CS3523:OS-2 ASSIGNMENT 3

REPORT

PLAGIARISM STATEMENT

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand our responsibility to report honour violations by other students if we become aware of it.

Names: B.Chiranjeevi Sai Ganesh,T.Krishna Prasanth.

Date:21-06-2020.

Signatures: Sai Ganesh, Krishna Prashanth.

mykmod\_main.c file:-

We used devinfo struct to keep per device information which contains pointer to data and size of data.

struct devinfo {

char \*data;

size\_t size;

};

We used linked list data structure to keep track of all devices, in which each node contains pointer to devinfo, minor no. of the device special file, forward pointer and backward pointer.We have declared head and tail pointers of struct list globally.

struct list {

struct devinfo \*info;

int minor; //minor number for reference

struct list \*p;

struct list \*pre\_p;

};

We used vma\_struct to keep per VMA info, which contains pointer to devinfo and no. of page faults.

struct vma\_struct {

struct devinfo \*info;

int npagefaults;

};

In mykmod\_init\_module function,

We used the register\_chrdev function to register our device driver in kernel and to get a major no. for our driver and we have initialized head and tail pointers to NULL.

In mykmod\_cleanup\_module,

This function unregisters our device driver from the kernel.

We have freed the memory allocated to store device info and we have also freed the memory allocated for the device table.

In mykmod\_open,

We have allocated memory to store device info and thus the obtained pointer is stored in the info variable.

and we have allocated MYDEV\_LEN size memory and thus the obtained pointer is stored in info->data and info->size contains MYDEV\_LEN.

We have assigned info to inodep->i\_private and filep->private\_data. (inodep and filep are parameters of mykmod\_open function).

And we have made an entry in the device table. We used minor no. for identifying the devices.

In mykmod\_close,

We have freed all memory in mykmod\_cleanup\_module.

In mykmod\_mmap,

We have created a vminfo variable, which is a pointer to struct vma\_struct,which points to dynamically allocated memory.

Variable flip is supplied as parameter to mykmod\_mmap,

We have assigned filp->private\_data to vminfo->info and vminfo to vma->vm\_private\_data.

Variable vma is supplied as parameter to mykmod\_mmap,

We assigned &mykmod\_vm\_ops to vma->vm\_ops.

And we have assigned necessary flag to vma->vm\_flags.

We have called the function mykmod\_vm\_open passing vma as argument.

In mykmod\_vm\_open,

We have initialized vma->vm\_private\_data->npagefaults to 0.

In mykmod\_vm\_close,

We have freed the memory in which we stored the per vma info.

In mykmod\_vm\_fault,

We have incremented no. of page faults by one.

We have calculated the physical address of the page where recent page fault happened by using vma->vm\_pgoff and vmf->pgoff and converted that physical address into page number.

And we obtained a struct page by using the pfn\_to\_page function, we called the get\_page function by passing the page as argument and assigned the page to vmf->page.

Memutil.cpp file:-

In this file we have assigned MAP\_SHARED to mmap\_flags in case of demand paging and MAP\_POPULATE in case of prefetch.

In case of reading a file,

We have implemented a for loop(through which we traverse the whole file) to read data from the file. If the data does not match with msg we report error. Reading the file happens even if the msg\_len is 0.

In case of writing to a file,

We have used a nested for loop to fill the whole 1MB with msg by repeatedly writing the same msg.

Everything putting together:-

When we run the following commands:-

1)# insmod kernel/mykmod.ko

mykmod\_init\_modlue is called and our device driver gets a major no.

2)# grep mykmod /proc/devices

We get major no. for our device driver.

3) # mknod

We can make a device special file using this command.

When the open function in memutil.cpp is called mykmod\_open function in mykmod\_main.c is executed.

When the mmap function in memutil.cpp is called mykmod\_mmap function(which internally calls mykmod\_vm\_open function)in mykmod\_main.c is executed.

When the close function in memutil.cpp is called mykmod\_close function in mykmod\_main.c is executed.

Whenever page faults happen while reading or writing to file mykmod\_vm\_fault function in mykmod\_main.c is executed.

In case of prefetch all page faults happen at the time of mmap function call.

But in the case of demand paging page faults happen only when reading or writing happens on a new page.

And finally we can unregister our device driver using

“rmmod mykmod” command,when this command is executed mykmod\_cleanup\_module function in mykmod\_main.c is executed.

Sample Input/Output:-

