설계과제 1 개요 : Soongsil Score

Linux System Programming, School of CSE, Soongsil University, Spring 2019

○ 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 " 보고서.hwp "(개요, 상세설계, 구현방법, 결과 및 소스코드와 실행결과가 함께 있는 워드(hwp 또는 MS-Word) 파일))와 " 소스코드 "(makefile, obj, *.c, *.h 등 컴파일하고 실행하기 위한 모든 파일)를 제출해야 함
- 모든 설계과제 결과물은 "#P설계과제번호_학번_버전.zip" (예. #P1_20160000_v1.0.zip)형태로 파일 이름을 명명하고, zip프로그램으로 압축하여 제출해야 함.
- 압축파일 내 "보고서" 디렉토리와 "소스코드" 디렉토리 2개 만들어 제출해야 함
- 제출한 압축 파일을 풀었을 때 해당 디렉토리에서 컴파일 및 실행이 되어야 함. 해당 디렉토리에서 컴파일이나 실행되지 않을 경우, 기본과제 및 설계과제 제출 방법을 따르지 않는 경우 감점 20% 외 추가 20% 감점
- 기타 내용은 syllabus 참고

○ 제출 기한
- 4월 1일(월) 오후 11시 59분 59초 (서버 시간이 30분 정도 빠를 수 있기 때문에 1시간 지연 허용)

○ 보고서 양식
- 보고서는 다음과 같은 양식으로 작성

1. 과제 개요 // 위 개요를 더 상세하게 작성
2. 설계 // 함수 기능별 흐름도(순서도) 반드시 포함
3. 구현 // 함수 프로토타입 반드시 포함
4. 테스트 및 결과 // 테스트 프로그램의 실행 결과 캡쳐 및 분석
5. 소스코드 // 주석

○ 설계 및 구현
- ssu_score <STD_DIR> <ANS_DIR> [OPTION]
  1) <STD_DIR> : 채점 대상이 될 학생이 제출한 답안 디렉토리로 학생들 학번 서브디렉토리를 포함 (최대 100개)
  2) <ANS_DIR> : 채점 기준이 될 정답이 있는 디렉토리로 하나의 문제당 하나의 서브디렉토리를 포함하고 각 문제 서브디렉토리에는 빈칸 채우기 문제의 경우 정답 리스트(다수의 답이 있을 경우 콜론으로 구분), 프로그램 문제의 경우(정답 소스 프로그램(문제번호.c), 정답 실행프로그램(문제번호.ext), 정답 실행프로그래밍 실행결과(문제번호.stdout) 파일 포함

- 출제 시험 문제 형태, 정답 파일, 학생 답안 파일
  ✓ 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 이며 (예. 5.txt), 프로그램 작성 문제의 답안 파일 형태는 문제번호.c (에. 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 문제번호.txt 와 프로그램문제번호 디렉토리(문제번호.c / 문제번호.exe / 문제번호. stdout 등을 포함하는)로 구성
- ✓ 출제되는 문제 수는 최대 100문제까지 허용

○ 빈칸 채우기 문제 예) 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.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
 1-2.txt

#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 =  1-5.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
```

○ 프로그램 문제 예) 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 출력했다는 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@localhost:~/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$
```

○ 빈칸 문제 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
extern char **environ : extern char *environ[]	extern char **environ
	extern char *environ[]

- 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 정답 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

○ 프로그램 문제 (표준출력) 채점 방법
- 학생이 작성해서 답안으로 제출한 프로그램을 컴파일 후 실행파일은 "./STD/학번/문제번호.stdexe" 이름으로 저장 그리고 정답 프로그램의 실행 파일은 "./ANS/문제번호/문제번호.exe" 이름으로 생성
- 각 문제에 대한 정답 프로그램인 "./ANS/문제번호/문제번호.c"의 실행 파일인 "./ANS/문제번호/문제번호.exe"를 실행시켜 실행결과를 "./ANS/문제번호/문제번호.stdout"으로 저장
- 학생이 작성해서 답안으로 제출한 프로그램의 실행파일인 "./STD/학번/문제번호.sdtexe"을 자동으로 실행시키고 실행결과를 "./STD/학번/문제번호.stdout" 자동으로 저장
- "./ANS/문제번호/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

○ 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

○ 옵션 [OPTION]

- 옵션 없이 실행하는 경우

실행 예. 옵션 없는 경우

```
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 : 0.5
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..
```

- -e 옵션 : -e [DIRNAME] : DIRNAME/학번/문제번호_error.txt에 에러 메시지가 출력

실행 예. -e 옵션

```
oslab@minipc:~/ssu_score$ ./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..
oslab@minipc:~/ssu_score$ ls error/20190005/
11_error.txt  13_error.txt  14_error.txt
oslab@minipc:~/ssu_score$ 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
```

- -p 옵션 : 채점을 진행하면서 각 학생의 점수 출력 및 전체 평균 출력

실행 예. -p 옵션

```
oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET -p
grading student's test papers..
20190001 is finished.. score : 46.8
20190002 is finished.. score : 52.8
20190003 is finished.. score : 52.4
20190004 is finished.. score : 46.8
20190005 is finished.. score : 39.3
20190006 is finished.. score : 52.8
20190007 is finished.. score : 52.8
Total average : 49.1
```

