



2016년 2학기 운영체제실습 12주차

# Memory Management

Dept. of Computer Engineering,  
Kwangwoon Univ.

# Contents

---

- ▶ **Abc..**

# 메모리 관리의 이해와 기법 소개

---

## ▶ 메모리 관리의 기본 핵심

- ▶ 커널은 task를 언제, 어디에, 어떻게 물리 메모리에 적재할 지 관리해야 함

## ▶ 한정적인 물리 메모리의 해결 방안

- ▶ 가상 메모리
- ▶ 물리 메모리에는 수행에 필요한 부분만 적재

# 물리 메모리와 가상 메모리

---

## ▶ 물리 메모리

- ▶ 시스템에 장착된 실제 메모리
- ▶ 메모리에 대한 실제 물리 주소를 가짐
  - ▶ 0번지부터 장착된 메모리 크기까지의 범위

## ▶ 가상 메모리

- ▶ 실제 존재하지는 않지만 큰 메모리가 존재하는 것과 같은 효과
- ▶ 가상 주소는 물리 주소와 상관없이 각 task마다 할당되는 논리적인 주소
- ▶ 리눅스에서는 각 task마다 4GB 가상 주소공간을 할당
  - ▶ 각 task: 3GB, 커널 영역: 1GB

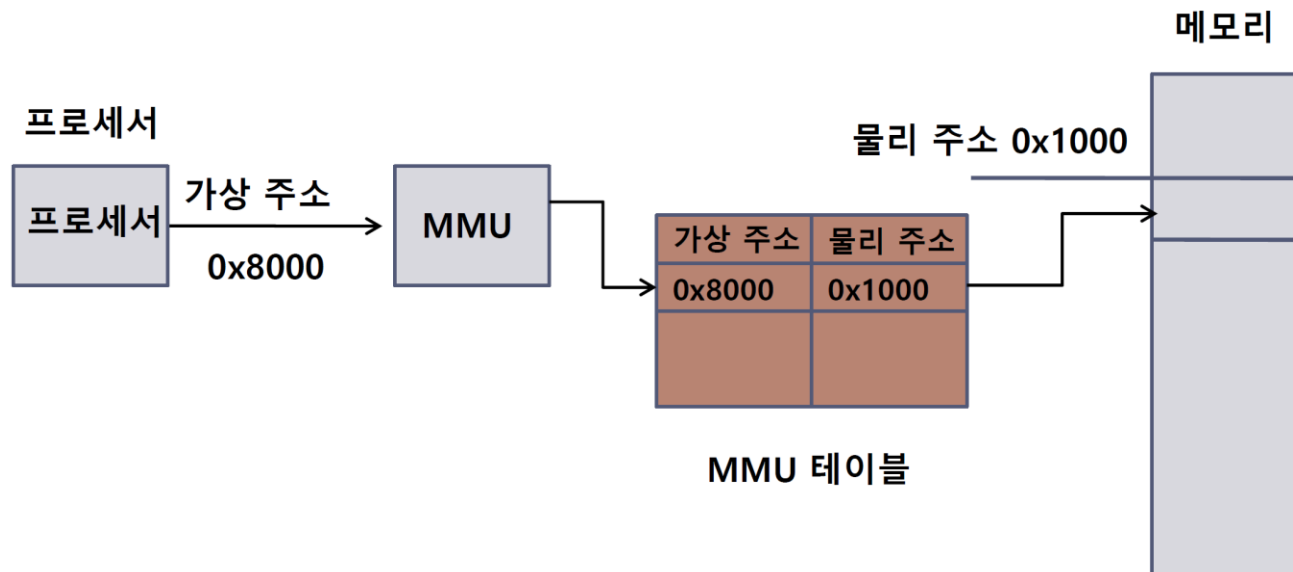
## ▶ 가상 주소를 물리 주소로 변환하는 기법 필요

- ▶ Paging 기법을 사용

# 물리 메모리와 가상 메모리

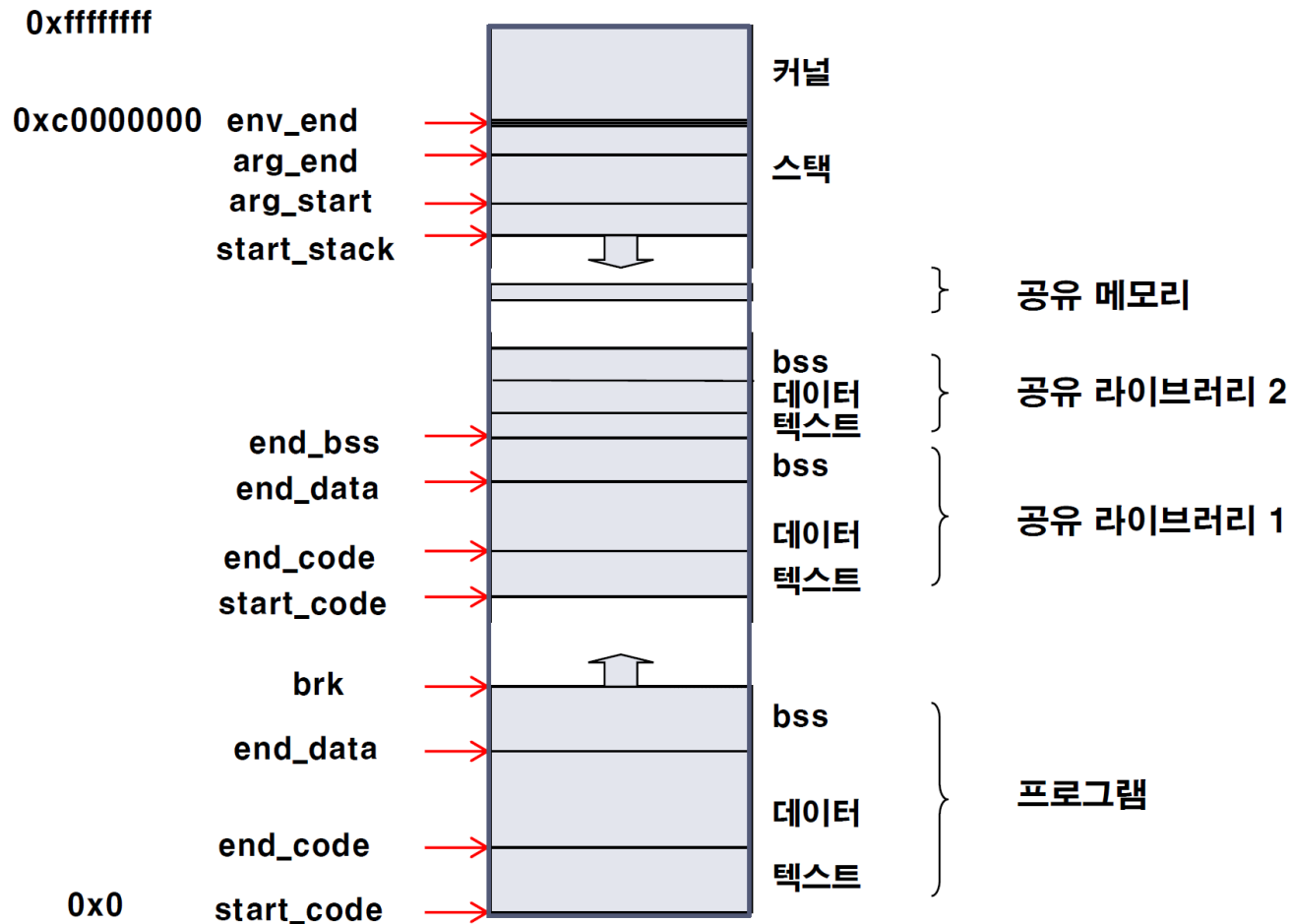
## ▶ 가상 주소는

- ▶ 내부적으로 메모리 관리 기능을 통해 물리주소로 변환
