

1. 실행 결과 (terminal 스크린샷)

1) Part1

```
2018-16371@sp4:~/handout/part1$ make run test1
cc -I. -I ../utils -o libmentrace.so -shared -fPIC mentrace.c ../utils/memlog.c ../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x5633772612d0
[0003] malloc( 32 ) = 0x5633772616e0
[0004] malloc( 1 ) = 0x563377261710
[0005] free( 0x563377261710 )
[0006] free( 0x5633772616e0 )
[0007]
[0008] Statistics
[0009] allocated_total    1057
[0010] allocated_avg       352
[0011] freed_total        0
[0012]
[0013] Memory tracer stopped.
2018-16371@sp4:~/handout/part1$ make run test2
cc -I. -I ../utils -o libmentrace.so -shared -fPIC mentrace.c ../utils/memlog.c ../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x55e99839a2d0
[0003] free( 0x55e99839a2d0 )
[0004]
[0005] Statistics
[0006] allocated_total    1024
[0007] allocated_avg     1024
[0008] freed_total      0
[0009]
[0010] Memory tracer stopped.
2018-16371@sp4:~/handout/part1$ make run test3
cc -I. -I ../utils -o libmentrace.so -shared -fPIC mentrace.c ../utils/memlog.c ../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 35409 ) = 0x5558919342d0
[0003] malloc( 63510 ) = 0x55589193cd30
[0004] calloc( 1 , 50902 ) = 0x55589194c550
[0005] malloc( 21589 ) = 0x555891958c30
[0006] malloc( 36130 ) = 0x55589195e090
[0007] calloc( 1 , 61262 ) = 0x555891966dc0
[0008] malloc( 15800 ) = 0x555891975d20
[0009] malloc( 48337 ) = 0x555891979ae0
[0010] calloc( 1 , 11583 ) = 0x5558919857c0
[0011] calloc( 1 , 43419 ) = 0x555891988510
[0012] free( 0x555891988510 )
[0013] free( 0x5558919857c0 )
[0014] free( 0x555891979ae0 )
[0015] free( 0x555891975d20 )
[0016] free( 0x555891966dc0 )
[0017] free( 0x55589195e090 )
[0018] free( 0x555891958c30 )
[0019] free( 0x55589194c550 )
[0020] free( 0x55589193cd30 )
[0021] free( 0x5558919342d0 )
[0022]
[0023] Statistics
[0024] allocated_total    387941
[0025] allocated_avg     38794
[0026] freed_total      0
[0027]
[0028] Memory tracer stopped.
2018-16371@sp4:~/handout/part1$ make run test4
cc -I. -I ../utils -o libmentrace.so -shared -fPIC mentrace.c ../utils/memlog.c ../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x5630d83ee2d0
[0003] free( 0x5630d83ee2d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
2018-16371@sp4:~/handout/part1$ make run test5
cc -I. -I ../utils -o libmentrace.so -shared -fPIC mentrace.c ../utils/memlog.c ../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 10 ) = 0x5582f47592d0
[0003] realloc( 0x5582f47592d0 , 100 ) = 0x5582f47592d0
[0004] realloc( 0x5582f47592d0 , 1000 ) = 0x5582f47592d0
[0005] realloc( 0x5582f47592d0 , 10000 ) = 0x5582f47592d0
[0006] realloc( 0x5582f47592d0 , 100000 ) = 0x5582f47592d0
[0007] free( 0x5582f47592d0 )
[0008]
[0009] Statistics
[0010] allocated_total    111110
[0011] allocated_avg     22222
[0012] freed_total      0
[0013]
[0014] Memory tracer stopped.
```

