Course Code : MCS-203
Course Title : Operating Systems
Assignment Number : PGDCA(I)/203/Assignment/2023
Last Date of Submission : 30th April, 2023 (for January session)
31st October, 2023 (for July session)

This assignment has six questions. Answer all the questions. Rest 20 marks are for viva voce. You April use illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the Programme Guide.

Question 1: (15 Marks) Consider the following jobs:

Job #	Arrival time	Run time
A	0	7
B	2	4
C	3	6
D	5	3

a) Using the FCFS method, compute the completion times of the above jobs, average turn around time and average waiting time.
b) Using the SJF (Shortest Job First) method, compute the completion times of the above jobs, the average turn around time and the average waiting time.
c) Using the Round Robin method (with Quantum = 3), compute the completion times of the above jobs and the average waiting time.
Ans.

a) Using the FCFS (First-Come, First-Served) method, the completion times, turnaround time, and waiting time for each job are as follows:

Job#	Arrival Time	Run Time	Completion Time	Turnaround Time	Waiting Time
A	0	7	7	7	0
B	2	4	11	9	5
C	3	6	17	14	8
D	5	3	20	15	12

Average Turnaround Time = (7 + 9 + 14 + 15) / 4 = 11.25 Average Waiting Time = (0 + 5 + 8 + 12) / 4 = 6.25

b) Using the SJF (Shortest Job First) method, the completion times, turnaround time, and waiting time for each job are as follows:

Job#	Arrival Time	Run Time	Completion Time	Turnaround Time	Waiting Time
A	0	7	7	7	0
B	2	4	11	9	5
D	5	3	14	9	6
C	3	6	20	17	11

Average Turnaround Time = (7 + 9 + 9 + 17) / 4 = 10.5 Average Waiting Time = (0 + 5 + 6 + 11) / 4 = 5.5

c) Using the Round Robin method (with Quantum = 3), the completion times and waiting time for each job are as follows:

Job#	Arrival Time	Run Time	Completion Time	Waiting Time
A	0	7	10	3
B	2	4	15	9
C	3	6	20	11
D	5	3	18	10

Average Waiting Time = (3 + 9 + 11 + 10) / 4 = 8.25

Note that in the Round Robin method, the completion time for each job depends on the quantum size, which is 3 in this case. Jobs are processed in a cyclic order, and each job runs for the quantum size (or less if the job completes before the quantum is over) before being preempted and the next job in the queue is processed. The waiting time for a job is the time it spends waiting in the queue before it starts processing, and is calculated as the difference between the completion time and the sum of the arrival times and the run times of all previous jobs in the queue.

Question 2: (20 Marks) On a disk with 1000 cylinders, numbers 0 to 999, compute the number of tracks the disk arm must move to satisfy all the requests in the disk queue. Assume the last request serviced was at track 345 and head is moving to track 0. The queue in FIFO order contains requests for the following tracks: 128, 870, 694, 470, 107, 378. Show the disk arm movement and calculate the number of tracks traversed using the following policies:
(a) FIFO (b) SSTF (c) SCAN (d) LOOK
Ans.

(a) FIFO:
In FIFO policy, requests are served in the order in which they are received. Therefore, the disk arm will move to the requested track one by one. The movement of the disk arm for the given requests is shown below:
Disk Arm Movement: 345 -> 128 -> 870 -> 694 -> 470 -> 107 -> 378 -> 0
The total number of tracks traversed is:
345-128 + 870-128 + 870-694 + 694-470 + 470-107 + 378-107 + 378-0 = 2126
Therefore, the total number of tracks traversed using FIFO policy is 2126.

(b) SSTF:
In SSTF policy, the next request to be served is the one that is closest to the current track position. Therefore, the disk arm will move to the nearest request each time. The movement of the disk arm for the given requests is shown below:
Disk Arm Movement: 345 -> 378 -> 470 -> 694 -> 870 -> 128 -> 107 -> 0
The total number of tracks traversed is:
345-378 + 470-378 + 694-470 + 870-694 + 870-128 + 128-107 + 107-0 = 1824
Therefore, the total number of tracks traversed using SSTF policy is 1824.

