위해 필요한 명령(shell)

```
Macro

Target : Dependency1 Dependency2 ...
<-Tab->Command 1
<-Tab->Command 2
<-Tab->...
```

- Makefile의 동작 순서
  - ✓ make 사용 시 타겟을 지정하지 않으면 제일 처음의 타겟을 수행
  - ✓ 타겟과 종속 항목들은 관습적으로 파일명을 명시
  - ✓ 명령 항목들이 충족되었을 때 타겟을 생성하기 위해 명령 (command 라인의 맨 위부터 순차적으로 수행)
  - ✓ 종속 항목의 마지막 수정 시간(st_mtime)을 비교 후 수행
- Makefile 작성 시 참고사항
  - ✓ 명령의 시작은 반드시 Tab으로 시작해야함
  - ✓ 한 줄 주석은 #, 여러 줄 주석은 ₩
  - ✓ 자동 매크로 : 현재 타겟의 이름이나 종속 파일을 표현하는 매크로

매크로	설명
$?	타겟보다 최근에 변경된 종속 항목 리스트 (확장자 규칙에서 사용 불가능)
$^	현재 타겟의 종속 항목 (확장자 규칙에서 사용 불가능)
$<	타겟보다 최근에 변경된 종속 항목 리스트 (확장자 규칙에서만 사용 가능)
$*	타겟보다 최근에 변경된 종속 항목 리스트 (확장자 규칙에서만 사용 가능)
$@	현재 타겟의 이름

- Makefile 작성 예시

(예 1). Makefile

```
test : test.o add.o sub.o
    gcc test.o add.o sub.o -o test

test.o: test.c
    gcc -c test.c

add.o: add.c
    gcc -c add.c

sub.o: sub.c
    gcc -c sub.c

clean :
    rm test.o
    rm add.o
    rm sub.o
    rm test
```

(예 2). 매크로를 사용한 경우

```
OBJECTS = test.o add.o sub.o
TARGET = test
CC = gcc

$(TARGET) : $(OBJECTS)
    $(CC) -o $(TARGET) $(OBJECTS)

test.o: test.c
    $(CC) -c test.c

add.o: add.c
    $(CC) -c add.c

sub.o: sub.c
    $(CC) -c sub.c
```

(예 3). 자동 매크로를 사용한 경우

```
OBJECTS = test.o add.o sub.o
TARGET = test
CC = gcc

$(TARGET) : $(OBJECTS)
    $(CC) -o $@ $^

test.o: test.c
    $(CC) -c $^

add.o: add.c
    $(CC) -c $^

sub.o: sub.c
    $(CC) -c $^
```

- (예 4). Makefile 수행 예시

```
oslab@a-VirtualBox:~$ make
gcc -c test.c
gcc -c add.c
gcc -c sub.c
gcc test.o add.o sub.o -o test
oslab@a-VirtualBox:~$
```

```
oslab@a-VirtualBox:~$ make clean
rm test.o
rm add.o
rm sub.o
rm test
oslab@a-VirtualBox:~$
```

○ 보고서 제출 시 유의 사항

- 보고서 제출 마감은 제출일 11:59PM까지 (구글 서버시간)
- 지연 제출 시 감점 : 1일 지연 시 30% 감점, 2일 이후 미제출 처리

- 압축 오류, 파일 누락 관련 감점 syllabus 참고

- 배점(100점 만점. 프로그램 구현 및 실행 여부 배점(아래 1~10번) 80점으로 최종 환산하며, 보고서 15점과 소스코드 주석 5점은 별도, 강의계획서 확인)
- 0번과 7번 제외 모두 필수

0. 라인단위 수준 분석 및 주석 : 라인 단위 주석과 함수들간 call graph 등이 보고서 점수 15점 + 소스코드주석 5점 부여
1. ssu_score : 4점
2. 옵션 1. -n : 7점
3. 옵션 2. -m : 3점
4. 옵션 3. -c : 13점
5. 옵션 4. -p : 13점
6. 옵션 5. -t : 3점
7. 옵션 6. -s : 30점
8. 옵션 7. -e : 3점
9. 옵션 8. -h : 2점
10. makefile 작성(매크로 사용 권장) : 2점