I/O Systems

CS 256/456

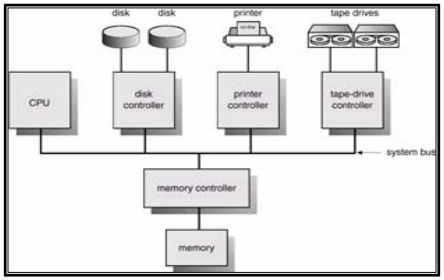
Dept. of Computer Science, University of Rochester

I/O Hardware

Some typical device, network, and bus data rates:

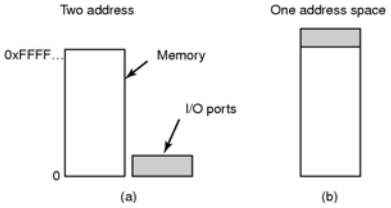
Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Device Controllers



- I/O devices have mechanical component & electronic component
- The electronic component is the device controller
  - It contains control logic, command registers, status registers, and on-board buffer space

I/O Ports & Memory-Mapped I/O



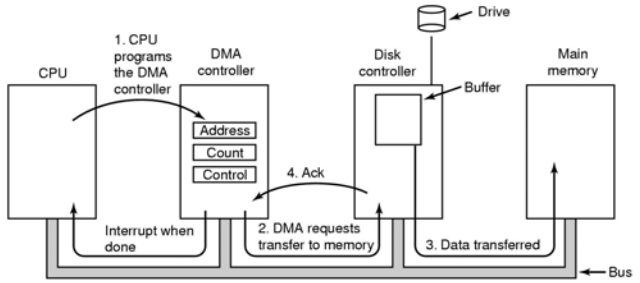
I/O methods:

- Separate I/O and memory space; special I/O commands (IN/OUT)
- Memory-mapped I/O

Issues with them:

- Convenience/efficiency when use high-level language;
- Protection mechanisms;
- Special data access schemes: TSL
- Data caching

Direct Memory Access (DMA)

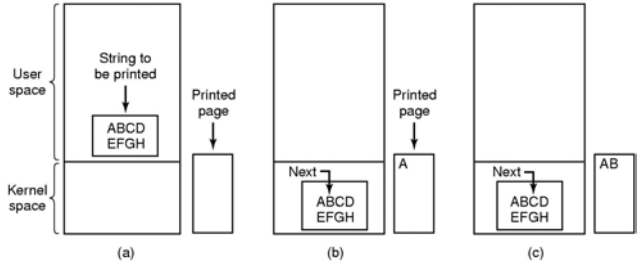


- Are the addresses CPU sends to the DMA controller virtual or physical addresses?
- Can the disk controller directly reads data into the main memory (bypassing the controller buffer)?

How is I/O accomplished?

- Polling-based
  - CPU spins and polls the I/O until it completes
- Interrupt-driven
  - CPU initiates I/O and then does something else; gets notified when the I/O is done (interrupts)

Polling-based I/O




```
copy_from_user(buffer, p, count);
for (i = 0; i < count; i++) {
      while (*printer_status_reg != READY) ;
      *printer_data_register = p[i];
}
return_to_user( );
```

```
/* p is the kernel bufer */
/* loop on every character */
/* loop until ready */
/* output one character */
```


Interrupt-Driven I/O

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count – 1;
     i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

- Writing a string to the printer using interrupt-driven I/O
  - (a) Code executed when print system call is made
  - (b) Interrupt service procedure

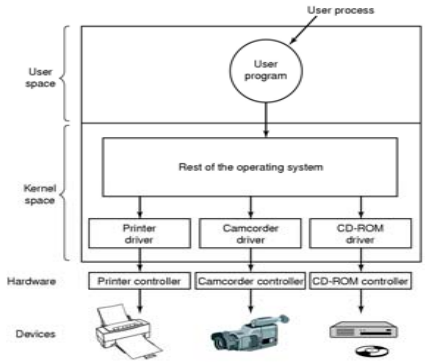
Interrupt Handlers

1. Save registers of the old process
2. Set up context for interrupt service procedure (switch from the user space to kernel space: MMU, stack, …)
3. Run service procedure; when safe, re-enable interrupts
4. Run scheduler to choose the new process to run next
5. Set up context (MMU, registers) for process to run next
6. Start running the new process

How much cost is it? Is it a big deal?

For Gigabit Ethernet, each packet arrives once every 12us.

I/O Software Layers



- Device-dependent OS I/O software; directly interacts with controller hardware
- Interface to upper-layer OS code is standardized.

Device Drivers

- Device driver is the device-specific part of the kernel-space I/O software. It also includes interrupt handlers.
- Device drivers must run in kernel mode. Why?
- ⇒ The crash of a device driver brings down the whole system.
- Device drivers are probably the buggiest part of the OS. Why?

Upper-level I/O Software

- Device independence
  - reuse software as much as possible across different types of devices
- Buffering
  - data coming off a device is stored in intermediate buffers
- Need for buffering
  - speed matching with I/O devices
  - caching
  - speculative I/O

Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).