EI331 Course Project

By: Wang Haoxuan

Instructor: Wu ChenTao

December 30, 2018

I. PROJECT 7: CONTIGUOUS MEMORY ALLOCATION

A. Introduction

This project is aimed at designing algorithms for allocating contiguous memory. Memory allocationg includes requesting the memory, releasing part of it, and compacting the holes in memory. Algorithms for requesting memory includes first fit, best fit and worst fit. Releasing the memory indicates releasing the memory used by some process, and compacting indicates that holes would be organized into one big part. The process of running the program is shown below:

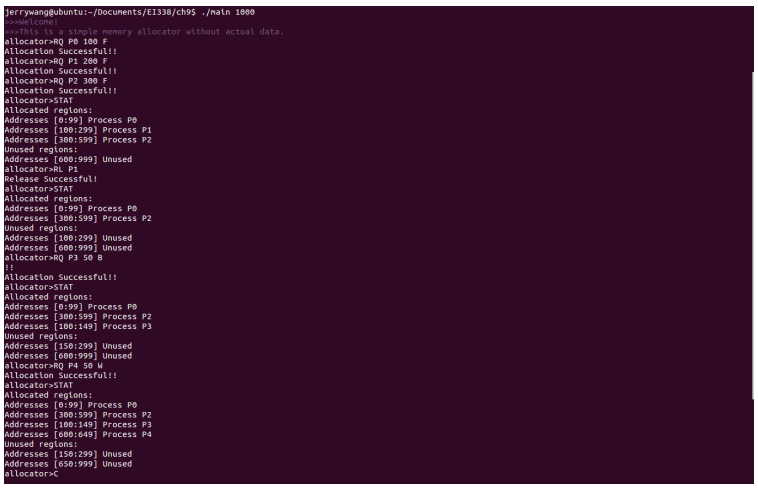

FIG. 1: Process of Running the Program(1)

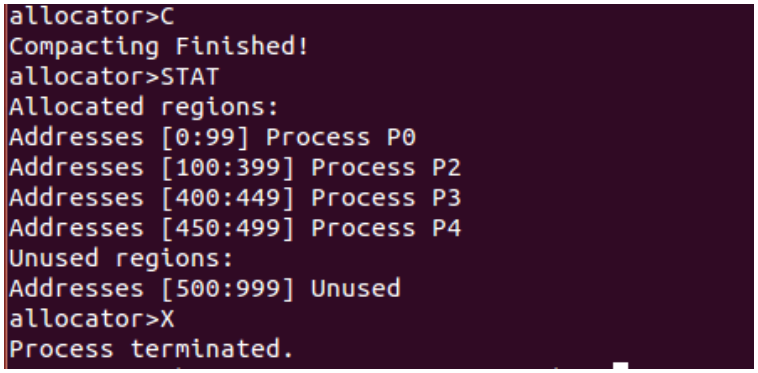

FIG. 2: Process of Running the Program(2)

We first allocate the whole memory size of 1000, then allocate 100 for P0 with first-fit method, 200 for P1 with first-fit, and 300 for P2 with first fit. We show the status, and then release P1, and show the status again. To demostrate that the algorithms are correctly implemented, we the allocate 50 for P3 with best-fit, and then allocate 50 for P4 with worst-fit. From the status shown, we can see that the memory is correctly allocated. At last, we compact the memory, and reach the result in the second screenshot.

B. Implementation Details

The implementation structure is quite simple. We specify the whole size of the memory in the shell, and run the program in a shell-like interface. Each command first specify the type of command: request, release, compact or showing status. Requesting and releasing the memory also needs to specify the name of the process. Requesting the memory would also need the specification of the space of memory needed and the method to be used. Status command would show the condition of the memory, and X would exit the program.

In the program, we need to keep track of all the processes and all the holes. Two structures of process and hole are constructed to save their ids, space needed as well as the start and end positions. The total number of processes and holes are also recorded as global variables. *allocator.c* included all the necessary functions for the program. What's more, a sort function is also implemented to sort the memory status, that is, we do not show the memory allocation in the order of the commands, but rather sort them by the starting position of the memory allocated. This makes the compacting process easier as well as making the user easier to manage the memory.

The last thing to mention is that the user has to be careful about the total space of memory allocated when calling the porgram, the PC's or virtual machine's actual memory might not be able to handle memory that large. All the code is shown in the *code* file.