Lab1 Linklab

2018-11727
응용생물화학부 김민기

Part 1 : Tracing Dynamic Memory Allocation

- 구현 방법

Part 1에서 구현해야 할 것은, malloc, calloc, realloc, free 함수가 실행될 때 allocation size & address, deallocation address, total & average allocated size 를 기록하는 것이다. 이를 위해서는 Library Interpositioning 을 진행해야 하는데, 세 가지 방법(Compilte time, Link time, Load/Run time)이 있지만 과제 스펙 상 요구하는 것은 Load-Time 에 interpositioning 을 하는 것이므로 해당 방법으로 각각의 함수를 interpositioning 했다. 이는 dlsym 함수와 dlerror 함수를 적절히 이용하면 구현해낼 수 있는데, 과제 자료의 예시를 참고했다.

malloc 함수를 예시로 좀 더 자세히 설명해보자면, dlsym 함수의 경우 첫 번째 인자로는 shared object handle을 받고 두 번째 인자로는 해당 shared object 에서 찾고자 하는 함수명 또는 변수명을 받아 해당 심볼과 연관된 주소값을 반환한다. 이에, 먼저 만들어둔 static function pointer mallocp 에다가 현재 있는 라이브러리가 아닌 다음에 오는 (RTLD_NEXT) 라이브러리(standard library)에서 malloc 함수를 찾아서 주소를 매핑 시켜주어, 이후부터는 mallocp로 호출해서 사용할 수 있게끔 해주었다. 이 후, malloc 을 포함한 calloc, realloc, free 함수들도 같은 방식으로 호출한 다음, 반환받는 정보들을 이용해 LOG 를 남기고, statistic 에 반영될 정보들을 업데이트 시켜주는 방식으로 구현했다.

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

  if (!mallocp) {
    mallocp = dlsym(RTLD_NEXT, "malloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }
  ptr = mallocp(size);
  LOG_MALLOC(size, ptr);
  n_malloc++;
  n_allocb += size;
  return ptr;
}
```

```c
void *calloc(size_t nmemb, size_t size)
{
  char *error;
  void *ptr;

  if (!callocp) {
    callocp = dlsym(RTLD_NEXT, "calloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }
  ptr = callocp(nmemb, size);
  LOG_CALLOC(nmemb, size, ptr);
  n_calloc++;
  n_allocb += nmemb * size;
  return ptr;
}
```

```c
void *realloc(void *ptr, size_t size)
{
  char *error;
  if (!reallocp) {
    reallocp = dlsym(RTLD_NEXT, "realloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }

  void *res = ptr;

  res = reallocp(ptr, size);
  LOG_REALLOC(ptr, size, res);
  n_realloc++;
  n_allocb += size;

  return res;
}
```

```c
void free(void *ptr)
{
  char *error;

  if (!freep) {
    freep = dlsym(RTLD_NEXT, "free");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }

  if (ptr != NULL) {
    LOG_FREE(ptr);
    freep(ptr);
  }
}
```

```c
__attribute__((destructor))
void fini(void)
{
  // ...
  unsigned long n_alloc = n_malloc + n_calloc + n_realloc;
  unsigned long avg_allocb = 0;
  if (n_alloc)
    avg_allocb = n_allocb / n_alloc;

  LOG_STATISTICS(n_allocb, avg_allocb, n_freeb);

  LOG_STOP();

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

- 어려웠던 점

우선, 스켈레톤 코드 각각이 의미하는 것을 이해하는데 시간이 오래걸렸고, 과제의 예시를 응용해서 직접 interpositioning 이 되게끔 하는 것에 어려움을 겪었다. 여러 번의 시행착오를 겪고 나서 malloc 을 interpositioning 시킨 다음에는, 각 함수들의 정확한 역할을 파악하지 못한 상태여서 각각의 기능을 이해하는데 많은 시간을 소요했다. man 을 통해서 각 함수의 동작 방식과 세부적인 정보를 이해했고, Etl 의 질의응답 게시글과 Q&A 세션 등을 통해서 각 함수들의 corner case 에서의 작동 방식을 이해하는데 많은 도움을 받을 수 있었다. 추가적으로 어려웠던 점은 Ubuntu 환경 안에 기본적으로 있는 vi 편집기의 세팅이 너무 생소해서 기존 코드 파악 및 코드 작성에 시간이 너무 많이 소요됐다. 결국, vim 을 설치하고 특정 설정을 해줌으로써 코드 가독성과 생산성을 높일 수 있었다.

