Rachel Lewis
5/6/2020
Diaz

CS 4310 Program 3 Report

Introduction

The purpose of this project was to create a simulated memory management unit that would allocate memory space to processes of varying sizes based off of three different memory management policies and three different fit algorithms. The unit was supposed to place processes in a queue if they couldn't be placed in memory or if they could, they were to fill a hole in the memory map based on the fit algorithm.

Implementation

For this program it was implemented using two different data structures and three different classes that represent different entities within the program's execution. The classes that were implemented were Process, frame, and hole. These classes held important information about various things that were needed for the program to run.

The purpose of the Process class was to represent information about one process that is introduced to memory. As the input files were read in, information was read from the files and used to instantiate a new Process object to represent an entity that holds all the information provided by the input files. It was these Process objects that were operated on throughout the program's execution. One of the data structures that was created and used to organize these processes was the processQueue. This queue was built to hold onto a process that couldn't be placed into memory and one it was moved to memory, it was removed from the processQueue.

Another class that was used was the frame class. This frame class was mainly used for the memory management policies of paging and segmentation and it wasn't used at all for VSP. The purpose of the frame class was to hold important information about a frame object that was instantiated and constructed using user input and information from the input files. For paging, the size of the frame was given by the user, for segmentation, the sizes of the segments/frames came from the files, and for VSP the size of one frame was always instantiated to 1. The total size of the process was however many frames it took up in memory for VSP. Each frame was assigned to a process that held that frame or 0 if the frame represented a hole

The frame objects that were created during the program were organized into a processFrameList data structure. The processFrameList was a data structure that was to represent a list of frames. This processFrameList is the most complex aspect of the project since it is this data structure that is called every time the memory unit needed to check if a process can fit in memory, physically add a process to memory, deallocate and terminate a process, and finally print the contents of the memory map.

The final class that was used for the project was the hole class. This class is quite simple and it's mainly used by the processFrameList to identify holes in the memory map, and check to see if there is a process in the processQueue that can fit in the hole. This class is instantiated every time a hole in memory is identified and the hole object only contains information about the starting index of where the hole was found and the ending index of the hole. An array of holes is maintained by the processFrameList and it is constantly referenced to see if a process can be placed in memory and where.