

운영체제_SWE3004-43

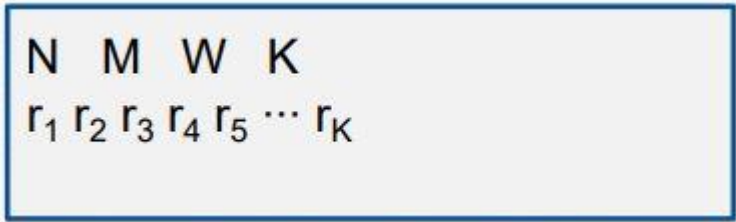
Project 3: Virtual Memory Management 구현

소프트웨어학과 2017314626 이재훈

이 프로그램은, Virtual Memory를 management하는 기법 중 FA의 MIN, FIFO, LRU, LFU 기법과 VA의 WS 기법을 구현한다. Input.txt에서 프로세스의 page 개수, page frame 개수, window size, page reference string을 입력 받은 후 그 정보에 따라 총 page fault 횟수와 메모리의 상태 변화 과정을 보여준다.

input.txt

input.txt의 형식은 다음과 같다. 1행에는 프로세스가 가지는 page 개수 (N), 할당 page frame 개수 (M), WS 기법에서 사용할 window size (W), page



The diagram shows a rectangular box containing two lines of text. The first line has four space-separated characters: N, M, W, K. The second line has a sequence of characters: r₁, r₂, r₃, r₄, r₅, followed by an ellipsis (three dots), and then r_K.

reference string의 길이 (K)가 입력된다. 2행부터는 page reference string이 나타난다. 이 때, page의 번호는 0부터 시작하며, page frame의 개수는 최대 20, window size의 최대 값은 100이다. 또한 page reference string의 길이는 최대 1000이다.

고려사항

초기 할당된 page frame들은 모두 비어 있는 것으로 가정한다. 또한, MIN 기법에서 tie breaking rule로는 page with smallest page number를 교체한다. 그리고 LFU 기법에서 tie breaking rule로는 LRU 기법을 사용한다. 이 외 input의 최대값에 대한 고려사항은 위의 input.txt에 나와있다.

설계 및 구현 내용

이 프로그램은 main.c 로 구성된다.

Input.txt에서 받아온 정보들은 각 변수에 저장된다. 이 프로그램에서 사용되는 변수는 다음과 같다.

N: 프로세스가 가지는 page의 개수를 저장한다.

M: 할당 page frame의 개수를 저장한다.

W: window size를 저장한다.

K: page reference string의 길이를 저장한다.

pg_ref: page reference string을 저장한다.

memory: 메모리의 상태(어떤 page가 있는지)를 저장한다.

timestamp_load: page가 로드된 시간을 기록한다.

timestamp_ref: page가 가장 최근 참조된 시간을 기록한다.

ref_cnt: page가 참조된 횟수를 기록한다.

ws: working set에 존재하는지 없는지 상태여부를 저장한다.

exist = 현재 참조하려고 하는 page가 이미 메모리에 존재하는지 판단한다.

empty = 현재 page frame 중 빈 공간이 있는지 판단한다.

구현 과정은 다음과 같다.

1. 필요한 변수를 선언 및 동적할당 해준 후 'input.txt' 파일에서 각 정보를 받아온다.
Input.txt에서 정보를 받아오지 않는 나머지 변수들은 초기화한다.
2. MIN 기법을 시작한다. 아래 과정은 time = 1에서 time = K+1까지 반복한다.
 - 1) Forward distance 기록을 위한 Df 변수를 선언 및 동적 할당한다.
 - 2) 이미 메모리에 해당 page가 존재하는지 확인한다.
 - 3) 존재하지 않는다면, page fault 횟수를 1 증가시킨다.
 - 4) 현재 메모리에 빈 page frame이 있는지 검사한다. 있으면 page를 넣는다.
 - 5) 현재 메모리에 있는 page 중 교체될 page를 찾기 위해, 각 page의 Forward distance를 계산한다.
 - 6) Forward distance가 가장 큰 page를 교체하되, 만약 tie 상황이 생긴다면, page number가 작은 page를 교체한다.
3. FIFO 기법을 시작한다. 아래 과정은 time = 1에서 time = K+1까지 반복한다. 아래 과정에서 page가 load될 때는 항상 timestamp_load를 갱신한다.
 - 1) 이미 메모리에 해당 page가 존재하는지 확인한다
 - 2) 존재하지 않는다면, page fault 횟수를 1 증가시킨다.
 - 3) 현재 메모리에 빈 page frame이 있는지 검사한다. 있으면 page를 load한다.

