

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define PAGESIZE (32)
6  #define PAS_FRAMES (256)
7  #define PAS_SIZE (PAGESIZE * PAS_FRAMES)
8  #define VAS_PAGES (64)
9  #define PTE_SIZE (4)
10 #define PAGE_INVALID (0)
11 #define PAGE_VALID (1)
12 #define MAX_REFERENCES (256)
13 #define MAX_PROCESSES (10)
14 #define L1_PT_ENTRIES (8)
15 #define L2_PT_ENTRIES (8)
16
17 typedef struct {
18     unsigned char frame;
19     unsigned char vflag;
20     unsigned char ref;
21     unsigned char pad;
22 } pte;
23
24 typedef struct {
25     int pid;
26     int ref_len;
27     unsigned char *references;
28     pte *l1_page_table;
29     int page_faults;
30     int ref_count;
31 } process;
32
33 unsigned char pas[PAS_SIZE];
34 int allocated_frame_count = 0;
35
36 int allocate_frame() {
37     if (allocated_frame_count >= PAS_FRAMES)
38         return -1;
39     return allocated_frame_count++;
40 }
41
42 // 페이지 테이블 프레임들 하나 할당하고, 해당 프레임들 0으로 초기화하여 반환하는 함수
43 // 2단계 페이지 테이블 구조에서 1단계/2단계 모두 8개 엔트리만 필요하므로 프레임 하나만 할당
44 // 반환값: 할당된 페이지 테이블의 시작 주소(실패 시 NULL)
45 pte *allocate_pagetable_frame() {
46     int frame = allocate_frame(); // 사용 가능한 프레임 번호 할당
47     if (frame == -1)
48         return NULL; // 프레임 할당 실패 시 NULL 반환
49     pte *page_table_ptr = (pte *)&pas[frame * PAGESIZE]; // 프레임 시작 주소를 pte 포인터로 변환
50     memset(page_table_ptr, 0, PAGESIZE); // 해당 프레임(32B)을 0으로 초기화
51     return page_table_ptr; // 페이지 테이블 포인터 반환
52 }
53
54 int load_process(FILE *fp, process *proc) {
55     if (fread(&proc->pid, sizeof(int), 1, fp) != 1)
56         return 0;
57     if (fread(&proc->ref_len, sizeof(int), 1, fp) != 1)
58         return 0;
59     proc->references = malloc(proc->ref_len);
60     if (fread(proc->references, 1, proc->ref_len, fp) != proc->ref_len)
61         return 0;
62
63     printf("%d %d\n", proc->pid, proc->ref_len);
64     for (int i = 0; i < proc->ref_len; i++) {
65         printf("%02d ", proc->references[i]);
66     }
67     printf("\n");
68
69     proc->page_faults = 0;
70     proc->ref_count = 0;
71     if ((proc->l1_page_table = allocate_pagetable_frame()) == NULL)
72         return -1;
73     return 1;

```

```

74 }
75
76
77 void simulate(process *procs, int proc_count) {
78     printf("simulate() start\n");
79
80     /* 라운드-로빈 방식으로 모든 프로세스의 참조를 순차적으로 처리 */
81     for (int idx = 0; ; ++idx) {
82         int all_done = 1; /* 더 처리할 참조가 없으면 종료 */
83
84         for (int p = 0; p < proc_count; ++p) {
85             process *proc = &procs[p];
86             if (idx >= proc->ref_len) /* 현재 프로세스의 참조가 끝났으면 건너뛰기 */
87                 continue;
88
89             all_done = 0; /* 아직 남은 일이 있음 */
90             unsigned char vpn = proc->references[idx]; /* 가상 페이지 번호(0-63) */
91             int l1_idx = vpn >> 3; /* 상위 3 비트 (0-7) */
92             int l2_idx = vpn & 0x07; /* 하위 3 비트 (0-7) */
93
94             /* ----- 1단계 페이지 테이블 ----- */
95             pte *l1_entry = &proc->l1_page_table[l1_idx];
96             int l1_pf = 0;
97             if (l1_entry->vflag == PAGE_INVALID) { /* L1 미할당 → 페이지 폴트 */
98                 int new_frame = allocate_frame();
99                 if (new_frame == -1) { puts("Out of memory!"); exit(1); }
100                 l1_entry->frame = (unsigned char)new_frame;
101                 l1_entry->vflag = PAGE_VALID;
102                 memset(&pas[new_frame * PAGESIZE], 0, PAGESIZE); /* 새 L2 PT 초기화 */
103                 l1_pf = 1;
104                 proc->page_faults++;
105             }
106
107             /* ----- 2단계 페이지 테이블 ----- */
108             pte *l2_pt = (pte *)&pas[l1_entry->frame * PAGESIZE];
109             pte *l2_entry = &l2_pt[l2_idx];
110             int l2_pf = 0;
111             if (l2_entry->vflag == PAGE_INVALID) { /* 실제 데이터 페이지 할당 */
112                 int new_frame = allocate_frame();
113                 if (new_frame == -1) { puts("Out of memory!"); exit(1); }
114                 l2_entry->frame = (unsigned char)new_frame;
115                 l2_entry->vflag = PAGE_VALID;
116                 l2_pf = 1;
117                 proc->page_faults++;
118             }
119
120             l2_entry->ref++; /* 페이지 참조 횟수 증가 */
121             proc->ref_count++; /* 프로세스 총 참조 횟수 증가 */
122
123             /* ----- trace 메시지 ----- */
124             printf("[PID %02d IDX:%03d] Page access %03d: ",
125                 proc->pid, idx, vpn);
126
127             printf("(L1PT) ");
128             if (l1_pf)
129                 printf("PF -> Allocated Frame %03d(PTE %03d), ",
130                     l1_entry->frame, l1_idx);
131             else
132                 printf("Frame %03d, ", l1_entry->frame);
133
134             printf("(L2PT) ");
135             if (l2_pf)
136                 printf("PF -> Allocated Frame %03d\n", l2_entry->frame);
137             else
138                 printf("Frame %03d\n", l2_entry->frame);
139
140         }
141
142         if (all_done) break; /* 전 프로세스 참조 처리 완료 */
143     }
144     printf("simulate() end\n");
145 }
146

```

