OPERATING SYSTEMS

CMPN-303

Project Phase-1

Mohamed Ahmed Zakaria Reyad 1200053
Abdallah Ahmed Mohamed 1200033
Ahmad Mohsen Tahoon 1170093

Submitted to : Dr. Mona Farouk
Eng.Mohamed Hesham
Eng.Mohamed Alaa

Data structure Used :

1-Queue:

Which was used in round robin algorithm and Memory queue that is responsible to halt any process that has not been allocated due to memory being full..
Function used:

a) Queue(int Process):used to enqueue the id of a process.
b) dequeue():return int used to to get the pid of the first process of the queue.

2-Priority Queue:

Which is used in SRTN and HPF
Function used:

a) insert(int proc, int prio) : used to enqueue the id with the priority or AT in case of SRTN
b) pqdequeue(int proc): used to get priority with removing.
c) pqpeek(int proc): used to get priority without removing.

3-Message buffer:

a) Between process generator and scheduler.
b) Between process and scheduler and vice versa.

4-Signal:

a) Between process generator and clock.
b) Between process and scheduler and vice versa.

5-Binary Tree:

Typically the buddy memory allocation system is implemented with the use of a binary tree to represent used or unused split memory blocks.

6-Linked List:

Binary trees are basically two dimensional linked lists. Each node has a value and pointers to two sub-trees, one to the left and one the right. Both sub-trees may either be the value None or be the root node to another binary tree.

Algorithm used and results:

The communication between the .c files were done by message buffer and files were created between each other using fork and execl and we send needed parameters between the parent and the child using the arguments of the execl.

At first we ask the user to select a type of algorithm and then the scheduler chooses between the available algorithm available and then uses the algorithm picked by the user. As soon as a process is received by the scheduler a process.c file is created and it is enabled by a Scheduler if the given process is running.

HPF:

1. Getting the processes from the process generator at the arrival time.
2. Putting the processes in the PCB table.
3. Inserting the processes in the priority queue according to its priority.
4. Dequeuing the process to run it.
5. The process runs until it finishes its running time.
6. The scheduler receives the process parameters and save it in the PCB table
7. Then the scheduler signals to the next process in the queue to start its running process.
8. The scheduler saves every action along the process of execution (Starting, Resume, Stopped, and Finished).

SRTN:

1. Getting the processes from the process generator at the arrival time.
2. Putting the processes in the PCB table.
3. Inserting the processes in the priority queue using the remaining time as the argument for the priority queue.
4. If a process is running and a process has arrived it compares the remaining time of the two processes to select which one to run.
5. We have two cases:
   -In case the arriving process's remaining time is greater than the running process's remaining time,the scheduler won't switch the processes.
   -In case the arriving process's remaining time is less than the running process's remaining time,the scheduler switches the processes.
6. In the two cases when the process finishes the scheduler will start the next process in the queue (if any).
7. The scheduler saves every action along the process of execution (Starting, Resume, Stopped, and Finished).

RR:

1. Getting the processes from the process generator at the arrival time.
2. Putting the processes in the PCB table.
3. Inserting the processes in the queue.
4. Scheduling the processes according to their position in the queue.
5. While the process is running we see if the process generator sends any process.
6. We have two cases:
   - the process remaining time is greater than the quanta so after the time of quanta is finished, the scheduler will signal to the process to stop.
   -The process remaining time is less than the quanta so the process will run until it finishes the remaining time then sends its parameters to the scheduler.
7. In the two cases when the process finishes the scheduler will start the next process in the queue (if any).
8. The scheduler saves every action along the process of execution (Starting, Resume, Stopped, and Finished).

The memory allocation:

When scheduler receives a process, it calls allocate memory function that calls a function that finds nearest power of two that is equal or greater than the memory size of process, then calls a function that traverse on the tree to search for node with the same size of nearest power of two and free, then return the node that the process will occupy.

When a process is finished, deallocate function is called. It make the node free and call function merge buddies that merge two free buddies if the second buddy was free of processes

Our Assumptions:

1- If 2 processes have the same arrival time and Priority (in case of HPF), it chooses according which process' line came first.

2- If 2 processes have the same remaining time (in case of SRTN) it chooses according which process' line came first.

3- In SRTN, If a process enters with the same remaining as the running process, the running process continues without switching.

4-In RR if a process is running and just finished its quanta and a process arrived at the same time the one that just finished its quanta enter the queue first.

Workload per Individual:

Mohamed Ahmed Zakaria	1. HPF Algorithm 2. Corner cases of HPF 3. Corner cases of clock sync 4. 4.RR Algorithm 5. Saving the processes data of the input file 6. Priority queue
Abdallah Ahmed Mohamed	1. RR Algorithm 2. Corner Case RR 3. SRTN Algorithm 4. Message Buffers between Process generator and scheduler 5. Message Buffers between scheduler and Process 6. Printing to .log and .pref files 7. PCB struct 8. Process struct
Ahmad Mohsen Tahoon	1. HPF Algorithm 2. Corner cases of SRTN 3. Corner Case RR 4. SRTN Algorithm 5. The calculations on the .log and .pref file 6. Queue

Time Management:

We all worked in parallel all the time

Monday (12 hours)	Reading input file (30 min)
	Creating Messages buffers between the .c files and making sure its working (2 hours)
	The Layout of the scheduler (2 hour)
	HPF(7.5 Hours)
Tuesday (12 hours)	RR(12 hours)
	Corner cases for HPF(2 Hours)
	SRTN(2 Hours)
Thursday (4 hours)	RR(3 hours)
	Corner cases for RR(1 Hours)
	Calculating the the calculations of the .pref and .log file (3 hours)
Friday (12 hours)	Synchronization in RR(5)
	Corner cases for SRTN(6 Hours)
	Corner cases for HPF (4 Hours)
	Corner cases for clock sync(6 Hours)
Saturday (6 hours)	Corner cases for clock sync(4 Hours)
	Corner cases for RR(5 Hours)
	Printing to the output files(1 Hours)