

1.

소속 : 소프트웨어학부

학번 : 20210292

이름 : 김희영

개발환경 : 맥 -> 리눅스 터미널

2. 수정 및 작성한 소스코드에 대한 설명

(1) 20210292.c 파일

```
#include <assignment2.h> -> #include "20210292.h"
```

(2) 20210292.h 파일

```
void change_rodata(fd, sh) {  
    char* buff = read_section(fd, sh); // .rodata 섹션을 읽어온 후 버퍼에 반환한다.  
    char* start = buff; // 포인터 형식으로 시작 위치를 가리킨다.  
    char* end = buff + sh.sh_size; // 끝 위치는 시작위치부터 버퍼 사이즈만큼이기 때문에  
    위와 같이 작성했다.  
    while(start < end) {  
        if (!strcmp(start, "software", 8)) { // start 문자열중에서 "software"랑 일치  
            하는 부분을 찾는다.  
                strncpy(start, "hackers!", 8); // software를 hackers!로 변경한다.  
                start = start + 8; // hackers! 가 8바이트이니 8만큼 위치를 이동한다.  
            }  
        else  
            start++; // 일치하는 부분이 없으면 1만큼 이동한다.  
    }  
    assert(lseek(fd, (off_t)sh.sh_offset, SEEK_SET) == (off_t)sh.sh_offset); // .rodata  
    offset으로 이동한다.  
    assert(write(fd, buff, sh.sh_size) == sh.sh_size); // 변경될 걸 파일에 작성한다.  
}
```

3. first, you need to implement a function to check whether the file is really an ELF file or not.

readelf.h 파일 안에 작성되어있는 is_elf 함수를 통해 It's not ELF file 이 출력되지 않았기 때문에 elf 파일임을 알 수 있다.

4. 과제 수행 중 발생한 문제점과 해결 방법

(1) Makefile 작성 .h 파일

```
CC=gcc
CFLAGS=-I.

DEPS = 20210292.h

OBJ = 20210292.o

%.o: %.c $(DEPS)
    $(CC) -c -o $@ $< $(CFLAGS)

20210292: $(OBJ)
    $(CC) -o $@ $^ $(CFLAGS)

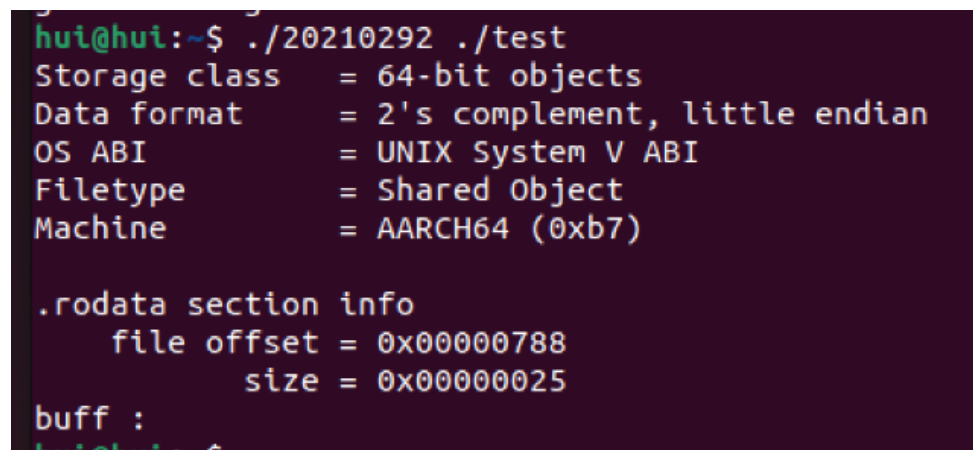
.PHONY: clean
clean:
    rm -f $(OBJ) 20210292
```

(저번에 작성하였던 Makefile)

이번 과제에서는 헤더 파일에 직접적으로 코드를 작성하는 방식이기 때문에 저번에 작성하였던 Makefile을 사용하면 헤더파일이 컴파일이 되지 않은 상태로 .c파일과 합쳐진다.