- 새롭게 배웠던 점

malloc 을 비롯한 각 메모리 동적할당에 관여하는 함수들의 동작 방식에 대해 잘 이해할 수 있었다. 또한, 동적 라이브러리 로드에 사용되는 dlsym, dlerror 등의 함수들에 대해서도 좀 더 자세히 이해할 수 있었고, 각 함수들을 Interpositioning 을 시키는 방법을 숙지할 수 있었다.

Part 2 : Tracing Unfreed Memory

- 구현 방법

Part 2에서 구현해야 할 것은, 프로그램 실행 중 할당된 블록 중 free 되지 않은 블록과 총 free 된 크기를 기록하는 것이다. 이를 위해서는 스켈레톤 코드의 list function 들을 잘 이해했어야 했다. alloc 함수과 dealloc 함수 등의 작동 방식과 list 구조를 이해한 뒤, 블록이 할당될 때 alloc 을 해주고, 할당된 블록이 free 될 때 dealloc 을 해주고 동시에 n_freeb 도 업데이트 해주어서 free 된 전체 크기를 기록해주었다. 또한, 과제 스펙을 보면, 메모리 블록 중 free 되지 않은 블록이 존재할 때만, 그 블록들의 정보를 출력해주어야 했기 때문에, fini 함수에서 n_allocb 와 n_freeb의 크기를 비교해서 출력 여부를 결정했다. 또, alloc 과 dealloc 의 메커니즘을 이해하고 나니, deallocated 되지 않은 block 들의 cnt 가 0보다 크다는 점을 이용해 while 문으로 list 를 돌면서 deallocated 되지 않은 블록의 정보를 출력했다.

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

  if (!mallocp) {
    mallocp = dlsym(RTLD_NEXT, "malloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }
  ptr = mallocp(size);
  LOG_MALLOC(size, ptr);
  alloc(list, ptr, size);
  n_malloc++;
  n_allocb += size;
  return ptr;
}
void *calloc(size_t nmemb, size_t size)
{
  char *error;
  void *ptr;

  if (!callocp) {
    callocp = dlsym(RTLD_NEXT, "calloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }
  ptr = callocp(nmemb, size);
  LOG_CALLOC(nmemb, size, ptr);
  alloc(list, ptr, nmemb*size);
  n_calloc++;
  n_allocb += nmemb * size;
  return ptr;
}
```

```
void *realloc(void *ptr, size_t size)
{
  char *error;
  if (!reallocp) {
    reallocp = dlsym(RTLD_NEXT, "realloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }

  item *old_block = find(list, ptr);
  void *res = ptr;

  res = reallocp(ptr, size);
  LOG_REALLOC(ptr, size, res);
  if (old_block->size < size) {
    dealloc(list, ptr);
    n_freeb += old_block->size;
    alloc(list, res, size);
    n_realloc++;
    n_allocb += size;
  }

  return res;
}
```

```
void free(void *ptr)
{
  char *error;

  if (!freep) {
    freep = dlsym(RTLD_NEXT, "free");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }

  if (ptr != NULL) {
    LOG_FREE(ptr);
    freep(ptr);
  }
}
```

```c
__attribute__((destructor))
void fini(void)
{
  // ...
  unsigned long n_alloc = n_malloc + n_calloc + n_realloc;
  unsigned long avg_allocb = 0;
  if (n_alloc)
    avg_allocb = n_allocb / n_alloc;

  LOG_STATISTICS(n_allocb, avg_allocb, n_freeb);

  if (n_allocb > n_freeb) {
    LOG_NONFREED_START();
    item* block = list->next;
    while (block != NULL) {
      if (block->cnt > 0)
        LOG_BLOCK(block->ptr, block->size, block->cnt);
      block = block->next;
    }
  }
  LOG_STOP();

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

- 어려웠던 점

가장 어려웠던 점은 특정 주소에 어느 사이즈의 메모리가 할당되어져 있는지를 알아내는 것이었다. malloc, calloc, realloc 을 할 때는 인자로 사이즈를 받았는데, free 의 경우는 주소만 넘겨받기 때문에, n_freeb에 알맞은 사이즈 크기를 추가하기 위해서는 deallocated 되는 block 의 사이즈를 알아냈어야 했는데, 리스트의 구조나 함수들의 동작 방식이 잘 이해가 안된 상태에서 시행착오를 많이 겪었다. 구글링을 통해서 _msize 함수를 사용하는 방법을 알아냈는데, 선언되어 있지 않은 malloc.h 헤더를 추가적으로 include 해줘야 했기 때문에 고민을 하다가, memlist 파일을 조금 더 살펴보다 보니, find 함수나 dealloc 함수를 통해서 해당 블록의 사이즈를 알아낼 수 있음을 확인했다.

