**FILESYSTEMS**

Ken Wong  
Washington University, St. Louis  
kenw@arl.wustl.edu  
www.arl.wustl.edu/~kenw

---

**THE UNIX FILESYSTEM**

- Powerful, elegant filesystem from small number of mechanisms

![Filesystem Diagram]

- Types of Files
  
  - *Ordinary File*: An *unstructured* sequence of bytes
  
  - *Directory*: A set (grouping) of files (even other directories)
  
  - *Special File*: An I/O device (Normally in /dev/)

---

**BSD UNIX KERNEL I/O STRUCTURE**

<table>
<thead>
<tr>
<th>System-Call Interface to Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active File Entries</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Socket</th>
<th>VNODE Layer</th>
<th>Special Devices</th>
<th>VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFS</td>
<td>UFS</td>
<td>LFS</td>
<td>Cooked Disk</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network Protocols</th>
<th>MFS</th>
<th>FFS</th>
<th>LFS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Network Drivers</th>
<th>Buffer Cache</th>
<th>Block-Device Driver</th>
<th>Character-Dev. Driver</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Hardware</th>
</tr>
</thead>
</table>

---

**MONOLITHIC OS DEVICE DRIVERS (1)**

- Autoconfigure and Initialize
  
  - *Probes* hardware devices during boot process
  
  - Determines existence and device characteristics → Initialize software state

- Service I/O Requests (Top Half)
  
  - Handles system calls from users
  
  - Executes synchronously with top half of OS kernel and can block

- Service Interrupts (Bottom Half)
  
  - Handles device interrupts asynchronously from the top half of OS kernel
  
  - Does not block

- A *threaded kernel* uses mutex locks in place of manipulating interrupt levels
MONOLITHIC OS DEVICE DRIVERS (2)

User Process
System Call

I/O Library

Function Call

I/O Subsystem

Write Queues

Read Queue

Device Interface

Sw Interrupt @ splbio

Hdw Interrupt @ splimp (caused by device)

BSD UNIX I/O BUFFERS

read(), fread(), mmap()

User Buffer

I/O Buffer

User Space

Kernel Space

fread() 

read()

raw device or 
mmap()

Device Interface

I-NODE (INDEX NODE) STRUCTURE

mode
owners
timestamps
size
direct blocks
single indirect
double indirect
triple indirect
block count
reference count
flags
generation number

w KB
1

data

y GB

x MB

12
d

y GB

> y GB

d

d
d

d

d

UNIX FILESYSTEM DETAILS

Disk

i-list

Data Blocks

Directory

Block

i-nodes

File Name

i-node number
**BSD UNIX BLOCK I/O SYSTEM (1)**

```
write(fd, buf, cnt);
```

```
buf
```

```
logical file
```

```
system buffers
```

```
0 1 2 3
```

```
logical file blocks
```

```
0: #26328 1: #26680 2: #4480 3: #98034
```

```
disk blocks
```

**BSD UNIX SPACE ALLOCATION**

- **Example:** 4,096-byte blocks, 1,024-byte fragments (4096/1024)

- **Block Map**
  - One associated with each cylinder group
  - Records free fragments in the cylinder group
  - Free fragments: 0-3, 4-5, 10-11

<table>
<thead>
<tr>
<th>Bit Map</th>
<th>****</th>
<th>***1</th>
<th>11**</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragment Numbers</td>
<td>0-3</td>
<td>4-7</td>
<td>8-11</td>
<td>12-15</td>
</tr>
<tr>
<td>Block Numbers</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- **6,000-byte file allocation:** 1 full block and 2 consecutive fragments within block
  - May need to split a full block to get the 2 fragments (leaves 2 free)

**SECONDARY STORAGE MANAGEMENT**

- **Space Allocation**
  - **Contiguous Allocation Size**
    - Tradeoff between single file efficiency and system efficiency
  - **File Allocation Methods**
    - **Contiguous**
    - **Chained**
    - **Indexed** (Fixed versus Variable-Length Allocations)
  - **Tracking space assigned to a file (file allocation table)**

- **Free Space Management**
CONTIGUOUS ALLOCATION

- File Allocation Table

<table>
<thead>
<tr>
<th>File Name</th>
<th>Start Block</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileA</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>FileB</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>FileC</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>FileD</td>
<td>19</td>
<td>3</td>
</tr>
</tbody>
</table>

CHAINED ALLOCATION

- File Allocation Table

<table>
<thead>
<tr>
<th>File Name</th>
<th>Start Block</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileX</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

INDEXED ALLOCATION

- File Allocation Table

<table>
<thead>
<tr>
<th>File Name</th>
<th>Index Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>FileX</td>
<td>19</td>
</tr>
</tbody>
</table>

NFS (NETWORK FILE SYSTEM)

- **Goal**: Provide transparent remote access to shared file systems across networks

```
  File-System Interface
    VFS Interface
      Local File System
      Local File System
      Remote File System
        Network
      Disks
      Disks
```

- **vnode**: Numeric id for each file (network-wide unique)

- **Protocol design is independent of Machine, Operating System, Network Architecture, Transport Protocol**
  - Achieved through the use of RPC (Remote Procedure Call) on top of XDR (eXternal Data Representation)

- **RFC 1813**: NFS Version 3 Protocol Specification (Evolution from Version 2 (RFC 1094))
**FILE ORGANIZATION AND ACCESS METHODS**

- **File Organization**: The logical structure of the records (determined by the access patterns)
  - Standard Unix provides only a flat file organization
- **Physical Organization**: The actual storage structure (e.g., blocking, file allocation)
- **File Organization Criteria (sometimes conflicting)**
  - Access speed
  - Update simplicity
  - Storage requirements
  - Maintenance
  - Reliability

**FILES, RECORDS, FIELDS**

- **Sequential File**

```
B 3    E 7    •••    X4
```

- **Indexed Sequential File**

```
E 7    G 1    •••    X4
```

**BASIC FILE ORGANIZATIONS (EXAMPLE)**

- **Keys of Records, 1 per record**: A1, A2, B2, C3, E5, H8, M13, U21
- **Loading Sequence**: 1) E5, 2) B2, 3) M13, 4) A2, 5) C3, 6) U21, 7) A1, 8) H8

- **Pile**

```
E5    B2    M13   A2    C3    U21   A1    H8
```

- **Sequential**

```
A1    A2    B2    C3    E5    H8    M13    U21
```

- **Indexed Sequential**

```
A2    C3    H8    U21
```

**BASIC FILE ORGANIZATIONS (EXAMPLE)**

- **Indexed**

```
A1    A2    B2    C3    E5    H8    M13    U21
```

- **Hashed**

```
E5    B2    H8    M13    U21
```

```
A2    C3    A1
```