운영체제 HW1.

201613989 김진산

1. Page Replacement Algorithm

1-1) 현재 제공된 페이지 교체 알고리즘에 구현되어 있는 FIFO, Optimal, LRU, LFU, Second Chance 방식의 소스코드에 대한 분석 내용을 기술하시오.

1-1-1) 전역 변수

|  |  |
| --- | --- |
| Int n | 요구 페이지의 수 |
| Int nf | 프레임의 크기 |
| Int in[] | 요구 페이지의 번호 |
| Int p[] | 프레임의 상태, 현재 어떤 페이지가 프레임에 있는지 저장 |
| Int hit | 페이지 적중을 나타내는 변수 |
| Int I, j, k | For문 사용에 이용되는 변수 |
| Int pgfaultcnt | 페이지 부재 발생 횟수 |

1-1-2) 함수

|  |  |
| --- | --- |
| void getData() | 데이터를 입력 받는 함수. 사용자는 n, in[], nf를 입력한다. 즉 요구 페이지의 수, 각각의 요구 페이지의 이름, 프레임의 크기를 입력한다. |
| void initialize() | 초기화 함수, pgfaultcnt, p[]를 초기화 한다. 즉 페이지 부재 발생 수와 프레임을 초기화 한다. |
| int isHit(int data) | 페이지 적중 확인 함수, 파라미터로 받은 data가 프레임에 있는지 확인한다. 페이지 적중 시 1을, 페이지 부재 시 0을 반환한다. |
| int getHitIndex(int data) | 파라미터로 받은 data가 프레임에 있는지 확인하고, 그 위치를 반환한다. 페이지 부재일 경우는 생각하지 않는다. |
| void dispPages() | 프레임의 상태를 출력하는 함수, 프레임에 저장된 페이지를 모두 출력한다. |
| void dispPgFaultCnt() | 페이지 부재 발생 횟수를 출력하는 함수 |
| void fifo() | 선입선출 페이지 교체 알고리즘  가장 먼저 프레임에 들어온 페이지를 대상 페이지로 선정한다.  코드:  1. initialize()를 통해 초기화 한다.  2. isHit()를 통해 페이지 적중, 부재를 확인한다.  3-1. 페이지 부재일 경우 프레임의 뒤에서부터 페이지를 추가한다. 뒤에부터 추가하기 때문에 맨 앞에 있는 페이지가 시간상 가장 먼저 들어온 페이지이다. 그러므로 페이지 교체 발생시 프레임의 맨 앞의 값을 대상 페이지로 선정한다. 페이지 교체 후 프레임의 상태를 출력한다.  3-2. 페이지 적중일 경우 메시지를 출력한다.  4. 페이지 부재 발생수를 출력한다. |
| void optimal() | 최적 페이지 교체 알고리즘  앞으로 사용하지 않을 페이지를 대상 페이지로 선정한다.  추가 변수:  Int near[]: 이후 나올 페이지의 가장 가까운 위치를 저장한다.  Int pg: 프레임에 있는 특정 페이지가 이후에 나올지 요구 페이지들과 비교하기 위해 프레임의 특정 페이지를 임시로 저장하는 변수  Int found: 프레임에 있는 페이지가 이후에 존재하는지 판단하는 변수, 존재 시 1, 존재하지 않을 경우 0을 저장한다.  Int max: 대상 프레임 선정 시 가장 나중에 다시 사용되는 인덱스를 저장한다.  Int repindex: 변경할 프레임의 위치를 나타낸다.  코드:  1. initialize()를 통해 초기화 한다.  2. isHit()를 통해 페이지 적중, 부재를 확인한다.  3-1. 페이지 부재일 경우, 프레임 안에 있는 모든 페이지에 대하여 다음에 페이지가 나올지 검색한다. 프레임 내의 각각의 페이지 별로 이후 가장 먼저 나오는 동일한 페이지의 인덱스를 near에 저장한다. near값이 가장 큰 프레임의 페이지를 대상 페이지로 선정한다. 페이지 교체 후 페이지 부재 수를 증가시키고 프레임의 상태를 출력한다.  3-2. 페이지 적중일 경우 메시지를 출력한다.  4. 페이지 부재 발생수를 출력한다.  오류:  주어진 요구페이지에서 페이지 적중이 존재하지 않을 경우 프레임을 모두 사용하지 않는 현상이 존재. Repindex가 0에서 증가하지 않아 오류가 발생하고 있다. |
