동국대학교
dongguk university

제출일	2023.04.11	학과	컴퓨터공학전공
과목	컴퓨터보안	학번	2018112007
담당교수	김영부 교수님	이름	이승현

1) 실습 환경

(1)
운영 체제: Microsoft Windows 11 Home 64bit
프로세서 : Intel(R) Core(TM) i7-10510U @ 1.80GHz (8 CPUs), ~ 2.3GHz
메모리 : DDR4 16GB 2,667MHz
그래픽 카드 : Intel UHD Graphics

(2)
운영 체제: Microsoft Windows 10 Home 64bit
프로세서 : Intel(R) Core(TM) i7-7700HQ @ 2.80GHz (8 CPUs), ~ 2.8GHz
메모리 : DDR4 8GB 2,133MHz
그래픽 카드 : Intel HD Graphics 630, NVIDIA GeForce GTX 1050

2) 실습 진행

1. 문제 분석

1) 프로그램 메모리 세그먼트

- 프로그램을 실행하게 되면, CPU 프로세서는 보조기억장치(HDD, SDD)에 있는 프로그램 정보를 읽어와 RAM 메모리(캐시, 주기억장치)에 로드(Load)한다. 메모리 공간은 프로세스에 할당되어 CPU에 의해 수행된다. 이때, 프로그램이 저장되는 메모리 공간은 일반적으로 code, data, bss, stack, heap과 같이 5가지의 세그먼트로 분류된다.
- 세그먼트 방식은 가상주소(Virtual Address)인 논리적 주소(Logical Address)를 사용하여 프로그램에 따른 상대적인 위치를 지정한 후, 시작 위치(Offset)를 더하는 방식으로 메모리의 물리적인 주소(Physical Address)에 접근하는 방식이다.
- 이번 실습에는 전역 변수와 정적 변수의 선언 여부와 값의 초기화 여부에 따른 메모리 세그먼트의 변화를 살펴보고, 변화의 이유에 대해 생각해본다.

2) SetUID Program

- SetUID는 유닉스 운영 체제에서 중요한 보호 메커니즘이다. 보통 소유자(owner) 권한에서 Set-UID 프로그램이 실행되며, 만약 프로그램의 소유자가 루트라면, 어떤 사용자이든 프로그램을 실행시키면 프로그램이 실행되는 동안 그 프로그램은 루트의 권한을 얻게 된다. Set-UID를 통해서 수많은 흥미로운 것들을 할 수 있지만, SetUID는 많은 문제의 주범이기도 하다.
- SetUID Program Vulnerability Lab의 문제를 풀어보고, 실습을 통해서 Set-UID가 왜 필요하고 어떻게 구현되고 있는지를 이해하고 장점을 확인하는 것이다. 또한 SetUID의 보안 문제를 확인하고 신중하게 사용할 필요성을 깨닫게 하는 것이다.

2. 실습

1) 프로그램 메모리 세그먼트

1. 'Practice1.c' 파일을 생성한다. 이를 result1의 이름으로 컴파일하여 실행 파일을 생성하고 이후 size 명령을 통해 생성된 실행 파일을 대상으로 세그먼트 크기 값을 확인하고 결과에 대해 분석하시오

```c
#include <stdio.h>

int main()
{
    return 0;
}
```

- 아무런 전역 변수의 선언 없이 메인 함수만 존재하며, 메인 함수 안에서도 변수가 없이, 0을 반환한다.

```
kocan@DESKTOP-6UMUOFM:~$ size result1
   text    data     bss     dec     hex filename
   1228     544       8    1780     6f4 result1
```

- 위의 코드를 컴파일한 파일의 세그먼트 크기 값을 출력한 모습이다. text 세그먼트의 크기는 1,228바이트이며, data 세그먼트는 544바이트, bss 세그먼트는 8바이트, 총합은 1,780바이트이다. text 세그먼트는 프로그램의 코드 부분을 저장하는 저장 공간이며, 프로그램에서 사용하는 함수의 개수와 코드 크기, 각종 상수 등의 크기에 따라 결정된다. 이 프로그램에서는 함수가 메인 함수 하나이고, 코드 크기가 크지 않음에도 1,228바이트를 차지하는 모습을 볼 수 있다. 또한 아무런 변수가 선언되어 있지 않음에도 불구하고 data와 bss의 크기 값이 0이 아닌데, data 세그먼트의 경우에는 컴파일러나 링커의 최적화 과정에서 불필요한 데이터나 여분의 공간을 삭제하지 못할 가능성이 있다. bss 세그먼트의 경우에는 64bit 아키텍처에서의 포인터 크기가 8바이트인데, 메인 함수의 반환 값인 0이 포인터 크기와 같은 8바이트의 메모리 공간에 저장되기 때문이다. 이때, 메모리 공간은 bss 세그먼트에 할당된다.

2. 기존 Practice1.c 코드에 정수형 global 변수를 전역으로 선언하고 result2의 이름으로 컴파일하여 기존 세그먼트 크기와 어떤 차이점이 존재하는지 분석하시오.

```c
#include <stdio.h>

int glabal;

int main()
{
    return 0;
}
```

- 이전의 코드에서 전역 변수를 추가한 모습이다. 초기화되지 않은 전역 변수가 선언되었으니 bss가 증가할 것이다.

```
kocan@DESKTOP-6UMUOFM:~$ size result2
   text    data     bss     dec     hex filename
   1228     544       8    1780     6f4 result2
```

- 그러나 size로 출력한 세그먼트의 크기를 살펴보면 bss가 늘어나지 않은 모습을 볼 수 있다. 왜 bss가 안 늘어났냐면 컴파일러나 링커의 최적화 과정에서 만약 전역 변수가 사용되지 않으면 컴파일러에서 전역 변수를 제거할 수 있고, 전역 변수가 0으로 초기화될 필요가 없는 경우에도 bss를 사용하지 않기 때문이다.

3. 메인 함수 내에 정수형 i 변수를 정적으로 선언하고 result3의 이름으로 컴파일하여 기존 세그먼트 크기와 어떤 차이점이 존재하는지 분석하시오.

```c
#include <stdio.h>

int glabal;

int main()
{
    static int i;
    return 0;
}
```

- 전역 변수와 정적 변수가 선언된 모습이다. 전역 변수와 정적 변수가 추가되었기 때문에 bss가 8바이트 늘어나 있어야 할 것이다.

```
kocan@DESKTOP-6UMUOFM:~$ size result3
   text    data     bss     dec     hex filename
   1228     544      16    1788     6fc result3
```

- 이번에는 이전 단계와 달리 bss가 정상적으로 늘어난 모습을 볼 수 있다. 정수형 전역 변수와 정수형 정적 변

수가 선언되었기 때문에, 변수 하나당 4바이트가 bss에 할당되어 8바이트가 늘어난 모습을 볼 수 있다.

4. 전역 변수 global을 10으로 초기화하여 result4의 이름으로 컴파일하고 기존 세그먼트 크기와 어떤 차이점이 존재하는지 분석하시오.

```c
#include <stdio.h>

int glabal = 10;

int main()
{
    static int i;
    return 0;
}
```

- 앞서 선언된 전역 변수가 10으로 초기화된 모습을 볼 수 있다. 전역 변수가 초기화가 되었으니 data의 크기가 늘어날 것이고, bss의 크기가 줄어들 것이다.

```
kocan@DESKTOP-6UMUOFM:~$ size result4
   text    data     bss     dec     hex filename
   1228     548      12    1788     6fc result4
```

- 예상대로 bss가 4바이트 줄고, data가 4바이트 증가한 모습을 볼 수 있다. 이는 초기화되지 않은 전역 변수나 정적 변수가 bss에 할당되는데, 전역 변수 global이 초기화되었으니 bss에 할당된 메모리는 반환하고, data로 할당된 것이다.

5. 정적 변수 i를 100으로 초기화하여 result5의 이름으로 컴파일하고 기존 세그먼트 크기와 어떤 차이점이 존재하는지 분석하시오.

```c
#include <stdio.h>

int glabal = 10;

int main()
{
    static int i = 100;
    return 0;
}
```

- 전역 변수에 이어서 정적 변수까지 초기화된 모습을 볼 수 있다. bss가 또 줄고, data는 또 늘어날 것이다.

```
kocan@DESKTOP-6UMUOFM:~$ size result5
   text    data     bss     dec     hex filename
   1228     552       8    1788     6fc result5
```

- 예상대로 bss가 4바이트 줄어 처음과 같은 크기가 되었고, data는 4바이트 늘어 552바이트가 되었다. 전역 변수나 정적 변수가 초기화되지 않았을 때는 bss에 할당하지만, 초기화되었다면 data에 할당되기 때문에 초기화할 때마다 bss는 줄고, data가 증가하는 것이다.

6.1), 2), 3), 4), 5)를 통해 확인한 세그먼트 크기들이 각각 다르다면 그 원인에 관해 분석하고 설명하시오.

- 변수의 형태별로 메모리 세그먼트에 할당되는 영역이 다르다. 초기화되지 않은 정적 변수와 정적 변수는 bss에 할당되었다가, 중간에 초기화되었다면 bss에 할당된 메모리를 반환하고 data에 할당하게 된다. 지역 변수는 stack에 할당되고, 동적 할당된 변수인 경우는 heap, 프로그램의 코드는 text에 할당된다. 앞서 실습했던 코드는 지역 변수가 선언되지 않았고, 프로그램에 사용된 함수에는 변화가 없었으며, 동적 할당된 변수는 존재하지 않았다. 오로지 전역 변수와 정적 변수가 선언되어 초기화의 유무에 따라 bss와 data가 변화하는 것이었다. 처음에는 전역 변수와 정적 변수가 초기화되지 않았기에 bss에 할당되었지만, 전역 변수가 초기화되었을 때는 bss에 할당된 메모리가 반환되고, data에 할당되었고, 정적 변수 또한 초기화되었을 때 전역 변수와 같

은 작용이 일어났다. 따라서 실습을 진행할 때마다 bss가 줄고 data가 늘어나는 모습이 나타난 것이다.

