시스템 프로그래밍 Lab1 Report

경영학과 2017-15108
박지상

1. 실행결과 : part1 – test1

```
2017-15108@sp2:~/HW1/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 ) = 0x5559bc29c2d0
[0003]           malloc( 32 ) = 0x5559bc29c6e0
[0004]           malloc( 1 ) = 0x5559bc29c710
[0005]           free( 0x5559bc29c710 )
[0006]           free( 0x5559bc29c6e0 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          0
[0012]
[0013] Memory tracer stopped.
```

1. 실행결과 : part1 – test2

```
[2017-15108@sp2:~/HW1/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 ) = 0x55db9ec5e2d0
[0003]           free( 0x55db9ec5e2d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          0
[0009]
[0010] Memory tracer stopped.
```

1. 실행결과 : part1 – test3

```
2017-15108@sp2:~/HW1/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 , 12371 ) = 0x563e58c6b2d0
[0003]          malloc( 36907 ) = 0x563e58c6e330
[0004]          calloc( 1 , 47648 ) = 0x563e58c77370
[0005]          malloc( 53982 ) = 0x563e58c82da0
[0006]          calloc( 1 , 49593 ) = 0x563e58c90090
[0007]          calloc( 1 , 41886 ) = 0x563e58c9c260
[0008]          calloc( 1 , 33699 ) = 0x563e58ca6610
[0009]          calloc( 1 , 31965 ) = 0x563e58cae9c0
[0010]          malloc( 22802 ) = 0x563e58cb66b0
[0011]          calloc( 1 , 4568 ) = 0x563e58cbbfd0
[0012]          free( 0x563e58cbbfd0 )
[0013]          free( 0x563e58cb66b0 )
[0014]          free( 0x563e58cae9c0 )
[0015]          free( 0x563e58ca6610 )
[0016]          free( 0x563e58c9c260 )
[0017]          free( 0x563e58c90090 )
[0018]          free( 0x563e58c82da0 )
[0019]          free( 0x563e58c77370 )
[0020]          free( 0x563e58c6e330 )
[0021]          free( 0x563e58c6b2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      335421
[0025]   allocated_avg        33542
[0026]   freed_total          0
[0027]
[0028] Memory tracer stopped.
```

1. 실행결과 : part1 – test4

```
2017-15108@sp2:~/HW1/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 ) = 0x55b8b3f012d0
[0003]           free( 0x55b8b3f012d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
```

1. 실행결과 : part1 – test5

```
2017-15108@sp2:~/HW1/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 ) = 0x564479ff02d0
[0003]           realloc( 0x564479ff02d0 , 100 ) = 0x564479ff02d0
[0004]           realloc( 0x564479ff02d0 , 1000 ) = 0x564479ff02d0
[0005]           realloc( 0x564479ff02d0 , 10000 ) = 0x564479ff02d0
[0006]           realloc( 0x564479ff02d0 , 100000 ) = 0x564479ff02d0
[0007]           free( 0x564479ff02d0 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          0
[0013]
[0014] Memory tracer stopped.
```

1. 실행결과 : part2 – test1

```
2017-15108@sp2:~/HW1/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 ) = 0x55aecc4422d0
[0003]           malloc( 32 ) = 0x55aecc442710
[0004]           malloc( 1 ) = 0x55aecc442770
[0005]           free( 0x55aecc442770 )
[0006]           free( 0x55aecc442710 )
[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]   0x55aecc4422d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

1. 실행결과 : part2 – test2

```
2017-15108@sp2:~/HW1/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 ) = 0x55d62a0662d0
[0003]           free( 0x55d62a0662d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

1. 실행결과 : part2 – test3

```
2017-15108@sp2:~/HW1/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 , 12583 ) = 0x55d406cab2d0
[0003]          calloc( 1 , 30056 ) = 0x55d406cae430
[0004]          calloc( 1 , 51669 ) = 0x55d406cb59d0
[0005]          calloc( 1 , 18110 ) = 0x55d406cc23e0
[0006]          malloc( 13206 ) = 0x55d406cc6ae0
[0007]          malloc( 47944 ) = 0x55d406cc9eb0
[0008]          calloc( 1 , 27506 ) = 0x55d406cd5a30
[0009]          calloc( 1 , 8537 ) = 0x55d406cdc5e0
[0010]          malloc( 33494 ) = 0x55d406cde780
[0011]          malloc( 18572 ) = 0x55d406ce6a90
[0012]          free( 0x55d406ce6a90 )
[0013]          free( 0x55d406cde780 )
[0014]          free( 0x55d406cdc5e0 )
[0015]          free( 0x55d406cd5a30 )
[0016]          free( 0x55d406cc9eb0 )
[0017]          free( 0x55d406cc6ae0 )
[0018]          free( 0x55d406cc23e0 )
[0019]          free( 0x55d406cb59d0 )
[0020]          free( 0x55d406cae430 )
[0021]          free( 0x55d406cab2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      261677
[0025]   allocated_avg        26167
[0026]   freed_total          261677
[0027]
[0028] Memory tracer stopped.
```

