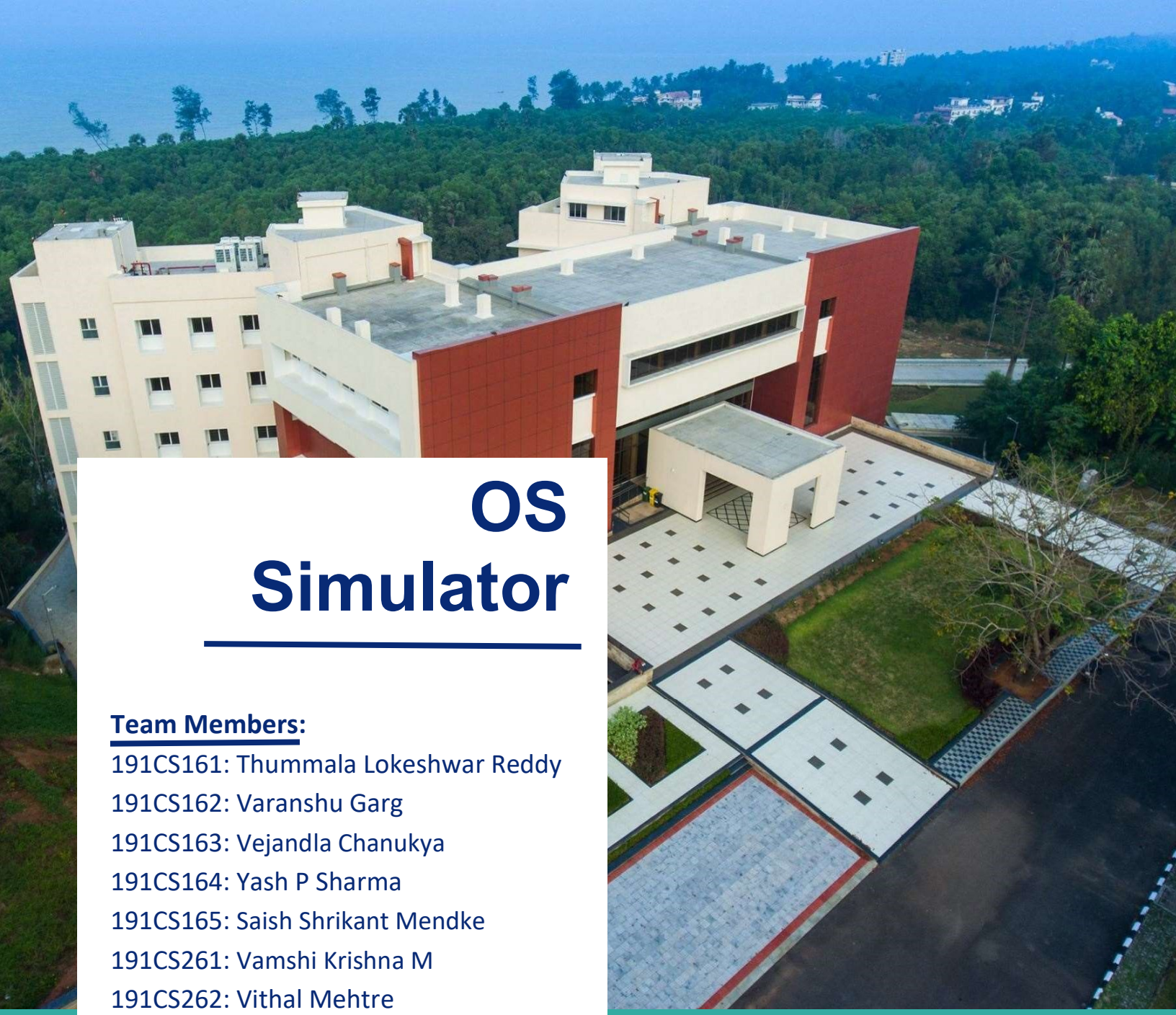

OS Simulator

Team Members:

191CS161: Thummala Lokeshwar Reddy
191CS162: Varanshu Garg
191CS163: Vejandla Chanukya
191CS164: Yash P Sharma
191CS165: Saish Shrikant Mendke
191CS261: Vamshi Krishna M
191CS262: Vithal Mehtre
191CS263: Venkat Kumar Yenumula
191CS264: Yukta Patil
191CS265: Vinesh S Talpankar

APRIL **23**

Submitted to: Dr. Shashidhar Sir

NITK Surathkal
Mangalore, India

Contents

Contributions

- **191CS161**: Thummala Lokeshwar Reddy & **191CS163**: Vejandla Chanukya
  - ✓ MVT – Best-Fit, Worst-Fit, First-Fit, Next-Fit
  - ✓ MFT – Best-Fit, Worst-Fit, First-Fit, Next-Fit
- **191CS162**: Varanshu Garg & **191CS262**: Vithal Mhetre
  - ✓ Banker's Algorithm
  - ✓ Priority-Scheduling, Preemptive and Non preemptive
  - ✓ HRRN
- **191CS164**: Yash P Sharma
  - ✓ Compilation of all the codes
  - ✓ CPU Scheduling Algorithms
- **191CS165**: Saish Shrikant Mendke
  - ✓ Producer-Consumer
  - ✓ Reader-Writer
  - ✓ Dining Philosophers
  - ✓ MVT and MFT
- **191CS261**: Vamshi Krishna M
  - ✓ FIFO, RR
  - ✓ SFJ, SRTF
  - ✓ LRTF, LJF
- **191CS263**: Venkat Kumar Yenumula & **191CS264**: Yukta Patil
  - ✓ Page replacement algorithms - LRU, MRU, Optimal, FIFO
  - ✓ Paging with Page table and offset (FIFO)
  - ✓ Banker's Algorithm
  - ✓ Report Making
- **191CS265**: Vinesh S Talpankar
  - ✓ FCFS, SSTF
  - ✓ SCAN, C-SCAN
  - ✓ LOOK, C-LOOK

Aim

This project aims to build a Menu-driven program that simulates various concepts of operating systems in a typical Operating System such as CPU scheduling, process synchronization, memory management & allocation, deadlock handling, page replacement, paging, and disk scheduling using C++ language.