2) Part 2

```

3010-163710sp4:~/handout/part2$ make run test1
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x558240bf72d0
[0003] malloc( 32 ) = 0x558240bf7710
[0004] malloc( 1 ) = 0x558240bf7770
[0005] free( 0x558240bf7770 )
[0006] free( 0x558240bf7710 )
[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] 0x558240bf72d0  1024      1
[0016]
[0017] Memory tracer stopped.
3010-163710sp4:~/handout/part2$ make run test2
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x5555e7c7f2d0
[0003] free( 0x5555e7c7f2d0 )
[0004]
[0005] Statistics
[0006] allocated_total      1024
[0007] allocated_avg         1024
[0008] freed_total          1024
[0009]
[0010] Memory tracer stopped.
3010-163710sp4:~/handout/part2$ make run test3
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] calloc( 1, 55046 ) = 0x55f0dc0f82d0
[0003] calloc( 1, 47062 ) = 0x55f0dc105a10
[0004] malloc( 20423 ) = 0x55f0dc111220
[0005] malloc( 44973 ) = 0x55f0dc116220
[0006] malloc( 43252 ) = 0x55f0dc121210
[0007] calloc( 1, 16956 ) = 0x55f0dc12bb40
[0008] malloc( 15853 ) = 0x55f0dc12fde0
[0009] malloc( 38357 ) = 0x55f0dc133bf0
[0010] malloc( 31157 ) = 0x55f0dc13d200
[0011] malloc( 57006 ) = 0x55f0dc144bf0
[0012] free( 0x55f0dc144bf0 )
[0013] free( 0x55f0dc13d200 )
[0014] free( 0x55f0dc133bf0 )
[0015] free( 0x55f0dc12fde0 )
[0016] free( 0x55f0dc12bb40 )
[0017] free( 0x55f0dc121210 )
[0018] free( 0x55f0dc116220 )
[0019] free( 0x55f0dc111220 )
[0020] free( 0x55f0dc105a10 )
[0021] free( 0x55f0dc0f82d0 )
[0022]
[0023] Statistics
[0024] allocated_total      370085
[0025] allocated_avg         37008
[0026] freed_total          370085
[0027]
[0028] Memory tracer stopped.
3010-163710sp4:~/handout/part2$ make run test4
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x55b89aed02d0
[0003] free( 0x55b89aed02d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
3010-163710sp4:~/handout/part2$ make run tests
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 10 ) = 0x56320d23e2d0
[0003] realloc( 0x56320d23e2d0, 100 ) = 0x56320d23e320
[0004] realloc( 0x56320d23e320, 1000 ) = 0x56320d23e3c0
[0005] realloc( 0x56320d23e3c0, 10000 ) = 0x56320d23e7e0
[0006] realloc( 0x56320d23e7e0, 100000 ) = 0x56320d240f30
[0007] free( 0x56320d240f30 )
[0008]
[0009] Statistics
[0010] allocated_total      111110
[0011] allocated_avg         22222
[0012] freed_total          111110
[0013]
[0014] Memory tracer stopped.

```

3) Part 3

```

3010-163710sp4:~/handout/part3$ make run test1
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x56099b2ab2d0
[0003] malloc( 32 ) = 0x56099b2ab710
[0004] malloc( 1 ) = 0x56099b2ab770
[0005] free( 0x56099b2ab770 )
[0006] free( 0x56099b2ab710 )
[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] 0x56099b2ab2d0  1024      1
[0016]
[0017] Memory tracer stopped.
3010-163710sp4:~/handout/part3$ make run test2
cc -I. -I ../utils -o libmemtrace.so -shared -fPIC memtrace.c ../utils/memlog.c ../utils/menlist.c -ldl
[0001] Memory tracer started.
[0002] malloc( 1024 ) = 0x56302126c2d0
[0003] free( 0x56302126c2d0 )
[0004]
[0005] Statistics
[0006] allocated_total      1024
[0007] allocated_avg         1024
[0008] freed_total          1024
[0009]
[0010] Memory tracer stopped.