2) 오브젝트 파일 수정 및 링킹(Linking) 실습 1

1. 첫 번째 실습에서 생성했던 result1 실행 파일을 대상으로 ELF 파일 형식을 확인하시오.

```
kocan@DESKTOP-6UMUOFM:~$ readelf -h result1
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1040
  Start of program headers:          64 (bytes into file)
  Start of section headers:          13928 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```

- result1의 ELF 헤더의 정보를 출력한 모습이다. ELF란 행 파일과 공유 라이브러리 등을 위한 이식 가능한 이진 형식이다. ELF 형식은 크게 ELF 헤더와 세그먼트로 구성되고, ELF 헤더를 출력해보면 ELF 헤더의 크기와 endian 정보, 파일의 타입, 사용하는 아키텍처 정보 등 여러 가지 정보가 포함된 모습을 볼 수 있다. 이 파일의 type이 DYN이며, 특징은 실행 위치에 상관없이 메모리에 로드되어 실행될 수 있도록 만들어지고 위치 독립적인 실행 파일이다. DYN 파일을 사용하면 프로그램이 실행될 때 로드되는 공유 라이브러리의 주소가 고정되지 않아도 되므로, 같은 라이브러리를 여러 프로세스에서 공유하여 메모리 사용을 효율적으로 관리할 수 있다.

2. 'Practice2.c' 파일을 생성한다. 이를 컴파일을 통해 오브젝트 파일을 생성하고 ELF 파일 형식을 확인하여 1)의 결과와 어떤 차이가 존재하는지 분석하시오.

```c
#include <stdio.h>

int main()
{
    printf("hello world!!");
    return 0;
}
```

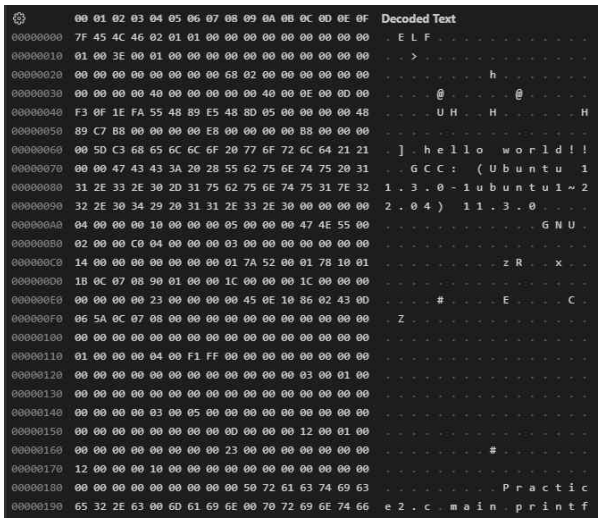
- Practice2.c 파일이다. hello world!!를 출력하는 간단한 파일이다.

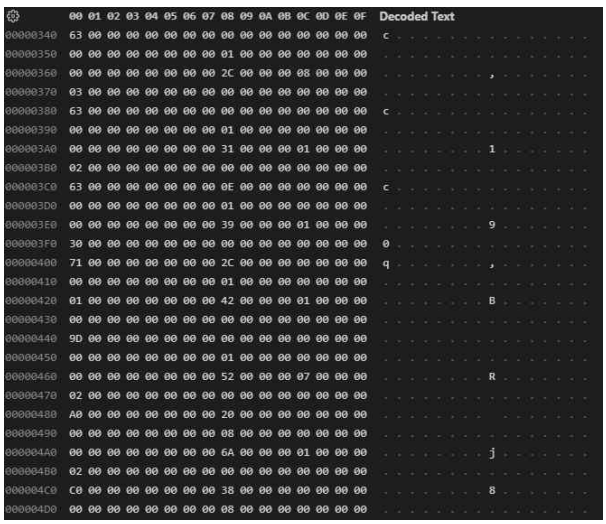
```
kocan@DESKTOP-6UMUOFM:~$ readelf -h Practice2.o
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          616 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           64 (bytes)
  Number of section headers:         14
  Section header string table index: 13
```

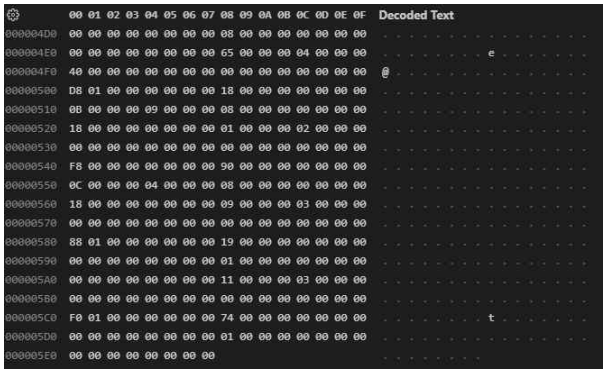
- Practice2의 오브젝트 파일을 생성하여 ELF 파일 형식을 확인한 모습이다. 헤더 크기나 개수를 제외하고 큰 차이점이라면 파일의 타입이다. result1의 파일 타입은 DYN(Position-Independent Executable file)으로 실행 위치에 상관없이 메모리에 로드되어 실행될 수 있도록 만들어지고 위치 독립적으로 실행할 수 있으며, 프로그램이 실행될 때 로드되는 공유 라이브러리의 주소가 고정되지 않아도 되므로, 같은 라이브러리를 여러 프로세스에서 공유하여 메모리 사용을 효율적으로 관리할 수 있다. Practice2는 REL(Relocatable file)로 컴파일

러가 생성하는 오브젝트 파일이나 라이브러리 파일이 이런 타입에 해당한다. REL은 배치 정보와 상대 주소로 참조되는 코드와 데이터를 포함하고 있으며, 상대 주소로 참조되는 코드와 데이터는 실제 메모리 주소로 매핑되어 실행될 때 재배치가 필요하다. 따라서 REL 파일은 로드 주소에 따라 메모리에 위치가 달라질 수 있는 위치 종속적인 코드와 데이터를 포함한다.

3. Hex Editer를 설치하여 Practice2.o 오브젝트 파일을 열어보고 어떤 정보들을 확인할 수 있는지 분석하시오.

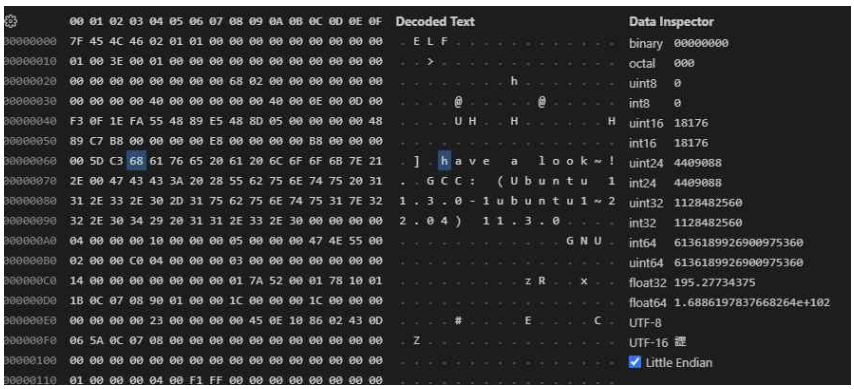






- ELF라는 글자가 보이는 것을 보면 ELF 헤더가 존재하는 것을 볼 수 있고, printf함수로 출력하려는 hello world!! 가 16진수로 표현된 모습을 확인할 수 있다. 바로 뒤에는 gcc와 linux의 계열과 버전을 확인할 수 있고 GNU라는 단어를 볼 수 있다. 밑에는 파일의 이름과 함수명이 기록되어 있고, 더 밑에는 나눠진 각 섹션의 이름을 확인할 수 있다. 그 외에는 Decoded Text로 확인할 수 없지만, 변수와 각 함수가 지정된 코드가 기록되어 있을 것이고, 섹션 정보가 기록되어 있는 부분에서도 이름뿐만 아니라 주소, 크기도 확인할 수 있을 것이다.

4. 기존 문자열을 “have a look~!”으로 변경하여 저장하시오.



- hello world!!를 Decoded Text에서 have a look~!으로 변경한 모습이다. Decoded Text에서 수정해도 hex code에서는 16진수로 값이 변경되는 모습을 확인할 수 있다.

5. 수정된 Practice.o 오브젝트 파일을 gcc을 이용한 링킹으로 modified라는 이름의 실행 파일을 생성하고 실행하여 결과에 대해 분석하시오.

- 원래라면 Practice2를 실행시키면 hello world!!가 출력되어야 하나, Practice2.o 파일을 hex editor로 수정하여 -o 옵션을 통해 오브젝트 파일을 지정하고, 링킹하여 라이브러리와 파일을 하나로 합치기 때문에 수정된 have a look~!이 출력되는 모습을 볼 수 있다.

3) 오브젝트 파일 수정 및 링킹(Linking) 실습 2

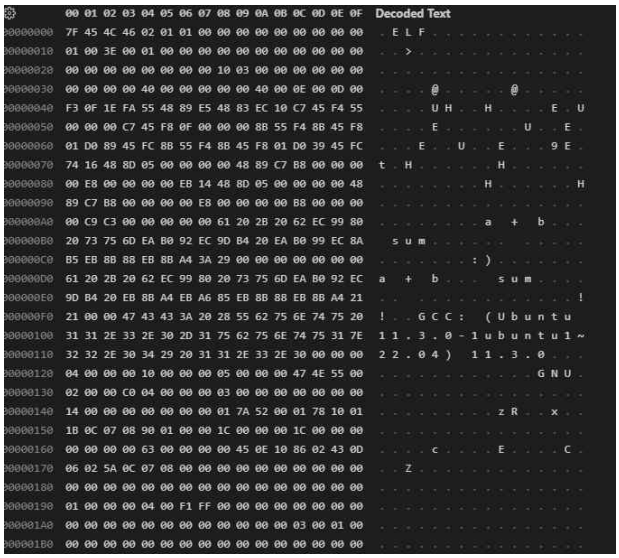
1. “smile.c” 프로그램을 다음과 같이 생성하고 컴파일을 통해 오브젝트 파일을 생성하시오.