CPU Scheduling Algorithms

CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them. Short-term schedulers, also known as dispatchers, make the decision of which process to execute next.
The CPU scheduling algorithms that we implemented are:

First Come First Serve (FCFS): Simplest scheduling algorithm that schedules according to arrival times of processes. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first.

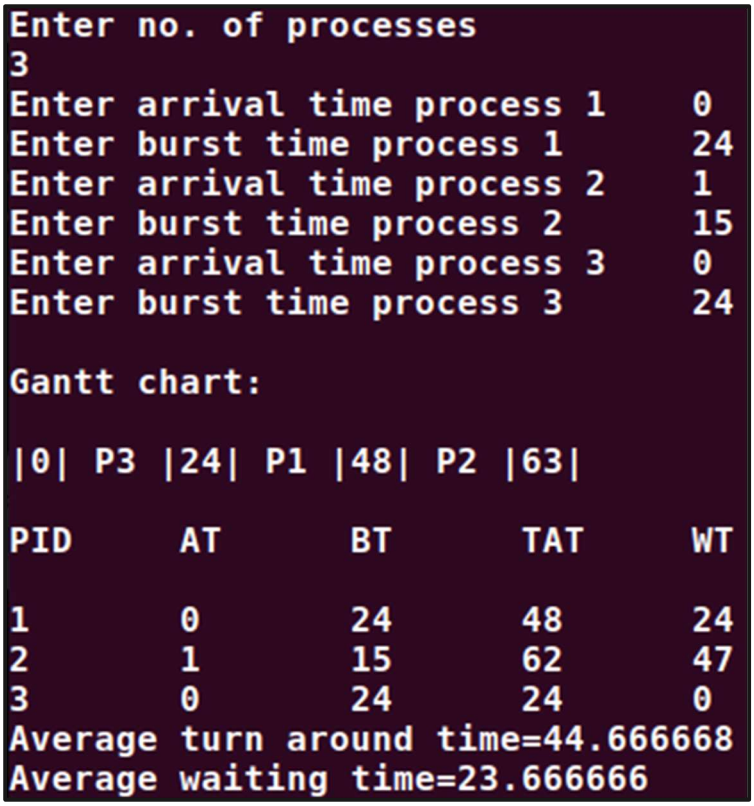

Fig. FCFS-Scheduling Algorithm

Shortest Job First (SJF): Process which have the shortest burst time are scheduled first. If two processes have the same bust time then FCFS is used to break the tie. It is a non-preemptive scheduling algorithm.

```
Enter no. of processes
3
Enter arrival time process 1    0
Enter burst time process 1      24
Enter arrival time process 2    0
Enter burst time process 2      15
Enter arrival time process 3    1
Enter burst time process 3      5

Gantt chart:

0 P2 15 P3 20 P1 44

PID     AT      BT      TAT     WT

1       0       24      44      20
2       0       15      15      0
3       1       5       19      14
Average turn around time=26.000000
Average waiting time=11.333333
```

Fig. SJF-Scheduling Algorithm

Shortest Remaining Time First (SRTF): It is preemptive mode of SJF algorithm in which jobs are schedule according to shortest remaining time.

```
Enter no of processes
3
Enter Arrival time of process 1
0
Enter the burst time of process 1
2
Enter Arrival time of process 2
0
Enter the burst time of process 2
5
Enter Arrival time of process 3
1
Enter the burst time of process 3
5
P.No  AT   BT   CT   TAT   WT
1     0    2    2    2     0
2     0    5    7    7     2
3     1    5    12    11    6
Average Turnaround Time 6.66667
Average Waiting Time    2.66667
Gantt Chart
|0 P1 1 | |1 P1 2 | |2 P2 3 | |3 P2 4 | |4 P2 5 | |5 P2 6 | |6 P2 7 | |7 P3 8 | |8 P3 9 | |9 P3 10 | |10 P3 11 | |11 P3 12 |
```

Fig. SRTF-Scheduling Algorithm

Round Robin Scheduling: Each process is assigned a fixed time (Time Quantum/Time Slice) in cyclic way. It is designed especially for the time-sharing system. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1-time quantum.

```
  Enter number of processes: 3
  Enter arrival time  of 1      0
  Enter burst time of 1 2
  Enter arrival time  of 2      2
  Enter burst time of 2 3
  Enter arrival time  of 3      2
  Enter burst time of 3 3
enter the time slice  2

Gantt chart:
 P1  |1|  P1  |2|  P2  |3|  P2  |4|  P3  |5|  P3  |6|  P2  |7|  P3  |8|

  P.ID  AT      BT      CT      TAT     WT
  1     0       2       2       2       0
  2     2       3       7       5       2
  3     2       3       8       6       3

  The average  TAT is 4.333333
  The average WT is 1.666667
```

Fig. Round Robin

Longest Job First (LJF): It is similar to SJF scheduling algorithm. But, in this scheduling algorithm, we give priority to the process having the longest burst time. This is non-preemptive in nature i.e., when any process starts executing, can't be interrupted before complete execution.

```
Enter no. of processes
3
Enter arrival time process 1    0
Enter burst time process 1      24

Enter arrival time process 2    0
Enter burst time process 2      15

Enter arrival time process 3    1
Enter burst time process 3      5


Gantt chart:

|0| P1 |24| P2 |39| P3 |44|

PID     AT      BT      TAT     WT

1       0       24      24      0
2       0       15      39      24
3       1       5       43      38

Average turn around time=35.333332
Average waiting time=20.666666
```

Fig. LFJ

Longest Remaining Time First (LRTF): It is preemptive mode of LJF algorithm in which we give priority to the process having largest burst time remaining.

```
Enter no of processes
3
Enter Arrival time process 1
0
Enter Brust time process 1
3
Enter Arrival time process 2
0
Enter Brust time process 2
2
Enter Arrival time process 3
1
Enter Brust time process 3
5
P.No  AT   BT   CT   TAT   WT
1     0    3    8    8     5
2     0    2    9    9     7
3     1    5    10    9     4
Average Turnaround Time         8.66667
Average Waiting Time    5.33333
Gantt Chart
|0 P1 1 | |1 P3 2 | |2 P3 3 | |3 P3 4 | |4 P1 5 | |5 P2 6 | |6 P3 7 | |7 P1 8 | |8 P2 9 | |9 P3 10 |
```

