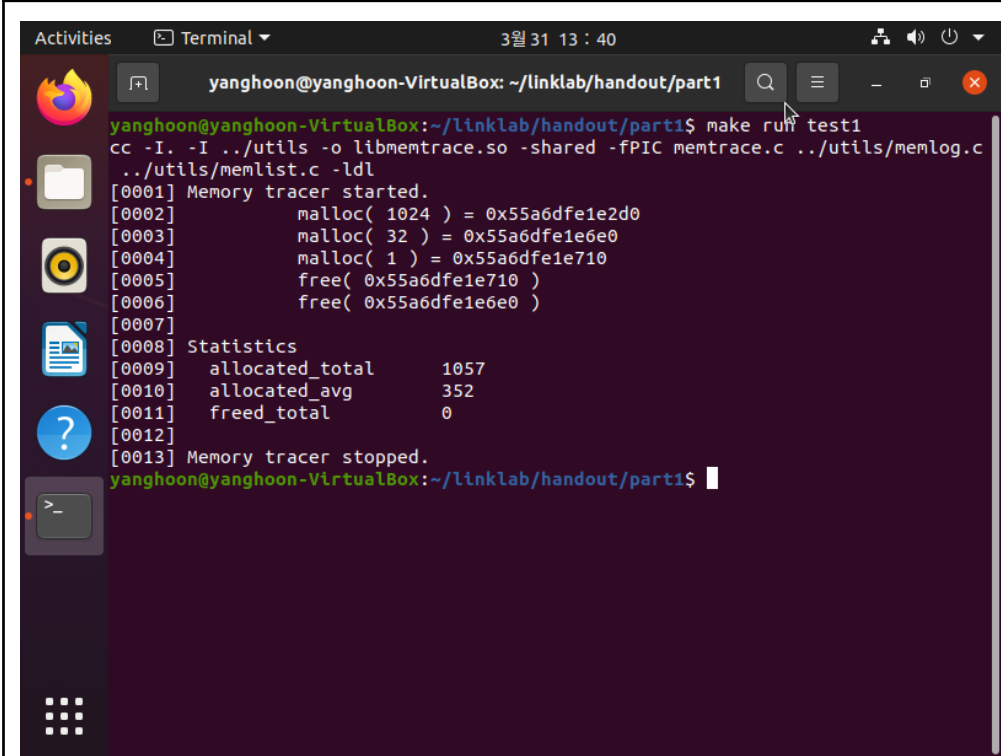


2021.03.31.
시스템 프로그래밍

Linklab : Report

사범대학 국어교육과
2017-14342 도양훈

1. 실행 결과 1.1. Part1



```
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part1
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$ make run test1
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x55a6dfe1e2d0
[0003]         malloc( 32 ) = 0x55a6dfe1e6e0
[0004]         malloc( 1 ) = 0x55a6dfe1e710
[0005]         free( 0x55a6dfe1e710 )
[0006]         free( 0x55a6dfe1e6e0 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          0
[0012]
[0013] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$
```

part1 \$ make run test1

```
Activities Terminal 3월 31 13 : 40
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part1
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$ make run test2
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]      malloc( 1024 ) = 0x55e6c58bd2d0
[0003]      free( 0x55e6c58bd2d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          0
[0009]
[0010] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$
```

part1 \$ make run test2

```
Activities Terminal 3월 31 13 : 40
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part1
[0001] Memory tracer started.
[0002]      malloc( 40741 ) = 0x55ff727ba2d0
[0003]      calloc( 1 , 35328 ) = 0x55ff727c4200
[0004]      malloc( 18242 ) = 0x55ff727ccc10
[0005]      malloc( 13727 ) = 0x55ff727d1360
[0006]      malloc( 7882 ) = 0x55ff727d4910
[0007]      calloc( 1 , 2532 ) = 0x55ff727d67f0
[0008]      malloc( 34164 ) = 0x55ff727d71e0
[0009]      calloc( 1 , 10355 ) = 0x55ff727df760
[0010]      malloc( 12950 ) = 0x55ff727e1fe0
[0011]      calloc( 1 , 35246 ) = 0x55ff727e5280
[0012]      free( 0x55ff727e5280 )
[0013]      free( 0x55ff727e1fe0 )
[0014]      free( 0x55ff727df760 )
[0015]      free( 0x55ff727d71e0 )
[0016]      free( 0x55ff727d67f0 )
[0017]      free( 0x55ff727d4910 )
[0018]      free( 0x55ff727d1360 )
[0019]      free( 0x55ff727ccc10 )
[0020]      free( 0x55ff727c4200 )
[0021]      free( 0x55ff727ba2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      211167
[0025]   allocated_avg        21116
[0026]   freed_total          0
[0027]
[0028] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$
```

