

# 과제 #3

---

3-1. Demanded Paging

3-2. Demanded Paging with 2-Level Hierarchical Page Table

# 자율 진도표

3-1 단계	완료 여부
Frame 변수와 Pte 변수의 연결성 확인	○
PAS에서 Page table 위치, Free frame 정의	○
Allocated, Pagefault, Reference 값 계산	○
3-1 JOTA 채점 완료	○

3-2 단계	완료 여부
1-level, 2level Page table 개념 적용	○
한 개의 frame으로 2-level page table 접근 (1-level)	○
2-level page table 에서 접근할 frame 계산	○
3-2 JOTA 채점 완료	○

## 코드 : 변수 선언부(process raw)

```
139 ┌ typedef struct{  
140 │     int pid;  
141 │     int ref_len;  
142 │     unsigned char *ref;  
143 │     int pagefault_cnt;  
144 │     int reference_cnt;  
145 │     struct list_head list;  
146 └ } process_raw;
```

- pagefault\_cnt : Pagefault가 일어난 횟수
- reference\_cnt : reference가 일어난 횟수
- list : list구조체 이용하기 위한 연결고리

## 코드 : 변수 선언부(전역)

```
152 process_raw *cur, *next;  
153 LIST_HEAD(job);  
154 frame * pas;  
155 int free_frame;           // free상태 frame index  
156 int proc_num = 0;        // process의 개수
```

- free\_frame : free상태의 frame index의 시작위치를 저장
- proc\_num : process\_raw의 개수 저장

## 코드 : 함수(Load\_proc)

```
158 void Load_proc(){
159     process_raw data;
160     while(fread(&data, sizeof(int) * 2, 1, stdin) != 0){
161         cur = malloc(sizeof(process_raw));
162         cur->pid = data.pid;
163         cur->ref_len = data.ref_len;
164         cur->pagefault_cnt = 0;
165         cur->reference_cnt = 0;
166         cur->ref = malloc(sizeof(unsigned char) * cur->ref_len);
167         for(int i=0; i<cur->ref_len; i++){
168             fread(&cur->ref[i], sizeof(unsigned char), 1, stdin);
169         }
170         INIT_LIST_HEAD(&cur->list);
171         list_add_tail(&cur->list, &job);
172         proc_num++;
173     }
174 }
```

- 파일로부터 process\_raw의 입력을 받아 list에 연결해주는 함수
- While문 1번 돌때마다 proc\_num 변수를 1만큼 증가

## 코드 : 함수(init\_pas)

```
176 void init_pas(){
177     pas = (frame*)malloc(PAS_SIZE);
178     free_frame = proc_num;
179     for(int i=0; i<free_frame; i++){
180         pte *cur_pte = (pte*)&pas[i];
181         for(int j=0; j<8; j++){
182             cur_pte[j].vflag = PAGE_INVALID;
183             cur_pte[j].ref = 0;
184         }
185     }
186 }
```

- PAS를 초기화하는 함수
- frame \* 형으로 PAS\_SIZE만큼을 frame\* 단위로 malloc  
-> 메모리는 8192b, 배열 크기는 256임.
- 위 코드는 3-2이므로 할당 된 1-level page table의 크기가 1 frame임  
-> 따라서 free\_frame의 첫 값은 proc\_num과 같음. (3-1번에선 proc\_num \* 8)
- 1-level page table의 vflag와 ref를 0으로 초기화

