CS 342 Operating Systems

Spring 2021

Project Final Report

Zeynep Cankara

21703381

Section: 1

Instructor: İbrahim Körpeoğlu

Introduction

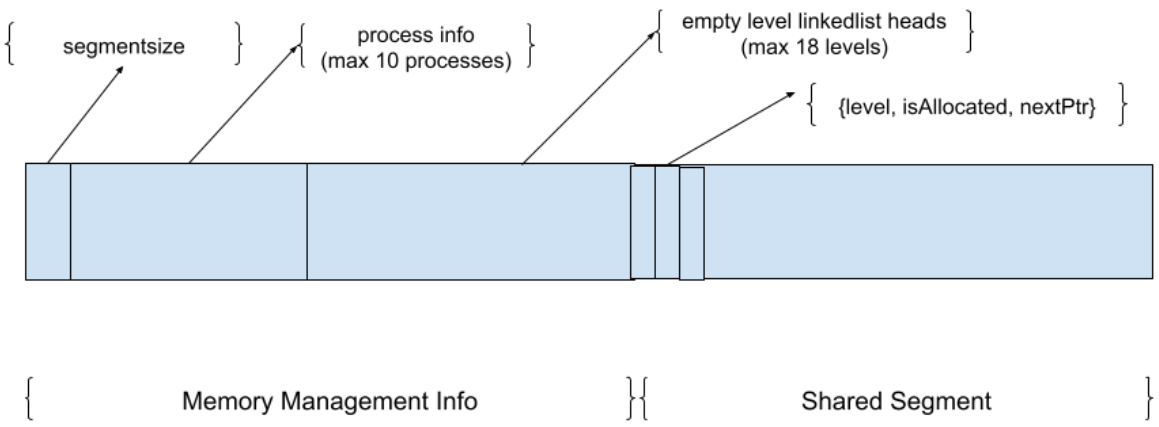
The project is about implementing a memory management library that makes use of the buddy algorithm [1,2]. The algorithm used in the project helps to allocate space in shared memory replacing the malloc function which allocates spaces dynamically in the common standard c libraries. The buddy algorithm requires all memory blocks to be power of two and tracks the empty memory blocks via a linked list data structure.

Project Description

The memory management methods suffer from the fragmentation; commonly divided into two groups internal and external fragmentation. Internal fragmentation happens when the memory allocates more memory than than the requested amount. Thus, the internal fragmentation can be measured by the difference between memory allocated and memory used. External fragmentation happens when non-contiguous free memory spaces exist in the memory. Therefore, even though there exists a total memory space to fulfill a memory allocation request the memory can't be allocated since the free spaces exist in non-contagious chunks.

Implementation Details

The shared memory model implements buddy algorithm to allocate memory

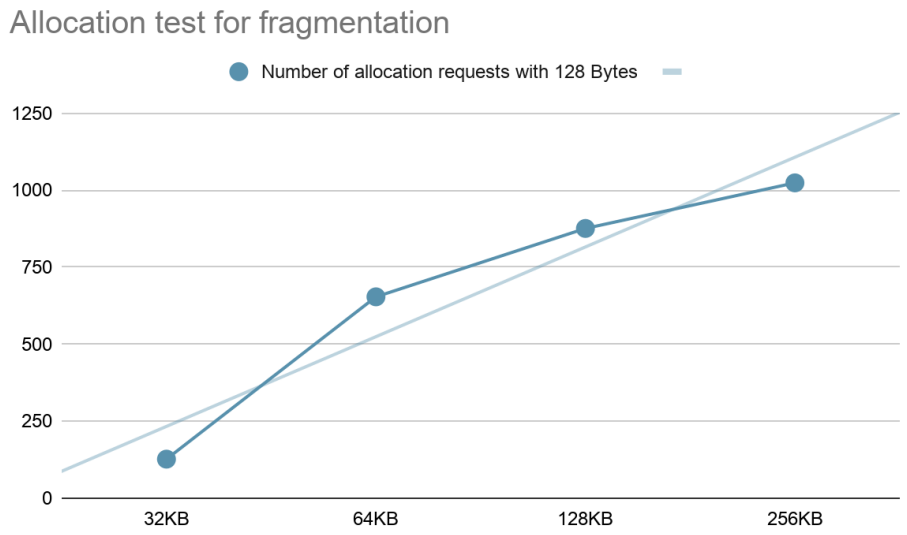


The memory management info available to all processes. The first slot contains the segment size which is written on the shared memory after the first *sbmem_init* call. The second slot contains the information about the processes. The third slot contains the linkedlist heads which helps to track the empty levels. Rest of the shared segment belongs to the user. Each allocated block contains information about the level of the block, whether the block is allocated or not and the next pointer connecting to the not allocated block belonging to the same level.

Experiment(s) and Results

Experiment 1)

We wanted to measure the external fragmentation. We initialised a shared memory with the segment size 128KB, ideally we should be able to do 1000 allocation requests with the 128 we accumulated over the number of allocation request we send out and the number of allocations we could do for different memory sizes obtained the following plot:



In the ideal scenario above we should always need to have a space to allocate if the total allocations do not exceed the shared segment size but in the scenario above we observed less allocations can be made. We hypothesize that since each block has an overhead of 16 bytes since we store 4 bytes integer level, 4 bytes integer isAllocated and 8 bytes void pointer next pointer information at the beginning of each allocated block which leads to higher allocated space than the user requested. We defined this phenomenon as internal fragmentation above. Also this leaves us with non-contiguous free space that the user won't use since we allocate the next power

of two then the user sends an allocation request which incurs additional external fragmentation.

Conclusion

This assignment helped us to understand how we can implement our own system program for handling the memory management. I believe after completing the project I can see the importance of how various memory management algorithms made trade offs in design in our case the way the algorithm handles external and internal fragmentation. Additionally, we learnt a lot about how to use named semaphores and shared memory in Linux which enhanced our knowledge about how multi-processor programs can share data without facing race conditions.

References

[1] The Art of Programming. Donald Knuth.

[2] Buddy Memory Allocation.
https://en.wikipedia.org/wiki/Buddy_memory_allocation