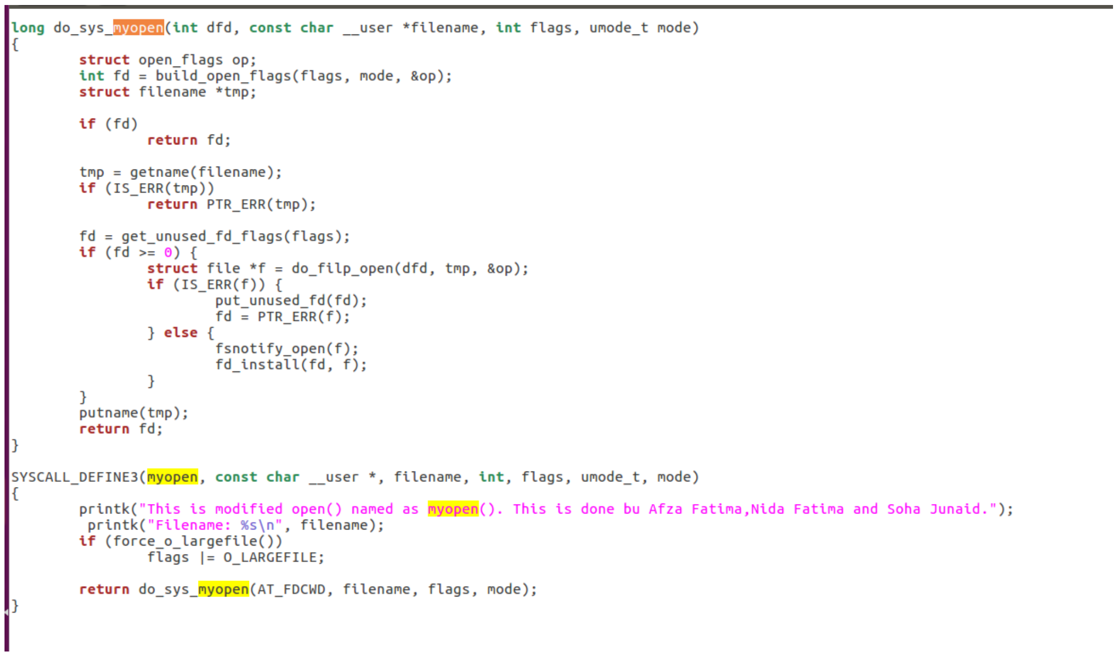
Explanation of Modifications in system calls

This document will explain the changes done in the kernel source files along with their snippets.

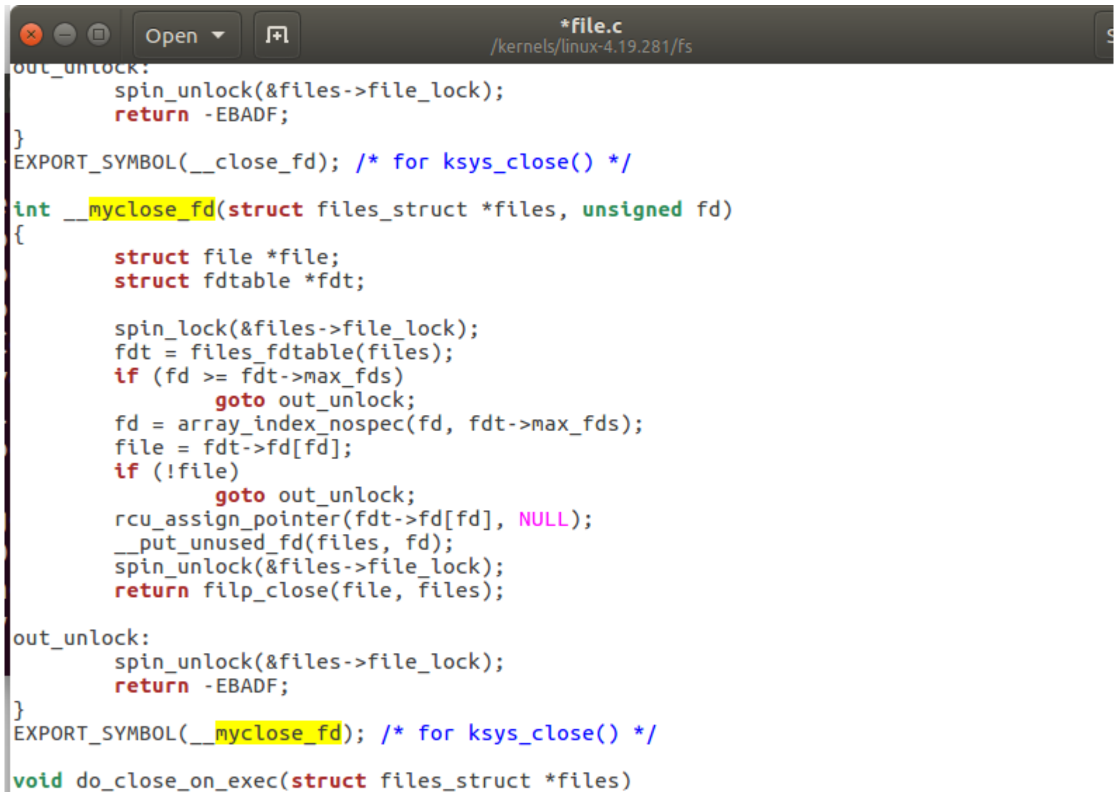
For open() and close() system call:

The code snippet below shows the myopen() in open.c file in the linux-4.19.281/fs directory. . It also displays additional information using printk to the kernel logs when a file is opened.



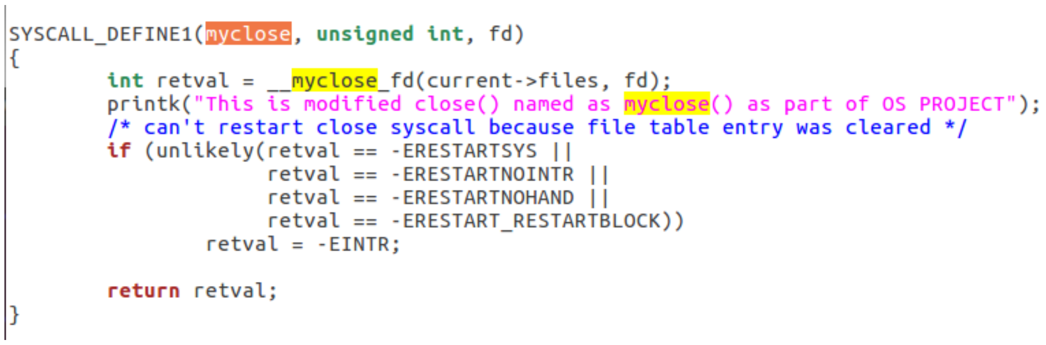
```
long do_sys_myopen(int dfd, const char __user *filename, int flags, umode_t mode)
{
        struct open_flags op;
        int fd = build_open_flags(flags, mode, &op);
        struct filename *tmp;

        if (fd)
                return fd;

        tmp = getname(filename);
        if (IS_ERR(tmp))
                return PTR_ERR(tmp);

        fd = get_unused_fd_flags(flags);
        if (fd >= 0) {
                struct file *f = do_filp_open(dfd, tmp, &op);
                if (IS_ERR(f)) {
                        put_unused_fd(fd);
                        fd = PTR_ERR(f);
                } else {
                        fsnotify_open(f);
                        fd_install(fd, f);
                }
        }
        putname(tmp);
        return fd;
}

SYSCALL_DEFINE3(myopen, const char __user *, filename, int, flags, umode_t, mode)
{
        printk("This is modified open() named as myopen(). This is done bu Afza Fatima,Nida Fatima and Soha Junaid.");
         printk("Filename: %s\n", filename);
        if (force_o_largefile())
                flags |= O_LARGEFILE;

        return do_sys_myopen(AT_FDCWD, filename, flags, mode);
}
```

The code snippet below shows the myclose() in file.c file in the linux-4.19.281/fs directory.


