시스템 프로그래밍 과제 1(linklab) 보고서

2015-16828 김민규

1. part1

1.1 시행결과

test1의 시행결과

```
[0001] Memory tracer started.
[0002]           (nil) : malloc( 1024 ) = 0x18bd060
[0003]           (nil) : malloc( 32 ) = 0x18bd470
[0004]           (nil) : malloc( 1 ) = 0x18bd4a0
[0005]           (nil) : free( 0x18bd4a0 )
[0006]           (nil) : free( 0x18bd470 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          0
[0012]
[0013] Memory tracer stopped.
```

test2의 시행결과

```
[0001] Memory tracer started.
[0002]           (nil) : malloc( 1024 ) = 0x13e0060
[0003]           (nil) : free( 0x13e0060 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          0
[0009]
[0010] Memory tracer stopped.
```

test3의 시행결과

```
[0001] Memory tracer started.
[0002]           (nil) : calloc( 1 , 19683 ) = 0x1c26060
[0003]           (nil) : calloc( 1 , 241 ) = 0x1c2ad50
[0004]           (nil) : calloc( 1 , 9053 ) = 0x1c2ae50
[0005]           (nil) : calloc( 1 , 13498 ) = 0x1c2d1c0
[0006]           (nil) : malloc( 6877 ) = 0x1c30690
[0007]           (nil) : calloc( 1 , 19362 ) = 0x1c32180
[0008]           (nil) : calloc( 1 , 22634 ) = 0x1c36d30
[0009]           (nil) : calloc( 1 , 30006 ) = 0x1c3c5b0
[0010]           (nil) : malloc( 22625 ) = 0x1c43af0
[0011]           (nil) : calloc( 1 , 14997 ) = 0x1c49360
[0012]           (nil) : free( 0x1c49360 )
[0013]           (nil) : free( 0x1c43af0 )
[0014]           (nil) : free( 0x1c3c5b0 )
[0015]           (nil) : free( 0x1c36d30 )
[0016]           (nil) : free( 0x1c32180 )
[0017]           (nil) : free( 0x1c30690 )
[0018]           (nil) : free( 0x1c2d1c0 )
[0019]           (nil) : free( 0x1c2ae50 )
[0020]           (nil) : free( 0x1c2ad50 )
[0021]           (nil) : free( 0x1c26060 )
[0022]
[0023] Statistics
[0024]   allocated_total      158976
[0025]   allocated_avg        15897
[0026]   freed_total          0
[0027]
[0028] Memory tracer stopped.
```

test5의 시행결과

```
[0001] Memory tracer started.
[0002]           (nil) : malloc( 10 ) = 0x1eff060
[0003]           (nil) : realloc( 0x1eff060 , 100 ) = 0x1eff060
[0004]           (nil) : realloc( 0x1eff060 , 1000 ) = 0x1eff060
[0005]           (nil) : realloc( 0x1eff060 , 10000 ) = 0x1eff060
[0006]           (nil) : realloc( 0x1eff060 , 100000 ) = 0x1eff060
[0007]           (nil) : free( 0x1eff060 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          0
[0013]
[0014] Memory tracer stopped.
```

1.2 부연설명

수업 시간에 나왔던 것처럼, interpositioning을 이용해 기존의 malloc, calloc, realloc, free에서 추가 작업을 하게 만들었다. 주요 내용은 첫째로는 함수가 실행될 때마다 LOG함수들을 이용해 메모리가 어떻게 할당되고 free되는지 로그를 남기고, 둘째로는 모든

작업이 끝났을 때 fini 함수에서 LOG_STATISTICS함수를 이용해 할당된 메모리의 총합과 평균을 출력하게 하였다.

이를 위해서 dlsym을 이용해 interpositioning을 하였고 malloc, calloc, realloc함수가 불릴 때 마다 새로 할당되는 메모리를 n_allcob에 더해 총 할당된 메모리를 기록하였으며, 각각 n_malloc, n_calloc, n_realloc을 세서 메모리 할당이 몇 번 일어났는지 기록하여 평균을 계산할 수 있었다.

