Paging Simulator Design Document

Author: Jacob Walton

Date: April 5, 2018

**Purpose:**

This is a program that is used to simulate the way an Operating System handles paging. It is used for educational purposes.

**Overview:**

User will load in a trace file that has a process ID, and either the size of text and data segments, or a halt command to remove the process from memory. If the process is to be stored in memory, it will be given free frames in memory and stored. Each process will have individual page tables, and the OS will keep a master frame table of all the frames in memory. If a process is to be removed from memory, its frames will be freed.

**Description of Flow/Data Structures:**

The program is initiated, the user must load a trace file. The user can than go to the next or previous trace line and view the graphical representation of memory. At any point, the user can load a different trace file and go through that tape.

Frames are assigned to a process from the first available on the free frame list. When a process is removed, all the frames it used are restored to the free frame list, and the list is sorted by frame index.

An Operating System object is created when a trace is loaded. The OS stores a list of frames currently in memory, a free frame list, and a process list of all processes currently in memory. The OS is the main data structure that stores all of the information about a trace file. At each change, the current state of the OS is stored in a list so that the trace can be reverted to the previous step easily. If a process is removed from memory, all of its frames in memory are cleared and added to the free frame list. A Page Table object is created for each process which creates both a text and data page table, while also adding all of process’s frames to the OS’s memory list.

An App object is created for the GUI part of the program. The App class extends the design2.py file created using QT Designer which is a drag and drop style GUI creation tool. App extends this file, so the GUI can be modified and the code I added is not overwritten each time. The App object stores some info about the current spot in the trace file and current state of process tables so that the “Back” button will work correctly.

The GUI represents the OS’s memory and the page table for each process. Each process is color coded for easy viewing. If either end of the trace tape is hit, the program does not crash, but waits for the tape to be moved through the other way, or a new trace to be load.

**Limitations/Assumptions:**

This program is limited to 8 free frames in memory, and that no trace tape will ask for more frames than are available. The trace file must be in the format specified in the assignment document.