# 코드 : 함수(simulator)

```
188 void simulator(){
189     pte * cur_pte_lev1;
190     pte * cur_pte_lev2;
191     int ref_cnt = 0;
192     while(1){
193         int isbreak = 1;
194         list_for_each_entry(cur, &job, list){
195             // 모든 수행했을 때
196             if(cur->ref_len <= ref_cnt){
197                 continue;
198             }
199             else isbreak = 0;
200             cur_pte_lev1 = (pte*)&pas[cur->pid];
201             // level 1 page fault
202             int ref_num = cur->ref[ref_cnt];
203             if(cur_pte_lev1[ref_num / 8].vflag == PAGE_INVALID){
204                 // out of range
205                 if(free_frame >= MAX_REFERENCES){
206                     printf("Out of memory!!\n");
207                     isbreak = 1;
208                     break;
209                 }
210                 cur_pte_lev1[ref_num / 8].vflag = PAGE_VALID;
211                 cur_pte_lev1[ref_num / 8].frame = free_frame++;
212                 cur->pagefault_cnt++;
213             }
214             // not level 1 page fault -> 아무 동작 하지 않음.
215             else{}
216
217             cur_pte_lev2 = (pte*)&pas[cur_pte_lev1[ref_num / 8].frame];
218             // level 2 page fault
219             if(cur_pte_lev2[ref_num % 8].vflag == PAGE_INVALID){
220                 if(free_frame >= MAX_REFERENCES){
221                     printf("Out of memory!!\n");
222                     isbreak = 1;
223                     break;
224                 }
225                 cur_pte_lev2[ref_num % 8].vflag = PAGE_VALID;
226                 cur_pte_lev2[ref_num % 8].frame = free_frame++;
227                 cur_pte_lev2[ref_num % 8].ref = 1;
228                 cur->pagefault_cnt++;
229                 cur->reference_cnt++;
230             }
231             // not level 2 page fault
232             else{
233                 cur_pte_lev2[ref_num % 8].ref++;
234                 cur->reference_cnt++;
235             }
236         }
237         if(isbreak) break;
238         ref_cnt++;
239     }
240 }
```

- 1-level, 2-level에 대한 pte \* 형 변수 선언
- ref\_cnt는 process\_raw의 ref배열의 접근할 인덱스 정보
- 무한루프 안에는 isbreak가 1로 초기화 되어있음.
  - > 모든 process\_raw가 ref\_len만큼 수행했거나, OOM이 발생하는 경우 while문 탈출할 조건으로 사용
- cur\_pte\_lev1 = (pte\*)&pas[cur->pid];
  - > pas의 cur->pid의 인덱스의 시작지점을 cur\_pte\_lev1으로 가리킴
  - > ref\_num = cur->ref[ref\_cnt]로 초기화
  - > cur\_pte\_lev1[ref\_num / 8]의 값을 접근
- cur\_pte\_lev2는 pas의 cur\_pte\_lev1[ref\_num / 8]의 frame정보를 시작지점으로 가짐.
  - > cur\_pte\_lev2[ref\_num % 8]의 값을 접근

# 코드 : 함수(simulator)

```
188 void simulator(){
189     pte * cur_pte_level1;
190     pte * cur_pte_level2;
191     int ref_cnt = 0;
192     while(1){
193         int isbreak = 1;
194         list_for_each_entry(cur, &job, list){
195             // 코드 수정할 때
196             if(cur->ref_len <= ref_cnt){
197                 continue;
198             }
199             else isbreak = 0;
200             cur_pte_level1 = (pte*)&pas[cur->pid];
201             // level 1 page fault
202             int ref_num = cur->ref[ref_cnt];
203             if(cur_pte_level1[ref_num / 8].vflag == PAGE_INVALID){
204                 // out of range
205                 if(free_frame >= MAX_REFERENCES){
206                     printf("Out of memory!!\n");
207                     isbreak = 1;
208                     break;
209                 }
210                 cur_pte_level1[ref_num / 8].vflag = PAGE_VALID;
211                 cur_pte_level1[ref_num / 8].frame = free_frame++;
212                 cur->pagefault_cnt++;
213             }
214             // not level 1 page fault -> 아무 동작 하지 않음.
215             else{}
216
217             cur_pte_level2 = (pte*)&pas[cur_pte_level1[ref_num / 8].frame];
218             // level 2 page fault
219             if(cur_pte_level2[ref_num % 8].vflag == PAGE_INVALID){
220                 if(free_frame >= MAX_REFERENCES){
221                     printf("Out of memory!!\n");
222                     isbreak = 1;
223                     break;
224                 }
225                 cur_pte_level2[ref_num % 8].vflag = PAGE_VALID;
226                 cur_pte_level2[ref_num % 8].frame = free_frame++;
227                 cur_pte_level2[ref_num % 8].ref = 1;
228                 cur->pagefault_cnt++;
229                 cur->reference_cnt++;
230             }
231             // not level 2 page fault
232             else{
233                 cur_pte_level2[ref_num % 8].ref++;
234                 cur->reference_cnt++;
235             }
236         }
237         if(isbreak) break;
238         ref_cnt++;
239     }
240 }
```

