Project 4 : Pintos Virtual Memory

[CSE4070]

Fall 2018

운영체제 1반 조교 강현구

서강대학교
SOGANG UNIVERSITY

Overview

- **Reading pintos document is highly recommended especially for this project (pp. 39-49)**
- Basically, the 'vm' directory contains only 'Makefile's
- In project 1, 2 and 3, a program was terminated when a page fault occurs
- In this project, you will make the pintos to be more reliable from page faults and to run the programs properly
- All code you write will be in new files or in files introduced in earlier projects

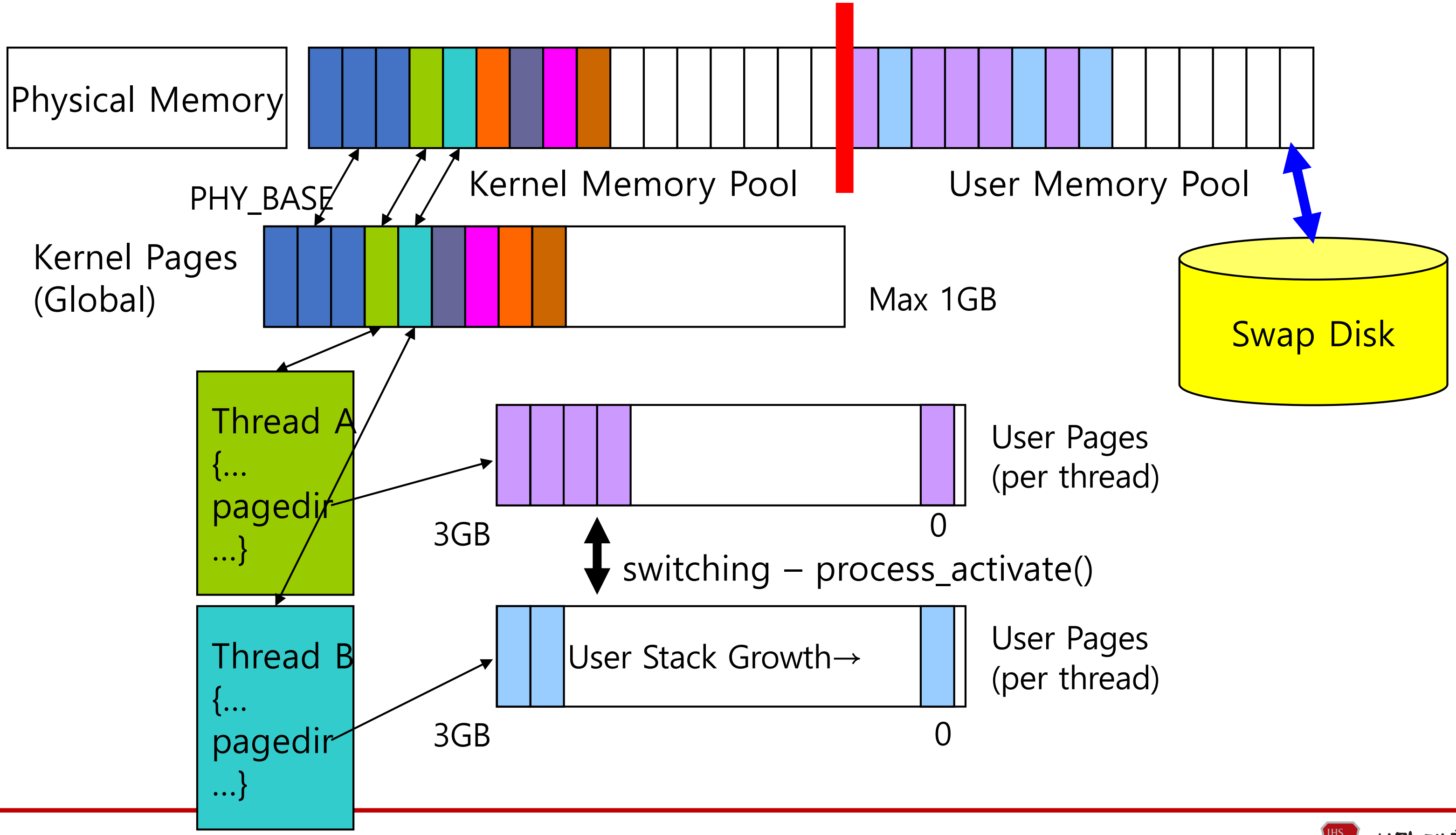
Project Requirement

- Page Table Management
  - Supplemental page table and page fault handling

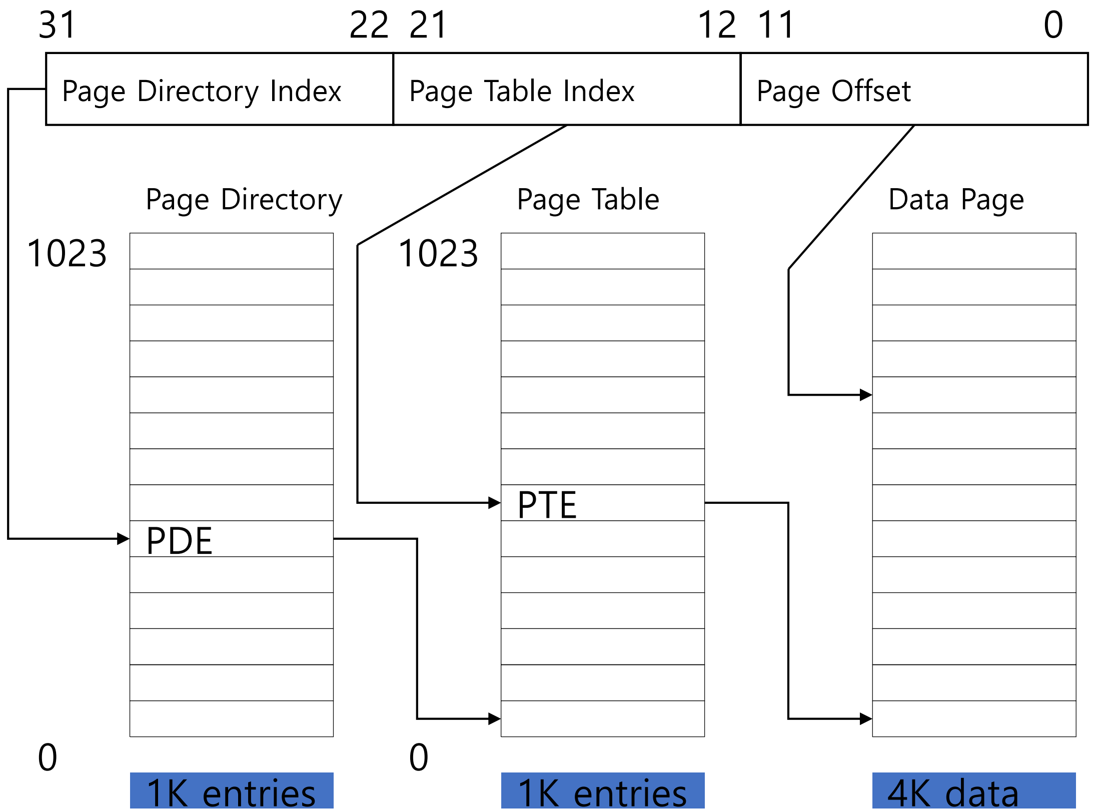
- Paging to and from (swap) disk
  - Implement pseudo-LRU policies (second chance)