*file.c
/kernels/linux-4.19.281/fs

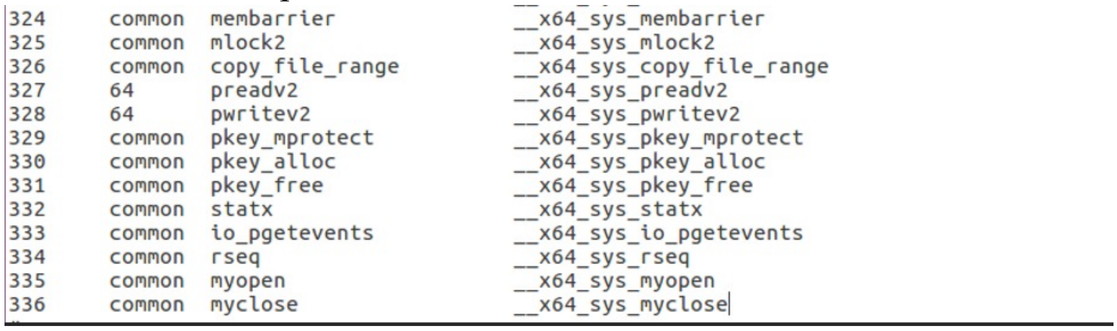
```c
out_unlock:
        spin_unlock(&files->file_lock);
        return -EBADF;
}
EXPORT_SYMBOL(__close_fd); /* for ksys_close() */

int __myclose_fd(struct files_struct *files, unsigned fd)
{
        struct file *file;
        struct fdtable *fdt;

        spin_lock(&files->file_lock);
        fdt = files_fdtable(files);
        if (fd >= fdt->max_fds)
                goto out_unlock;
        fd = array_index_nospec(fd, fdt->max_fds);
        file = fdt->fd[fd];
        if (!file)
                goto out_unlock;
        rcu_assign_pointer(fdt->fd[fd], NULL);
        __put_unused_fd(files, fd);
        spin_unlock(&files->file_lock);
        return filp_close(file, files);

out_unlock:
        spin_unlock(&files->file_lock);
        return -EBADF;
}
EXPORT_SYMBOL(__myclose_fd); /* for ksys_close() */

void do_close_on_exec(struct files_struct *files)
```

Its system define can be found in open.c which prints additional information using printk in kernel logs.



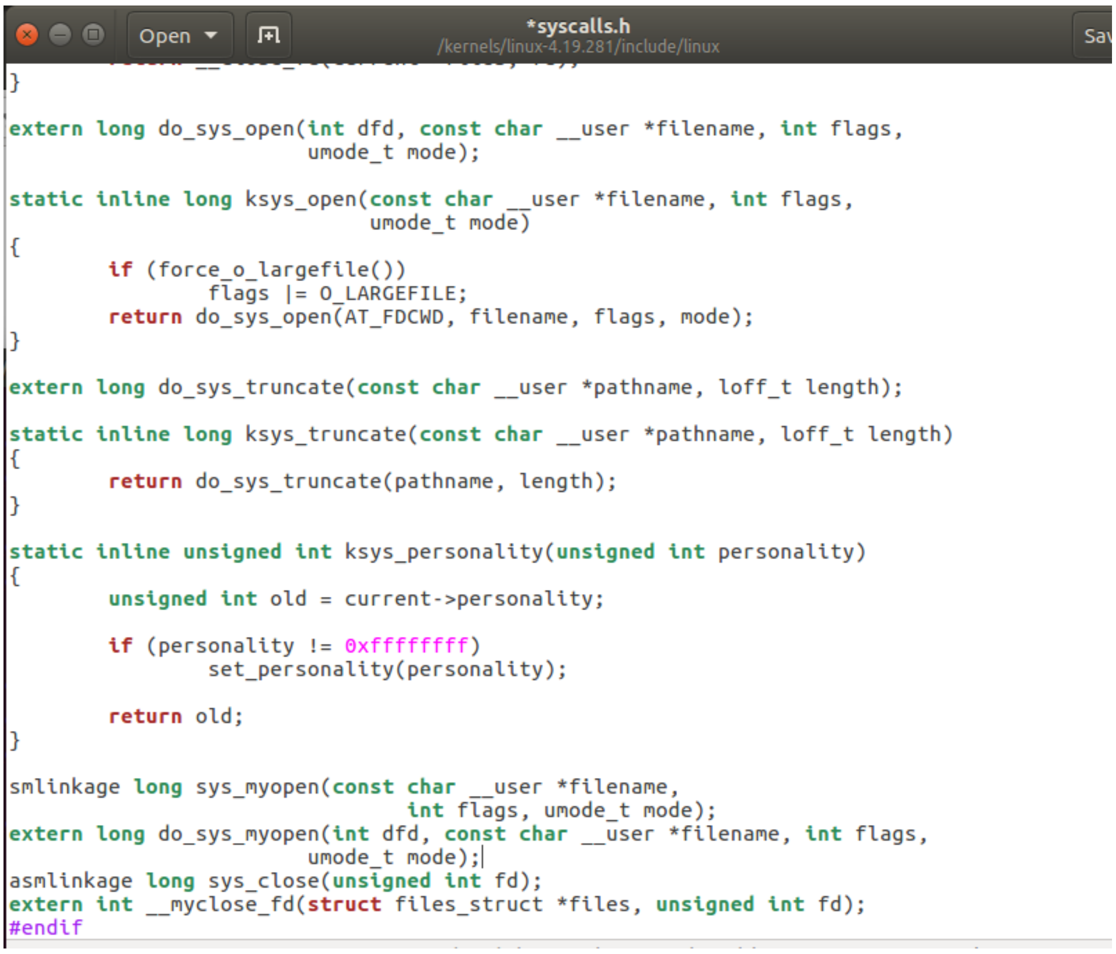
```c
SYSCALL_DEFINE1(myclose, unsigned int, fd)
{
        int retval = __myclose_fd(current->files, fd);
        printk("This is modified close() named as myclose() as part of OS PROJECT");
        /* can't restart close syscall because file table entry was cleared */
        if (unlikely(retval == -ERESTARTSYS ||
                     retval == -ERESTARTNOINTR ||
                     retval == -ERESTARTNOHAND ||
                     retval == -ERESTART_RESTARTBLOCK))
                retval = -EINTR;

        return retval;
}
```