- 1. 1-level page table의 pagefault가 일어나면
  - 1) oom인지 확인 -> 맞다면 경고문 출력 후 무한루프 탈출
  - 2) 1-level page table에 pte의 vflag, frame값을 갱신
  - 3) 현재 접근한 process\_raw의 pagefault\_cnt값을 1증가
- 2. 1-level page table에서 pagefault가 일어나지 않으면  
-> 아무것도 하지않음!!
- 3. 2-level page table의 pagefault가 일어나면  
-> 기본적으로 1-level page table과 동일한 연산  
+ 2-level page table에 pte의 ref값을 1로 초기화
- 4. 2-level page table에서 pagefault가 일어나지 않으면  
-> 2-level page table에 접근한 pte의 ref 값을 1증가  
+ 현재 접근한 process\_raw의 reference 값을 1증가



# 코드 : 함수(result\_print)

```
242 void result_print(){
243     int allocated = 0;
244     int pagefault = 0;
245     int reference = 0;
246
247     list_for_each_entry(cur, &job, list){
248         allocated += (cur->pagefault_cnt + 1);
249         pagefault += cur->pagefault_cnt;
250         reference += cur->reference_cnt;
251         // process_raw 정보 출력
252         printf("*** Process %03d: Allocated Frames=%03d PageFaults/References=%03d/%03d\n",
253             cur->pid, cur->pagefault_cnt + 1, cur->pagefault_cnt, cur->reference_cnt);
254
255         pte *cur_pte_lev1 = (pte*)&pas[cur->pid];
256         for(int i=0; i<8; i++){
257             if(cur_pte_lev1[i].vflag == PAGE_VALID){
258                 printf("(L1PT) %03d -> %03d\n", i, cur_pte_lev1[i].frame);
259                 pte *cur_pte_lev2 = (pte*)&pas[cur_pte_lev1[i].frame];
260                 for(int j=0; j<8; j++){
261                     if(cur_pte_lev2[j].vflag == PAGE_VALID){
262                         printf("(L2PT) %03d -> %03d REF=%03d\n",
263                             i*8+j, cur_pte_lev2[j].frame, cur_pte_lev2[j].ref);
264                     }
265                 }
266             }
267         }
268     }
269     // 전체 정보 출력
270     printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n",
271         allocated, pagefault, reference);
272 }
```


- 출력할 정보인 allocated, pagefault, reference를 0으로 초기화
- Job list를 순회하면서 현재 접근한 process\_raw의 pagefault\_cnt, reference\_cnt값을 이용하여 allocated는 cur->pagefault\_cnt+1(1-level page table 크기), pagefault와 reference는 cur->pagefault\_cnt, reference\_cnt를 더해줌
- 현재 process\_raw의 1-level page table의 위치에서 pte 크기만큼 순회하면서
  - 1) cur\_pte\_lev1[i]가 valid하다면 정보 출력
  - 2) 2-level page table의 초기위치를 valid한 값의 frame에서 시작
  - 3) frame크기만큼 순회하면서
    - 3-1) cur\_pte\_lev2[j]가 valid하다면 정보 출력
- 계산된 allocated, pagefault, reference값을 출력

## 코드 : 함수(mem\_free)

```
274 void mem_free(){  
275     list_for_each_entry_safe(cur, next, &job, list){  
276         list_del(&cur->list);    // job_q에서 연결 해제  
277         free(cur->ref);          // ref 할당 해제  
278         free(cur);               // process 할당 해제  
279     }  
280     free(pas);  
281 }
```

- 할당 한 데이터 해제
  1. process\_raw \* 형 변수 -> job에서 연결을 해제한 후  
cur->ref와 cur 을 free
  2. frame \* 형 변수 pas free

## 코드 : main문

```
282  int main(){  
283     Load_proc();  
284     init_pas();  
285     simulator();  
286     result_print();  
287     mem_free();  
288     return 0;  
289 }
```

