SYSCALL IMPLEMENTATION

SYSCALL_DEFINE3 is used as 3 parameters are passed.
Both arrays are defined in the user space, so their data is not directly accessible to the kernel so copy_to_user and copy_from_user functions are used.

```
SYSCALL_DEFINE3(kernel_2d_memcpy,
                void __user *, src_array,
                void __user *, dst_array,
                unsigned long, len)
```

Here len contains the number of bytes to be copied from the source array to the destination array.

Pointer to source array, a pointer to destination array, and the length of bytes to be copied is passed as arguments to the syscall
To get the desired result two functions are used copy_from_user() and coy_to_user().
Firstly the data in the source array is copied to a temp array by using a pointer to this array(which is defined in the kernel space) using copy_from_user()
Then this temp array data is now used and is copied to the destination_array by the function copy_to_user().

Error Handling and Return Values:
If the process is successful then the syscall would return 0, if the syscall is not able to copy, all the bytes EFAULT is returned and if the syscall is not found then -1 is returned.
If the data to be copied was too long or if there was a problem copying, we EFAULT is returned.
Kernel log:
Also when the system call is used, data relevant to the syscall is also printed in the kernel logs.

In case of any error it is printed in the kernel log and the log can be accessed by dmesg | tail

Returned Values:
0 if syscall is implemented successfully.
EFAULT if there is any error.

Diff file contains the difference between the stock kernel and the patched kernel containing the newly implemented syscall.