Name: Trần Hoàng Bảo Ly

ID: 21521109

Class: IT007.N11.KHTN

OPERATING SYSTEM
LAB 6'S REPORT

SUMMARY

Task		Status	Page
Section 6.4	Bài 6.4	Done	3
Section 6.5	Bài 6.5.1	Done	10
	Bài 6.5.2	Done	12

Self-scrores: 10

Note: Export file to **PDF and name the file by following format:*
LAB X – <Student ID>.pdf

Mục lục

6.4 Hướng dẫn thực hành

6.4.1.1. Soạn thảo chương trình FIFO-OTP-LRU.cpp

```cpp
#include <iostream>
#include <limits>
using namespace std;
int n = 11;
int nOpf = 3;
int NewArr[100] = {2,1,5,2,1,1,0,9,0,0,7};
int PageFrames[100][100];
//This array to log the time page access to pageframes
int PageFrameExtend[100];
//This array to sign page fault symbol, use for print.
char PageFaultRowExtend[100];
int PageFaultCount = 0;
int FIFOPointer = 0;
// This function return true if a page are in pageframe and update extend(for LRU
algorithm)
// or return false  otherwise
bool checkPageInPageFrame(int pos){
    for (size_t i = 0; i < nOpf; i++){
        if (PageFrames[i][pos] == NewArr[pos]){
            PageFrameExtend[i] = pos;
            return true;
        }
    }
    return false;
}
//This function-just for LRU  algorithm return true and push page to void space
//and update extend if there a void space
//or return false otherwise
bool IsHasVoidSpaceLRU(int pos){
    for (int j = 0; j < nOpf; j++){
        if (PageFrames[j][pos] == -1){
            PageFrames[j][pos] = NewArr[pos];
            PageFrameExtend[j] = pos;
            return true;
        }
    }
    return false;
}
//This function return true and push page to void space if there a void space
// return false otherwise
bool IsHasVoidSpace(int pos){
    for (int j = 0; j < nOpf; j++){
        if (PageFrames[j][pos] == -1){
```

```
            PageFrames[j][pos] = NewArr[pos];
            return true;
        }
    }
    return false;
}
//this funcion wil choose a victim to swap out and swap in new page use LRU
algorithm
void SwapVictimPageLRU(int pos){
    int victimPos = 0;
    int earlyIn = PageFrameExtend[0];
    //find page earliest in pageframe
    for (int i = 0; i < nOpf; i++){
        if (PageFrameExtend[i] < earlyIn){
            victimPos = i;
            earlyIn = PageFrameExtend[i];
        }
    }
    PageFrames[victimPos][pos] = NewArr[pos];
    PageFrameExtend[victimPos] = pos;
}
//This funcion to copy current state of page frame to new state of pageframe
void CopytoNewPos(int pos){
    for (size_t i = 0; i < nOpf; i++)
        PageFrames[i][pos] = PageFrames[i][pos-1];
}
void LRUAlgorithmStart(){
    for (size_t i = 0; i < n; i++){
        //If not start page, copy from old state of page frame to new state of
page frame
        if (i > 0)
            CopytoNewPos(i);
        //If page is in frame, just continnue
        if (checkPageInPageFrame(i)){
            PageFaultRowExtend[i] = ' ';
        //otherwise
        } else {
            PageFaultRowExtend[i] = '*';
            PageFaultCount++;
            //if there a void space in page frame, push coming page to it
            if (!IsHasVoidSpaceLRU(i))
            //otherwise choose a victim to swap out and swap in coming page
                SwapVictimPageLRU(i);
        }
    }
```

```
}
//this funcion wil choose a victim to swap out and swap in new page use FIFO
algorithm
void SwapVictimPageFIFO(int pos){
    //pointer start from zero(first)
    PageFrames[FIFOPointer][pos] = NewArr[pos];
    //when swap pointer wil be update to next pageframe (page comes secondly)
    FIFOPointer++;
    //cyclic pointer in [0,n-1]
    if (FIFOPointer >= nOpf) FIFOPointer = 0;
}
void FIFOAlgorithmStart(){
    for (size_t i = 0; i < n; i++){
        if (i > 0)
            CopytoNewPos(i);
        if (checkPageInPageFrame(i)){
            PageFaultRowExtend[i] = ' ';
        } else {
            PageFaultRowExtend[i] = '*';
            PageFaultCount++;
            if (!IsHasVoidSpace(i))
                SwapVictimPageFIFO(i);
        }
    }
}
//this funcion wil choose a victim to swap out and swap in new page use OTP
algorithm
void SwapVictimPageOTP(int pos){
    int remainCount = nOpf;
    int Map[nOpf] {0};
    // find first appear in future of pageframe in access string from page in
pageframes
    for (size_t i = pos+1; i < n; i++){
        for (int j = 0; j < nOpf; j++){
            if (NewArr[i] == PageFrames[j][pos] && Map[j] == 0){
                    remainCount--;
                    Map[j] = i;
            }
        }
        if (remainCount == 0) break;
    }
    int maxValue = Map[0];
    int lastPos = 0;
    // from page find out above, choose page appear latest or not appear to swap
out
```

```
    for (size_t i = 1; i < nOpf; i++){
        if (Map[i] > maxValue){
            maxValue = Map[i];
            lastPos = i;
        }else if (Map[i] == 0){
            lastPos = i;
            break;
        }
    }
    // swap in coming page.
    PageFrames[lastPos][pos] = NewArr[pos];
}
void OTPAlgorithmStart()
{
    for (size_t i = 0; i < n; i++){
        if (i > 0)
            CopytoNewPos(i);
        if (checkPageInPageFrame(i)){
            PageFaultRowExtend[i] = ' ';
        } else {
            PageFaultRowExtend[i] = '*';
            PageFaultCount++;
            if (!IsHasVoidSpace(i))
                SwapVictimPageOTP(i);
        }
    }
}
void init()
{
    for (size_t i = 0; i < nOpf; i++){
        for (size_t j = 0; j < n; j++)
            PageFrames[i][j] = -1;
    }
     for (size_t j = 0; j < nOpf; j++)
        PageFrameExtend[j] = INT_MAX;
}
int main(){
    int option1 = 1, option2 = 1;
    cout <<"--- Page Replacement algorithm ---\n";
    cout <<"1. Default referenced sequence\n";
    printf("2. Manual input sequence\n");
    cin >> option1;
    if (option1 != 1){
        cout <<"--Please enter number of reference string:";
        cin >> n;
```

```
        cout <<"--Input " << n << " element in reference string--\n";
        for (size_t i = 0; i < n; i++)
            cin >> NewArr[i];
    }
    cout <<"--- Page Replacement algorithm ---\n";
    cout <<"Input page frames:";
    cin >>nOpf;

    cout << "--- Page Replacement algorithm ---\n";
    cout <<"1. FIFO algorithm\n";
    cout <<"2. OPT algorithm\n";
    cout <<"3. LRU algorithm\n";
    cin >> option2;
    init();
    if (option2 == 1)
        FIFOAlgorithmStart();
    else if (option2 == 2)
        OTPAlgorithmStart();
    else
        LRUAlgorithmStart();

    for (size_t j = 0; j < n; j++)
            cout << NewArr[j] << " ";
    cout << endl;
    for (size_t i = 0; i < nOpf; i++){
        for (size_t j = 0; j < n; j++){
            if (PageFrames[i][j] > -1)
                cout << PageFrames[i][j] << " ";
            else
                cout << "  ";
        }
        cout << endl;
    }
    for (size_t j = 0; j < n; j++)
        cout << PageFaultRowExtend[j] << " ";
    cout << endl;
    cout << "Number of page Fault: " << PageFaultCount;
    return 0;
}
```