part1 \$ make run test3

```
Activities Terminal 3월 31 13 : 40
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part1
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$ make run test4
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x55a3531b72d0
[0003]         free( 0x55a3531b72d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$
```

part1 \$ make run test4

아직 illegal free와 double free에 대한 처리를 하기 전이라 오류가 발생한다.

```
Activities Terminal 3월 31 13 : 40
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part1
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$ make run test5
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 10 ) = 0x55a9f78762d0
[0003]         realloc( 0x55a9f78762d0 , 100 ) = 0x55a9f78762d0
[0004]         realloc( 0x55a9f78762d0 , 1000 ) = 0x55a9f78762d0
[0005]         realloc( 0x55a9f78762d0 , 10000 ) = 0x55a9f78762d0
[0006]         realloc( 0x55a9f78762d0 , 100000 ) = 0x55a9f78762d0
[0007]         free( 0x55a9f78762d0 )
[0008]
[0009] Statistics
[0010]     allocated_total      111110
[0011]     allocated_avg       22222
[0012]     freed_total         0
[0013]
[0014] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part1$
```

part1 \$ make run test5

1.2. Part2

1.3.

```
Activities Terminal 3월 31 13 : 40
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part2
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$ make run test1
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x5623838c42d0
[0003]         malloc( 32 ) = 0x5623838c4710
[0004]         malloc( 1 ) = 0x5623838c4770
[0005]         free( 0x5623838c4770 )
[0006]         free( 0x5623838c4710 )
[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]   0x5623838c42d0    1024        1
[0016]
[0017] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$
```

part2 \$ make run test1

deallocated되지 않은 memory block이 있다면 이를 tracing하여 로그한다. 또한 statistics에도 freed_total이 제시된다.

```
Activities Terminal 3월 31 13 : 40
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part2
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$ make run test2
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x558369f302d0
[0003]         free( 0x558369f302d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$
```

part2 \$ make run test2

```
Activities Terminal 3월 31 13 : 41
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part2

[0001] Memory tracer started.
[0002]         malloc( 12937 ) = 0x564896e2e2d0
[0003]         malloc( 15644 ) = 0x564896e315a0
[0004]         malloc( 55565 ) = 0x564896e35300
[0005]         calloc( 1 , 22567 ) = 0x564896e42c50
[0006]         malloc( 829 ) = 0x564896e484b0
[0007]         calloc( 1 , 43951 ) = 0x564896e48830
[0008]         calloc( 1 , 12300 ) = 0x564896e53420
[0009]         calloc( 1 , 41003 ) = 0x564896e56470
[0010]         calloc( 1 , 46985 ) = 0x564896e604e0
[0011]         malloc( 28689 ) = 0x564896e6bcb0
[0012]         free( 0x564896e6bcb0 )
[0013]         free( 0x564896e604e0 )
[0014]         free( 0x564896e56470 )
[0015]         free( 0x564896e53420 )
[0016]         free( 0x564896e48830 )
[0017]         free( 0x564896e484b0 )
[0018]         free( 0x564896e42c50 )
[0019]         free( 0x564896e35300 )
[0020]         free( 0x564896e315a0 )
[0021]         free( 0x564896e2e2d0 )
[0022]
[0023] Statistics
[0024] allocated_total      280470
[0025] allocated_avg        28047
[0026] freed_total          280470
[0027]
[0028] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$
```

part2 \$ make run test3

```
Activities Terminal 3월 31 13 : 41
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part2

yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$ make run test4
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x5567cf1172d0
[0003]         free( 0x5567cf1172d0 )
[0004] free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$
```