- Stack Growth

Virtual Memory Overview



Page Table

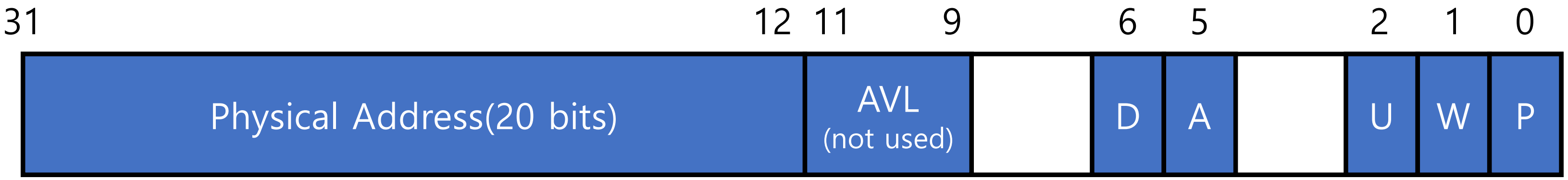

31
22
21
12
11
0
Page Directory Index
Page Table Index
Page Offset
Page Directory
Page Table
Data Page
1023
1023
PDE
PTE
0
0
1K entries
1K entries
4K data

Page Table Management

- 기존 Page Table에 필요한 정보 추가
  - 주로 Page fault handling을 위한 정보를 추가
- Page Fault handler 구현

Supplemental Page Table

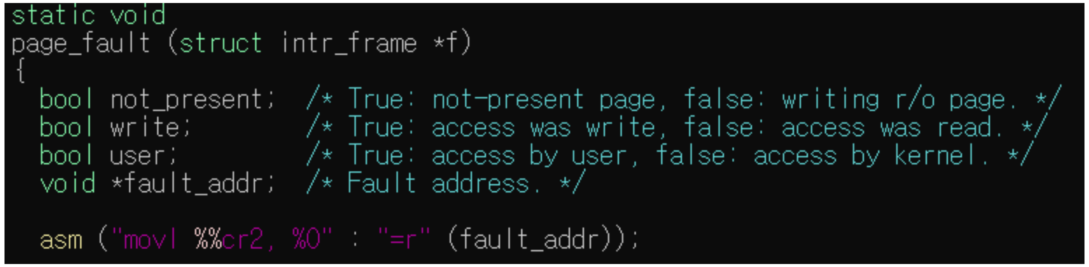
- Since the given page table in pintos has limitations, we need to supplement the page table with additional data about each page
- You can exploit the functions in userprog/pagedir.c to implement supplemental page table
- Page table entry format (flags are defined in threads/pte.h)



PTE_P	present bit
PTE_W	read(0)/write(1) bit
PTE_U	kernel(0)/user(1) bit
PTE_A	accessed bit
PTE_D	dirty bit
PTE_AVL	not used in pintos

Page Fault Handler

- userprog/exception.c의 page_fault()
- Page Fault situation
  - page from a file or swap
- Access is invalid
  - If the page is unmapped, that is, if there’s no data where the page is referenced
  - If the access is an attempt to write to a read-only page (unprivileged access)
- CR2 : register storing faulted address
- Some boolean variables in page_fault() will help you
  - bool not_present; // not present in memory or rights violation
  - bool write; // write or read fault
  - bool user; // fault from user or kernel space



