

11. I/O Systems

- 11.1 Basic Issues in Device Management
- 11.2 A Hierarchical Model
- 11.3 I/O Devices
- 11.4 Device Drivers
 - Memory-Mapped vs Explicit Device Interfaces
 - Programmed I/O with Polling
 - Programmed I/O with Interrupts
 - Direct Memory Access (DMA)
- 11.5 Device Management
 - Buffering
 - Error Handling
 - Disk Scheduling
 - Device Sharing
- 11.6 The Abstract I/O Interface

ICS 143

1

Basic issues

- I/O devices:
 - communication devices
 - input only (keyboard, mouse, joystick)
 - output only (printer, display)
 - input/output (network card)
 - storage devices
 - input/output (disk, tape)
 - input only (CD-ROM)

ICS 143

2

Basic issues

- main tasks of I/O system:
 - present logical (abstract) view of devices
 - hide details of hardware interface
 - hide error handling
 - facilitate efficient use
 - overlap CPU and I/O
 - support sharing of devices
 - protection when device is shared (disk)
 - scheduling when exclusive access needed (printer)

ICS 143

3

A hierarchical model of I/O

Figure 11-1

- abstract I/O interface
 - block devices, character devices, network
- device-independent software
 - buffering, scheduling, caching
- device-dependent software
 - I/O drivers (supplied by device manufacturer)

ICS 143

4

I/O devices

- display monitors
 - character or graphics oriented

Figure 11-2

- different data rates:
 - 25 x 80 characters vs 800 x 600 x 256
 - 30-60 times/sec

ICS 143

5

I/O devices

- keyboards
 - most common: QWERTY
 - very low data rate (<10 char/sec)
- pointing devices
 - mouse (optical, optical-mechanical)
 - trackball
 - joystick
 - low data rate (hundreds of bytes/sec)

ICS 143

6

I/O devices

- printers
 - line printers, dot-matrix, ink-jet, laser
 - low data rates
 - character-oriented
- scanners
 - digitize picture into bit map (similar to video RAM)
 - low data rates

ICS 143

7

I/O devices

- floppy disks
 - surface, tracks/surface, sectors/track, bytes/sector
 - all sectors numbered sequentially 0..(n-1): logical disk

Figure 11-3 (a,b)

ICS 143

8

I/O devices

- floppy disks
 - track skew
 - account for seek to next track to minimize latency
 - double-sided floppy
 - tracks with same diameter: cylinder
 - number all sector within cylinder consecutively to minimize seek

Figure 11-3 (c)

Figure 11-3 (d)

ICS 143

9

I/O devices

- hard disks
 - multiple surfaces

Figure 11-4

- higher densities and data rates than floppy

	floppy	hard disk
bytes/sec	512	512-4096
sec/track	9, 15, 18, 36	100-400
tracks/surf	40, 80, 160	1000-10,000
# surf	1-2	2-24
seek	30-100 ms	5-12 ms
rotation	400-700 rpm	3600-10,000 rpm

ICS 143

10

I/O devices

- optical disks
 - CD-ROM, CD-R (WORM), CD-RW
 - originally designed for music
 - data stored as continuous spiral, subdivided into sectors
 - constant linear speed (200-530 rpm)
 - higher storage capacity than magnetic disks:
0.66 GB/surface

ICS 143

11

I/O devices

- data transfer rates of disks
 - sustained: continuous data delivery
 - peek: transfer once r/w head is in place
 - depends on rotation speed and data density
 - 1 revolution passes over all sectors of 1 track
 - Example: 7200 rpm, 100 sect/track, 512 B/sect
 - 7200 rpm: $60,000/7200=8.3$ ms/rev
 - $8.3/100 = 0.083$ ms/sector
 - 512 bytes transferred in 0.083 ms: ~6MB/s

ICS 143

12

I/O devices

- magnetic tapes (reel or cartridge)
 - large storage capacity (GB)
 - data transfer rate: ~ 2 MB/sec
- networks (interface card)
 - Ethernet, token ring, slotted ring
 - controller implements protocol to accept, transmit, receive packets
 - modem
 - convert between analog and digital (phone lines)
 - character-oriented (like printer and keyboard)

ICS 143

13

Device drivers

- accept command from application
 - get/put char, read/write block, send/rec. packet
- interact with device controller (HW) to carry out command
- typical device controller interface: set of registers

Figure 11-6

- Example: serial or parallel port on PC
 - generic driver reads/writes chars to registers

ICS 143

14

Device drivers

- memory-mapped vs explicit dev interface
 - similar idea to memory-mapped files
- Figure 11-7
- explicit: special I/O instruction:
io_store cpu_reg, dev_no, dev_reg
 - memory-mapped: regular CPU instruction:
store cpu_reg, n
(n is a memory address)

ICS 143

15

Programmed I/O with polling

- CPU is responsible for
 - moving every char to/from controller buffer
 - detecting when I/O operation completed
- protocol to input a character:

Figure 11-8

ICS 143

16

Programmed I/O with polling

- driver operation to input sequence of chars

```
i = 0;
do { write_reg(opcode, read);
      while (busy_flag == true) {...};
      mm_in_area[i] = data_buffer;
      increment i;
      compute;
    } while (...)
```
- what to do while waiting?
 - idle (busy wait)
 - some other computation
 - how frequently to poll? -- Figure 11-9
 - give up CPU
 - device may remain unused for a long time

ICS 143

17

Programmed I/O with interrupts

- CPU is responsible for
 - moving chars to/from controller buffer, but
- interrupt signal informs CPU when I/O operation completes
- protocol to input a character:

Figure 11-10

ICS 143

18

Programmed I/O with interrupts

- compare polling with interrupts:

```
i = 0;
do { write_reg(opcode, read);
      while (busy_flag == true) {...};
      mm_in_area[i] = data_buffer;
      increment i;
      compute;
    } while (...)
```

```
i = 0;
do { write_reg(opcode, read);
      block to wait for interrupt;
      mm_in_area[i] = data_buffer;
      increment i;
      compute;
    } while (...)
```

ICS 143

19

Programmed I/O with interrupts

- Example: keyboard driver

```
i = 0;
do { block to wait for interrupt;
      mm_in_area[i] = data_buffer;
      increment i;
      compute(mm_in_area[]);
    } while (data_buffer != ENTER)
```

- timing of interrupt-driven I/O

Figure 11-11

– more OS overhead but better device utilization

ICS 143

20

DMA

- CPU only initiates operation
- DMA controller transfers data directly to/from main memory
- interrupt when transfer completed
- protocol to input data using DMA:

Figure 11-12

ICS 143

21

DMA

- driver operation to input sequence of chars

```
write_reg(mm_buf, m);
write_reg(count, n);
write_reg(opcode, read);
block to wait for interrupt;
```

 - writing opcode triggers DMA controller
 - DMA controller issues interrupt after n chars in memory
- I/O processor (channel)
 - extended DMA controller
 - executes I/O program in own memory

ICS 143

22

Device management

- device-independent techniques
- reasons for buffering
 - allows asynchronous operation of producers and consumers
 - allows different granularities of data
 - consumer or producer can be swapped out while waiting for buffer fill/empty

ICS 143

23

Device management

- single buffer operation
Figure 11-14 (a)
- double buffer (buffer swapping)
Figure 11-14 (b)
 - increases overlap
 - ideal when:
 - time to fill = time to empty = constant
 - when times differ, benefits diminish

ICS 143

24

Device management

- circular buffer
 - when average times to fill and empty are comparable but vary over time: circular buffer absorbs bursts
 - producer and consumer each use an index
 - nextin gives position of next input
 - nextout gives position of next output
 - both are incremented modulo n at end of operation

ICS 143

25

Device management

- error handling
 - persistent vs transient; SW vs HW
 - persistent SW error
 - repair/reinstall program
 - other errors: build in defense mechanisms
 - Examples:
 - transient SW errors: error correcting codes, retransmission
 - transient HW errors: retry disk seek/read/write
 - persistent HW errors: redundancy in storage media

ICS 143

26

Device management

- bad block detection and handling
 - block may be defective as a manufacturing fault or during use (a common problem)
 - parity bit is used to detect faulty block
 - the controller bypasses faulty block by renumbering
 - a spare block is used instead
 - two possible remappings:
 - Figure 11-17
 - more work but contiguity of allocation preserved

ICS 143

27

Device management

- stable storage
 - some applications cannot tolerate any loss of data (even temporarily)
 - stable storage protocols:
 - use 2 independent disks, A and B
 - write: write to A; if successful, write to B
 - read: read from A and B; if A!=B, go to recovery
 - recovery from media failure: A or B contains correct data; remap failed disk block
 - recovery from crash: if before writing A, B is correct; if after writing A, A is correct; recover

ICS 143

28

Device management

- RAID (redundant array of independent disks)
 - increased performance through parallel access
 - increased reliability through redundant data
 - maintain exact replicas of all disks
 - Figure 11-19(a)
 - most reliable but wasteful
 - maintain only partial recovery information (e.g. error correcting codes)
 - Figure 11-19(b)

ICS 143

29

Device management

- disk scheduling
 - minimize seek time and rotational delay
 - requests from different processes arrive concurrently:
 - scheduler must attempt to preserve locality
 - rotational delay:
 - order requests to blocks on each track in the direction of rotation: access in one rotation
 - proceed with next track on same cylinder

ICS 143

30

Device management

- minimizing seek time: more difficult
 - r/w arm can move in two directions
 - minimize total travel distance
 - guarantee fairness
 - FIFO: simple, fair, but inefficient
Figure 11-20 (a)
 - SSTF: most efficient but prone to starvation
Figure 11-20 (b)
 - Scan: fair, acceptable performance
Figure 11-20 (c)

ICS 143

31