part2 \$ make run test4

역시 아직 illegal free와 double free에 대한 처리는 구현하기 이전이다.

```
Activities Terminal 3월 31 13 : 41
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part2
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$ make run test5
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]     malloc( 10 ) = 0x558f6180f2d0
[0003]     realloc( 0x558f6180f2d0 , 100 ) = 0x558f6180f320
[0004]     realloc( 0x558f6180f320 , 1000 ) = 0x558f6180f3c0
[0005]     realloc( 0x558f6180f3c0 , 10000 ) = 0x558f6180f7e0
[0006]     realloc( 0x558f6180f7e0 , 100000 ) = 0x558f61811f30
[0007]     free( 0x558f61811f30 )
[0008]
[0009] Statistics
[0010]     allocated_total      111110
[0011]     allocated_avg       22222
[0012]     freed_total         111110
[0013]
[0014] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part2$
```

part2 \$ make run test5

1.4. Part3

```
Activities Terminal 3월 31 13 : 41
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part3
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$ make run test1
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]     malloc( 1024 ) = 0x56506e8692d0
[0003]     malloc( 32 ) = 0x56506e869710
[0004]     malloc( 1 ) = 0x56506e869770
[0005]     free( 0x56506e869770 )
[0006]     free( 0x56506e869710 )
[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]     0x56506e8692d0  1024      1
[0016]
[0017] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$
```

part3 \$ make run test1

```
Activities Terminal 3월 31 13 : 42
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part3
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$ make run test2
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x5564cbbfc2d0
[0003]         free( 0x5564cbbfc2d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$
```

part3 \$ make run test2

```
Activities Terminal 3월 31 13 : 42
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part3
[0001] Memory tracer started.
[0002]         malloc( 4580 ) = 0x565025ba62d0
[0003]         calloc( 1 , 31604 ) = 0x565025ba74f0
[0004]         calloc( 1 , 61976 ) = 0x565025baf0a0
[0005]         calloc( 1 , 13716 ) = 0x565025bbe2f0
[0006]         malloc( 62527 ) = 0x565025bc18c0
[0007]         calloc( 1 , 27765 ) = 0x565025bd0d40
[0008]         calloc( 1 , 12827 ) = 0x565025bd79f0
[0009]         calloc( 1 , 7630 ) = 0x565025bdac50
[0010]         calloc( 1 , 15249 ) = 0x565025bdca60
[0011]         malloc( 18648 ) = 0x565025be0630
[0012]         free( 0x565025be0630 )
[0013]         free( 0x565025bdca60 )
[0014]         free( 0x565025bdac50 )
[0015]         free( 0x565025bd79f0 )
[0016]         free( 0x565025bd0d40 )
[0017]         free( 0x565025bc18c0 )
[0018]         free( 0x565025bbe2f0 )
[0019]         free( 0x565025baf0a0 )
[0020]         free( 0x565025ba74f0 )
[0021]         free( 0x565025ba62d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      256522
[0025]   allocated_avg        25652
[0026]   freed_total          256522
[0027]
[0028] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$
```

part3 \$ make run test3

테스트 코드를 보니, 항상 랜덤한 size로 랜덤한 비율의 malloc과 calloc을 호출하기 때문에 매 테스트 수행마다 output이 다르다.

```
Activities Terminal 3월 31 13 : 42
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part3
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$ make run test4
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 1024 ) = 0x5636c5e042d0
[0003]         free( 0x5636c5e042d0 )
[0004]         free( 0x5636c5e042d0 )
[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.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$
```

part3 \$ make run test4

Illegal free와 double free가 발생하여도 ignore하도록 구현하였다. 이때 위의 두 잘못된 free는 freed_total에도 더해지지 않는다.

```
Activities Terminal 3월 31 13 : 42
yanghoon@yanghoon-VirtualBox: ~/linklab/handout/part3
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$ make run test5
cc -I. -I ../utils -o libmentrace.so -shared -fPIC memtrace.c ../utils/memlog.c
../utils/memlist.c -ldl
[0001] Memory tracer started.
[0002]         malloc( 10 ) = 0x56334d18d2d0
[0003]         realloc( 0x56334d18d2d0 , 100 ) = 0x56334d18d320
[0004]         realloc( 0x56334d18d320 , 1000 ) = 0x56334d18d3c0
[0005]         realloc( 0x56334d18d3c0 , 10000 ) = 0x56334d18d7e0
[0006]         realloc( 0x56334d18d7e0 , 100000 ) = 0x56334d18ff30
[0007]         free( 0x56334d18ff30 )
[0008]
[0009] Statistics
[0010] allocated_total      111110
[0011] allocated_avg        22222
[0012] freed_total          111110
[0013]
[0014] Memory tracer stopped.
yanghoon@yanghoon-VirtualBox:~/linklab/handout/part3$
```

part3 \$ make run test5

2. 구현 과정

모든 구현이 파트별로 나누어져 단계적으로 접근하여 구현할 수 있었다.

