... airplane fuel injection system of a car.

Non-traditional devices: joystick, robot actuators, flying surfaces of an airplane, fuel injection system of a car.

– The device itself
  - The device itself reads/writes data onto the system bus.
  - Controller: receives commands from the system bus, translates them into device actions. Reads/writes data from the device, sends data back to the controller.

– System bus: allows the device to communicate with the CPU, typically shared by multiple devices.

– Key components
  - System bus: allows the device to communicate with the CPU.
  - Controller: receives commands from the system bus, translates them into device actions.
What happens if the device is slow compared to the CPU?

- Good choice if data must be handled properly, like for a modem or keyboard.
- CPU observes the change to idle and reads the data if it was an interrupt operation.
- Data-in is an input command.
- Controller reads the command register and performs the command, placing a value in
  command-reads to command-ready = controller sets status to busy
- CPU sets the command register and de-selects it if an output operation.
- CPU busy-waits until the status is idle.

### Communication using Polling

### I/O Services Provided by OS

- Device drivers to implement device-specific behaviors.
  - Example: error handling and failure recovery associated with devices (command return, for
  - I/O scheduling.
  - Buffering, caching, and spooling to allow efficient communication with devices.
  - Device allocation.
  - Operations appropriate to the files and devices.
  - Access control.
  - Naming of files and devices. (On Unix, devices appear as files in the /dev directory).
Transfer Issue:
CPU somewhat but still providing better performance than if the CPU had to do the

- DMA controller and the CPU compete for the memory bus, slowing down the
  transfer is complete, instead of when each byte is ready.
- The DMA controller operates the bus and interrupts the CPU when the entire
  destination of the DMA transfer is complete.
- The DMA controller copies the DMA transfer is complete.
- Use a sophisticated DMA controller that can write directly to memory.

\[ \text{Solution: Direct memory access (DMA)} \]

For devices that transfer large volumes of data at a time (like a disk

Direct Memory Access

Communication Using Interrupts

On an I/O interrupt:

- Start the next operation for that device.
- Stop the last command in an input operation, retrieve the data from the device register.
- Determine which device caused the interrupt.
- Complete an I/O operation.
- Rather than using busy waiting, the device can interrupt the CPU when it

Department of Computer Science, Virginia Tech
The DMA controller interrupts the CPU when the transfer is done.

- Physical memory
- It is transferred over the bus by the DMA controller into a buffer in
- A disk buffer stores a block when it is read from the disk.

I/O buffering

I/O devices typically contain a small on-board memory where they can store

- Examples: keyboard (sequential, character), disk (block, random or sequential)
- Operations: input, output, or both
- Shareable or dedicated
- O/I Implement blocking
- Most devices are asynchronous, while I/O system calls are synchronous:
  - Timing: synchronous or asynchronous
  - Access method: sequential or random access
  - Transfer unit: character or block

Device characteristics:

- New devices can be supported by providing a new device driver.
- Devices dependencies are encapsulated in device drivers.
- Standard interfaces are provided for related devices.

I/O OS provides a high-level interface to devices, greatly simplifying the

Application Programmer's View of I/O Devices
What should happen when we write to a cache?

- Else allocate space in memory, read block from disk, and update value in memory
  - Example: Write (disk address)

  - Else read (disk address)

  - Else read (in memory) return value from memory

  - Example: Read (disk address)

If block in memory then move to memory

- Else: Keep recently used disk blocks in main memory after the I/O call that brought

  - Idea: Keep recently used disk blocks in main memory after the number of disk accesses.

  - Improve disk performance by reducing the number of disk accesses.

Caching

Why buffer on the OS side?

- To copy kernel data to a kernel buffer and return control to the user program. The

  - Writes = Copy data to a kernel buffer and return control to the user program. The

  - To minimize the time a user process is blocked on a write.

To happen one block at a time.

- Example: How brings the file over the network one packet at a time. Stores to disk

To cope with devices that have different data transfer sizes.

- Example: Compute the contents of a display in a buffer (slow) and then zap the

To cope with speed mismatches between device and CPU.

Buffer to the screen (fast)

- Example: Compute the contents of a display in a buffer (slow) and then zap the
Utilization:
- Increase CPU utilization
- Increase number of devices to reduce contention for a single device and thereby
  O/S load computation from the main CPU by using DMA controllers.
- Reduce interrupt frequency by using large data transfers.
- Reduce data copying by caching in memory.

Approaches to improving performance:
- 1/O is瓶颈 supported via system calls and interrupt handling, which are slow.
- 1/O is expensive for several reasons:

Summary

call.

4. When the process gets the CPU, it begins execution following the system
ready queue.

3. O/S transfers the data to the user process and places the process in the

(a) DMA controller interrupts the CPU when the transfer is complete.
(b) Device driver tells the DMA controller what to do and blocks until
(c) DMA controller transfers the data to the kernel buffer when it has all been retrieved
(d) O/S tells the device driver to perform input.

2. O/S checks if data is in a buffer. If not,

1. User process requests a read from a device.

Putting the Pieces Together - A Typical Read Call