| Void lru() | 최근 최소 사용 페이지 교체 알고리즘  가장 오래전에 사용된 페이지를 대상 페이지로 선정한다.  변수:  Int least[]: 현재 프레임에 있는 페이지들 각각의 가장 최근에 사용된 인덱스를 저장한다.  Int pg: 프레임에 있는 특정 페이지가 이전에 언제 마지막을 나왔었는지 검색하기 위해 프레임의 특정 페이지를 임시로 저장하는 변수  Int min: 대상 프레임 선정 시 가장 오래전에 사용된 인덱스를 저장한다.  Int repindex: 변경할 프레임의 위치를 나타낸다.  코드:  1. initialize()를 통해 초기화 한다.  2. isHit()를 통해 페이지 적중, 부재를 확인한다.  3-1) 페이지 부재 시 프레임에 있는 각각의 페이지에 대해 가장 최근에 사용된 인덱스를 구한다. 이 값이 가장 작은 것이 가장 오래전에 사용되었던 페이지로, 이를 대상 페이지로 선정한다. 페이지 교체 후 페이지 부재 발생 수를 증가시키고 프레임의 상태를 출력한다.  3-2) 페이지 적중 시 메시지를 출력한다.  4. 페이지 부재 발생 수를 출력한다. |
| Void lfu() | 최소 빈도 사용 알고리즘  사용되었던 횟수를 바탕으로 가장 적은 페이지를 대상 페이지로 선정한다.  변수:  Int usedcnt[]: 프레임에 있는 페이지들 각각의 이전까지 사용한 횟수를 저장하는 배열  Int least: 가장 적게 사용된 페이지의 사용 횟수를 저장한다.  Int repin: 교체할 프레임의 인덱스를 저장한다.  Int sofarcnt: 페이지 교체가 발생하면 교체한 페이지의 사용횟수를 저장한다.  Int bn: 실행한 요구페이지의 개수를 나타낸다.  Int hitind: 프레임의 인덱스를 저장한다.  코드:  1. initialize()를 통해 초기화 한다.  2. 사용 횟수를 나타내는 usedcnt[]를 초기화 한다.  3. isHit()를 통해 페이지 적중, 부재를 확인한다.  4-1. 페이지 적중인 경우, getHitIndex()를 통해서 적중한 프레임의 인덱스를 찾아서 usedcnt를 증가시킨다.  4-2. 페이지 부재인 경우, 페이지 부재 횟수를 증가시킨다. 현재 실행한 요구 페이지의 수가 프레임의 크기보다 작은 경우 순서대로 프레임에 입력하고 usedcnt를 증가시킨다. 현재 실행한 요구 페이지가 프레임의 크기보다 큰 경우 usedcnt를 사용하여 가장 적게 사용된 프레임의 페이지를 대상 페이지로 선정한다. 페이지 교체 후 교체한 페이지의 이전까지 사용된 횟수를 usedcnt에 입력한다. 이후 프레임의 상태를 출력한다.  5. 페이지 부재 발생 수를 출력한다.  오류:  Bn의 초기값이 설정되어 있지 않아서 쓰레기값으로 존재하고 있었다. 그래서 실행 오류가 발생하였다. |
| Void secondChance() | 2차 기회 페이지 교체 알고리즘  특정 페이지에 접근하여 페이지 부재 없이 성공할 경우 해당 페이지를 큐의 맨 뒤로 이동하여 대상 페이지에서 제외한다.  변수:  Int usedbit[]: 페이지가 사용되었음을 표기하는 배열, 페이지 적중시 1로 증가시킨다.  Int victimptr: 대상 페이지를 표시하는 포인터  Int hitindex: 프레임의 인덱스를 저장한다.  코드:  1. initialize()를 통해 초기화 한다.  2. usedbit를 초기화 시킨다.  3. isHit()를 통해 페이지 적중, 부재를 확인한다.  4-1. 페이지 적중인 경우, 메시지를 출력하고, 해당 프레임의 페이지의 usedbit[]를 1로 변경한다.  4-2. 페이지 부재인 경우, 페이지 부재 수를 증가시키고 usedbit가 0인 것을 찾는다. uedbit가 1인 경우 0으로 감소시키고 다음 프레임의 페이지를 확인한다. 대상 페이지가 서정 및 교체되면 교체된 프레임의 페이지의 usedbit를 1로 설정한다. 이후 프레임의 상태를 출력한다.  5. 페이지 부재수를 출력한다. |
| Void main() | 무한루프 속에서 각각의 메뉴를 출력하고 선택을 기다린다. |

1-2) 제공된 코드를 이용하여, NUR 알고리즘을 구현하고, 그림 9-9 공통으로 사용할 메모리 접근 패턴을 이용하여, 그림 9-18과 같은 결과가 출력되도록 코드를 작성하고 실행 결과 화면을 보이시오. (텍스트 정보로 출력하면 됨)

1-2-1) NUR

1-2-2) 코드

1-2-3) 실행 화면

1-3) 제공된 코드를 이용하여, FIFO 변형 알고리즘 중에서 시계 알고리즘을 구현하고, 그림 9-9 공 통으로 사용할 메모리 접근패턴을 이용하여, 그림 9-21과 같은 결과가 출력되도록 코드를 작성하고 실행 결과 화면을 보이시오. (텍스트 정보로 출력하면 됨)

1-3-1) clock

1-3-2) 코드

1-3-3) 실행 화면

1-4) 표를 이용하여, 수정전 코드에 대비하여 수정한 내용을 기술하시오.

1-5) 전체 코드