```c
#include <stdio.h>

int main()
{
    int a = 85;
    int b = 15;
    int sum = a + b;

    if(sum != (a + b))
    {
        printf("a + b와 sum값이 같습니다:)");
    }
    else
    {
        printf("a + b와 sum값이 다릅니다!!");
    }
    return 0;
}
```

- smile.c 파일의 모습이다. sum과 a+b의 값이 다른 경우 a+b와 sum 값이 같다는 메시지가 출력되고, 같은 경우에 a+b와 sum 값이 다르다는 값이 출력된다.

```
DESKTOP-6UMUOFM%gcc -c smile.c
```

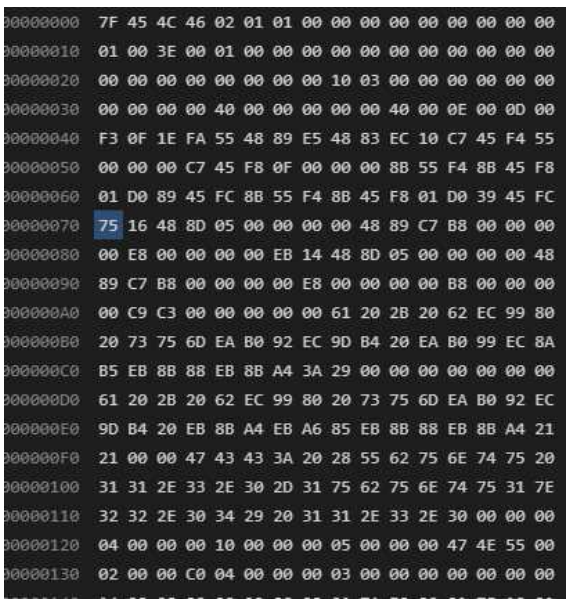
- gcc -c 옵션을 통해 smile.c를 오브젝트 파일로 컴파일한 모습이다.

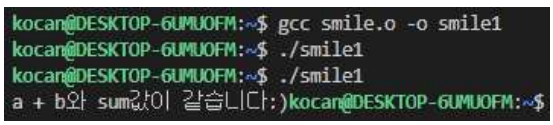


- smile.o를 hex editor로 열어본 모습이다. 중간에 if문에 따른 printf문의 매개변수인 문자열 값이 저장된 모습을 확인할 수 있다.

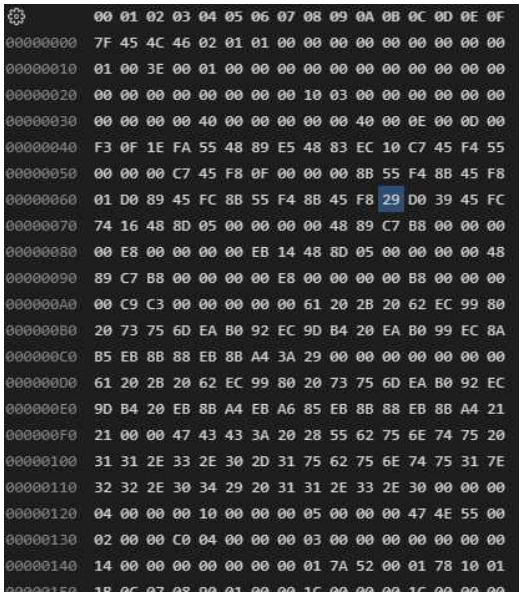
2. 생성된 smile.o 파일을 Hex Editer로 오픈 후 현재 코드에서 “a+b와 sum값이 같습니다:)”의 문자열이 출력될 수 있도록 수정하시오. 또한 이를 가능케 하는 모든 경우의 수를 찾아보시오. (smile.o 파일을 Hex Editer 상에서만 변경해야함)

i. if(sum!=(a+b)) -> if(sum==(a+b)) (JE -> JNE)





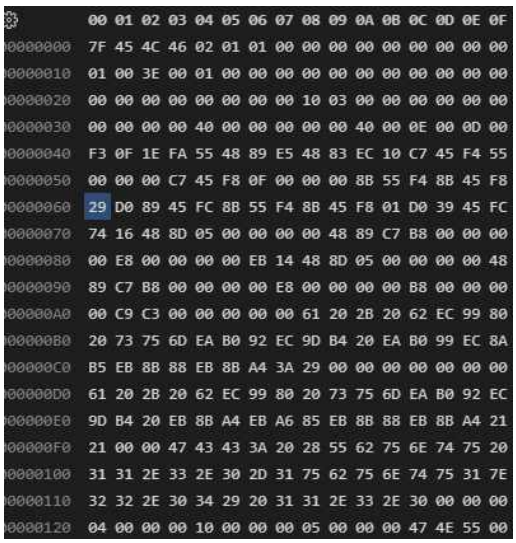
ii. if(sum!=(a+b)) -> if(sum!=(a-b)) (ADD -> SUB)



```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  01 D0 89 45 FC 8B 55 F4 8B 45 F8 29 D0 39 45 FC
00000070  74 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
00000100  31 31 2E 33 2E 30 2D 31 75 62 75 6E 74 75 31 7E
00000110  32 32 2E 30 34 29 20 31 31 2E 33 2E 30 00 00 00
00000120  04 00 00 00 10 00 00 00 05 00 00 00 47 4E 55 00
00000130  02 00 00 C0 04 00 00 00 03 00 00 00 00 00 00 00
00000140  14 00 00 00 00 00 00 00 01 7A 52 00 01 78 10 01
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile2
kocan@DESKTOP-6UMUOFM:~$ ./smile2
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

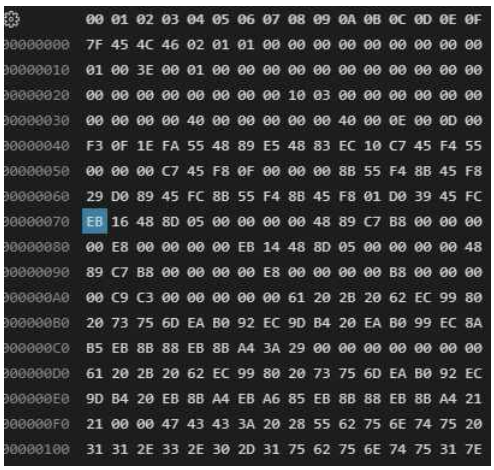
iii. int sum=a+b -> int sum=a-b (ADD -> SUB)



```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  29 D0 89 45 FC 8B 55 F4 8B 45 F8 01 D0 39 45 FC
00000070  74 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
00000100  31 31 2E 33 2E 30 2D 31 75 62 75 6E 74 75 31 7E
00000110  32 32 2E 30 34 29 20 31 31 2E 33 2E 30 00 00 00
00000120  04 00 00 00 10 00 00 00 05 00 00 00 47 4E 55 00
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile3
kocan@DESKTOP-6UMUOFM:~$ ./smile3
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

iv. else 문으로 바로 이동 (JE -> JMP)



```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  29 D0 89 45 FC 8B 55 F4 8B 45 F8 01 D0 39 45 FC
00000070  EB 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
00000100  31 31 2E 33 2E 30 2D 31 75 62 75 6E 74 75 31 7E
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile4
kocan@DESKTOP-6UMUOFM:~$ ./smile4
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

v. if(sum!=(a+b)) -> if(sum>(a+b)) (JE -> JG)

```
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  01 D0 89 45 FC 8B 55 F4 8B 45 F8 01 D0 39 45 FC
00000070  7F 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
00000100  31 31 2E 33 2E 30 2D 31 75 62 75 6E 74 75 31 7E
00000110  32 32 2E 30 34 29 20 31 31 2E 33 2E 30 00 00 00
00000120  04 00 00 00 10 00 00 00 05 00 00 00 47 4E 55 00
00000130  02 00 00 C0 04 00 00 00 03 00 00 00 00 00 00 00
00000140  14 00 00 00 00 00 00 00 01 7A 52 00 01 78 10 01
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile5
kocan@DESKTOP-6UMUOFM:~$ ./smile5
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

vi. if(sum!=(a+b)) -> if(sum<(a+b)) (JE -> JL)

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  01 D0 89 45 FC 8B 55 F4 8B 45 F8 01 D0 39 45 FC
00000070  7C 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile6
kocan@DESKTOP-6UMUOFM:~$ ./smile6
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

vii. int sum=a+b, if(sum!=(a+b)) -> int sum=a-b, if(sum<(a+b)) (JE -> JG)

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  29 D0 89 45 FC 8B 55 F4 8B 45 F8 01 D0 39 45 FC
00000070  7F 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile7
kocan@DESKTOP-6UMUOFM:~$ ./smile7
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

viii. if(sum!=(a+b)) -> if(sum>(a-b)) (JE -> JNGE)

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  01 D0 89 45 FC 8B 55 F4 8B 45 F8 29 D0 39 45 FC
00000070  7C 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
```

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile8
kocan@DESKTOP-6UMUOFM:~$ ./smile8
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

ix. int sum=a+b, if(sum!=(a+b)) -> int sum=a-b, if(sum==(a-b)) (JE -> JNE)

```
        00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  29 D0 89 45 FC 8B 55 F4 8B 45 F8 29 D0 39 45 FC
00000070  75 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
```

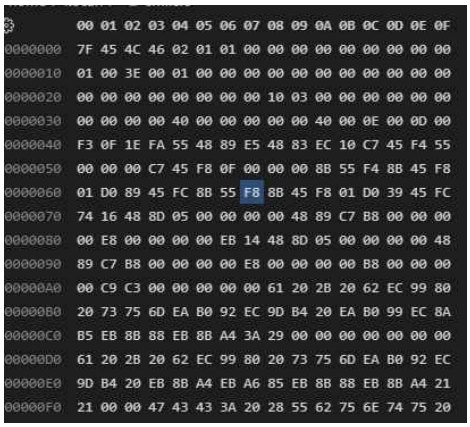
```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile9
kocan@DESKTOP-6UMUOFM:~$ ./smile9
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

x. if(sum!=(a+b)) -> if(sum!=(a+a))

