컴퓨터 보안_01

실습 4주차

동국대학교 CSDC Lab.
실습조교 김선규
2022.03.29 (Wed.)

Index

1. **프로그램 메모리 세그먼트 실습**
- 프로그램 메모리 세그먼트 구조란?
- 변수 선언과 초기화에 따른 세그먼트 크기 분석 실습
- 오브젝트 파일 수정 및 링킹(Linking) 실습 1
- 오브젝트 파일 수정 및 링킹(Linking) 실습 2

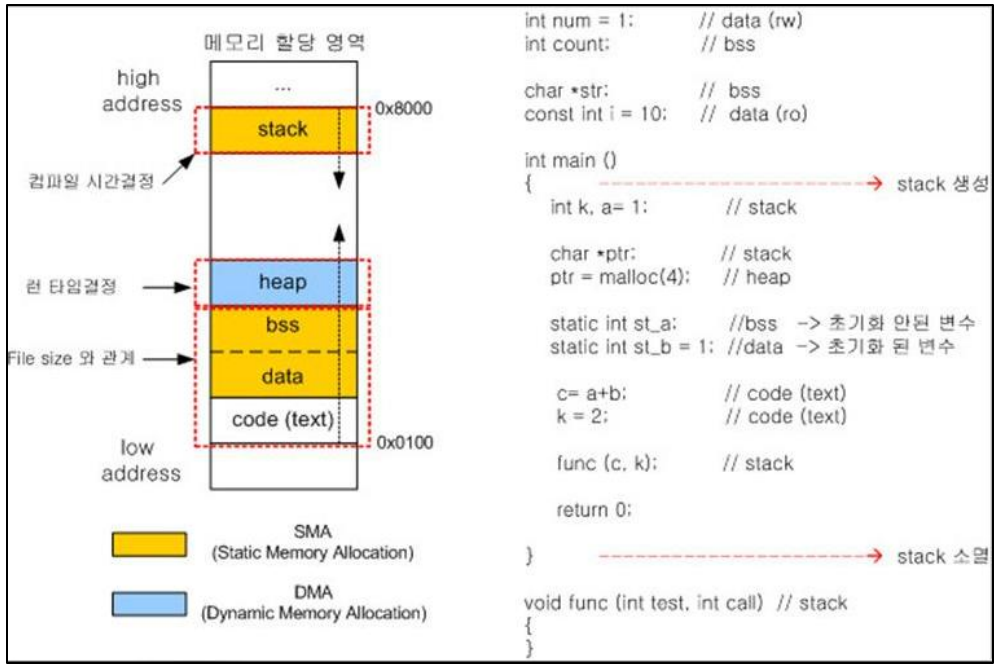
2. SetUID Program
- SetUID Program이란?
- SetUID Program 실습

3. Q & A

프로그램 메모리 세그먼트 실습

➢ 프로그램 메모리 세그먼트 구조란?

- 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. 이 때, 프로그램이 저장되는 메모리 공간은 일반적으로 code, data, bss, stack, heap 과 같이 5가지의 세그먼트로 분류된다.
- 세그먼트 방식은 가상주소(Virtual Address)인 논리적 주소(Logical Address)를 사용하여 프로그램에 따른 상대적인 위치를 지정한 후, 시작위치(Offset)를 더하는 방식으로 메모리의 물리적인 주소(Physical Address)에 접근하는 방식이다.



프로그램의 일반적인 메모리 세그먼트 구조

1. Text Segment 혹은 Code Segment (.code)
2. Initialized Data Segment (.data)
3. Uninitialized Data Segment (.bss)
4. Stack
5. Heap

프로그램 메모리 세그먼트 실습

dongguk UNIVERSITY

➢ 변수 선언과 초기화에 따른 세그먼트 크기 분석 실습

- 실습환경: Unix or linux 계열의 OS(e.g. wsl을 통해 설치한 Kali linux), Windows OS

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

- vi Practice1.c

```c
#include<stdio.h>

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

- gcc Practice1.c -o result1

- size result1

