

Kennesaw State University
CSE 3502 Operating Systems - Spring 2020
Project 3 - Memory Management

Points Possible: 100

Given the above virtual memory areas used by a process, write a system call *sys_getPageInfo* (follow the steps in Project 1) to report the current status of specific addresses in these virtual memory areas. The system call takes the pid of a process as input and outputs the following information of start address *vm_start* in each *vm_area_struct* that the process has:

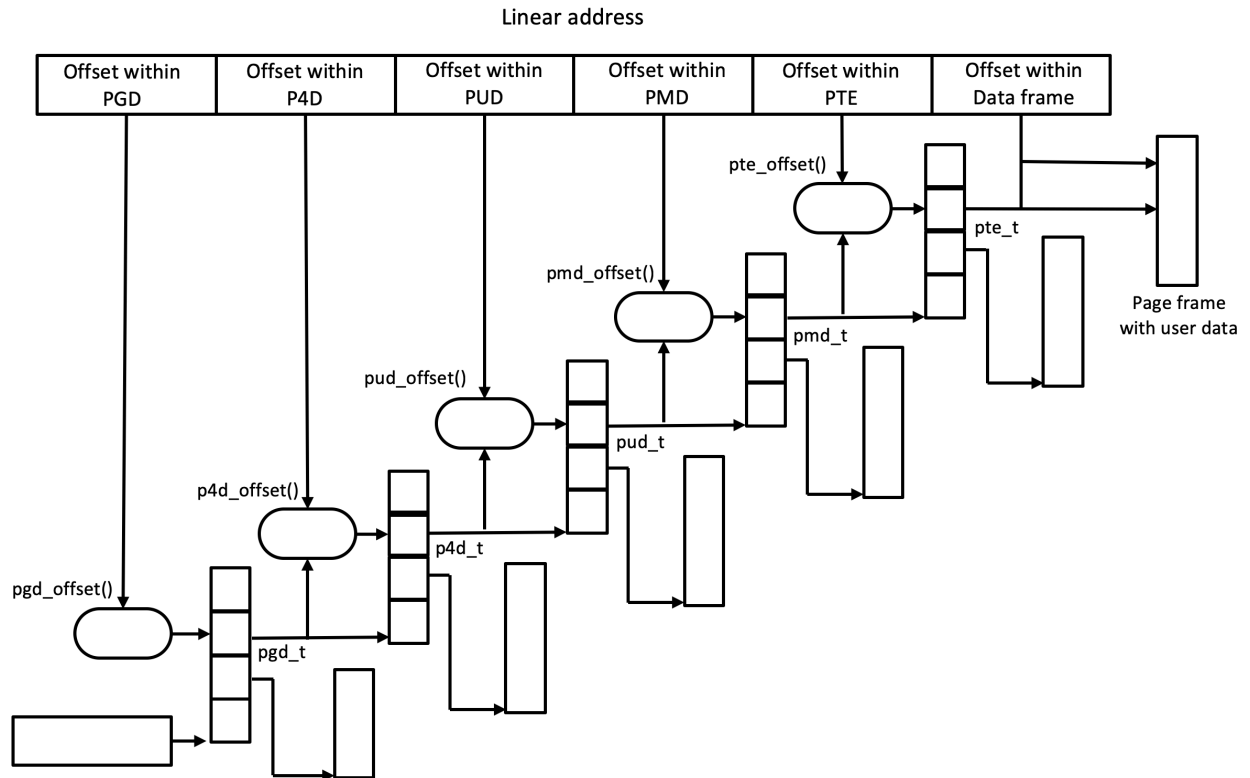
1. If the data in this address is in memory or on disk.
2. If the page which this address belongs to has been referenced or not.
3. If the page which this address belongs to is dirty or not.

Hints:

The Linux kernel uses the memory descriptor data structure to represent a process's address space. The memory descriptor struct *mm_struct* is defined in *<linux/mm_types.h>* and is included in a process's data structure *task_struct*.