2. part2

2.1 시행 결과

test1의 시행결과

```
[0001] Memory tracer started.
[0002]          (nil) : malloc( 1024 ) = 0x12dd060
[0003]          (nil) : malloc( 32 ) = 0x12dd4c0
[0004]          (nil) : malloc( 1 ) = 0x12dd540
[0005]          (nil) : free( 0x12dd540 )
[0006]          (nil) : free( 0x12dd4c0 )
[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   caller
[0015]   0x12dd060          1024       1         ???:0
[0016]
[0017] Memory tracer stopped.
```

test2의 시행결과

```
[0001] Memory tracer started.
[0002]          (nil) : malloc( 1024 ) = 0x147c060
[0003]          (nil) : free( 0x147c060 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

test4의 시행결과

```
[0001] Memory tracer started.
[0002]           (nil) : malloc( 38297 ) = 0x2381060
[0003]           (nil) : calloc( 1 , 65289 ) = 0x238a660
[0004]           (nil) : calloc( 1 , 30291 ) = 0x239a5d0
[0005]           (nil) : calloc( 1 , 6827 ) = 0x23a1c80
[0006]           (nil) : malloc( 40878 ) = 0x23a3790
[0007]           (nil) : calloc( 1 , 38042 ) = 0x23ad7a0
[0008]           (nil) : malloc( 33736 ) = 0x23b6ca0
[0009]           (nil) : malloc( 61531 ) = 0x23bf0c0
[0010]           (nil) : calloc( 1 , 54888 ) = 0x23ce180
[0011]           (nil) : malloc( 43844 ) = 0x23db840
[0012]           (nil) : free( 0x23db840 )
[0013]           (nil) : free( 0x23ce180 )
[0014]           (nil) : free( 0x23bf0c0 )
[0015]           (nil) : free( 0x23b6ca0 )
[0016]           (nil) : free( 0x23ad7a0 )
[0017]           (nil) : free( 0x23a3790 )
[0018]           (nil) : free( 0x23a1c80 )
[0019]           (nil) : free( 0x239a5d0 )
[0020]           (nil) : free( 0x238a660 )
[0021]           (nil) : free( 0x2381060 )
[0022]
[0023] Statistics
[0024]   allocated_total      413623
[0025]   allocated_avg        41362
[0026]   freed_total          413623
[0027]
[0028] Memory tracer stopped.
```

test5의 시행결과

```
[0001] Memory tracer started.
[0002]           (nil) : malloc( 10 ) = 0x2452060
[0003]           (nil) : realloc( 0x2452060 , 100 ) = 0x24520d0
[0004]           (nil) : realloc( 0x24520d0 , 1000 ) = 0x2452190
[0005]           (nil) : realloc( 0x2452190 , 10000 ) = 0x24525d0
[0006]           (nil) : realloc( 0x24525d0 , 100000 ) = 0x24525d0
[0007]           (nil) : free( 0x24525d0 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```

2.2 부연설명

part1에서 추가된 점은 freed된 총 메모리 양을 구하고, 프로세스의 끝에 non-freed 된 메모리 블록이 있다면 그것에 대한 정보를 프린트한다는 것이다. 이를 위해서 메모리가 할당되거나 free될 때마다 업데이트되는 링크드 리스트를 활용하였다. malloc, calloc으로 메모리가 할당될 때마다 (구현되어 있는)alloc함수를 이용하여 링크드 리스트에 아이템(포인터, 사이즈 등의 정보로 이루어져 있다)을 새롭게 추가하였고, free시에는 링크드 리스트에서 find함수를 이용해 그 아이템을 찾고, dealloc함수를 이용해 ref cnt를 1 감소시켜 free된 아이템을 구분할 수 있게 표시하였다. realloc시에는 free후 allocate로 보았기 때문에, 기존의 포인터를 dealloc함수로 기록하고, alloc함수로 새롭게 받은 주소를 list에 기록하였다. 또한 free함수나 realloc함수가 불릴 때마다 find함수로 그 아이템을 찾아서, 그 아이템이 가지고 있는 사이즈를 n_freeb에 더해주어 총 free된 메모리의 양을 기록하게 하였다. part2에서는 free가 모두 정상적으로 일어난다고 가정하였다.