- 새롭게 배웠던 점

시행착오를 겪는 과정에, 이번 과제처럼 리스트 구조가 잡혀있지 않은 경우에는 malloc.h 헤더를 include 하고 _msize 함수를 통해서 동적할당된 메모리의 크기를 알아낼 수 있다는 사실을 배웠다.

Part 3 : Detect and Ignore Illegal Deallocations

- 구현 방법

Part 3에서 구현해야 할 것은, 이미 free 된 주소를 다시 free 하려고 시도하는 Double free 와 메모리가 할당된 적이 없는 주소를 free하려고 하는 Illegal free가 발생하는 상황을 체크해서 에러를 발생시키지 않고, 로그를 남겨주는 것이다. 이를 위해서는 우선적으로 각각의 에러가 어떤 상황에서 발생하는지를 자세히 알아야 했다.

Double free 나 Illegal free 둘 다, dealloc 이 일어나는 free 함수와 realloc 함수 모두에서 발생할 수 있다. 우선, 넘겨받은 주소와 find 함수를 이용해서 할당된 블록 구조체의 정보를 받아온다. 이 때, 블록이 NULL 인 경우, list 에 해당 주소의 블록이 없는 것으로,

프로그램 실행 중 메모리가 할당된 적이 없는 블록인 것으로 Illegal free에 해당한다. 만약 블록이 있는 경우, 블록의 cnt 수를 조회해서 0이면 이미 dealloc 된 경우로 double free에 해당한다. 각각의 상황에 맞게 로그를 남기고, statistic 을 업데이트 해주었다.

```
void *realloc(void *ptr, size_t size)
{
  char *error;
  if (!reallocp) {
    reallocp = dlsym(RTLD_NEXT, "realloc");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }

  item *old_block = find(list, ptr);
  void *res = ptr;

  if (old_block != NULL) {
    if (old_block->cnt) {
      res = reallocp(ptr, size);
      LOG_REALLOC(ptr, size, res);
      if (old_block->size < size) {
        dealloc(list, ptr);
        n_freeb += old_block->size;
        alloc(list, res, size);
        n_realloc++;
        n_allocb += size;
      }
    } else {
      // DOUBLE FREE
      res = reallocp(NULL, size);
      alloc(list, res, size);
      LOG_REALLOC(ptr, size, res);
      LOG_DOUBLE_FREE();
      n_realloc++;
      n_allocb += size;
    }
  } else {
    // ILLEGAL FREE
    res = reallocp(NULL, size);
    alloc(list, res, size);
    LOG_REALLOC(ptr, size, res);
    LOG_ILL_FREE();
    n_realloc++;
    n_allocb += size;
  }

  return res;
}
```

```
void free(void *ptr)
{
  char *error;

  if (!freep) {
    freep = dlsym(RTLD_NEXT, "free");
    if ((error = dlerror()) != NULL) {
      fputs(error, stderr);
      exit(1);
    }
  }

  if (ptr != NULL) {
    LOG_FREE(ptr);
    item *target_block = find(list, ptr);
    if (target_block != NULL) {
      if (target_block->cnt) {
        freep(ptr);
        item *freed_item = dealloc(list, ptr);
        n_freeb += freed_item->size;
      } else {
        LOG_DOUBLE_FREE();
      }
    } else {
      LOG_ILL_FREE();
    }
  }
```

