```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define PAGESIZE (32)
#define PAS_FRAMES (256)
#define PAS_SIZE (PAGESIZE * PAS_FRAMES)
#define VAS_PAGES (64)
#define PTE_SIZE (4)
#define PAGE_INVALID (0)
#define PAGE_VALID (1)
#define MAX_REFERENCES (256)
#define MAX_PROCESSES (10)
#define L1_PT_ENTRIES (8)
#define L2_PT_ENTRIES (8)

typedef struct {
    unsigned char frame;
    unsigned char vflag;
    unsigned char ref;
    unsigned char pad;
} pte;

typedef struct {
    int pid;
    int ref_len;
    unsigned char *references;
    pte *L1_page_table;
    int page_faults;
    int ref_count;
} process;

unsigned char pas[PAS_SIZE];
int allocated_frame_count = 0;

int allocate_frame() {
    if (allocated_frame_count >= PAS_FRAMES)
        return -1;
    return allocated_frame_count++;
}

// 페이지 테이블 프레임을 하나 할당하고, 해당 프레임을 0으로 초기화하여 반환하는 함수
// 2단계 페이지 테이블 구조에서 1단계/2단계 모두 8개 엔트리만 필요하므로 프레임 하나만 할당
// 반환값: 할당된 페이지 테이블의 시작 주소(실패 시 NULL)
pte *allocate_pagetable_frame() {
    int frame = allocate_frame(); // 사용 가능한 프레임 번호 할당
    if (frame == -1)
        return NULL; // 프레임 할당 실패 시 NULL 반환
    pte *page_table_ptr = (pte *)&pas[frame * PAGESIZE]; // 프레임 시작 주소를 pte 포인터로 변환
    memset(page_table_ptr, 0, PAGESIZE); // 해당 프레임(32B)을 0으로 초기화
    return page_table_ptr; // 페이지 테이블 포인터 반환
}

int load_process(FILE *fp, process *proc) {
    if (fread(&proc->pid, sizeof(int), 1, fp) != 1)
        return 0;
    if (fread(&proc->ref_len, sizeof(int), 1, fp) != 1)
        return 0;
    proc->references = malloc(proc->ref_len);
    if (fread(proc->references, 1, proc->ref_len, fp) != proc->ref_len)
        return 0;

    printf("%d %d\n", proc->pid, proc->ref_len);
    for (int i = 0; i < proc->ref_len; i++) {
        printf("%02d ", proc->references[i]);
    }
    printf("\n");

    proc->page_faults = 0;
    proc->ref_count = 0;
    if ((proc->L1_page_table = allocate_pagetable_frame()) == NULL)
        return -1;
    return 1;
```

```c
}


void simulate(process *procs, int proc_count) {
    printf("simulate() start\n");

    /* 라운드-로빈 방식으로 모든 프로세스의 참조를 순차적으로 처리 */
    for (int idx = 0;; ++idx) {
        int all_done = 1;                   /* 더 처리할 참조가 없으면 종료 */

        for (int p = 0; p < proc_count; ++p) {
            process *proc = &procs[p];
            if (idx >= proc->ref_len)       /* 현재 프로세스의 참조가 끝났으면 건너뜀 */
                continue;

            all_done = 0;                   /* 아직 남은 일이 있음 */
            unsigned char vpn   = proc->references[idx];   /* 가상 페이지 번호(0-63) */
            int          l1_idx = vpn >> 3;                /* 상위 3 비트  (0-7) */
            int          l2_idx = vpn & 0x07;              /* 하위 3 비트  (0-7) */

            /* ---------- 1단계 페이지 테이블 ---------- */
            pte *l1_entry = &proc->L1_page_table[l1_idx];
            int  l1_pf = 0;
            if (l1_entry->vflag == PAGE_INVALID) {          /* L1 미할당 → 페이지 폴트 */
                int new_frame = allocate_frame();
                if (new_frame == -1) { puts("Out of memory!"); exit(1); }
                l1_entry->frame = (unsigned char)new_frame;
                l1_entry->vflag = PAGE_VALID;
                memset(&pas[new_frame * PAGESIZE], 0, PAGESIZE);  /* 새 L2 PT 초기화 */
                l1_pf = 1;
                proc->page_faults++;
            }

            /* ---------- 2단계 페이지 테이블 ---------- */
            pte *l2_pt    = (pte *)&pas[l1_entry->frame * PAGESIZE];
            pte *l2_entry = &l2_pt[l2_idx];
            int  l2_pf = 0;
            if (l2_entry->vflag == PAGE_INVALID) {          /* 실제 데이터 페이지 할당 */
                int new_frame = allocate_frame();
