CSE 120: Principles of Operating Systems

Lecture 11

I/O System Software

November 18, 2003

Prof. Joe Pasquale

Department of Computer Science and Engineering

University of California, San Diego

© 2003 by Joseph Pasquale
Before We Begin ...

Read Chapter 11 (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 in 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

I/O bus interconnects CPU, memory, I/O devs

A device generally connects to the bus via a controller

Controller has registers that CPU can read/write via
  • I/O instructions
  • memory instructions (memory-mapped)

Data transfer: programmed I/O vs. DMA

Synchronization: polling vs. interrupts
Classifying Devices

So many different types of devices

• classify by shared characteristics
• imposes structure: shared code, lower complexity

Dimensions

• character stream vs. block
• sequential vs. random-access
• synchronous vs. asynchronous
• sharable vs. dedicated
• 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

Generally must implement a standard interface

Code contains device-specific register reads/writes

Interrupt handlers

- Handler executes when I/O operation completes
- May be converted to a “software interrupt”
- Updates data structures
- Wakes up process that was waiting
Device-Independent I/O Software

Uniform interfacing for device drivers

Naming, protection

Buffering, caching

Device-independent block size

Storage allocation for block devices

Locking (dedicated devices)

Error handling
User-Space I/O Software

Richer or simplified interface
  • e.g., putchar/getchar vs. read/write

User-level buffering
  • e.g. stdio library

Spooling daemons
  • Printer
Overall Operation

- **User Process**
  - Device-Independent I/O
  - Device Driver: Upper (synch)
  - Interrupt Handlers (asynch)
  - Hardware

  - I/O request
  - I/O response

- **Device**
  - Perform I/O operation
  - Wakeup driver when I/O completed
  - Control via device registers; check status
  - Make I/O call; format I./O; spooling
  - Naming, protection, blocking, buffering, allocation
Example: Unix I/O

File Subsystem

Buffer Cache

Character  Block

Device Drivers
Unix: I/O System Calls

Via file system interface

• open
• close
• read
• write
• IOCTL
Unix: Buffered vs. Unbuffered I/O

Buffered I/O

• system buffers: buffer cache, character queues

Unbuffered I/O

• data transferred directly between proc and dev
• 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
Unix: The Buffer Cache

Cache of recently used blocks

Each block has an address (device #, block #)

Three lists

- device list (hash table + linked lists)
- free list
- driver I/O queue

LRU is used for replacement