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: G PRAJWAL, REVANTH ROKKAM

Date:25-06-2020

Signatures: GP,RR

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

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

Stepwise Implementation

mykmod_main.c:

1.For storing the per device info we declare **struct mykmod_dev_info** which includes **char* data** and **int* pageid.**

2.For storing the device table we declare **struct mykmod_dev_info * Table[256] ;** and we initialize **dev_count** to zero since it keeps track of the devices.

3. For keeping the data for vma area we declare **struct vma_info** which has **atomic_t count, struct mykmod_dev_info *dev_data,long npagefaults.**
The npagefaults is initialized to zero.

4.In **mykmod_init** module no need to initialize the device table since we already declared it globally.

5.In **mykmod_cleanup** module we clean up the device table by using “for loop and **kfree()** function.

6.In **mykmod_open** we allocate memory for devinfo and store it in the device table (**Table[dev_count]**) and **i_private** and we increment the **dev_count**

7.The file data in the kernel region is mapped using **mykmod mmap** module. We Initialize structure with **dev_info** and **npagefaults** and The mykmod vm ops functions are given to **vma->vm ops** for the operations in the implementation of the virtual memory regions and corresponding flags and other variables In that and the flags corresponding are set.

8. In mykmod_open we stored
filep->private_data = inodep->i_private;
So that the **dev->dev_data** only had access to **filep->private_data**

9.**Mykmod vm fault** module will be called automatically when a page fault occurs during mapping the kernel data is initialized with the structure mykmid dev info and vma track. Virtual address mapping is done to get the age accordingly. Finally npagefaults is incremented by “1”.

Memutil.cpp:

It is the file used to check driver usage from the user area.The file reads the message “msg” that is passed by the argument

1.when the paging is demand paging then the flag used is **mmap_flags = MAP_SHARED.**

2.when the paging is prefetch then the flag used is **mmap_flags = MAP_SHARED | MAP_POPULATE**

3.If the message passed is based on read operation(**MAPREAD**) then, we memory map the **dev_mem** 's kernel buffer into user-space segment, and then we use by using for loop we Compare the data read from devicemem with msg. If the data is unmatched then munmapping and **return EXIT_FAILURE** right away. After checking we unmap the devicemem's kernel buffer.

4.If the message passed is based on read operation(**MAPWRITE**) then, we memory map the **dev_mem** 's kernel buffer into user-space segment,
and then we use by using for loop we write the message passed in to **dev_mem** and finally unmapping.

OUTPUT:


root@cs3523:~/99_devmmap_paging
File Edit View Search Terminal Tabs Help
root@cs3523:~/99_devmmap_paging
PASS - Test 1 : Single process reading using mapping
PASS - Test 2 : Single process writing using mapping
PASS - Test 3 : Multiple process reading using mapping
PASS - Test 4 : Multiple process writing using mapping
PASS - Test 5 : One process writing using mapping and other process reading usin
g mapping
PASS - Test 6 : One process writing to one dev and other process reading from an
other dev
[root@cs3523 99_devmmap_paging]#