Operating System

CA 02

Name: Ajay Ashokrao Deshmukh
Div: D10B
Roll no: 12 **DOS:28-03-2024**

Module 4 - Write a c program to implement Paging technique for memory management.

→ **Code -**

```
#include <stdio.h>

#define MEMORY_SIZE 1000 // Size of memory in bytes
#define PAGE_SIZE 4096 // Size of each page in bytes

// Function to simulate paging technique for memory management
void paging(int processSize) {
   int numPagesRequired = (processSize + PAGE_SIZE - 1) / PAGE_SIZE;
   printf("Number of pages required for process of size %d bytes: %d\n", processSize,
numPagesRequired);
}

int main() {
   int processSize;
   printf("Enter size of process: ");
   scanf("%d", &processSize);
   paging(processSize);
   return 0;
}
```

Output -

```
Enter size of process: 5
Number of pages required for process of size 5 bytes: 1


=== Code Execution Successful ===
```

```
Enter size of process: 5131332
Number of pages required for process of size 5131332 bytes: 1253


=== Code Execution Successful ===
```

Module 5 - Implement various disk scheduling algorithms like SCAN, and C-SCAN in C/Python/Java.

→ **Code -**

```
#include <stdio.h>
#include <stdlib.h>

// Function to implement SCAN disk scheduling algorithm
void scan(int arr[], int head, int size) {
   int totalSeekTime = 0;
   int distance, cur_track;
   char direction;

   printf("Enter direction (left/l or right/r): ");
   scanf(" %c", &direction);

   if (direction == 'l' || direction == 'L') {
      direction = 'l';
   } else {
      direction = 'r';
   }

   printf("Seek Sequence: ");
   cur_track = head;

   if (direction == 'l') {
      for (int i = head; i >= 0; i--) {
         printf("%d ", i);
         distance = abs(cur_track - i);
         totalSeekTime += distance;
         cur_track = i;
      }
      totalSeekTime += head;
      cur_track = 0;
```

```
        for (int i = 0; i < size; i++) {
            printf("%d ", i);
            distance = abs(cur_track - i);
            totalSeekTime += distance;
            cur_track = i;
        }
    } else if (direction == 'r') {
        for (int i = head; i < size; i++) {
            printf("%d ", i);
            distance = abs(cur_track - i);
            totalSeekTime += distance;
            cur_track = i;
        }
        totalSeekTime += (size - head - 1);
        cur_track = size - 1;
        for (int i = size - 1; i >= 0; i--) {
            printf("%d ", i);
            distance = abs(cur_track - i);
            totalSeekTime += distance;
            cur_track = i;
        }
    }

    printf("\nTotal Seek Time: %d\n", totalSeekTime);
}

// Function to implement C-SCAN disk scheduling algorithm
void cscan(int arr[], int head, int size) {
    int totalSeekTime = 0;
    int distance, cur_track;

    printf("Seek Sequence: ");

    // Sort the disk requests in ascending order
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    // Find the index where head is located in sorted array
    int index = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] >= head) {
            index = i;
            break;
        }
```

```
    }

    // Traverse right from head
    for (int i = index; i < size; i++) {
        printf("%d ", arr[i]);
        distance = abs(cur_track - arr[i]);
        totalSeekTime += distance;
        cur_track = arr[i];
    }

    // Go to the first track (0) if reached end
    if (index == size) {
        printf("0 ");
        totalSeekTime += cur_track;
        cur_track = 0;
    }

    // Traverse left from 0 to the head
    for (int i = 0; i < index; i++) {
        printf("%d ", arr[i]);
        distance = abs(cur_track - arr[i]);
        totalSeekTime += distance;
        cur_track = arr[i];
    }

    printf("\nTotal Seek Time: %d\n", totalSeekTime);
}

int main() {
    int requests[] = {98, 183, 37, 122, 14, 124, 65, 67};
    int head, size;

    size = sizeof(requests) / sizeof(requests[0]);

    printf("Enter initial position of head: ");
    scanf("%d", &head);

    printf("\nSCAN Algorithm:\n");
    scan(requests, head, size);

    printf("\nC-SCAN Algorithm:\n");
    cscan(requests, head, size);

    return 0;
}
```

