

## 운영체제 HW3 보고서

컴퓨터공학부 202219352 김담은

### # 자율 진도표

단계	완료 여부
3-1-0. 과제 설명 영상 보기	O
3-1-1. 프로세스 정보 읽어오기	O
3-1-2. pas 메모리 할당	O
3-1-3. page_table 메모리 할당 및 초기화	O
3-1-4. pm 구조체 생성 및 초기화	O
3-1-5. start() 함수 작성 – 프로세스 순회	O
3-1-6. start() 함수 작성 – page_fault 및 출력	O
3-1-7. start() 함수 작성 – 종료 조건	O
3-1-8. print_result() 함수 작성	O
3-1. 과제 3-1 완료 (JOTA)	O
3-2-0. 과제 설명 영상 보기	O

3-2-1. os3-1.c와 중복되는 함수 작성	O
3-2-2. start() 함수 작성	O
3-2-3. Page_table 포인터 배열 삭제 및 리팩토링	O
3-2-4. page_fault_handler 함수 작성 및 리팩토링	O
3-2-5. print_result() 함수 작성	O
3-2-6. Out of memory 관련 리팩토링	O
3-2. 과제 3-1 완료 (JOTA)	O
+ os3-1.c 리팩토링	O
+ os3-2.c 리팩토링 및 RTE 문제 해결	O

## # os3-1.c

전체 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define PAGESIZE (32)
#define PAS_FRAMES (256) //fit for unsigned char frame in PTE
#define PAS_SIZE (PAGESIZE*PAS_FRAMES) //32*256 = 8192 B
#define VAS_PAGES (64)
#define VAS_SIZE (PAGESIZE*VAS_PAGES) //32*64 = 2048 B
#define PTE_SIZE (4) //sizeof(pte)
```

```

#define PAGETABLE_FRAMES (VAS_PAGES*PTE_SIZE/PAGESIZE) //64*4/32 = 8 consecutive
frames
#define PAGE_INVALID (0)
#define PAGE_VALID (1)
#define MAX_REFERENCES (256)
#define MAX_PROCESS (10)

typedef struct{
    int pid;
    int ref_len; //Less than 255
    unsigned char *references;
} process_raw;

typedef struct{
    unsigned char frame; //allocated frame
    unsigned char vflag; //valid-invalid bit
    unsigned char ref; //reference bit
    unsigned char pad; //padding
} pte; // Page Table Entry (total 4 Bytes, always)

typedef struct {
    unsigned char b[PAGESIZE];
} frame;

typedef struct{
    process_raw *process;
    int ref_idx;
    int page_fault_num;
} pm;

int process_num;
pm *process_arr;
frame *pas;
int free_frame = 0;

/* 메모리 할당 */
void init_pas() {
    pas = (frame *)malloc(PAS_SIZE);
    process_arr = (pm *)malloc(MAX_PROCESS * sizeof(pm));

```

```
}
```

```
/* page table 초기화 */
```

```
void init_pageTable(int pid) {  
    pte *cur_pte = (pte *) &pas[free_frame];  
    for(int i=0; i<VAS_PAGES; i++) {  
        cur_pte[i].frame = free_frame + i;  
        cur_pte[i].vflag = PAGE_INVALID;  
        cur_pte[i].ref = 0;  
        cur_pte[i].pad = 0;  
    }  
    free_frame += 8;  
}
```

```
/* 프로세스 정보 읽어오기 */
```

```
void load_process() {  
    //printf("load_process() start\n");  
    process_raw *cur;  
    while(1) {  
        cur = malloc(sizeof(*cur));  
        if(fread(cur, sizeof(int) * 2, 1, stdin) == 1) {  
            /* 프로세스 정보 읽어오기 */  
            // pid, ref_len 정보 읽어오기  
            cur->references = malloc(cur->ref_len);  
            //printf("%d %d\n", cur->pid, cur->ref_len);  
            // references 정보 읽어오기  
            for(int i=0; i<cur->ref_len; i++) {  
                fread(&cur->references[i], sizeof(unsigned char), 1, stdin);  
                //printf("%d ", cur->references[i]);  
            }  
            //printf("\n");  
            /* page_table 초기화 */  
            init_pageTable(cur->pid);  
            /* pm 초기화 */  
            process_arr[cur->pid].process = cur;  
            process_arr[cur->pid].ref_idx = 0;  
            process_arr[cur->pid].page_fault_num = 0;  
            process_num++;  
        }  
    }  
}
```

