Dianne Lopez
Kailie Chang
Heriberto Juache
CPSC 351-01
12/02/19

Design of Sleeping Teaching Assistant

Modularization

5 Modules:

- MM.cpp: sets up global variables (including structs), prompts user for memory size, page size, name of the file to be read. Additionally, prints the average turnaround time per process.
- MM.h: creates a class called "MM" that gathers the various functions from all files.
- process.h: creates a struct of "Processes" and its data (id, arrival time, lifetime, etc.).
- queue.h: creates an input queue for the processes to be inserted, checks the processes index in the queue, and displays the input queue.
- Memory.h: creates two structs ("Frame"/"FrameList") that work together to create a frame list, place process into memory, and print the frame list.

Concurrency

In this application, the main file (MM.cpp) is executing in parallel with the header files. When the main file is executed, it essentially starts the execution of the other files (i.e. they won't run without the main file running). The header files set up what is to be done in the application (i.e. the process.h only sets up a struct called Process if the main file is run first). Doing this, concurrency of the modules allows the main file and the header files to be executed parallel to each other, simulating the execution of the processes as well as decisions of Memory Manager (MM). This simulation generates an output file with the trace of important events, as well as the memory map and the input queue status after each event.

Coupling and Cohesion

The cohesion between the main file and the header files is very high since they all need to work together for the application to work properly. All header files serve different functions but essentially they only execute once the main file is run. This application offers various degrees of cohesion. For instance, since elements of modules are grouped together and executed sequentially and work on the same data (information), this is called Communicational Cohesion. Additionally, since the elements of modules are grouped together and executed sequentially to perform a task, this is also Procedural Cohesion.

The coupling between the main file and header files is very high. This means that they have a strong relationship with one another and are dependent on each other. Since the multiple modules in the application share a common data structure and work on different parts of it, this is called Stamp Coupling. Additionally, since multiple modules have read and write access to the same global data, this is also considered Common Coupling.

Design Verification

Process (P):

ProcessID
Arrival Time
Lifetime
Address Space

Example:

Input Queue (empty)

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Time = 0:

Process 1 Arrives:
Input Queue:

P1							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Process 2 Arrives:

Input Queue:

P1	P2						
0	1	2	3	4	5	6	7

MM moves P1 → Memory

Memory

P1							
0	1	2	3	4	5	6	7

Input Queue

P2							
0	1	2	3	4	5	6	7

Processing in memory…

MM moves P2 → Memory

Memory

P1	P2						
0	1	2	3	4	5	6	7

Input Queue

0	1	2	3	4	5	6	7

Processing in memory…

Time = 100;

Process 3 Arrives:
Input Queue:

P3							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Process 4 Arrives:
Input Queue:

P3	**P4**						
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

MM moves P3 → Memory

Memory

P1	**P2**	**P3**					
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Input Queue

P4							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Processing in memory…

MM moves P4 → Memory

Memory

P1	**P2**	**P3**	**P4**				
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Input Queue

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**

rProcessing in memory