Operating System Lab 10

Q1> Implement FIFO and LRU page-replacement algorithms

Write a program that implements the FIFO and LRU page-replacement algorithms.

First, generate a random page reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm.

Implement the replacement algorithms so that the number of page frames can vary from 1 to 7. Assume that demand paging is used.

example->

```
page reference string is :

1 4 5 6 5 1 8 4 2 4 4 2

please enter page frame :

1

4

5

6

5

1

8

4

2

4

4

2


FIFO page fault = 11
```

1

4

5

6

5

1

8

4

2

4

4

2

LRU page fault = 11

Q2> The Catalan numbers are an integer sequence Cn that appear in tree enumeration problems. The first Catalan numbers for n = 1, 2, 3,...are 1, 2, 5, 14, 42, 132, ... A formula generating Cn is Cn = 1/(n+1) (2n/n) = (2n)!/(n+1)!n! Design two programs that communicate with shared memory using the Win32 API as outlined in Section 9.7.2. The producer process will generate the Catalan sequence and write it to a shared memory object. The consumer process will then read and output the sequence from shared memory. In this instance, the producer process will be passed an integer parameter on the command line specifying the number of Catalan numbers to produce, i.e. providing 5 on the command line means the producer process will generate the first 5 Catalan numbers.

example->

input

5

output

Catalan Numbers:

1 1 2 5 14

Libraries used

#include<time.h>

#include <Windows.h>

Functions used

srand(time(NULL));

The expression srand(time(NULL)); is commonly used in C and C++ programming to seed the random number generator (rand() function) with a value based on the current time. Here's a breakdown of what each part of this expression does:

time(NULL): The time() function in C/C++ is used to get the current time in seconds since the epoch (00:00:00 Coordinated Universal Time (UTC), January 1, 1970). The NULL argument means it doesn't require a pointer to a time_t object; it returns the time directly.

srand(): This function is used to seed the random number generator. The srand() function sets the starting point for producing a series of pseudo-random integers by rand().

Combining srand(time(NULL));: By passing the current time as a seed to srand(), you ensure that each time your program runs, it starts with a different seed based on the current time. This helps in generating a more varied sequence of random numbers, making the sequence less predictable.

CreateFileMapping:

Purpose: Creates or opens a named or unnamed file-mapping object.

Syntax:

```
HANDLE CreateFileMapping(

    HANDLE hFile,

    LPSECURITY_ATTRIBUTES lpAttributes,

    DWORD flProtect,

    DWORD dwMaximumSizeHigh,

    DWORD dwMaximumSizeLow,

    LPCTSTR lpName
```

);

Parameters:

hFile: A handle to the file from which to create a mapping.

lpAttributes: A pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle can be inherited by child processes.

flProtect: Specifies the protection desired for the file view.

dwMaximumSizeHigh and dwMaximumSizeLow: The high and low-order DWORD values of the maximum size of the file mapping object.

lpName: The name of the mapping object.

GetLastError:

Purpose: Retrieves the calling thread's last-error code value.

Syntax:

DWORD GetLastError();

Return Value: The return value is the calling thread's last-error code value.

MapViewOfFile:

Purpose: Maps a view of a file mapping into the address space of a calling process.

Syntax:

```
LPVOID MapViewOfFile(

    HANDLE hFileMappingObject,

    DWORD dwDesiredAccess,

    DWORD dwFileOffsetHigh,

    DWORD dwFileOffsetLow,

    SIZE_T dwNumberOfBytesToMap

);
```

Parameters:

hFileMappingObject: A handle to a file mapping object created by CreateFileMapping.

dwDesiredAccess: The type of access to the file view.

dwFileOffsetHigh and dwFileOffsetLow: The high and low-order DWORD values of the file offset where mapping should begin.

dwNumberOfBytesToMap: The number of bytes of a file mapping to map to the view.

UnmapViewOfFile:

Purpose: Unmaps a mapped view of a file from the calling process's address space.

Syntax:

```
BOOL UnmapViewOfFile(
        LPCVOID lpBaseAddress
);
```

Parameters:

lpBaseAddress: A pointer to the base address of the mapped view of a file that is to be unmapped.

CloseHandle:

Purpose: Closes an open object handle.

Syntax:

```
BOOL CloseHandle(
        HANDLE hObject
);
```

Parameters:

hObject: A handle to an open object.

OpenFileMapping:

Purpose: Opens a named file mapping object.

Syntax:

```
HANDLE OpenFileMapping(
    DWORD dwDesiredAccess,
    BOOL bInheritHandle,
    LPCTSTR lpName
);
```

Parameters:

dwDesiredAccess: The access to the file mapping object.

bInheritHandle: If this parameter is TRUE, a process created by the CreateProcess function can inherit the handle.

lpName: The name of the file mapping object to be opened.

These functions are commonly used for interprocess communication in Windows programming, particularly when multiple processes need to share data using shared memory.