```

    }
    else {
        free(cur);
        break;
    }
}
//printf("load_process() end\n");
}

/* Demand Paging 수행 */
void start() {
    //printf("Start() start\n");
    while(1) {
        bool flag = true; // 종료 조건 확인 변수
        for(int i=0; i<process_num; i++) { // 프로세스 순회
            process_raw *cur = process_arr[i].process;
            int idx = process_arr[i].ref_idx;
            unsigned char page = cur->references[idx];

            // 레퍼런스를 모두 봤다면 건너뛰기
            if(idx == cur->ref_len) continue;
            flag = false; // 건너뛰지 않은 프로세스가 있는지 확인

            pte *cur_pte = (pte *) &pas[PAGETABLE_FRAMES * i];

            if(cur_pte[page].vflag == PAGE_INVALID) { // page_fault
                if(free_frame >= PAS_FRAMES) {
                    printf("Out of memory!!\n"); return;
                }
                process_arr[i].page_fault_num++;
                cur_pte[page].frame = free_frame;
                cur_pte[page].vflag = PAGE_VALID;
                cur_pte[page].ref = 1;
                //printf("[PID %02d REF: %03d] Page access %03d: PF,Allocated
Frame %03d\n", cur->pid, idx, page, cur_pte[page].frame);
                free_frame++;
            }
            else {
                cur_pte[page].ref++;
            }
        }
    }
}

```

```

        //printf("[PID %02d REF: %03d] Page access %03d: Frame %03d\n",
cur->pid, idx, page, cur_pte[page].frame);
    }
    process_arr[i].ref_idx++;
}
    if(flag) break; // 모두 건너뛰었다면 종료
}
//printf("Start() end\n");
}

/* 결과 출력 */
void print_result() {
    // 결과 출력용 변수
    int total_allocated_frame = 0;
    int total_page_fault = 0;
    int total_references = 0;

    for(int i=0; i<process_num; i++) { // 프로세스
        process_raw *cur = process_arr[i].process;
        int page_fault = process_arr[i].page_fault_num;
        int refer_num = process_arr[i].ref_idx;

        printf("** Process %03d: Allocated Frames=%03d
PageFaults/References=%03d/%03d\n", cur->pid, page_fault+8, page_fault,
refer_num);

        pte *cur_pte = (pte *) &pas[PAGETABLE_FRAMES * i];
        for(int j=0; j<VAS_PAGES; j++) { // pte
            if(cur_pte[j].vflag == PAGE_INVALID) continue;
            printf("%03d -> %03d REF=%03d\n", j, cur_pte[j].frame,
cur_pte[j].ref);
        }

        total_allocated_frame += (page_fault + 8);
        total_page_fault += page_fault;
        total_references += refer_num;
    }
}

```

```

        printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n",
total_allocated_frame, total_page_fault, total_references);
    }

/* 동적 메모리 해제 */
void free_memory() {
    for(int i=0; i<process_num; i++) {
        free(process_arr[i].process->references);
        free(process_arr[i].process);
    }
    free(process_arr);
    free(pas);
}

int main(void){

    init_pas();    // 초기 메모리 할당
    load_process(); // 프로세스 정보 읽어오기
    start();       // Demand Paging
    print_result(); // 결과 출력
    free_memory(); // 메모리 해제

    return 0;
}

```

## 주요 코드 부분 서술

```
34  typedef struct{
35      process_raw *process;
36      int ref_idx;
37      int page_fault_num;
38  } pm;
39
40  int process_num;
41  pm *process_arr;
```

- pm 구조체를 만들어 프로세스 관리

