LAB 1 - Linker

정성태 2018-15515

Part 1

Part 1의 목표는 malloc(), calloc(), realloc(), free()를 통해 heap 공간에 새로 할당되거나 해제되어 사라지는 메모리의 크기를 추적할 수 있도록 runtime interpositioning을 수행하는 것이다.

별도로 만든 파일 memtrace.c에 malloc()을 비롯한 타깃 함수들을 새로 정의한 뒤, 이를 shared object 파일로 만들어서 다른 파일에서 사용할 수 있도록 했다. 이때 함수를 새로 정의하는 과정에서 기존의 libc에 정의되어 있는 함수들을 불러올 필요가 있는데, 이를 위해 <dlfcn.h>에 정의되어 있는 shared library를 불러올 수 있도록 해주는 함수들을 이용했다.

```c
void* malloc(size_t size) {
    void* ptr;

    // Get address of libc malloc
    if (!mallocp) {
        mallocp = dlsym(RTLD_NEXT, "malloc");
    }

    ptr = mallocp(size);
    LOG_MALLOC(size, ptr);

    n_malloc++;
    n_allocb += size;

    return ptr;
}
```

memtrace.c에서 새로 정의한 malloc() 함수는 위와 같다. dlsym()을 사용해 libc에 정의되어 있는 malloc 함수의 포인터를 불러와 mallocp에 저장하여 이용하는 방식이다. 이때 LOG_MALLOC() 매크로를 이용해 heap 메모리에 새로 할당되는 공간의 크기와 주소를 화면에 출력하도록 했으며, malloc의 호출 횟수를 나타내는 n_malloc과 할당된 메모리의 전체 크기를 나타내는 n_allocb 변수를 업데이트했다. 마지막에는 새로 할당된 메모리 공간을 가리키는 포인터 변수를 반환함으로써 libc에 정의된 malloc() 함수와 동일하게 동작하도록 했다.

calloc(), realloc(), free()도 이와 유사하게 libc에 정의된 함수를 dlsym()으로 불러와 사용하는 방식으로 재정의했다. 내부에서 LOG API를 사용해 메모리 할당 내역을 출력하도록 했으며, 통계에 사용할 변수를 적절히 업데이트하여 기록이 누적되도록 했다.

```c
__attribute__((destructor))
void fini(void)
{
    long n_alloc_total = n_malloc + n_calloc + n_realloc;
    long avg_allocb = n_allocb / n_alloc_total;

    LOG_STATISTICS(n_allocb, avg_allocb, 0L);

    LOG_STOP();

    // free list (not needed for part 1)
    free_list(list);
}
```

통계 결과를 화면에 출력하기 위해 shared library가 unload 될 때 호출되는 fini()를 위와 같이 수정했다. LOG_STATISTICS() 매크로를 사용해 할당된 메모리의 전체 크기, 1회 당 할당된 메모리의 평균 크기를 출력하도록 했다. free 된 메모리의 전체 크기는 아직 추적하도록 하지 않았기에 값을 0으로 출력하도록 했다.

```
~/system_programming/lab01_link/handout/part1
> make run test1
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/memli
st.c -ldl
[0001] Memory tracer started.
[0002]          malloc( 1024 ) = 0x55ac067232d0
[0003]          malloc( 32 ) = 0x55ac067236e0
[0004]          malloc( 1 ) = 0x55ac06723710
[0005]          free( 0x55ac06723710 )
[0006]          free( 0x55ac067236e0 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          0
[0012]
[0013] Memory tracer stopped.
```

테스트를 수행해본 결과 위와 같이 메모리 할당 및 해제 내역이 정상적으로 추적되는 것을 확인할 수 있었다.

Part 2

Part 2의 목표는 free()를 통해 해제된 메모리의 전체 크기를 추적하고, 할당되었으나 미처 해제되지 못한 메모리 공간이 존재하는지를 파악하는 기능을 추가하는 것이다.

이러한 기능을 구현하기 위해서는 할당된 메모리의 크기와 주소를 누적하여 기록해둘 필요가 있다. 그래야 특정 주소에 대해 free()가 호출되었을 때 그 메모리 공간의 크기가 얼마였는지를 확인할 수 있고, 마지막에 해제되지 않고 남아 있는 공간이 존재하는지를 파악할 수 있기 때문이다.