그리하여 결론적으로는 프로세스가 끝날 때 fini함수에서 free된 메모리양을 포함한 통계량을 출력한다. 그 다음 링크드 리스트를 처음부터 끝까지 돌면서 ref_cnt가 0이 아닌, 즉 free되지 않은 메모리 블록들을 찾아 그 정보를 출력하며, 이것이 Non-deallocated memory blocks항목이다. 만약 free되지 않은 메모리 블록이 없다면 이 항목은 출력하지 않게 되는데, 이는 링크드 리스트를 처음부터 쭉 돌며 ref cnt가 0이 아닌 아이템이 있는지 여부를 통해 판단한다.

3. part3

3.1시행결과

test1 시행결과

```
[0001] Memory tracer started.
[0002]         main:6  : malloc( 1024 ) = 0x7ee060
[0003]         main:10 : malloc( 32 ) = 0x7ee4c0
[0004]         main:1d : malloc( 1 ) = 0x7ee540
[0005]         main:25 : free( 0x7ee540 )
[0006]         main:2d : free( 0x7ee4c0 )
[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   caller
[0015]   0x7ee060           1024       1         main:6
[0016]
[0017] Memory tracer stopped.
```

test2 시행결과

```
[0001] Memory tracer started.
[0002]         main:9  : malloc( 1024 ) = 0xaa0060
[0003]         main:11 : free( 0xaa0060 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

test3 시행결과

```
[0001] Memory tracer started.
[0002]         main:37 : malloc( 2040 ) = 0xf8f060
[0003]         main:37 : malloc( 2049 ) = 0xf8f8b0
[0004]         main:6e : calloc( 1 , 37057 ) = 0xf90110
[0005]         main:37 : malloc( 36432 ) = 0xf99230
[0006]         main:6e : calloc( 1 , 44907 ) = 0xfa20e0
[0007]         main:6e : calloc( 1 , 10852 ) = 0xfad0b0
[0008]         main:6e : calloc( 1 , 28624 ) = 0xfafb70
[0009]         main:37 : malloc( 2232 ) = 0xfb6ba0
[0010]         main:6e : calloc( 1 , 508 ) = 0xfb74b0
[0011]         main:37 : malloc( 25373 ) = 0xfb7710
[0012]         main:97 : free( 0xfb7710 )
[0013]         main:97 : free( 0xfb74b0 )
[0014]         main:97 : free( 0xfb6ba0 )
[0015]         main:97 : free( 0xfafb70 )
[0016]         main:97 : free( 0xfad0b0 )
[0017]         main:97 : free( 0xfa20e0 )
[0018]         main:97 : free( 0xf99230 )
[0019]         main:97 : free( 0xf90110 )
[0020]         main:97 : free( 0xf8f8b0 )
[0021]         main:97 : free( 0xf8f060 )
[0022]
[0023] Statistics
[0024]   allocated_total      190074
[0025]   allocated_avg        19007
[0026]   freed_total          190074
[0027]
[0028] Memory tracer stopped.
```

test5 시행결과

```
[0001] Memory tracer started.
[0002]         main:9  : malloc( 10 ) = 0x229e060
[0003]         main:16 : realloc( 0x229e060 , 100 ) = 0x229e0d0
[0004]         main:23 : realloc( 0x229e0d0 , 1000 ) = 0x229e190
[0005]         main:30 : realloc( 0x229e190 , 10000 ) = 0x229e5d0
[0006]         main:3d : realloc( 0x229e5d0 , 100000 ) = 0x229e5d0
[0007]         main:45 : free( 0x229e5d0 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```

