The requirement for basic assignment is to implement file descriptor table and open file table.

file descriptor table are all the file descriptors that current process is using.

Each descriptor have an open file entry (which is the file struct below) in open file table.

So we can easily access process fd table throught curproc and each fd point to a file struct (see below struct details).

struct file {

/\* point to the vnode in vfs \*/

struct vnode \*file\_vnode;

/\* create a lock for exclusive access to the open file struct \*/

struct lock \*file\_lock;

/\* a pointer that use to trace the current offset \*/

off\_t file\_offset;

/\* the mode flag that the process have for this file \*/

int file\_mode;

/\* count how many handles that this file struct have \*/

int file\_ref;

}

file\_open:

1. get file name from userland to kernel.
2. get the index of descriptor table that haven’t populated yet as the descriptor for current open file.
3. Create a new open file struct.
4. Pass to vfs\_open .
5. Use lock\_create to create the open file lock

And other initializations……

1. Store new file struct in the curproc->descriptor\_table [descriptor] (curproc is always the current thread's process.)

file-close:

we need to consider the situation when serval descriptor point to the same file (fork or dup2).

1. Lock the access to the open file table.

2. Check the reference to see if it is the last reference to the file, if so, close file via vfs\_close().

file\_read:

1. Check the input fileNumber is valid.

2. Check the mode is not O\_WRONLY.

3. Then we need Initialize a uio suitable for I/O from a kernel buffer using: “struct iovec iov;” and “struct uio myuio;”

4. The pass to vop\_read with the open file pointer to read in right offset(for this we need calculate the offset using the offset return by uio)

Note that during the process of the read, the file\_lock should be used.

File\_write:

Write is essentially same with the read, just call vop\_write install read, and initialize the UIO in UIO\_WRITE mode and check it is not read only mode.

file-dup2:

simply copy the file struct in current file descriptor to new file descriptor. close the file if newfd is occupied, add one reference to the file ref.

file-lseek:

1. do EBADF and ESPIPE check first.
2. set the offset according to whence (SEEK\_SET, SEEK\_CUR or SEEK\_END).

Note that offset is a 64 bits value so we need to assign two 32 bits arguments to one 64 bits value in file syscall.c .