```
00000000  7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00
00000010  01 00 3E 00 01 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 10 03 00 00 00 00 00 00
00000030  00 00 00 00 40 00 00 00 00 00 40 00 0E 00 0D 00
00000040  F3 0F 1E FA 55 48 89 E5 48 83 EC 10 C7 45 F4 55
00000050  00 00 00 C7 45 F8 0F 00 00 00 8B 55 F4 8B 45 F8
00000060  01 D0 89 45 FC 8B 55 F4 8B 45 F4 01 D0 39 45 FC
00000070  74 16 48 8D 05 00 00 00 00 48 89 C7 B8 00 00 00
00000080  00 E8 00 00 00 00 EB 14 48 8D 05 00 00 00 00 48
00000090  89 C7 B8 00 00 00 00 E8 00 00 00 00 B8 00 00 00
000000A0  00 C9 C3 00 00 00 00 00 61 20 2B 20 62 EC 99 80
000000B0  20 73 75 6D EA B0 92 EC 9D B4 20 EA B0 99 EC 8A
000000C0  B5 EB 8B 88 EB 8B A4 3A 29 00 00 00 00 00 00 00
000000D0  61 20 2B 20 62 EC 99 80 20 73 75 6D EA B0 92 EC
000000E0  9D B4 20 EB 8B A4 EB A6 85 EB 8B 88 EB 8B A4 21
000000F0  21 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20
00000100  31 31 2E 33 2E 30 2D 31 75 62 75 6E 74 75 31 7E
```

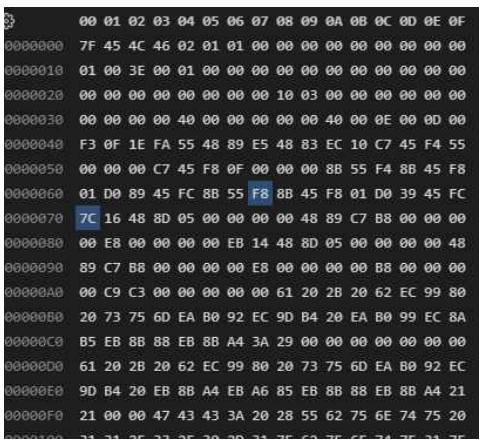
```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile10
kocan@DESKTOP-6UMUOFM:~$ ./smile10
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xi. if(sum!=(a+b)) -> if(sum!=(b+b))



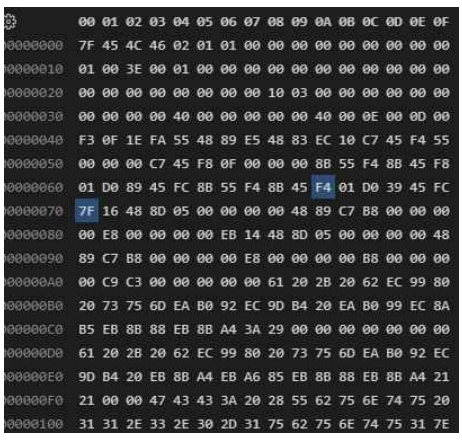

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile11
kocan@DESKTOP-6UMUOFM:~$ ./smile11
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xii. if(sum!=(a+b)) -> if(sum>=(b+b)) (JE -> JNGE)



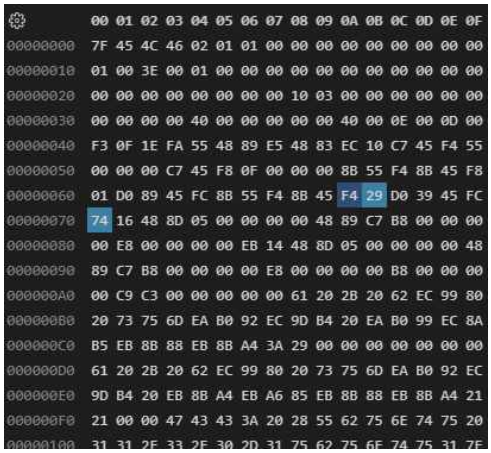

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile12
kocan@DESKTOP-6UMUOFM:~$ ./smile12
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xiii. if(sum!=(a+b)) -> if(sum<(a+a)) (JE -> JNLE)



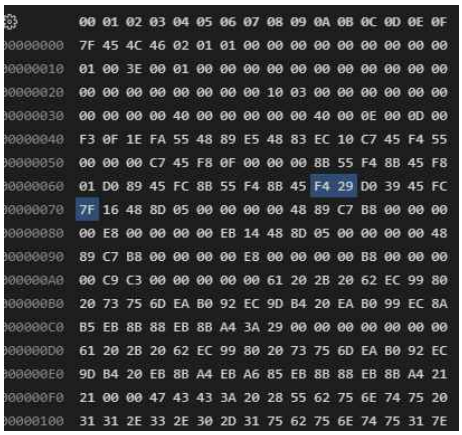

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile13
kocan@DESKTOP-6UMUOFM:~$ ./smile13
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xiv. if(sum!=(a+b)) -> if(sum!=(a-a))



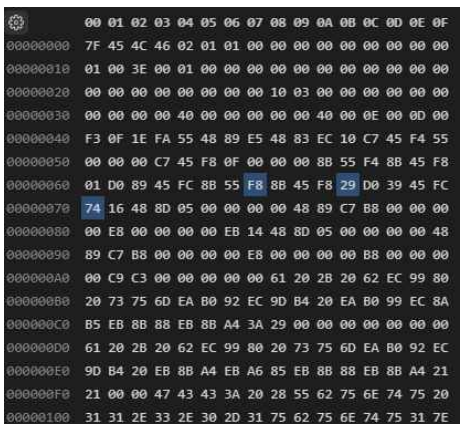

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile14
kocan@DESKTOP-6UMUOFM:~$ ./smile14
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xv. if(sum!=(a+b)) -> if(sum>(a-a)) (JE -> JG)




```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile15
kocan@DESKTOP-6UMUOFM:~$ ./smile15
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xvi. if(sum!=(a+b)) -> if(sum!=(b-b))

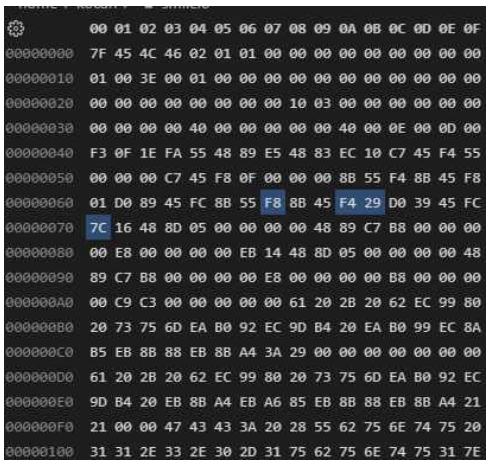



```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile16
kocan@DESKTOP-6UMUOFM:~$ ./smile16
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xvii. if(sum!=(a+b)) -> if(sum>(b-b)) (JE -> JNGE)

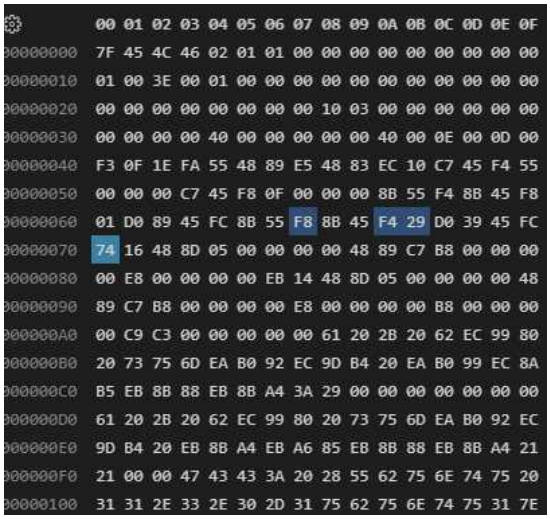

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile17
kocan@DESKTOP-6UMUOFM:~$ ./smile17
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xviii. if(sum!=(a+b)) -> if(sum>(b-a)) (JE -> JNGE)



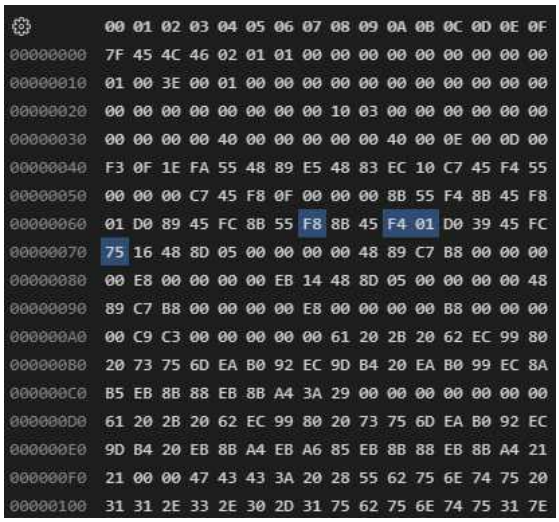

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile18
kocan@DESKTOP-6UMUOFM:~$ ./smile18
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xix. if(sum!=(a+b)) -> if(sum!=(b-a))

