Operating System Project

CIE 302

Phase 1 : Scheduler

Schedulers algorithms:

SJF : Shortest Job First (nonpreemptive)

RR: Round Robin (Non-preemptive)

HPF: Highest Priority First (Preemptive version of RR)

SRTN: Shortest Remaining Time Next(Preemptive version of SJF)

Data Structures:

- Priority Queues with different priorities.
- FIFO Queue.

IPCS:

- Message Queue
- Semaphores

Process Generator:

- Input file
- Initiate Clk
- Creating the message queue.
- Forking the schedulers
- Clear the IPCS

Scheduler SJF:

- The scheduler picks the process at the front of the priority queue and sends a SIGCONT signal so that process continues its work.

- Then the scheduler waits for the chosen process to finish its work using waitpid() system call. When the process finishes the scheduler starts this loop again.

Scheduler RR:

- The scheduler picks the first process. Then the scheduler sends a SIGCONT signal so that process continues its work and goes to sleep for rem_sleep.

- When the scheduler wakes up, it checks whether the process has finished or not and take action accordingly.

Scheduler HPF:

- The scheduler picks the process at the front of the priority queue, sends a SIGCONT signal so that process continues its work.

- Then the scheduler goes to sleep until it finishes the process or gets another process with higher priority.

Scheduler SRTN:

- It works with the same logic as the scheduling_HPF but orders its process according to the remaining time not according to the priority.

Phase 2 : Memory

Data Structures:

- Priority Queues (Leaves Queue).
- Binary Tree.

Leaves queue:

- Storing the free nodes of the binary tree with a priority of the node memory size.
- Priority queue of pointers on memory blocks according to their both sizes and starting index of the memory block.
- keep track of the empty memory blocks.
- Facilitate the allocation process.

Binary Tree:

- Splitting the memory to the suitable and usable storage.
- keep track of the free and allocated memory blocks.
- It is main function is to organize the deallocation and the merging of memory spaces.

Allocation Function

- Check the process memory size.
- Pick the first suitable free node and delete it from the **Leaves Queue**
- Change the state from free to allocated.
- Same function for all schedulers

Deallocation Function

- Change the state of the node from allocated to free
- Check the neighbour node size from the binary tree if it is ready for merging.
- Insert the free node to the **Leaves Queue.**
- Same function for all schedulers

Test cases

HPF

Input1.txt

```
#id arrival runtime priority memsize
1   1   6   5 256
2   2   4   3 64
3   5   3   2 100
4   7   5   1 63
5   7   5   2 120
6   7   6   3 250
```

HPF.log

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 2 process 1 Stoped arr 1 total 6 remain 5 wait 0
At time 2 process 2 Started arr 2 total 4 remain 4 wait 0
At time 5 process 2 Stoped arr 2 total 4 remain 1 wait 0
At time 5 process 3 Started arr 5 total 3 remain 3 wait 0
At time 7 process 3 Stoped arr 5 total 3 remain 1 wait 0
At time 7 process 4 Started arr 7 total 5 remain 5 wait 0
At time 12 process 4 Finshed arr 7 total 5 remain 0 wait 0 TA 5 WTA 1.00
At time 12 process 3 Resumed arr 5 total 3 remain 1 wait 5
At time 13 process 3 Finshed arr 5 total 3 remain 0 wait 5 TA 8 WTA 2.67
At time 13 process 5 Started arr 7 total 5 remain 5 wait 6
At time 18 process 5 Finshed arr 7 total 5 remain 0 wait 6 TA 11 WTA 2.20
At time 18 process 6 Started arr 7 total 6 remain 6 wait 11
At time 24 process 6 Finshed arr 7 total 6 remain 0 wait 11 TA 17 WTA 2.83
At time 24 process 2 Resumed arr 2 total 4 remain 1 wait 19
At time 25 process 2 Finshed arr 2 total 4 remain 0 wait 19 TA 23 WTA 5.75
At time 25 process 1 Resumed arr 1 total 6 remain 5 wait 23
At time 30 process 1 Finshed arr 1 total 6 remain 0 wait 23 TA 29 WTA 4.83

```

HPFmemory.log

```
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 256 bytes for process 1 from 0 to 255
#At time 2 allocated 64 bytes for process 2 from 256 to 319
#At time 5 allocated 100 bytes for process 3 from 384 to 511
#At time 7 allocated 63 bytes for process 4 from 320 to 383
#At time 7 allocated 120 bytes for process 5 from 512 to 639
#At time 7 allocated 250 bytes for process 6 from 768 to 1023
#At time 12 freed 63 bytes for process 4 from 320 to 383
#At time 13 freed 100 bytes for process 3 from 384 to 511
#At time 18 freed 120 bytes for process 5 from 512 to 639
#At time 24 freed 250 bytes for process 6 from 768 to 1023
#At time 25 freed 64 bytes for process 2 from 256 to 319
#At time 30 freed 256 bytes for process 1 from 0 to 255

```

Thank you