BLG312E Operating Systems Assignment 2

Abdullah Shamout
School of Computer Engineering
Istanbul Technical University
150200919
Email: shamout21@itu.edu.tr

***Abstract*—This report presents a modified version of the malloc() and free() calls, designed to manage the heap of a user-level process. The implementation involves requesting memory space from the system using the mmap() system call at the process startup, thus establishing a fixed-size heap. The memory allocator maintains a free-list to manage currently empty spaces within the heap. Incoming memory allocation requests are fulfilled by selecting the most suitable free spaces from this list. The implementation ensures efficient memory management while adhering to the specified constraints. This report discusses the design, implementation, and evaluation of the modified memory allocation functions.**

I. Introduction

Operating systems rely on efficient memory allocation and deallocation mechanisms to manage system resources effectively. One critical aspect of this is the management of the process heap. In this assignment, we were tasked with implementing a modified version of the malloc() and free() functions to manage the heap of a user-level process.

II. Implementation Details

A. InitMyMalloc()

The `InitMyMalloc()` function is responsible for initializing the heap. It is called when the program first starts to run and creates a fixed-size heap. To implement this function, we first round up the heap size to the nearest multiple of the page size using the `roundUpToPageSize()` function. Then, we use the `mmap()` system call to allocate memory for the heap. We initialize a free list to manage the empty spaces within the heap. The free list is implemented using a linked list data structure, with each node representing a free block of memory.

B. MyMalloc()

The `MyMalloc()` function replaces the standard `malloc()` function. It is used to allocate memory on the heap using different allocation strategies such as Best Fit, Worst Fit, First Fit, and Next Fit. To implement this function, we traverse the free list to find a suitable block of memory based on the specified allocation strategy. Once a suitable block is found, we allocate memory from that block and update the free list accordingly.

C. MyFree()

The `MyFree()` function replaces the standard `free()` function. It is used to release the memory allocated using `MyMalloc()`. To implement this function, we mark the allocated block as free and merge adjacent free blocks to prevent fragmentation. However it is probably more simplistic

D. DumpFreeList()

The `DumpFreeList()` function is for debugging purposes. It displays the available memory spaces in the heap, along with their addresses, sizes, and status (full or empty). To implement this function, we traverse the free list and print out the information for each block of memory.

III. Code Outputs

To test the implementation, we created four different processes (P1, P2, P3, P4), each requiring a specific size of memory to run. We used memory allocation strategies such as Best Fit, Worst Fit, First Fit, and Next Fit to place each process in the heap. The strategy type for memory allocation, the status of allocation for each process (succeeded or failed), and the addresses of allocated memory were printed. The output of the `DumpFreeList()` function was also displayed.

IV. Conclusion

In conclusion, the modified version of the `malloc()` and `free()` functions provides an efficient memory allocation mechanism for managing the heap of a user-level process. By using mmap() for dynamic memory allocation and implementing custom memory allocation and deallocation functions, we ensure efficient memory management while adhering to the specified constraints.

Answers to Questions

1. Main differences between malloc and mmap:

- **malloc**: It is a standard C library function used for dynamic memory allocation. It allocates memory from the heap, which is managed by the operating system's memory manager.
- **mmap**: It is a system call used to map files or devices into memory. It can also be used to allocate memory, but unlike `malloc`, it directly allocates memory from the operating system kernel and returns a pointer to that memory.

Decision between `malloc` and `mmap`:

- If I had to choose between `malloc` and `mmap`, I would consider the following factors:
  - `malloc` is more portable and easier to use since it is a standard C library function.

  - `mmap` provides more control over memory allocation and can be used to allocate large blocks of memory more efficiently.
  - If I need to allocate a large block of memory or need more control over memory allocation, I would choose `mmap`. Otherwise, I would use `malloc` for its simplicity and portability.

2. Method/call used in MyFree() instead of free():

In `MyFree()`, I used a custom memory deallocation method instead of `free()`. The custom method marks the memory block as free and merges adjacent free blocks to prevent fragmentation. I believe that the custom method used in `MyFree()` is as successful as `free()` at correctly and safely releasing the memory back because it follows similar principles. It marks the memory block as free and ensures that the memory is available for reuse. Additionally, it merges adjacent free blocks to prevent fragmentation, similar to what `free()` does.