- ▶ 이를 실제 물리 메모리에 매핑

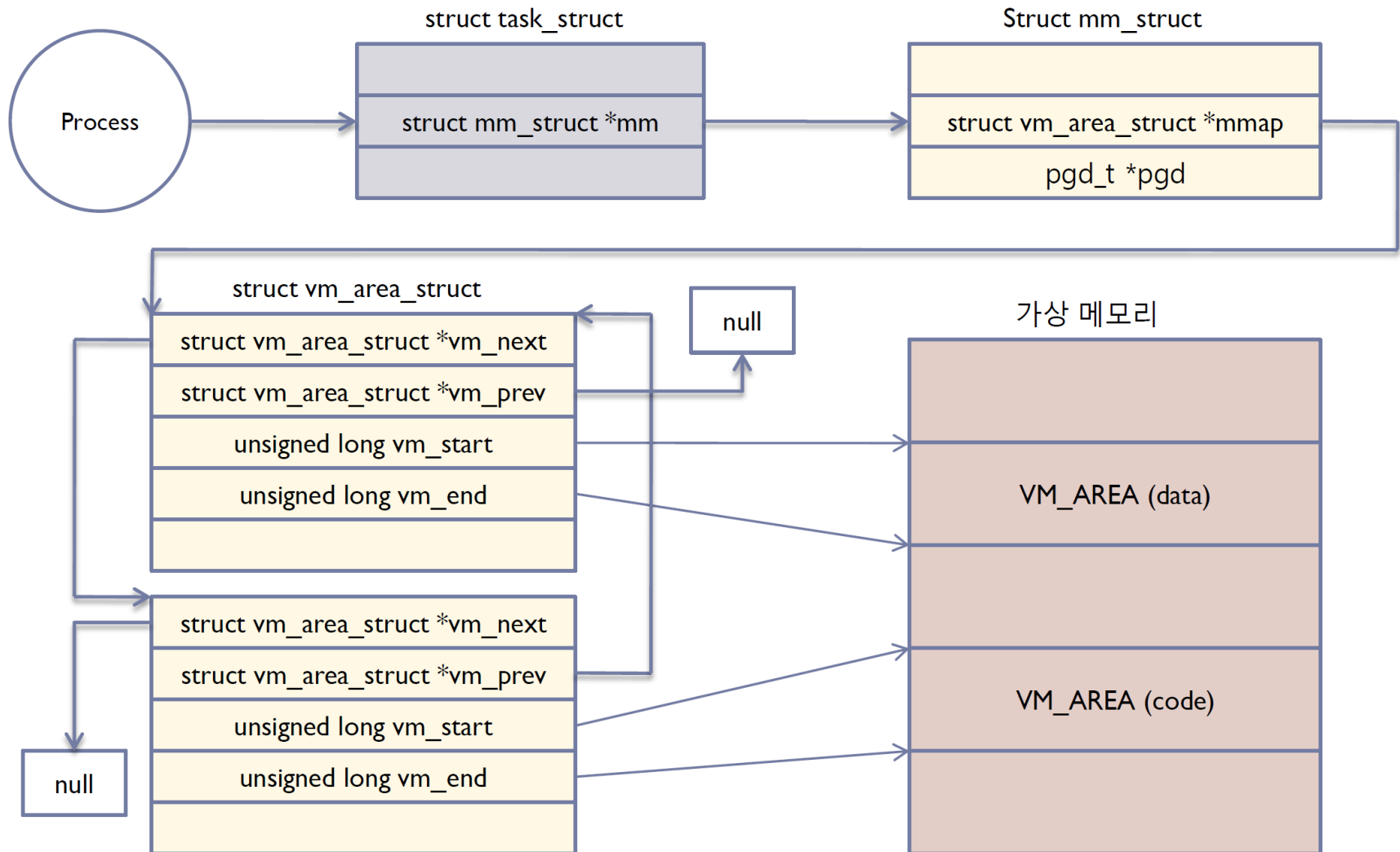


# 물리 메모리와 가상 메모리

## ▶ 리눅스 커널의 가상 메모리



# 메모리 관리를 위한 자료 구조



# 메모리 관리를 위한 자료 구조

---

## ▶ mm\_struct 주요 함수

- ▶ `struct mm_struct *get_task_mm(struct task_struct *task)`
  - ▶ use count를 1 증가
  - ▶ task의 mm\_struct를 가져옴
- ▶ `void mmput(struct mm_struct *mm)`
  - ▶ use count를 1 감소
  - ▶ use count가 0이 되면 할당된 메모리 공간을 해제



