CS518:OPERATING SYSTEMS

[A1 Umalloc – Project Report]

Anubhav Garikapadu **[NetID: AG2112]**
Vijay Swaminathan **[NetID: VS797]**

A. Prototyping:

Main Functions:

1) void * umalloc(size_t req_Size, char *file, int line)

- Checks whether the requested memory is available along with the additional metadata size.
- Scan entire memory until we reach a memory segment that is free and big enough to accommodate the block.
- Splits the memory into used and unused segments.

2) void ufree(void *addr_To_Free, char *file, int line)

- Checks if address to be freed is within memory range.
- Checks if memory has ever been allocated by malloc.
- Checks if memory is already free.
- Frees the blocks located at the pointer address provided accounting for the metadata and it returns an error if already freed.

Metadata Structure:

```
typedef struct meta_data {
    unsigned long seg_Size;
    char is_Free;
} meta_Node;
```

The meta_data structure consists of 2 variables:

- The seg_size variable holds the size of the memory block allocated
- Is_Free denotes whether the block is free

B. Memgrind Output:

0) **Consistency: (Fig 0)**

```
 1. Checking consistency
pointer 1 address -1645170607
pointer 2 address -1645170607
```

1) **Maximization: (Fig 1)**

```
 2. Finding Maximum Value
memgrind.c:30 Error:: Requested Size beyond memory limit.
memgrind.c:48 Error:: Memory is not full but there is not enough free memory for the allocation.
memgrind.c:48 Error:: Memory is not full but there is not enough free memory for the allocation.

 Max value:  9437184
```

2) **Basic Coalescence: (Fig 2)**

```
3. Basic Coalescence

H Mem Loc - -1645170607
Q Mem Loc - -1640451998
Max Allocated Memory Location - -1645170607
```

3) **Saturation: (Fig 3)**

```
 4. Saturation
memgrind.c:90 Error:: Memory is not full but there is not enough free memory for the allocation.

 Saturation Complete
```

4) **Time Overhead: (Fig 4)**

```
5. Time Overhead

Memory Location of last Allocation: -1634684887
Max time overhead: 0.000644
```

5) **Intermediate Coalescence: (Fig 5)**

```
6. Intermediate Coalescence

Final Full Memory Location - -1645170607
```

Error Handling:

- Since we have a direct view of all memory, we can determine when certain operations are good and bad ideas:
- On discovery of an improper use of malloc() or free(), you should output a diagnostic error message such as those outputted in Figures 1 and 3.

Instances where error should be raised:

Malloc:

- When there is a request to allocate more memory than is available in the user memory space.
- When there are no more free blocks to allocate.

Free:

- If the memory space to be freed exceeds the total memory size.
- If the memory address does not lie within the fixed memory range.
- If the memory hasn't been allocated by umalloc.
- The memory is already freed.