University of Saskatchewan
Lab 6: Implement a Memory Allocation Simulator using Monitors

Out: October 28th, 2024

Due: At the end of the tutorial (actually by 9:00 p.m. October 31st, 2024, final deadline).

Marks: 5

You **may work in groups of at most 2 students. Each student must supply their own solution**.

Description

In this lab, you will be implementing the best fit algorithm to determine where to load a process into memory. You will use your monitor implementation from either Lab 4 or Assignment 2 to simulate threads accessing and free-ing memory. Your "application" should consist of a number of threads that allocate and free memory. Each thread will randomly choose a memory size to allocate (in a contiguous block - thus simulating a process arrival), sleep for a while and then randomly select whether or not to free() one of its existing blocks of memory that has been previously allocated. It does this for a pre-determined number of times. Parameters for the simulation that can be set include the following: maxSleepTime, maxAllocation, freeProbability, numberOfIterations. Feel free to choose more, but these will be necessary at least.

If a block of memory cannot be allocated, because there is no such space available, then the thread will have to wait until memory is made available by another thread free-ing one of its blocks. Thus, it calls MonWait(memAvail). When a thread frees memory, it should signal that condition variable. Thus, there are 2 monitor procedures: BF-Allocate(size) and Free(address), much like StartRead() and StopRead() in Assignment 2. The memory space is a global data structure shared amongst all threads and should be implemented as a list of free spaces in memory (start address, size). When adjacent blocks become free, they should be coalesced into one block. You may hard-code (using a #define) the size of the entire memory space.

A naive implementation will be subject to potential deadlock. Attempts to provide a deadlock free implementation will be granted bonus credit. The best way to avoid deadlock in this situation is to choose parameters that still leave enough memory available for the largest allocation possible.n

The application threads will have to use lists to keep track of which memory blocks they have allocated so that they can go through the list to determine which block they will free during the next iteration.

It may be useful to include a design document to help the marker follow your solution strategy.

Deliverables and Grading

A single tar file named `lab6.tar` containing:

- `Makefile` (1 mark)
- `BestFit.c` (1 mark) with the implementation of the threads.
- `BestFitMonitor.c` (1 mark) with the implementation of the monitor procedures BF-Allocate and Free.
- `output.txt` (1 mark) log of the output of the simulator.
- `gitlog.txt` (1 mark)
- Any other necessary `source code` for compilation. List files and monitor files. No .a files to be submitted. (0 marks)

GENERAL: Ensure that your name, NSID, and student number are at the beginning of every file that you hand in. **If a lab is group work, all partners' information is required.**

If you do not have a working monitor solution, borrow from another group. Monitor.c and Monitor.h only.