- 어려웠던 점

  Double free 의 경우, 명확하게 상황이 이해가 갔지만, Illegal free 의 경우 개념이 헷갈려서

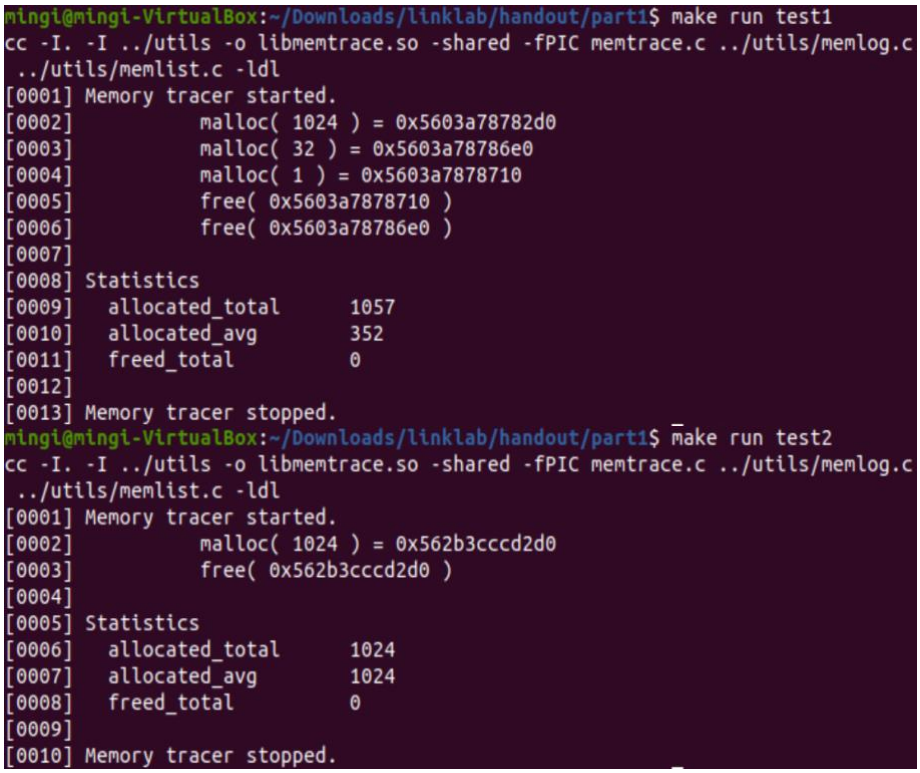
etl 질의응답을 통해서 개념을 확실히 이해한 뒤에야 구현할 수 있었다. 또한 corner case 라고 생각했던 부분들이 컴파일러의 최적화 과정에서 아예 함수 호출이 무시되거나 realloc 이 malloc 으로 호출되는 식으로 변화되어서 많이 헷갈렸다.

- 새롭게 배웠던 점

  free 함수와 realloc 함수들이 corner case 의 경우 어떻게 작동되는지를 배웠다. (예 : realloc(NULL, size)로 함수가 호출되면 자동으로 malloc(size)로 처리된다는 것, free(NULL)의 경우 컴파일러에 의해 실행이 되지 않는 다는 점 등)

- 실행 결과

Part1



```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/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 , 33990 ) = 0x55c4411302d0
[0003]           calloc( 1 , 9865 ) = 0x55c4411387a0
[0004]           malloc( 28128 ) = 0x55c44113ae40
[0005]           calloc( 1 , 25547 ) = 0x55c441141c30
[0006]           malloc( 12837 ) = 0x55c441148010
[0007]           malloc( 61088 ) = 0x55c44114b240
[0008]           calloc( 1 , 54610 ) = 0x55c44115a0f0
[0009]           malloc( 57638 ) = 0x55c441167650
[0010]           malloc( 16589 ) = 0x55c441175780
[0011]           calloc( 1 , 5206 ) = 0x55c441179860
[0012]           free( 0x55c441179860 )
[0013]           free( 0x55c441175780 )
[0014]           free( 0x55c441167650 )
[0015]           free( 0x55c44115a0f0 )
[0016]           free( 0x55c44114b240 )
[0017]           free( 0x55c441148010 )
[0018]           free( 0x55c441141c30 )
[0019]           free( 0x55c44113ae40 )
[0020]           free( 0x55c4411387a0 )
[0021]           free( 0x55c4411302d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      305498
[0025]   allocated_avg        30549
[0026]   freed_total          0
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/part1$ 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 ) = 0x555ed88692d0
[0003]           free( 0x555ed88692d0 )
[0004]           free( 0x555ed88692d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/part1$ 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 ) = 0x558a776492d0
[0003]           realloc( 0x558a776492d0 , 100 ) = 0x558a776492d0
[0004]           realloc( 0x558a776492d0 , 1000 ) = 0x558a776492d0
[0005]           realloc( 0x558a776492d0 , 10000 ) = 0x558a776492d0
[0006]           realloc( 0x558a776492d0 , 100000 ) = 0x558a776492d0
[0007]           free( 0x558a776492d0 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          0
[0013]
[0014] Memory tracer stopped.
```