Fig. LRTF-Scheduling Algorithm

Priority Based scheduling (Non-Preemptive): In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then schedule according to arrival time. Here starvation of process is possible.

```
Enter no of process : 5

Enter arrival time and burst time and priority of Process 0  : 5 2 4

Enter arrival time and burst time and priority of Process 1  : 3 5 3

Enter arrival time and burst time and priority of Process 2  : 6 2 1

Enter arrival time and burst time and priority of Process 3  : 10 4 2

Enter arrival time and burst time and priority of Process 4  : 4 3 5

Process Number  Arrival Time    Burst Time      Waiting Time    TurnAround Time
1               3               5               0               5
2               6               2               2               4
3               10              4               0               4
0               5               2               9               11
4               4               3               12              15
Gantt chart-
0 idle 3 P1 8 P2 10 P3 14 P0 16 P4 19
Average waiting time:4.600000
Average Turn Around time:7.800000
```

Fig. Priority-Scheduling Algorithm(Non-Preemptive)

Preemptive Priority Scheduling: at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time.

```
Enter the number of processes: 5

Enter the arrival ,burst time and priority of process 1: 2 3 3

Enter the arrival ,burst time and priority of process 2: 4 2 2

Enter the arrival ,burst time and priority of process 3: 5 5 1

Enter the arrival ,burst time and priority of process 4: 16 5 4

Enter the arrival ,burst time and priority of process 5: 16 3 5

Process Number  Arrival Time    Burst Time      Waiting Time    Turn Around Time
1               2               3               7               10
2               4               2               5               7
3               5               5               0               5
4               16              5               0               5
5               16              3               5               8
Gantt Chart-
0 idle 1 idle 2 P1 3 P1 4 P2 5 P3 6 P3 7 P3 8 P3 9 P3 10 P2 11 P1 12 idle 13 idle 14 idle 15 idle 16 P4 17 P4 18 P4 19 P4 20 P4 21 P5 22 P5 23 P5 24
The average turnaround time is 7.000000 and the average waiting time is 3.400000
```

Fig. Priority-Scheduling Algorithm(Preemptive)

Highest Response Ratio Next (HRRN): In this scheduling, processes with highest response ratio are scheduled. This algorithm avoids starvation.

```
Enter no of process : 5

Enter arrival time and burst time of  Process 0  : 5 4

Enter arrival time and burst time of  Process 1  : 2 2

Enter arrival time and burst time of  Process 2  : 14 1

Enter arrival time and burst time of  Process 3  : 6 2

Enter arrival time and burst time of  Process 4  : 6 1

Process Number  Arrival Time    Burst Time      Waiting Time    TurnAround Time
1               2               2               0               2
0               5               4               0               4
4               6               1               3               4
3               6               2               4               6
2               14              1               0               1
Gantt chart-
0 idle 2 P1 4 idle 5 P0 9 P4 10 P3 12 idle 14 P2 15
Average waiting time:1.400000
Average Turn Around time:3.400000
```

Fig. HRRN-Scheduling Algorithm

Synchronization Problems

The following are our implementations of classical problems of synchronization as examples of a large class of concurrency-control problems.

Producer-Consumer Problem: Bounded Buffer problem is also called producer consumer problem. This problem is generalized in terms of the Producer-Consumer problem. Solution to this problem is, creating two counting semaphores “full” and “empty” to keep track of the current number of full and empty buffers respectively. Producers produce a product and consumers consume the product, but both use of one of the containers each time.

Readers and Writers Problem: A database is to be shared among several concurrent processes. Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database. We distinguish between these two types of processes by referring to the former as readers and to the latter as writers. In OS we call this situation as the readers-writers problem.

Dining-Philosophers Problem: The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both. This problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.

```
Enter 1 for reader-writer, 2 for producer-consumer, 3 for dining philosophers and any other number to exit: 1
Enter the number of writers and readers: 3 3
Enter the arrival and burst time of writer 0: 2 1
Enter the arrival and burst time of writer 1: 3 2
Enter the arrival and burst time of writer 2: 3 3
Enter the arrival and burst time of reader 0: 0 2
Enter the arrival and burst time of reader 1: 2 2
Enter the arrival and burst time of reader 2: 4 5
A reader has arrived before the first writer. Waiting for writers to start writing
Writer 0 started writing at t = 2
Reader 0 started reading at t = 3
Reader 1 started reading at t = 3
Reader 2 started reading at t = 4
Writer 1 started writing at t = 9
Writer 2 started writing at t = 11

Enter 1 for reader-writer, 2 for producer-consumer, 3 for dining philosophers and any other number to exit: 2
Enter the buffer size:4
Enter the number of producers and consumers: 3 3
Enter the arrival and burst time of producer 0: 1 2
Enter the arrival and burst time of producer 1: 3 1
Enter the arrival and burst time of producer 2: 5 3
Enter the arrival and burst time of consumer 0: 0 2
Enter the arrival and burst time of consumer 1: 2 1
Enter the arrival and burst time of consumer 2: 4 2
Consumer has arrived before a producer but the buffer is empty
Producer 0 started producing at time 1
Consumer 0 started consuming at time 3
Consumer has arrived before a producer but the buffer is empty
Producer 1 started producing at time 5
Consumer 1 started consuming at time 6
Consumer has arrived before a producer but the buffer is empty
Producer 2 started producing at time 7
Consumer 2 started consuming at time 10

Enter 1 for reader-writer, 2 for producer-consumer, 3 for dining philosophers and any other number to exit: 3
Enter the number of philosophers: 5
Enter the arrival and burst time of philosopher 0 : 2 1
Enter the arrival and burst time of philosopher 1 : 0 2
Enter the arrival and burst time of philosopher 2 : 3 2
Enter the arrival and burst time of philosopher 3 : 0 3
Enter the arrival and burst time of philosopher 4 : 1 2
Philosopher 1 has started dining at 0
Philosopher 3 has started dining at 0
Philosopher 0 has started dining at 2
Philosopher 4 has started dining at 3
Philosopher 2 has started dining at 3
```