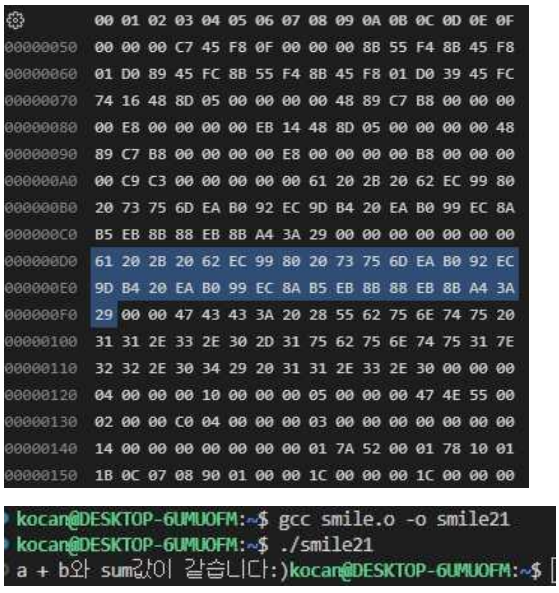



```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile19
kocan@DESKTOP-6UMUOFM:~$ ./smile19
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

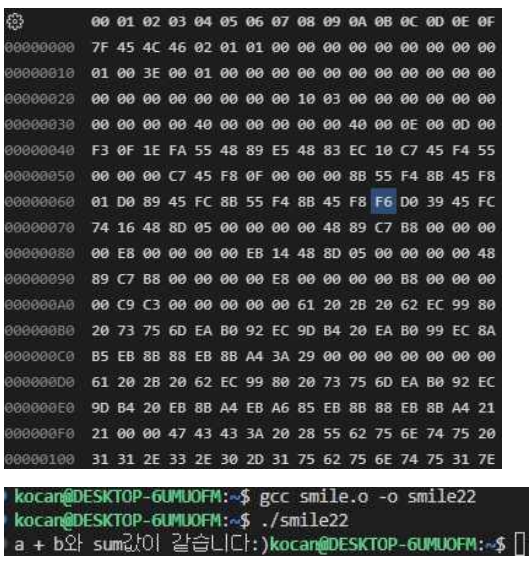
xx. if(sum!=(a+b)) -> if(sum==(b+a)) (JE -> JNE)



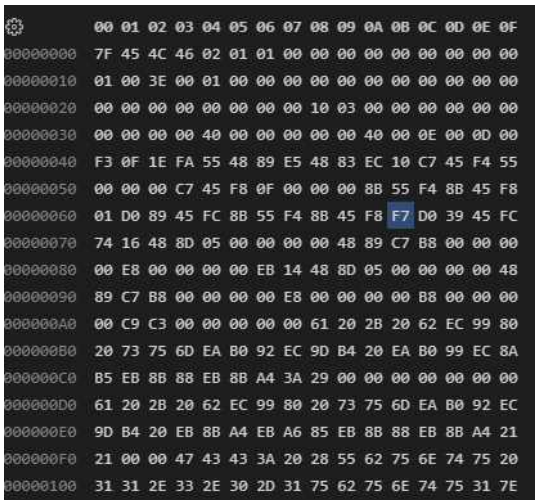
xxi. else 문의 printf문을 true 문 속 printf문의 내용으로 수정

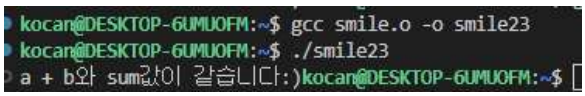


xxii. if(sum!=(a+b)) -> if(sum!=(a*b))

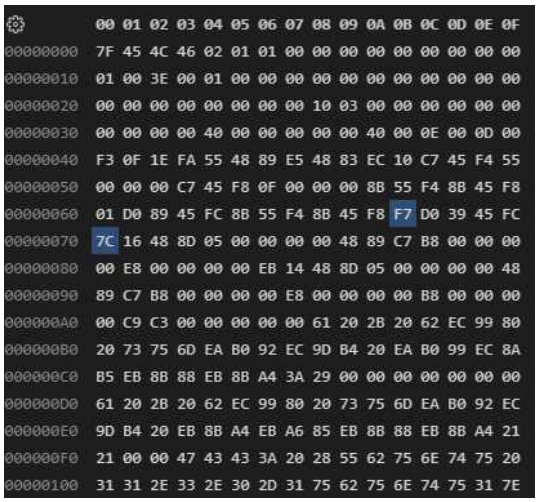


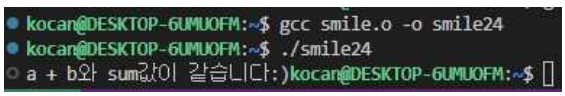
xxiii. if(sum!=(a+b)) -> if(sum!=(a/b))



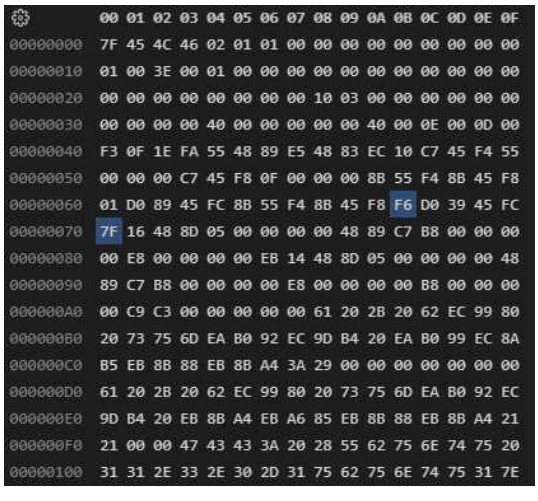


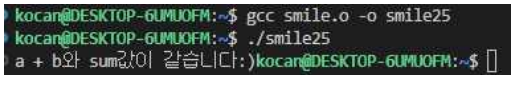
xxiv. if(sum!=(a+b)) -> if(sum>(a/b)) (JE -> JL)



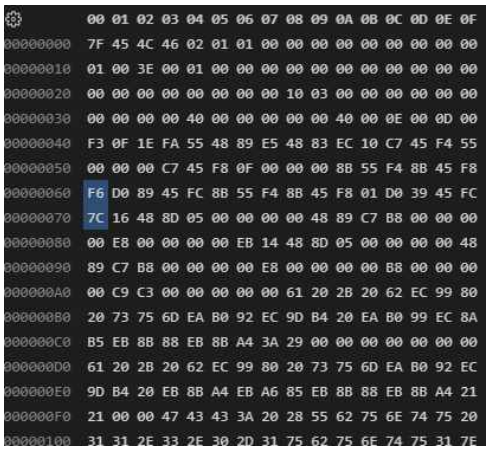


xxv. if(sum!=(a+b)) -> if(sum<(a*b)) (JE -> JG)



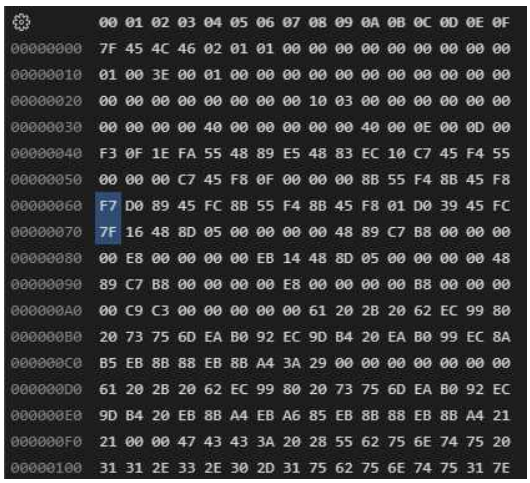


xxvi. int sum=a+b, if(sum!=(a+b)) -> int sum=a*b, if(sum>(a+b)) (JE -> JL)



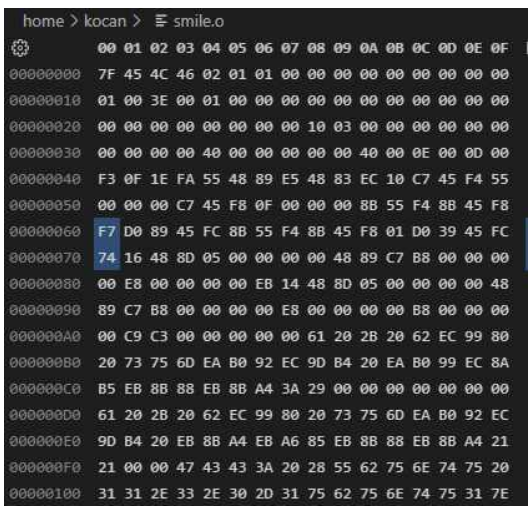

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile27
kocan@DESKTOP-6UMUOFM:~$ ./smile27
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

xxvii. int sum= a+b, if(sum!=(a+b)) -> int sum= a/b, if(sum<(a+b)) (JE -> JG)



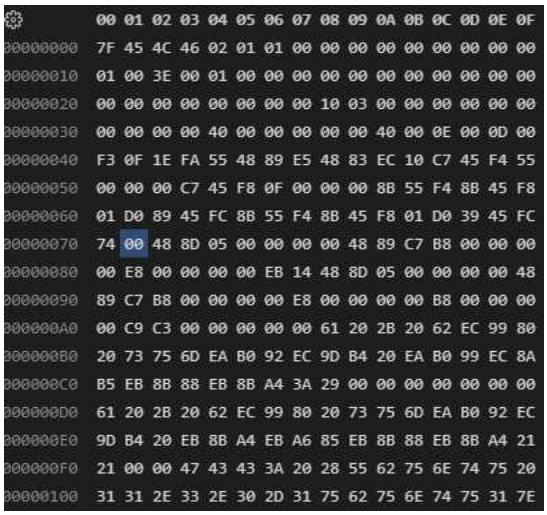

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile28
kocan@DESKTOP-6UMUOFM:~$ ./smile28
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

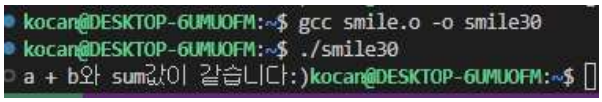
xxviii. int sum=a+b -> int sum=a/b



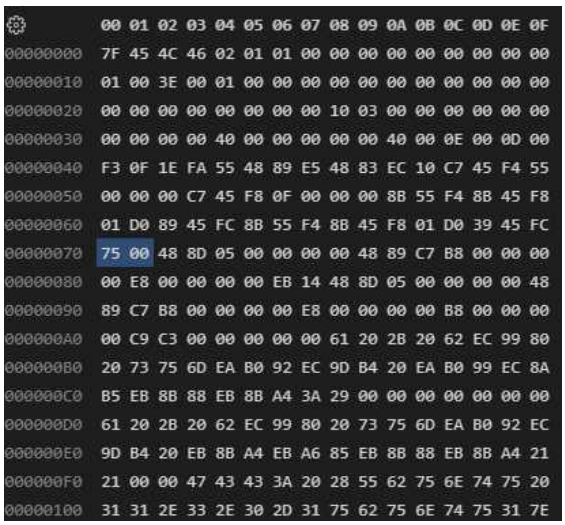

