Operating Systems
Lab-7: Implementing Contiguous Memory Allocation Schemes

Due 11:55PM, Dec 6

For this assignment, you will construct a simulation of a memory system with variable size partitions. You will use the simulation to collect empirical measurements of fragmentation under different placement policies (best fit, worst fit, first fit, and next fit).

The Simulation

Simulate memory policy on a hypothetical machine with a memory of 1024 KB (1MB). Jobs on this hypothetical machine will request a partition of a given size. Upon being assigned a partition of the requested size, they will run for some period of time, eventually terminate, and there upon release the memory in their assigned partition.
The simulation should create a loop in which the following steps are carried out repeatedly. Each time through the loop is referred to as a time step.

1. A new job is generated.
2. If a hole exists that is large enough to accommodate a partition of the requested size, then the partition is created and assigned to the new job.
3. Otherwise, one of the running jobs is chosen at random and removed from memory. (This represents the natural termination of some job after an undefined length of time.) If this creates enough space, then the new job is run; otherwise, additional randomly chosen jobs are terminated from memory until a sufficiently large hole has been created.
4. The degree of fragmentation (fraction of memory occupied by holes) is calculated and stored for later reference.

A simulation will normally pass through several phases. Initially, the entire memory is empty, and the first few jobs generated will fit entirely in memory, without any holes between them. This will probably be the largest number of jobs in memory during the entire simulation. Once the memory has been filled, it will become necessary to remove processes periodically before adding others. However, it may take some time before a state of equilibrium is reached, such that the average number of jobs in memory is steady over long periods of time. Therefore you should allow the simulation to run for a long time (say 1000 time steps) before starting to take any measurements.

To Do

Your job is to write a program that carries out the simulation described above, under a variety of experimental conditions. The first condition to vary is the average size of the job. Consider three different size distribution profiles as described below.
Profile 1 (Any sized jobs): Each job can be of any random size between 1 and 1024.

Profile 2 (Small jobs): Each job can be of any random size between 1 and 100.
Profile 3 (Large jobs): Each job can be of any random size between 500 and 1000.
The second condition to vary is the replacement policy. Again, your program should allow the user to choose among the three policies, best fit (Policy 1), worst fit (Policy 2) and first fit (Policy 3).

Input: Job profile (1, 2 or 3) and Replacement Policy (1, 2 or 3)
As an example *./a.out 1 3* should simulate memory management for any sized jobs using first fit.

Output: The following parameters, one in each line

Run a simulation for each policy, recording the following information over 1000 time steps (don't forget that you must run the simulation for 1000 steps before starting to record, making 2000 time steps in all):

- The average fragmentation (as a fraction of total memory)
- The average size of a hole
- The average number of holes that must be examined to satisfy a single new partition request
- The highest fragmentation measured in any time step (as a fraction of total memory)
- The lowest fragmentation measured in any time step (as a fraction of total memory)
- The average number of jobs in memory
- The highest number of jobs ever in memory
- The lowest number of jobs ever in memory
- The average size of a job
- The average number of holes
- The highest number of holes recorded
- The lowest number of holes recorded
- The largest number of partitions created in a row without any intervening evictions
- The largest number of evictions required in satisfying a single new partition request

When you've done the simulation under all nine conditions, you should assemble your results in a text file named summary.pdf. This should include tables that allow all nine conditions to be compared based upon each of the statistics described above. (Don't just give me the raw program output unless you have set the program up to produce such tabular data.) In addition, you should provide an analysis of your results. Do they confirm what we said about the different policies in class? How does the average job size affect the results, if at all?

The care you take with your summary will make up a significant portion of your grade, so please don't stint on it. The analysis should comprise at least a handful of well-written paragraphs addressing all of the above questions. It should summarize all that you have learned from the simulation, and anything else of interest that you may note.

Additional Investigations (For brownie points)

If you're interested in pursuing the simulation farther, there are interesting questions that can be asked about non-equilibrium behavior. How long does the system take to reach equilibrium at the start? (Probably the number is much less than 1000 time steps.) If you implement memory compaction (disturbing equilibrium), how many time steps does it take before equilibrium is again reached? These are the questions that would have to be asked by someone designing a system that seeks to avoid fragmentation through periodic memory compaction.

To Submit

You should submit source code (C/C++ file) for your program, readme.txt, and a summary of the results.
Place your files in a folder named as your complete roll number (capital case). Your C file should also be named as [your_roll_number].c. Zip the folder. The zip file should also be named as [your_roll_number].

Note: Zip the folder. Do not zip the files directly.

Upload the zip file.