6.4.2. Kết quả thực hiện chương trình:

6.4.2.1. FIFO

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
1
2 1 5 2 1 1 0 9 0 0 7
2 2 2 2 2 2 0 0 0 0 0
  1 1 1 1 1 1 9 9 9 9
    5 5 5 5 5 5 5 5 7
* * *       * *     *
Number of page Fault: 6
```

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
--Please enter number of reference string:20
--Input 20 element in reference string--
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
1
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 4 4 4 0 0 0 0 0 0 0 7 7 7
  0 0 0 0 3 3 3 2 2 2 2 2 1 1 1 1 1 0 0
    1 1 1 1 0 0 0 3 3 3 3 3 2 2 2 2 2 1
* * * *   * * * * * *     * *     * * *
Number of page Fault: 15
PS C:\Users\drawt\Downloads\Subject\CPPPARAC>
```

6.4.2.2. OTP

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
2
2 1 5 2 1 1 0 9 0 0 7
2 2 2 2 2 2 2 2 2 2 2
  1 1 1 1 1 0 0 0 0 7
    5 5 5 5 5 9 9 9 9
* * *       * *     *
Number of page Fault: 6
```

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
--Please enter number of reference string:20
--Input 20 element in reference string--
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
2
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  0 0 0 0 0 0 4 4 4 0 0 0 0 0 0 0 0 0 1
    1 1 1 3 3 3 3 3 3 3 3 1 1 1 1 7 7 7
* * * *   *   *     *     *       *   *
Number of page Fault: 10
```

