정적링킹 및 동적링킹 실습(도전과제)

20185309
황명원

(1)proc의 maps를 이용해서 각각의 변수가 어디에 있는지 찾아본다.

우선 data,stack,heap 에 어떤 것들이 저장 되있는지 간단히 정리해 보겠습니다.

1. data
   전역변수,정적변수,배열,구조체
2. stack
   지역변수,매개변수,복귀 번지
3. heap
   프로그래머가 동적으로 사용하는 영역(malloc,free,new,delete에 의하여 할당 또는 반환 되는 영역)

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ ./challenge&
[1] 4076
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ num1 value is 1
num1 address is 0xaaaadcb92010
num2 value is 3
num2 address is 0xaaaadcb92020
num3 value is 4
num3 address is 0xfffff90680e0
num4 value is 2
num4 address is 0xfffff90680e4
mp value is 5
mp address is 0xaaaaf7e412a0

parallels@ubuntu-linux-20-04-desktop:~/Desktop$ cat /proc/4076/maps
aaaadcb80000-aaaadcb81000 r-xp 00000000 08:02 1311309                    /home/parallels/Desktop/challenge
aaaadcb91000-aaaadcb92000 r--p 00001000 08:02 1311309                    /home/parallels/Desktop/challenge
aaaadcb92000-aaaadcb93000 rw-p 00002000 08:02 1311309                    /home/parallels/Desktop/challenge
aaaaf7e41000-aaaaf7e62000 rw-p 00000000 00:00 0                          [heap]
ffff97b6e000-ffff97cc8000 r-xp 00000000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffff97cc8000-ffff97cd8000 ---p 0015a000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffff97cd8000-ffff97cdb000 r--p 0015a000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffff97cdb000-ffff97cde000 rw-p 0015d000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffff97cde000-ffff97ce1000 rw-p 00000000 00:00 0
ffff97cf2000-ffff97d13000 r-xp 00000000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffff97d1e000-ffff97d20000 rw-p 00000000 00:00 0
ffff97d20000-ffff97d22000 r--p 00000000 00:00 0                          [vvar]
ffff97d22000-ffff97d23000 r-xp 00000000 00:00 0                          [vdso]
ffff97d23000-ffff97d24000 r--p 00021000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffff97d24000-ffff97d26000 rw-p 00022000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
fffff9049000-fffff906a000 rw-p 00000000 00:00 0                          [stack]
```

1)int num1 = 1;
-> 0xaaaadcb92010 이고 전역변수 이므로
aaaadcb92000-aaaadcb93000 rw-p 00002000 08:02 1311309 /home/parallels/Desktop/challenge
위 주소에 있습니다. (data 영역)

2)int num2;
-> 0xaaaadcb92020 이고 전역변수 이므로

aaaadcb92000-aaaadcb93000 rw-p 00002000 08:02 1311309 /home/parallels/Desktop/challenge
위 주소에 있습니다. (data 영역)

3)int num3 = 4;
-> 0xfffff90680e0 이고 지역변수 이므로
fffff9049000-fffff906a000 rw-p 00000000 00:00 0 [stack]
위 주소에 있습니다.(stack 영역)

4)int num4;
-> 0xfffff90680e4 이고 지역변수 이므로
fffff9049000-fffff906a000 rw-p 00000000 00:00 0 [stack]
위 주소에 있습니다.(stack 영역)

5)int *mp;
-> 0xaaaaf7e412a0 이고 malloc 에 의해 할당 되었으므로
aaaaf7e41000-aaaaf7e62000 rw-p 00000000 00:00 0 [heap]
위 주소에 있습니다.(heap 영역)

(2) 코드를 약간 수정해서 foo 함수의 주소값을 출력해보고, 메모리맵 어디에 위치하는지 찾아보시오.

수정한 코드:

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

int num1 = 1;
int num2;
int *mp;

int *foo();

int main(){

    int num3 = 4;

    num2 = 3;
```

```
    mp = malloc(sizeof(int));

    *mp = (int) 5;

    printf("num1 value is %d\n", num1);
    printf("num1 address is %p\n", &num1);
    printf("num2 value is %d\n", num2);
    printf("num2 address is %p\n", &num2);
    printf("num3 value is %d\n", num3);
    printf("num3 address is %p\n", &num3);
    printf("foo() value is %d\n", *foo());
    printf("foo() address is %p\n", foo());
    printf("mp value is %d\n", *mp);
    printf("mp address is %p\n", mp);

    sleep(1000);
    free(mp);
}
int *foo() {
    static int num_foo=2;
    return &num_foo;
}
```

