Vahe Karamian
Operating Systems
Wednesday, August 22, 2001
Dr. Lee

Project III

The purpose of project III was to simulate an interactive program which manages main memory based on the paging system. The following are the system specifications: We had a total of 16MB of usable memory. CPU scheduling was implemented as a uniprocessor environment with Round Robin algorithm. Each process runs as soon as it is allocated with main memory. Arrival and termination is determined by user input.

The main memory management strategy is as following: We have a paging system with page size equal to 0.5MB. Initially the system is empty, frames are initialized as ### and relative page number (0-63). Upon arrival of each process, Main Memory Management (MMM) tries to allocate this process to pages. If there is enough space in the main memory, than we store them and update the tables. On the other hand if there is not enough room in the memory, the process has to wait until other processes are completed and enough number of pages are available. At the termination of a task, all relevant pages are release.

The following data structures have been developed for this project: A Task Table which contains the process id, the process size, the process state, the process page-map-base, and the calculated number of pages required. We have a Master Page Table which is an array of pointers to allocated Page Map Tables upon the arrival and memory entry of each process. Our Page Table is a structure which is allocated dynamically depending on the size of the required pages. The Page Table contains the process id as well as the frame number where it is stored at in the main memory. And finally we have our Physical Memory which is a fixed size of 32 with two fields of process is and page number which is in accordance with the Master Page Map Table.

The execution sequence was as follows: Initialize all the necessary data structures, and display a menu so the user starts a process, terminate a process, or quit the program.

The logic behind the simulation is as follows: First we initialize the necessary data structures, then we display a menu which the user has the options of starting a process, terminating a process, or ending the simulation. When the user select to start a process, we get the process id from the user with the process size, we store the information into the Task Table – and let Task Table calculate the number of required pages for this specific task and store it internally in the Task Table. Next we use the calculated value to determine if there are enough pages in the main memory to be allocated for this process. If so we allocate a Page Map Table and we go to the main memory and start storing the process into the frames and updating the PMT. At the end of the process, we update the Master PMT a pointer to the newly allocated PMT, and the Task Table with the Page-Map-Base. If there is not enough room in the memory, then we add the process to the waiting queue. We display the menu again, the user selects to Terminate a process, we get the process id which the user wishes to terminate and we get the Page-Map-Base from the Task Table and the number of pages which the process required, based on this information we know where to go

and how to de-allocate the memory occupied by the process. When this process is done, we check to see if there are any processes in the waiting queue, if so we go through them and see if we have enough frames in the main memory to use for the waiting process, if so we allocate them to the process and continue, if not then we don't and continue. When the user ends the simulation, we print out all of the tables and quit.

This project like most of my other projects has been developed under Win32 using Visual C++. The source code may be compiled under UNIX with some minor modifications, such as in the following example which is the only problem with this project:

```
For( int i=0; i<WHATEVER; i++ )
  DoSomething

For( i=0; i<WHATEVERl i++ )
  DoSOmething
```

This is allowed in the Visual C++ environment, however, in the UNIX environment we have to have separate variables declared for the two for loops. That is the only change which has to be made in the program in order for it to run under the UNIX environment.

There are only two files for this project:

taskTable.h
mmm.cpp