이를 위해 미리 구현되어 있는 linked list API를 활용했다. 새로운 메모리 공간이 할당될 때마다 그 정보를 linked list에 새로운 node로 저장하는 것이다. 이렇게 하면 free()가 호출될 때 해당 주소에 위치한 메모리 공간의 정보를 조회할 수 있게 된다.

```
void* malloc(size_t size) {
    void* ptr;

    // Get address of libc malloc
    if (!mallocp) {
        mallocp = dlsym(RTLD_NEXT, "malloc");
    }

    ptr = mallocp(size);
    LOG_MALLOC(size, ptr);

    n_malloc++;
    n_allocb += size;

    alloc(list, ptr, size);

    return ptr;
}
```

재정의한 malloc()의 내용을 위와 같이 수정했다. 할당 내역을 출력하고 통계 변수에 반영하는 것 외에도 alloc()을 사용해 linked list에 새로 할당된 공간의 정보를 기록하는 부분을 추가했다. calloc()과 realloc()에도 동일하게 새로 할당된 공간의 정보를 linked list에 기록하는 부분을 추가했다.

```
void free(void* ptr) {
    item* node;

    if (!freep) {
        freep = dlsym(RTLD_NEXT, "free");
    }

    freep(ptr);
    LOG_FREE(ptr);

    node = dealloc(list, ptr);
    n_freeb += node->size;
}
```

free()에는 dealloc()을 사용해 linked list에 등록되어 있는 정보를 조회하고 해제 처리하는 부분을 추가했다. 이렇게 하면 linked list의 해당 블록에 저장된 count 변수가 0으로 바뀌면서 여전히 할당 상태인 메모리 공간과 구별할 수 있게 된다. 또한 조회한 블록을 node에 저장하고 거기 저장된 크기 정보를 가져와 n_freeb를 업데이트함으로써 free()를 사용해 해제한 메모리 공간의 전체 크기를 추적할 수도 있다. realloc()도 메모리를 해제하는 동작을 포함하므로 이와 동일하게 dealloc()을 사용해 linked list의 블록을 불러온 뒤 크기 정보를 가지고 n_freeb를 업데이트하는 동작을 추가했다.

```c
__attribute__((destructor))
void fini(void)
{
    long n_alloc_total = n_malloc + n_calloc + n_realloc;
    long avg_allocb = n_allocb / n_alloc_total;
    item* node = list;
    bool found = false;

    LOG_STATISTICS(n_allocb, avg_allocb, n_freeb);

    while ((node = node->next) != NULL) {
        if (node->cnt > 0) {
            if (!found) {
                LOG_NONFREED_START();
                found = true;
            }
            LOG_BLOCK(node->ptr, node->size, node->cnt);
        }
    }

    LOG_STOP();

    // free list (not needed for part 1)
    free_list(list);
}
```

fini()에서 LOG_STATISTICS()를 호출할 때 n_freeb도 함께 출력하도록 했다. 또한 non-freed block을 추적하고 그 내용을 출력하기 위한 while 문을 추가했다. 이 while 문은 linked list의 첫 번째 node부터 시작해 순차적으로 다음 node를 살펴보면서 count가 0이 아닌, 즉 할당되었으나 해제되지는 않은 메모리 공간을 찾아내어 출력하는 방식으로 작동한다. non-freed 변수가 처음으로 발견되는 시점에만 LOG_NONFREED_START()를 호출하기 위해서 bool 변수인 found를 사용했다.

```
~/system_programming/lab01_link/handout/part2
> make run test1
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/memli
st.c -ldl
[0001] Memory tracer started.
[0002]          malloc( 1024 ) = 0x558e54f092d0
[0003]          malloc( 32 ) = 0x558e54f09710
[0004]          malloc( 1 ) = 0x558e54f09770
[0005]          free( 0x558e54f09770 )
[0006]          free( 0x558e54f09710 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          33
[0012]
[0013] Non-deallocated memory blocks
[0014]   block              size       ref cnt
[0015]   0x558e54f092d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

테스트 수행 결과 freed_total이 정상적으로 표시되며 non-freed block의 존재 역시 정상적으로 파악되

고 있는 것을 확인할 수 있었다.

Part 3

Part 3 의 목표는 free()가 호출될 때 double-free 와 illegal-free 를 판정하여 차단하는 기능을 추가하는 것이다.

이는 Part 2 에서 구현해놓은 linked list 를 이용해 구현할 수 있다.