                if (new_frame == -1) { puts("Out of memory!"); exit(1); }
                l2_entry->frame = (unsigned char)new_frame;
                l2_entry->vflag = PAGE_VALID;
                l2_pf = 1;
                proc->page_faults++;
            }

            l2_entry->ref++;        /* 페이지 참조 횟수 증가            */
            proc->ref_count++;      /* 프로세스 총 참조 횟수 증가       */

            /* ---------- trace 메시지 ---------- */
            printf("[PID %02d IDX:%03d] Page access %03d: ",
                   proc->pid, idx, vpn);

            printf("(L1PT) ");
            if (l1_pf)
                printf("PF -> Allocated Frame %03d(PTE %03d), ",
                       l1_entry->frame, l1_idx);
            else
                printf("Frame %03d, ", l1_entry->frame);

            printf("(L2PT) ");
            if (l2_pf)
                printf("PF -> Allocated Frame %03d\n", l2_entry->frame);
            else
                printf("Frame %03d\n", l2_entry->frame);
        }

        if (all_done) break;        /* 전 프로세스 참조 처리 완료 */
    }

    printf("simulate() end\n");
}

```

```c
void print_page_tables(process *procs, int proc_count) {
    int total_pf   = 0;
    int total_refs = 0;

    for (int p = 0; p < proc_count; ++p) {
        process *proc = &procs[p];
        total_pf   += proc->page_faults;
        total_refs += proc->ref_count;

        /* ---------------- 프로세스별 프레임 수 계산 (L1 PT 제외) ---------------- */
        int frames_used = 0;
        for (int l1 = 0; l1 < L1_PT_ENTRIES; ++l1) {
            pte *l1_entry = &proc->L1_page_table[l1];
            if (l1_entry->vflag != PAGE_VALID) continue;

            frames_used++;                          /* L2 페이지 테이블 프레임 */
            pte *l2_pt = (pte *)&pas[l1_entry->frame * PAGESIZE];

            for (int l2 = 0; l2 < L2_PT_ENTRIES; ++l2) {
                if (l2_pt[l2].vflag == PAGE_VALID)      /* 실제 데이터 페이지 프레임 */
                    frames_used++;
            }
        }

        printf("** Process %03d: Allocated Frames=%03d "
               "PageFaults/References=%03d/%03d\n",
               proc->pid, frames_used,
               proc->page_faults, proc->ref_count);

        /* ---------------- 페이지 테이블 내용 출력 ---------------- */
        for (int l1 = 0; l1 < L1_PT_ENTRIES; ++l1) {
            pte *l1_entry = &proc->L1_page_table[l1];
            if (l1_entry->vflag != PAGE_VALID) continue;

            printf("(L1PT) [PTE] %03d -> [FRAME] %03d\n",
                   l1, l1_entry->frame);

            pte *l2_pt = (pte *)&pas[l1_entry->frame * PAGESIZE];
            for (int l2 = 0; l2 < L2_PT_ENTRIES; ++l2) {
                pte *l2_entry = &l2_pt[l2];
                if (l2_entry->vflag != PAGE_VALID) continue;

                int page_num = (l1 << 3) | l2;          /* 실제 가상 페이지 번호 */
                printf("(L2PT) [PAGE] %03d -> [FRAME] %03d REF=%03d\n",
                       page_num, l2_entry->frame, l2_entry->ref);
            }
        }
    }

    /* ---------------- 전체 통계 ---------------- */
    printf("Total: Allocated Frames=%03d Page Faults/References=%03d/%03d\n",
           allocated_frame_count, total_pf, total_refs);
}




int main() {
    process procs[MAX_PROCESSES];
    int count = 0;

    printf("load_process() start\n");
    while (count < MAX_PROCESSES) {
        int ret = load_process(stdin, &procs[count]);
        if (ret == 0)
            break;
        if (ret == -1) {
            printf("Out of memory!!\n");
            return 1;
        }
        count++;
    }
    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     }
```