Lecture 10

I/O Device Management

- How are I/O devices connected to a computer?
- How does the system accommodate devices of different characteristics, namely form and speed?
- What techniques are used for the management of I/O devices?

I/O subsystem

- A peripheral device provides data, connects to the network or is the one we use to communicate with computer.
- It is accessed by a (hardware) device controller that can be very simple, allowing it to execute basic operations, or complex ones that commands sequences of operations; sometimes, a controller is called *channel* (IBM), or *IO Processors* (IOP).
- The trend is to develop standardized controller/device interfaces, such as Universal Serial Bus (USB), or the Small Computer System Interface (SCSI).
- The device driver is the software that runs the controller. A part of it is the interrupt handler that manages events generated by the controller.
- Drivers are extensions of the kernel; they run with kernel privileges.

I/O device hardware features

- Heterogeneous in terms of functionality and performance, especially speed.
- Different in terms of reliability and the level of detail of the control.
- Functionally:
  - Interrupt/poll in terms of service they receive;
  - DMA to free CPU;
  - Controller registers memory-mapped or addressed within the I/O space;
  - Operations can be blocking or non-blocking.

I/O drivers

- An I/O driver belongs to a *family of drivers* that abstracts in the kernel all devices of a particular type.
- *Examples*: bus protocols (SCSI Parallel, USB), storage (disk) devices, cards for network services (ethernet),...
- A driver becomes a member of a family through inheritance; as a member of a family, the driver inherits the data structures and the behaviours that are common to all members of the family. For example, all SCSI controllers scan the SCSI bus as the *SCSI Parallel family* defines and implements this scanning functionality.
- The bulk of a driver's interaction with its own family involves implementing member functions that the family invokes. These are typically client configuration and I/O requests.

Driver's interactions

- Typically, the driver communicates with the *family* to which it belongs and the object that represents the *access point* and *communication channel* to the bus/protocol to which the device is connected.
- *Example*: a PCI Ethernet driver uses an IOPCIDevice object from the PCI family to attach to and communicate over the PCI bus.
- The I/O layered architecture models the chain of connections between the system's hardware buses and devices, gathering common functionality into classes the driver can interact with. Each layer is a client of the layer below it and a provider of services to the layer above it.

Scenario: new PCI device plugged in

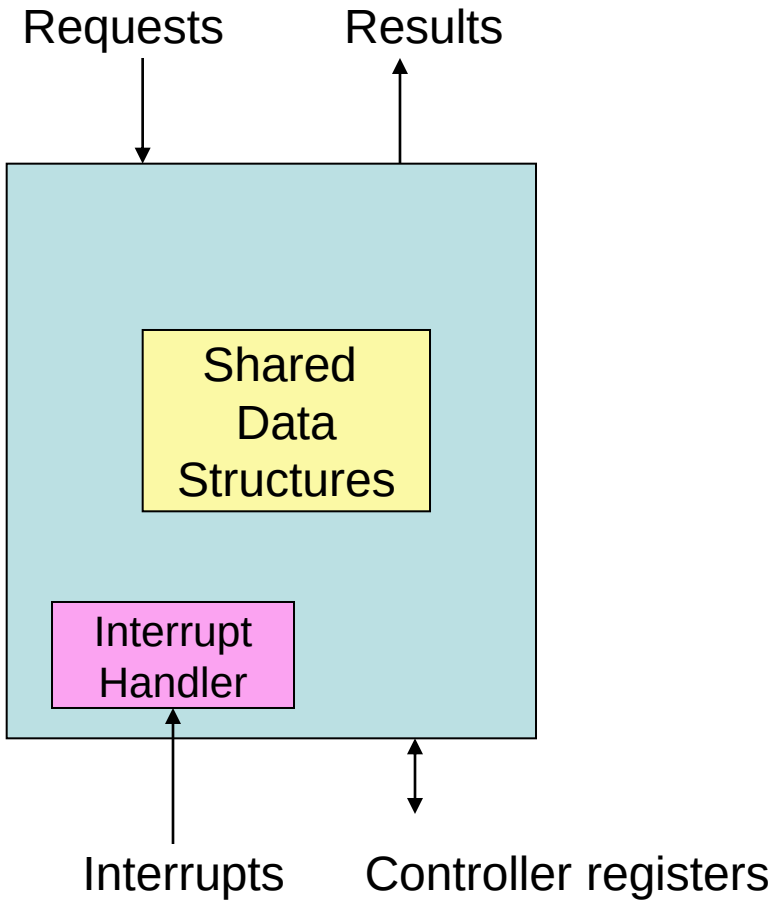
1. The PCI bus controller driver (member of the PCI family) discovers a PCI device and announces its presence by creating an object (IOPCIDevice).
2. The object identifies (matches) an appropriate device driver—in this case, a SCSI controller driver—and requests that it be loaded. Loading the SCSI controller driver causes the SCSI Parallel family, and all families that it depends on, to be loaded as well. The SCSI controller driver is given a reference to the IOPCIDevice object.
3. The SCSI controller driver, which is a client of the PCI family and a provider of SCSI Parallel family services, scans the SCSI bus for devices that might be clients of these services. Upon finding such a device (a disk), the driver announces the device's presence by creating an object (IOSCSIDevice).
4. The object, using the matching procedure, finds a device (disk) driver that is appropriate for the device and requests that this driver be loaded. Loading the disk driver causes the Storage family, and all families that it depends on, to be loaded as well. The disk driver is now a client of the SCSI Parallel family and a member of the Storage family. The disk driver is given a reference to the IOSCSIDevice object.

The device driver work

Example:

- Process A starts a disk read operation;
- The device driver translates the params into controller params and writes them in regs
- The driver initiates the read.
- Process A is replaced by process B;
- B requests a disk write on the same disk;
- The driver queues B's request
- The controller finished the read and issues an interrupt;

……



Device driver structure

- There are two cooperating control flows that share a data structure:
  - one flow of control that is executed on behalf of a user process requesting an I/O service;
  - a second flow of control executed as a response to interrupts.
  - Because the first flow can be interrupted, the two flows must coordinate their access to the request queue they share.
  - Similarly, they should not try to command the controller in the same time.

- The driver can be divided in two halves:
  - The upper deals with user requests;
  - The lower deals with the controller.
  - The two threads of control use mutual exclusion techniques in order to prevent corruption of the shared queue.

Device management techniques

- Buffers as interfaces between the producers and consumers of data when their rates are different.
- Water marks are used with buffers:
  - the mark close to the full end (high water mark) signals when the buffer is about to become full (a message is sent to the producer);
  - The mark close to the empty end (low water mark) signals when the buffer is about to become empty.

  The exact values of the two marks are not critical. However, if the high water mark is too high, the producer might send enough items before processing the stop message.

- In the case of device drivers, buffers store requests and data.
- Interleaving the sectors on a track in order to give the interrupt handler time to process the next request before the data is under the head.
- RAID (redundant array of inexpensive disks) is a technique for using multiple disks as if they were a single disk. To the OS, the whole RAID array looks like a single disk – the concept is implemented by the controller.

Conclusions

- Devices drivers are the software interface to computer peripheral devices.
- Device drivers are loaded as kernel extensions when required (new devices are connected plug & play).
- There are concurrent control flows in the driver.
- Driver families provide a level of abstraction that accommodates all devices of the same type.