```
void free(void* ptr) {
    item* node;

    LOG_FREE(ptr);

    // validity check
    node = find(list, ptr);
    if (node == NULL) {
        LOG_ILL_FREE();
        return;
    }
    if (node->cnt == 0) {
        LOG_DOUBLE_FREE();
        return;
    }

    if (!freep) {
        freep = dlsym(RTLD_NEXT, "free");
    }

    freep(ptr);

    node = dealloc(list, ptr);
    n_freeb += node->size;
}
```

free()의 내부에 위와 같이 validity check 를 수행하는 부분을 추가했다. free()가 호출되면 find()를 이용해 현재 linked list 입력된 주소와 일치하는 정보를 가진 블록이 있는지를 검사한다. 만약 그런 블록이 존재하지 않는다면(NULL) 할당된 적 없는 주소에 대해 free 를 시도한 것이므로 illegal-free 로 판정한다. 만약 그런 블록이 존재하지만 count 값이 0 이라면 할당되었다가 이미 해제된 주소이므로 double-free 로 판정한다. 이 두 경우에는 나머지 동작을 수행하지 않아야 하므로 에러 메시지를 출력한 후 바로 return 하도록 구현했다.

```
~/system_programming/lab01_link/handout/part3
> make run test4
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/memli
st.c -ldl
[0001] Memory tracer started.
[0002]          malloc( 1024 ) = 0x5617a96882d0
[0003]          free( 0x5617a96882d0 )
[0004]          free( 0x5617a96882d0 )
[0005]     *** DOUBLE_FREE  *** (ignoring)
[0006]          free( 0x1706e90 )
[0007]     *** ILLEGAL_FREE *** (ignoring)
[0008]
[0009] Statistics
[0010]   allocated_total      1024
[0011]   allocated_avg        1024
[0012]   freed_total          1024
[0013]
[0014] Memory tracer stopped.
```

테스트 결과 정상적으로 작동했다.

Review

과제를 수행하며 다음의 사실들을 새로 알게 되었다.

- realloc()은 기본적으로 입력된 주소에 할당된 메모리 공간을 확대하거나 축소하는 방식으로 작동하지만, 그것이 불가능할 경우 새로운 주소에 공간을 할당할 수도 있다. 따라서 입력된 포인터와 반환된 포인터의 주소가 반드시 일치하지는 않는다.
- dlsym() 호출 시 입력하는 RTLD_NEXT 는 첫 번째로 발견되는 함수는 무시하고 그 다음으로 발견되는 함수를 가져오라는 의미이다. 이 과제에서는 memtrace.c 에서 직접 정의한 함수를 건너뛰고 libc 에 정의된 기존의 함수를 불러오는 것이라고 해석할 수 있다.
- <stdbool.h>을 include 하면 bool 타입의 변수를 선언할 수 있으며 true 와 false 등 키워드도 사용할 수 있게 된다.

또한 다음과 같은 어려움이 있었다.

- 구현되어 있는 linked list API에서 alloc()과 dealloc()이 작동하는 방식을 제대로 이해하지 못해 여러 번 실패를 겪었다. 특히 각 블록의 cnt 값이 어떻게 변화하는지를 파악하기 위해 memlist.c를 천천히 읽어보아야 했다.

테스트 실행 결과

Part 1

Test 1

```
2018-15515@sp4:~/lab01/part1$ make run test1
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]           malloc( 1024 ) = 0x557b565432d0
[0003]           malloc( 32 ) = 0x557b565436e0
[0004]           malloc( 1 ) = 0x557b56543710
[0005]           free( 0x557b56543710 )
[0006]           free( 0x557b565436e0 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          0
[0012]
[0013] Memory tracer stopped.
```

Test 2

```
2018-15515@sp4:~/lab01/part1$ make run test2
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]           malloc( 1024 ) = 0x5651b0b702d0
[0003]           free( 0x5651b0b702d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          0
[0009]
[0010] Memory tracer stopped.
```

Test 3