Part2

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/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 ) = 0x556c5090a2d0
[0003]           malloc( 32 ) = 0x556c5090a710
[0004]           malloc( 1 ) = 0x556c5090a770
[0005]           free( 0x556c5090a770 )
[0006]           free( 0x556c5090a710 )
[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]   0x556c5090a2d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/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 ) = 0x560d2beee2d0
[0003]           free( 0x560d2beee2d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/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 , 12222 ) = 0x561577bcf2d0
[0003]           calloc( 1 , 41374 ) = 0x561577bd22d0
[0004]           malloc( 20550 ) = 0x561577bdc4b0
[0005]           malloc( 46901 ) = 0x561577be1530
[0006]           malloc( 9651 ) = 0x561577becca0
[0007]           calloc( 1 , 2827 ) = 0x561577bef290
[0008]           malloc( 58348 ) = 0x561577befde0
[0009]           malloc( 11741 ) = 0x561577bfe210
[0010]           malloc( 28770 ) = 0x561577c01030
[0011]           calloc( 1 , 64219 ) = 0x561577c080d0
[0012]           free( 0x561577c080d0 )
[0013]           free( 0x561577c01030 )
[0014]           free( 0x561577bfe210 )
[0015]           free( 0x561577befde0 )
[0016]           free( 0x561577bef290 )
[0017]           free( 0x561577becca0 )
[0018]           free( 0x561577be1530 )
[0019]           free( 0x561577bdc4b0 )
[0020]           free( 0x561577bd22d0 )
[0021]           free( 0x561577bcf2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      296603
[0025]   allocated_avg        29660
[0026]   freed_total          296603
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/part2$ 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 ) = 0x55618ba0f2d0
[0003]           free( 0x55618ba0f2d0 )
[0004]           free( 0x55618ba0f2d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/part2$ 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 ) = 0x563e22e632d0
[0003]           realloc( 0x563e22e632d0 , 100 ) = 0x563e22e63320
[0004]           realloc( 0x563e22e63320 , 1000 ) = 0x563e22e633c0
[0005]           realloc( 0x563e22e633c0 , 10000 ) = 0x563e22e637e0
[0006]           realloc( 0x563e22e637e0 , 100000 ) = 0x563e22e65f30
[0007]           free( 0x563e22e65f30 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```

Part3

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/part3$ 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 ) = 0x564c7b3a82d0
[0003]           malloc( 32 ) = 0x564c7b3a8710
[0004]           malloc( 1 ) = 0x564c7b3a8770
[0005]           free( 0x564c7b3a8770 )
[0006]           free( 0x564c7b3a8710 )
[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]   0x564c7b3a82d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/handout/part3$ 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 ) = 0x55d2a89fc2d0
[0003]           free( 0x55d2a89fc2d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/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]           calloc( 1 , 8405 ) = 0x55661ef8c2d0
[0003]           calloc( 1 , 7184 ) = 0x55661ef8e3e0
[0004]           calloc( 1 , 45939 ) = 0x55661ef90030
[0005]           calloc( 1 , 3422 ) = 0x55661ef9b3e0
[0006]           calloc( 1 , 42196 ) = 0x55661ef9c180
[0007]           malloc( 52079 ) = 0x55661efa6690
[0008]           malloc( 40276 ) = 0x55661efb3240
[0009]           malloc( 27286 ) = 0x55661efbcfd0
[0010]           calloc( 1 , 16728 ) = 0x55661efc3aa0
[0011]           malloc( 50066 ) = 0x55661efc7c30
[0012]           free( 0x55661efc7c30 )
[0013]           free( 0x55661efc3aa0 )
[0014]           free( 0x55661efbcfd0 )
[0015]           free( 0x55661efb3240 )
[0016]           free( 0x55661efa6690 )
[0017]           free( 0x55661ef9c180 )
[0018]           free( 0x55661ef9b3e0 )
[    Help        free( 0x55661ef90030 )
[                free( 0x55661ef8e3e0 )
[0021]           free( 0x55661ef8c2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      293581
[0025]   allocated_avg        29358
[0026]   freed_total          293581
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/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 ) = 0x55740d3fd2d0
[0003]           free( 0x55740d3fd2d0 )
[0004]           free( 0x55740d3fd2d0 )
[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.
```

```
mingi@mingi-VirtualBox:~/Downloads/linklab/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 ) = 0x56262c6b02d0
[0003]           realloc( 0x56262c6b02d0 , 100 ) = 0x56262c6b0320
[0004]           realloc( 0x56262c6b0320 , 1000 ) = 0x56262c6b03c0
[0005]           realloc( 0x56262c6b03c0 , 10000 ) = 0x56262c6b07e0
[0006]           realloc( 0x56262c6b07e0 , 100000 ) = 0x56262c6b2f30
[0007]           free( 0x56262c6b2f30 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```