2.1. 스켈레톤 코드 분석

2.1.1. memtrace.c

memtrace.c는 내가 implement 해야 하는 주요 코드이다. 함수의 시작과 끝이 주어지고, 함수가 실행되는 동안 리스트를 생성해 memory allocation을 추적한다. 하지만 추적하는 기능은 미구현이므로 memory allocation을 library interpositioning하는 방법으로 구현해야한다.

2.1.2. memlist.c / memlist.h

memlist에서는 item이라는 struct를 이용해 memory allocation의 주소와 size, deallocation 여부를 저장한다. 이들은 첫 항목을 dummy로 하는 linked list로 저장된다. item과 관련한 함수들은 header 파일을 통해 그 기능을 확인할 수 있으며, 핵심 기능인 alloc과 dealloc은 각각 memory allocation과 memory deallocation을 list에 기록하는 역할을 한다.

```
//  
// add information about newly allocated block to list  
//  
//  
// list      pointer to list  
// ptr       pointer to newly allocated block  
// size      size of newly allocated block  
//  
// returns  
// item*     pointer to item holding information about the block  
//  
// the reference count is updated automatically. If the block is  
re-allocated  
// (i.e., there already is an item to ptr) the size of the item is  
adjusted.  
//  
item *alloc(item *list, void *ptr, size_t size);  
  
//  
// update information on freed block  
//  
// list      pointer to list  
// ptr       pointer to freed block  
//  
// returns  
// item*     pointer to item holding information about freed block  
//  
// the reference count is updated automatically  
//  
item *dealloc(item *list, void *ptr);
```

utils/memlist.h

2.1.3. memlog.c / memlog.h

memlist가 backend라면 memlog는 frontend의 역할을 하는데, memlog.h에는 여러 상황에 맞는 로그를 출력하는 매크로가 정의되어 있다. 이를 통해 앞서

memlist를 통해 확인된 정보를 memlog를 활용하여 출력하는 형식의 implementation을 수행할 수 있다.

2.2. Part1 구현

Part1에서는 실제 run-time library interpositioning을 구현하고 이를 통해 allocated memory를 계산하는 것이 main implementation이 되었다.

2.2.1. implement run-time library interpositioning

수업을 참고하여 dlsym(RTLD_NEXT, "interpositioning하고자 하는 함수")를 활용하였다. 이 함수는 이 라이브러리 다음부터 주어진 이름을 가지고 있는 함수의 주소를 찾아 반환한다. 따라서 wrapper 함수 속에서 실제 함수를 찾아갈 수 있다. 이때 에러가 발생할 수 있으므로 관련하여 종료시키는 알고리즘을 함께 활용한다.

```
if(!mallocp)
{
    mallocp = dlsym(RTLD_NEXT, "malloc");
    if((error = dlerror()) != NULL)
    {
        fputs(error, stderr);
        exit(1);
    }
}
```

part1/memtrace.c/void *malloc (size_t size)

이후 반환된 주소를 활용하여 원함수를 호출하는 것이다.

```
ret = mallocp(size);
```

2.2.2. log statistics

Part1에서는 총 allocated된 용량과 이를 횡수로 나눈 평균 메모리 용량을 계산한다. 따라서 free를 제외한 함수가 정상적으로 실행될 때마다 주어진 size를 n_allocb에 더하는 방법으로 구현하였다. size가 0으로 주어질 때 횡수의 증가 여부 등 corner case가 많았지만 TA 선생님께서 specify 해주신 덕분에 이를 크게 걱정하지 않고 구현할 수 있었다.

```
LOG_STATISTICS(n_allocb, n_allocb / (n_malloc + n_calloc + n_realloc), 0L);
```

part1/memtrace.c/void fini(void)

2.3. Part2 구현

Part2에서는 memory deallocation과 관련한 tracing이 main implementation이다.