```
2018-15515@sp4:~/lab01/part1$ make run test3
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]           calloc( 1 , 30301 ) = 0x557dd94e92d0
[0003]           malloc( 20121 ) = 0x557dd94f0940
[0004]           calloc( 1 , 55900 ) = 0x557dd94f57f0
[0005]           calloc( 1 , 3216 ) = 0x557dd9503260
[0006]           malloc( 48597 ) = 0x557dd9503f00
[0007]           calloc( 1 , 11289 ) = 0x557dd950fce0
[0008]           malloc( 41592 ) = 0x557dd9512910
[0009]           malloc( 26553 ) = 0x557dd951cb90
[0010]           calloc( 1 , 21076 ) = 0x557dd9523360
[0011]           calloc( 1 , 13080 ) = 0x557dd95285c0
[0012]           free( 0x557dd95285c0 )
[0013]           free( 0x557dd9523360 )
[0014]           free( 0x557dd951cb90 )
[0015]           free( 0x557dd9512910 )
[0016]           free( 0x557dd950fce0 )
[0017]           free( 0x557dd9503f00 )
[0018]           free( 0x557dd9503260 )
[0019]           free( 0x557dd94f57f0 )
[0020]           free( 0x557dd94f0940 )
[0021]           free( 0x557dd94e92d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      271725
[0025]   allocated_avg        27172
[0026]   freed_total          0
[0027]
[0028] Memory tracer stopped.
```

Part 2

Test 1

```
2018-15515@sp4:~/lab01/part2$ make run test1
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]          malloc( 1024 ) = 0x55fa5f41e2d0
[0003]          malloc( 32 ) = 0x55fa5f41e710
[0004]          malloc( 1 ) = 0x55fa5f41e770
[0005]          free( 0x55fa5f41e770 )
[0006]          free( 0x55fa5f41e710 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          33
[0012]
[0013] Non-deallocated memory blocks
[0014]   block              size       ref cnt
[0015]   0x55fa5f41e2d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

Test 2

```
2018-15515@sp4:~/lab01/part2$ make run test2
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]          malloc( 1024 ) = 0x56175b4462d0
[0003]          free( 0x56175b4462d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

Test 3

```
2018-15515@sp4:~/lab01/part2$ make run test3
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]          calloc( 1 , 8077 ) = 0x5611e5b232d0
[0003]          malloc( 24263 ) = 0x5611e5b252a0
[0004]          calloc( 1 , 58994 ) = 0x5611e5b2b1a0
[0005]          malloc( 36689 ) = 0x5611e5b39850
[0006]          calloc( 1 , 23481 ) = 0x5611e5b427e0
[0007]          calloc( 1 , 39543 ) = 0x5611e5b483e0
[0008]          malloc( 55504 ) = 0x5611e5b51e90
[0009]          calloc( 1 , 56103 ) = 0x5611e5b5f7a0
[0010]          malloc( 19997 ) = 0x5611e5b6d300
[0011]          malloc( 59407 ) = 0x5611e5b72160
[0012]          free( 0x5611e5b72160 )
[0013]          free( 0x5611e5b6d300 )
[0014]          free( 0x5611e5b5f7a0 )
[0015]          free( 0x5611e5b51e90 )
[0016]          free( 0x5611e5b483e0 )
[0017]          free( 0x5611e5b427e0 )
[0018]          free( 0x5611e5b39850 )
[0019]          free( 0x5611e5b2b1a0 )
[0020]          free( 0x5611e5b252a0 )
[0021]          free( 0x5611e5b232d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      382058
[0025]   allocated_avg        38205
[0026]   freed_total          382058
[0027]
[0028] Memory tracer stopped.
```

Part 3

Test 4

```
2018-15515@sp4:~/lab01/part3$ make run test4
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]           malloc( 1024 ) = 0x55da99f232d0
[0003]           free( 0x55da99f232d0 )
[0004]           free( 0x55da99f232d0 )
[0005]     *** DOUBLE_FREE  *** (ignoring)
[0006]           free( 0x1706e90 )
[0007]     *** ILLEGAL_FREE *** (ignoring)
[0008]
[0009] Statistics
[0010]   allocated_total      1024
[0011]   allocated_avg        1024
[0012]   freed_total          1024
[0013]
[0014] Memory tracer stopped.
```

Test 5

```
2018-15515@sp4:~/lab01/part3$ make run test5
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]           malloc( 10 ) = 0x55bb0e7222d0
[0003]           realloc( 0x55bb0e7222d0 , 100 ) = 0x55bb0e722320
[0004]           realloc( 0x55bb0e722320 , 1000 ) = 0x55bb0e7223c0
[0005]           realloc( 0x55bb0e7223c0 , 10000 ) = 0x55bb0e7227e0
[0006]           realloc( 0x55bb0e7227e0 , 100000 ) = 0x55bb0e724f30
[0007]           free( 0x55bb0e724f30 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```