21BDS0340

Abhinav Dinesh Srivatsa

Operating Systems Lab

Assignment – V

Question 1

Program: FIFO

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

void fifoPageReplacement(int pages[], int numPages, int numFrames)
{
    int frame[numFrames];
    int i, j;
    int pageFaults = 0;
    int currentIndex = 0;
    int exists = 0;

    for (i = 0; i < numFrames; i++)
    {
        frame[i] = -1;
    }

    for (i = 0; i < numPages; i++)
    {
        exists = 0;
        for (j = 0; j < numFrames; j++)
        {
            if (frame[j] == pages[i])
            {
                exists = 1;
                break;
            }
        }

        if (exists == 0)
        {
            frame[currentIndex] = pages[i];
            currentIndex = (currentIndex + 1) % numFrames;
            pageFaults++;
        }

        printf("\nPage %d: ", pages[i]);
        for (j = 0; j < numFrames; j++)
        {
            printf("%d ", frame[j]);
```

```
        }
    }

    printf("\n\nTotal Page Faults: %d\n", pageFaults);
}

int main()
{
    int numPages, numFrames, i;

    printf("Enter the number of pages: ");
    scanf("%d", &numPages);

    printf("Enter the number of frames: ");
    scanf("%d", &numFrames);

    int pages[numPages];

    printf("Enter the page sequence:\n");
    for (i = 0; i < numPages; i++)
    {
        printf("Page %d: ", i + 1);
        scanf("%d", &pages[i]);
    }

    printf("\nFIFO Page Replacement Algorithm:\n");
    fifoPageReplacement(pages, numPages, numFrames);

    return 0;
}
```

Output:

```
Enter the number of pages: 10
Enter the number of frames: 4
Enter the page sequence:
Page 1: 1
Page 2: 2
Page 3: 3
Page 4: 4
Page 5: 5
Page 6: 1
Page 7: 2
Page 8: 3
Page 9: 4
Page 10: 1

FIFO Page Replacement Algorithm:

Page 1: 1 -1 -1 -1
Page 2: 1 2 -1 -1
Page 3: 1 2 3 -1
Page 4: 1 2 3 4
Page 5: 5 2 3 4
Page 1: 5 1 3 4
Page 2: 5 1 2 4
Page 3: 5 1 2 3
Page 4: 4 1 2 3
Page 1: 4 1 2 3

Total Page Faults: 9
```

Program: LRU

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

void lruPageReplacement(int pages[], int numPages, int numFrames)
{
    int frame[numFrames];
    int i, j, k;
    int pageFaults = 0;
    int currentIndex = 0;
    int exists = 0;
    int *counters = (int *)malloc(numFrames * sizeof(int));

    for (i = 0; i < numFrames; i++)
    {
        frame[i] = -1;
        counters[i] = 0;
    }
```

```
    for (i = 0; i < numPages; i++)
    {
        exists = 0;
        for (j = 0; j < numFrames; j++)
        {
            if (frame[j] == pages[i])
            {
                exists = 1;
                counters[j] = i + 1;
                break;
            }
        }

        if (exists == 0)
        {
            int minIndex = 0;
            for (k = 1; k < numFrames; k++)
            {
                if (counters[k] < counters[minIndex])
                {
                    minIndex = k;
                }
            }

            frame[minIndex] = pages[i];
            counters[minIndex] = i + 1;
            pageFaults++;
        }

        printf("\nPage %d: ", pages[i]);
        for (j = 0; j < numFrames; j++)
        {
            printf("%d ", frame[j]);
        }
    }

    printf("\n\nTotal Page Faults: %d\n", pageFaults);
    free(counters);
}

int main()
{
    int numPages, numFrames, i;

    printf("Enter the number of pages: ");
    scanf("%d", &numPages);

    printf("Enter the number of frames: ");
    scanf("%d", &numFrames);
```

```
    int pages[numPages];

    printf("Enter the page sequence:\n");
    for (i = 0; i < numPages; i++)
    {
        printf("Page %d: ", i + 1);
        scanf("%d", &pages[i]);
    }

    printf("\nLRU Page Replacement Algorithm:\n");
    lruPageReplacement(pages, numPages, numFrames);

    return 0;
}
```

Output:

```
Enter the number of pages: 10
Enter the number of frames: 4
Enter the page sequence:
Page 1: 1
Page 2: 2
Page 3: 3
Page 4: 4
Page 5: 5
Page 6: 1
Page 7: 2
Page 8: 3
Page 9: 4
Page 10: 1

LRU Page Replacement Algorithm:

Page 1: 1 -1 -1 -1
Page 2: 1 2 -1 -1
Page 3: 1 2 3 -1
Page 4: 1 2 3 4
Page 5: 5 2 3 4
Page 1: 5 1 3 4
Page 2: 5 1 2 4
Page 3: 5 1 2 3
Page 4: 4 1 2 3
Page 1: 4 1 2 3

Total Page Faults: 9
```