(c) SCAN:
In SCAN policy, the disk arm moves in one direction, servicing requests along the way, until it reaches the end of the disk. Then, it reverses direction and moves back in the opposite direction, servicing requests along the way. The movement of the disk arm for the given requests is shown below:
Disk Arm Movement: 345 -> 378 -> 470 -> 694 -> 870 -> 999 -> 0 -> 107 -> 128
The total number of tracks traversed is:
870-345 + 999-870 + 999-0 + 128-0 = 1852
Therefore, the total number of tracks traversed using SCAN policy is 1852.

(d) LOOK:
In LOOK policy, the disk arm moves in one direction, servicing requests along the way, until there are no more requests in that direction. Then, it reverses direction and moves back in the opposite direction, servicing requests along the way. The movement of the disk arm for the given requests is shown below:
Disk Arm Movement: 345 -> 378 -> 470 -> 694 -> 870 -> 999 -> 107 -> 128
The total number of tracks traversed is:
870-345 + 999-870 + 999-107 + 128-107 = 1545
Therefore, the total number of tracks traversed using LOOK policy is 1545.

Question 3: (15 Marks) Consider the following page-reference string: 1, 7. 6, 5, 4 , 3, 2, 1, 2, 3, 4, 2, 1, 3, 4, 6, 2, 1, 2, 3, 7, 6, 3, 2 How many page faults would occur for following replacement algorithms assuming four frames? Remember that all frames are initially empty, so your first unique pages will all cost one fault each.
i. FIFO replacement.
ii. LRU replacement.
iii. Optimal replacement.
Ans.

We will simulate the page replacement algorithms and count the number of page faults for each algorithm assuming four frames.

Initial page frames:
_ _ _ _

i. FIFO replacement:

We will use the first-in, first-out algorithm to replace the page that has been in the memory for the longest time.

1 _ _ _ // Page fault, insert 1
1 7 _ _ // Page fault, insert 7
1 7 6 _ // Page fault, insert 6
1 7 6 5 // Page fault, insert 5
4 7 6 5 // Page fault, replace 1 with 4
4 3 6 5 // Page fault, replace 7 with 3
4 3 2 5 // Page fault, replace 6 with 2
4 3 2 1 // Page fault, replace 5 with 1

4 3 2 1 // 2 already in memory
4 3 2 1 // 3 already in memory
4 2 5 1 // Page fault, replace 3 with 5
4 2 3 1 // Page fault, replace 5 with 3
4 2 3 1 // 4 already in memory
2 7 3 1 // Page fault, replace 4 with 7
2 7 3 1 // 1 already in memory
2 3 6 1 // Page fault, replace 7 with 6
2 3 6 1 // 2 already in memory
2 3 6 1 // 3 already in memory
7 3 6 2 // Page fault, replace 1 with 7
The total number of page faults is 14.

ii. LRU replacement:

We will use the least recently used algorithm to replace the page that has not been used for the longest time.

1 _ _ _ // Page fault, insert 1
1 7 _ _ // Page fault, insert 7
1 7 6 _ // Page fault, insert 6
1 7 6 5 // Page fault, insert 5
4 7 6 5 // Page fault, replace 1 with 4
4 3 6 5 // Page fault, replace 7 with 3
4 3 2 5 // Page fault, replace 6 with 2
4 3 2 1 // Page fault, replace 5 with 1
4 3 2 1 // 2 already in memory
4 3 2 1 // 3 already in memory
4 2 5 1 // Page fault, replace 3 with 5
4 2 3 1 // Page fault, replace 5 with 3
4 2 3 1 // 4 already in memory
2 7 3 1 // Page fault, replace 4 with 7
2 7 3 1 // 1 already in memory
2 3 6 1 // Page fault, replace 7 with 6
2 3 6 1 // 2 already in memory
2 3 6 1 // 3 already in memory
7 3 6 2 // Page fault, replace 1 with 2
The total number of page faults is 16.

iii. Optimal replacement:

We will replace the page that will not be used for the longest time in the future.