1. 실행결과 : part2 – test4

```
2017-15108@sp2:~/HW1/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 ) = 0x5565cf01d2d0
[0003]           free( 0x5565cf01d2d0 )
free(): double free detected in tcache 2
Aborted (core dumped)
make: *** [Makefile:37: run] Error 134
```

1. 실행결과 : part2 – test5

```
2017-15108@sp2:~/HW1/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 ) = 0x56112bc182d0
[0003]           realloc( 0x56112bc182d0 , 100 ) = 0x56112bc18320
[0004]           realloc( 0x56112bc18320 , 1000 ) = 0x56112bc183c0
[0005]           realloc( 0x56112bc183c0 , 10000 ) = 0x56112bc187e0
[0006]           realloc( 0x56112bc187e0 , 100000 ) = 0x56112bc1af30
[0007]           free( 0x56112bc1af30 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```

1. 실행결과 : part3 – test1

```
2017-15108@sp2:~/HW1/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 ) = 0x5601190782d0
[0003]          malloc( 32 ) = 0x560119078710
[0004]          malloc( 1 ) = 0x560119078770
[0005]          free( 0x560119078770 )
[0006]          free( 0x560119078710 )
[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]   0x5601190782d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

1. 실행결과 : part3 – test2

```
2017-15108@sp2:~/HW1/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 ) = 0x56099ca452d0
[0003]           free( 0x56099ca452d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

1. 실행결과 : part3 – test3

```
2017-15108@sp2:~/HW1/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 , 16350 ) = 0x555e356582d0
[0003]           malloc( 21302 ) = 0x555e3565c2f0
[0004]           calloc( 1 , 33875 ) = 0x555e35661660
[0005]           malloc( 22591 ) = 0x555e35669af0
[0006]           calloc( 1 , 36852 ) = 0x555e3566f370
[0007]           calloc( 1 , 46192 ) = 0x555e356783a0
[0008]           calloc( 1 , 34171 ) = 0x555e35683850
[0009]           malloc( 54725 ) = 0x555e3568be10
[0010]           malloc( 35659 ) = 0x555e35699410
[0011]           calloc( 1 , 63491 ) = 0x555e356a1fa0
[0012]           free( 0x555e356a1fa0 )
[0013]           free( 0x555e35699410 )
[0014]           free( 0x555e3568be10 )
[0015]           free( 0x555e35683850 )
[0016]           free( 0x555e356783a0 )
[0017]           free( 0x555e3566f370 )
[0018]           free( 0x555e35669af0 )
[0019]           free( 0x555e35661660 )
[0020]           free( 0x555e3565c2f0 )
[0021]           free( 0x555e356582d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      365208
[0025]   allocated_avg        36520
[0026]   freed_total          365208
[0027]
[0028] Memory tracer stopped.
```

1. 실행결과 : part3 – test4

```
2017-15108@sp2:~/HW1/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 ) = 0x556157d8e2d0
[0003]           free( 0x556157d8e2d0 )
[0004]           free( 0x556157d8e2d0 )
[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.
```

1. 실행결과 : part3 – test5

```
2017-15108@sp2:~/HW1/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 ) = 0x5618d98222d0
[0003]           realloc( 0x5618d98222d0 , 100 ) = 0x5618d9822320
[0004]           realloc( 0x5618d9822320 , 1000 ) = 0x5618d98223c0
[0005]           realloc( 0x5618d98223c0 , 10000 ) = 0x5618d98227e0
[0006]           realloc( 0x5618d98227e0 , 100000 ) = 0x5618d9824f30
[0007]           free( 0x5618d9824f30 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```

2. 구현 방법 : part1

```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);
  n_malloc += 1;
  n_allocb += size;
  LOG_MALLOC(size, ptr);
  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);
  n_calloc += 1;
  n_allocb += nmemb * size;
  LOG_CALLOC(nmemb,size, ptr);
  return ptr;
}
```

```c
void *realloc(void *ptr, size_t size){
  char *error;
  void *rptr;

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

  rptr = reallocp(ptr, size);
  n_realloc += 1;
  n_allocb += size;
  LOG_REALLOC(ptr, size, rptr);
  return rptr;
}
```

강의/실습 노트에 나와있는 방식대로 Load/Run-time interpositioning을 구현하였다.

2. 구현 방법 : part1

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

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

  freep(ptr);
  LOG_FREE(ptr);
}
```

```c
__attribute__((destructor))
void fini(void)
{
  unsigned long n_alloc = n_malloc + n_calloc + n_realloc;

  LOG_STATISTICS(n_allocb, (n_allocb/n_alloc), 0L);

  LOG_STOP();

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

강의/실습 노트에 나와있는 방식대로 Load/Run-time interpositioning을 구현하였다.

Part1에서 Freed_total은 0인 상태이므로 free()와 realloc()시 n_freeb 값을 변경시키지 않았다.

