System Programming Lab report
2021-14284 Taehyun Yang

1. Runtime Results

- Part 1.

Test 1.

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part1$ sudo 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 ) = 0x5588222242d0
[0003]          malloc( 32 ) = 0x5588222246e0
[0004]          malloc( 1 ) = 0x558822224710
[0005]          free( 0x558822224710 )
[0006]          free( 0x5588222246e0 )
[0007]
[0008] Statistics
[0009]   allocated_total      1057
[0010]   allocated_avg        352
[0011]   freed_total          0
[0012]
[0013] Memory tracer stopped.
```

Test 2.

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part1$ sudo 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 ) = 0x56087f8ac2d0
[0003]          free( 0x56087f8ac2d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          0
[0009]
[0010] Memory tracer stopped.
```

Test 3.

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part1$ sudo 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 , 50509 ) = 0x555fb7cbd2d0
[0003]          calloc( 1 , 32191 ) = 0x555fb7cc9830
[0004]          calloc( 1 , 26722 ) = 0x555fb7cd1600
[0005]          malloc( 41607 ) = 0x555fb7cd7e70
[0006]          calloc( 1 , 44248 ) = 0x555fb7ce2100
[0007]          calloc( 1 , 5712 ) = 0x555fb7cecde0
[0008]          calloc( 1 , 44379 ) = 0x555fb7cee440
[0009]          malloc( 48546 ) = 0x555fb7cf91b0
[0010]          malloc( 16382 ) = 0x555fb7d04f60
[0011]          calloc( 1 , 44499 ) = 0x555fb7d08f70
[0012]          free( 0x555fb7d08f70 )
[0013]          free( 0x555fb7d04f60 )
[0014]          free( 0x555fb7cf91b0 )
[0015]          free( 0x555fb7cee440 )
[0016]          free( 0x555fb7cecde0 )
[0017]          free( 0x555fb7ce2100 )
[0018]          free( 0x555fb7cd7e70 )
[0019]          free( 0x555fb7cd1600 )
[0020]          free( 0x555fb7cc9830 )
[0021]          free( 0x555fb7cbd2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      354795
[0025]   allocated_avg        35479
[0026]   freed_total          0
[0027]
[0028] Memory tracer stopped.
```

- Part 2.

Test1.

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part2$ sudo 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 ) = 0x559e4df402d0
[0003]           malloc( 32 ) = 0x559e4df40710
[0004]           malloc( 1 ) = 0x559e4df40770
[0005]           free( 0x559e4df40770 )
[0006]           free( 0x559e4df40710 )
[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]   0x559e4df402d0     1024       1
[0016]
[0017] Memory tracer stopped.
```

Test 2.

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part2$ sudo 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 ) = 0x559a347e92d0
[0003]           free( 0x559a347e92d0 )
[0004]
[0005] Statistics
[0006]   allocated_total      1024
[0007]   allocated_avg        1024
[0008]   freed_total          1024
[0009]
[0010] Memory tracer stopped.
```

Test 3.

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part2$ sudo 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 , 54791 ) = 0x55d2fd92e2d0
[0003]           calloc( 1 , 34632 ) = 0x55d2fd93b910
[0004]           malloc( 19826 ) = 0x55d2fd944090
[0005]           calloc( 1 , 58631 ) = 0x55d2fd948e40
[0006]           calloc( 1 , 15851 ) = 0x55d2fd957380
[0007]           malloc( 29469 ) = 0x55d2fd95b1b0
[0008]           calloc( 1 , 60091 ) = 0x55d2fd962510
[0009]           malloc( 36530 ) = 0x55d2fd971010
[0010]           malloc( 12527 ) = 0x55d2fd979f00
[0011]           malloc( 44738 ) = 0x55d2fd97d030
[0012]           free( 0x55d2fd97d030 )
[0013]           free( 0x55d2fd979f00 )
[0014]           free( 0x55d2fd971010 )
[0015]           free( 0x55d2fd962510 )
[0016]           free( 0x55d2fd95b1b0 )
[0017]           free( 0x55d2fd957380 )
[0018]           free( 0x55d2fd948e40 )
[0019]           free( 0x55d2fd944090 )
[0020]           free( 0x55d2fd93b910 )
[0021]           free( 0x55d2fd92e2d0 )
[0022]
[0023] Statistics
[0024]   allocated_total      367086
[0025]   allocated_avg        36708
[0026]   freed_total          367086
[0027]
[0028] Memory tracer stopped.
```

- Part 3

Test 4

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part3$ sudo 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 ) = 0x5581bb3992d0
[0003]           free( 0x5581bb3992d0 )
[0004]           free( 0x5581bb3992d0 )
[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.
```

Test 5