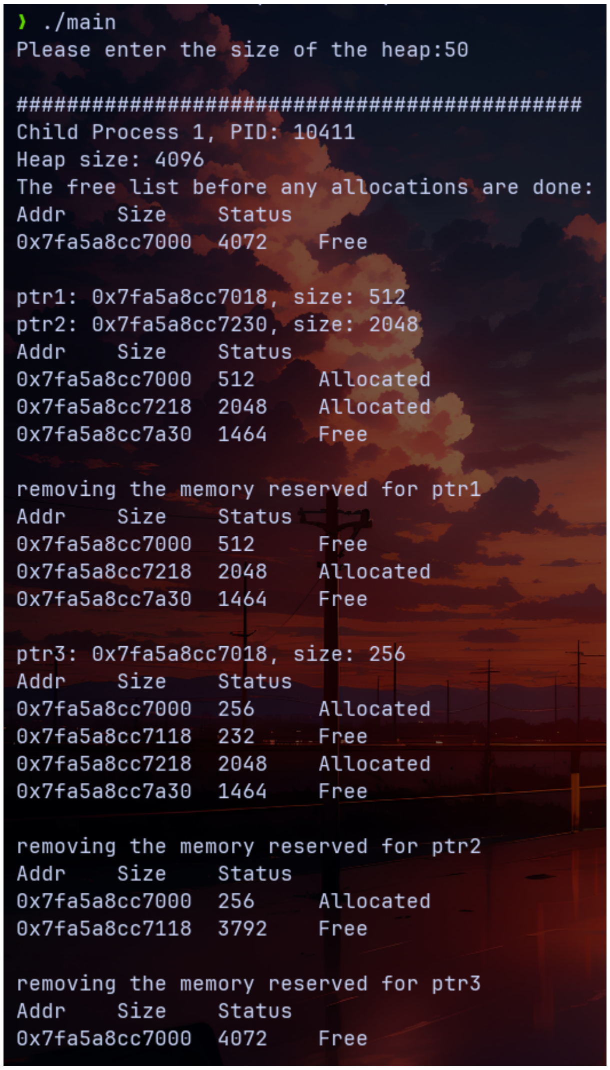
3. Outputs/screenshots of Linux command line:

- **Show the amount of free memory space before and after allocation:**
  Show the memory allocated processes (PID) and their memory addresses:
  Prove that when you release the allocated memory in your code using MyFree(), the memory is actually freed:

```
   ./main
Please enter the size of the heap:50

####################################
Child Process 1, PID: 10411
Heap size: 4096
The free list before any allocations
are done:
Addr    Size    Status
0x7fa5a8cc7000  4072    Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr    Size    Status
0x7fa5a8cc7000  512     Allocated
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr1
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

ptr3: 0x7fa5a8cc7018, size: 256
Addr    Size    Status
0x7fa5a8cc7000  256     Allocated
0x7fa5a8cc7118  232     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr2
Addr    Size    Status
0x7fa5a8cc7000  256     Allocated
0x7fa5a8cc7118  3792    Free

removing the memory reserved for ptr3
Addr    Size    Status
0x7fa5a8cc7000  4072    Free


####################################
Child Process 2, PID: 10413
Heap size: 4096
The free list before any allocations
are done:
Addr    Size    Status
0x7fa5a8cc7000  4072    Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr    Size    Status
0x7fa5a8cc7000  512     Allocated
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr1
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

ptr3: 0x7fa5a8cc7a48, size: 256
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  256     Allocated
0x7fa5a8cc7b48  1184    Free

removing the memory reserved for ptr2
Addr    Size    Status
0x7fa5a8cc7000  2584    Free
0x7fa5a8cc7a30  256     Allocated
0x7fa5a8cc7b48  1184    Free

removing the memory reserved for ptr3
Addr    Size    Status
0x7fa5a8cc7000  4072    Free


####################################
Child Process 3, PID: 10414
Heap size: 4096
The free list before any allocations
```

