페이지 교체 기법
시뮬레이터 구현 보고서

Makefile ->X

컴파일 방법 : gcc OS_5.c -> ./a.out

김시온
20172609

1)서론

가상메모리 관리 기법의 하나인 페이지 교체 기법 중 OPT, FIFO, LRU, Second-Chance 를 구현하고 동작 과정을 보여주는 시뮬레이터 구현한다.

1.FIFO(First In First Out)

FIFO 방식은 가장 먼저 메모리에 적재된 페이지를 먼저 내보낸다.

2.OPT (Optimal replacement, 최적 교체)

OPT 방식은 미래를 보고 앞으로 가장 사용 안될 페이지를 교체해 준다.

3.LRU (Least Recently Used)

LRU 방식은 최근에 가장 오랫동안 사용되지 않은 페이지를 교체해준다.

4. Second Chance Algorithm

FIFO 를 기반으로 바꿀 페이지의 참조비트가 1 이면 한번 더 기회를 준다.

소스코드를 먼저 서술하고,

그 후에 함수의 결과화면이 배치 되어 있습니다.

2)본론

사용자에게 input 파일을 입력받고, 그 파일이 존재하면, 양식에 따라 페이지 프레임의 개수와, 참조되는 Page reference string 을 배열에 넣어준다.

양식에 맞지 않은 값이 들어올 경우,

(Frame 개수가 1~4 가 아닐 때, input 파일이 존재하지 않을 때) 프로그램을 종료시킨다.

그 후 사용자가 원하는 방식으로 페이지 관리한 결과를 보여준다.

1: OPT 2: FIFO 3:LRU 4:Second-Chance

main 함수 :

```
int main()

{
    char fname[20];
    printf("%s","Please Put File name : ");
    scanf("%s",fname);
    FILE* fp=fopen(fname,"r");
    if (fp == NULL)
    {
        printf("File Name is wrong.\n");
        exit(1);
    }
    fgets(str1, LEN, fp);
    fgets(str2, LEN, fp);
    strcpy(PageRefence, str2);
    fclose(fp);
    F_N = atoi(&str1[0]);

    if (F_N < 1 || F_N > 4)
    {
        printf("put number between 1 ~ 4\n");
        return -1;
    }

    char *ptr = strtok(str2, " "); // " " 공백 문자를 기준으로 문자열을 자름, 포인터 반환
    while (ptr != NULL)           // 자른 문자열이 나오지 않을 때까지 반복
    {
        page[R_N] = atoi(&ptr[0]); // 자른 문자열 출력
        R_N++;
```

```
        ptr = strtok(NULL, " ");
        if (R_N > 30)
        {
            printf("Reference Num must be 1~30\n");
            return -1;
        }
    }
    if (R_N == 0)
    {
        printf("Reference Num must be 1~30\n");
        return -1;
    }
    memset(Frame, -1, sizeof(Frame));

    printf("Used method (1 : OPT, 2: FIFO, 3:LRU, 4:Second-Chance) : ");
    int button = 0;
    scanf("%d", &button);
    switch (button)
    {
    case 1:
        OPT();
        break;
    case 2:
        FIFO();
        break;
    case 3:
        LRU();
        break;
    case 4:
        Second_Chance();
        break;
    default:
        printf("put 1~4\n");
    }
}
```

FIFO, LRU, Second-Chance, OPT 함수는 비슷한 형태로 함수가 구현이 되었는데,

```
HeadPrint();
```

함수를 사용해 양식에 맞는 출력을 진행하고, 각 페이지 입력 시간마다

```
printOnce();
```

를 사용해 출력후 ,

```
printf("Number of  page faults : %d times\n", Fault_Times);
```

를 통해 page fault가 몇번 일어났는지 출력한다.

1) 처리에 필요한 전역변수들 , 출력함수들.

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

#define LEN 100001
int page[31];

char str1[LEN]; //
char str2[LEN];

int F_N = 0;
int R_N = 0;

int Frame[4];
int Times = 0;
int Page_Fault[31];
int Fault_Times = 0;
char PageRefence[LEN];
bool bit[5];

int tmp[4];
void printOnce()
{
    printf("%d\t\t", Times + 1);
    for (int i = 0; i < F_N; i++)
    {
        if (Frame[i] == -1)
            printf("\t");
        else
            printf("%d\t", Frame[i]);
    }
    if (Page_Fault[Times])
    {
        printf("F");
        Fault_Times++;
    }
    printf("\n");
    Times++;
}
void HeadPrint()
{
    printf("Page reference string : ");
    printf("%s\n", PageRefence);
    printf("\tframe\t");
    for (int i = 1; i <= F_N; i++)
```

```
        printf("%d\t", i);
    printf("page fault\n");
    printf("time\n");
}
```

OPT 함수 :

참조 페이지 배열을 탐색하여, 가장 늦게 나타나는 페이지번호가 있는 프레임의 값을 바꿔준다.

