20172621 컴퓨터학부 노정환

1. 구현 설명

이번 과제의 목표는 페이지 교체 기법 중 OPT, FIFO, LRU, Second-Chance의 동작을 보여주는 프로그램을 구현하는 것이다.

프로그램이 실행되면, input 파일의 이름을 먼저 입력받는다. 이때 존재하지 않는 파일 이름을 입력하면 다시 입력할 수 있다. 입력 받은 파일 명으로 fopen함수를 호출해 파일을 열고, 페이지 프레임 개수는 frame_cnt에, 페이지 잠조 횟수는 ref_cnt에 각각 저장한다. 참조할 페이지 번호는 page_ref 배열에 저장한다.

그 다음 과정은 출력 결과 예시에 나온 형식과 같다. “Used method : ” 문장 다음에 사용할 교체 기법의 이름을 입력한다. 그러면 해당 기법에 맞는 함수들을 호출한다. 프로그램을 종료하려면 여기에 “quit”를 입력한다. 만약 정의되지 않은 명령이 입력되면 다시 입력하게끔 한다.

구현에 사용한 파일은 task.c 파일 하나이며, 별다른 컴파일 옵션은 없다.

각 기법의 수행 방식에 대해서, 기본 틀은 동일하므로 기본 틀에 대해 먼저 설명하고, 각 기법의 구현 방식에 대해 설명한다.

형식에 맞는 출력문들을 제외하면, 페이지 참조 번호가 페이지 프레임에 있다면 히트가 발생한 것이므로 넘어간다. 그렇지 않다면 페이지 폴트가 발생하는 것은 확실시한다. 만약 비어있는 페이지 프레임이 있다면 페이지 폴트가 발생한 페이지 번호를 그곳에 저장한다. 비어있는 페이지 프레임이 없다면 각 기법에 맞춰 다른 방식으로 수행한다.

OPT 기법의 경우, 페이지 프레임에서 가장 나중에 참조할 페이지 번호를 찾아 내쫓는다. 이 페이지 번호를 찾기 위해서 앞으로 참조할 페이지 번호 목록을 탐색해서, 참조하지 않을 페이지 번호를 찾는다.

앞으로 한 번도 참조하지 않는 페이지 번호가 있다면 해당 페이지 번호를 프레임에서 내쫓고 새 번호를 할당한다. 만약 없다면, 현재 시점에서 페이지 번호가 참조되는 시점까지의 거리(len)를 계산해서 비교하고 내쫓을 프레임 번호(target)을 정한다.

FIFO 기법을 구현하기 위해서 큐(queue)를 사용했다. 큐의 길이는 페이지 폴트가 발생할 때 한 번만 증가하므로, 큐의 크기는 참조할 페이지 목록의 길이(ref_cnt)로 정한다. 큐의 앞부분은 target이 가리키고, 끝부분은 end_q가 가리킨다.

비어있는 페이지 프레임에 페이지 번호를 저장할 때는 end_q가 가리키는 곳에 번호를 저장하고 end_q를 늘린다. 기존의 페이지 프레임에서 내쫓고 새로 넣는 경우, queue[target]이 가리키는 프레임 번호의 값을 내쫓고 참조할 페이지 번호를 저장한다. 이때 참조할 번호를 새로 넣는 것도 처리해야 하므로, queue에서 end_q가 가리키는 곳에 queue[target]에 저장된 프레임 번호를 넣고, end_q와 target 값을 1씩 늘린다.

LRU 방식은 OPT 방식을 거꾸로 한다. 가장 오래 전에 참조한 페이지 번호가 들어간 페이지 프레임을 찾아서 내쫓고 새 값을 할당한다.

가장 최근에 참조한 페이지 번호를 찾기 위해서 이전에 참조한 페이지 목록을 탐색해서 그 거리(len)를 계산하고, 비교해서 가장 오래 전에 참조한 페이지 번호의 위치(target)을 찾는다.

Second-Chance 방식은 기본적으로는 FIFO 방식으로 동작하지만, 각 페이지 프레임이 hit label을 사용해서 hit 횟수를 기록한다.

페이지 번호를 참조할 때, 페이지 프레임에 있는 번호라면 해당 프레임의 hit값을 1 늘린다. 페이지 폴트가 발생하면 queue를 처음부터 탐색하는데, 일반적인 FIFO와 달리 항상 맨 앞의 프레임이 쫓겨나지 않기 때문이다. cur_q가 가리키는 queue 값이 음수이면 프레임 번호가 이미 쫓겨난 것이므로 넘어간다. hit 값이 양수이면 hit 값을 1 줄이고, queue의 맨 끝으로 프레임 번호를 옮긴다. 0 이하의 hit 값을 갖는 프레임 번호가 나오면 해당 프레임에 참조할 페이지 번호를 저장하고, queue의 맨 끝으로 프레임 번호를 옮긴다.

2. 기법 간의 비교 분석

먼저 아래 실행 결과에 나온 예시처럼 비슷한 페이지 번호가 섞여서 연속으로 나오는 경우, 페이지 프레임 개수와 상관없이 OPT 기법이 가장 적은 페이지 폴트 발생을 보이고, FIFO 기법이 많은 페이지 폴트 발생을 보였다.

```
root@ubuntu:/home/potato/task# cat i2.txt
4
5 1 2 3 1 4 3 4 2 14 13 12 13 11 12 15 11 12
root@ubuntu:/home/potato/task# cat i3.txt
3
1 2 3 2 3 1 2 3 10 11 10 12 11 10 12 10 21 23 22 23 22 21 22 23
```

이처럼 메모리의 공간적 지역성을 고려하여, 어느 한 영역의 페이지 번호들을 참조하다가 다른 영역으로 넘어갈 경우에는 전체적으로 비슷한 페이지 폴트 발생 횟수를 보여주었다. 다만 Second-Chance 기법의 경우, 다른 영역으로 넘어갈 때 이전 영역에서 증가한 hit label 값 때문에, 새로 번호가 할당된 프레임만을 다시 내쫓고 새로 할당받는 일이 발생하여 페이지 폴트 발생 횟수가 다른 기법에 비해 많은 경우가 있었다.

4. 실행 결과

```
root@ubuntu:/home/potato/task# cat input.txt
3
2 3 2 1 5 2 4 5 3 2 5 2
root@ubuntu:/home/potato/task# ./task
Type file name : input.txt

Used method : OPT
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
```

```
Used method : FIFO
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
```

```
Used method : LRU
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
```

```
Used method : Second-Chance
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

Used method : quit

program terminated.

root@ubuntu:/home/potato/task#
```