**OS - Practical 9**

**Name: Rishabh Jain**

**Branch & Sem: CSE - IV**

**Sec: A**

**Roll no.:54**

**Aim:** Write C programs to implement different disk scheduling algorithms, page replacement algorithms and to demonstrate different memory management

schemes.

**9A-Memory Management Scheme**

**Worst Fit**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_BLOCKS 6

#define MAX\_PROCESSES 10

void worstFit(int mem\_partition[], int process[], int blocks, int p)

{

    int IF = 0, EF = 0;

    int occupy[MAX\_BLOCKS], allocation[MAX\_PROCESSES];

    for (int i = 0; i < p; i++)

    {

        allocation[i] = -1;

    }

    for (int i = 0; i < blocks; i++)

    {

        occupy[i] = 0;

    }

    for (int i = 0; i < p; i++)

    {

        int worstBlockIdx = -1;

        for (int j = 0; j < blocks; j++)

        {

            if (mem\_partition[j] >= process[i] && !occupy[j])

            {

                if (worstBlockIdx == -1 || mem\_partition[j] > mem\_partition[worstBlockIdx])

                {

                    worstBlockIdx = j;

                }

            }

        }

        if (worstBlockIdx != -1)

        {

            allocation[i] = worstBlockIdx;

            occupy[worstBlockIdx] = 1;

            mem\_partition[worstBlockIdx] -= process[i];

            IF += process[i];

        }

    }

    for (int i = 0; i < blocks; i++)

    {

        if (!occupy[i])

        {

            EF += mem\_partition[i];

        }

    }

    printf("\nProcess No.\tProcess Size\tBlock no.\n");

    for (int i = 0; i < p; i++)

    {

        printf("%d \t\t\t %d \t\t\t", i + 1, process[i]);

        if (allocation[i] != -1)

            printf("%d\n", allocation[i] + 1);

        else

            printf("Not Allocated\n");

    }

    printf("\n");

    printf("Internal Fragmentation is : %d\n", IF);

    printf("External Fragmentation is : %d\n", EF);

}

int main()

{

    int mem\_partition[MAX\_BLOCKS];

    int p;

    printf("Enter total Processes: ");

    scanf("%d", &p);

    int process[MAX\_PROCESSES];

    printf("Enter Partitions:\n");

    for (int i = 0; i < MAX\_BLOCKS; i++)

    {

        printf("Enter Partition %d: ", i + 1);

        scanf("%d", &mem\_partition[i]);

    }

    printf("Enter Process Size:\n");

    for (int i = 0; i < p; i++)

    {

        printf("Enter Process Size %d: ", i + 1);

        scanf("%d", &process[i]);

    }

    printf("\n");

    int blocks = MAX\_BLOCKS;

    worstFit(mem\_partition, process, blocks, p);

}

**OUTPUT:**

****

**9B-Page Replacement Algorithms**

**LRU**

#include <stdio.h>

#define MAX\_REFERENCES 100

#define MAX\_FRAMES 3

int main()

{

    int r, a;

    int pageFaults = 0;

    printf("Enter total references: ");

    scanf("%d", &r);

    if (r <= 0 || r > MAX\_REFERENCES)

    {

        printf("Invalid number of references.\n");

        return 1;

    }

    int s[MAX\_REFERENCES];

    printf("Enter reference String:\n");

    for (int i = 0; i < r; i++)

    {

        scanf("%d", &s[i]);

    }

    printf("We have 3 frames.\n");

    printf("Reference String\tFrame 1\t\tFrame 2\t\tFrame 3\n");

    int temp[MAX\_FRAMES] = {-1};

    int lastUsed[MAX\_FRAMES] = {0};

    for (int i = 0; i < r; i++)

    {

        a = 0;

        for (int j = 0; j < MAX\_FRAMES; j++)

        {

            if (s[i] == temp[j])

            {

                a++;

                pageFaults--;

                lastUsed[j] = i;

            }

        }

        pageFaults++;

        if ((pageFaults <= MAX\_FRAMES) && (a == 0))

        {

            temp[i] = s[i];

            lastUsed[i] = i;

        }

        else if (a == 0)

        {

            int lruIndex = 0;

            int lruValue = lastUsed[0];

            for (int j = 1; j < MAX\_FRAMES; j++)

            {

                if (lastUsed[j] < lruValue)

                {

                    lruIndex = j;

                    lruValue = lastUsed[j];

                }

            }

            temp[lruIndex] = s[i];

            lastUsed[lruIndex] = i;

