Victor Wu

Group 4 – CS350

Input/Output - OS integrated with Programming Languages
Overview

• Hardware
• Application
• Kernel

• How programming languages work with all these
Hardware – quick review

• Devices connected through ports
• Bus
  • PCI bus, expansion bus, SCSI bus, daisy-chain bus
• Registers
  • Data-in, data-out, status, control
• Memory-mapped VS. Direct Memory Access
• Polling (Programmed I/O)
• Interrupts
  • Non-blocking vs blocking
Application

• block I/O, character I/O (stream)
  • Operations like read(), write(), and seek() on UNIX
  • Could be layered with memory-mapping

• network sockets
  • Socket “#include <sys/socket.h>” Linux networking socket layer UI
  • “select()” system call

• Special devices
  • time-of-day clock and system timer

• UNIX, ioctl() system call is an escape, or backdoor
  • Applications sending commands directly to device drivers
Kernel

- Scheduling
- Buffering
- Caching
- Spooling
  - “AddJob()” Windows Print Spooler API
- Error handling
  - UNIX sets “errno”
- I/O Protection
Higher abstraction from User-perspective

- System API -> System call
  - File system API

- Transform I/O request to Hardware
  - UNIX uses *mount table* and *device files*
  - *Linux uses an elevator scheduler*
  - *Windows has an I/O manager*

- Many stuff happening behind scene
  - Front-end processors enhancing performance
  - Application -> Kernel -> Device-driver -> Device-controller -> Device
    - Users provide file name, it gets mapped
Conclusion

• Questions?
Sources