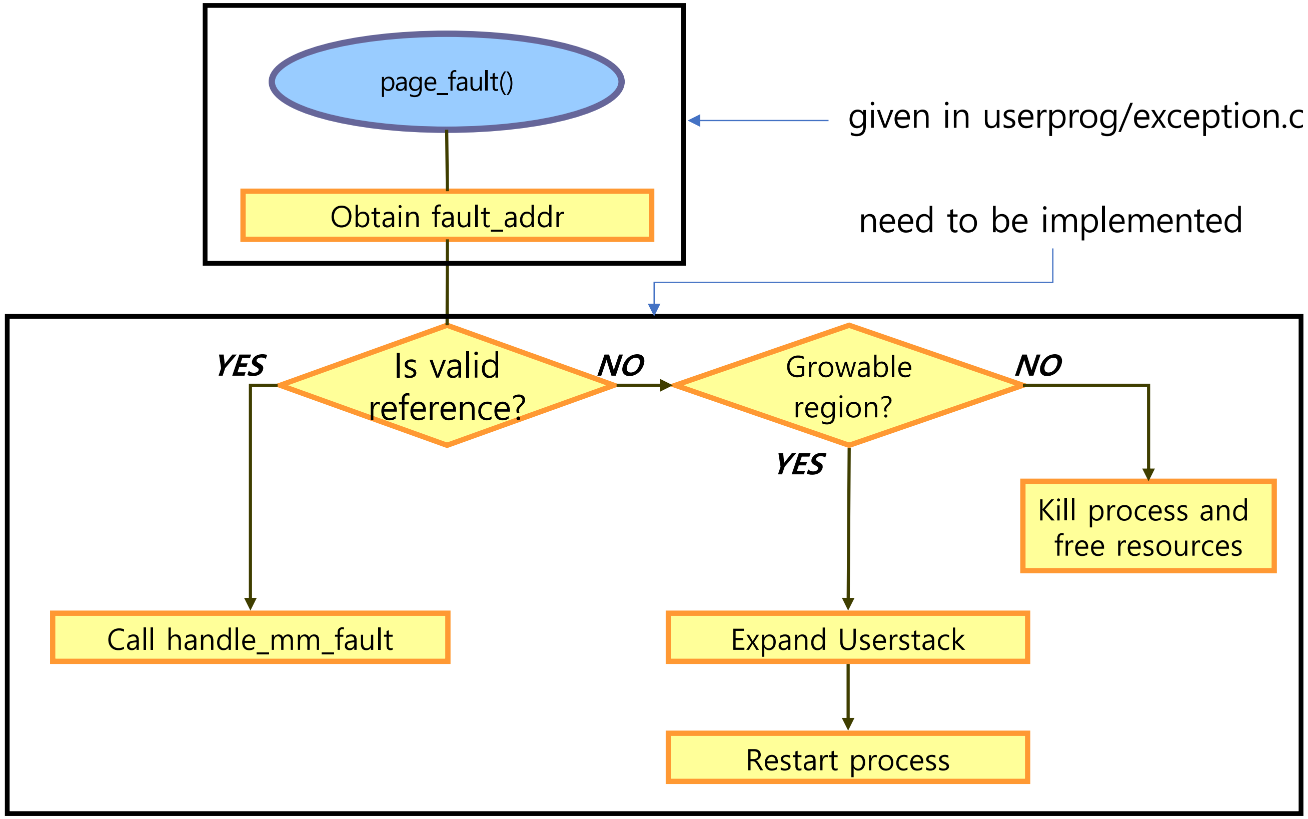
```
static void
page_fault (struct intr_frame *f)
{
  bool not_present;  /* True: not-present page, false: writing r/o page. */
  bool write;        /* True: access was write, false: access was read. */
  bool user;         /* True: access by user, false: access by kernel. */
  void *fault_addr;  /* Fault address. */

  asm ("movl %%cr2, %0" : "=r" (fault_addr));
```