# 메모리 관리를 위한 자료 구조

---

## ▶ mm\_struct 주요 멤버 변수

- ▶ struct vm\_area\_struct \*mmap
  - ▶ vm\_area\_struct list의 시작 주소
- ▶ pgd\_t \*pgd
  - ▶ 페이지 글로벌 디렉토리의 시작 주소
  - ▶ 페이지 글로벌 디렉토리
    - 가상 주소를 물리 주소로 변환하기 위한 최상위 테이블
- ▶ start\_code, end\_code
  - ▶ 코드 세그먼트 영역
    - 프로그램의 명령들이 들어가는 영역
- ▶ start\_data, end\_data
  - ▶ 데이터 세그먼트 영역
    - 프로그램에 선언된 전역 변수들로 구성
    - 초기화 된 변수가 들어가는 데이터 영역과 초기화 되지 않은 변수가 들어가는 BSS(Block Started by Symbol)로 구분

# 메모리 관리를 위한 자료 구조

---

## ▶ mm\_struct 주요 멤버 변수 (cont'd)

- ▶ start\_brk, brk
  - ▶ 힙(heap)의 시작과 끝 주소를 가지는 변수
- ▶ start\_stack
  - ▶ 스택의 시작 위치
  - ▶ 함수 호출 시 전달되는 인자들과 복귀 주소 및 지역 변수로 구성
  - ▶ 스택의 메모리 할당 방향은 힙과 반대
- ▶ total\_vm
  - ▶ 할당된 전체 가상 메모리 크기
- ▶ reserved\_vm
  - ▶ 예약된 메모리 크기

# 메모리 관리를 위한 자료 구조

---

## ▶ `vm_area_struct` 주요 멤버 변수

### ▶ `vm_mm`

- ▶ 이 가상 메모리 영역을 사용하고 있는 `mm_struct` 구조체를 가리키는 포인터 변수

### ▶ `vm_start`

- ▶ 영역의 시작 주소

### ▶ `vm_end`

- ▶ 영역의 끝 주소

### ▶ `vm_next`

- ▶ 다른 가상 메모리 블록을 가리키는 포인터 변수
- ▶ 이를 따라가면 task가 사용하는 전체 가상 공간을 알 수 있음
- ▶ 마지막 노드의 `vm_next`는 NULL 값을 가짐

# 메모리 매핑

---

- ▶ 파일시스템의 파일과 메모리 공간을 매핑하는 방법
  - ▶ 파일에 대한 read/write 연산 마다,
    - ▶ 메모리 공간의 data를 업데이트 하고,
    - ▶ 저장장치의 내용을 업데이트 하는 것은 비효율적
  - ▶ 따라서 이를 **mapping** 하여 처리

# 메모리 매핑

---

## ▶ 관련 함수

- ▶ `void * mmap`  
(`void *start`, `size_t length`, `int prot`, `int flags`, `int fd`, `off_t offset`);
  - ▶ 메모리에 파일이나 장치를 map
  - ▶ 호출한 프로세스의 가상 메모리 공간에 새로운 매핑을 생성
  - ▶ Parameters
    - `start` : 시작 위치
    - `length` : 매핑 길이
    - `prot` : 페이지 권한
      - `PROT_EXEC` : 실행 권한
      - `PROT_READ` : 읽기 권한
      - `PROT_WRITE` : 쓰기 권한
      - `PROT_NONE` : 권한 없음
    - `flags` : 매핑 방식
      - `MAP_SHARED` : 같은 파일을 매핑한 다른 프로세스들에게 변경 사항 공유
      - `MAP_PRIVATE` : 변경 사항이 다른 프로세스들에게 공유되지 않음
      - Other options... : \$ man mmap
    - `fd` : 장치나 파일에 대한 file descriptor
    - `offset` : fd에 해당하는 장치/파일에서의 시작 위치 (PAGE\_SIZE의 배수)