```

147 void print_page_tables(process *procs, int proc_count) {
148     int total_pf = 0;
149     int total_refs = 0;
150
151     for (int p = 0; p < proc_count; ++p) {
152         process *proc = &procs[p];
153         total_pf += proc->page_faults;
154         total_refs += proc->ref_count;
155
156         /* ----- 프로세스별 프레임 수 계산 (L1 PT 제외) ----- */
157         int frames_used = 0;
158         for (int l1 = 0; l1 < L1_PT_ENTRIES; ++l1) {
159             pte *l1_entry = &proc->l1_page_table[l1];
160             if (l1_entry->vflag != PAGE_VALID) continue;
161
162             frames_used++; /* L2 페이지 테이블 프레임 */
163             pte *l2_pt = (pte *)&pas[l1_entry->frame * PAGESIZE];
164
165             for (int l2 = 0; l2 < L2_PT_ENTRIES; ++l2) {
166                 if (l2_pt[l2].vflag == PAGE_VALID) /* 실제 데이터 페이지 프레임 */
167                     frames_used++;
168             }
169         }
170
171         printf("*** Process %03d: Allocated Frames=%03d "
172                "PageFaults/References=%03d/%03d\n",
173                proc->pid, frames_used,
174                proc->page_faults, proc->ref_count);
175
176         /* ----- 페이지 테이블 내용 출력 ----- */
177         for (int l1 = 0; l1 < L1_PT_ENTRIES; ++l1) {
178             pte *l1_entry = &proc->l1_page_table[l1];
179             if (l1_entry->vflag != PAGE_VALID) continue;
180
181             printf("(L1PT) [PTE] %03d -> [FRAME] %03d\n",
182                    l1, l1_entry->frame);
183
184             pte *l2_pt = (pte *)&pas[l1_entry->frame * PAGESIZE];
185             for (int l2 = 0; l2 < L2_PT_ENTRIES; ++l2) {
186                 pte *l2_entry = &l2_pt[l2];
187                 if (l2_entry->vflag != PAGE_VALID) continue;
188
189                 int page_num = (l1 << 3) | l2; /* 실제 가상 페이지 번호 */
190                 printf("(L2PT) [PAGE] %03d -> [FRAME] %03d REF=%03d\n",
191                        page_num, l2_entry->frame, l2_entry->ref);
192             }
193         }
194     }
195
196     /* ----- 전체 통계 ----- */
197     printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n",
198            allocated_frame_count, total_pf, total_refs);
199 }
200
201
202
203
204 int main() {
205     process procs[MAX_PROCESSES];
206     int count = 0;
207
208     printf("load_process() start\n");
209     while (count < MAX_PROCESSES) {
210         int ret = load_process(stdin, &procs[count]);
211         if (ret == 0)
212             break;
213         if (ret == -1) {
214             printf("Out of memory!!\n");
215             return 1;
216         }
217         count++;
218     }
219     printf("load_process() end\n");

```

```

220
221     simulate(procs, count);
222     print_page_tables(procs, count);
223
224     for (int i = 0; i < count; i++) {
225         free(procs[i].references);
226     }
227
228     return 0;
229 }

```