Fig. Reader-Writers, Producers-Consumers, Dining-Philosophers

Banker's Algorithm

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "safe-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

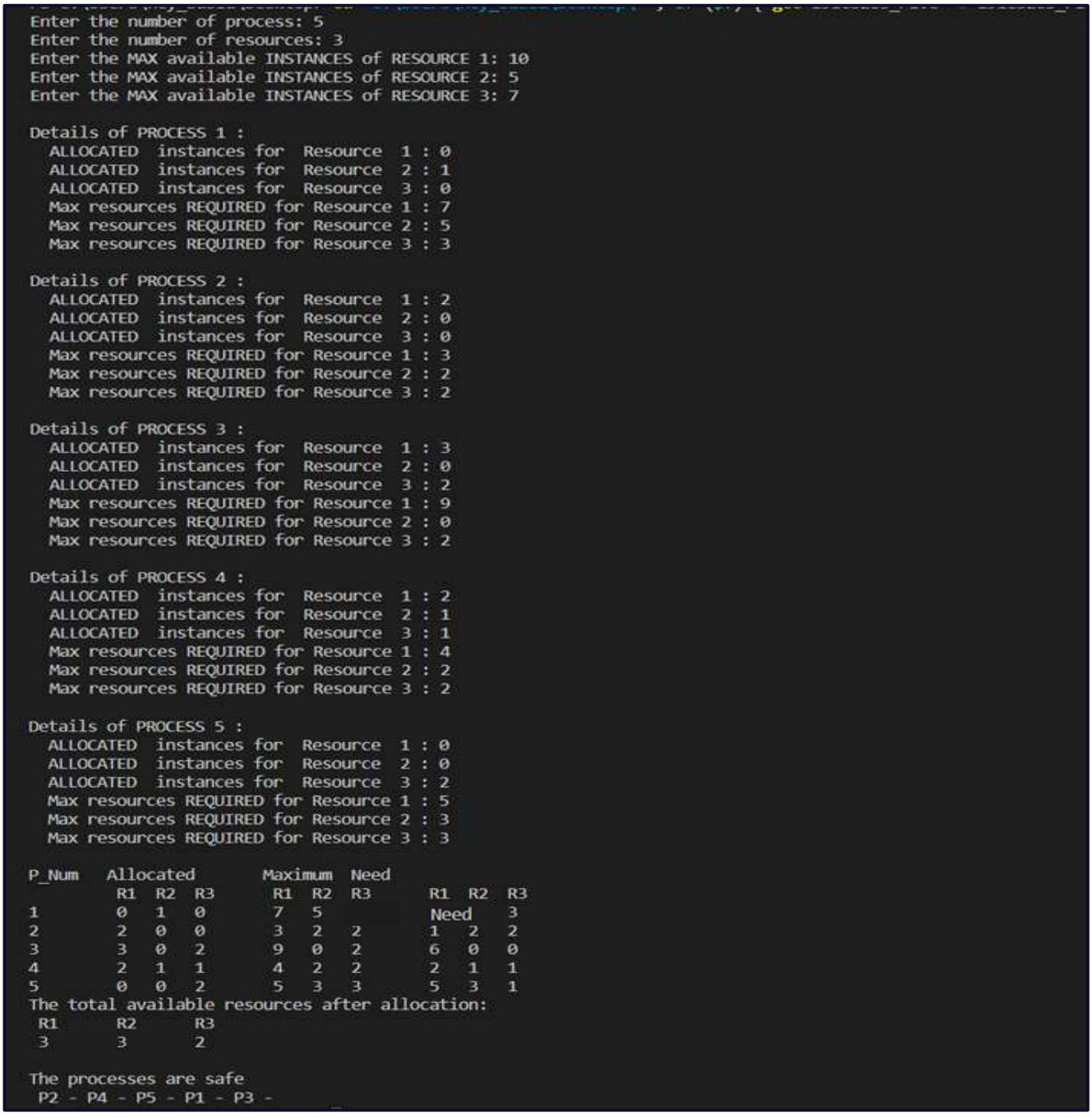

Fig. Banker's Algorithm

Memory Management Techniques

Memory Management is the function responsible for allocating and managing computer's main memory. Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory. Executing process must be loaded entirely in main-memory. In order to load them into the memory here we follow to methods:

MFT (Multiprogramming with a Fixed number of Tasks): one of the old memory management techniques in which the memory is partitioned into fixed size partitions and each job is assigned to a partition. The memory assigned to a partition does not change.

MVT (Multiprogramming with a Variable num. of Tasks): The memory management technique in which each job gets just the amount of memory it needs. That is, the partitioning of memory is dynamic and changes as jobs enter and leave the system. MVT is a more "efficient" user of resources. MFT suffers with the problem of internal fragmentation and MVT suffers with external fragmentation.

In **Memory Allocation**, when there is more than one partition freely available to accommodate a process's request, a partition must be selected. To choose a particular partition, a partition allocation method is needed. A partition allocation method is considered better if it avoids internal fragmentation. We have implemented the following allocation methods:

First Fit: In the first fit, the partition is allocated which is the first sufficient block from the top of Main Memory. It scans memory from the beginning and chooses the first available block that is large enough. Thus, it allocates the first hole that is large enough.

Best Fit: Allocate the process to the partition which is the first smallest sufficient partition among the free available partition. It searches the entire list of holes to find the smallest hole whose size is greater than or equal to the size of the process.

Worst Fit: Allocate the process to the partition which is the largest sufficient among the freely available partitions available in the main memory. It is opposite to the best-fit algorithm. It searches the entire list of holes to find the largest hole and allocate it to process.

Next Fit: Next fit is a modified version of 'first fit'. It begins as the first fit to find a free partition but when called next time it starts searching from where it left off, not from the beginning. This policy makes use of a roving pointer. The pointer moves along the memory chain to search for a next fit. This helps in, to avoid the usage of memory always from the head (beginning) of the free block chain.