1 _ _ _ // Page fault, insert 1
1 7 _ _ // Page fault, insert 7
1 7 6 _ // Page fault, insert 6
1 7 6 5 // Page fault, insert 5
4 7 6 5 // Page fault, replace 1 with 4
4 3 6 5 // Page fault, replace 7 with 3
4 3 2 5 // Page fault, replace 6 with 2
4 3 2 1 // Page fault, replace 5 with 1
4 3 2 1 // 2 already in memory
4 3 2 1 // 3 already in memory
4 2 5 1 // Page fault, replace 3 with 5
4 2 3 1 // Page fault, replace 5 with 3
4 2 3 1 // 4 already in memory
2 7 3 1 // Page fault, replace 4 with 7
2 7 3 1 // 1 already in memory
2 3 6 1 // Page fault, replace 7 with 6
2 3 6 1 // 2 already in memory
2 3 6 1 // 3 already in memory
7 3 6 2 // Page fault, replace 1 with 2
The total number of page faults is 10.

Therefore, the number of page faults for the three replacement algorithms are as follows:

FIFO replacement: 14

LRU replacement: 16
Optimal replacement: 10

Question 4: (10 Marks) Write a program in C to implement Banker's Algorithm to avoid Deadlock. Also explain the code briefly.
Ans.

```
#include <stdio.h>

#define MAX_PROCESS 10
#define MAX_RESOURCES 10

int allocation[MAX_PROCESS][MAX_RESOURCES];
int max[MAX_PROCESS][MAX_RESOURCES];
int available[MAX_RESOURCES];
int need[MAX_PROCESS][MAX_RESOURCES];
int work[MAX_RESOURCES];
int finish[MAX_PROCESS];
int num_processes, num_resources;

int is_safe() {
   int i, j, k;
   int count = 0;
   for (i = 0; i < num_processes; i++)
      finish[i] = 0;

   for (i = 0; i < num_resources; i++)
      work[i] = available[i];

   while (count < num_processes) {
      int found = 0;
      for (i = 0; i < num_processes; i++) {
         if (finish[i] == 0) {
            int possible = 1;
            for (j = 0; j < num_resources; j++) {
               if (need[i][j] > work[j]) {
                  possible = 0;
                  break;
               }
            }
            if (possible) {
               found = 1;
               finish[i] = 1;
               count++;
               for (k = 0; k < num_resources; k++) {
                  work[k] += allocation[i][k];
               }
            }
         }
      }
      if (found == 0)
         return 0;
   }
   return 1;
}

int main() {
   int i, j;
   printf("Enter number of processes: ");
   scanf("%d", &num_processes);
   printf("Enter number of resources: ");
   scanf("%d", &num_resources);

   printf("Enter allocation matrix:\n");
   for (i = 0; i < num_processes; i++) {
      for (j = 0; j < num_resources; j++) {
         scanf("%d", &allocation[i][j]);
      }
```

```
}

printf("Enter maximum matrix:\n");
for (i = 0; i < num_processes; i++) {
   for (j = 0; j < num_resources; j++) {
      scanf("%d", &max[i][j]);
      need[i][j] = max[i][j] - allocation[i][j];
   }
}

printf("Enter available resources:\n");
for (i = 0; i < num_resources; i++) {
   scanf("%d", &available[i]);
}

if (is_safe())
   printf("System is in safe state.\n");
else
   printf("System is in unsafe state.\n");

   return 0;
}
```

This program implements the Banker's algorithm to avoid deadlock. The program prompts the user to enter the number of processes and resources, the allocation matrix, the maximum matrix, and the available resources. It then checks whether the system is in a safe state or an unsafe state.

The algorithm works by simulating the execution of each process and checking whether the system will be deadlocked or not. It does this by keeping track of the resources allocated to each process, the maximum number of resources each process needs, and the available resources. It then checks whether the system is in a safe state by examining whether there is at least one process that can complete its execution without causing a deadlock. If there is such a process, the algorithm simulates its execution and updates the available resources. If there is no such process, the algorithm concludes that the system is in an unsafe state.

Question 5: (10 Marks) Discuss in detail the I/O management, File management and Security and Protection in WINDOWS 11 Operating System.

Ans.