- 4) 현재 메모리에 있는 page중 교체될 page를 찾기 위해, 각 page의 timestamp_load를 비교하여 가장 오래된 page를 교체한다.
4. LRU 기법을 시작한다. 아래 과정은 time = 1에서 time = K+1까지 반복한다. 아래 과정 중 page가 참조될 때는 항상 timestamp_ref를 갱신한다.
 - 1) 이미 메모리에 해당 page가 존재하는지 확인한다
 - 2) 존재하지 않는다면, page fault 횟수를 1 증가시킨다.
 - 3) 현재 메모리에 빈 page frame이 있는지 검사한다. 있으면 page를 load한다.
 - 4) 현재 메모리에 있는 page중 교체될 page를 찾기 위해, 각 page의 timestamp_ref를 비교하여 가장 참조된 지 오래된 page를 교체한다.
5. LFU 기법을 시작한다. 아래 과정은 time = 1에서 time = K+1까지 반복한다. 아래 과정 중 page가 참조될 때는 항상 ref_cnt를 증가시킨다. 또한 tie breaking rule을 위한 LRU 기법을 위해 timestamp_ref도 갱신한다.
 - 1) 이미 메모리에 해당 page가 존재하는지 확인한다
 - 2) 존재하지 않는다면, page fault 횟수를 1 증가시킨다.
 - 3) 현재 메모리에 빈 page frame이 있는지 검사한다. 있으면 page를 load한다.
 - 4) 현재 메모리에 있는 page중 교체될 page를 찾기 위해, 각 page의 ref_cnt를 비교하여 가장 참조된 횟수가 적은 page를 교체한다.
 - 5) 만약 ref_cnt가 같은 tie한 상황이 생긴다면, timestamp_ref를 비교하여 LRU를 적용한다.
6. WS 기법을 시작한다. 아래 과정은 time = 1에서 time = K+1까지 반복한다.
 - 1) Window set에서 제거되어야 하는 이전의 page를 제거한다.
 - 2) 현재 참조하려는 page가 이미 window set에 존재하는지 확인한다. 없다면 page fault를 1 증가시킨다.
 - 3) 현재 참조한 page의 ws 값을 1 증가시킨다. 만약 이 값이 0이라면 window set에 해당 page가 포함되지 않는 것이다.
7. 각 기법이 끝날 때마다 총 page fault 횟수와 메모리의 상태가 출력된다.
8. 동적 할당된 변수들을 free로 메모리 해제해 준 후 프로그램이 종료된다.

실행 환경 및 실행 방법

이 프로그램은 Linux 우분투 시스템에서 C 언어를 사용하여 작성되었으며, GCC로 실행된다. 실행 방법은 c파일을 gcc -o를 통해 실행파일로 만든 후 실행파일을 실행한다. 이 과정은 다음과 같다.

```
jaehun@jaehun-VirtualBox:~/다운로드$ gcc -o pa3 main.c
jaehun@jaehun-VirtualBox:~/다운로드$ ./pa3
```

첫 번째 줄이 실행파일을 생성하는 것이고 두번째 줄이 실행파일을 실행하는 것이다.