```
taehyun@taehyun:/media/HostShared/lab1_2021-14284/handout/part3$ sudo 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 ) = 0x562127eb52d0
[0003]           realloc( 0x562127eb52d0 , 100 ) = 0x7fdfb878d2e8
[0004]           realloc( 0x562127eb5320 , 1000 ) = 0x562127eb5320
[0005]           realloc( 0x562127eb53c0 , 10000 ) = 0x562127eb53c0
[0006]           realloc( 0x562127eb57e0 , 100000 ) = 0x562127eb57e0
[0007]           free( 0x562127eb7f30 )
[0008]
[0009] Statistics
[0010]   allocated_total      111110
[0011]   allocated_avg        22222
[0012]   freed_total          111110
[0013]
[0014] Memory tracer stopped.
```

2. Implementation by parts

- Part 1

```
void *malloc(size_t size){ //create new methods for each command
    char *error;
    void *ptr;

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

Load/Run-time Interpositioning

```
#ifdef RUNTIME
 /* Run-time interposition of malloc and free based on
  * dynamic linker's (ld-linux.so) LD_PRELOAD mechanism */
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

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

    /* get address of libc malloc */
    if (!mallocp) {
        mallocp = dlsym(RTLD_NEXT, "malloc");
        if ((error = dlerror()) != NULL) {
            fputs(error, stderr);
            exit(1);
        }
    }
    ptr = mallocp(size);
    fprintf(stderr, "malloc(%d) = %p\n", (int)size, ptr);
    return ptr;
}
```

mymalloc.c

By essentially translating the given Load/Run-time interpositioning code given from the lab files, I was able to create a malloc, calloc, realloc, and free method. With the given template, I simply replaced the fputs to mlog, replace the fprintf function to LOG_(Respective method) and adjust the proper variables to the given methods. For malloc, I simply incremented the number of malloc, n_malloc and then increased n_allocb by the size of the allocated memory.

```
void fini(void)
{
  // ...
   unsigned long average= n_allocb/(n_calloc+n_realloc+n_malloc);

   LOG_STATISTICS(n_allocb, average, 0L);

  LOG_STOP();

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

// ...
```

For the average calculation, I divide n_allocb to the times calloc, realloc, malloc has been called. Then I use the LOG_STATISTICS function to print out that n_allocb is the total allocated size and average. The Third parameter is meant for part 2.

- Part 2

```c
void free(void *ptr)
{
    char *error;
    unsigned long find_v=0;

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

    LOG_FREE(ptr);
    freep(ptr);
    find_v= find(list,ptr)-> size;
    n_freeb+=find_v;
    dealloc(list,ptr);
}
```

```c
void fini(void)
{
  // ...
   unsigned long average= n_allocb/(n_calloc+n_realloc+n_malloc);

   LOG_STATISTICS(n_allocb, average, n_freeb);

    item *linkedl = list->next;

    while(linkedl != NULL) {// iterate through linked list
        linkedl = linkedl->next;
    }

   LOG_STOP();

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

We use the methods we have used from part 1 to trace the number of non-freed block and total freed size. For us to do that, we add additional counting code on the free method and realloc method as it is the two methods that can potentially free memory. For example, in Free method, we use the linked list function find inorder to add the size of the list that is placed at ptr. Then we define that as the found value in find_v and add that on the n_freeb inorder to count all the freed size. In the fini function, we have a simple linked list iterating function that goes through the list that is created.

- Part 3

```c
void *realloc(void *ptr_n, size_t size)
{
    char *error;
    void *ptr;
    unsigned long find_v=0;
    item* foundl=find(list,ptr_n);

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

    LOG_REALLOC(ptr_n,size,ptr);
    n_realloc++;
    n_allocb+=size;



    if (foundl==NULL){
        LOG_ILL_FREE();
        ptr = reallocp(NULL,size);
    }
    else if (foundl->cnt==0){
        LOG_DOUBLE_FREE();
        ptr=reallocp(NULL,size);
    }

    else {
        ptr=reallocp(ptr_n,size);
        n_freeb+= foundl->size;
        dealloc(list,ptr_n);
    }


    alloc(list,ptr,size);

    return ptr;
}
```

In order to detect double-free or illegal free allocations of memory, we have to utilize the list used in part 2. If the address does not exist in the list, then the program would indicate an illegal free however, if the count is 0, then it would indicate a double free. The same logic is applied to Realloc. The challenge was in changing the iterative function in fini. I have implemented a flag so that when cnt!=0 for the first time, it would activate a non freed start and a log block. This is because we don't delete the released memory in the list inorder to identify a double free and illegal free.

```c
void free(void *ptr)
{
    char *error;
    unsigned long find_v=0;
    item* foundl=find(list,ptr);

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

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

- **Challenges**

Finding the algorithm for a proper allocation when we had to define the illegal frees and double free was a challenge for me. I had to have a better understanding of what the linked list actions were doing. I was able to figure it out by realizing that we don't have to delete the freed address into the list but rather make the value 0.

- **Things I have learned**

I struggled initially because this is the first time I am creating a program that relates to linking and memory allocation. I was able to learn more about the exact procedures that goes into memory alloc, realloc, and freeing the memory. I also learned when errors occur and how we would track the memories that are allocated or freed. Thankfully the initial code in the slide gave me a head start into knowing when to begin and what to do.