CSC 452/552 Operations Systems

Project 3 Threads

Name: Tanmai Kumar Ghosh

Bronco ID: 114224102

Date: 12/05/2024

# 1. Project Overview

The buddy memory allocation algorithm is a powerful memory management technique that efficiently manages contiguous memory blocks. The system described here is implemented following the method outlined by Dr. Donald Knuth in The Art of Computer Programming, Volume 1 - Fundamental Algorithms. This memory manager combines simplicity with efficiency, leveraging the buddy system to manage memory dynamically while minimizing fragmentation. It is ideal for systems where memory is allocated and deallocated frequently, such as operating systems or real-time applications.

# 2. Project Management Plan

## a) Task 3 – Implement Buddy:

This section implements a memory management system based on the buddy allocation algorithm. This system manages memory efficiently by dividing it into blocks of sizes that are powers of two. Here's a summary of the implementation:

Concepts:

1. Memory pool:
   a) Memory is allocated using **mmap().**
   b) Managed as a contiguous block, subdivided into smaller blocks as needed.

2. Buddy System:
   a) Blocks of size $2^k$ can be split into two equal "buddies" of size $2^{k-1}$.
   b) Buddies can merge back when both are free and of the same size.
3. Free Lists:
   a) A linked list for each block size $(2^k)$ stores available blocks.

b) Blocks are tagged as BLOCK_AVAIL (free) or BLOCK_RESERVED (in use).

### b) Task 4 – Write Unit tests:

There are existing three test cases. **test_buddy_malloc_one_byte** frees the block and checks that the pool is restored to its original state. **test_buddy_malloc_one_large** allocates a single block large enough to consume the entire memory pool. It validates that subsequent allocation attempts fail. It also frees the block and confirms the pool is restored. I added a new test method **test_buddy_realloc** which expands a block to a larger size and ensures memory is relocated. It also frees the reallocated block and verifies pool consistency.

## 3. Project Deliveries

### a) How to compile and use my code?

1. Clean the make build
   **make clean**
2. Build the project
   **make**
3. Run the project
   **./myprogram**
4. Run the test
   **./test-lab**

### b) Any self-modification?

I have not done any modification except the **src/lab.c** and **tests/test-lab.c** files.

### c) Summary of Results.

After executing **./myprogram**

```
@tanmaibsu → /workspaces/buddy_allocator (main) $ ./myprogram
hello world
@tanmaibsu → /workspaces/buddy_allocator (main) $
```

After executing **./test-lab**

```
@tanmaibsu → /workspaces/buddy_allocator (main) $ ./test-lab
Random seed:1733443463
Running memory tests.
->Testing buddy init
tests/test-lab.c:177:test_buddy_init:PASS
->Test allocating and freeing 1 byte
tests/test-lab.c:178:test_buddy_malloc_one_byte:PASS
->Testing size that will consume entire memory pool
tests/test-lab.c:179:test_buddy_malloc_one_large:PASS
-> Testing buddy realloc functionality
tests/test-lab.c:180:test_buddy_realloc:PASS

-----------------------
4 Tests 0 Failures 0 Ignored
OK
@tanmaibsu → /workspaces/buddy_allocator (main) $
```

## 4. Self-Reflection of Project

This project focused on implementing a memory pool management system using the buddy allocation algorithm, designed to efficiently allocate and deallocate memory blocks of varying sizes while minimizing fragmentation. The buddy system works by splitting memory blocks into powers of two and keeping track of free blocks in a free list. The system handles memory allocation (**buddy_malloc**), deallocation (**buddy_free**), and reallocation (**buddy_realloc**), ensuring that memory is used efficiently by splitting and merging blocks as needed. One of the challenges in the project was managing memory fragmentation. The buddy system addresses this by merging free blocks of the same size, reducing external fragmentation over time. Efficiently finding the smallest block that fits the requested size was another challenge, solved by maintaining separate free lists for each block size. Additionally, boundary conditions and memory rounding were handled with the **btok** function, which calculates the smallest k such that 2^k is greater than or equal to the requested size. This project helped deepen my understanding of low-level memory management techniques, including allocation, deallocation, and fragmentation. It also provided hands-on experience with linked lists and error handling in memory management. Despite the success of the implementation, there are areas for improvement, such as adding thread safety for concurrent environments and optimizing the split and merge operations. Additionally, expanding the system to allow dynamic resizing of the memory pool and integrating memory profiling tools would enhance the system's capabilities. Overall, the buddy allocation algorithm provides an efficient way to manage memory, balancing speed and low fragmentation. This project has reinforced the importance of efficient memory management in system design.

## 5. Comments for Project (optional)

Known Issue:
You can face the AddressSanitizer issue rarely. If you face that issue, please use **ctrl + c** to stop.

## 6. Use of AI for debugging (optional)