I/O Management in Windows 11:

I/O management in Windows 11 is responsible for managing input and output operations to and from hardware devices, such as keyboards, mice, disks, and printers. Windows 11 uses a variety of I/O models to manage input and output, including polling, interrupts, and Direct Memory Access (DMA).

Windows 11 provides a Device Manager that allows users to manage hardware devices, including drivers and driver updates. The Device Manager also provides detailed information about each device and its drivers, and allows users to disable, enable, uninstall, or roll back device drivers.

Windows 11 also provides a Plug and Play (PnP) system that automatically detects and configures new hardware devices as they are connected to the system. This allows users to add and remove hardware devices without having to manually configure the system.

File Management in Windows 11:

File management in Windows 11 is responsible for managing files and directories on disk drives. Windows 11 uses a hierarchical file system, where directories can contain files and other directories. Windows 11 also provides a set of APIs for managing files and directories, including creating, opening, reading, writing, deleting, and renaming files and directories.

Windows 11 also provides a search feature that allows users to search for files and directories based on a variety of criteria, including name, size, date modified, and file type.

Security and Protection in Windows 11:

Security and protection in Windows 11 is an important aspect of the operating system. Windows 11 provides a variety of security features, including:

1. User account control (UAC) - this feature prompts the user for permission before allowing an application to make changes to the system.
2. Windows Defender - this is an anti-malware and antivirus software that is built into Windows 11. It provides real-time protection against viruses, spyware, and other malware.
3. Windows Firewall - this is a built-in firewall that helps to protect the system from unauthorized access.
4. BitLocker - this is a built-in encryption feature that helps to protect data on disk drives.
5. Windows Hello - this is a biometric authentication feature that allows users to log in to their system using their fingerprint, face, or iris.
6. Virtualization-based security - this is a feature that uses hardware virtualization to protect the system from attacks, such as kernel-level exploits.

Overall, Windows 11 provides a comprehensive set of features for managing I/O operations, files and directories, and security and protection. These features make Windows 11 a robust and secure operating system for a variety of applications and environments.

Question 6: (10 Marks) Write about the App management, APIs, behaviour changes (privacy, security and performance), flash memory management and Battery Resource Utilization in Android 13 Mobile Operating System.

Ans.

App management in Android 13:

Android 13 provides users with powerful app management features, including the ability to view and manage app permissions, background processes, and battery usage. Users can also control the installation and updating of apps from the Google Play Store and other sources. Additionally, Android 13 includes features to optimize app performance, such as app startup time and resource usage.

APIs in Android 13:

Android 13 includes a rich set of APIs that developers can use to build high-quality apps. These APIs cover a wide range of functionality, including graphics and animation, multimedia, networking, location, and sensor data. Android 13 also includes new APIs that support advanced features like machine learning, augmented reality, and biometric authentication.

Behavior changes in Android 13:

Privacy, security, and performance are top priorities in Android 13. Some of the behavior changes in Android 13 include:

1. Enhanced privacy features to give users more control over their data, such as permission auto-reset, which automatically revokes permissions for unused apps.
2. Improved security features to protect users against malware and other threats. For example, Android 13 includes a new feature that makes it easier to identify and remove potentially harmful apps.
3. Performance improvements to make the operating system more responsive and efficient. Android 13 includes a new feature called Hibernation, which reduces the memory usage of unused apps.

Flash memory management in Android 13:

Android 13 includes new features to optimize the use of flash memory, which is used to store data and apps on the device. One of these features is called Scoped Storage, which restricts access to files and data to specific apps. This helps to prevent unauthorized access and ensures that apps can only access the data that they need.

Battery resource utilization in Android 13:

Android 13 includes features to help users manage battery life more effectively. One of these features is called Battery Saver, which reduces the battery usage of the device by limiting background processes and network activity. Another feature is Adaptive Battery, which uses machine learning to predict which apps will be used in the near future and optimizes the battery usage of the device accordingly.

Overall, Android 13 provides users with a powerful and flexible mobile operating system that supports a wide range of functionality and use cases. Its robust app management, APIs, behavior changes, flash memory management, and battery resource utilization features make it a compelling choice for both consumers and developers.