Topic: I/O Device Management
1. Storage (disks)
2. Network (Ethernet)
3. Graphics
4. USB

Stuff in the first lecture

I/O Devices – The process has to access data and read data and mapping it into memory space. For example, registers is used by the devices

Interrupt doesn’t fit in memory model. It forces function calls to interrupt handler
A request across the bus to device reads a block from disk. The kernel makes request to hardware. The hardware tells I/O Device when it will done.
A bit more detail

DMA: Device can read and write memory by itself

Graph3
A full description of what is going on in Graph1

Read() in user space

Read() in kernel space: decide what device will be called

Memory buffer: we want the data to go into

0000

DMA Descriptor

1000

Device driver

Command

Length

Pointer

PCI Buses

cmd

DMA

cmd

length

Pointer

read

Graph4
The idea of driver: top half and bottom half

A request coming in and send sth to the hardware

Top half

Bottom half

At some later time, you get interrupt

The top half knows exactly what the top half are talking about, it is running on behalf of the application, the bottom half is very restricted, because it is running as an interrupt

Graph5

Block Devices
Typically it is something you put on top of file system. Use SCSI to access disk.

Read block 1742, length = 10

OK

Data

Or error may occurred if you try to access block at the end of disk

Error

Graph6
Continue with Graph3

Fiber Channel Switch

Hitachi for example

Read disks

EMC for example

Real disks

Graph8
RAID
The idea is to create faster and more reliable system or both by using multiple disk drives instead of a single one.

RAID1: mirroring

If either of the disk fails, the entire disk fails. When an array comes in, we write each data to disks.

first    second
RAID0: Instead of adding reliability, we add speed.

If reading or writing a lot of data, we can keep both disk busy, better performance. It can even have more than two drives.

Graph 10

RAID4: Parity

If calculate parity bit, you can reconstruct the lose bit.

1 0 1 1 0 1 1 Parity: 1
1 0 1 1 _ 1 1 The parity bit let us recover it
1 0 1 1 0 1 1 We can calculate and get it is 0

The parity use one bit to protect all the other ones

The RAID4 can let us get similar reliability with RAID1
RAID 5:

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>P</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>P</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>P</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>
```

Simple Rotation

Graph 11

RAID10:

```
<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
```

Stripping plus mirroring

Graph 12