-> all 대상을 TARGET으로 하여 의존성을 갖도록 하였다. TARGET 대상은 헤더 파일을 의존하도록 Makefile을 작성한다. 따라서 Makefile이 헤더 파일에 변경이 있을 경우 인식하여 실행 파일을 빌드할 수 있게 도와준다.

(2) .rodata 출력



```
hui@hui:~$ ./20210292 ./test
Storage class      = 64-bit objects
Data format        = 2's complement, little endian
OS ABI             = UNIX System V ABI
Filetype           = Shared Object
Machine            = AARCH64 (0xb7)

.rodata section info
  file offset = 0x00000788
  size = 0x00000025
buff :
```

printf("buff : %s \n", buff);를 통해 데이터를 출력하려고 하였지만 아무것도 출력되지 않았다.

-> 데이터 형식이 이진수이기 때문에 %s를 통해 출력이 불가능하다.

(3) 위치 이동

```

hui@hui:~$ ./20210292 ./test
Storage class      = 64-bit objects
Data format        = 2's complement, little endian
OS ABI             = UNIX System V ABI
Filetype           = Shared Object
Machine            = AARCH64 (0xb7)

.rodata section info
  file offset = 0x00000788
  size = 0x00000037

```

.rodata 데이터를 변경하였지만 함수가 종료되지 않고 계속 로딩하는 현상이 있었다.

-> 위치를 변경하지 않아 생긴 문제였다. start를 통해 .rodata를 바꾸는 코드를 작성하였는데 위치를 이동하여 데이터를 읽었어야 하는데 위치를 이동하는 코드를 작성하지 않아 생긴 문제였다. software문자열이 있을 경우는 8바이트를 이동하고 아닐 경우는 1바이트씩 위치를 변경하는 코드를 작성하였다.

(4) 파일 변경

```

hui@hui:~$ make 20210292
gcc -Wall -g -o 20210292 20210292.c
hui@hui:~$ ./20210292 ./test
Storage class      = 64-bit objects
Data format        = 2's complement, little endian
OS ABI             = UNIX System V ABI
Filetype           = Shared Object
Machine            = AARCH64 (0xb7)

.rodata section info
  file offset = 0x00000788
  size = 0x00000037
hui@hui:~$ ./test
Hello software the school of software software

```

(모든 "software"문자열이 바뀌는 걸 확인하기 위해 test.c 파일을 위와 같이 작성하였습니다.)

변경된 .rodata 섹션의 내용이 실제 파일에 반영되지 않았다.

-> 실제 변경된 .rodata섹션의 내용을 파일에 쓰는 코드가 없어서 발생한 문제였다. .rodata 섹션 내용 상에서만 수정된 것이기 때문에 파일에 쓰는 코드를 작성해야 했다. read_section함수와 마찬가지로 lseek함수와 함께 파일에서 읽는 것이 아니라 써야 하기 때문에 write함수를 사용하여 코드를 작성하였다.

(5) sizeof(buff)

```

hui@hui:~$ make 20210292
gcc -Wall -g -o 20210292 20210292.c
hui@hui:~$ ./test
Hello software the school of software software
hui@hui:~$ ./20210292 ./test
Storage class      = 64-bit objects
Data format        = 2's complement, little endian
OS ABI             = UNIX System V ABI
Filetype           = Shared Object
Machine            = AARCH64 (0xb7)

.rodata section info
  file offset = 0x00000788
    size = 0x00000037
hui@hui:~$ ./test
Hello software the school of software software

```

char* end = buff + sizeof(buff)를 사용하였는데 내용이 바뀌지 않는 문제가 발생하였다.
-> buff는 포인터변수이기 때문에 실제 할당된 버퍼의 크기가 아니기 때문에 발생한 문제이다.
char* end = buff + sh.sh_size; 로 변경하였다.