6.4.2.3. LRU

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
1
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
3
2 1 5 2 1 1 0 9 0 0 7
2 2 2 2 2 2 2 9 9 9 9
  1 1 1 1 1 1 1 1 1 7
    5 5 5 5 0 0 0 0 0 0
* * *       * *     *
Number of page Fault: 6
```

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
--Please enter number of reference string:20
--Input 20 element in reference string--
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
3
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 4 4 4 0 0 0 1 1 1 1 1 1 1
  0 0 0 0 0 0 0 0 3 3 3 3 3 3 0 0 0 0 0
    1 1 1 3 3 3 2 2 2 2 2 2 2 2 2 7 7 7
* * * *   *   * * * *     *   *   *
Number of page Fault: 12
```

6.5 Bài tập ôn tập

6.5.1. Nghịch lý Belady là gì? Sử dụng chương trình đã viết trên để chứng minh nghịch lý này.

- Nghịch lý Belady là sự bất thường diễn ra ở giải thuật FIFO, khi số frame tăng, tuy nhiên số pagefault không giảm ngược lại còn tăng theo.
- Chứng minh nghịch lý bằng chương trình đã viết:

Chương trình có chuỗi: 1 2 3 4 1 2 5 1 2 3 4 5

❖ Với 3 khung trang

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
--Please enter number of reference string:12
--Input 12 element in reference string--
1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames:3
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
1
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 4 4 4 5 5 5 5 5 5
  2 2 2 1 1 1 1 1 3 3 3
    3 3 3 2 2 2 2 2 4 4
* * * * * * *     * *
Number of page Fault: 9
```

Số pagefault là 9

❖ Với 4 khung trang:

```
PS C:\Users\drawt\Downloads\Subject\CPPPARAC> .\FIFO-OTP-LRU.exe
--- Page Replacement algorithm ---
1. Default referenced sequence
2. Manual input sequence
2
--Please enter number of reference string:12
--Input 12 element in reference string--
1 2 3 4 1 2 5 1 2 3 4 5
--- Page Replacement algorithm ---
Input page frames:4
--- Page Replacement algorithm ---
1. FIFO algorithm
2. OPT algorithm
3. LRU algorithm
1
1 2 3 4 1 2 5 1 2 3 4 5
1 1 1 1 1 1 5 5 5 5 4 4
  2 2 2 2 2 2 1 1 1 1 5
    3 3 3 3 3 3 2 2 2 2
      4 4 4 4 4 4 3 3 3
* * * *     * * * * * *
Number of page Fault: 10
```

Số pagefault là 10

⇒ Ta có số khung trang từ 3 lên 4, nhưng số pagefault lại tăng từ 9 lên 10. Thỏa nghịch lý Belady ở trên (đpcm).

6.5.2. Nhận xét về mức độ hiệu quả và tính khả thi của các giải thuật FIFO, OPT, LRU.

6.5.2.1. Về mức độ hiệu quả.

Ta có bảng so sánh 3 thuật toán dưới đây (dựa vào kết quả thực hiện chương trình từ phần 6.4).

Chuỗi tham chiếu	**7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1**	
Number of Pageframes		**3**
Giải thuật	**Số page fault**	**Thứ tự**
FIFO	**15**	**3**
LRU	**12**	**2**
OTP	**10**	**1**

Dựa vào bảng trên có thể thấy độ hiệu quả của thuật toán OTP là cao nhất, độ hiệu quả của thuật toán FIFO là thấp nhất, LRU có độ hiệu quả trung bình.

6.5.2.2. Về tính khả thi

- Trong thực tế giải thuật bất khả thi nhất là OTP vì chúng ta không xác định được chuỗi tham chiếu trong tương lai, nên việc dựa vào tương lai để lựa chọn victim theo thuật toán OTP lầ bất khả thi.
- Giải thuật phức tạp nhất là giải thuật LRU, vì hoặc chúng ta sẽ lặp lại toàn bộ các trạng thái của pageframe trước để kiểm tra page nào đến sớm nhất, hoặc lưu lại thời gian đến của từng page và cập nhật lại thời gian đén của page đó nếu nó đã ở trong pageframes, điều này gây tiêu tốn bộ nhớ và tài nguyên của máy tính. Tuy nhiên nó lại có độ hiệu quả nhất định nên thường được sử dụng hơn 2 thuật toán trên.