운영체제 과제 #3

Buddy System & Virtual memory

2015147543 정유정

1. **개발 환경 : Linux 4.9.15**

```
p83596@p83596:~/hw3$ uname -a
Linux p83596 4.9.15-2015147543 #1 SMP Thu Mar 16 23:08:43 KST 2017 x86_64 x86_64 x86_64 GNU/Linux
```

2. **구현 방식 / 동작 과정**

먼저, 각각의 프로세스에 대해 사용할 페이지 테이블 구조체와 LRU 방식을 사용하기 위한 LRU 구조체를 만든다. PageTable 구조체는 allocation id, valid, frameIndex 정보를 담을 수 있는 64 size 배열을 가지고 있다. LRU 구조체는 연결리스트를 염두에 두고 정의하였고 각각의 구조체에는 allocation id, start , end (메모리 상에서 해당 allocation id의 시작 주소와 끝 주소)에 대한 정보와 next LRU node를 가지고 있다. LRU head를 통해 LRU 연결리스트에 접근할 수 있다. 다음은 추가적으로 정의한 함수들이다.

void allocation(PageTable *p, int AID, int num); : 할당할 allocation id와 그 개수를 받아서 PageTable에 할당해주는 함수이다.
int getValue(PageTable p, int AID); : 해당 allocation id의 valid 값을 리턴한다.
void update_LRU(LRU *head, LRU *new); : LRU 연결리스트의 끝에 새 구조체 new를 추가한다.
bool isEmpty(int memory[], int start, int end); : 메모리에 새 allocation id를 할당하기 위해 start 지점부터 end 지점까지 memory가 비어있는지 여부를 리턴한다.

다음은 위에서 정의한 구조체와 함수를 이용한 구체적인 구현 방식이다.
먼저, 프로세스의 수 pNum을 입력받아 PageTable 구조체의 pNum size 배열을 만든다. 그리고 PageTable의 valid = -1, memory = -1로 초기화한다. 메모리는 32size 배열이고, -1은 아무 값도 들어오지 않은 초기 상태라는 뜻이다. 그리고 명령어의 수 N을 입력 받아 N회수만큼 ID 할당 / 메모리 접근을 수행할 것이다. 한 사이클마다 한번씩 총 N번 PID, AccessType, FrameNum을 입력받는다.

AccesType = 1, 즉 allocation의 경우는 allocation()을 이용해 간단히 AID를 PageTable에 할당해줄 수 있다. AccesType = 0, 즉 access의 경우에는 접근하려는 AID가 현재 메모리에 존재하는지 아닌지를 먼저 확인한다. 만약 메모리에 이미 AID가 할당되어 있다면, LRU에 존재하는 해당 AID를 가진 구조체를 최신으로 업데이트 해 주기만 하면 된다. 메모리에 해당 AID가 할당되어 있지 않다면, 메모리에 해당 AID를 할당할 만한 공간이 있는지 아닌지에 따라서 해야 할 일이 달라진다. AID가 차지하는 size는 buddy system에 따라 $2^{n-1} < frameNum \leq 2^n$ 일 때 2^n 이 된다. isEmpty()를 이용해 이만큼의 공간이 있는지

확인하고, 있다면 memory와 PageTable의 값을 변경해주고 LRU에 최신으로 업데이트할 새 구조체를 만들어준다. 만약, 공간이 없다면 LRU head를 통해 LRU 연결리스트에 접근해서 가장 오랫동안 참조되지 않은 AID의 할당을 해제해 주어야 한다. Memory 값과 PageTable 값을 변경시키고, free()를 통해 해당 AID를 가진 LRU 구조체의 메모리할당을 해제한다.

3. **결과 화면**

input.txt 로 사용한 문서는 아래와 같다.

```
2       // 프로세스 수
20      // 입력되는 명령어 개수
0   1   1   16      // PID 0에서 16개의 Page를 Allocation 요청
0   1   2   12
0   1   3   22
0   1   4   14
1   1   5   11
1   1   6   9
1   1   7   20
1   1   8   10
1   1   9   14
0   0   2               // PID 0에서 Allocation ID가 2인 메모리에 접근
1   0   5
1   0   6
1   0   5
0   0   2
0   0   3
0   0   4
1   0   6
1   0   8
0   0   4
0   0   2
```

아래는 해당 input으로 memory_simulation을 진행한 결과를 스텝 별로 자른 것이다.

```
p83596@p83596:~/hw3$ ./memory_simulation < input.txt
* Input : Pid [0] Function [ALLOCATION] Alloc ID [1] Page Num [16]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> LRU :
```

