Jason Woodworth Edward Woods

jww7675 ebw3559

Ian Callaway Scott Roddy

ipc0385 smr8991

**CMPS 455 - Stage 5: Dessert**

**Task 1: Lock**

A lock is a simpler semaphore in the sense that there are only two states for the integer value, off and on, thus there can only be one process acquiring the resources behind a lock.

**Task 2: File System**

We implemented changes to the file header by branching the file header’s behaviors based on the file size. The functions changed to alter its behavior were the Allocation, Deallocation, and ByteToSector functions. The Allocation and Deallocation functions operate by finding vacant sectors on the disk and assigning that sector to a pointer. Every sector of the file is allocated individually. For files able to be allocated strictly using direct pointers, every sector allocated for the file is assigned to a direct pointer in the file header. For files of a size larger than the amount of sectors available to the direct pointers of a file header, the last pointer becomes a single indirect pointer which is allocated to a sector, but the sector’s memory is filled with direct pointers to sectors allocated for the file. For files bigger than a single indirect pointer can allocate, an additional double indirect pointer is used to point to an allocated sector of pointers and those pointers behave as single indirect pointers. Allocation and deallocation mirror each other, and yet ByteToSector behaves differently. ByteToSector works by branching off which tier of pointer is being used. The lowest level using only direct pointers is only the byte offset divided by the sector size. The next level of single indirect pointers shaves off the direct pointer offset and then returns the offset divided by the amount of memory allocated for each direct pointer. The last level of double indirect pointer shaves off the direct and single indirect offsets and returns the offset divided by the memory allocated for a single indirect pointer.

**Task 3: Documentation**

Writing the documentation for NachOS did not further our understanding of NachOS because in order to successfully extend NachOS, these functions and classes must’ve been understood. Maybe documenting this would’ve provided insight before we had started the projects.

**Task 4: Report**

1. The problems that we encountered with this project were problems with structuring the code for Allocation and Deallocation and ByteToSector that made it intuitive to understanding what needed to be done. Allocation and ByteToSector functions had to be implemented in tandem for the project to successfully run and problems in NachOS were nearly identical and problems could not be easily identified. Completing Allocation’s double indirect pointer provided little insight to ByteToSector’s input as it isn’t mirrored like Deallocation.

2. No data structures were used in this project, but the iterative loop was used extensively. Speed and efficiency did not affect our choices but code legibility and familiar concepts did.