Security Architecture HW1

User Blast

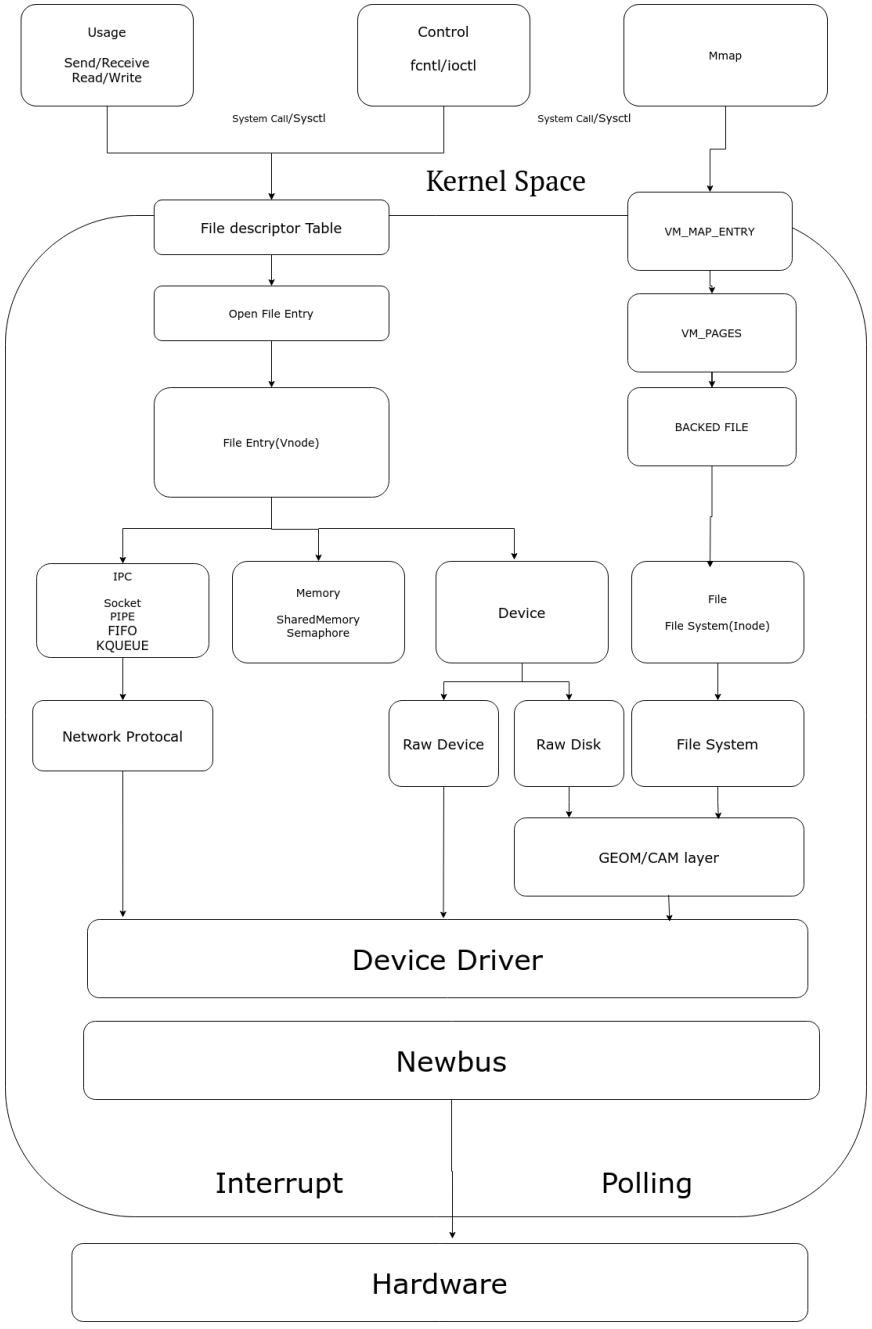
October 19, 2023

Contents

1 Overall

The system I want to describe is the operating system kernel in FreeBSD. Because the FreeBSD operating system kernel provides many services for userspace. In here, We only want to focus on introducing the file access module and it's subsystem for kernel.

The total design chart is as follow:


Usage
Send/Receive
Read/Write
Control
fcntl/ioctl
Mmap
System Call/Sysctl
System Call/Sysctl
Kernel Space
File descriptor Table
VM_MAP_ENTRY
Open File Entry
VM_PAGES
File Entry(Vnode)
BACKED FILE
IPC
Socket
PIPE
FIFO
KQUEUE
Memory
SharedMemory
Semaphore
Device
File
File System(Inode)
Network Protocal
Raw Device
Raw Disk
File System
GEOM/CAM layer
Device Driver
Newbus
Interrupt
Polling
Hardware

2 Functional Architecture

So at first, we have to define our usecase. The user is the process to require the kernel to manipuate a specific file. And the service provider is the operating system kernel and it's backing storage hardware. Notice that the file in here means the generic files (e.g. File, Socket, SharedMemory.. etc). We divided the operatins as the following three categories.

- Open/Read/Write the file
- Control the attribute of the opened file and changed the behavior of these file (e.g. make the file non-block, chage credentia).
- Process also want some security guarantee from it's provider – that is, authentication and role based security framework.

3 Software Architecture

The client (user process) reuires file manipuation (open/read/write/control) from kernel by using (system call and sysctl). Because the operating system want to decouple the implementation of file (e.g. File, Socket). So the kernel uses layer approach. Each layer insider kernel use function call to communicate with each other. And for process safety. The data maintained in the callee is protected by locked. Notice that some layer doesn't use lock (e.g. Per-Process Memory Management) because these resources only allocated on one core.

When entering the kernel, the kernel search the per-process file descriptor entry from the file descriptor given from the user in file descriptor table. It is a pointer to the open file entry. This entry records the process related information for underlining file (e.g. file offset). And open file entry points to the file entry (vnode). It will decide the underlining file implementation (e.g. File, Socket... etc) and call the underlining implementation function. At this steps, the userspace data have been copy to the kernel space.

If we use mmap, the mmap system call will use the memory and demanding paging to cache the whole file. Each mmap creation will or will not backing with a file and the following operatinon is as use the open/read/write system call.

When entering in specific implementation, the requiremnet from userspace is queued and use 1 or more threads in each layer to handle these request sequentially. These requirement may be clustered as a speedup method. Of

course, the kernel may turn on clearner threads to clean-up useless resource like vnode.

In some layers, the kernel doesn't waiting for the whole procedure done. For example, with non-block I/O, the kernel copy data from userspace and return to userspace. The kernel space may use polling threads to polling and do function call to under layer so that the under layer is not called from userspace process.

In the botton of the kernel, kernel have to communicate with underlining hardware. It uses interrupt/polling for sending message and use bus to transfer data. Each hardware interrupt will have 1 thread to do polling.

4 Data Architecture

The data of the operating system kernel is stored in the memory with lock in specific location. We can use kmalloc to require the memory.

The entry of each layer may stores the upper request by using linked list because of the good time complexity of adding/deleting entry (e.g. $\mathrm{VM}_{\mathrm{PAGES}}$) layer. And array for good random access performance (e.g. File descriptor table). Each layer should maintain(lock/unlock) their locks.

For some layer like GEOM/Layer. The may use data structure like ring buffer to store the request in order to sequentially handle effetiently.