- 1. Load\_proc()으로 파일 읽어들이м
- 2. init\_pas()로 pas 생성 및 여러 값 초기화
- 3. simulator()로 page table의 값 연산
- 4. result\_print()로 연산 된 page table의 정보 출력
- 5. mem\_free()로 할당 한 메모리 해제

# test3.bin 수행결과

## 3-1번

```
ubuntu@201716443:~/hw3$ cat test3.bin | ./os3_1
** Process 000: Allocated Frames=013 PageFaults/References=005/008
017 -> 024 REF=001
050 -> 022 REF=001
051 -> 019 REF=002
052 -> 016 REF=002
053 -> 021 REF=002
** Process 001: Allocated Frames=013 PageFaults/References=005/007
004 -> 018 REF=002
005 -> 023 REF=001
006 -> 020 REF=001
007 -> 017 REF=002
021 -> 025 REF=001
Total: Allocated Frames=026 Page Faults/References=010/015
ubuntu@201716443:~/hw3$ █
```

## 3-2번

```
ubuntu@201716443:~/hw3$ cat test3.bin | ./os3_2
** Process 000: Allocated Frames=008 PageFaults/References=007/008
(L1PT) 002 -> 012
(L2PT) 017 -> 013 REF=001
(L1PT) 006 -> 002
(L2PT) 050 -> 010 REF=001
(L2PT) 051 -> 007 REF=002
(L2PT) 052 -> 003 REF=002
(L2PT) 053 -> 009 REF=002
** Process 001: Allocated Frames=008 PageFaults/References=007/007
(L1PT) 000 -> 004
(L2PT) 004 -> 006 REF=002
(L2PT) 005 -> 011 REF=001
(L2PT) 006 -> 008 REF=001
(L2PT) 007 -> 005 REF=002
(L1PT) 002 -> 014
(L2PT) 021 -> 015 REF=001
Total: Allocated Frames=016 Page Faults/References=014/015
ubuntu@201716443:~/hw3$ █
```

# JOTA 채점 결과

## 3-1번

Submission of 2021운영체제 과제 3-1 by os201716443

[View source](#)  
[Resubmit](#)

### Compilation Warnings

```
oshw31c.c: In function 'Load_proc':  
oshw31c.c:168:13: warning: ignoring return value of 'fread', declared with attribute warn_unused_result [-Wunused-result]  
    fread(&cur->ref[i], sizeof(unsigned char), 1, stdin);
```

### Execution Results

✓✓✓✓✓

- Test case #1: AC [0.041s, 820.00 KB] (2/2)
- Test case #2: AC [0.045s, 820.00 KB] (2/2)
- Test case #3: AC [0.040s, 820.00 KB] (2/2)
- Test case #4: AC [0.049s, 820.00 KB] (2/2)
- Test case #5: AC [0.041s, 820.00 KB] (2/2)

Resources: 0.215s, 820.00 KB  
Final score: 10/10 (10.0/10 points)

## 3-2번

Submission of 2021운영체제 과제 3-2 by os201716443

[View source](#)  
[Resubmit](#)

### Compilation Warnings

```
oshw32c.c: In function 'Load_proc':  
oshw32c.c:168:13: warning: ignoring return value of 'fread', declared with attribute warn_unused_result [-Wunused-result]  
    fread(&cur->ref[i], sizeof(unsigned char), 1, stdin);
```

### Execution Results

✓✓✓✓✓

- Test case #1: AC [0.045s, 820.00 KB] (2/2)
- Test case #2: AC [0.032s, 820.00 KB] (2/2)
- Test case #3: AC [0.053s, 820.00 KB] (2/2)
- Test case #4: AC [0.048s, 820.00 KB] (2/2)
- Test case #5: AC [0.040s, 820.00 KB] (2/2)

Resources: 0.218s, 820.00 KB  
Final score: 10/10 (10.0/10 points)

# 어려웠던 점 및 해결방안

---

- 처음에 vflag값이 이상한 값이 저장되어 !=PAGE\_VALID로 접근할 때 접근이 되었었다.  
-> init\_pas()함수를 이용해 vflag와 ref값을 각각 PAGE\_INVALID, 0으로 초기화해서 해결했다.
- 3-2번을 해결 할 때 3-1번과 다르게 page table의 크기가 1frame인 것을 제대로 인지하지 못해 삽질을 했었다.