MFT (Multiprogramming with a Fixed number of Tasks):

First Fit

```
Enter 1 for first fit, 2 for best fit ,3 for worst fit and 4 for next fit: 1
Enter the total size: 22
Total number of partitions: 5
Size of partition 0: 5
Size of partition 1: 4
Size of partition 2: 6
Size of partition 3: 7
Size of partition 4: 3
Partition 4 cannot be made due to insufficient space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 4
Process has been accomodated in partition 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 3
Process has been accomodated in partition 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 7
Process has been accomodated in partition 3

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 8KB
Partition 0 has P0 with size 4KB and internal fragmentation 1
Partition 1 has P1 with size 3KB and internal fragmentation 1
Partition 2 is empty with size 6KB free space
Partition 3 has P2 with size 7KB and internal fragmentation 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 4
Enter the size of process 4: 5
Process has been accomodated in partition 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 5
Enter the size of process 5: 1
Process cannot be accomodated due to internal fragmentation
```

Fig. MFT First-Fit

Best Fit

```
Enter 1 for first fit, 2 for best fit ,3 for worst fit and 4 for next fit: 2
Enter the total size: 22
Total number of partitions: 5
Size of partition 0: 5
Size of partition 1: 4
Size of partition 2: 6
Size of partition 3: 7
Size of partition 4: 3
Partition 4 cannot be made due to insufficient space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 3
Process has been accomodated in partition 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 7
Process has been accomodated in partition 3

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 5
Process has been accomodated in partition 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 7KB
Partition 0 has P2 with size 5KB and internal fragmentation 0
Partition 1 has P0 with size 3KB and internal fragmentation 1
Partition 2 is empty with size 6KB free space
Partition 3 has P1 with size 7KB and internal fragmentation 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 2
Enter the index of process to be deleted: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 4
Enter the size of process 4: 6
Process has been accomodated in partition 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 8KB
Partition 0 has P2 with size 5KB and internal fragmentation 0
Partition 1 has P0 with size 3KB and internal fragmentation 1
Partition 2 has P4 with size 6KB and internal fragmentation 0
Partition 3 is empty with size 7KB free space
```

Fig. MFT Best-Fit

Worst Fit

```
Enter 1 for first fit, 2 for best fit ,3 for worst fit and 4 for next fit: 3
Enter the total size: 22
Total number of partitions: 5
Size of partition 0: 5
Size of partition 1: 4
Size of partition 2: 6
Size of partition 3: 7
Size of partition 4: 4
Partition 4 cannot be made due to insufficient space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 3
Process has been accomodated in partition 3

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 5
Process has been accomodated in partition 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 5
Process has been accomodated in partition 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 9KB
Partition 0 has P2 with size 5KB and internal fragmentation 0
Partition 1 is empty with size 4KB free space
Partition 2 has P1 with size 5KB and internal fragmentation 1
Partition 3 has P0 with size 3KB and internal fragmentation 4

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 3
Enter the size of process 3: 5
Process cannot be accomodated due to external fragmentation

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 4
Enter the size of process 4: 4
Process has been accomodated in partition 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 5KB
Partition 0 has P2 with size 5KB and internal fragmentation 0
Partition 1 has P4 with size 4KB and internal fragmentation 0
Partition 2 has P1 with size 5KB and internal fragmentation 1
Partition 3 has P0 with size 3KB and internal fragmentation 4
```

Fig. MFT Worst-Fit

Next Fit

```
Enter 1 for first fit, 2 for best fit ,3 for worst fit and 4 for next fit: 4
Enter the total size: 22
Total number of partitions: 5
Size of partition 0: 5
Size of partition 1: 4
Size of partition 2: 7
Size of partition 3: 6
Size of partition 4: 4
Partition 4 cannot be made due to insufficient space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 3
Process has been accomodated in partition 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 7
Process has been accomodated in partition 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 5
Process has been accomodated in partition 3

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 7KB
Partition 0 has P0 with size 3KB and internal fragmentation 2
Partition 1 is empty with size 4KB free space
Partition 2 has P1 with size 7KB and internal fragmentation 0
Partition 3 has P2 with size 5KB and internal fragmentation 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 2
Enter the index of process to be deleted: 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 4
Enter the size of process 4: 2
Process has been accomodated in partition 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 10KB
Partition 0 has P0 with size 3KB and internal fragmentation 2
Partition 1 has P4 with size 2KB and internal fragmentation 2
Partition 2 has P1 with size 7KB and internal fragmentation 0
Partition 3 is empty with size 6KB free space
```

Fig. MFT Next-Fit

MVT (Multiprogramming with a Variable num. of Tasks)

First Fit

```
Total size: 22
Enter 1 for first fit, 2 for best fit, 3 for worst fit and 4 for next fit: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 5

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 10

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 6

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 1KB
Partition 1 has P0 with size 5KB
Partition 2 has P1 with size 10KB
Partition 3 has P2 with size 6KB
Partition 4 is empty with size 1KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 2
Enter the index of process to be deleted: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 3
Enter the size of process 3: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 10KB
Partition 1 has P0 with size 5KB
Partition 2 has P3 with size 1KB
Partition 3 is empty with size 9KB free space
Partition 4 has P2 with size 6KB
Partition 5 is empty with size 1KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 5
Enter the size of process 5: 10
Process cannot be accomodated due to external fragmentation

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 10KB
Partition 1 has P0 with size 5KB
Partition 2 has P3 with size 1KB
Partition 3 is empty with size 9KB free space
Partition 4 has P2 with size 6KB
Partition 5 is empty with size 1KB free space
```

Fig. MVT First-Fit

Best Fit

```
Total size: 22
Enter 1 for first fit, 2 for best fit, 3 for worst fit and 4 for next fit: 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 6

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 13

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 2

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 1KB
Partition 1 has P0 with size 6KB
Partition 2 has P1 with size 13KB
Partition 3 has P2 with size 2KB
Partition 4 is empty with size 1KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 2
Enter the index of process to be deleted: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 14KB
Partition 1 has P0 with size 6KB
Partition 2 is empty with size 13KB free space
Partition 3 has P2 with size 2KB
Partition 4 is empty with size 1KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 3
Enter the size of process 3: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 13KB
Partition 1 has P0 with size 6KB
Partition 2 is empty with size 13KB free space
Partition 3 has P2 with size 2KB
Partition 4 has P3 with size 1KB

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 5
Enter the size of process 5: 11
```

