# CS3523: OS-II - ASSIGNMENT 3

**Aim: The aim of this assignment is to understand how virtual memory paging happens in Linux.**

## Background:

Linux supports mmap() and munmap() system calls on regular files (eg. /etc/os-release), device special files (eg. /dev/zero). The mmap()/munmap() are the most heavily used system calls in Linux.

For example, when we run any program (eg. ls, cat, firefox, etc), the program executable file is loaded into memory using many mmap() calls. Superior performance of mmap() is the key for faster loading of programs and booting of the OS. By deferring loading of program fragments (eg. segments of text, data) from disk, till they are really needed is the primary reason for Linux's fast boot-up. Conceptually this is called demand paging.

## Problem:

Goal: The goal of this assignment is to implement prefetch and demand paging on a device special file.

Details:

1. Enhance the "`mykmod`" Linux character device driver shared in the assignment, to implement pseudo character devices. Each device instance represents a pseudo device of 1MB. The 1MB is kernel memory allocated using `kmalloc()`.
2. The driver already supports `open(2)` and `close(2)` system calls using `file_operations`. Enhance it to support `mmap(2)` syscall on the devices. Whenever `mmap(2)` is done by an application program, the 1MB in kernel memory must be mapped to the process's address space.
3. No need to implement `read(2)` and `write(2)` syscalls.
4. Implement `open`, `close`, `fault` operations in `vm_operations_struct` for mmap of the 1MB to support prefetching and demand paging.
5. The device retains data that was written to it till the driver unloads or system reboots. Once mmap'd write is done on a device, mmap'd reading from the device must return whatever was written.
6. In the prefetching case, page faults for the entire 1MB are generated in the context of `mmap(2)` itself.
7. In the demand paging case, page faults are generated when the application starts reading/writing from/to the memory.
8. When a VM segment is opened, npagefaults (number of page faults) should be initialized to zero.

9. When a VM segment is closed, it must print (using printk) the npagefaults received, and re-initialize it to zero.

memutil.cpp: The program opens a given device special file using open(2). Does mmap(2) system call followed by read/write memory operations. At last the program unmaps memory using munmap(2) system call, and closes the file using close(2). Skeleton of this program is shared with the assignment.

## Syntax:

```
Usage: ./util/memutil [options] <devname>
Options:
--operation <optype> : Where optype can be mapread, mapwrite
--message <message>  : Message to be written/read-compare to/from the device memory
--paging <ptype>    : Where ptype can be prefetch or demand
--help              : Show this help
```

## Example usages:

### Building & Loading the driver:

```
# cd 99_devmmap_paging
# make
# insmod kernel/mykmod.ko
# grep mykmod /proc/devices
243 mykmod
```

### Prefetch (Read) :

```
# mknod /tmp/mydev_pR6 c 243 10
# ./util/memutil /tmp/mydev_pR6 --pt prefetch --op mapread

# dmesg | grep -e mykmod_vm_open -e mykmod_vm_close
[  476.174464] mykmod_vm_open: vma=ffff9971ee1dfaf8 npagefaults:0
[  476.178813] mykmod_vm_close: vma=ffff9971ee1dfaf8 npagefaults:256
```

### Demand paging (Read):

```
# mknod /tmp/mydev_JZl c 243 11
# ./util/memutil /tmp/mydev_JZl --pt demand --op mapread

# dmesg | grep -e mykmod_vm_open -e mykmod_vm_close
[  476.193956] mykmod_vm_open: vma=ffff9971ee182288 npagefaults:0
[  476.197009] mykmod_vm_close: vma=ffff9971ee182288 npagefaults:128
```

## Prefetch paging (Write & Read):

```
# mknod /tmp/mydev_fBc c 243 20
# ./util/memutil /tmp/mydev_fBc --pt prefetch --op mapwrite --op mapread --mes test2

# dmesg | grep -e mykmod_vm_open -e mykmod_vm_close
[  476.209981] mykmod_vm_open: vma=ffff9971ee1d8ca8 npagefaults:0
[  476.211928] mykmod_vm_close: vma=ffff9971ee1d8ca8 npagefaults:256
[  476.212130] mykmod_vm_open: vma=ffff9971ee1d8ca8 npagefaults:0
[  476.214705] mykmod_vm_close: vma=ffff9971ee1d8ca8 npagefaults:256
```

## Demand paging (Write & Read):

```
# mknod /tmp/mydev_Ln5 c 243 21
# ./util/memutil /tmp/mydev_Ln5 --pt demand --op mapwrite --op mapread --mes test2

# dmesg | grep -e mykmod_vm_open -e mykmod_vm_close
[  476.225507] mykmod_vm_open: vma=ffff9971f3559360 npagefaults:0
[  476.226311] mykmod_vm_close: vma=ffff9971f3559360 npagefaults:128
[  476.226408] mykmod_vm_open: vma=ffff9971f3559360 npagefaults:0
[  476.228335] mykmod_vm_close: vma=ffff9971f3559360 npagefaults:128
```

## Unloading the driver:

```
# rm -f /tmp/mydev*
# rmmod mykmod
```