```

```

2018-163710sp4:~/handout/part3$ 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] malloc( 55615 ) = 0x55f509d4c2d0
[0003] calloc( 1, 27747 ) = 0x55f509d59c50
[0004] malloc( 53152 ) = 0x55f509d608f0
[0005] calloc( 1, 2330 ) = 0x55f509dd8bd0
[0006] calloc( 1, 35095 ) = 0x55f509d6e230
[0007] calloc( 1, 26592 ) = 0x55f509d76b80
[0008] malloc( 34619 ) = 0x55f509d7d3a0
[0009] calloc( 1, 22351 ) = 0x55f509d85b20
[0010] calloc( 1, 11599 ) = 0x55f509d8b2b0
[0011] malloc( 26375 ) = 0x55f509d8e040
[0012] free( 0x55f509d8e040 )
[0013] free( 0x55f509d8b2b0 )
[0014] free( 0x55f509d85b20 )
[0015] free( 0x55f509d7d3a0 )
[0016] free( 0x55f509d76b80 )
[0017] free( 0x55f509d6e230 )
[0018] free( 0x55f509d608f0 )
[0019] free( 0x55f509d59c50 )
[0020] free( 0x55f509d4c2d0 )
[0021]
[0022]
[0023] Statistics
[0024] allocated_total 295475
[0025] allocated_avg 29547
[0026] freed_total 295475
[0027]
[0028] Memory tracer stopped.
2018-163710sp4:~/handout/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 ) = 0x55e1b814a2d0
[0003] free( 0x55e1b814a2d0 )
[0004] free( 0x55e1b814a2d0 )
[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.
2018-163710sp4:~/handout/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 ) = 0x562c9afda2d0
[0003] realloc( 0x562c9afda2d0, 100 ) = 0x562c9afda320
[0004] realloc( 0x562c9afda320, 1000 ) = 0x562c9afda3c0
[0005] realloc( 0x562c9afda3c0, 10000 ) = 0x562c9afda7e0
[0006] realloc( 0x562c9afda7e0, 100000 ) = 0x562c9afdcf30
[0007] free( 0x562c9afdcf30 )
[0008]
[0009] Statistics
[0010] allocated_total 111110
[0011] allocated_avg 22222
[0012] freed_total 111110
[0013]
[0014] Memory tracer stopped.

```

2. 구현 방법

1) Part 1

Part 1에서는 네 개의 함수를 모두 구현하되 malloc, realloc, calloc 등 메모리 할당을 하는 부분만 트레이싱했다. 네 개의 함수는 함수 안에 dlsym(RTLD_NEXT, "함수명")을 활용하여 진짜 함수의 포인터를 받아오고, 이 포인터를 활용해 진짜 함수를 실행하는 방식으로 구현했다. 각 함수 모두 memlog.h에서 대응되는 LOG함수를 찾아 호출한다 (malloc – LOG_MALLOC, realloc-LOG_REALLOC, calloc – LOG_CALLOC, free – LOG_FREE). 트레이싱은 malloc, realloc, calloc 안에서 (재)할당한 크기만큼을 n_allocb에 누적하고, 각 함수의 실행 횟수를 기록(n_malloc, n_realloc, n_calloc에 저장)하는 방식으로 구현했다. 트레이싱 결과 구할 수 있는 allocated_total과 allocated_avg는 마지막에 출력되고, 각자 (n_allocb), (n_allocb/(n_malloc+n_calloc+n_realloc))으로 계산했다. 여기서 출력은 memlog의 LOG_STATISTICS를 사용했다. 구체적인 사항은 코드를 읽어 보면 알 수 있다.


2) Part 2

Part2에서는 Part1에서 구현한 코드들을 모두 동일하게 유지하되 free, realloc에서와 같이 할당한 메모리를 해제하는 부분을 추가로 트레이싱했다. 트레이싱은 함수 안에서 해제되는 메모리의 사이즈를 n_freeb에 누적하는 방식으로 구현했다. 또한 이를 위해 이때까지 어느 주소에 얼마만큼의 메모리를 할당했는지 추적할 필요가 있다. 따라서 memlist를 활용하여 메모리 할당 상황을 관리했다. 모든 함수는 메모리를 할당할 때마다

memlist의 alloc함수를 이용해서 리스트에 할당한 메모리 주소와 사이즈, 레퍼런스 하고 있는 개수 등을 담고 있는 `_item`을 추가한다. 그리고 메모리를 해제할 때에는 memlist의 `dealloc` 함수를 활용하여 해제하고자 하는 메모리의 정보를 담고 있는 `_item`의 정보를 수정한다. 트레이싱 결과 구할 수 있는 `freed_total`은 `n_freeb`로 마지막에 출력했다. Part2와 마찬가지로 `LOG_STATISTICS`를 사용했다. 또한 메모리 할당을 관리하는 리스트에 아직 해제되지 않은 메모리가 있다고 나오는 경우, 이 역시 `memlog`를 활용하여, `LOG_NON-FREED_START()`, `LOG_BLOCK()`으로 출력했다. 구체적인 사항은 코드를 읽어 보면 알 수 있다.

