BILKENT UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

CS 342
OPERATING SYSTEMS

Project 3

Group Members:

Emre Açıkgöz	21801914	Section 3
Burak Yiğit Uslu	21801745	Section 2

22/04/2021

Emre Açıkgöz
Burak Yiğit Uslu

Introduction

This document contains brief explanations about our implementation of the buddy algorithm, as well as a section dedicated to the experiments and discussions regarding the internal fragmentation metrics of our implementation.

Our Algorithm

Firstly, to better explain our code and our experiments, it is important to explain our algorithm. Our algorithm for this question is inspired and is largely based on the buddy algorithm described in the “The Art Of Computer Programming” by Donald E. Knuth.

We have an overhead at the beginning of the shared memory segment. In this overhead we store things such as the number of processes currently using the library, the size of the segment and such. Also, we store an array called avail, which keeps track of available memory spaces. The avail array holds fundamentally a linked list of “block”s, which are overheads found on free memory blocks, and holds the level of the block, next and prev pointers and such. Once a request is made, either one of the free memory blocks at the desired level are allocated, or if none exists, a block from a higher level is broken down and allocated to the caller process. Once a memory block is allocated to a process, the “block” struct of the free memory block is replaced with the “overhead” struct, which holds only the “tag” showing that the memory block is allocated or not and a short int for storing the level of the block. Therefore, every memory block allocated to a caller process has an 8 byte overhead. Consequently, when a request is made for, for example 254 bytes, instead of 256 bytes, a 512 needs to be allocated in order to make room for the overhead. With that said, in general, storing this simple overhead in the beginning of each allocated memory block saves from making bigger/more costly allocations at the overhead for the shared memory segment. Because if we were to store all the information at the beginning of the shared memory segment, we would need to act according to the worst case, which would, in a lot of cases, result in a considerable amount of memory wasted.

Emre Açıkgöz
Burak Yiğit Uslu

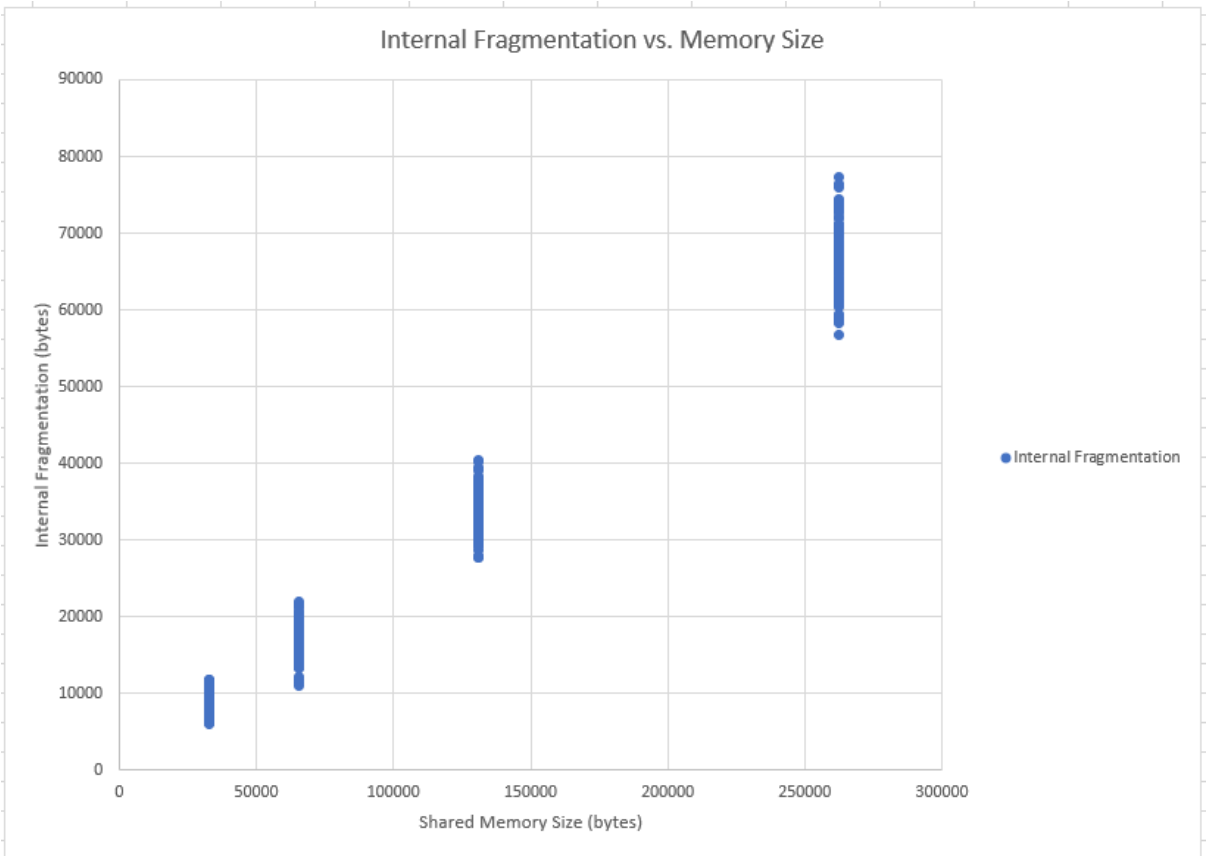
Experiments & Discussion

a. Experiments

Procedure (pseudo-code):

For each segment size of [32kb, 64kb, 128kb, 256kb]
    repeat 100 times
        initialize shared segment
        open library
        until segment is full:
            make allocation with request size, which is between 128 bytes
                and 4096 bytes and is uniformly distributed.
        collect internal fragmentation information.
        close library
        remove shared segment

b. Results and Discussion



The internal fragmentation in the memory highly depends on the total memory size. That is, since there is no external fragmentation thanks to the buddy algorithm, all of the fragmentation is internal and is amortized by the buddy algorithm. According to experiment results, approximately %30 of the memory is wasted.