```
100  while(1) {
101      bool flag = true; // 종료 조건 확인 변수
102      for(int i=0; i<process_num; i++) { // 프로세스 순회
103          process_raw *cur = process_arr[i].process;
104          int idx = process_arr[i].ref_idx;
105          unsigned char page = cur->references[idx];
106
107          // 레퍼런스를 모두 봤다면 건너뛰기
108          if(idx == cur->ref_len) continue;
109          flag = false; // 건너뛰지 않은 프로세스가 있는지 확인
110
111          pte *cur_pte = (pte *) &pas[PAGETABLE_FRAMES * i];
112
113          if(cur_pte[page].vflag == PAGE_INVALID) { // page_fault
114              if(free_frame >= PAS_FRAMES) {
115                  printf("Out of memory!!\n"); return;
116              }
117              process_arr[i].page_fault_num++;
118              cur_pte[page].frame = free_frame;
119              cur_pte[page].vflag = PAGE_VALID;
120              cur_pte[page].ref = 1;
121              //printf("[PID %02d REF: %03d] Page access %03d: PF, Allocated Frame %03d\n", cur->pid, idx, page, cur_pte[page].frame);
122              free_frame++;
123          }
124          else {
125              cur_pte[page].ref++;
126              //printf("[PID %02d REF: %03d] Page access %03d: Frame %03d\n", cur->pid, idx, page, cur_pte[page].frame);
127          }
128          process_arr[i].ref_idx++;
129      }
130      if(flag) break; // 모두 건너뛰었다면 종료
131  }
```

- start() 함수 - demand paging 수행 부분
- 프로세스를 0, 1, 2, .. 순회하며 레퍼런스를 하나씩 확인하고, page\_fault 처리 확인



```

135 /* 결과 출력 */
136 void print_result() {
137     // 결과 출력용 변수
138     int total_allocated_frame = 0;
139     int total_page_fault = 0;
140     int total_references = 0;
141
142     for(int i=0; i<process_num; i++) { // 프로세스
143         process_raw *cur = process_arr[i].process;
144         int page_fault = process_arr[i].page_fault_num;
145         int refer_num = process_arr[i].ref_idx;
146
147         printf("*** Process %03d: Allocated Frames=%03d PageFaults/References=%03d/%03d\n", cur->pid, page_fault+8, page_fault, refer_num);
148
149         pte *cur_pte = (pte *) &pas[PAGETABLE_FRAMES * i];
150         for(int j=0; j<VAS_PAGES; j++) { // pte
151             if(cur_pte[j].vflag == PAGE_INVALID) continue;
152             printf("%03d -> %03d REF=%03d\n", j, cur_pte[j].frame, cur_pte[j].ref);
153         }
154
155         total_allocated_frame += (page_fault + 8);
156         total_page_fault += page_fault;
157         total_references += refer_num;
158     }
159
160     printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n", total_allocated_frame, total_page_fault, total_references);
161 }

```

- print\_result() 함수, 결과 출력 부분
- 결과 출력용 변수를 선언하고 각 프로세스를 순회하며 정보를 갱신 후 출력

## test3.bin 수행 결과

```

ubuntu
hw3 > C os3-1.c x C os3-2.c
hw3 > C os3-1.c
131     }
132     //printf("Start() end\n");
133 }
134
135 /* 결과 출력 */
136 void print_result() {
137     // 결과 출력용 변수
138     int total_allocated_frame = 0;
139     int total_page_fault = 0;
140     int total_references = 0;
141
142     for(int i=0; i<process_num; i++) { // 프로세스
143         process_raw *cur = process_arr[i].process;
144         int page_fault = process_arr[i].page_fault_num;
145         int refer_num = process_arr[i].ref_idx;
146
147         printf("*** Process %03d: Allocated Frames=%03d PageFaults/References=%03d/%03d\n", cur->pid, page_fault+8, page_fault, refer_num);
148
149         pte *cur_pte = (pte *) &pas[PAGETABLE_FRAMES * i];
150         for(int j=0; j<VAS_PAGES; j++) { // pte
151             if(cur_pte[j].vflag == PAGE_INVALID) continue;
152             printf("%03d -> %03d REF=%03d\n", j, cur_pte[j].frame, cur_pte[j].ref);
153         }
154
155         total_allocated_frame += (page_fault + 8);
156         total_page_fault += page_fault;
157         total_references += refer_num;
158     }
159
160     printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n", total_allocated_frame, total_page_fault, total_references);
161 }