3) Part 3

Part3에서는 Part2까지 구현한 코드들을 모두 동일하게 유지하되 `illegal free`, `double free`로 인해 발생하는 에러를 `LOG`로 알려주고 무시하고 계속 진행할 수 있도록 했다. 자료에는 `realloc`, `free` 모두를 고려하라고 했지만, 아래 etl 질문글 답변에서 `realloc`의 경우는 무시해도 좋다는 답변을 받아 `free`내에서만 해당 에러 처리를 구현했다.



신혜원

2022-03-25 19:15


안녕하세요, 성민님

공유드셨던 test case외에 다른 case에 대해 test를 진행하고자 해당 부분을 추가로 써 두었습니다만, (원래 의미는 illegal free와 double free에 대해서 realloc역시 추가구현 하여라 입니다)

이번 과제에 대해서는 공유드셨던 test case(test1-5)에 대해서만 만족하도록 구현하면 됩니다.

-> 따라서 해당 부분은 무시해 주시면 됩니다.

감사합니다



이성민

2022-03-25 22:51

그렇게 하겠습니다. 감사합니다.

이 부분은 진짜 `free`함수를 호출 하기 전에 지금까지의 메모리 할당 내역을 관리하고 있는 `list`를 확인하여 적절한 메모리 해제인지를 확인하는 방식으로 구현했다. 적절하지 않은 메모리 해제는 요청이 들어온 `ptr`에 메모리 할당을 한 적이 없는 경우와 요청이 들어온 `ptr`이 이미 해제된 경우(해당 메모리를 레퍼런스하는 개수가 0인 경우)이다. 각 경우는 `memlist`의 `find`함수를 통해 해당 `ptr`에 해당하는 메모리 할당 및 해제 내역을 확인하여 구분될 수 있다. 전자의 경우 `LOG_ILL_FREE()`를 하고, 후자의 경우에는 `LOG_DOUBLE_FREE()`를 한다. 두 경우 모두 아니라면 그 때 앞서 구현한 메모리 해제 코드(실제 `free`함수를 부르고, `list`를 수정하거나 `n_freeb`를 관리하는 코드)를 실행한다.

3. 어려웠던 점

처음 part3을 구현할 때, "ignore illegal free (not to invoke error)"가 시그널 핸들러를 활용하라는 뜻인 줄 알아서 난항을 겪었다. 여기서 "not to invoke error"를 사용자 입장에서만 생각하여, 내부적으로는 에러가 발생해도 `memtrace` 프로세스에서 적절히 처리하여 `abort`되지 않도록 하라는 것으로 이해했던 것이다. 그러나 `illegal free`나 `double free`에서 발생하는 에러들(`SIGSEGV`, `SIGTRAP`)은 시그널 핸들러 처리를 해도 에러를 발생시킨 그 코드로 다시 돌아와서 에러를 또 발생시키기 때문에 naïve하게 짠 코드에서는 제대로 처리할 수 없었다. 다시 생각해 보니 먼저 `illegal free`, `double free`를 하게 하는지

를 판단하고 만약 그렇다면 아예 메모리 해제를 하지 못하도록 하면 된다는 생각이 들었다. 이렇게 구현하니 테스트도 모두 통과할 수 있었다. 앞으로 사회에 나가서 더 큰 시스템에서 이렇게 스펙 문서를 오독하면 큰일이 나겠다는 생각이 들어 앞으로는 좀 더 신중하게 생각하고 구현을 해야겠다고 다짐했다.

4. 새롭게 배운 점

라이브러리 인터포지셔닝을 이론으로 배웠을 때는 잘 기억에 남지 않았는데 이렇게 직접 구현을 해 보니 확실히 어떻게 이루어지는 것인지 이해할 수 있었다. 그리고 3에서 말한 이상한 행동을 했었기 때문에 확실히 SIGSEGV, SIGTRAP은 좀 더 정교한 코드로 핸들링해야한다는 사실도 배우고, 시그널 핸들러가 에러가 난 그 코드로 다시 돌아온다는 것도 배우게 되었다. 앞으로도 이렇게 직접 코드를 짜 보면서 공부해야겠다는 생각을 했다.