# 메모리 매핑

---

## ▶ 관련 함수

- ▶ `int munmap (void *start, size_t length);`
  - ▶ 메모리에 매핑된 파일이나 장치를 unmap
  - ▶ Parameters
    - `start` : 매핑 시작 주소
    - `length` : 매핑된 길이
- ▶ `int msync (void *start, size_t length, int flags);`
  - ▶ 메모리 매핑 후 변경된 사항을 파일에 반영 (동기화)
  - ▶ Parameters
    - `start` : 매핑 시작 주소
    - `length` : 매핑된 길이
    - `flags` : 동기화 방식
      - `MS_SYNC` : 동기화를 요청하고, 동기화가 끝날 때 까지 대기
      - `MS_ASYNC` : 동기화를 요청하고, 즉시 return
      - `MS_INVALIDATE` : 같은 파일에 대한 다른 매핑의 데이터를 현재 매핑에 반영

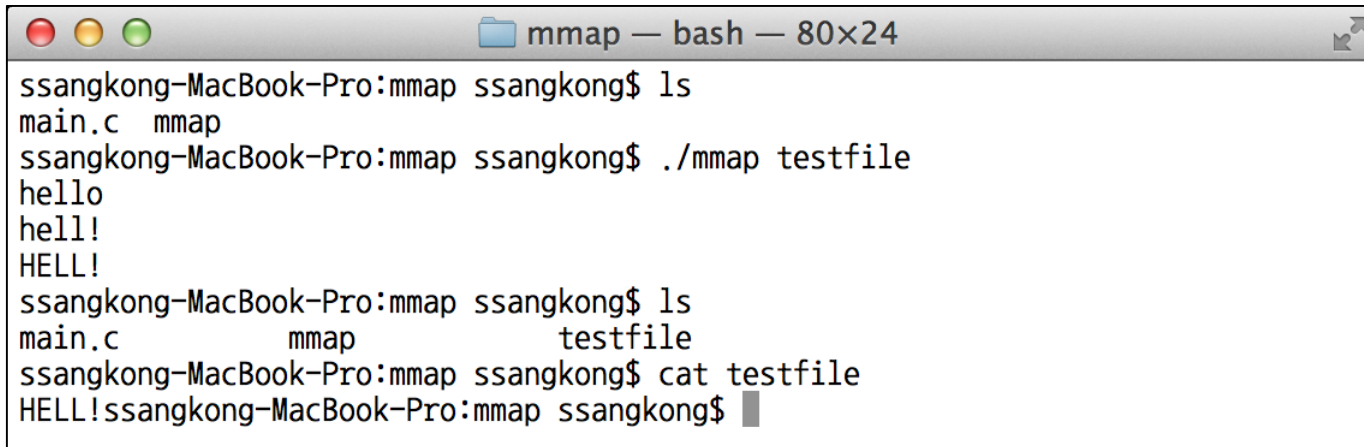
# Example

```
ssangkong — vim — 143x45

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <sys/mman.h>
6 #include <assert.h>
7
8 void create_data(const char *filename);
9 void display_data(const char *filename);
10 void change_data(const char *filename);
11 void mmap_data(const char *filename);
12
13 int main(int argc, const char *argv[])
14 {
15     const char *filename;
16     assert(argc == 2);
17     filename = argv[1];
18     create_data(filename);
19     display_data(filename);
20     change_data(filename);
21     display_data(filename);
22     mmap_data(filename);
23     display_data(filename);
24
25     return EXIT_SUCCESS;
26 }
27
28 void create_data(const char *filename)
29 {
30     int fd;
31     int i;
32
33     fd = open(filename, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);
34     assert(fd != -1);
35     write(fd, "hello", 5);
36     close(fd);
37 }
38
39 void display_data(const char *filename)
40 {
41     int fd;
42     char data;
43     int i;
44
45     fd = open(filename, O_RDONLY);
46     assert(fd != -1);
47     for( i=0 ; i<5 ; ++i )
48         if( read(fd, &data, sizeof(char)) == sizeof(char) )
49             printf("%c", data);
50     printf("\n");
51     close(fd);
52 }
53
54 void change_data(const char *filename)
55 {
56     int fd;
57     char data;
58
59     fd = open(filename, O_RDWR);
60     assert(fd != -1);
61     lseek(fd, 4 * sizeof(char), SEEK_SET);
62     read(fd, &data, sizeof(char));
63     data = '!';
64     lseek(fd, -sizeof(char), SEEK_CUR);
65     write(fd, &data, sizeof(char));
66     close(fd);
67 }
68
69 void mmap_data(const char *filename)
70 {
71     int fd;
72     char *map;
73     int pagesize;
74
75     fd = open(filename, O_RDWR);
76     pagesize = getpagesize();
77     map = mmap(0, pagesize, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
78     assert(map != MAP_FAILED);
79     map[0] &= ~0x20;
80     map[1] ^= 0x20;
81     map[2] &= 0xDF;
82     map[3] ^= ~0xDF;
83     msync(map, pagesize, MS_ASYNC);
84     munmap(map, pagesize);
85 }
```