```
kocan@DESKTOP-6UMUOFM:~$ gcc smile.o -o smile29
kocan@DESKTOP-6UMUOFM:~$ ./smile29
a + b와 sum값이 같습니다:)kocan@DESKTOP-6UMUOFM:~$
```

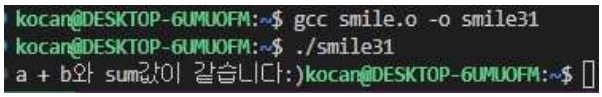
xxix. sum과 a+b가 같은 경우 true 문을 통과할 수 있도록 점프 지점을 true 문 바로 앞 주소로 지정





xxx. if(sum!=(a+b)) -> if(sum==(a+b)) (JE -> JNE), sum과 a+b가 다른 경우에 점프하는 주소 지정





4) SetUID Program

1.

(1) passwd, chsh, su, sudo를 사용할 때 Set-UID가 필요한 이유는 앞서 기술된 명령어들은 보안상 중요한 역할을 수행하므로 이러한 명령어를 일반 사용자가 실행할 때 루트 권한으로 실행되도록 Set-UID가 설정되어야 한다.

(2) 만약 Set-UID가 설정되어 있지 않다면 일반 사용자는 앞의 명령어가 필요로 하는 권한(루트 권한)을 얻을 수 없으므로 실행할 수 없다. 따라서 루트 권한으로 명령어를 실행해야 하는 번거로움이 발생한다.

(3) (a) passwd 명령어는 사용자 계정의 암호를 변경하는 명령어이다. 이 명령어는 시스템 파일에 접근하는데 일반 사용자는 시스템 파일에 쓰기 권한이 없으므로 Set-UID가 설정되어 있지 않다면 암호를 변경할 수 없다.

(b) chsh 명령어는 사용자의 로그인 shell을 변경하는 명령어이다. 사용자가 자신의 로그인 셸을 변경할 수

있도록 하기 위해 Set-UID로 설정되며, Set-UID가 없으면 사용자는 자신의 로그인 셸을 변경할 수 없다.

(c) su 명령어는 다른 사용자 계정으로 전환하는 명령어이다. Set-UID가 없으면 일반 사용자가 루트 권한으로 전환할 수 없다. 따라서 일반 사용자가 루트 권한으로 실행할 필요가 있는 명령어를 실행하려면 Set-UID가 필요하다.

(d) sudo 명령어는 다른 사용자 권한으로 명령어를 실행할 수 있는 명령어이다. Set-UID가 없으면 일반 사용자가 다른 사용자 권한으로 실행할 수 없는 명령어를 실행할 수 없으므로 일반 사용자가 특정한 권한이 필요한 명령어를 실행할 필요가 있는 경우 Set-UID가 필요하다.

(4)

```
kocan@DESKTOP-6UMUOFM:~/homework$ mkdir homework
```

- 복사한 파일을 저장하기 위한 homework 디렉토리를 생성한다.

```
kocan@DESKTOP-6UMUOFM:~$ ls
Practice1.c  modified  result4
Practice2.c  result1   result5
Practice2.o  result2   smile.c
homework     result3   smile.o
```

- ls 명령어로 homework 디렉터리를 생성된 모습을 확인할 수 있다.

```
kocan@DESKTOP-6UMUOFM:~$ cp /bin/passwd /home/kocan/homework/passwd
kocan@DESKTOP-6UMUOFM:~$ cp /bin/chsh /home/kocan/homework/chsh
kocan@DESKTOP-6UMUOFM:~$ cp /bin/su /home/kocan/homework/su
kocan@DESKTOP-6UMUOFM:~$ cp /bin/sudo /home/kocan/homework/sudo
```

- cp 명령어로 bin 디렉터리에 저장된 passwd, chsh, su, sudo 파일을 homework 디렉터리에 복사한다.

```
kocan@DESKTOP-6UMUOFM:~/homework$ ls
chsh  passwd  su  sudo
```

- ls 명령어로 passwd, chsh, su, sudo 파일이 정상적으로 복사된 모습을 확인할 수 있다.

```
kocan@DESKTOP-6UMUOFM:~/homework$ ls -al
total 396
drwxr-xr-x 2 kocan kocan   4096 Apr  8 16:15 .
drwxr-x--- 7 kocan kocan   4096 Apr  8 16:13 ..
-rwxr-xr-x 1 kocan kocan  44808 Apr  8 16:14 chsh
-rwxr-xr-x 1 kocan kocan  59976 Apr  8 16:14 passwd
-rwxr-xr-x 1 kocan kocan  55672 Apr  8 16:15 su
-rwxr-xr-x 1 kocan kocan 232416 Apr  8 16:15 sudo
```

- ls -al 명령어로 파일의 권한을 확인할 수 있다. 현재 Set-UID로 설정되어 있지 않으며, 소유자가 사용자로 되어있는 모습을 볼 수 있다.

```
kocan@DESKTOP-6UMUOFM:~/homework$ ./passwd
Changing password for kocan.
Current password:
```

- 복사된 passwd 파일을 실행한 모습이다. 패스워드 변경을 정상적으로 수행하는 모습을 볼 수 있다.

```
kocan@DESKTOP-6UMUOFM:~/homework$ chsh
Password:
Changing the login shell for kocan
Enter the new value, or press ENTER for the default
        Login Shell [/bin/zsh]:
```

- 복사된 chsh 파일을 실행한 모습이다. 사용자의 로그인 shell를 변경하는 모습을 확인할 수 있다.

```
kocan@DESKTOP-6UMUOFM:~/homework$ su
Password:
root@DESKTOP-6UMUOFM:/home/kocan/homework#
```

- 복사된 su 파일을 실행한 모습이다. 루트 계정에 정상적으로 로그인된 모습을 볼 수 있다.

```
kocan@DESKTOP-6UMUOFM:~/homework$ sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-ABknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-ABknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
usage: sudo [-ABbEHknPS] [-r role] [-t type] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory]
            [-T timeout] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-ABknS] [-r role] [-t type] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-R directory] [-T
            timeout] [-u user] file ...
```

- 복사된 sudo 파일을 실행한 모습이다. sudo 명령어가 정상적으로 실행되며, 각종 옵션이 출력되는 모습을 확인할 수 있다.

2.

(a)

```
kocan@DESKTOP-6UMUOFM:~$ mkdir tmp
```

- /bin/zsh를 복사해서 저장할 tmp 디렉터리를 생성한다.

```
kocan@DESKTOP-6UMUOFM:~$ su
Password:
root@DESKTOP-6UMUOFM:/home/kocan# cp /bin/zsh /home/kocan/tmp
```

- su 명령어로 루트 계정에 로그인한 다음, cp 명령어로 /bin/zsh 파일을 /tmp에 복사한다.

```
root@DESKTOP-6UMUOFM:/home/kocan/tmp# ls
zsh
root@DESKTOP-6UMUOFM:/home/kocan/tmp# chmod 4775 zsh
root@DESKTOP-6UMUOFM:/home/kocan/tmp# ls -al
total 1000
drwxr-xr-x 2 kocan kocan    4096 Apr  8 16:47 .
drwxr-x--- 7 kocan kocan    4096 Apr  8 16:47 ..
-rwsrwxr-x 1 root  root  1013328 Apr  8 16:47 zsh
```

- zsh 파일이 정상적으로 복사한 것을 확인할 수 있다.
- chmod 4775 zsh 명령어로 zsh 파일에 Set-UID를 설정한다.
- ls -al을 통해 zsh에 Set-UID가 설정된 모습을 확인할 수 있다.
- zsh의 소유자는 루트 계정이다.

```
kocan@DESKTOP-6UMUOFM:~/tmp$ ./zsh
DESKTOP-6UMUOFM#
```

- 일반 사용자 계정으로 로그인한 다음, zsh를 실행시키면 정상적으로 zsh shell로 변경된 모습을 볼 수 있다.
- 일반 사용자는 shell을 변경할 수 없지만, 현재 zsh에 Set-UID가 설정되어 있기에 zsh를 실행할 때 임시로 소유자 권한, 즉 루트 계정의 권한을 얻어 변경할 수 있다.

(b)

```
DESKTOP-6UMUOFM# su
Password:
root@DESKTOP-6UMUOFM:/home/kocan/tmp# cp /bin/bash bash
root@DESKTOP-6UMUOFM:/home/kocan/tmp# ls
bash  zsh
```

- su 명령어로 루트 계정에 로그인한 다음, cp 명령어로 /bin/bash 파일을 /tmp에 복사한다.
- ls 명령어를 통해 bash 파일이 tmp 디렉터리에 복사된 모습을 확인할 수 있다.

```
root@DESKTOP-6UMUOFM:/home/kocan/tmp# chmod 4775 bash
root@DESKTOP-6UMUOFM:/home/kocan/tmp# ls -al
total 2364
drwxr-xr-x 2 kocan kocan    4096 Apr  8 16:51 .
drwxr-x--- 7 kocan kocan    4096 Apr  8 16:47 ..
-rwsrwxr-x 1 root  root  1396520 Apr  8 16:51 bash
-rwsrwxr-x 1 root  root  1013328 Apr  8 16:47 zsh
```

- chmod 4775 bash 명령어로 bash 파일에 Set-UID를 설정한다.
- ls -al을 통해 bash에 Set-UID가 설정된 모습을 확인할 수 있다.
- bash의 소유자는 루트 계정이다.

```
DESKTOP-6UMUOFM# ./bash
bash-5.1$
```

- 일반 사용자 계정으로 로그인한 다음, bash를 실행시키면 정상적으로 bash shell로 변경된 모습을 볼 수 있다.
- 일반 사용자는 shell을 변경할 수 없지만, 현재 bash에 Set-UID가 설정되어 있기에 bash를 실행할 때 임시로 소유자 권한, 즉 루트 계정의 권한을 얻어 변경할 수 있다.

3.