```
void OPT()
{
    HeadPrint();
    int flag = 0;
    for (int i = 0; i < R_N; i++) //매줄
    {
        memset(tmp, 0, sizeof(tmp));
        int num = page[i]; //바꿀친구
        int exist = 0;
        for (int j = 0; j < F_N; j++) //페이지 fault 체크
        {
            if (Frame[j] == num)
            {
                exist = 1;
                break;
            }
        }
        if (exist)
        {
            exist = 0;
            printOnce();
            continue;
        }
        Page_Fault[i] = 1;
        for (int j = 0; j < F_N; j++)
        {
            if (Frame[j] == -1) // 아직 암것도 안들어갔을때
            {
                Frame[j] = num;
                flag = 1;
                break;
            }

            if (i != (R_N - 1))              //맨끝경우
```

```
                for (int k = i; k < R_N; k++) // 뒤에 몇번나오는지 기록하기
                {
                    if (page[k + 1] == Frame[j]) //겹치면
                    {
                        tmp[j] = k; // tmp[j] == j 번째 테이블에 있는 값이 언제 처음 나오는지
                        break;
                    }
                }
            if (tmp[j] == 0) // 안나오면
            {
                // printf("%d 번째 frame 은 뒤에 안나옴", j + 1);
                Frame[j] = num;
                flag = 1;
                break;
            }
        }
        if (flag)
        {
            flag = 0;
            printOnce();
            continue;
        }

        int changeIndex = 0;
        int max = 0;
        for (int k = 0; k < F_N; k++)
        {
            if (max < tmp[k])
            { // k 번째 프레임이 가장 늦을때
                max = tmp[k];
                changeIndex = k;
            }
        }
        if (i == 10)
            printf("바꿀 프레임 번호 : %d\n", changeIndex + 1);
        Frame[changeIndex] = num;

        printOnce();
    }

    printf("Number of  page faults : %d times\n", Fault_Times);
}
```

FIFO함수 : 프레임 개수만큼 배열을 설정하여, 만약 프레임개수가 3이면,

1 2 3 1 2 3 이런식으로 계속 순회하며 페이지 교체를 진행한다.

```
void FIFO()
{
    HeadPrint();
    int flag = 0;
    char *a = NULL;
    int Index = 0;
    for (int i = 0; i < R_N; i++) //매줄
    {
        memset(tmp, 0, sizeof(tmp));
        int num = page[i]; //바꿀친구
        int exist = 0;
        for (int j = 0; j < F_N; j++) //페이지 fault 체크
        {
            if (Frame[j] == num)
            {
                exist = 1;
                break;
            }
        }
        if (exist)
        {
            exist = 0;
            printOnce();
            continue;
        }
        Page_Fault[i] = 1;

        if (Index < F_N) // 3보다 작다. 0 1 2
        {
            Frame[Index] = num;
            Index++;
        }
        else
            printf("범위 넘어섬\n");

        if (Index >= F_N)
            Index = 0;
        printOnce();
    }
    printf("Number of  page faults : %d times\n", Fault_Times);
}
```

LRU 함수 : OPT 기법과 비슷하나, 자신의 앞에 있는 배열의 값들중에

가장 오래전에 참조했던 페이지번호를 가진 페이지 프레임의 값을 바꿔준다.

```
void LRU()
{
    HeadPrint();
    int flag = 0;
    for (int i = 0; i < R_N; i++) //매줄
    {
        for (int i = 0; i < F_N; i++)
            tmp[i] = 999;

        int num = page[i]; //바꿀친구
        int exist = 0;
        for (int j = 0; j < F_N; j++) //페이지 fault 체크
        {
            if (Frame[j] == num)
            {
                exist = 1;
                break;
            }
        }
        if (exist)
        {
            exist = 0;
            printOnce();
            continue;
        }
        Page_Fault[i] = 1;
        for (int j = 0; j < F_N; j++)
        {
            if (Frame[j] == -1) // 아직 암것도 안들어갔을때
            {
                Frame[j] = num;
                flag = 1;
                break;
            }

            //맨끝경우
            for (int k = i; k > 0; k--) // 뒤에 몇번나오는지 기록하기
            {
                if (page[k - 1] == Frame[j]) //겹치면, and i가 0일땐 올수가없으니 괜찮
                {
                    tmp[j] = k; // tmp[j] == j번째 테이블에 있는 값이 언제 처음 나오는지
                    break;
```

```
                }
            }

            if (tmp[j] == 0) // 안나오면
            {
                // printf("%d 번째 frame 은 뒤에 안나옴", j + 1);
                Frame[j] = num;
                flag = 1;
                break;
            }
        }
        if (flag)
        {
            flag = 0;
            printOnce();
            continue;
        }

        int changeIndex = 0;
        int min = 99;
        for (int k = 0; k < F_N; k++)
        {
            if (min > tmp[k])
            { // k 번째 프레임이 가장 늦을때
                min = tmp[k];
                changeIndex = k;
            } //세개 다 없으면?
        }
        Frame[changeIndex] = num;
        printOnce();
    }
    printf("Number of  page faults : %d times\n", Fault_Times);
}
```

Second_Chance 함수: FIFO 방식을 기반으로, 참조비트를 하나 더 만들어 초깃값을 0으로 설정한다.

그 후 어떠한 페이지 프레임에 있는 값 덕분에 page fault가 일어나지 않은 경우, 그 페이지 프레임에 대응하는 참조 비트를 1로 설정한다.

page fault가 일어났을 때, FIFO 방식으로 바꾸기로 약속된 페이지 프레임에 대응하는 참조비트를 보고,

만약 0이면 그대로 페이지 교체를 하고, 1이면 그 비트를 0으로 바꾸고, 다음 프레임을 살펴본다.