main.c 1,1 Top main.c 79,1-4 Bot

# Example



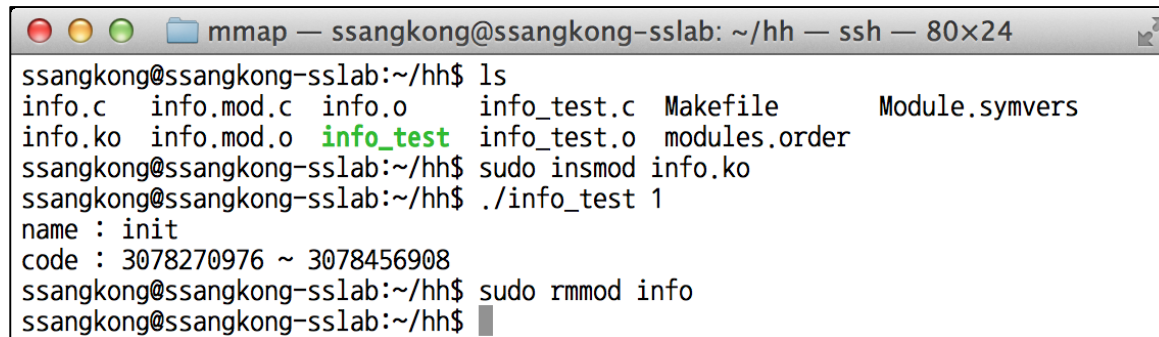
```
ssangkong-MacBook-Pro:mmap ssangkong$ ls
main.c  mmap
ssangkong-MacBook-Pro:mmap ssangkong$ ./mmap testfile
hello
hell!
HELL!
ssangkong-MacBook-Pro:mmap ssangkong$ ls
main.c      mmap      testfile
ssangkong-MacBook-Pro:mmap ssangkong$ cat testfile
HELL!ssangkong-MacBook-Pro:mmap ssangkong$
```