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ubuntu@jcode-client1-222:~/hw3$ gcc os3-1.c && cat test3.bin | ./a.out
** 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@jcode-client1-222:~/hw3$

```

## JOTA 운영체제 과제 3-1 캡처

The screenshot shows a web browser window displaying a submission page for a JOTA (Jongnam Open Training Academy) operating system assignment. The page title is "os202219352의 운영체제 과제 3-1 제출". The browser address bar shows "jota.jnu.ac.kr/submission/13630". The page has a dark header with the DM::OJ logo and navigation links: PROBLEMS, SUBMISSIONS, USERS, CONTESTS, ABOUT. The submission details include the user ID "os202219352", the assignment name "운영체제 과제 3-1 제출", and the submission time "2024년 6월 7일 오전 5시 56분". Below this, there is a "View source" link and a "다시 제출" (Resubmit) button. The "컴파일 경고" (Compiler Warnings) section shows a warning about ignoring the return value of 'fread'. The "Execution Results" section shows five test cases, all passing with a score of 10/10. The "Resources" section shows a runtime of 0.018s and a memory usage of 1.02 MB. The page footer includes a timestamp "운영체제과제 - 23일 03:03:31" and a note "proudly powered by DM::OJ | 한국어 (ko)".

os202219352의 운영체제 과제 3-1 제출

2024년 6월 7일 오전 5시 56분

View source  
다시 제출

컴파일 경고

```
oshw31c.c: In function 'load_process':  
oshw31c.c:76:17: warning: ignoring return value of 'fread' declared with attribute 'warn_unused_result' [-Wunused-result]  
76 | fread(&cur->references[i], sizeof(unsigned char), 1, stdin);  
   | ^~~~~~
```

Execution Results

✓✓✓✓✓

> Test case #1: AC [0.004s, 1.02 MB] (2/2)  
> Test case #2: AC [0.004s, 1.02 MB] (2/2)  
> Test case #3: AC [0.003s, 1.02 MB] (2/2)  
> Test case #4: AC [0.003s, 1.02 MB] (2/2)  
> Test case #5: AC [0.003s, 1.02 MB] (2/2)

Resources: 0.018s, 1.02 MB  
Maximum single-case runtime: 0.004s  
Final score: 10/10 (10.0/10 points)

운영체제과제 - 23일 03:03:31

proudly powered by DM::OJ | 한국어 (ko)

## 과제 수행 시 어려웠던 점 및 해결 방안

- 어려웠던 점1: 캐스팅 관련 내용이 처음이라 어려웠다.
- 해결방안1: 바이트 수를 생각하며 대조해보았고, 관련 내용을 찾아보며 사용법을 익혔다.
- 어려웠던 점2: 프로세스, 페이지 테이블, 실제 물리 메모리, pte의 동작 과정을 이해하는 데에 어려움이 있었다.
- 해결방안2: 동작 순서를 천천히 생각하며 고민해보았다. 동작 과정을 떠올리는 과정에서 시간이 걸렸으나, 동작 과정을 이해하니 금방 코드를 작성할 수 있었다.

## # os3-2.c

전체 코드

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define PAGESIZE (32)
#define PAS_FRAMES (256) //fit for unsigned char frame in PTE
#define PAS_SIZE (PAGESIZE*PAS_FRAMES) //32*256 = 8192 B
#define VAS_PAGES (64)
#define VAS_SIZE (PAGESIZE*VAS_PAGES) //32*64 = 2048 B
#define PTE_SIZE (4) //sizeof(pte)
#define PAGETABLE_FRAMES (VAS_PAGES*PTE_SIZE/PAGESIZE) //64*4/32 = 8 consecutive frames
#define PAGE_INVALID (0)
#define PAGE_VALID (1)
#define MAX_REFERENCES (256)
#define MAX_PROCESS (10)

typedef struct{
    int pid;
    int ref_len; //Less than 255
    unsigned char *references;
} process_raw;

typedef struct{
    unsigned char frame; //allocated frame
    unsigned char vflag; //valid-invalid bit
    unsigned char ref; //reference bit
    unsigned char pad; //padding
} pte; // Page Table Entry (total 4 Bytes, always)

typedef struct {
    unsigned char b[PAGESIZE];
} frame;
```

```

typedef struct{
    process_raw *process;
    int ref_idx;
    int page_fault_num;
} pm;

int process_num = 0;
pm *process_arr;
frame *pas;
pte *page_table[MAX_PROCESS];
int free_frame = 0;

/* 메모리 할당 */
void init_pas() {
    pas = (frame *)malloc(PAS_SIZE);
    process_arr = (pm *)malloc(MAX_PROCESS * sizeof(pm));
}

/* page table 초기화 */
bool init_pageTable() {
    // out of memory 확인
    if(free_frame >= PAS_FRAMES) return false;
    // 1 frame, 8 pte 할당
    pte *cur_pte = (pte *) &pas[free_frame];
    for(int i=0; i<PAGETABLE_FRAMES; i++) {
        cur_pte[i].frame = -1;
        cur_pte[i].vflag = PAGE_INVALID;
        cur_pte[i].ref = 0;
        cur_pte[i].pad = 0;
    }
    return true;
}

/* page_fault 핸들러 */
bool page_fault_handler(pte *cur_pte, int pte_idx) {
    // out of memory 확인
    if(free_frame >= PAS_FRAMES) return false;
    cur_pte[pte_idx].frame = free_frame;