foo 함수의 주소를 저장하기 위해 foo 앞에 *을 붙였고 그렇게 foo 의 값은 *foo()로, foo 의 주소는 foo()로 얻을수가 있었습니다.
그리고 foo 함수가 int 형 이기 때문에 foo 함수도 수정했습니다.
foo 함수안에 num_foo 에서 static 으로 하지 않으니 원하는대로 실행이 되지 않았고 num_foo 의 메모리가 유지되어야 다른 함수에서도 불러 쓸수 있기 때문에 정적변수로 사용했습니다.

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ vi challenge2.c
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ gcc -o challenge2 challenge2.c
```

(새로운 c 파일,challenge2.c 추가)

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ ./challenge2&
[2] 14096
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ num1 value is 1
num1 address is 0xaaaacd852010
num2 value is 3
num2 address is 0xaaaacd852020
num3 value is 4
num3 address is 0xffffd61e6374
foo() value is 2
foo() address is 0xaaaacd852014
mp value is 5
mp address is 0xaaab014eb2a0

parallels@ubuntu-linux-20-04-desktop:~/Desktop$ cat /proc/14096/maps
aaaacd840000-aaaacd841000 r-xp 00000000 08:02 1310835                    /home/parallels/Desktop/challenge2
aaaacd851000-aaaacd852000 r--p 00001000 08:02 1310835                    /home/parallels/Desktop/challenge2
aaaacd852000-aaaacd853000 rw-p 00002000 08:02 1310835                    /home/parallels/Desktop/challenge2
aaab014eb000-aaab0150c000 rw-p 00000000 00:00 0                          [heap]
ffffb028f000-ffffb03e9000 r-xp 00000000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffb03e9000-ffffb03f9000 ---p 0015a000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffb03f9000-ffffb03fc000 r--p 0015a000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffb03fc000-ffffb03ff000 rw-p 0015d000 08:02 1836414                    /usr/lib/aarch64-linux-gnu/libc-2.31.so
ffffb03ff000-ffffb0402000 rw-p 00000000 00:00 0
ffffb0413000-ffffb0434000 r-xp 00000000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffffb043f000-ffffb0441000 rw-p 00000000 00:00 0
ffffb0441000-ffffb0443000 r--p 00000000 00:00 0                          [vvar]
ffffb0443000-ffffb0444000 r-xp 00000000 00:00 0                          [vdso]
ffffb0444000-ffffb0445000 r--p 00021000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffffb0445000-ffffb0447000 rw-p 00022000 08:02 1836318                    /usr/lib/aarch64-linux-gnu/ld-2.31.so
ffffd61c7000-ffffd61e8000 rw-p 00000000 00:00 0                          [stack]
```

(challenge2 를 백그라운드로 수행하고 메모리맵 불러오기)

foo() 의 주소는 0xaaaacd852014 이고 정적변수로 함수를 만들었기 때문에 이 주소는 aaaacd852000-aaaacd853000 rw-p 00002000 08:02 1310835 /home/parallels/Desktop/challenge2

위 주소에 있습니다. (data 영역)