Page Fault Handler

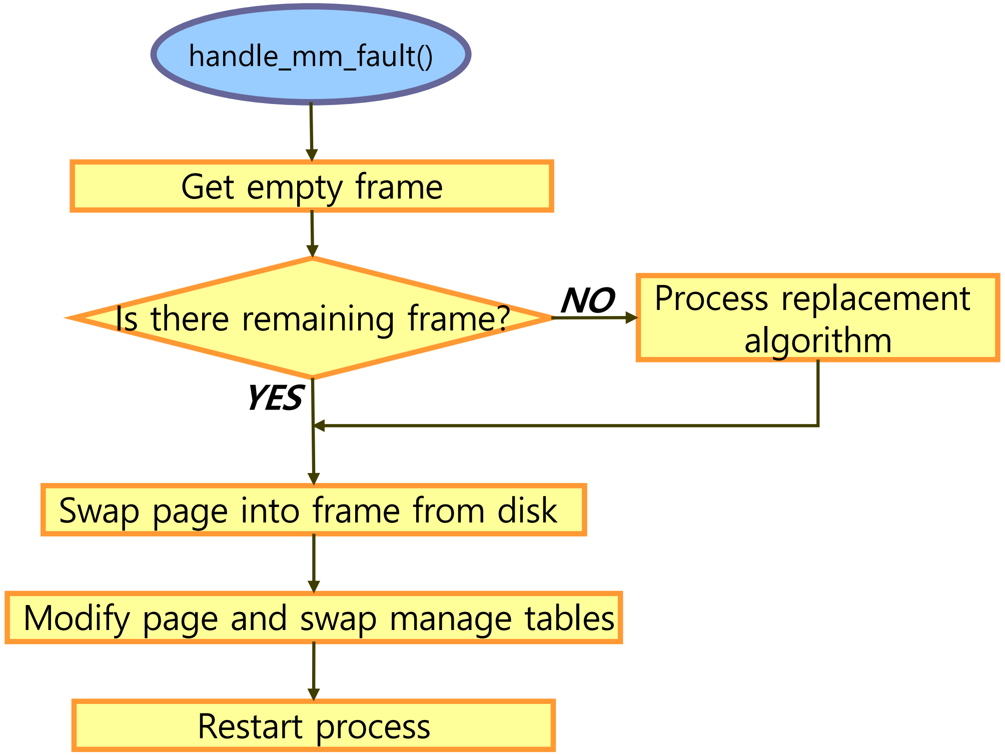
- Page Fault handling procedures

1. Processor (CPU) triggers page fault
2. Control is passed to the kernel, which calls the page fault handler (userprog/exception.c:page_fault())
3. Get the faulted address from CR2 register
4. If the memory reference is valid
    - Obtain a frame to store the page
    - Fetch the data into the frame, by reading it from the file system or swap, zeroing it, etc.
    - Point the page table entry for the faulting virtual address to the physical page
5. If the access is invalid
    - Any invalid access terminates the process and thereby frees all of its resources

Page Fault Handler: page_fault()



Page Fault Handler: handle_mm_fault()


handle_mm_fault()
Get empty frame
Is there remaining frame?
NO
Process replacement algorithm
YES
Swap page into frame from disk
Modify page and swap manage tables
Restart process