        }

        printf("%d\t\t\t", s[i]);

        for (int j = 0; j < MAX\_FRAMES; j++)

        {

            if (temp[j] != -1)

                printf("%d\t\t", temp[j]);

            else

                printf("-\t\t");

        }

        printf("\n");

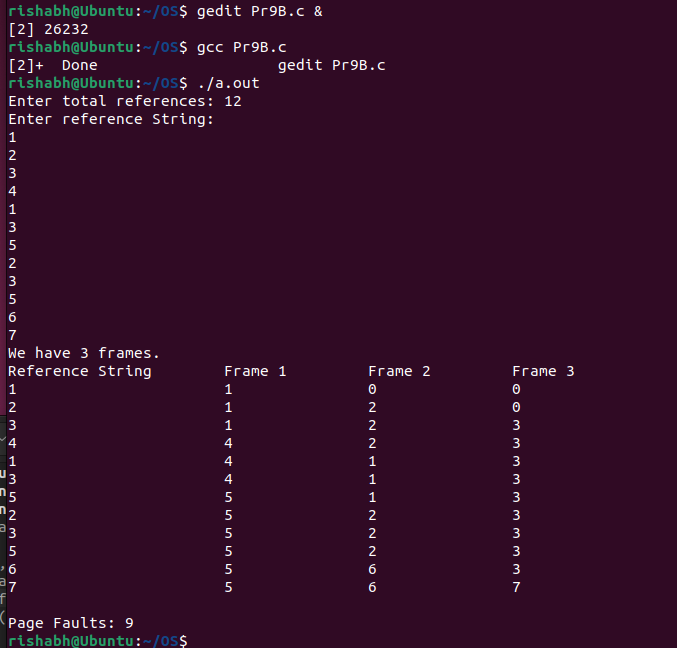
    }

    printf("\nPage Faults: %d\n", pageFaults);

    return 0;

}

**OUTPUT:**

****

**9C-Disk Scheduling Algorithms**

**CSCAN**

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int i, n, req[50], mov = 0;

    int current\_position, index[50], min;

    int a[50], j = 0, mini, current\_position1;

    printf("Enter the number of requests: ");

    scanf("%d", &n);

    printf("Enter the current position: ");

    scanf("%d", &current\_position);

    current\_position1 = current\_position;

    printf("Enter the request order:\n");

    for (i = 0; i < n; i++)

    {

        scanf("%d", &req[i]);

    }

    for (i = 0; i < n - 1; i++)

    {

        for (int j = 0; j < n - i - 1; j++)

        {

            if (req[j] > req[j + 1])

            {

                int temp = req[j];

                req[j] = req[j + 1];

                req[j + 1] = temp;

            }

        }

    }

    int currentPositionIndex = 0;

    for (i = 0; i < n; i++)

    {

        if (current\_position <= req[i])

        {

            currentPositionIndex = i;

            break;

        }

    }

    for (i = currentPositionIndex; i < n; i++)

    {

        a[j] = req[i];

        j++;

    }

    for (i = 0; i < currentPositionIndex; i++)

    {

        a[j] = req[i];

        j++;

    }

    printf("Sequence is: ");

    printf("%d", current\_position1);

    mov = mov + abs(current\_position1 - a[0]);

    printf(" -> %d", a[0]);

    for (i = 1; i < n; i++)

    {

        mov = mov + abs(a[i] - a[i - 1]);

        printf(" -> %d", a[i]);

    }

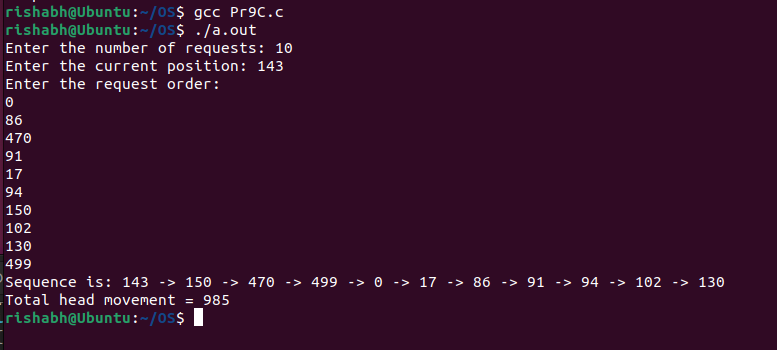
    printf("\n");

    printf("Total head movement = %d\n", mov);

    return 0;

}

**OUTPUT:**

****

**Result:** We LearnedC programs to implement disk scheduling algorithms, page replacement algorithms and to demonstrate different memory management schemes have been implemented.