CSE 120: Principles of Operating Systems

Lecture 11

Input/Output

February 27, 2006

Prof. Joe Pasquale
Department of Computer Science and Engineering
University of California, San Diego

© 2006 by Joseph Pasquale
Before We Begin ...

Read Chapter 13 (on I/O Systems)
What is I/O

Input/output between CPU/memory and I/O devices

Problems

• So many different types of I/O devices
• Wide range: speed, operation, data transfer units

Questions

• How does a process initiate I/O?
• How is synchronization achieved?
• How is data transferred?
Background on I/O Hardware

CPU and device (controller) communicate via
- I/O instructions
- Memory instructions (memory-mapped)

Data transfer: programmed I/O vs. DMA

Synchronization: polling vs. interrupts
Process-Device I/O

Process A

Process B

Device X

Device Y

Kernel

Unbuffered I/O

Buffered I/O
Classifying Devices

So many different types of devices

• Classify by shared characteristics
• Imposes structure: shared code, lower complexity

Dimensions

• Variable vs. fixed size units
• Sequential vs. random-access
• Synchronous vs. asynchronous
• Speed of operation
I/O Layered Software Structure

User

User I/O (stdio library)

Kernel

Device-Independent I/O
(buffering, caching, block vs. char, …)

Device driver

Device driver

Device driver

Hardware

Device controller

Device controller

Device controller

Device controller

Device

Dev

Dev

Device

Dev

Dev
Device Drivers

Encapsulates device-dependent code
  • contains device-specific register reads/writes

Generally must implement a standard interface
  • Example: open, close, read, write

Interrupt handlers
  • Executes when I/O completes
  • Updates data structures
  • Wakes up waiting process
  • May schedule work later: “software interrupt”
Device-Independent I/O Software

Uniform interfacing for device drivers

Naming, protection

Buffering, caching

Uniform block size

Storage allocation (block)

Locking (dedicated devices)

Error handling

read(), write()

d1read(), d1write() d2read(), d2write()
User-Space I/O Software

Richer or simplified interface
- putchar/getchar vs. read/write

User-level buffering
- Unix: stdio library

Spooling daemons
- Printer

Device drivers

Device-Independent I/O

User I/O Library

User program

getchar(), putchar(), fread(), fwrite()

buffering

read(), write()
Overall Operation

I/O request

User Process

Device-Independent I/O

Device Driver Upper (synch)

Interrupt Handlers (asynch)

Hardware

I/O response

Make I/O call; format I/O; spooling

Naming, protection, blocking, buffering, allocation

Control via device registers; check status

Wakeup driver when I/O completed

Perform I/O operation
Example: Unix I/O

- File Subsystem
- Device Drivers
- Character
- Block
- Buffer Cache
- Device Drivers
Unix: I/O System Calls

Via file system interface

- \texttt{fd = open (“/dev/devname”, ...)}
- \texttt{close (fd)}
- \texttt{nr = read (fd, buf, n)}
- \texttt{nw = write (fd, buf, n)}
- \texttt{iocltl (fd, cmd, buf)}
Unix: Buffered vs. Unbuffered I/O

Buffered I/O
- System buffers
- Buffer cache
- Character queues

Unbuffered I/O
- Direct transfer between process and device
- Can be via DMA
Unix: Standard I/O Library

fopen, fread, fwrite, fprintf, fscanf, fclose

Private buffer kept in user space

Minimizes the number of I/O system calls
Unix: Block vs. Character Devices

Block
- transfer units in fixed-size blocks
- blocks are addressable (random access)
- kept in buffer cache

Character
- transfer units in variable-size sequence of bytes
- used for all non-block devices
- linked list of character queues