```
text    data     bss     dec     hex filename
1216     528       8    1752     6d8 result1
```

프로그램 메모리 세그먼트 실습

dongguk UNIVERSITY

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

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

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

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

6) 1), 2), 3), 4), 5)를 통해 확인한 세그먼트 크기들이 각각 다르다면 그 원인에 관해 분석하고 설명하시오.

프로그램 메모리 세그먼트 실습

dongguk UNIVERSITY

➢ 오브젝트 파일 수정 및 링킹(Linking) 실습 1

- 실습환경: Unix or linux 계열의 OS(e.g. wsl을 통해 설치한 Kali linux), Windows OS

1) 첫번째 실습에서 생성했던 result1 실행 파일을 대상으로 ELF 파일 형식을 확인하시오.

- readelf -h result1

```
└$ 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:          13912 (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:         30
  Section header string table index: 29
```

프로그램 메모리 세그먼트 실습

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

- vi Practice2.c

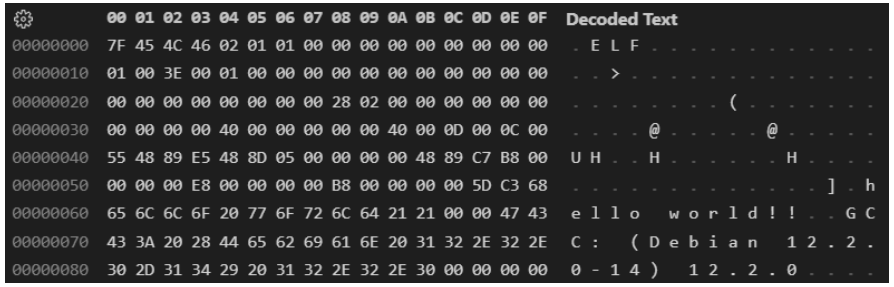
```c
#include<stdio.h>

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

- gcc -c Practice2.c
- readelf -h Practice2.o

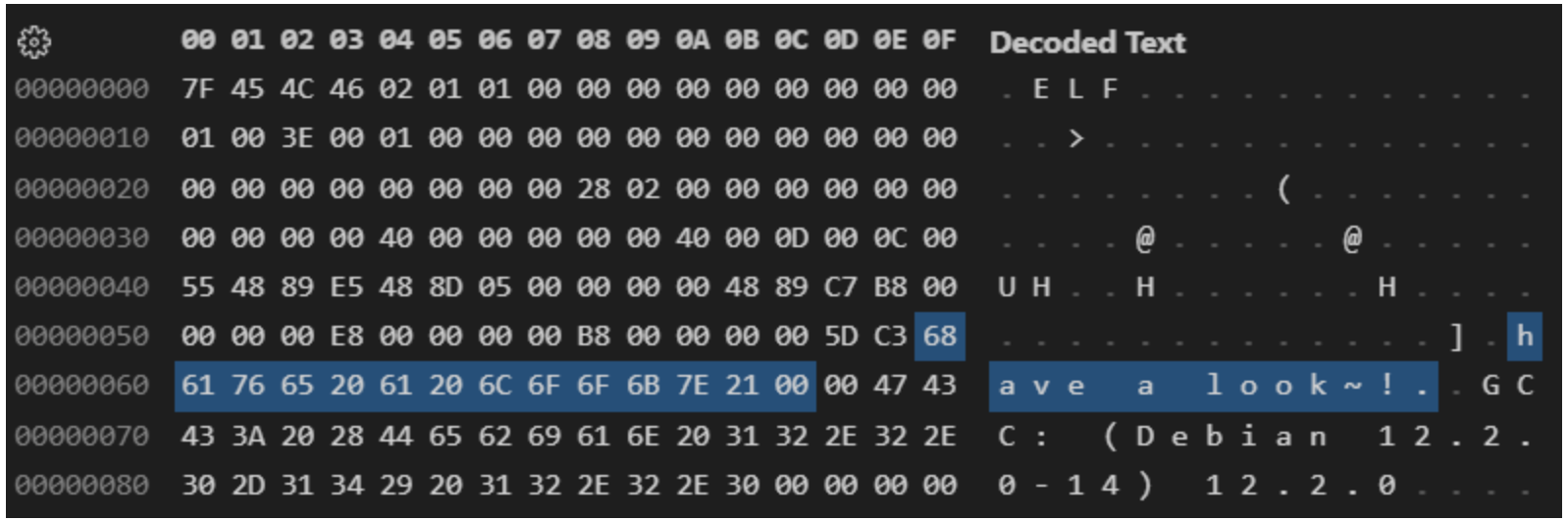
3) Hex Editer를 설치하여 Practice2.o 오브젝트 파일을 열어보고 어떤 정보들을 확인할 수 있는지 분석하시오.