2.3.1. initialize and keep list for tracing memory allocation/deallocation

memory tracing을 수행하기 위해 list는 allocation된 memory block을 기록해두었다가 deallocation될 때 이를 cnt 변수에 반영하여 저장한다. list에서 아예 삭제하는 방법도 있을 수 있겠지만, 이후 part3에서 double free issue를 처리하기 위해서 cnt 변수를 활용한다.

```
i->cnt = 1;
```

utils/memlist.c/item *alloc(item *list, void *ptr, size_t size)

이같은 연산은 memlist의 alloc과 dealloc 함수에서 실행되므로 각 wrapper 함수에서 alloc과 dealloc을 적절히 호출하는 implementation이 필요하다. 이때 alloc과 dealloc이 각각 방금 할당 혹은 해제된 memory block의 item 주소를 반환한다는 점을 고려하여 n_allocb와 n_freeb를 함께 계산할 수 있도록

수정하였다. 다만 **reallocation** 과정에서 재할당된 메모리 크기만큼 **deallocate**하지 않도록 **n_freeb**를 먼저 계산해야 한다.

```
n_freeb += dealloc(list, ptr) -> size;
n_allocb += alloc(list, ret, size) -> size;
```

part2/memtrace.c/void *realloc(void *ptr, size_t size)

2.3.2. calculate freed bytes

part1에서 구현하지 않았던 **n_freeb** 계산을 한다. **free**와 **realloc**은 실행시 **memory block**을 **deallocation**하기 때문에 두 **wrapper** 함수에서는 **n_freeb**를 계산한다.

2.3.3. log deallocated memory blocks

적절히 **deallocation**되지 않은 **memory block**을 출력한다. **deallocate**되었다면 **cnt = 0**이므로 **cnt**가 0이 아닌 **memory block**을 로그로 내보낸다.

```
if(n_allocb > n_freeb)
{
    LOG_NONFREED_START();
    item *i = list -> next;
    while(i != NULL)
    {
        if(i -> cnt)
        {
            LOG_BLOCK(i -> ptr, i -> size, i -> cnt);
        }
        i = i -> next;
    }
}
```

part2/memtrace.c/void fini(void)

2.3.4. deal with realloc corner case

realloc의 경우, 기존 **memory block**의 **size**보다 작은 **size**로 **realloc**하게 되면, 해당 **instruction**을 수행하지 않도록 구현한다. 따라서 **realloc**을 수행할 때에는 **list**를 이용해 기존 **memory block**의 **size**를 검색한 뒤 적절한 조치를 한다.

```
if(find(list, ptr) -> size >= size) // Deal with case: new size <=
old size
{
    ret = ptr;
    LOG_REALLOC(ptr, size, ret);
    return ret;
}
```

part2/memtrace.c/void *realloc(void *ptr, size_t size)

2.4. Part3 구현

Part3에서는 **illegal free**와 **double free**에 대해 **exception** 처리를 한다. 즉, 프로세스를 **quit**하지 않고 해당 **instruction**을 **ignore**한 뒤 프로세스를 이어나가도록 하는 것이다. **illegal free**와 **double free**는 **free**에서 뿐 아니라 **realloc**에서도 발생할 수 있음에 주의해야 한다.

2.4.1. illegal/double free in free

free에서 **illegal free**와 **double free**를 **detect**한 경우, **log**만 출력한 뒤 종료한다.

```
if(find(list, ptr) == NULL) // deal with case : ILLEGAL FREE
```

```

{
    LOG_FREE(ptr);
    LOG_ILLEGAL_FREE();
    return;
}
if(find(list, ptr) -> cnt == 0) // deal with case : DOUBLE FREE
{
    LOG_FREE(ptr);
    LOG_DOUBLE_FREE();
    return;
}

```

part3/memtrace.c/void free(void *ptr)