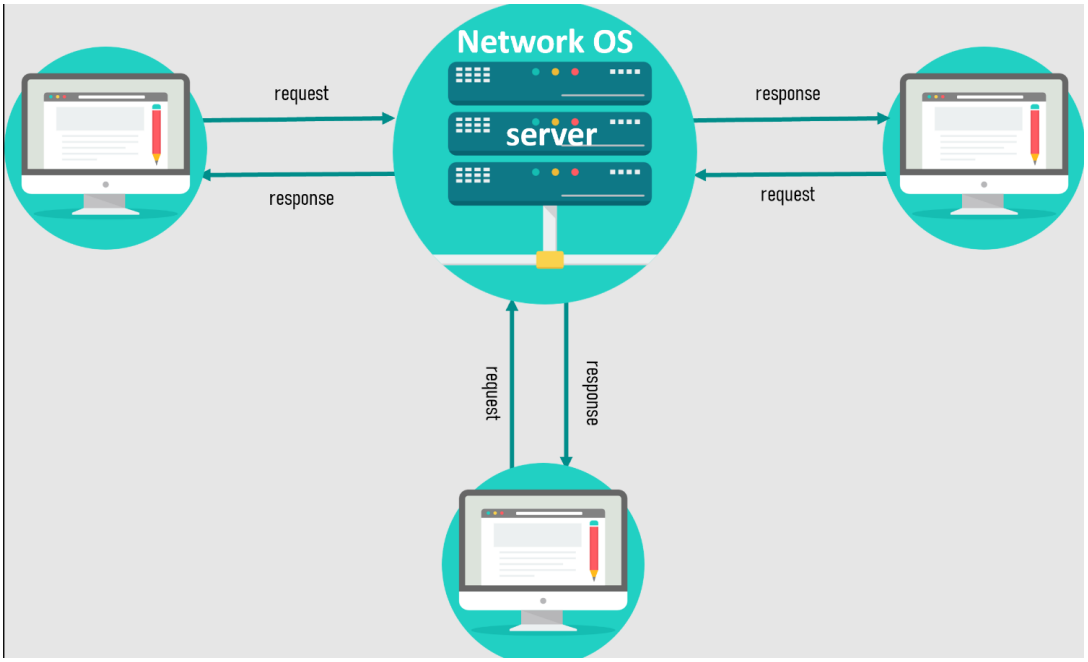
Output:

```
Enter initial position of head: 4
SCAN Algorithm:
Enter direction (left/l or right/r): l
Seek Sequence: 4 3 2 1 0 0 1 2 3 4 5 6 7
Total Seek Time: 15

C-SCAN Algorithm:
Seek Sequence: 14 37 65 67 98 122 124 183
Total Seek Time: 170
```

Module 6 - Case Study on Network Operating System (Mac OS Server).

→



Introduction:
macOS Server, formerly known as Mac OS X Server, is Apple's server operating system built on top of macOS. It provides a range of services for networking, file sharing, device management, and collaboration. This case study explores the use of macOS Server in a medium-sized company named "TechSolutions" to manage their network infrastructure and improve productivity.

Background:

A medium-sized organization with diverse departments and remote teams is looking to streamline its network management, enhance collaboration, and improve data security. They decide to implement macOS Server as their network operating system to address these challenges.

Key Challenges:

- Centralized File Storage: The organization needs a centralized file storage solution accessible to all employees, allowing seamless file sharing and collaboration.
- User and Device Management: Efficient management of user accounts, permissions, and device configurations is essential to ensure security and productivity.
- Email and Communication: An integrated email and messaging platform is required for internal communication, scheduling meetings, and sharing updates.
- Backup and Recovery: Robust backup and disaster recovery mechanisms are needed to safeguard critical data and minimize downtime in case of system failures.

Implementation of macOS Server:

- File Sharing and Collaboration:
  - macOS Server's File Sharing service is configured to create shared folders with granular access controls based on user roles and departments.
  - Collaboration tools such as Pages, Numbers, and Keynote are integrated, enabling real-time document editing and version history tracking.
  - Time Machine backup is utilized to automatically back up files on macOS and iOS devices, ensuring data protection and easy recovery.
- User and Device Management:
  - Open Directory service in macOS Server is implemented for centralized user authentication, account management, and access permissions.
  - Profile Manager is used for mobile device management (MDM), allowing administrators to manage macOS and iOS devices, enforce security policies, and distribute apps and settings remotely.
- Email and Communication Services:
  - macOS Server's Mail service is set up to provide secure email hosting with features like spam filtering, encryption, and email aliases.
  - The Messages service is configured for instant messaging and communication within the organization, supporting group chats and file sharing.
- Backup and Disaster Recovery:
  - Time Machine backups are scheduled for regular backups of server data and configurations, ensuring data integrity and quick recovery in case of data loss.
  - Additional backup solutions, such as cloud backups or external backups, may be implemented for off-site data storage and disaster recovery planning.

Benefits and Outcomes:

- Enhanced Collaboration: macOS Server facilitates seamless file sharing and collaboration, improving teamwork and productivity across departments and remote teams.

- Efficient Management: Centralized user and device management simplifies user onboarding, access control, and device configuration, reducing IT administrative overhead.
- Secure Communication: Integrated email, messaging, and security features ensure secure communication and data protection, meeting compliance and privacy requirements.
- Reliable Backup and Recovery: Robust backup solutions, including Time Machine backups and off-site backups, ensure data integrity and minimize downtime risks, enhancing business continuity.

Conclusion:

Implementing macOS Server as a network operating system offers significant advantages in network management, collaboration, user management, and data security. The organization benefits from a centralized and efficient network infrastructure that supports productivity, communication, and data protection.