PageTable에 AID 1이 할당되고, 아직 메모리에 할당되지 않았기 때문에 Valid는 0이다.

```
* Input : Pid [0] Function [ALLOCATION] Alloc ID [2] Page Num [12]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|----|----|----|----|----|----|----|----|----|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> LRU :
```

```
* Input : Pid [0] Function [ALLOCATION] Alloc ID [3] Page Num [22]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3333|33--|----|----|----|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|00--|----|----|----|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> LRU :
```

```
* Input : Pid [0] Function [ALLOCATION] Alloc ID [4] Page Num [14]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> LRU :
```

```
* Input : Pid [1] Function [ALLOCATION] Alloc ID [5] Page Num [11]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|555-|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|000-|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
>> LRU :
```

```
* Input : Pid [1] Function [ALLOCATION] Alloc ID [6] Page Num [9]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|----|----|----|----|----|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|----|----|----|----|----|----|----|----|----|----|----|
>> LRU :
```

```
* Input : Pid [1] Function [ALLOCATION] Alloc ID [7] Page Num [20]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|----|----|----|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|----|----|----|----|----|----|
>> LRU :
```

```
* Input : Pid [1] Function [ALLOCATION] Alloc ID [8] Page Num [10]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|88--|----|----|----|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|00--|----|----|----|
>> LRU :
```

```
* Input : Pid [1] Function [ALLOCATION] Alloc ID [9] Page Num [14]
>> Physical Memory :          |----|----|----|----|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> LRU :
```

```
* Input : Pid [0] Function [ACCESS] Alloc ID [2] Page Num [12]
>> Physical Memory :          |2222|2222|2222|2222|----|----|----|----|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> LRU :                      2:0-15
```

2번 AID가 메모리에 할당돼서 Valid가 1이 되고 LRU에 AID 2가 추가되었다.

```
* Input : Pid [1] Function [ACCESS] Alloc ID [5] Page Num [11]
>> Physical Memory :          |2222|2222|2222|2222|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1110|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> LRU :                      2:0-15 5:16-31
```

5번 AID가 메모리에 할당돼서 Valid가 1이 되고 LRU에 AID 5가 추가되었다.

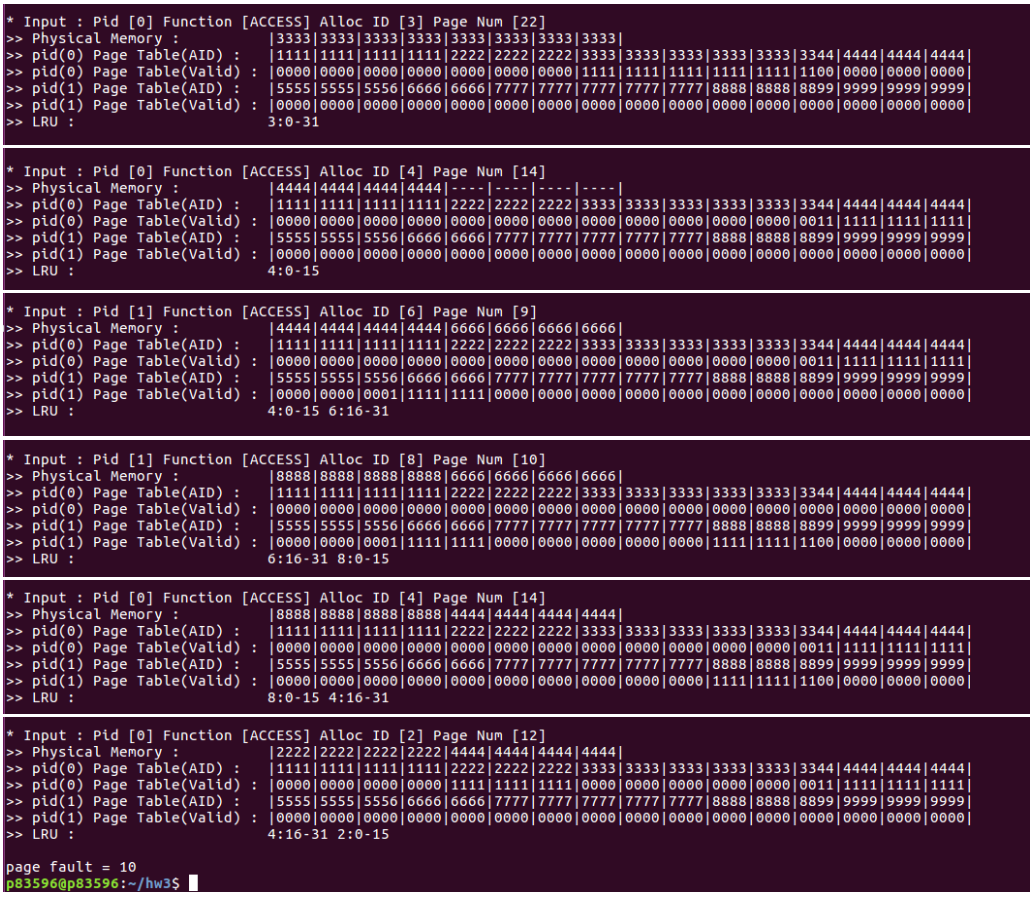
```
* Input : Pid [1] Function [ACCESS] Alloc ID [6] Page Num [9]
>> Physical Memory :          |6666|6666|6666|6666|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> LRU :                      5:16-31 6:0-15
```