# 실습

- ▶ pid를 입력 받고,
  - ▶ 이에 대한 프로세스의 이름과
  - ▶ 코드 영역의 시작 주소 값, 끝 주소 값을 가져오는 모듈 작성

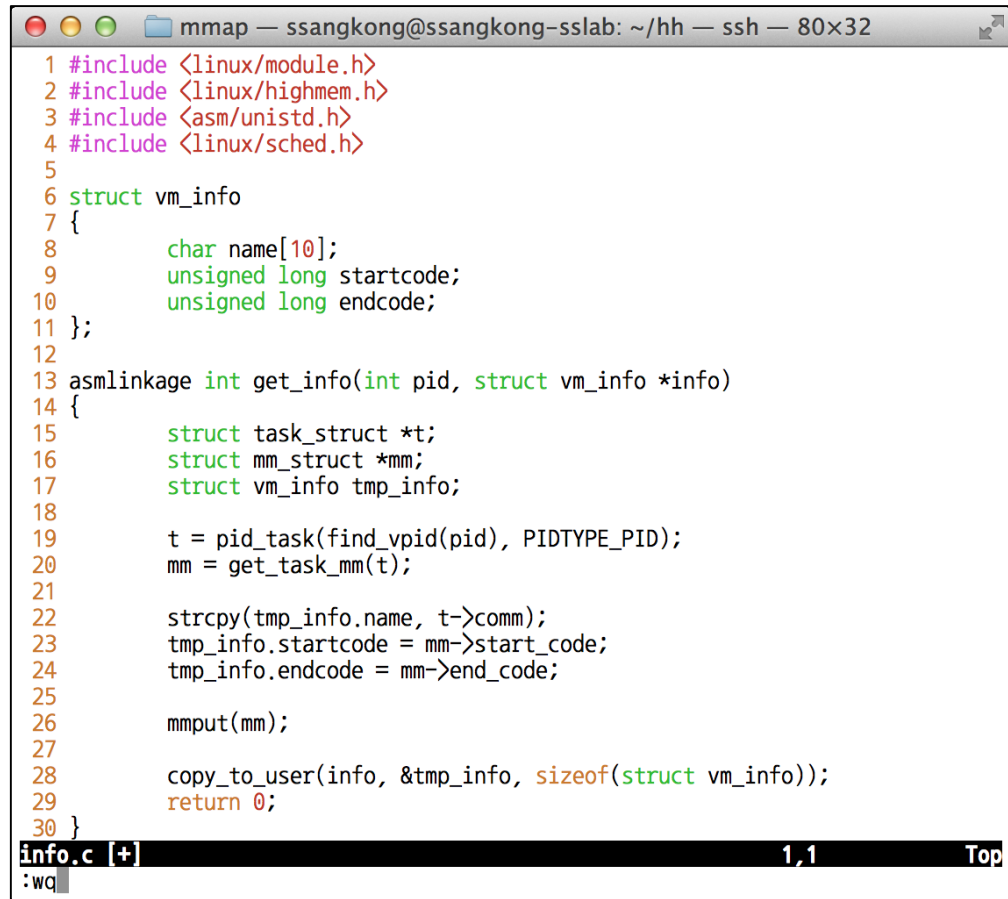
## ▶ 결과 예시



```
mmap — ssangkong@ssangkong-sslslab: ~/hh — ssh — 80x24
ssangkong@ssangkong-sslslab:~/hh$ ls
info.c  info.mod.c  info.o  info_test.c  Makefile  Module.symvers
info.ko  info.mod.o  info_test  info_test.o  modules.order
ssangkong@ssangkong-sslslab:~/hh$ sudo insmod info.ko
ssangkong@ssangkong-sslslab:~/hh$ ./info_test 1
name : init
code : 3078270976 ~ 3078456908
ssangkong@ssangkong-sslslab:~/hh$ sudo rmmod info
ssangkong@ssangkong-sslslab:~/hh$
```

# 실습

## ▶ 시스템 콜을 wrapping 하여 구현



```
mmap — ssangkong@ssangkong-sslab: ~/hh — ssh — 80x32
1 #include <linux/module.h>
2 #include <linux/highmem.h>
3 #include <asm/unistd.h>
4 #include <linux/sched.h>
5
6 struct vm_info
7 {
8     char name[10];
9     unsigned long startcode;
10    unsigned long endcode;
11 };
12
13 asmlinkage int get_info(int pid, struct vm_info *info)
14 {
15     struct task_struct *t;
16     struct mm_struct *mm;
17     struct vm_info tmp_info;
18
19     t = pid_task(find_vpid(pid), PIDTYPE_PID);
20     mm = get_task_mm(t);
21
22     strcpy(tmp_info.name, t->comm);
23     tmp_info.startcode = mm->start_code;
24     tmp_info.endcode = mm->end_code;
25
26     mmput(mm);
27
28     copy_to_user(info, &tmp_info, sizeof(struct vm_info));
29     return 0;
30 }
info.c [+] 1,1 Top
:wq
```

# 실습

## ▶ 테스트 프로그램

```
mmap — ssangkong@ssangkong-sslab: ~/hh — ssh — 80×24
1 #include <linux/unistd.h>
2 #include <stdio.h>
3 #include <assert.h>
4
5 struct vm_info
6 {
7     char name[10];
8     unsigned long startcode;
9     unsigned long endcode;
10 };
11
12 int main(int argc, const char *argv[])
13 {
14     int pid;
15     struct vm_info info;
16     assert(argc == 2);
17     pid = atoi(argv[1]);
18     syscall(__NR_add, pid, &info);
19     printf("name : %s\n", info.name);
20     printf("code : %lu ~ %lu\n", info.startcode, info.endcode);
21     return 0;
22 }
info_test.c 1,1 Top
:wq
```