1 Objective

In this assignment, you will add a new system call, `addr_info()`, to the Linux kernel that provides various system information about a given virtual address, such as the associated page frame number, physical address, and access permissions.

2 Resources

Read this!!!!


You may also find the following functions/macros useful:

- `kmalloc()`: used to obtained byte-sized memory chunks of kernel memory
- `kfree()`: frees blocks of memory previously allocated by `kmalloc()`
- `sys_brk()`: system call to change the data segment size.
- `page_address()`, `lookup_address()`: address lookup functions
- `make_pages_present()`, `get_user_pages()`: used in managing user pages.

The following files may prove useful as well: `pgtable.h`, `mm.h`, `page-flags.h`, `pageattr.c`

3 Assignment

There are three parts to this assignment. First, you need to define a new structure that your new system call will populate, with information on the supplied virtual address, when it is invoked. Next, you need to add the `addr_info()` system call to your kernel. Last, you need to implement your new system call to find and report kernel-level information on the supplied virtual address. This last step is the most difficult and should be done incrementally (e.g. implement one field of the structure at a time).
3.1 Adding a virtual memory information structure

You must add the structure below in the file include/linux/addr_info.h. Also, you must make the constants (used to access information contained in the structure’s flags) indicated below available. Make sure you separate your include file so that it correctly gives the information needed for either user programs and kernel code.

```c
struct addr_info {
    int user_count;  /* No. of users who’s VM contains this addr */
    unsigned long ma_start_addr;  /* starting virtual addr of memory area */
    unsigned long ma_end_addr;  /* ending virtual addr of memory area */
    unsigned long ma_permissions;  /* virtual memory area permissions */
    unsigned long ppn;  /* associated physical page number */
    unsigned long vpn;  /* associated virtual page number */
    unsigned long physical_addr;  /* associated physical address */
    unsigned long page_flags;  /* page flags */
};
```

ma_permissions flag values:
1) [http://lxr.linux.no/source/include/linux/mm.h?v=2.6.11#L138](http://lxr.linux.no/source/include/linux/mm.h?v=2.6.11#L138)

page_flags flag values:
1) [http://lxr.linux.no/source/include/linux/page-flags.h?v=2.6.11#L12](http://lxr.linux.no/source/include/linux/page-flags.h?v=2.6.11#L12)

3.2 Adding the addr_info() System Call

You are going to add a system call, addr_info(), that returns information about a given virtual address. The addr_info() system call takes a pointer to a struct addr_info and an unsigned long. The unsigned long contains the virtual address. The system call will fill out the struct addr_info structure with information appropriate to the given virtual address. If an error occurs, then the system call will return a negative number that indicates the nature of the error. On success, the call returns 0. Therefore, the prototype of the function is:

```c
int addr_info(struct addr_info *aip, unsigned long vaddr);
```

3.3 Using addr_info()

You need to write a user-level function that takes a virtual address as input and will print information about the address to standard out. Here’s the prototype of the function:

```c
void print_addr_info(unsigned long vaddr);
```

The output should be formatted to conform to the following template:

```
Virtual address: <vaddr>
Physical address: <paddr>
```
Virtual page number: <vpn>
Physical page number: <ppn>
Virtual page number: <vpn>
Virtual memory area starting address: <vma_start>
Virtual memory area ending address: <vma_end>
User reference count: <user_count>
Virtual memory area flags:
   <FLAG>
   ...
   <FLAG>
Page flags:
   <FLAG>
   ...
   <Flag>

Some things to consider:

- What if the page associated with the virtual address is not resident in memory?
- Can you sleep during the addr_info() system call?
- When you demo your work in lab, how are you going to prove you are reporting correct statistics?
  (Hint: use /proc/<pid>/maps)

4  Attention to Detail

Treat this assignment as if this was a real addition to the Linux kernel. Make sure you protect shared data, validate input parameters, perform error checking, and handle security issues. Also, your implementation should be as efficient as possible, e.g. inline functions where feasible.

Deliverables

You need to demonstrate the completion of this assignment in lab on May 7. The demonstration will consist of viewing your code, running several user-space programs (that you must have ready for the demo) that use print_addr_info(), and convincing me that your results are correct (you must have proof!).

You also need to handin a kernel patch that implements addr_info() and a README file (must be spelled as shown...no lowercase) that describes your implementation of addr_info(), any problems/bugs with your implementation, and any other information you feel would be useful. If your demonstration is successful, these files will just be checked off that they were submitted. If your demonstration is unsuccessful, you will be awarded some partial credit for submitting these files. These files must be submitted by midnight on May 7.