# 정적링킹 및 동적링킹 실습 과제

20185309
황명원

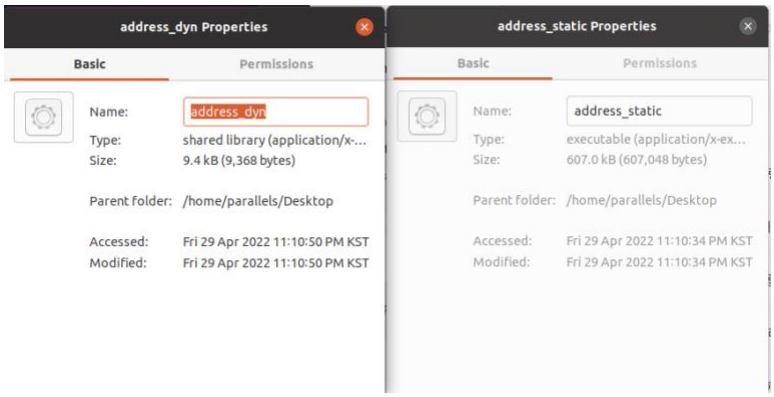
(1) 다음 코드를 동적 링킹으로 컴파일 및 정적 링킹으로 컴파일 해서 실행파일의 크기를 비교하세요.

우선 정적링킹 과 동적링킹 의 뜻을 살펴 보겠습니다.
정적링킹 이란 실행 가능한 목적 파일을 만들 때 프로그램에서 사용하는 모든 라이브러리 모듈을 복사하는 방식을 말합니다. 즉 저장하는 양이 많아집니다.
동적링킹 이란 실행 가능한 목적 파일을 만들 때 프로그램에서 사용하는 모든 라이브러리 모듈을 복사하지 않고 해당 모듈의 주소만을 가지고 있다가, 런타임에 실행 파일과 라이브러리가 메모리에 위치될 때 해당 모듈의 주소로 가서 필요한 것을 들고 오는 방식을 말합니다.
(이때 런타임은 운영체제에 의해 이루어집니다.)

저는 c 파일을 먼저 생성한 후 실행파일을 각각 동적,정적으로 생성해 두 실행파일의 크기를 직접비교해봤습니다.

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ vi address.c
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ gcc -o address_static address.c
-static
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ gcc -o address_dyn  address.c
```

(c파일 생성후 동적,정적으로 실행파일 생성)




두 실행파일을 비교해보면 동적으로 생성한 것 보다 정적으로 생성한 것이 훨씬 큰 것을 확인 할 수 있습니다.

(2) 그리고 해당 프로세스의 메모리맵을 출력해보기 바랍니다.

두개의 실행파일을 생성하고 백그라운드로 실행한 뒤 각각 메모리맵을 출력 해봤습니다.

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ ./address_static&
[2] 12647
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ num1 value is 3
num1 address is 0x48a048
num2 value is 4
num2 address is 0x48a04c

parallels@ubuntu-linux-20-04-desktop:~/Desktop$ cat /proc/12647/maps
00400000-00477000 r-xp 00000000 08:02 1310946                            /home/parallels/Desktop/address_static
00487000-0048a000 r--p 00077000 08:02 1310946                            /home/parallels/Desktop/address_static
0048a000-0048c000 rw-p 0007a000 08:02 1310946                            /home/parallels/Desktop/address_static
0048c000-0048e000 rw-p 00000000 00:00 0
02a9d000-02abf000 rw-p 00000000 00:00 0                                  [heap]
ffff84ed5000-ffff84ed7000 r--p 00000000 00:00 0                          [vvar]
ffff84ed7000-ffff84ed8000 r-xp 00000000 00:00 0                          [vdso]
ffffec9ee000-ffffeca0f000 rw-p 00000000 00:00 0                          [stack]
```

(정적으로 만든 실행파일의 메모리맵)