We add the modified calls in the syscall_64.tbl in the /arch/x86/entry/syscalls directory so that we can use the 335 and 336 reference number to make a call to this modified open and close call when we need to open or close the file.



```
324     common  membarrier              __x64_sys_membarrier
325     common  mlock2                  __x64_sys_mlock2
326     common  copy_file_range         __x64_sys_copy_file_range
327     64      preadv2                 __x64_sys_preadv2
328     64      pwritev2                __x64_sys_pwritev2
329     common  pkey_mprotect           __x64_sys_pkey_mprotect
330     common  pkey_alloc              __x64_sys_pkey_alloc
331     common  pkey_free               __x64_sys_pkey_free
332     common  statx                   __x64_sys_statx
333     common  io_pgetevents           __x64_sys_io_pgetevents
334     common  rseq                    __x64_sys_rseq
335     common  myopen                  __x64_sys_myopen
336     common  myclose                 __x64_sys_myclose
```

Then we write call declarations in the syscall.h header file in the directory /include/linux so that the kernel registers them while building



```
}

extern long do_sys_open(int dfd, const char __user *filename, int flags,
                        umode_t mode);

static inline long ksys_open(const char __user *filename, int flags,
                             umode_t mode)
{
        if (force_o_largefile())
                flags |= O_LARGEFILE;
        return do_sys_open(AT_FDCWD, filename, flags, mode);
}

extern long do_sys_truncate(const char __user *pathname, loff_t length);

static inline long ksys_truncate(const char __user *pathname, loff_t length)
{
        return do_sys_truncate(pathname, length);
}

static inline unsigned int ksys_personality(unsigned int personality)
{
        unsigned int old = current->personality;

        if (personality != 0xffffffff)
                set_personality(personality);

        return old;
}

smlinkage long sys_myopen(const char __user *filename,
                                int flags, umode_t mode);
extern long do_sys_myopen(int dfd, const char __user *filename, int flags,
                        umode_t mode);
asmlinkage long sys_close(unsigned int fd);
extern int __myclose_fd(struct files_struct *files, unsigned int fd);
#endif
```

then we reboot the kernel after running make <target> and installing modules. That's how a system call is modified.