```
kocan@DESKTOP-6UMUOFM:~$ su
Password:
root@DESKTOP-6UMUOFM:/home/kocan# cd /bin
root@DESKTOP-6UMUOFM:/bin# rm sh
root@DESKTOP-6UMUOFM:/bin# ln -s zsh sh
```

- su 명령어를 통해 루트 계정으로 로그인한다.
- cd /bin 명령어로 bin 디렉터리에 접근한다.
- rm sh로 기본 shell 파일을 삭제한다.
- ln -s zsh sh 명령어로 zsh shell을 sh라는 이름으로 심볼릭 링크를 생성한다.
- 이 과정을 통해 sh라는 이름으로 zsh shell을 사용할 수 있으며, 리눅스 시스템을 부팅할 때마다 자동으로 로드되어 zsh shell을 기본으로 사용할 수 있게 된다.

4.

```
root@DESKTOP-6UMUOFM:/home/kocan# export PATH=/home/kocan:$PATH ls
```

- export PATH=/home/kocan:$PATH ls를 통해 ls 명령어를 입력했을 때 /home/kocan에 저장된 ls를 실행할 수 있도록 환경변수를 설정한다.

```c
#include <stdlib.h>

int main()
{
    system("ls");
    return 0;
}
```

- do_ls.c 파일이다.
- 이 파일을 컴파일하고 실행할 경우(do_ls), ls 명령어를 실행한다.

```c
#include <stdlib.h>

int main()
{
    system("rm test.txt");
    return 0;
}
```

- ls.c 파일이다.
- 이 파일을 컴파일하고 실행할 경우(ls), test.txt 파일을 삭제하는 명령어를 실행한다.

(a)

```
root@DESKTOP-6UMUOFM:/home/kocan# chmod 4775 do_ls
```

- 컴파일한 do_ls 파일에 Set-UID를 설정한다.

```
root@DESKTOP-6UMUOFM:/home/kocan# chown root:root do_ls
```

- do_ls 파일의 소유자를 루트 계정으로 변경한다.

```
-rwsrwxr-x 1 root  root  15960 Apr  8 17:49 do_ls
```

- ls -al 명령어를 통해 do_ls 파일의 상태를 살펴보면 Set-UID가 설정되어 있으며, 소유자가 루트 계정으로 된 모습을 볼 수 있다.

```
-rwxr-xr-x 1 root  root  15960 Apr  8 18:10 ls
```

- ls 파일의 경우 소유자가 루트 계정으로 되어있으며, Set-UID는 설정하지 않았다.

```
DESKTOP-6UMUOFM% ./do_ls
rm: cannot remove 'test.txt': No such file or directory
```

- do_ls 파일을 실행시키면 do_ls 안에서 실행한 system("ls")를 통해 ls 명령어를 실행시킨다. 이때, PATH 환경변수를 수정하여 ls 명령어가 입력되면 /home/kocan 디렉터리에 저장된 ls 파일이 실행되도록 하였고, /home/kocan에 저장된 ls 파일은 system("rm test.txt")를 실행하기 때문에, test.txt 파일을 삭제하게 된다.
- 만약 PATH 환경변수가 수정되지 않았을 때, do_ls를 실행하면 /bin/ls를 실행하여, 디렉터리 내 파일 리스트를 출력하게 된다.
- do_ls에 Set-UID가 설정되어 있어 파일의 소유자가 루트 계정임에도 불구하고 일반 사용자 계정으로 실행될 때 임시로 루트 계정의 권한을 획득하기 때문에 do_ls가 실행된다.
- 또한, ls 파일의 소유자가 루트 계정이고, Set-UID가 설정되어 있지 않아 원래라면 일반 사용자는 ls 파일을 실행할 수 없지만, do_ls 파일에 Set-UID가 설정되어 있어 do_ls 파일을 실행할 때 임시로 루트 계정을 획득하기 때문에, do_ls 속 system("ls")를 실행할 때도 루트 권한으로 실행하게 된다. 따라서 일반 사용자라도 do_ls를 통해 간접적으로 ls를 실행할 수 있다.

(b)

```
kocan@DESKTOP-6UMUOFM:~$ ./do_ls
rm: cannot remove 'test.txt': No such file or directory
```

- do_ls 파일을 실행시키면 do_ls 안에서 실행한 system("ls")를 통해 ls 명령어를 실행시킨다. 이때, PATH 환경변수를 수정하여 ls 명령어가 입력되면 /home/kocan 디렉터리에 저장된 ls 파일이 실행되도록 하였고, /home/kocan에 저장된 ls 파일은 system("rm test.txt")를 실행하기 때문에, test.txt 파일을 삭제하게 된다.
- 만약 PATH 환경변수가 수정되지 않았을 때, do_ls를 실행하면 /bin/ls를 실행하여, 디렉터리 내 파일 리스트를 출력하게 된다.
- do_ls에 Set-UID가 설정되어 있어 파일의 소유자가 루트 계정임에도 불구하고 일반 사용자 계정으로 실행될 때 임시로 루트 계정의 권한을 획득하기 때문에 do_ls가 실행된다.
- 또한, ls 파일의 소유자가 루트 계정이고, Set-UID가 설정되어 있지 않아 원래라면 일반 사용자는 ls 파일을 실행할 수 없지만, do_ls 파일에 Set-UID가 설정되어 있어 do_ls 파일을 실행할 때 임시로 루트 계정을 획득하기 때문에, do_ls 속 system("ls")를 실행할 때도 루트 권한으로 실행하게 된다. 따라서 일반 사용자라도 do_ls를 통해 간접적으로 ls를 실행할 수 있다.

5.

(a)

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char *v[3];
    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }
    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
    /* Set q = 0 for Question a, and q = 1 for Question b */
    int q = 0;
    if (q == 0){
        char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
        sprintf(command, "%s %s", v[0], v[1]);
        system(command);
    }
    else execve(v[0], v, 0);
    return 0 ;
}
```

- q가 0이므로 system 함수를 실행하게 된다. 명령 인자에서 내용을 출력할 파일명을 가져와 “/bin/cat”과 concatenation을 하고, 이를 system 함수의 매개변수로 전달되어 파일을 출력하게 된다.

```
-rwsrwxr-x 1 root  root  16232 Apr  8 18:49 sys
```

- 위의 파일을 컴파일하여 sys라는 실행 파일로 생성하였다.
- 소유자는 루트 계정이며, Set-UID가 설정되어 이 파일이 실행될 때 임시로 루트 권한을 얻는다.

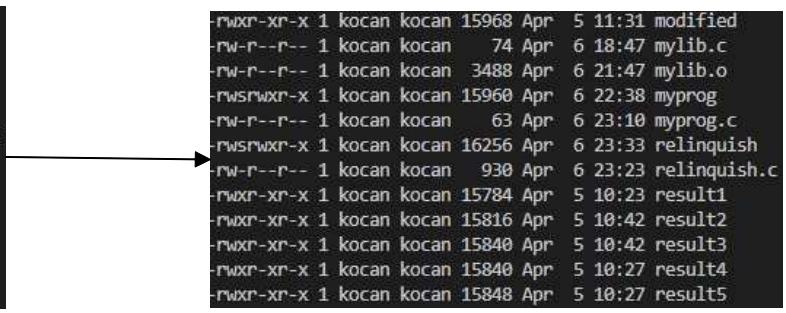
```
-rw-r--r-- 1 root  root     17 Apr  8 19:06 protected.txt
```

- 삭제할 대상의 파일은 소유자가 루트 계정이며, 일반 사용자에게 쓰기 금지가 되어있다.

```
DESKTOP-6UMUOFM% ./sys test_sys_exe.c; rm protected.txt
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char *v[3];
    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }
    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
    /* Set q = 0 for Question a, and q = 1 for Question b */
    int q = 1;
    if (q == 0){
        char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
        sprintf(command, "%s %s", v[0], v[1]);
        system(command);
    }
    else execve(v[0], v, 0);
    return 0 ;
}
rm: remove write-protected regular file 'protected.txt'? y
DESKTOP-6UMUOFM%
```

- sys 파일을 실행한 모습이다. 명령 인자로 test_sys_exe.c; rm protected.txt를 주었다.
- 실행하면 test_sys_exe.c의 내용이 출력되며, 세미콜론 뒤이어 나온 rm protected.txt.가 실행되었다.
- 쓰기가 금지되어 있기에 삭제할 것인지 물어보고, y를 입력하면 삭제하게 된다.

```
-rwxr-xr-x 1 kocan kocan 15968 Apr  5 11:31 modified
-rw-r--r-- 1 kocan kocan    74 Apr  6 18:47 mylib.c
-rw-r--r-- 1 kocan kocan  3488 Apr  6 21:47 mylib.o
-rwsrwxr-x 1 kocan kocan 15960 Apr  6 22:38 myprog
-rw-r--r-- 1 kocan kocan    63 Apr  6 23:10 myprog.c
-rw-r--r-- 1 root  root     17 Apr  8 19:06 protected.txt
-rwsrwxr-x 1 kocan kocan 16256 Apr  6 23:33 relinquish
-rw-r--r-- 1 kocan kocan   930 Apr  6 23:23 relinquish.c
-rwxr-xr-x 1 kocan kocan 15784 Apr  5 10:23 result1
-rwxr-xr-x 1 kocan kocan 15816 Apr  5 10:42 result2
-rwxr-xr-x 1 kocan kocan 15840 Apr  5 10:42 result3
-rwxr-xr-x 1 kocan kocan 15840 Apr  5 10:27 result4
-rwxr-xr-x 1 kocan kocan 15848 Apr  5 10:27 result5
```



- 이미지를 비교해보면 protected.txt가 일반 사용자 계정에서 쓰기 금지가 되어있으며, 파일이 실행된 후 삭제된 모습을 확인할 수 있다.
- system 함수는 쉘 환경에서 명령어를 실행하므로, 쉘 환경에서 명령어를 실행하면 명령어의 인자가 미리 정해

진 규칙에 따라 해석되고 실행된다. 따라서, 이처럼 shell 환경을 악용하여 명령어를 임의로 실행시키는 공격이 발생할 수 있다.