```
void Second_Chance()
{
    HeadPrint();
    int flag = 0;
    char *a = NULL;
    int Index = 0;
    for (int i = 0; i < R_N; i++) //매줄
    {
        memset(tmp, 0, sizeof(tmp));
        int num = page[i]; //바꿀친구
        int exist = 0;
        for (int j = 0; j < F_N; j++) //페이지 fault 체크
        {
            if (Frame[j] == num) //있을때
            {
                bit[j] = true;
                exist = 1;
                break;
            }
        }
        if (exist)
        {
            exist = 0;
            printOnce();
            continue;
        }
        Page_Fault[i] = 1;

        while (true)
        {
            if (Index < F_N)
            {
                if (bit[Index])
                    bit[Index] = false;
                else
                {
                    Frame[Index] = num;
                    Index++;
                    break;
                }
                Index++;
            }
            if (Index >= F_N)
                Index = 0;
        }
        printOnce();
    }
    printf("Number of  page faults : %d times\n", Fault_Times);}
```

실행 결과:

```
sion@ubuntu:~$ cat input.txt
3
2 3 2 1 5 2 4 5 3 2 5 2
```

1) OPT

```
sion@ubuntu:~$ gcc OS_5.c
sion@ubuntu:~$ ./a.out
Please Put File name : input.txt
Used method (1 : OPT, 2: FIFO, 3:LRU, 4:Second-Chance) : 1
Page reference string : 2 3 2 1 5 2 4 5 3 2 5 2

        frame   1       2       3       page fault
time
1               2                       F
2               2       3               F
3               2       3
4               2       3       1       F
5               2       3       5       F
6               2       3       5
7               4       3       5       F
8               4       3       5
9               4       3       5
10              2       3       5       F
11              2       3       5
12              2       3       5
Number of  page faults : 6 times
```

2) FIFO

```
sion@ubuntu:~$ ./a.out
Please Put File name : input.txt
Used method (1 : OPT, 2: FIFO, 3:LRU, 4:Second-Chance) : 2
Page reference string : 2 3 2 1 5 2 4 5 3 2 5 2

        frame   1       2       3       page fault
time
1               2                       F
2               2       3               F
3               2       3
4               2       3       1       F
5               5       3       1       F
6               5       2       1       F
7               5       2       4       F
8               5       2       4
9               3       2       4       F
10              3       2       4
11              3       5       4       F
12              3       5       2       F
Number of  page faults : 9 times
```

3) LRU

```
sion@ubuntu:~$ ./a.out
Please Put File name : input.txt
Used method (1 : OPT, 2: FIFO, 3:LRU, 4:Second-Chance) : 3
Page reference string : 2 3 2 1 5 2 4 5 3 2 5 2

        frame   1       2       3       page fault
time
1               2                       F
2               2       3               F
3               2       3
4               2       3       1       F
5               2       5       1       F
6               2       5       1
7               2       5       4       F
8               2       5       4
9               3       5       4       F
10              3       5       2       F
11              3       5       2
12              3       5       2
Number of  page faults : 7 times
```

4) Second_Chance

```
sion@ubuntu:~$ ./a.out
Please Put File name : input.txt
Used method (1 : OPT, 2: FIFO, 3:LRU, 4:Second-Chance) : 4
Page reference string : 2 3 2 1 5 2 4 5 3 2 5 2

        frame   1       2       3       page fault
time
1               2                       F
2               2       3               F
3               2       3
4               2       3       1       F
5               2       5       1       F
6               2       5       1
7               2       5       4       F
8               2       5       4
9               2       5       3       F
10              2       5       3
11              2       5       3
12              2       5       3
Number of  page faults : 6 times
```

모두 정상적으로 수행됨을 볼 수 있다.

3. 결론 및 느낀점

페이지 관리 기법에는 여러가지가 있다는 것을 깨달았고, 막연하게만 사용하던 컴퓨터는 상당히 복잡하고 효율적인 알고리즘을 항상 찾는 과정을 겪고 태어났구나 라는 생각이 들었다.

그중에 OPT 알고리즘은 미래를 봐서 예측하는 것이라 비현실적인 알고리즘이라는데, 역시 이론적으로 좋은게 항상 실제로 좋은게 아니라는걸 다시금 느끼는 계기가 되었다.