Program: Optimal

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

int findOptimal(int pages[], int numPages, int frame[], int numFrames, int
startIndex)
{
    int index = -1;
    int farthest = startIndex;

    for (int i = 0; i < numFrames; i++)
    {
        int j;
        for (j = startIndex; j < numPages; j++)
        {
            if (frame[i] == pages[j])
            {
                if (j > farthest)
                {
                    farthest = j;
                    index = i;
                }
                break;
            }
        }

        if (j == numPages)
            return i;
    }

    return (index == -1) ? 0 : index;
}

void optimalPageReplacement(int pages[], int numPages, int numFrames)
{
    int frame[numFrames];
    bool isPresent[numFrames];
    int pageFaults = 0;

    for (int i = 0; i < numFrames; i++)
    {
        frame[i] = -1;
        isPresent[i] = false;
    }

    for (int i = 0; i < numPages; i++)
    {
        int j;
        bool isFull = true;
```

```
        for (j = 0; j < numFrames; j++)
        {
            if (frame[j] == pages[i])
            {
                isPresent[j] = true;
                break;
            }
        }

        if (j == numFrames)
        {
            int k;
            for (k = 0; k < numFrames; k++)
            {
                if (!isPresent[k])
                {
                    frame[k] = pages[i];
                    isPresent[k] = true;
                    pageFaults++;
                    isFull = false;
                    break;
                }
            }

            if (k == numFrames)
            {
                int index = findOptimal(pages, numPages, frame, numFrames, i + 1);
                frame[index] = pages[i];
                pageFaults++;
            }
        }

        printf("\nPage %d: ", pages[i]);
        for (int k = 0; k < numFrames; k++)
        {
            printf("%d ", frame[k]);
        }
    }

    printf("\n\nTotal Page Faults: %d\n", pageFaults);
}

int main()
{
    int numPages, numFrames, i;

    printf("Enter the number of pages: ");
    scanf("%d", &numPages);

    printf("Enter the number of frames: ");
    scanf("%d", &numFrames);
```

```c
    int pages[numPages];

    printf("Enter the page sequence:\n");
    for (i = 0; i < numPages; i++)
    {
        printf("Page %d: ", i + 1);
        scanf("%d", &pages[i]);
    }

    printf("\nOptimal Page Replacement Algorithm:\n");
    optimalPageReplacement(pages, numPages, numFrames);

    return 0;
}
```

Output:

```
Enter the number of pages: 10
Enter the number of frames: 4
Enter the page sequence:
Page 1: 1
Page 2: 2
Page 3: 3
Page 4: 4
Page 5: 5
Page 6: 1
Page 7: 2
Page 8: 3
Page 9: 4
Page 10: 1

Optimal Page Replacement Algorithm:

Page 1: 1 -1 -1 -1
Page 2: 1 2 -1 -1
Page 3: 1 2 3 -1
Page 4: 1 2 3 4
Page 5: 1 2 3 5
Page 1: 1 2 3 5
Page 2: 1 2 3 5
Page 3: 1 2 3 5
Page 4: 1 4 3 5
Page 1: 1 4 3 5

Total Page Faults: 6
```

Question 2

Program:

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

void lockFile(const char *filename)
{
    int fileDescriptor = open(filename, O_WRONLY);

    if (fileDescriptor == -1)
    {
        perror("Error opening file");
        return;
    }

    struct flock fl;
    memset(&fl, 0, sizeof(fl));
    fl.l_type = F_WRLCK; // Write lock
    fl.l_whence = SEEK_SET;
    fl.l_start = 0;
    fl.l_len = 0; // Lock entire file

    if (fcntl(fileDescriptor, F_SETLK, &fl) == -1)
    {
        perror("Error locking file");
        close(fileDescriptor);
        return;
    }

    printf("File locked successfully.\n");

    // Simulating a locked file by pausing execution for a few seconds
    sleep(5);

    fl.l_type = F_UNLCK; // Unlock
    if (fcntl(fileDescriptor, F_SETLK, &fl) == -1)
    {
        perror("Error unlocking file");
    }
    else
    {
        printf("File unlocked successfully.\n");
    }

    close(fileDescriptor);
}
```

```
int main()
{
    const char *filename = "test.txt";

    lockFile(filename);

    return 0;
}
```

Output:

```
File locked successfully.
File unlocked successfully.
```