- VScode에서 Hex Editor 확장 프로그램을 설치하여 확인하는 방법
- HxD Hex Editor를 설치하여 확인하는 방법 - Link



프로그램 메모리 세그먼트 실습

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



	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded Text
00000000	7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00	. E L F . . . .
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 28 02 00 00 00 00 00 00	. . . . (. . . . . . .
00000030	00 00 00 00 40 00 00 00 00 00 40 00 0D 00 0C 00	. . . . @ . . . . . @ . . . . .
00000040	55 48 89 E5 48 8D 05 00 00 00 00 48 89 C7 B8 00	U H . . H . . . . . . H . . . .
00000050	00 00 00 E8 00 00 00 00 B8 00 00 00 00 5D C3 68	. . . .] . h
00000060	61 76 65 20 61 20 6C 6F 6F 6B 7E 21 00 00 47 43	a v e a l o o k ~ ! . . G C
00000070	43 3A 20 28 44 65 62 69 61 6E 20 31 32 2E 32 2E	C : (D e b i a n 1 2 . 2 .
00000080	30 2D 31 34 29 20 31 32 2E 32 2E 30 00 00 00 00	0 - 1 4) 1 2 . 2 . 0 . . . .

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

- gcc Practice2.o -o modified

```
- └─$ ./modified
have a look~!
```

프로그램 메모리 세그먼트 실습

dongguk UNIVERSITY

➢ 오브젝트 파일 수정 및 링킹(Linking) 실습 2

- 실습환경: Unix or linux 계열의 OS(e.g. wsl을 통해 설치한 Kali linux), Windows OS

1) "smile.c" 프로그램을 다음과 같이 생성하고 컴파일을 통해 오브젝트 파일을 생성하시오.

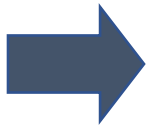
```c
#include<stdio.h>

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;
}
```

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)

```
└─$ ./smile
a+b와 sum값이 다릅니다!!
```



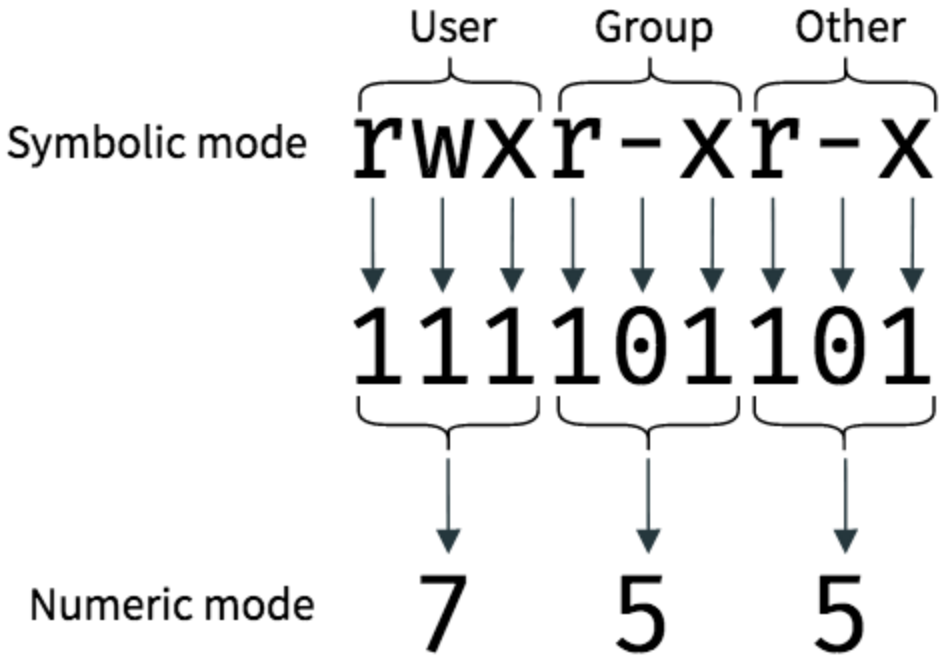
```
└─$ ./smile1
a+b와 sum값이 같습니다:)
```

dongguk UNIVERSITY

➢ SetUID Program이란?

