Operating System (H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Assignment 6

Mengxuan Wu

Question 1

When a process accesses a memory page not present in the physical memory, a page fault is raised, and the page fault handler is called. The operating system swaps the page from the disk to the physical memory and updates the page table. It also adds the page to TLB. The page fault handler then restarts the instruction that caused the page fault. The process continues to run.

Question 2

1.

Polling does not need to perform context switching, but it is inefficient because it wastes CPU time since the CPU is continuously checking the status of the device.
In contrast, interrupt-driven I/O is more efficient because the CPU can perform other tasks while waiting for the device to finish the I/O operation. The CPU is only interrupted when the device is ready. However, interrupt-driven I/O requires context switching, which is time-consuming. Also, if a huge number of interrupts are generated, the CPU may spend more time handling interrupts than performing useful work, which is called **livelock**.

2.

PIO transfers data between the device and the main memory one byte at a time. It is inefficient because it requires the CPU to be involved in every data transfer. The CPU must read the data from the device and write it to the main memory. This process is slow and wastes CPU time.
DMA transfers data between the device and the main memory without CPU intervention. The CPU initiates the data transfer and can perform other tasks while the data is being transferred. This allows overlapping of I/O operations with CPU operations, which improves system performance. However, DMA requires additional hardware.

3.

For memory-mapped I/O, we modify the MMU so that only the kernel can access the I/O address space, thus protecting the I/O devices from user processes. For explicit I/O, we make them privileged instructions that can only be executed in kernel mode, protecting the I/O devices from user processes.

Question 3

1.

READ / WRITE data time = seek time + rotational latency + transfer time

2.

Since the sector is random, we calculate the rotational latency as the average rotational latency, which is half of the time for one revolution

$$t_r = 0.5 \text{ Round}/12000 \text{ RPM} = 2.5\text{ms}$$

Since 6 sectors are read, the total rotational latency is

$$T_r = 6 \times 2.5\text{ms} = 15\text{ms}$$

FIFO

Access sequence: 70, 20, 90, 110, 60, 20

$$T_s = [(100-70)+(70-20)+(90-20)+(110-90)+(110-60)+(60-20)] * 1\text{ms} = 260\text{ms}$$

$$T_{\text{total}} = T_s + T_r = 260\text{ms} + 15\text{ms} = 275\text{ms}$$

SSTF

Access sequence: 110, 90, 70, 60, 20, 20

$$T_s = [(110-100)+(110-90)+(90-70)+(70-60)+(60-20)+(20-20)] * 1\text{ms} = 100\text{ms}$$

$$T_{\text{total}} = T_s + T_r = 100\text{ms} + 15\text{ms} = 115\text{ms}$$

SCAN

Access sequence: 110, 90, 70, 60, 20, 20

$$T_s = [(200-100)+(200-20)] * 1\text{ms} = 280\text{ms}$$

$$T_{\text{total}} = T_s + T_r = 280\text{ms} + 15\text{ms} = 295\text{ms}$$

CSCAN

Access sequence: 110, 20, 20, 60, 70, 90

$$T_s = [(200-100)+200+90] * 1\text{ms} = 390\text{ms}$$

$$T_{\text{total}} = T_s + T_r = 390\text{ms} + 15\text{ms} = 405\text{ms}$$