```

```

    cur_pte[pte_idx].vflag = PAGE_VALID;
    cur_pte[pte_idx].ref = 1;
    free_frame++;
    return true;
}

/* 프로세스 정보 읽어오기 */
void load_process() {
    //printf("load_process() start\n");
    process_raw *cur;
    while(1) {
        cur = malloc(sizeof(*cur));
        if(fread(cur, sizeof(int) * 2, 1, stdin) == 1) {
            /* 프로세스 정보 읽어오기 */
            // pid, ref_len 정보 읽어오기
            cur->references = malloc(cur->ref_len);
            //printf("%d %d\n", cur->pid, cur->ref_len);
            // references 정보 읽어오기
            for(int i=0; i<cur->ref_len; i++) {
                fread(&cur->references[i], sizeof(unsigned char), 1, stdin);
                //printf("%d ", cur->references[i]);
            }
            //printf("\n");
            /* page_table 초기화 */
            init_pageTable();
            free_frame++;
            /* pm 초기화 */
            process_arr[cur->pid].process = cur;
            process_arr[cur->pid].ref_idx = 0;
            process_arr[cur->pid].page_fault_num = 0;
            process_num++;

        }
        else {
            free(cur);
            break;
        }
    }
    //printf("load_process() end\n");
}

```

```
}
```

```
/* Demand Paging with 2-level Hierarchical Page Table 수행 */
void start() {
    //printf("Start() start\n");
    while(1) {
        bool flag = true; // 종료 조건 확인 변수

        for(int i=0; i<process_num; i++) {
            process_raw *cur = process_arr[i].process;
            int idx = process_arr[i].ref_idx;

            // 레퍼런스를 모두 봤다면 건너뛴
            if(idx >= cur->ref_len) continue;
            flag = false;

            //printf("[PID %02d REF: %03d] Page access %03d: ", cur->pid, idx,
            cur->references[idx]);

            /* L1PT 확인 */
            pte *l1pt = (pte *)&pas[cur->pid];
            int l1pt_idx = cur->references[idx] / PAGETABLE_FRAMES;
            // page_fault인 경우
            if(l1pt[l1pt_idx].vflag == PAGE_INVALID) {
                if(!init_pageTable()) { // 페이지 테이블 추가
                    printf("Out of memory!!\n");
                    return;
                }
                if(!page_fault_handler(l1pt, l1pt_idx)) { // page_fault 핸들러
                    printf("Out of memory!!\n");
                    return;
                }
                process_arr[i].page_fault_num++;
                //printf("(L1PT) PF,Allocated Frame %03d -> %03d,", l1pt_idx,
                l1pt[l1pt_idx].frame);
            }
            else {
                l1pt[l1pt_idx].ref++;
                //printf("(L1PT) Frame %03d,", l1pt[l1pt_idx].frame);
            }
        }
    }
}
```

```

    }

    /* L2PT 확인 */
    pte *l2pt = (pte *) &pas[l1pt[l1pt_idx].frame];
    int l2pt_idx = cur->references[idx] % PAGETABLE_FRAMES;
    // page_fault인 경우
    if(l2pt[l2pt_idx].vflag == PAGE_INVALID) {
        if(!page_fault_handler(l2pt, l2pt_idx)) { // page_fault 핸들러
            printf("Out of memory!!\n");
            return;
        }
        process_arr[i].page_fault_num++;
        //printf("(L2PT) PF,Allocated Frame %03d\n",
l2pt[l2pt_idx].frame);
    }
    else {
        l2pt[l2pt_idx].ref++;