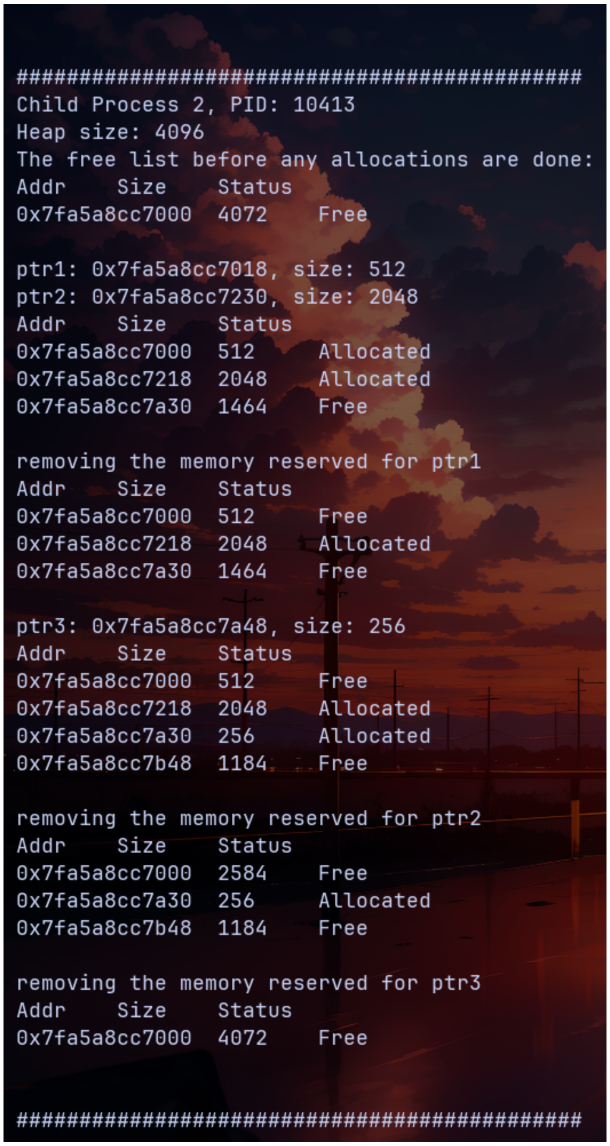
```
are done:
Addr     Size     Status
0x7fa5a8cc7000  4072     Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr     Size     Status
0x7fa5a8cc7000  512      Allocated
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

removing the memory reserved for ptr1
Addr     Size     Status
0x7fa5a8cc7000  512      Free
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

ptr3: 0x7fa5a8cc7018, size: 256
Addr     Size     Status
0x7fa5a8cc7000  256      Allocated
0x7fa5a8cc7118  232      Free
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

removing the memory reserved for ptr2
Addr     Size     Status
0x7fa5a8cc7000  256      Allocated
0x7fa5a8cc7118  3792     Free

removing the memory reserved for ptr3
Addr     Size     Status
0x7fa5a8cc7000  4072     Free


####################################
Child Process 4, PID: 10415
Heap size: 4096
The free list before any allocations
are done:
Addr     Size     Status
0x7fa5a8cc7000  4072     Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr     Size     Status
0x7fa5a8cc7000  512      Allocated
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

removing the memory reserved for ptr1
Addr     Size     Status
0x7fa5a8cc7000  512      Free
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

ptr3: 0x7fa5a8cc7018, size: 256
Addr     Size     Status
0x7fa5a8cc7000  256      Allocated
0x7fa5a8cc7118  232      Free
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

removing the memory reserved for ptr2
Addr     Size     Status
0x7fa5a8cc7000  256      Allocated
0x7fa5a8cc7118  3792     Free

removing the memory reserved for ptr3
Addr     Size     Status
0x7fa5a8cc7000  4072     Free
```