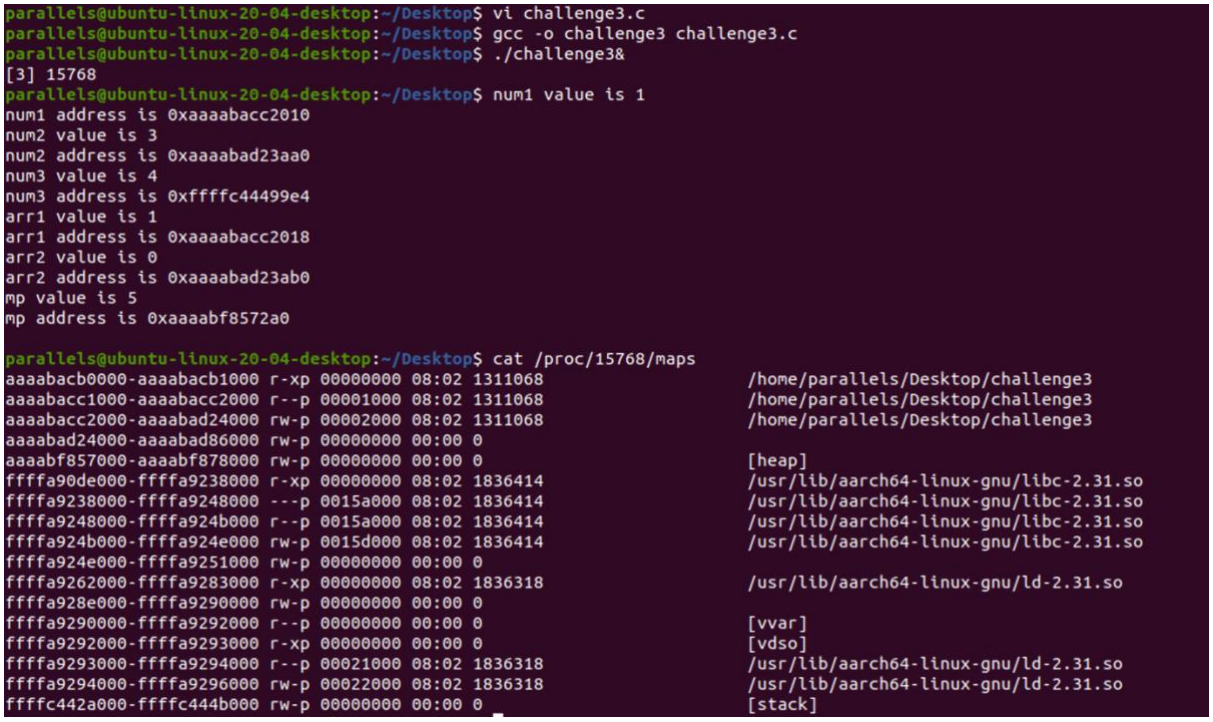
(3) 코드를 약간 수정해서 아래와 같이 arr1, arr2 를 추가하고 각 배열의 주소값을 출력해보자. 메모리맵 어떤 부분에 주소가 생성되었는지 분석해보자.

코드를 예제에 나온대로 수정한 후 새로운 c 파일을 만들고 동시에 실행과 메모리맵을 출력해 보았습니다.

수정한코드:

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

int num1 = 1;
int num2;
int arr1[100000] = {1,2,3};
int arr2[100000];
int *mp;


int main(){

    int num3 = 4;

    num2 = 3;



    mp = malloc(sizeof(int));

    *mp = (int) 5;

    printf("num1 value is %d\n", num1);
    printf("num1 address is %p\n", &num1);
    printf("num2 value is %d\n", num2);
    printf("num2 address is %p\n", &num2);
    printf("num3 value is %d\n", num3);
    printf("num3 address is %p\n", &num3);
    printf("arr1 value is %d\n", *arr1);
    printf("arr1 address is %p\n", arr1);
    printf("arr2 value is %d\n", *arr2);
    printf("arr2 address is %p\n", arr2);
    printf("mp value is %d\n", *mp);
    printf("mp address is %p\n", mp);

    sleep(1000);
    free(mp);
}
```

(배열의 값을 불러오기 위해 배열앞에 *를 붙였고 주소는 단순히 배열명만 써서 출력하도록 했습니다.)



(challenge3 파일을 만들고 실행파일 생성, 백그라운드로 실행후 메모리맵 출력)

arr1 을 보면
arr1 의 값은 1 (첫번째 값이 1 이기 때문입니다)이고, 주소 값은 0xaaaabacc2018 입니다.
이때 arr1 은 전역 변수이며 배열로 생성했으므로
aaaabacc2000-aaaabad24000 rw-p 00002000 08:02 1311068 /home/parallels/Desktop/challenge3
위 주소 안에 있습니다.(data 영역)

arr2 의 값은 0(첫번째 값이 없기 때문입니다.)이고, 주소 값은 0xaaaabad23ab0 입니다.
이때 arr2 는 전역변수이며 배열로 생성했으므로
aaaabacc2000-aaaabad24000 rw-p 00002000 08:02 1311068 /home/parallels/Desktop/challenge3
위 주소 안에 있습니다. (data 영역)