Fig. MVT Best-Fit

Worst Fit

```
Total size: 20
Enter 1 for first fit, 2 for best fit, 3 for worst fit and 4 for next fit: 3

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 5

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 7

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 4

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 4KB
Partition 1 has P0 with size 5KB
Partition 2 has P1 with size 7KB
Partition 3 has P2 with size 4KB
Partition 4 is empty with size 4KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 2
Enter the index of process to be deleted: 0

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 9KB
Partition 1 is empty with size 5KB free space
Partition 2 has P1 with size 7KB
Partition 3 has P2 with size 4KB
Partition 4 is empty with size 4KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 4
Enter the size of process 4: 5

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 4KB
Partition 1 has P4 with size 5KB
Partition 2 has P1 with size 7KB
Partition 3 has P2 with size 4KB
Partition 4 is empty with size 4KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 6
Enter the size of process 6: 5
Process cannot be accomodated as only 4KB is remaining
```

Fig. MVT Worst Fit

Next Fit

```
Total size: 22
Enter 1 for first fit, 2 for best fit, 3 for worst fit and 4 for next fit: 4

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 0
Enter the size of process 0: 6

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 1
Enter the size of process 1: 7

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 2
Enter the size of process 2: 3

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 6KB
Partition 1 has P0 with size 6KB
Partition 2 has P1 with size 7KB
Partition 3 has P2 with size 3KB
Partition 4 is empty with size 6KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 2
Enter the index of process to be deleted: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 3
Enter the size of process 3: 5

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 8KB
Partition 1 has P0 with size 6KB
Partition 2 is empty with size 7KB free space
Partition 3 has P2 with size 3KB
Partition 4 has P3 with size 5KB
Partition 5 is empty with size 1KB free space

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 1
Enter the process id: 6
Enter the size of process 6: 1

Enter 1 to add a process, 2 to delete a process and 3 to print details of process and other number to exit: 3
Total Space remaining: 7KB
Partition 1 has P0 with size 6KB
Partition 2 is empty with size 7KB free space
Partition 3 has P2 with size 3KB
Partition 4 has P3 with size 5KB
Partition 5 has P6 with size 1KB
```

Fig. MVT Next Fit

Page Replacement Algorithms

Physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults. The following are the page replacement algorithms that we implemented:

First in First Out (FIFO): This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

```
Enter the number of frames : 3
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 1

P1 got accomodated in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 2

P2 got accomodated in F1
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 3

P3 got accomodated in F2
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 1

P1 is already present in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 4

P4 got accomodated by replacing P1 in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 4

P4 got accomodated by replacing P1 in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 2

P2 is already present in F1
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 5

P5 got accomodated by replacing P2 in F1
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
2
--------------------
 1  2  3  1  4  2  5
--------------------
 1  1  1  1  4  4  4
    2  2  2  2  2  5
       3  3  3  3  3

Total number of page faults : 5

page fault ratio : 0.714286

1. Add pages into frame
2. Display
3. Exit
Enter the choice:
3
```