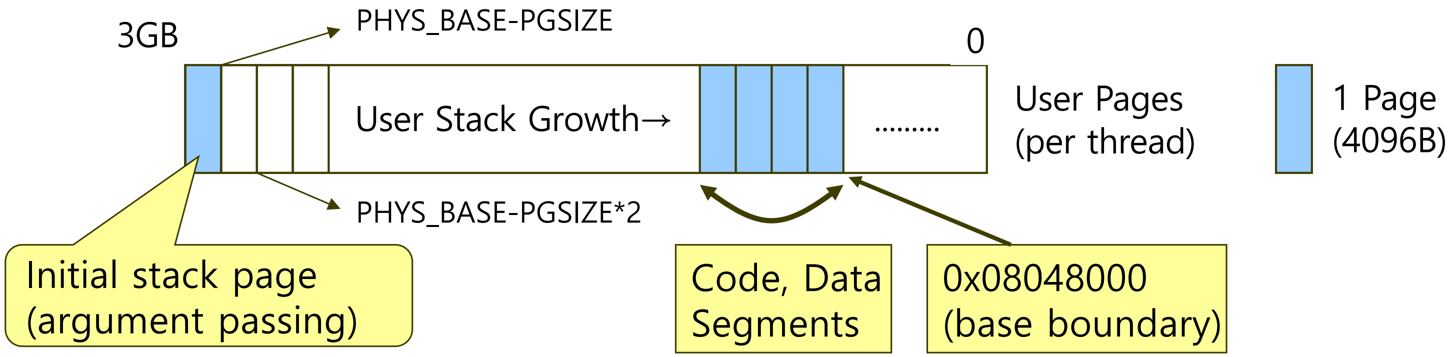
Paging to and from (swap) disk

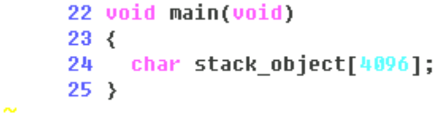
- Swap disk
  - Process에 할당해 줄 Physical memory가 부족할 때(User page pool에 free page가 없을 때) disk로 swap-out이 일어남
  - Swap 할 page의 결정은 page replacement algorithm 사용(LRU, LFU …)
- Swap table 작성
  - swap disk가 현재 사용하고 있는 슬롯과 빈 슬롯 관리
- devices/block.c 의 block_read() / block_write() 활용
- You may use the **BLOCK_SWAP block device** for swapping, obtaining the **struct block** that represents it by calling **block_get_role()**
- BLOCK_SWAP에 대해서는 devices/partition.c, thread/init.c 참조
- swap disk 생성
  - vm/build에서
    - pintos-mkdisk swap.dsk --swap-size=*n* --> swap.dsk 라는 이름으로 *n* MB swap disk 생성
    - swap.dsk는 pintos의 실행 시 자동으로 hd1:1에 attach됨
  - Pintos 실행 시 argument로 '--swap-disk=*n*'을 추가해도 n-MB swap disk가 생성됨

Stack Growth

- If page faults on an address that "appears" to be a stack access, allocate another stack page
- You should impose some absolute limit on stack size, as do most OSes.
  On many GNU/Linux systems, the default limit is 8 MB.
- First stack page can still be loaded at process load time (in order to get arguments, etc.)

Stack Growth (Example 1)

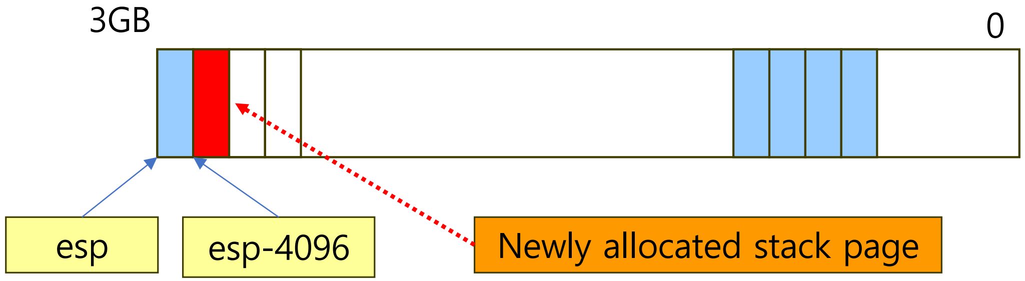




```
void main(void)
{
  char stack_object[4096];
}
```

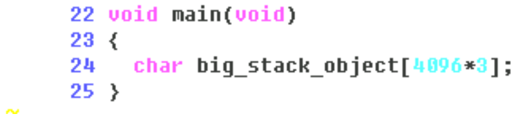
esp = esp-4096 ;allocate stack memory

but the new esp, "esp-4096", will cause page fault



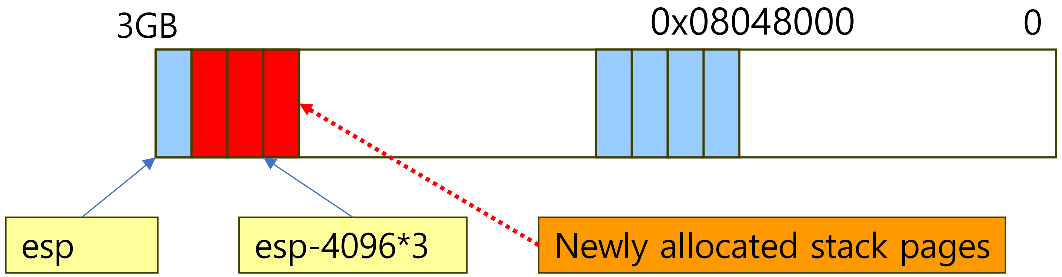
Stack Growth (Example 2)

What if user program tries to allocate more than a single PAGE size?



```
void main(void)
{
  char big_stack_object[4096*3];
}
```

esp = esp-4096*3 ;allocate 3 stack pages



What about page shrinking? Do not consider about shrinking
Consider only expanding!