        //printf("(L2PT) Frame %03d\n", l2pt[l2pt_idx].frame);
    }
    process_arr[i].ref_idx++;
}
if(flag) break;
}
//printf("Start() end\n");
}

/* 결과 출력 */
void print_result() {
    // 결과 출력용 변수
    int total_allocated_frame = 0;
    int total_page_fault = 0;
    int total_references = 0;

    for(int i=0; i<process_num; i++) { // 프로세스
        process_raw *cur = process_arr[i].process;
        int page_fault = process_arr[i].page_fault_num;
        int refer_num = process_arr[i].ref_idx;

```

```

        printf("** Process %03d: Allocated Frames=%03d
PageFaults/References=%03d/%03d\n", i, page_fault+1, page_fault, refer_num);

        /* L1 PT */
        pte *l1pt = (pte *) &pas[cur->pid];
        for(int j=0; j<PAGETABLE_FRAMES; j++) {
            if(l1pt[j].vflag == PAGE_INVALID) continue; // 조건 확인
            printf("(L1PT) %03d -> %03d\n", j, l1pt[j].frame);

            /* L2 PT */
            pte *l2pt = (pte *) &pas[l1pt[j].frame];
            for(int k=0; k<PAGETABLE_FRAMES; k++) {
                if(l2pt[k].vflag == PAGE_INVALID) continue; // 조건 확인
                printf("(L2PT) %03d -> %03d REF=%03d\n", j*8+k, l2pt[k].frame,
l2pt[k].ref);
            }
        }

        total_allocated_frame += (page_fault + 1);
        total_page_fault += page_fault;
        total_references += refer_num;
    }

    printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n",
total_allocated_frame, total_page_fault, total_references);
}

/* 동적 메모리 해제 */
void free_memory() {
    for(int i=0; i<process_num; i++) {
        free(process_arr[i].process->references);
        free(process_arr[i].process);
    }
    free(process_arr);
    free(pas);
}

int main(void){

```



```

init_pas();    // 초기 메모리 할당
load_process(); // 프로세스 정보 읽어오기
start();       // Demand Paging
print_result(); // 결과 출력
free_memory(); // 메모리 해제

return 0;
}

```

## 주요 코드 부분 서술

```

129  /* L1PT 확인 */
130  pte *l1pt = (pte *)&pas[cur->pid];
131  int l1pt_idx = cur->references[idx] / PAGETABLE_FRAMES;
132  // page_fault인 경우
133  if(l1pt[l1pt_idx].vflag == PAGE_INVALID) {
134      if(!init_pageTable()) { // 페이지 테이블 추가
135          printf("Out of memory!!\n");
136          return;
137      }
138      if(!page_fault_handler(l1pt, l1pt_idx)) { // page_fault 핸들러
139          printf("Out of memory!!\n");
140          return;
141      }
142      process_arr[i].page_fault_num++;
143      //printf("(L1PT) PF,Allocated Frame %03d -> %03d", l1pt_idx, l1pt[l1pt_idx].frame);
144  }
145  else {
146      l1pt[l1pt_idx].ref++;
147      //printf("(L1PT) Frame %03d", l1pt[l1pt_idx].frame);
148  }

150  /* L2PT 확인 */
151  pte *l2pt = (pte *)&pas[l1pt[l1pt_idx].frame];
152  int l2pt_idx = cur->references[idx] % PAGETABLE_FRAMES;
153  // page_fault인 경우
154  if(l2pt[l2pt_idx].vflag == PAGE_INVALID) {
155      if(!page_fault_handler(l2pt, l2pt_idx)) { // page_fault 핸들러
156          printf("Out of memory!!\n");
157          return;
158      }
159      process_arr[i].page_fault_num++;
160      //printf("(L2PT) PF,Allocated Frame %03d\n", l2pt[l2pt_idx].frame);
161  }
162  else {
163      l2pt[l2pt_idx].ref++;
164      //printf("(L2PT) Frame %03d\n", l2pt[l2pt_idx].frame);
165  }