2.4.2. illegal/double free in realloc

realloc에서 illegal free와 double free가 발생했을 때에는, deallocate만 불가능할 뿐 allocate는 수행되어야 한다. 따라서 allocation 관련한 instruction은 모두 포함시킨다. 다만, 이를 조금 더 세련되게 구현하지 못한 부분은 아쉽다. 다른 함수를 활용하거나 config 역할을 하는 변수를 사용하는 방식으로 개선할 수 있을 것 같다.

```

if(find(list, ptr) == NULL) // deal with case : ILLEGAL REALLOC
{
    ret = reallocp(NULL, size);
    LOG_REALLOC(ptr, size, ret);
    LOG_ILLEGAL_FREE();
    n_realloc++;
    n_allocb += alloc(list, ret, size) -> size;
    return ret;
}
if(find(list, ptr) -> cnt == 0) // deal with case : DOUBLE REALLOC
{
    ret = reallocp(NULL, size);
    LOG_REALLOC(ptr, size, ret);
    LOG_DOUBLE_FREE();
    n_realloc++;
    n_allocb += alloc(list, ret, size) -> size;
    return ret;
}

```

part3/memtrace.c/void *realloc(void *ptr, size_t size)

2.5. 테스트

- 2.5.1. 기존 memory block보다 작은 용량으로 realloc하는 경우
- 2.5.2. illegal/double free가 발생했을 때 statistics 반영

3. 어려웠던 점

3.1. Realloc corner case 구현

Realloc에 있어 다양한 corner case가 존재할 수 있다는 것을 확인하였는데, 각각 다른 기능을 수행하도록 해야하는 상황에서 이를 세련된 코드로 구현하는 것이 하나의 난점이었다. 처음에는 if else 구문을 사용하였으나, 보다 좋은 방법이 있을 것이라고 생각한다. spec 기준이 완화되어 결과적으로 이를 구현하지 않았지만, 다음에는 config 역할을 하는 변수를 추가하여 하나의 instruction이 여러 모습으로 수행될 수 있도록 하면 더 좋을 것 같다.

3.2. 스켈레톤 코드 분석

주어진 스켈레톤 코드를 분석하는 것 역시 어려웠다. 스켈레톤 코드가 여러 개의 **c** 파일과 헤더 파일로 나누어져 있어서 이들 사이의 관련성을 찾는 데에 노력을 기울였다. 처음에는 아날로그하지만 모든 코드를 프린트해서 읽어보았다. 만약 코드가 더 방대한 분량이었다면 이렇게 하기는 어려웠을 것이다.

3.3. Realloc의 allocation과 deallocation 사이의 관계

Realloc은 **memory block**의 크기를 변경하는 **instruction**이지만, 이를 **deallocation** 후 **allocation**이라고 해석할 수도 있다. 하지만 초기 구현 단계에서 이에 대해 충분히 고려하지 못해 **allocation** 후 **deallocation**을 수행하도록 하였다. 그 결과로 **n_freeb**가 실제보다 크게 계산되는 오류가 발생하였다.

4. 새로 배운 점

4.1. Run-time library interpositioning(dlsym)

Run-time에서 **library interpositioning**을 구현하는 과제였기 때문에 이에 대해 여러 생각을 할 수 있었다. 특히 인상 깊었던 것은 **illegal free**와 **double free**에 있어 오류를 발생시키지 않고 **ignore**하는 방향의 **implementation**이었다. 이를 활용하여 특정 상황에서 오류를 발생시키는 함수를 활용할 때에도 반드시 **process**를 종료하지 않을 수 있다는 것을 배웠다. 이전에는 새로운 함수를 처음부터 구현해야 한다고 생각했는데, 유사한 **library**와 **library interposition**을 활용하여 보다 효율적으로 구현하는 방법도 있다는 것을 알게 되었다.

4.2. UNIX 환경에서의 코딩

computer architecture를 수강하면서 **unix**를 처음 접해보았는데, 그때 **unix**에 흥미를 느껴 노트북에도 **ubuntu**를 설치하여 사용 중이었다. 따라서 이번 기회에 **unix** 환경에서의 개발에 익숙해지려고 노력하였다. **vi editor** 명령어를 검색해 컴퓨터 앞에 붙여놓고 활용하였다. 아직 **vi editor**가 더 편하다고 하긴 어렵지만, 이전처럼 당황하지 않고 사용할 수 있었다. 또 개발 초기 단계에서는 **git**을 사용하여 과제이지만 내 프로젝트를 진행하는 것처럼 연습할 수 있었다.