2. 구현 방법 : part2

```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);
  n_malloc += 1;

  item* i = alloc(list, ptr, size);
  n_allocb += i->size;

  LOG_MALLOC(size, ptr);

  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);
  n_calloc += 1;

  item* i = alloc(list, ptr, nmemb * size);
  n_allocb += i->size;

  LOG_CALLOC(nmemb, size, ptr);

  return ptr;
}
```

Part1의 방식에서 memlist.c의 alloc()을 활용하였다.

2. 구현 방법 : part2

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

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

  freep(ptr);

  item* i = dealloc(list, ptr);
  n_freeb += i->size;

  LOG_FREE(ptr);
}
```

dealloc된 사이즈만큼 n-freeb 값을 증가시킨다.

realloc() 은 기존 메모리 free 와 새 메모리 alloc 을 모두 수행한다고 가정하므로 dealloc/alloc이 같이 이루어졌다

```c
void *realloc(void *ptr, size_t size){
  char *error;
  void *rptr;

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

  rptr = reallocp(ptr, size);
  n_realloc += 1;

  item* i1 = dealloc(list, ptr);
  n_freeb += i1->size;
  item* i2 = alloc(list, rptr, size);
  n_allocb += i2->size;

  LOG_REALLOC(ptr, size, rptr);

  return rptr;
}
```

2. 구현 방법 : part2

```
//
// Self-made method
//
static void nonfreed_list(item* list);
__attribute__((destructor))
void fini(void)
{
  unsigned long n_alloc = n_malloc + n_calloc + n_realloc;

  LOG_STATISTICS(n_allocb, (n_allocb/n_alloc), n_freeb);

  nonfreed_list(list);

  LOG_STOP();

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

non-freed block을 trace하기 위해 별도의 메소드를 만들었다.
List의 block들을 순환하면서 cnt가 0이 아닌 block이 있다면 로그를 출력하였다.

```
void nonfreed_list(item* list){
  int num = 0;
  char *error;

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

  item *prev, *cur, *i;
  if(list == NULL) return;

  prev = list;
  cur = list->next;

  while(cur != NULL){
    if(cur->cnt > 0){
      if(num == 0) LOG_NONFREED_START();
      LOG_BLOCK(cur->ptr, cur->size, cur->cnt);
      num++;
    }
    prev = cur;
    cur = cur->next;
  }

  return;
}
```

2. 구현 방법 : part3

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

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

  item* i = find(list, ptr);

  LOG_FREE(ptr);

  if(i == NULL) LOG_ILL_FREE();
  else if(i->cnt == 0) LOG_DOUBLE_FREE();
  else {
    freep(ptr);
    item* i = dealloc(list, ptr);
    n_freeb += i->size;
  }
}
```

Illegal free와 double free를 탐지하기 위해 dealloc을 실행하기 전 list에서 주어진 ptr에 해당되는 block이 있는지 탐색하였다.

만약 block이 없다면 (i==NULL) illegal free를 출력한다.

만약 block이 있지만 cnt=0이라면 이미 free된 ptr이므로, double free를 출력한다.

위 두 가지 경우에 대해서는 추가적인 조치를 취하지 않고 바로 return함으로써 error를 invoke하지 않고 ignore 된다.

위의 두 가지에 해당되지 않는 정상적인 free에 대해서는 Part2에서 구현한 그대로 동작한다.

* realloc()에 대해서는 test case(test1-5)에 대해서만 만족하도록 구현하면 된다고 공지되었기에 별다른 추가적인 변경사항이 없음.

3. 어려웠던 점

- 과제 자체보다는 과제 개발 환경설정을 하는 부분이 어려웠다.
- M1 Mac을 사용하고 있는 관계로 UTM을 사용해야했으나, UTM의 속도가 매우 느렸다.
- Ubuntu와 SCP/SSH 프로토콜에 익숙하지 않아 이해하는데 시간이 오래걸렸다.
- UTM의 느린 속도 때문에 CLI 모드로 실행해야했고, VIM 에디터에 익숙하지 않은 만큼 코드를 작성하는데 어려움을 많이 겪었으나, 박신홍 학우가 vscode remote-ssh를 가르쳐줘서 mac 환경에서 상대적으로 편하게 코드를 작성할 수 있었다.
- C 프로그래밍에도 익숙하지 않아 make file의 역할을 이해하고 컴파일/실행하는 과정이 어려웠다.

4. 새롭게 배운 점

- Ubuntu의 동작 방식과 SSH/SCP 프로토콜에 대해 배울 수 있었다.
- C Makefile의 역할과 C 파일의 컴파일 / 실행 방법에 대해 배울 수 있었다.
- Load / Run-time Interpositioning을 구현하기 위해 함수를 후킹하는 방법에 대해 배울 수 있었다.
- illegal free / double free와 관련한 부분을 구현하면서, 포인터와 메모리에 대해 좀 더 구체적으로 이해했다.