File Mode		Owner	Group	File size (bytes)	Last modified date/time	File/directory name
drwxr-xr-x	3	root	root	4096	Jul 20 2016	**acpi**
-rw-r--r--	1	root	root	3028	Apr 21 2016	adduser.conf
drwxr-xr-x	2	root	root	4096	Mar 16 07:14	**alternatives**
drwxr-xr-x	3	root	root	4096	Mar 4 10:59	**apache2**
drwxr-xr-x	3	root	root	4096	Jul 20 2016	**apm**
drwxr-xr-x	3	root	root	4096	Mar 13 15:04	**apparmor**
drwxr-xr-x	8	root	root	4096	Mar 16 07:14	**apparmor.d**
drwxr-xr-x	3	root	root	4096	Feb 4 16:51	**apport**
drwxr-xr-x	6	root	root	4096	Mar 13 07:48	**apt**
-rw-r-----	1	root	daemon	144	Jan 14 2016	at.deny
-rw-r--r--	1	root	root	2188	Sep 1 2015	bash.bashrc
-rw-r--r--	1	root	root	45	Aug 12 2015	bash_completion
drwxr-xr-x	2	root	root	4096	Feb 25 23:18	**bash_completion.d**

- chmod 755 파일명



Octal Representation

0	000	-	-	-	No permissions
1	001	-	-	x	Only Execute
2	010	-	w	-	Only Write
3	011	-	w	x	Write and Execute
4	100	r	-	-	Only Read
5	101	r	-	x	Read and Execute
6	110	r	w	-	Read and Write
7	111	r	w	x	Read, Write and Execute

SetUID Program

➢ SetUID Program 실습

- Set-UID는 유닉스 운영체제에서 중요한 보호 메커니즘이다. 보통 소유자(owner) 권한에서 Set-UID 프로그램이 실행된다. 예를 들어, 프로그램의 소유자가 루트라면, 누구든지 이 프로그램을 실행시키면 프로그램이 실행되는 동안 그 프로그램은 루트의 권한을 얻게 된다. Set-UID를 통해서 우리는 많은 흥미있는 것들을 할 수 있지만, 불행히도 Set-UID는 또한 많은 나쁜 일들의 주범이기도 하다. 그러므로 이 실습의 목적은 두가지이다: (1) Set-UID가 왜 필요하고 어떻게 구현되고 있는지를 이해하고 이것의 좋은 면을 감상하는 것이다. (2) Set-UID의 잠재적인 보안 문제를 이해하고 나쁜 면을 주의하는 것이다.

➢ 과제 내용

- 별도의 파일(Set-UID Program Vulnerability Lab)에서 실습에 대한 자세한 내용과 과제를 설명한다. 여러분이 직접 해보고 경험한 것들을 실습 레포트에 자세히 반영하여 제출하시오. 또한 실습을 하는 동안 개인적으로 흥미가 있었거나 놀라운 경험에 대한 설명을 포함하는 것이 좋다.

실습 과제 및 보고서 진행방법 (1)

dongguk UNIVERSITY

➢ 실습 진행 방법

- 간단한 이론 복습 및 해당주차 실습문제 설명
- 실습 후 보고서를 작성하여 다음주 일요일 자정(23:59) 까지 E-Class에 제출(이메일 제출 불가, 반드시 E-Class를 통해 제출)
- 실습 과제 제출 기한 엄수 **(제출기한 이후로는 0점 처리)**

➢ 실습 보고서 [1/2]

- 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술
- 프로그램 설계 / 알고리즘
  - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
  - 문제 해결 과정 및 핵심 알고리즘 기술
- 결과 / 결과 분석
  - 결과 화면을 캡쳐 하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
- 소감
  - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

실습 과제 및 보고서 진행방법 (2)

dongguk UNIVERSITY

➢ 실습 보고서 [2/2]

- 제출 방법
  - 보고서를 작성하여 E-Class “과제” 메뉴를 통해 제출
    - **“이름_학번_실습주차.zip” 형태로 제출 (e.g. : 홍길동_2022123456_실습1주차.zip)**
    - 파일명에 공백, 특수 문자 등 사용 금지
    - 보고서 파일은 한글/워드 등 모두 상관 없지만 **PDF 변환** 후 이클래스 제출

- 유의사항
  - 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
  - 정해진 기한 내 제출
    - **기한을 넘길 시 0점 처리**
    - E-Class가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일 전에 제출
    - 당일 E-Class 오류로 인한 미제출은 불인정
  - 보고서를 자신이 작성하지 않은 경우 실습 전체 점수 0점 처리

Q & A