Evaluation

Project 4(VM)의 평가 테스트는 총 109개이며 Project 1,2(User program)와 겹치는 부분이 대부분임

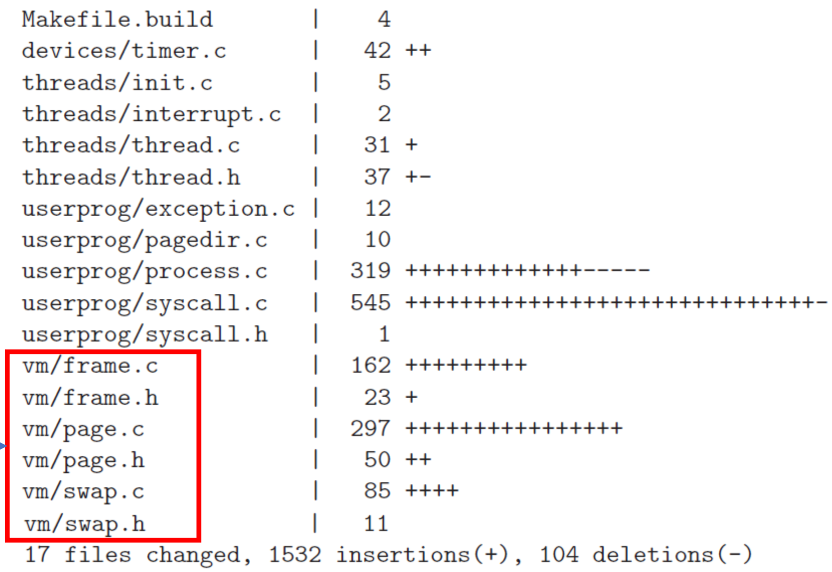
- **제시된 테스트(16개)만 통과하면 됨 (mmap* 제외)**

1) tests/vm/pt-grow-stack
2) tests/vm/pt-grow-pusha
3) tests/vm/pt-grow-bad
4) tests/vm/pt-big-stk-obj
5) tests/vm/pt-bad-addr
6) tests/vm/pt-bad-read
7) tests/vm/pt-write-code
8) tests/vm/pt-write-code2
9) tests/vm/pt-grow-stk-sc
10) tests/vm/page-linear
11) tests/vm/page-parallel
12) tests/vm/page-merge-seq
13) tests/vm/page-merge-par
14) tests/vm/page-merge-stk
15) tests/vm/page-merge-mm
16) tests/vm/page-shuffle

Reference

- 필요 시 src/vm/ 에 직접 파일 작성하여 추가
  - **(중요) 추가한 파일은 src/Makefile.build 에 추가하여야 함**

You can follow this approach if you want



```
Makefile.build          |    4
devices/timer.c         |   42 ++
threads/init.c          |    5
threads/interrupt.c     |    2
threads/thread.c        |   31 +
threads/thread.h        |   37 +-
userprog/exception.c    |   12
userprog/pagedir.c      |   10
userprog/process.c      |  319 +++++++++++++-----
userprog/syscall.c      |  545 ++++++++++++++++++++++++++++++-
userprog/syscall.h      |    1
vm/frame.c              |  162 +++++++++
vm/frame.h              |   23 +
vm/page.c               |  297 ++++++++++++++++
vm/page.h               |   50 ++
vm/swap.c               |   85 ++++
vm/swap.h               |   11
17 files changed, 1532 insertions(+), 104 deletions(-)
```

Submission

- Team Project로 진행한다.
- Deadline : **2018년 12월 26일 23시 59분** **(지각제출 불허)**
- 사이버 캠퍼스 제출
  - 압축 파일의 이름은 아래와 같다.

항목	형식	예시(project 4, 7조)
파일 제목	os_prj#_##.tar.gz	os_prj4_07.tar.gz

  - 자세한 압축 방법은 **공지사항의 '프로젝트 압축 방법'** 및 OS project guide를 참고한다.
  - 팀원 중 한 사람만 제출한다.
- Document 제출
  - **AS916**에 hardcopy 제출
(hard copy의 deadline도 source code의 deadline과 동일합니다.)