Indian Institute of Technology Mandi
February-June 2017 Semester

CS202: Data Structure and Algorithms

Programming Assignment 6 Problem Statements

Last date of submission of code: 24th April, 2017

Implement one of the following problems using C++ programing language.
Note:

1. Write a separate main programs to evaluate the functions in data structure minimum priority queue (i.e. binary heap or min heap). The main functions should have the options to read inputs from user and display. Implement minimum priority queue using MinPriorityQueue.hpp
2. Implement heap data structure using sequential linear list (array) data structure which you have implemented in first assignment i.e. using seqLinearList.hpp.
3. Write a separate main programs to simulate the process of handling the interrupts from I/O devices by processor.

Problem:

An I/O device generates interrupt signal to indicate processor that device is ready. Upon receiving the interrupt signal, processor pre-empt the currently running process and start running the interrupt service routine (ISR) of that I/O device. After the completion of the execution of ISR of that device, processor resumes back the execution of the process it has pre-empted. In reality multiple I/O devices are connected to a system and several I/O devices can generate interrupt at the same time or generate when ISR of one I/O device is executing. However, processor can execute only one ISR at a time. To handle the multiple devices at the same time interrupt priority is implemented. The operating system builds priority queue to implement this. This process can be simulated using the following information and algorithm.

Each interrupt can be considered as an element with following information: I/O device ID, Burst Time, Delay, and Priority Value.
I/O device ID: Device ID that uniquely identifies a particular I/O device
Burst Time: This is the amount of CPU time (in millisecond) the ISR will require to complete the execution. In real life this is not really known, but can be predicted with some degree of accuracy.
Delay: Delay (time difference) in millisecond between the appearance of previous interrupt and this interrupt in the sequence
Priority Value: Priority value of the device. Let this be an integer between 0 to 9. 0 being the highest priority and 9 being the lowest priority.
Apart from this processor also has priority value (**CPU priority**) equal to 10, i.e., lowest of all the priorities.

To simulate this consider *N* interrupt requests. Assume that the first interrupt signal from a device appear at 0th time. Next interrupt signal from the same device or some other device appear after "Delay" time and so on so forth.
For example: Let *N* = 4

Sl. No.	I/O Device ID	Burst Time (in ms)	Delay (in ms)	Priority Value
1	10	25	0	4
2	12	10	15	2
3	6	35	20	6
4	10	30	50	4

- All these interrupt requests are put into minimum priority queue based on their priority as and when they come.

- Maintain a counter to keep track of the time.
- Processor executes the ISR of the interrupt (device) with highest priority by removing it from priority queue. At the same time processor raises its priority level (**CPU priority**) equal to that of the device. This mechanism is to avoid any incoming interrupts with same or lower priority interrupting the current execution.
- If the **Burst Time** of the executing ISR is smaller than the appearance time (**Delay**) of the next interrupt request, processor simply completes the execution and then sets its priority (**CPU priority**) to 10.
- If the **Burst Time** of the executing ISR is larger than the appearance time (**Delay**) of the next interrupt request, processor checks the priority of the incoming interrupt request.
  - If the priority of the incoming interrupt is larger than the priority of the currently executing interrupt,
    - Processor pre-empt the currently running interrupt.
    - Insert the pre-empted interrupt request into the priority queue and make the **Burst Time** as the remaining execution time and **Delay** is not changed.
    - Start executing the incoming interrupt request.
    - Raise the **CPU priority** to the priority of the incoming interrupt request.
  - If the priority of the incoming interrupt is smaller than or equal to the priority of the currently executing interrupt,
    - Continue executing the current ISR

Continue these steps for all the interrupts in the queue. Make necessary valid assumptions wherever required.

Reference:

Chapter 4, Section 4.2.3: Carl Hamacher, Zvonko Vranesic, Safwat Zaky, “Computer Organization”, Fifth Edition