(b)

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char *v[3];
    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }
    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
    /* Set q = 0 for Question a, and q = 1 for Question b */
    int q = 1;
    if (q == 0){
        char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
        sprintf(command, "%s %s", v[0], v[1]);
        system(command);
    }
    else execve(v[0], v, 0);
    return 0 ;
}
```

q가 1이므로 execve 함수를 실행하게 된다. 명령 인자에서 내용을 출력할 파일명을 가져와 /bin/cat과 함께 execve 함수의 매개변수로 전달되어 파일을 출력하게 된다.

```
-rwsrwxr-x 1 root  root  16232 Apr  8 18:49 exe
```

- 위의 파일을 컴파일하여 exe라는 실행 파일로 생성하였다.
- 소유자는 루트 계정이며, Set-UID가 설정되어 이 파일이 실행될 때 임시로 루트 권한을 얻는다.

```
-rw-r--r-- 1 root  root     17 Apr  8 19:06 protected.txt
```

- 삭제할 대상의 파일은 소유자가 루트 계정이며, 일반 사용자에게 쓰기 금지가 되어있다.

```
DESKTOP-6UMUOFM% ./exe test_sys_exe.c; rm protected.txt
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char *v[3];
    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }
    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
    /* Set q = 0 for Question a, and q = 1 for Question b */
    int q = 1;
    if (q == 0){
        char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
        sprintf(command, "%s %s", v[0], v[1]);
        system(command);
    }
    else execve(v[0], v, 0);
    return 0 ;
}
rm: remove write-protected regular empty file 'protected.txt'? y
```

- exe 파일을 실행한 모습이다. 명령 인자로 test_sys_exe.c; rm protected.txt를 주었다.
- 실행하면 test_sys_exe.c의 내용이 출력되며, 세미콜론 뒤이어 나온 rm protected.txt.가 실행되었다.
- 쓰기가 금지되어 있기에 삭제할 것인지 물어보고, y를 입력하면 삭제하게 된다.

- 이미지를 비교해보면 protected.txt가 쓰기 금지가 되어있으며, 파일이 실행된 후 삭제된 모습을 확인할 수 있다.
- execve 함수가 명령어와 인자를 분리하여 처리하므로, execve 함수를 사용하면 명령어 인자를 엄격하게 검증할 필요가 없어진다. 다만 내가 실행했을 때는 system 함수와 마찬가지로 파일이 삭제되기 때문에, 중간에 인자 분리가 잘 안 되었거나 시스템의 문제로 제대로 작동하지 않을 가능성이 크다.

6.

(a)

```
DESKTOP-6UMUOFM% cat mylib.c
#include <stdio.h>
void sleep (int s)
{
printf("I am not sleeping!\n");
}
```

(b)

```
DESKTOP-6UMUOFM% gcc -fPIC -g -c mylib.c
DESKTOP-6UMUOFM% gcc -shared -Wl,-soname,libmylib.so.1 \
-o libmylib.so.1.0.1 mylib.o -lc
```

(c)

```
DESKTOP-6UMUOFM% export LD_PRELOAD=./libmylib.so.1.0.1
```

(d)

```
DESKTOP-6UMUOFM% cat myprog.c
#include <unistd.h>

int main()
{
    sleep(1);
    return 0;
}%
```

- Make myprog a regular program, and run it as a normal user

```
-rwxr-xr-x 1 kocan kocan 15960 Apr  8 23:39 myprog
```

- 소유자는 kocan이며, 소유자는 읽기, 쓰기, 실행이 가능하고, 그룹 사용자는 읽기와 실행, 그 외에는 실행만 가능하다.
- Set-UID는 설정되지 않았다.

```
DESKTOP-6UMUOFM% ./myprog
I am not sleeping!
```

- 앞서 정의한 mylib.c의 sleep 함수가 실행되어 I am not sleeping!이 출력되는 모습을 확인할 수 있다.

• Make myprog a Set-UID root program, and run it as a normal user.

```
-rwsrwxr-x 1 kocan2 kocan2 15960 Apr  8 23:24 myprog
```

- chown root:root myprog 명령어로 소유자를 kocan2로 변경했다.
- chmod 4775 myprog 명령어로 Set-UID를 설정하였다.

```
DESKTOP-6UMUOFM% ./myprog
DESKTOP-6UMUOFM%
```

- mylib.c의 sleep 함수가 아닌 시스템 함수 sleep이 실행되는 모습을 확인할 수 있다.

• Make myprog a Set-UID root program, and run it as a root account.

```
-rwsrwxr-x 1 kocan2 kocan2 15960 Apr  8 23:24 myprog
```

- chown root:root myprog 명령어로 소유자를 kocan2로 변경했다.
- chmod 4775 myprog 명령어로 Set-UID를 설정하였다.

```
DESKTOP-6UMUOFM% su
Password:
root@DESKTOP-6UMUOFM:/home/kocan# ./myprog
root@DESKTOP-6UMUOFM:/home/kocan#
```

- mylib.c의 sleep 함수가 아닌 시스템 함수 sleep이 실행되는 모습을 확인할 수 있다.

• Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), and run it as a different user (not-root user).

```
-rwsrwxr-x 1 kocan2 kocan2 15960 Apr  8 23:24 myprog
```

- chown kocan2:kocan2 myprog 명령어로 소유자를 kocan2로 변경했다.
- chmod 4775 myprog 명령어로 Set-UID를 설정하였다.

```
DESKTOP-6UMUOFM% ./myprog
DESKTOP-6UMUOFM%
```

- mylib.c의 sleep 함수가 아닌 시스템 함수 sleep이 실행되는 모습을 확인할 수 있다.

- runtime linker가 LD_PRELOAD 환경변수를 무시할 때는 Set-UID가 설정된 프로그램을 실행할 때이다. 왜냐하면 setuid 비트가 설정된 프로그램은 프로세스 권한을 변경해서 실행하는데 LD_PRELOAD 환경 변수를 이용하여 공격자가 프로그램의 동작을 조작하는 것을 막기 위해서이다.

7. a

```
-rwsrwxr-x 1 kocan kocan 16256 Apr  6 23:33 relinquish
```

- 프로그램을 컴파일한 후, chown root:root relinquish 명령어를 통해 소유자를 루트 계정으로 변경했다.
- chmod 4775 relinquish 명령어를 통해 Set-UID를 설정하였다.

```
DESKTOP-6UMUOFM% su
Password:
root@DESKTOP-6UMUOFM:/home/kocan# cd /etc
root@DESKTOP-6UMUOFM:/etc# touch zzz
```

- 루트 계정이 로그인하고 /etc 디렉터리로 이동해서 touch zzz 명령어를 통해 zzz라는 이름의 파일을 생성하였다.

```
root@DESKTOP-6UMUOFM:/etc# chmod 0644 zzz
```

- chmod 064 zzz 명령어를 통해 소유자는 읽기와 쓰기, 그 외의 사용자는 읽기만 가능하도록 권한을 설정하였

다.

```
-rw-r--r--  1 root root     0 Apr  6 23:24 zzz
```

- ls -al 명령어를 통해 zzz 파일의 접근 권한과 소유자를 확인하였다. 소유자와 접근 권한이 정상적으로 변경되었다.

```
DESKTOP-6UMUOFM% ./relinquish
```

-relinquish 파일을 실행하였다.

```
root@DESKTOP-6UMUOFM:/etc# vim zzz
```

- 실행 후 /etc 디렉터리로 이동해서 vim zzz 명령어를 통해 zzz 파일의 변화를 살펴보았다.
- zzz 파일을 살펴보면 아무런 변화가 없는 모습을 볼 수 있다.
- 프로그램을 살펴보면 도중에 루트 권한을 잃게 된다. 따라서 fork() 후 자식 프로세스에서 /etc/zzz 파일에 쓰기를 시도할 때 앞서 /etc/zzz의 파일 접근 권한을 0644로 설정했기 때문에, 일반 사용자 계정에는 쓰기 권한이 없어 파일에 내용이 기록되지 않는다.

3) 느낀 점

- 실습을 진행하면서 리눅스 운영 체제에 대한 이해도가 부쩍 늘어났다. 평소에 리눅스 운영 체제를 사용할 일이 없기에 처음에 실습을 시작했을 때는 명령어가 떠오르지 않고, 우왕좌왕했는데 실습을 진행하면서 각종 명령어를 터득하고, 파일을 이것저것 수정해보면서 리눅스 운영 체제가 어떻게 돌아가는지 어느 정도 알게 된 것 같다. 개인적으로 이번 실습에서 리눅스 운영 체제를 활용해본 것이 매우 재밌었다고 생각한다. windows 같은 GUI 환경에서 아이콘을 클릭하여 파일에 접근하는 것보다도 터미널에서 명령어를 입력하고 파일에 접근해서 여러 가지 작업을 하는 이 CLI 환경이 너무나도 흥미로웠다. 특히나 이번 실습에서 Set-UID를 설정하여 파일을 실행할 때 임시로 루트 권한을 얻는 것이 흥미로웠는데, windows에서도 관리자 권한으로 파일 실행이 존재하지만, 파일을 실행할 때마다 일일이 관리자 권한을 얻어야 하는데, 리눅스에서는 미리 setuid 비트를 설정하기 때문에 번거로움이 없었다. 그리고 명령어가 /bin 디렉터리에 파일로 저장되어있었는데, 명령어를 인식하고 실행하는 과정이 PATH 환경변수에 저장된 디렉터리를 순회하면서 명령어 파일이 존재하면 그 파일을 실행하는 방식인 것을 알고는 너무 신기했다. 이번 실습에서도 환경변수를 수정해서 명령어의 내용을 바꾸는 실습이 있었는데 이 실습을 통해서 많이 공부했었던 것 같다. 여러모로 Set-UID가 편리하면서도 보안상의 문제가 많다는 걸 이번 실습을 통해서 알게 되었는데 파일의 접근 권한을 빡빡하게 해야겠다는 생각이 들었다. 우리도 평소에 windows를 사용하면서 권한이 필요하다는 창이 뜨면 확인도 안하고 확인 버튼만 클릭하는데 이러한 행위가 여러모로 위험하다는 것을 깨달았다. 파일의 권한 설정에 대해서 귀찮아하지 않고 관심을 기울여야겠다는 생각이 들었다.