In *mm_struct*, the *mmap* field points to a linked list of struct *vm_area_struct*. The *vm_area_struct* describes a single memory area over a contiguous interval in an address space. The *vm_start* and *vm_end* point are the start and end address of individual *vm_area_struct*. All the virtual memory areas together form a process's virtual address space.

Using user-level programs (user.c) to test your system call.
Data structure:



Hints:

The page descriptor (data struct *pgd_t*) contains information about the page. You need to figure out how to obtain a reference to the page descriptor given a virtual address and read to information from the page descriptor. To get current status of specific addresses, you need to test the corresponding flag bit of the address's page table entry (data struct *pte_t*). Note that Linux uses multi-level page tables, you might need multiple steps to reach the page table entry of a given virtual address.

Table: Page Table Entry Status Bits

Bit	Function
<code>_PAGE_PRESENT</code>	Page is resident in memory and not swapped out.
<code>_PAGE_ACCESSED</code>	Set if the page is referenced.
<code>_PAGE_DIRTY</code>	Set if the page is written to.

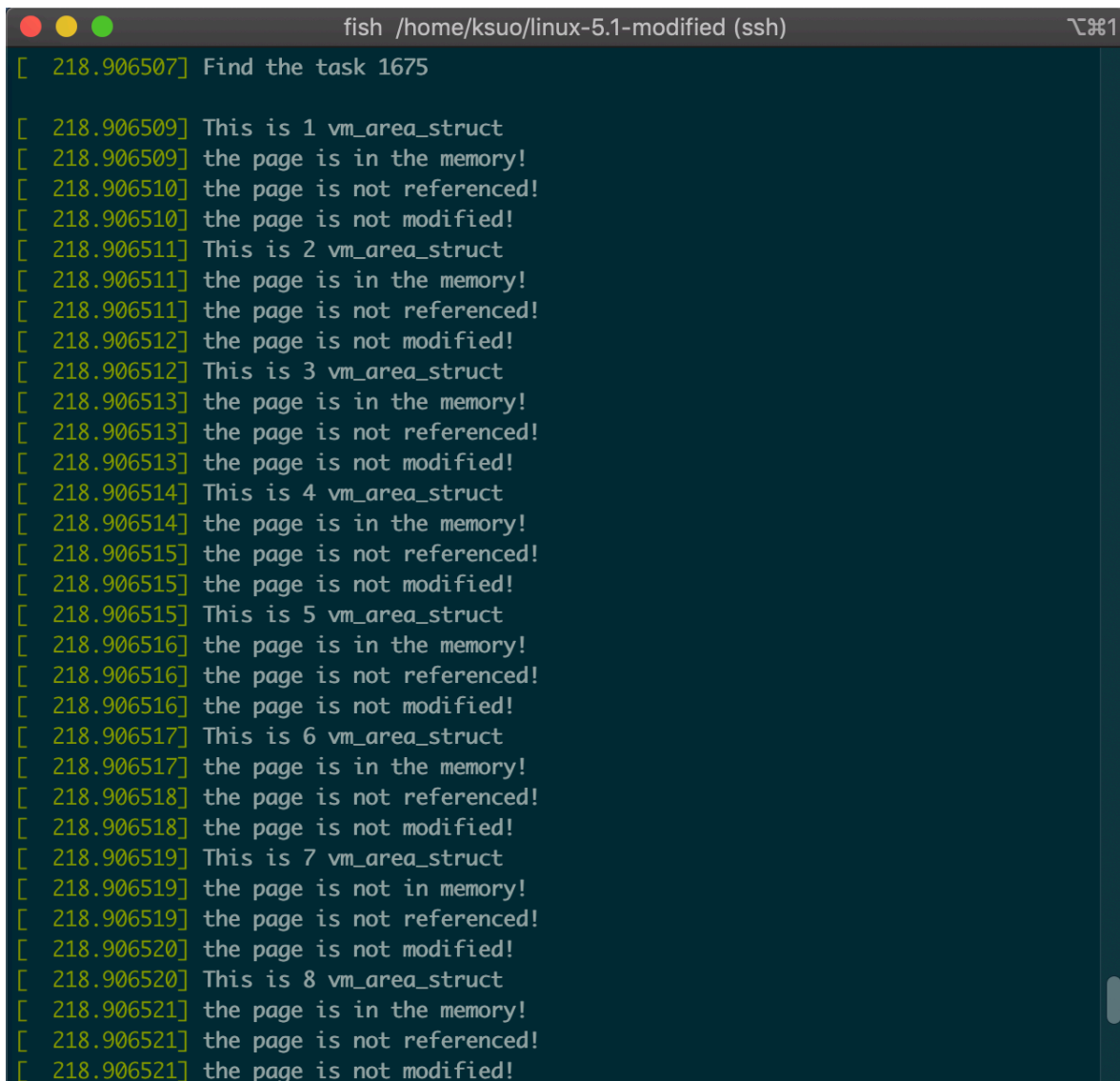
The function *pte_present* is used for judging a page is in memory or disk. If the return value is 1, the page is in the memory, otherwise it is in the disk. The function *pte_young* is used for judging a page is referenced or not. If the return value is 1, the page has been referenced, otherwise it has not been referenced yet. The function *pte_dirty* is used for judging a page is dirty or not. If the return value is 1, the page is dirty, otherwise it is not dirty.

