Operating Systems

Tutorial 2 & 16

Michael Tänzer

os-tut@nhng.de
http://os-tut.nhng.de

Calendar Week 6

Outline

True or False

- When reading one track it's possible to read all other tracks on the same cylinder simultaneously
- SSTF is always optimal
- Fixed size swap files usually don't get fragmented

True or False

- When reading one track it's possible to read all other tracks on the same cylinder simultaneously
- SSTF is always optimal
- Fixed size swap files usually don't get fragmented

True or False

- When reading one track it's possible to read all other tracks on the same cylinder simultaneously
- SSTF is always optimal
- Fixed size swap files usually don't get fragmented

True or False

- When reading one track it's possible to read all other tracks on the same cylinder simultaneously
- SSTF is always optimal
- Fixed size swap files usually don't get fragmented

Why categorise a disk as a block device?

What are the consequences of permitting byte-level access to a disk drive?

Why categorise a disk as a block device?

What are the consequences of permitting byte-level access to a disk drive?

- High throughput but also high latency $\Rightarrow$ bytewise access would be inefficient as the costs to position the head would outweigh the high throughput while transferring the data
- Also the cost of establishing a DMA transfer and dealing with the end-of-DMA interrupt should be amortised over a large data transfer
- A disk provides random access but with a character device you can only request currently present data

Why categorise a serial port as a character device?

Would it be possible to treat a serial port as a block device?

Why categorise a serial port as a character device?

Would it be possible to treat a serial port as a block device?

- Data can only be accessed sequentially, data arrives/is sent one byte at a time
- It's not possible to seek in the data stream (the driver has no influence on the order in which the data arrives)
- No block structure on the device $\Rightarrow$ block access would require to send/receive data in fix lengths but it's not guaranteed that the device delivers that much data

What is the responsibility of the CPU for device-to-memory data transfers in each of the three primary I/O models?

What is the responsibility of the CPU for device-to-memory data transfers in each of the three primary I/O models?

Programmed I/O (Polling)

Coordinate the entire data transfer

1. Issue an I/O command to the device
2. Poll for a response
3. Fetch the data directly from the device registers

What is the responsibility of the CPU for device-to-memory data transfers in each of the three primary I/O models?

Interrupt-Driven I/O

1. Initiate the I/O transaction
2. Do something else while waiting for the interrupt from the device
3. When the interrupt occurs fetch the data directly from the device registers

What is the responsibility of the CPU for device-to-memory data transfers in each of the three primary I/O models?

DMA

1. Initiate the transaction and provide a location in physical memory to the DMA controller (may be integrated in the device)
2. Do something else while waiting for the interrupt from the DMA controller
3. When the interrupt occurs all data is already in memory for further processing

Calculate the CPU overhead with polling for the following devices

Specs

- CPU frequency: 400 MHz
- One polling operation costs 400 cycles
- Mouse: 3000 Hz
- Floppy disk: 16 bit per poll at 50 kB/s
- Hard disk: 32 bit per poll at 8 MB/s

Write pseudo-code which expresses the functionality of the DMA engine for device-to-memory data transfer

Write pseudo-code which expresses the functionality of the DMA engine for device-to-memory data transfer

```
baseAddr = getMemoryDestination();
for count = 0 to blockSize - 1 do
    data = deviceRead(count);
    memoryWrite(baseAddr + count, data);
od
```

Why is it preferable to use a dedicated I/O bus, thus separating the devices from the CPU and memory bus?

Why is it preferable to use a dedicated I/O bus, thus separating the devices from the CPU and memory bus?

- The DMA engine has to access the bus twice for each loop iteration
- When the DMA controller accesses the bus the CPU has to wait for it to become available again to fetch data or instructions from memory
- When separating the I/O bus from the memory bus the DMA controller only needs one access to the memory bus per iteration

$\Rightarrow$ Less traffic on the memory bus, less wait cycles for the CPU

What is fly-by transfer?

Also called single-bus transfer

What is fly-by transfer?

Also called single-bus transfer

- Without fly-by transfer the data is sent two times over the bus
- Device, memory and DMA controller are all attached to the same bus

$\Rightarrow$ DMA controller 'tells' memory to listen when the data is sent from the device $\Rightarrow$ only one transfer over the bus

Questions & Answers

<!-- Your questions here -->

The End

Good luck for your exams