V. Screenshots



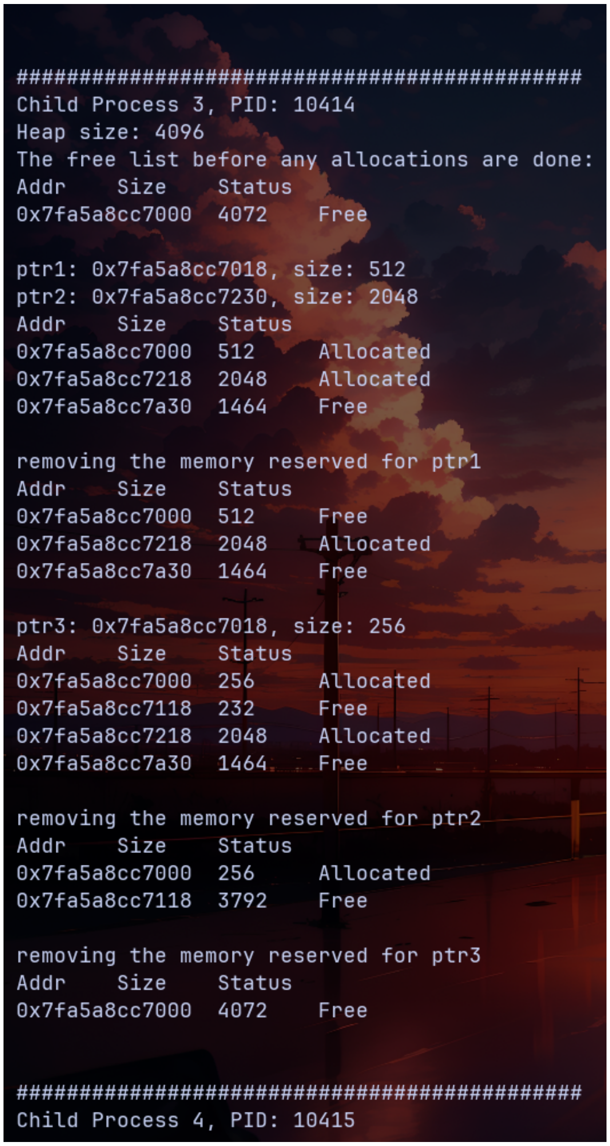
```
❯ ./main
Please enter the size of the heap:50

#############################################
Child Process 1, PID: 10411
Heap size: 4096
The free list before any allocations are done:
Addr     Size     Status
0x7fa5a8cc7000  4072     Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr     Size     Status
0x7fa5a8cc7000  512      Allocated
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

removing the memory reserved for ptr1
Addr     Size     Status
0x7fa5a8cc7000  512      Free
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

ptr3: 0x7fa5a8cc7018, size: 256
Addr     Size     Status
0x7fa5a8cc7000  256      Allocated
0x7fa5a8cc7118  232      Free
0x7fa5a8cc7218  2048     Allocated
0x7fa5a8cc7a30  1464     Free

removing the memory reserved for ptr2
Addr     Size     Status
0x7fa5a8cc7000  256      Allocated
0x7fa5a8cc7118  3792     Free

removing the memory reserved for ptr3
Addr     Size     Status
0x7fa5a8cc7000  4072     Free
```