출력 형태

출력 형태는 MIN, FIFO, LRU, LFU, WS 순으로 나타난다. 각 기법 별로 시간, 메모리의 page frame에 있는 page (메모리 상태), page fault 발생 여부가 출력된다. 마지막으로 총 page fault 횟수가 출력된다. 다음은 그 예시다.

```
MIN Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 3 1 2 page fault!
time: 5 memory: 3 1 2
time: 6 memory: 3 1 2
time: 7 memory: 3 1 4 page fault!
time: 8 memory: 5 1 4 page fault!
time: 9 memory: 5 1 4
time: 10 memory: 5 1 4
time: 11 memory: 5 3 4 page fault!
time: 12 memory: 5 3 4
time: 13 memory: 5 3 4
time: 14 memory: 5 3 4
time: 15 memory: 5 3 4
total page fault : 7
```

입출력 결과

첫번째 input으로는 과제 설명 pdf에 있는 예시를 사용하였다. 6개의 page 개수, 3개의 page frame, 3 크기의 window size, 그리고 15개의 page reference string을 가진다. 그 후 page reference string의 목록이 입력된다. 결과는 아래와 같다.



```
MIN Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 3 1 2 page fault!
time: 5 memory: 3 1 2
time: 6 memory: 3 1 2
time: 7 memory: 3 1 4 page fault!
time: 8 memory: 5 1 4 page fault!
time: 9 memory: 5 1 4
time: 10 memory: 5 1 4
time: 11 memory: 5 3 4 page fault!
time: 12 memory: 5 3 4
time: 13 memory: 5 3 4
time: 14 memory: 5 3 4
time: 15 memory: 5 3 4
total page fault : 7
```

```
FIFO Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 3 1 2 page fault!
time: 5 memory: 3 1 2
time: 6 memory: 3 1 2
time: 7 memory: 3 4 2 page fault!
time: 8 memory: 3 4 5 page fault!
time: 9 memory: 3 4 5
time: 10 memory: 1 4 5 page fault!
time: 11 memory: 1 3 5 page fault!
time: 12 memory: 1 3 4 page fault!
time: 13 memory: 1 3 4
time: 14 memory: 1 3 4
time: 15 memory: 5 3 4 page fault!
total page fault : 10
```

```

LRU Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 3 1 2 page fault!
time: 5 memory: 3 1 2
time: 6 memory: 3 1 2
time: 7 memory: 3 4 2 page fault!
time: 8 memory: 3 4 5 page fault!
time: 9 memory: 3 4 5
time: 10 memory: 1 4 5 page fault!
time: 11 memory: 1 4 3 page fault!
time: 12 memory: 1 4 3
time: 13 memory: 1 4 3
time: 14 memory: 1 4 3
time: 15 memory: 5 4 3 page fault!
total page fault : 9

```

```

LFU Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 3 1 2 page fault!
time: 5 memory: 3 1 2
time: 6 memory: 3 1 2
time: 7 memory: 3 4 2 page fault!
time: 8 memory: 3 5 2 page fault!
time: 9 memory: 3 4 2 page fault!
time: 10 memory: 3 4 1 page fault!
time: 11 memory: 3 4 1
time: 12 memory: 3 4 1
time: 13 memory: 3 4 1
time: 14 memory: 3 4 1
time: 15 memory: 3 4 5 page fault!
total page fault : 9

```

```

WS Algorithm starts
time: 1 memory: 0 -- -- -- -- -- page fault!
time: 2 memory: 0 1 -- -- -- -- -- page fault!
time: 3 memory: 0 1 2 -- -- -- -- -- page fault!
time: 4 memory: 0 1 2 3 -- -- -- -- -- page fault!
time: 5 memory: 0 1 2 3 -- -- -- -- --
time: 6 memory: -- 1 2 3 -- -- -- -- --
time: 7 memory: -- -- 2 3 4 -- -- -- -- -- page fault!
time: 8 memory: -- -- 2 3 4 5 -- -- -- -- -- page fault!
time: 9 memory: -- -- 2 3 4 5 -- -- -- -- --
time: 10 memory: -- 1 -- 3 4 5 -- -- -- -- -- page fault!
time: 11 memory: -- 1 -- 3 4 5 -- -- -- -- -- page fault!
time: 12 memory: -- 1 -- 3 4 5 -- -- -- -- --
time: 13 memory: -- 1 -- 3 4 -- -- -- -- --
time: 14 memory: -- 1 -- 3 4 -- -- -- -- --
time: 15 memory: -- -- -- 3 4 5 -- -- -- -- -- page fault!
total page fault: 9

```

각 기법 별로 메모리의 상태 변화를 볼 수 있으며, page가 들어있지 않은 page frame의 경우 '—'로 표시하였다. (ws algorithm의 경우 메모리에 없는 page는 전부 '—'로 표시하였다.) 또한 page fault가 일어난 time에는 page fault를 출력하였다. MIN 기법의 경우 7번, FIFO의 경우 10번, LRU, LFU, WS의 경우 9번의 page fault가 발생하였다.