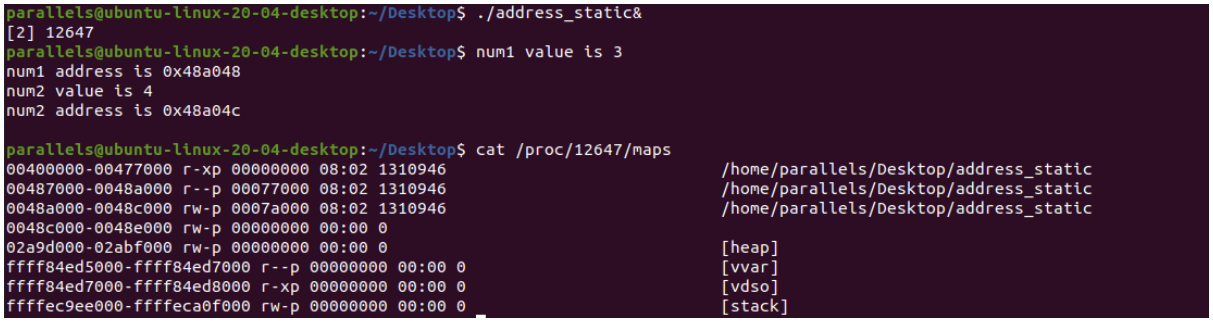
```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ ./address_dyn&
[1] 12484
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ num1 value is 3
num1 address is 0xaaaae7071010
num2 value is 4
num2 address is 0xaaaae7071014

parallels@ubuntu-linux-20-04-desktop:~/Desktop$ cat /proc/12484/maps
aaaae7060000-aaaae7061000 r-xp 00000000 08:02 1311192                    /home/parallels/Desktop/address_dyn
aaaae7070000-aaaae7071000 r--p 00000000 08:02 1311192                    /home/parallels/Desktop/address_dyn
aaaae7071000-aaaae7072000 rw-p 00001000 08:02 1311192                    /home/parallels/Desktop/address_dyn
aaab19835000-aaab19856000 rw-p 00000000 00:00 0                          [heap]
ffffa75fc000-ffffa7756000 r-xp 00000000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffa7756000-ffffa7766000 ---p 0015a000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffa7766000-ffffa7769000 r--p 0015a000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffa7769000-ffffa776c000 rw-p 0015d000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffa776c000-ffffa776f000 rw-p 00000000 00:00 0
ffffa7780000-ffffa77a1000 r-xp 00000000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffffa77ac000-ffffa77ae000 rw-p 00000000 00:00 0
ffffa77ae000-ffffa77b0000 r--p 00000000 00:00 0                          [vvar]
ffffa77b0000-ffffa77b1000 r-xp 00000000 00:00 0                          [vdso]
ffffa77b1000-ffffa77b2000 r--p 00021000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffffa77b2000-ffffa77b4000 rw-p 00022000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffffee54e000-ffffee56f000 rw-p 00000000 00:00 0                          [stack]
```

(동적으로 만든 실행파일의 메모리맵)

정적으로 만든 실행파일엔 없지만 동적으로 만든 실행파일의 메모리맵을 보면 .so로 되있는 파일이 heap에 쌓인것을 볼수 있습니다. 이는 동적으로 실행파일 만들시 정적으로 만든 실행파일과 다르게 printf와 같은 라이브러리 코드는 포함을 안한채로 저장하기 때문에 동적 라이브러리를 사용하기 위하여 위 사진 처럼 저장하는 것입니다. 정적으로 만든 실행파일은 이러한 라이브러리를 처음부터 저장하기 때문에 heap 에 저장할 필요가 없습니다.

(3) 아래 코드에서 num1 과 num2 가 메모리 맵에서 어떤 곳에 위치해 있는지 찾아보세요.

정적으로 실행한 파일의 메모리맵 사진을 참고로 살펴 보자면




num1의 주소는 0x48a048

num2의 주소는 0x48a04c 로 나와 있습니다.

0048a000-0048c000 rw-p 0007a000 08:02 1310946
/home/parallels/Desktop/address_static
cat으로 메모리맵 을 불러오면 위 두개의 주소는

0048a000-0048c000 이사이에 있는걸 확인할 수 있으므로
이 주소 안에 있는걸로 볼 수 있습니다.

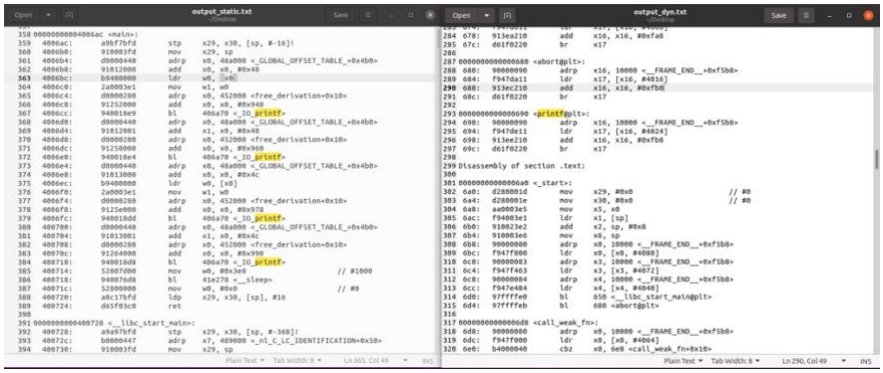
(동적으로 만든 실행파일도 이러한 방식으로 찾아볼수 있습니다.)

(4) objdump -D 실행파일 을 수행하면 기계어 코드가 출력이 되는데 실제 printf 함수가 정적링킹과 동적링킹에서 어떤 차이가 있는지 분석해서 보고서를 작성하기 바랍니다. 실행파일이 address 라고 했을때에 다음과 같이 파일로 저장한 후에 nano, vi 에디터로 내용을 확인해볼 수 있습니다.

각각 기계어 코드가 담긴 txt생성하기 위해

objdump -D address_static > output_static.txt
objdump -D address_dyn > output_dyn.txt

위 코드를 실행하려 생성 했습니다.




(생성한후 txt를 띄운 모습입니다.)

동적으로 실행파일을 생성할시 printf와 같은 함수를 정적으로 생성하는 실행파일 같이 저장하지 않으므로 동적 라이브러리를 사용해야하는 과정이 기계어 코드로 담겨져 있는 반면 정적으로 생성한 실행파일에선 메인함수에 있는 printf를 바로 인식하고 사용할려는 모습을 볼수 있습니다.