Fig. 1. image 1



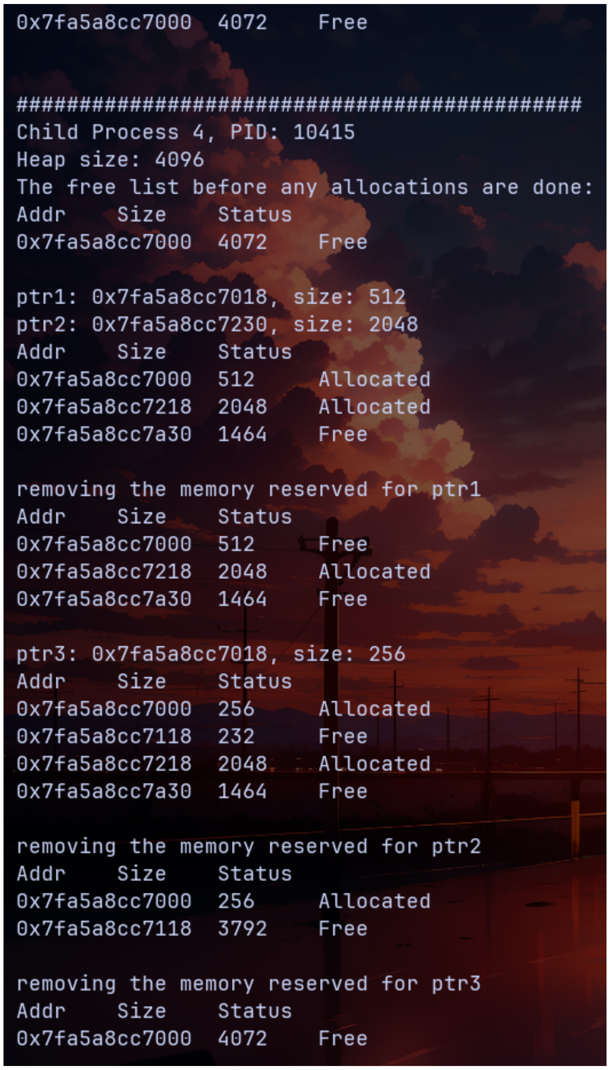
```
#############################################
Child Process 2, PID: 10413
Heap size: 4096
The free list before any allocations are done:
Addr    Size    Status
0x7fa5a8cc7000  4072    Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr    Size    Status
0x7fa5a8cc7000  512     Allocated
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr1
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

ptr3: 0x7fa5a8cc7a48, size: 256
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  256     Allocated
0x7fa5a8cc7b48  1184    Free

removing the memory reserved for ptr2
Addr    Size    Status
0x7fa5a8cc7000  2584    Free
0x7fa5a8cc7a30  256     Allocated
0x7fa5a8cc7b48  1184    Free

removing the memory reserved for ptr3
Addr    Size    Status
0x7fa5a8cc7000  4072    Free


#############################################
```

Fig. 2. image 2



```
#############################################
Child Process 3, PID: 10414
Heap size: 4096
The free list before any allocations are done:
Addr    Size    Status
0x7fa5a8cc7000  4072    Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr    Size    Status
0x7fa5a8cc7000  512     Allocated
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr1
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

ptr3: 0x7fa5a8cc7018, size: 256
Addr    Size    Status
0x7fa5a8cc7000  256     Allocated
0x7fa5a8cc7118  232     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr2
Addr    Size    Status
0x7fa5a8cc7000  256     Allocated
0x7fa5a8cc7118  3792    Free

removing the memory reserved for ptr3
Addr    Size    Status
0x7fa5a8cc7000  4072    Free


#############################################
Child Process 4, PID: 10415
```

Fig. 3. image 3



```
0x7fa5a8cc7000  4072    Free


#############################################
Child Process 4, PID: 10415
Heap size: 4096
The free list before any allocations are done:
Addr    Size    Status
0x7fa5a8cc7000  4072    Free

ptr1: 0x7fa5a8cc7018, size: 512
ptr2: 0x7fa5a8cc7230, size: 2048
Addr    Size    Status
0x7fa5a8cc7000  512     Allocated
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr1
Addr    Size    Status
0x7fa5a8cc7000  512     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

ptr3: 0x7fa5a8cc7018, size: 256
Addr    Size    Status
0x7fa5a8cc7000  256     Allocated
0x7fa5a8cc7118  232     Free
0x7fa5a8cc7218  2048    Allocated
0x7fa5a8cc7a30  1464    Free

removing the memory reserved for ptr2
Addr    Size    Status
0x7fa5a8cc7000  256     Allocated
0x7fa5a8cc7118  3792    Free

removing the memory reserved for ptr3
Addr    Size    Status
0x7fa5a8cc7000  4072    Free
```

Fig. 4. image 4

ACKNOWLEDGMENT

I would like to express my gratitude to my mom and dad.

REFERENCES

[1] My brain and fingers