두번째 input으로는 운영체제 수업의 퀴즈에 나온 input을 응용하였다. Input.txt는 다음과 같다.

```

7 3 3 | 14
0 1 2 0 1 2 3 4 5 6 3 4 5 6

```

Page reference string이 012, 3456이 반복되며, 총 7개의 page, 3개의 page frame과

window set, 14개의 page reference string을 가진다. 그 결과는 다음과 같다.

```

MIN Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 0 1 2
time: 5 memory: 0 1 2
time: 6 memory: 0 1 2
time: 7 memory: 3 1 2 page fault!
time: 8 memory: 3 4 2 page fault!
time: 9 memory: 3 4 5 page fault!
time: 10 memory: 3 4 6 page fault!
time: 11 memory: 3 4 6
time: 12 memory: 3 4 6
time: 13 memory: 5 4 6 page fault!
time: 14 memory: 5 4 6
total page fault : 8

```

```

FIFO Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 0 1 2
time: 5 memory: 0 1 2
time: 6 memory: 0 1 2
time: 7 memory: 3 1 2 page fault!
time: 8 memory: 3 4 2 page fault!
time: 9 memory: 3 4 5 page fault!
time: 10 memory: 6 4 5 page fault!
time: 11 memory: 6 3 5 page fault!
time: 12 memory: 6 3 4 page fault!
time: 13 memory: 5 3 4 page fault!
time: 14 memory: 5 6 4 page fault!
total page fault : 11

```



```

LRU Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 0 1 2
time: 5 memory: 0 1 2
time: 6 memory: 0 1 2
time: 7 memory: 3 1 2 page fault!
time: 8 memory: 3 4 2 page fault!
time: 9 memory: 3 4 5 page fault!
time: 10 memory: 6 4 5 page fault!
time: 11 memory: 6 3 5 page fault!
time: 12 memory: 6 3 4 page fault!
time: 13 memory: 5 3 4 page fault!
time: 14 memory: 5 6 4 page fault!
total page fault : 11

```

```

LFU Algorithm starts
time: 1 memory: 0 -- -- page fault!
time: 2 memory: 0 1 -- page fault!
time: 3 memory: 0 1 2 page fault!
time: 4 memory: 0 1 2
time: 5 memory: 0 1 2
time: 6 memory: 0 1 2
time: 7 memory: 3 1 2 page fault!
time: 8 memory: 4 1 2 page fault!
time: 9 memory: 5 1 2 page fault!
time: 10 memory: 6 1 2 page fault!
time: 11 memory: 3 1 2 page fault!
time: 12 memory: 3 4 2 page fault!
time: 13 memory: 3 4 5 page fault!
time: 14 memory: 6 4 5 page fault!
total page fault : 11

```

```

WS Algorithm starts
time: 1 memory: 0 -- -- -- -- -- -- page fault!
time: 2 memory: 0 1 -- -- -- -- -- page fault!
time: 3 memory: 0 1 2 -- -- -- -- -- page fault!
time: 4 memory: 0 1 2 -- -- -- -- --
time: 5 memory: 0 1 2 -- -- -- -- --
time: 6 memory: 0 1 2 -- -- -- -- --
time: 7 memory: 0 1 2 3 -- -- -- -- -- page fault!
time: 8 memory: 0 1 2 3 4 -- -- -- -- -- page fault!
time: 9 memory: -- 1 2 3 4 5 -- -- -- -- -- page fault!
time: 10 memory: -- -- 2 3 4 5 6 -- -- -- -- -- page fault!
time: 11 memory: -- -- -- 3 4 5 6
time: 12 memory: -- -- -- 3 4 5 6
time: 13 memory: -- -- -- 3 4 5 6
time: 14 memory: -- -- -- 3 4 5 6
total page fault: 7

```

이 경우 MIN 기법에서 8회, FIFO, LRU, LFU 기법에서 11회, WS 기법에서 7회의 page fault가 발생하였다.