ReadMe

In, this question we are required to do two things:

a. We need to add the syscall to the table of already existing sys calls.

b. Then we need to edit the files where the implementation of all syscalls are present and there we need to put in the implementation of our kernel\_2d\_memcpy.

Adding syscall to table of syscalls:

We first go to build/linux-6.xx.xx/arch/x86/entry/syscalls/ syscall\_64.tbl . Now depending upon our linux version we will have 450+ sys calls . We now add a new syscall as follows:

## 451 comman kernel\_2d\_memcpy sys\_kernel\_2d\_memcpy

Editing the files using vim./kernel/sys.c

Now we need to go to build/linux-6.xx.xx/kernel then we write ‘vim/kernel/sys.c’. Then, we have write SYSCALL\_DEFINE0 for kernel\_2d\_memcpy

# SYSCALL\_DEFINE4(kernel\_2d\_memcpy, float \*, mat1, float \*, mat2, int, row\_number, int, column\_number)

Here we take in 4 parameters, first we have the name of our sys call then we input two matrix and the other two parameters are the row and column.

We use two functions to achieve the same which are copy\_from\_user and copy\_to\_user:

In copy\_from\_user to get matrix1 copy to temporary matrix, we input a temporary matrix as our destination matrix and on our source we input another matrix . Now for bytes to copy we simply use sizeof(matrix) function. If this does not return 0 then we print an error.

In copy\_to\_user to copy matrix2 from temporary matrix, Here we input our matrix 1 where we need to provide the data of temperoy matrix we have created. For bytes, again simply use sizeof() function. If this does not return 0 then we print an error.

After doing the following we just need to update our kernel and thus we have achieved the making of kernel\_2d\_memcpy syscall.