3.2 부연설명

part3은 memtrace.c 파일은 part2와 완전히 같다. 다만 각 함수들이 어디서 불렸는지 알기 위한 callinfo.c파일을 만들어야 하는데, 이를 위해서 libunwind 패키지를 사용하였다. 먼저 unw_getcontext와 unw_init_local함수로 커서를 초기화해주었다. 그 다음 커를 원하는 곳에 위치시켜야 했는데, 모든 alloc과 free함수는 main에서만 불려진다는 사실을 이용하였다. while문 안에서 unw_step으로 계속 커서를 전진시키며, 그 때의 커서에서

unw_get_proc_name함수를 이용해 함수명을 가리키는 포인터를 찾아내고, strcmp함수를 써 그 함수명이 main이면 while문을 멈추고 원하는 커서를 찾을 수 있었다. 그 커서에서 unw_get_proc_name으로 원하는 정보들을 기록하였으며 또 추가적으로, callq함수가 5바이트를 차지하고 unw_get_proc_name함수는 callq가 끝난 후의 offset 값을 구하므로, 구해진 offset에 5바이트를 빼 줌으로서 callq함수의 시작지점을 구할 수 있었다.

또한 동일한 포인터가 할당되었다 free후 다시 할당되는 경우, 맨 처음 할당되었을 때의 callinfo를 가져온다. 이는 alloc함수의 특성상, 링크드 리스트에 처음 저장할 때 callinfo를 모조리 저장하고, 그 다음 그 포인터에서 dealloc이나 alloc을 다시 하게 되면 size와 ref cnt만 변경하기 때문이다.

part3를 하면서 처음 난해했던 것은 unw함수들이 원하는 정보들이 리턴 값으로 오는 것이 아니라, 리턴 값은 오로지 함수가 오류없이 실행되었는지만 알려주는 값이고 전부 포인터를 넘겨줘 함수 안에서 직접 값을 변경하는 방식이었다는 점이다. 처음에는 이 구조를 파악하지 못해 어떻게 접근해야 할 지 난감했다.

4. bonus

4.1 시행결과

test4의 시행결과

```
[0001] Memory tracer started.
[0002]         main:6  : malloc( 1024 ) = 0x1f07060
[0003]         main:11 : free( 0x1f07060 )
[0004]         main:19 : free( 0x1f07060 )
[0005]             *** DOUBLE_FREE  *** (ignoring)
[0006]         main:23 : 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.
```

4.2 부연설명

part1,2,3에서는 나타나지 않는다고 가정한 double free, illegal free를 detect하고 프린트하는 문제였다. double free는 free한 포인터를 다시 free하는 것으로, 앞서 언급한 것처럼 링크드 리스트에서 ref cnt가 0이면 이미 free된 메모리 블록이라는 것을 활용하였다. 그래서 free함수에서 free하고자 하는 포인터를 find함수를 통해 링크드 리스트에서 찾고, ref cnt를 체크해 0이면 double free로 취급하여 실제 free는 이루어지지 않고 원래 free의 log와 double free의 log만 출력하게 하였다.

둘째로, illegal free는 이 프로세스에서 한 번도 할당되지 않은 메모리 주소를 free하고자 할 때 나타난다. 이는 마찬가지로 free함수에서 free하고자 하는 포인터를 find함수를 통해 링크드 리스트에서 찾고, 만약 찾지 못한다면(즉 find 함수가 NULL을 리턴한다면) illegal free로 취급하여 실제 free가 이루어지지 않고 원래 free의 log와 illegal free의 log만 출력하게 하였다.

realloc 부분이 조금 까다로웠다. free후 allocation으로 봤을 때, 1)free하고자 하는 포인터가 잘 free가 되는 경우 2) illegal free 인 경우 3) double free인 경우 세 가지로 나누어 각각 다뤄 줘야 했다. 1)의 경우에는 part2,3에서 한 대로 reallocation을 하면 되고, 2)의 경우는 free가 이루어질 수 없으므로 reallocp(NULL,size)를 대신 실행시켜 allocation만 이루어질 수 있게 하고 illegal free의 log를 출력하게 하였다. 마찬가지로 3)의 경우도 reallocp(NULL,size)를 실행시켜 allocation만 이루어지고, double free의 log를 출력하게 하였다.