```

- start() 함수 - demand paging 수행 부분

- L1 PT 확인 후 L2 PT를 확인하는 부분으로, page\_fault인 경우 page\_fault\_handler 함수를 호출하는 방식으로 구현했다.

```
52  /* page table 초기화 */
53  bool init_pageTable() {
54      // out of memory 확인
55      if(free_frame >= PAS_FRAMES) return false;
56      // 1 frame, 8 pte 할당
57      pte *cur_pte = (pte *) &pas[free_frame];
58      for(int i=0; i<PAGETABLE_FRAMES; i++) {
59          cur_pte[i].frame = -1;
60          cur_pte[i].vflag = PAGE_INVALID;
61          cur_pte[i].ref = 0;
62          cur_pte[i].pad = 0;
63      }
64      return true;
65  }
66
67  /* page_fault 핸들러 */
68  bool page_fault_handler(pte *cur_pte,int pte_idx) {
69      // out of memory 확인
70      if(free_frame >= PAS_FRAMES) return false;
71      cur_pte[pte_idx].frame = free_frame;
72      cur_pte[pte_idx].vflag = PAGE_VALID;
73      cur_pte[pte_idx].ref = 1;
74      free_frame++;
75      return true;
76  }
```

- init\_pageTable, page\_fault\_handler 함수에서 모두 out of memory가 발생할 수 있기에, bool 타입을 반환하도록 해 out of memory를 확인할 수 있도록 했다.

```

173  /* 결과 출력 */
174  void print_result() {
175      // 결과 출력용 변수
176      int total_allocated_frame = 0;
177      int total_page_fault = 0;
178      int total_references = 0;
179
180      for(int i=0; i<process_num; i++) { // 프로세스
181          process_raw *cur = process_arr[i].process;
182          int page_fault = process_arr[i].page_fault_num;
183          int refer_num = process_arr[i].ref_idx;
184
185          printf("*** Process %03d: Allocated Frames=%03d PageFaults/References=%03d/%03d\n", i, page_fault+1, page_fault, refer_num);
186
187          /* L1 PT */
188          pte *l1pt = (pte *) &pas[cur->pid];
189          for(int j=0; j<PAGETABLE_FRAMES; j++) {
190              if(l1pt[j].vflag == PAGE_INVALID) continue; // 조건 확인
191              printf("(L1PT) %03d -> %03d\n", j, l1pt[j].frame);
192
193              /* L2 PT */
194              pte *l2pt = (pte *) &pas[l1pt[j].frame];
195              for(int k=0; k<PAGETABLE_FRAMES; k++) {
196                  if(l2pt[k].vflag == PAGE_INVALID) continue; // 조건 확인
197                  printf("(L2PT) %03d -> %03d REF=%03d\n", j*8+k, l2pt[k].frame, l2pt[k].ref);
198              }
199          }
200
201          total_allocated_frame += (page_fault + 1);
202          total_page_fault += page_fault;
203          total_references += refer_num;
204      }
205
206      printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n", total_allocated_frame, total_page_fault, total_references);
207  }

```

- print\_result() 함수
- 프로세스, L1 PT, L2 PT를 차례로 순회하며 PAGE\_VALID인 경우 출력하도록 했다.