6번 AID가 메모리에 할당돼야 하는데 할당할 공간이 없으므로 가장 오랫동안 참조되지 않은 2번 AID를 메모리에서 쫓아내고 AID 6이 할당되었다. 2번 AID의 할당이 해제되었기 때문에 PageTable에서 Valid가 0이 되었다.

```
* Input : Pid [1] Function [ACCESS] Alloc ID [5] Page Num [11]
>> Physical Memory :          |6666|6666|6666|6666|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> LRU :                      6:0-15 5:16-31
```

5번 AID가 메모리에 이미 할당돼 있는 상태에서 재 참조되었으므로 LRU에서 AID 5가 최신으로 업데이트되었다.

```
* Input : Pid [0] Function [ACCESS] Alloc ID [2] Page Num [12]
>> Physical Memory :          |2222|2222|2222|2222|5555|5555|5555|5555|
>> pid(0) Page Table(AID) :   |1111|1111|1111|1111|2222|2222|2222|3333|3333|3333|3333|3333|3344|4444|4444|4444|
>> pid(0) Page Table(Valid) : |0000|0000|0000|0000|1111|1111|1111|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> pid(1) Page Table(AID) :   |5555|5555|5556|6666|6666|7777|7777|7777|7777|7777|8888|8888|8899|9999|9999|9999|
>> pid(1) Page Table(Valid) : |1111|1111|1110|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|0000|
>> LRU :                      5:16-31 2:0-15
```



4. **토의**

   Buddy system을 이용한 할당은 항상 allocaton id가 필요로 하는 크기보다 더 큰 size를 할당하고, 내부에 남는 공간에 다른 id가 할당될 수 없기 때문에 할당하려는 allocation id의 frameNum이 클수록 internal fragments가 커질 것이다. 또 참조되는 순서에 따라 external fragments도 발생할 수 있다.

5. **어려웠던 점**

   제일 먼저 고민되었던 것은 LRU를 어떻게 구현해야 할 것인가 였다. 배열로 만든다면 LRU가 업데이트되거나 제거될 때마다 배열 인덱스를 전부 하나씩 앞으로, 뒤로 밀어줘야 할 텐데 번거로울 것 같았다. 그래서 연결리스트를 이용하면 될 거라고 생각했는데, C언어를 오랜만에 했더니 구조체도 낯설고 포인터도 낯설어서 조금 어렵게 느껴졌다. 포인터를 사용하다 보니 세그멘테이션 오류가 뜨는데 왜 뜨는건지 찾기도 힘들었다. 또, PageTable에 AID를 할당하는 것도 memory에 해당 AID를 새로 할당하고 LRU를 갱신해주는 것까지도 문제가 없었는데 문제는 memory에 할당을 위한 공간이 없을 때 LRU를 읽고 삭제할 AID를 정하고 메모리에서 해제하고 PageTable을 갱신하는 것이었다. 처리해야 될 일이 너무 많아서 어떤 순서로 진행해야 가장 코드가 깔끔해 질까 고민을 한 부분이다. 그리고 사실 모든 작업을 함수를 만들어서 함수 내에서 처리하고 싶었는데 포인터 사용이 익숙하지 않아서, 특히 배열 전체를 포인터로 함수에 넘기는게 생각처럼 안돼서,

결국 몇몇 함수만 만들고 나머지는 메인 함수 내에서 처리하게 하였다.