CIS*3110 Operating Systems (Winter 2022)

Assignment #3: Memory Management

Specifications

We discussed four algorithms for placing the memory requirement of processes into **“holes”** in memory: **best** fit, **worst** fit, **next** fit and **first** fit. Your job is to write a simulator that will take processes of varying sizes, load them into memory according to each of these algorithms and swap processes out as needed to create larger holes.

Let us make the following assumptions:

1. Memory is of size 1024 MB.
2. A process will be some whole number of megabytes in the range 1 to 512 (inclusive).
3. An initial list of process sizes is loaded from a file. The name of this file is the first argument on the command line of your program. These processes should be loaded into a queue of processes waiting to be loaded into memory. The second command line argument of your program is the allocation strategy (algorithm) used: first, best, next, worst.
4. Memory is initially empty (we will ignore the space taken by the operating system).
5. If a process needs to be loaded but there is no hole large enough to accommodate it, then processes should be swapped out, one at a time, until there is a hole large enough to hold the process that needs to be loaded.
6. If a process needs to be swapped out, then the process that has been "in memory" the longest should be the one to go.
7. When a process has been swapped out, it goes to the end of the queue waiting to be swapped in (see next item).
8. Once a process has been swapped out for a third time, we assume that its process has run to completion and it is not re-queued. Note: not all processes will be swapped out for a third time.

You can use whatever data structure you wish to simulate memory usage, but please make sure that the comments in the source file clearly indicate what data structure you are using and how it is implemented. Also put this information into your README file.

The "*process file*" will be in the following format:

Process id (number) <space> memory chunk size (an integer)

Here is a sample process file:

```
1 130
2 99
3 200
4 512
```

5 440
6 320
7 200
8 90

Your program should be able to do the following:

1. Simulate 4 allocation strategies, namely, **first** fit, **best** fit, **next** fit and **worst** fit as specified on the command-line.
2. Each time a process is loaded into memory, a line should be printed such as:

```
pid loaded, #processes = 5, #holes = 3, %memusage = 41, cumulative %mem = 40
```

where %memusage refers to the percent of memory that is currently occupied by processes, and cumulative %mem gives the average of all the %memusages up until and including the current process load.

3. When the queue is empty, a line such as the following should be printed:

```
Total loads = 33, average #processes = 14.4, average #holes = 6.3, cumulative %mem = 62
```

This program should be called **holes**, and should take two command-line argument as described previously:

```
$ ./holes testfile first

$ ./holes testfile best

$ ./holes testfile next

$ ./holes testfile worst
```