## test3.bin 수행 결과

```

173  /* 결과 출력 */
174  void print_result() {
175      // 결과 출력용 변수
176      int total_allocated_frame = 0;
177      int total_page_fault = 0;
178      int total_references = 0;
179
180      for(int i=0; i<process_num; i++) { // 프로세스
181          process_raw *cur = process_arr[i].process;
182          int page_fault = process_arr[i].page_fault_num;
183          int refer_num = process_arr[i].ref_idx;
184
185          printf("*** Process %03d: Allocated Frames=%03d PageFaults/References=%03d/%03d\n", i, page_fault+1, page_fault, refer_num);
186
187          /* L1 PT */
188          pte *l1pt = (pte *) &pas[cur->pid];
189          for(int j=0; j<PAGETABLE_FRAMES; j++) {
190              if(l1pt[j].vflag == PAGE_INVALID) continue; // 조건 확인
191              printf("(L1PT) %03d -> %03d\n", j, l1pt[j].frame);
192
193              /* L2 PT */
194              pte *l2pt = (pte *) &pas[l1pt[j].frame];
195              for(int k=0; k<PAGETABLE_FRAMES; k++) {
196                  if(l2pt[k].vflag == PAGE_INVALID) continue; // 조건 확인
197                  printf("(L2PT) %03d -> %03d REF=%03d\n", j*8+k, l2pt[k].frame, l2pt[k].ref);
198              }
199          }
200
201          total_allocated_frame += (page_fault + 1);
202          total_page_fault += page_fault;
203          total_references += refer_num;
204      }
205
206      printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n", total_allocated_frame, total_page_fault, total_references);
207  }

```

```

ubuntu@code-client1-222:~/hw3$ gcc os3-2.c && cat test3.bin | ./a.out
** 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

```

## JOTA 운영체제 과제 3-1 캡처

The screenshot shows a web browser window displaying the JOTA submission page for 'os202219352의 운영체제 과제 3-2 제출'. The page includes a navigation bar with 'DMOJ', 'PROBLEMS', 'SUBMISSIONS', 'USERS', 'CONTESTS', and 'ABOUT'. The submission details show the source code, compilation warnings, execution results, and resources. The execution results indicate that all five test cases passed (AC) with a final score of 10/10 (19.0/19 points). A timestamp at the bottom left indicates the submission was made on June 23rd at 02:48:45.

os202219352의 운영체제 과제 3-2 제출 2024년 6월 7일 오전 6시 11분 C

View source  
다시 제출

컴파일 경고

```
oshw32c.c: In function 'load_process':  
oshw32c.c:91:17: warning: ignoring return value of 'fread' declared with attribute 'warn_unused_result' [-Wunused-result]  
    91 |         fread(&cur->references[i], sizeof(unsigned char), 1, stdin);  
        |         ^~~~~~
```

Execution Results

✓✓✓✓✓

> Test case #1: AC [0.003s, 1.02 MB] (2/2)  
> Test case #2: AC [0.003s, 1.02 MB] (2/2)  
> Test case #3: AC [0.003s, 1.02 MB] (2/2)  
> Test case #4: AC [0.006s, 1.02 MB] (2/2)  
> Test case #5: AC [0.005s, 1.02 MB] (2/2)

Resources: 0.021s, 1.02 MB  
Maximum single-case runtime: 0.006s  
Final score: 10/10 (19.0/19 points)

운영체제과제 - 23일 02:48:45

proudly powered by DMOJ | 한국어 (ko)

## 과제 수행 시 어려웠던 점 및 해결 방안

- 어려웠던 점1: L1 PT, L2 PT의 동작 방식을 이해하는 데에 어려움이 있었다.
- 해결방안1: 예시 출력을 보고 고민해보고, 그림을 그려보며 동작 방식을 이해할 수 있었다.
- 어려웠던 점2: 일부 입력에서 RTE가 발생하는 문제가 있었다.
- 해결방안2: RTE가 발생하는 경우를 분석해보았고, 대부분 프로세스의 범위가 큰 경우임을 알 수 있었다. 이를 기반으로 고민해본 결과 3-1과 달리 L2 PT를 할당하는 과정에서 잘못된 메모리 접근이 있을 수 있음을 알 수 있었고, 예외를 처리해주었다.