Submission requirements:

- (1) Create one folder named **kernel** and put your modification (file diff1.txt, diff2.txt, ...) of the Linux kernel into this folder. Please use diff command to highlight your modification:
`$ diff -u original_file.c modified_file.c > diff.txt`
- (2) Create another folder named **user** and put your user-level test programs inside.
- (3) Create a folder to put the screenshot of your program output.
- (4) Put **kernel folder, user folder and output screenshot folder** into folder **Assignment-2**.

Zip all the files and folders together into one zip file and name it as CS3502_[your D2L user name], e.g., **CS3502_mahmed29.zip**, and upload the file onto D2L.

Examples of output (`$ dmesg`):



```
fish /home/ksuo/linux-5.1-modified (ssh) 1
[ 218.906507] Find the task 1675
[ 218.906509] This is 1 vm_area_struct
[ 218.906509] the page is in the memory!
[ 218.906510] the page is not referenced!
[ 218.906510] the page is not modified!
[ 218.906511] This is 2 vm_area_struct
[ 218.906511] the page is in the memory!
[ 218.906511] the page is not referenced!
[ 218.906512] the page is not modified!
[ 218.906512] This is 3 vm_area_struct
[ 218.906513] the page is in the memory!
[ 218.906513] the page is not referenced!
[ 218.906513] the page is not modified!
[ 218.906514] This is 4 vm_area_struct
[ 218.906514] the page is in the memory!
[ 218.906515] the page is not referenced!
[ 218.906515] the page is not modified!
[ 218.906515] This is 5 vm_area_struct
[ 218.906516] the page is in the memory!
[ 218.906516] the page is not referenced!
[ 218.906516] the page is not modified!
[ 218.906517] This is 6 vm_area_struct
[ 218.906517] the page is in the memory!
[ 218.906518] the page is not referenced!
[ 218.906518] the page is not modified!
[ 218.906519] This is 7 vm_area_struct
[ 218.906519] the page is not in memory!
[ 218.906519] the page is not referenced!
[ 218.906520] the page is not modified!
[ 218.906520] This is 8 vm_area_struct
[ 218.906521] the page is in the memory!
[ 218.906521] the page is not referenced!
[ 218.906521] the page is not modified!
```

Possible errors:

1, No space left on device

Please make sure your VM has 60GB or 80GB storage because the kernel compiling takes lots of space.

2, segmentation error

Most of the segmentation was due to incorrect use of memory pointer. Please check the pointer is null or not before using them.

3, system call not work

Make sure your compiling has no error and your VM is booted with the new kernel you just compiled and installed. If your system call here does not work, please check how you did in project 1.

If you have errors in your code, the best way to debug is to add “printk” or comment part of your code to locate where the bug code exists.