5. 고민했던 부분들

(1) .rodata 읽는 법

.rodata 섹션을 찾아서 읽어온 후 데이터를 변경해야 하는 것이기 때문에 먼저 .rodata 섹션 위치를 찾으려고 하였다. 섹션 헤더 테이블에 ELF파일의 모든 섹션 정보를 저장하고 있기 때문에 readelf -S 20210292 명령어를 통해 섹션 헤더 정보를 출력하여 .rodata 섹션의 정보를 확인하였다.

[12]	.plt	PROGBITS	0000000000000890	00000890
	0000000000000110	0000000000000000	AX 0 0	16
[13]	.text	PROGBITS	00000000000009c0	000009c0
	0000000000000ba0	0000000000000000	AX 0 0	64
[14]	.fini	PROGBITS	0000000000001560	00001560
	0000000000000014	0000000000000000	AX 0 0	4
[15]	.rodata	PROGBITS	0000000000001578	00001578
	000000000000055e	0000000000000000	A 0 0	8
[16]	.eh_frame_hdr	PROGBITS	0000000000001ad8	00001ad8
	000000000000006c	0000000000000000	A 0 0	4
[17]	.eh_frame	PROGBITS	0000000000001b48	00001b48

(readelf -S 20210292 명령어 출력결과)

위 정보를 통해 offset을 기준으로 데이터를 읽어오면 되겠다는 점을 알았다.

print_section_headers() 함수에서 sh_table[i]와 파일 디스크립터 fd를 파라미터로 사용하여 change_rodata 함수를 실행해야 한다.

먼저 .rodata 섹션으로 이동하여 읽어와야 한다. readelf.h 파일 안에 있는 read_section 함수는 lseek 함수와 read 함수를 통해 특정 섹션을 읽어오는 함수이다. lseek 함수는 seek pointer를 사용

하여 해당 위치로 이동하는 함수로 read나 write를 이용하여 해당 데이터를 읽거나 쓸 때 많이 사용된다. lseek함수를 통해 파일 디스크립터 fd를 .rodata 섹션 offset으로 이동한다. 즉, 해당 섹션의 시작 위치로 이동하게 된다. read 함수를 통해 파일로부터 섹션의 크기만큼 데이터를 읽어오는 함수이다. 읽어온 데이터를 동적으로 할당된 buff에 저장한 후 반환한다.

따라서 read_section 함수를 사용하여 .rodata 섹션을 읽어와야 한다. sh_table[i]는 섹션 헤더의 정보를 담고 있는 테이블에서 .rodata 섹션의 정보를 가지고 있는 i를 찾았기 때문에 그 sh_table[i]를 이용하여 read_section함수를 사용한다.

char* buff = read_section(fd, sh); 와 같이 작성하였다.

(2) .rodata 변경하는 법 (software -> hackers!)

시작과 끝 위치를 지정하여서 버퍼를 확인하면서 데이터를 변경하면 된다.

print_section_headers함수에서 한 것처럼 strcmp함수를 사용하여 데이터를 확인하였다.

strcmp함수는 문자열이 완전히 일치해야 true를 반환한다는 의미를 가진다. 문자열 중에서 지정된 문자열 수 만큼만 일치하면 되기 때문에 strncmp함수를 사용한다.

6. 실행 결과

```
hui@hui:~$ make 20210292
gcc -Wall -g -o 20210292 20210292.c
hui@hui:~$ ./test
Hello the school of software
hui@hui:~$ ./20210292 ./test
Storage class      = 64-bit objects
Data format        = 2's complement, little endian
OS ABI             = UNIX System V ABI
Filetype           = Shared Object
Machine            = AARCH64 (0xb7)

.rodata section info
  file offset = 0x00000788
  size = 0x00000025
hui@hui:~$ ./test
Hello the school of hackers!
hui@hui:~$
```

7. 참고자료

(1) ELF 구조

<https://sonseungha.tistory.com/460>

(2) section header

<https://docs.oracle.com/cd/E19455-01/806-3773/elf-2/index.html>