- -t [QNAMES] 옵션 : QNAME을 문제 번호로 하는 문제는 컴파일 시 -lpthrad 옵션 추가

실행 예. -t 옵션 적용 안 한 경우

```
oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET -p
grading student's test papers..
20190001 is finished.. score : 70.5
20190002 is finished.. score : 73.5
20190003 is finished.. score : 72.0
20190004 is finished.. score : 70.5
20190005 is finished.. score : 67.5
20190006 is finished.. score : 73.5
20190007 is finished.. score : 73.5
Total average : 71.6
```

실행 예. -t 옵션 적용 한 경우

```
oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET -p -t 12 15
grading student's test papers.
20190001 is finished.. score : 78.5
20190002 is finished.. score : 81.5
20190003 is finished.. score : 80.0
20190004 is finished.. score : 70.5
20190005 is finished.. score : 67.5
20190006 is finished.. score : 81.5
20190007 is finished.. score : 73.5
Total average : 76.1
```

- -h 옵션 : 사용법 출력

실행 예. -h 옵션

```
oslab@minipc:~/ssu_score$ ./ssu_score -h
Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]
Option :
 -e <DIRNAME>       print error on 'DIRNAME/ID/qname_error.txt' file
 -t <QNAMES>        compile QNAME.C with -lpthread option
 -h                 print usage
 -p                 print student's score and total average
 -c <IDS>           print ID's score
```

- -c [STUDENTIDS] 옵션 : 채점결과 파일이 있는 경우 해당 학생들의 점수 출력 => <STUDENTDIR> <TRUESETDIR>가 없어도 사용 가능

실행 예. -c 옵션

```
oslab@minipc:~/ssu_score$ ./ssu_score -c 20190000 20190001 20190002
20190001's score : 46.8
20190002's score : 52.8
```

○ 기타

- 모든 옵션은 동시에 쓸 수 있음. -h 옵션도 동시에 쓸 수 있으나 사용법을 출력 후 ssu_score가 종료
- 가변인자를 받는 옵션의 경우, 최대로 받을 수 있는 가변인자의 개수는 5개로 제한. 그 이상의 가변인자를 받으면 실행과 같은 메시지를 출력하고 수행에는 반영하지 않도록 구현

실행 예. 여러 옵션을 동시에 사용할 경우

```
oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET -c 20190001 20190002 -t 12 15 -e error -p
grading student's test papers..
20190001 is finished.. score : 73.5
20190002 is finished.. score : 76.5
20190003 is finished.. score : 72.0
20190004 is finished.. score : 70.5
20190005 is finished.. score : 67.5
20190006 is finished.. score : 76.5
20190007 is finished.. score : 76.5
Total average : 73.3
20190001's score : 73.5
20190002's score : 76.5
oslab@minipc:~/ssu_score$ 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
```

실행 예. 가변인자 최대 개수를 초과한 경우의 출력

```
oslab@minipc:~/ssu_score$ ./ssu_score STUDENT TRUESET -c 20190001 20190002 20190003 20190004 20190005 20190006 -t 11 12 13 14 15 16
Maximum Number of Argument Exceeded.  :: 20190006
Maximum Number of Argument Exceeded.  :: 16
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..
20190001's score : 73.5
20190002's score : 76.5
20190003's score : 75.0
20190004's score : 70.5
20190005's score : 67.5
```

- 존재하지 않는 디렉토리 혹은 파일을 대상으로 할 경우 에러 처리
- 실행결과 및 에러 메시지를 파일에 출력 시 system()함수 사용 금지
- 점수 출력은 소수점 둘째자리까지 출력
- popen(), fork(), exec() 계열 함수 사용 금지
- 게시판을 통해 2018년 리시프 문제, 정답 그리고 학생답 샘플 제공

<참고> 과제 구현에 필요한 함수(필수 사용 아님)

1. difftime() : 두 시간의 차이를 초 단위로 계산할 때 사용하는 라이브러리 함수

```
#include <time.h>
double difftime(time_t time1, time_t time0);
                                        리턴값: 두 시간 값의 차를 계산한 실수 값
```

difftime_example.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
void main(int argc, char *argv[])
{
        time_t first, second;
        first = time(NULL);
        getc(stdin);
        second = time(NULL);
        printf("wait time is %f seconds\n", difftime(second, first));
}
```

실행결과

```
oslab@localhost:~$ ./difftime_example
a↲ (입력 후 엔터)
wait time is 3.000000 seconds
```

2. gettimeofday() : p.519
3. getopt() : 파라미터를 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
```

4. system() : p.270

```
#include <stdlib.h>
int system(const char* string);
                                                    리턴값: 성공시 0이 아닌 값, 실패시 0
```

○ 보고서 제출 시 유의 사항
- 보고서 제출 마감은 제출일 자정까지 (1시간 지연 허용)
- 지연 제출 시 감점 : 1일 지연 시 마다 30% 감점, 3일 지연 후부터는 미제출 처리
- 압축 오류, 파일 누락 관련 감점 syllabus 참고

○ 구현 점수
가. 빈칸 채우기 문제 자동 채점 기능 40
나. 프로그램 작성문제 자동 채점 기능 40
다. 프로그램 채점 시 5초 이상 수행되는 문제 예외처리 기능 8
라. -e 옵션 3
마. -p 옵션 3
바. -t 옵션 2
사. -h 옵션 2
아. -c 옵션 2
자. gettimeofday()를 사용하여 프로그램의 수행 시간 측정 - 모든 과제 동일 (미구현시 -5)
차. 금지한 함수 사용 시 한 개 함수 사용 당 -30

○ **필수 구현 사항 : 가, 나, 라, 마, 바, 자**