Optimal Page replacement: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

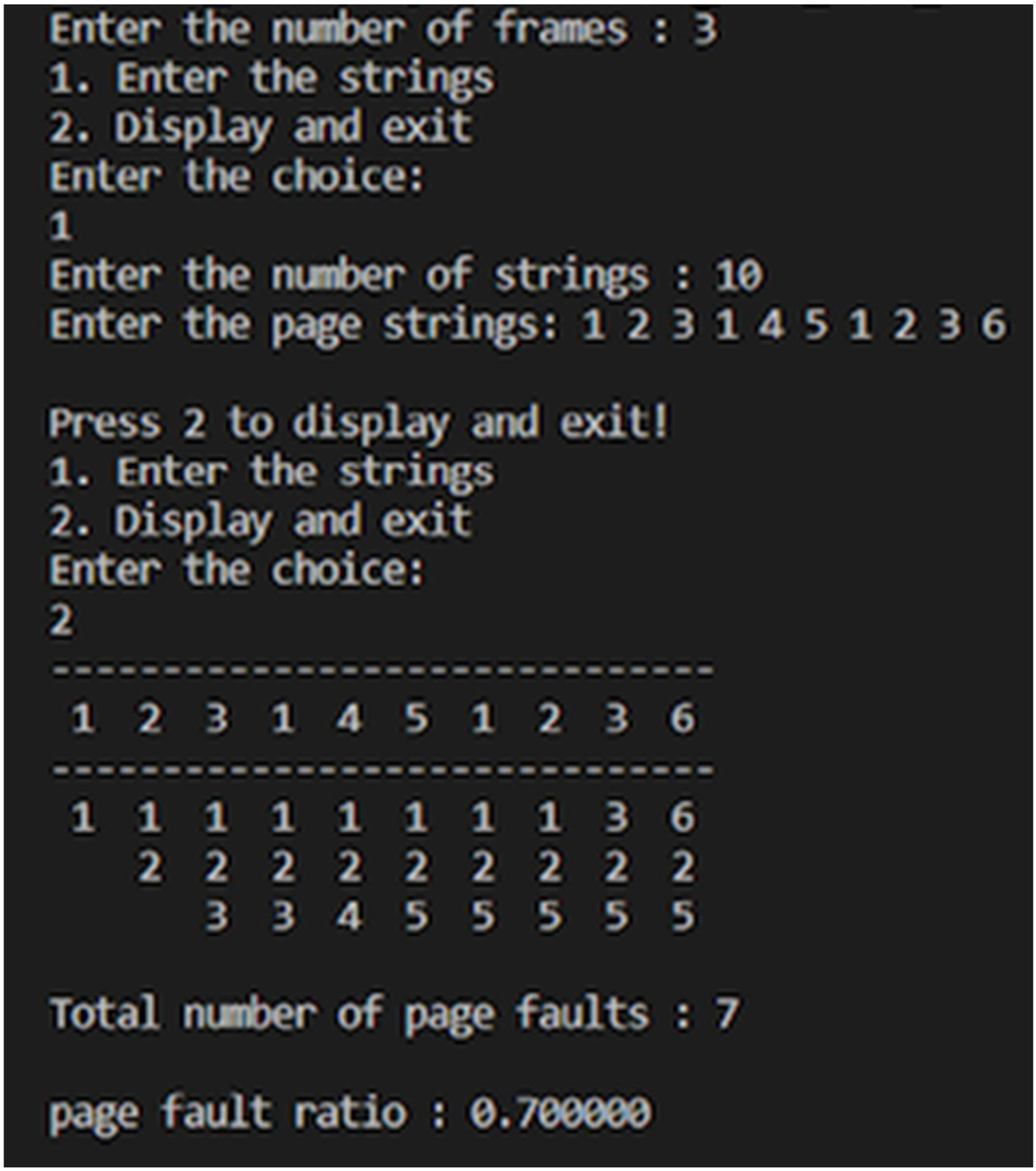

Fig. Optimal

Least Recently Used: In this, page will be replaced which is **L**east **R**ecently **U**sed.

```
Enter the number of frames : 3
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 1

P1 got accomodated in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 2

P2 got accomodated in F1
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 3

P3 got accomodated in F2
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 1

P1 is already present in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 4

P4 got accomodated by replacing P2 in F1
1. Add pages into frame
2. Display
3. Exit
```

```
3. Exit
Enter the choice:
1
Enter the page : 4

P4 got accomodated by replacing P2 in F1
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 3

P3 is already present in F2
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 5

P5 got accomodated by replacing P1 in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
2
--------------------
 1  2  3  1  4  3  5
--------------------
 1  1  1  1  1  1  5
    2  2  2  4  4  4
       3  3  3  3  3

Total number of page faults : 5

page fault ratio : 0.714286

1. Add pages into frame
2. Display
3. Exit
Enter the choice:
3
PS D:\01_Important\Studies\02_Year_2\12_CS
```

Fig. LRU

Most Recently Used: In this, page will be replaced which is **M**ost **R**ecently **U**sed.

```
Enter the number of frames : 3
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 1

P1 got accomodated in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 2

P2 got accomodated in F1
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 3

P3 got accomodated in F2
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 1

P1 is already present in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 4

P4 got accomodated by replacing P1 in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 4

P4 got accomodated by replacing P1 in F0
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 3

P3 is already present in F2
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
1
Enter the page : 5

P5 got accomodated by replacing P3 in F2
1. Add pages into frame
2. Display
3. Exit
Enter the choice:
2
--------------------
 1  2  3  1  4  3  5
--------------------
 1  1  1  1  4  4  4
    2  2  2  2  2  2
       3  3  3  3  5

Total number of page faults : 5

page fault ratio : 0.714286

1. Add pages into frame
2. Display
3. Exit
Enter the choice:
3
PS D:\01_Important\Studies\02_Year_2\12_CS
```

Fig. MRU

Paging Technique-Page Table

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous. Logical Address or Virtual Address (represented in bits): An address generated by the CPU. Physical Address (represented in bits): An address actually available on memory unit.

The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

Here we implemented paging technique that follows FIFO page replacement policy and addresses are translated accordingly.

```
Enter the size of Virtual Memory : 100
Enter the size of Physical Memory : 60
Enter the size each page : 20
Num of Pages in Virtual Memory : 5
Num of Pages in Physical Memory : 3

1.Enter the Virtual Address
2.Display
3.Exit
Enter the choice : 1
format: pageNumber|Word_offset
2 255
P2 NOt present in PM, and is accomodated at in Frame F0

1.Enter the Virtual Address
2.Display
3.Exit
Enter the choice : 1
format: pageNumber|Word_offset
4 912
P4 NOt present in PM, and is accomodated at in Frame F1

1.Enter the Virtual Address
2.Display
3.Exit
Enter the choice : 1
format: pageNumber|Word_offset
1 2045
P1 NOt present in PM, and is accomodated at in Frame F2

1.Enter the Virtual Address
2.Display
3.Exit
Enter the choice : 1
format: pageNumber|Word_offset
2 614
Phy Address for the above Virtual Address
0 614
1.Enter the Virtual Address
2.Display
3.Exit
Enter the choice : 1
format: pageNumber|Word_offset
2 614
Phy Address for the above Virtual Address
0 614
1.Enter the Virtual Address
2.Display
3.Exit
Enter the choice : 2

Page Table
0.
1. 2
2. 0
3.
4. 1

Physical Memory
0. 2
1. 4
2. 1

Disk
0. 0
1. 1
2. 2
3. 3
4. 4
total number of page faults : 3
total number of strings : 4
Page fault ratio : 0.750000
```

Disk Scheduling Algorithms

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling. Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So, the disk scheduling algorithm that gives minimum average seek time is better. These are the following Disk Scheduling Algorithms that we implemented:

FCFS: FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Example:

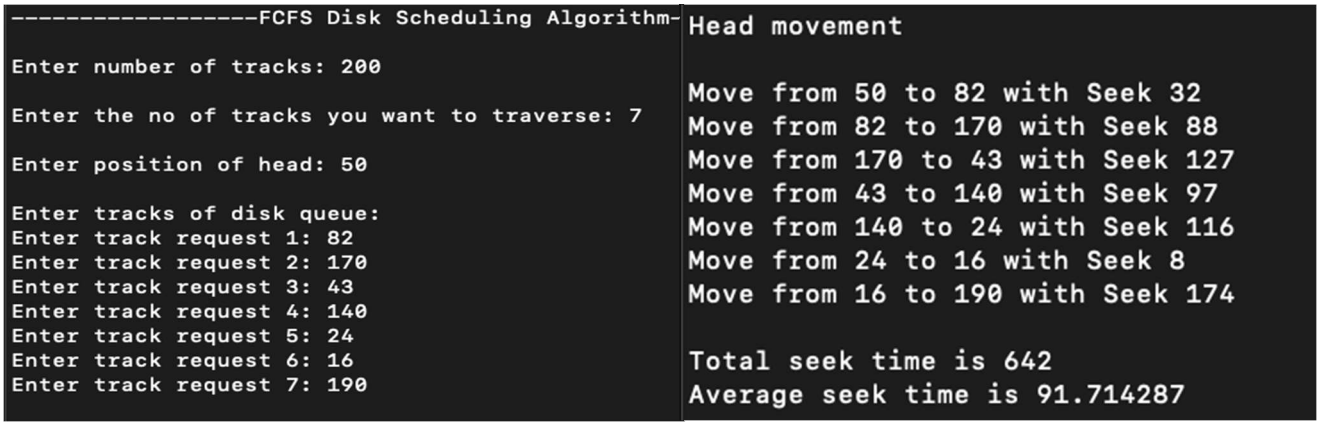

Fig. FCFS-Disk Scheduling

SSTF: In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

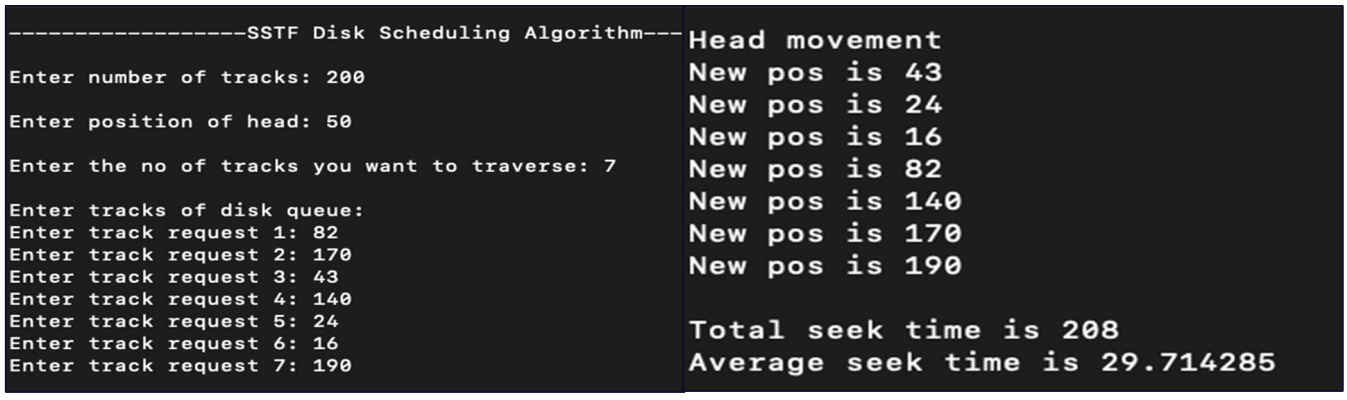

Fig. SSTF-Disk Scheduling

SCAN: In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as elevator algorithm. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

```
------------------SCAN Disk Scheduling Algorithm--

Enter number of tracks : 200
Enter number of requests : 7
Enter current track position : 50

Enter the requests
Enter track request 1: 82
Enter track request 2: 170
Enter track request 3: 43
Enter track request 4: 140
Enter track request 5: 24
Enter track request 6: 16
Enter track request 7: 190
Head movement
Move from 50 to 82 with Seek 32
Move from 82 to 140 with Seek 58
Move from 140 to 170 with Seek 30
Move from 170 to 190 with Seek 20
Move from 190 to 199 with Seek 9
Move from 199 to 43 with Seek 156
Move from 43 to 24 with Seek 19
Move from 24 to 16 with Seek 8

Total seek time is 332
Average seek time is 41.500000
```

Fig. SCAN-Disk Scheduling

C-SCAN: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

```
------------------CSCAN Disk Scheduling Algorithm--

Enter number of tracks : 200
Enter number of requests : 7
Enter current track position : 50

Enter the requests
Enter track request 1: 82
Enter track request 2: 170
Enter track request 3: 43
Enter track request 4: 140
Enter track request 5: 24
Enter track request 6: 16
Enter track request 7: 190
Head movement
Move from 50 to 82 with Seek 32
Move from 82 to 140 with Seek 58
Move from 140 to 170 with Seek 30
Move from 170 to 190 with Seek 20
Move from 190 to 199 with Seek 9
Move from 199 to 0 with Seek 199
-Move from 0 to 16 with Seek 16
Move from 16 to 24 with Seek 8
Move from 24 to 43 with Seek 19

Total seek time is 391
Average seek time is 48.875000
```

Fig. C-SCAN-Disk Scheduling

LOOK: It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus, it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

```
------------------LOOK Disk Scheduling Algorithm-

Enter number of tracks : 200
Enter number of requests : 7
Enter current track position : 50

Enter the requests
Enter track request 1: 82
Enter track request 2: 170
Enter track request 3: 43
Enter track request 4: 140
Enter track request 5: 24
Enter track request 6: 16
Enter track request 7: 190
```

```
Head movement
Move from 50 to 82 with Seek 32
Move from 82 to 140 with Seek 58
Move from 140 to 170 with Seek 30
Move from 170 to 190 with Seek 20
Move from 190 to 43 with Seek 147
Move from 43 to 24 with Seek 19
Move from 24 to 16 with Seek 8

Total seek time is 314
Average seek time is 39.250000
```

Fig. LOOK -Disk Scheduling

C-LOOK: As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

```
------------------CLOOK Disk Scheduling Algorithm-

Enter number of tracks : 200
Enter number of requests : 7
Enter current track position : 50

Enter the requests
Enter track request 1: 82
Enter track request 2: 170
Enter track request 3: 43
Enter track request 4: 140
Enter track request 5: 24
Enter track request 6: 16
Enter track request 7: 190
```

```
Head movement
Move from 50 to 82 with Seek 32
Move from 82 to 140 with Seek 58
Move from 140 to 170 with Seek 30
Move from 170 to 190 with Seek 20
Move from 190 to 16 with Seek 174
Move from 16 to 24 with Seek 8
Move from 24 to 43 with Seek 19

Total seek time is 341
Average seek time is 42.625000
```

Fig. C-LOOK -Disk Scheduling

Conclusion

Hence we have simulated all the Operating System Concepts taught to us in this course in a Menu-driven fashion. We now have a better understanding of the core concepts of an operating system. We can further develop this project in the future by adding more concepts related to Operating System.

---END---