1B11 Operating Systems

Input/Output and Devices

Prof. Steve R Wilbur
s.wilbur@cs.ucl.ac.uk

Lecture Objectives

- How do the bits of the I/O story fit together?
- What is a device driver?

Devices

- There are lots of different devices that can be attached to a computer:
  - Storage magnetic, optical disk, tape, etc.
  - Display, Keyboard, Pointer (mouse,tracker)
  - Network Interface Card
  - Audio, Video
  - Robot arms, toaster, coffee machine, 350bhp Chrysler V8 electronic ignition engine management system
- Good to have a unified interface in OS

Device Drivers

- Software in OS to manage I/O to a device is called a ***device driver***
- Devices can ***source*** or ***sink*** data
- Devices may need to be ***controlled*** in specific ways, e.g. ***rewind***
- So an OO design for a ***generic device driver*** might include two classes, each with a set of methods:
  - open(), read(), write(), close()
  - control(), handle_event()
- Unix (Unified Interface to everything) has similar interface to devices … files … and networks!

Device Drivers - 2

- A device driver abstracts specific device hardware into a ***generic*** model of I/O device
- Result is ***abstract models*** of disks, tapes, screens etc.
- Makes it easy to port OS and applications to new hardware - ***device independence***
- Internally, device driver is programmed with details of ***specific device***, but presents relevant ***abstraction*** at its interface

Device Drivers - 3

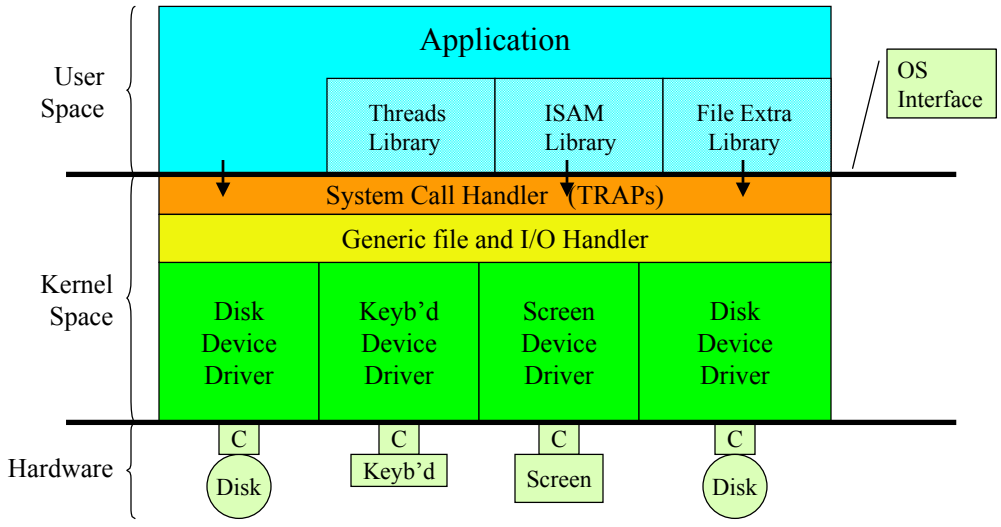
- However, we can't be completely generic
- Devices can be classed broadly into a small number of categories:
  - character (byte) devices
    - modem, keyboard, maybe mouse
  - block (bulk) devices
    - disk, net, screen
- Categories are an ***engineering trade-off*** to keep reasonable efficiency and functionality

Internal to Device Driver

- Driver needs to cope with device characteristics:
  - polled, DMA and interrupts
  - memory mapped
  - device controller commands etc
- So typical device driver includes:
  - initialise device (at OS boot time)
  - start output
  - start input
  - control device (seek or rewind)
  - shutdown device (at OS shutdown time)
  - interrupt service routine for input (and output and control)

1B11-5 1999 Slide 7

The Big Picture



1B11-5 1999 Slide 8

Example - Write to Disk

- Simplified version!
- User makes ***system call*** to write specified block of data to disk
- Processor switches to ***kernel mode***
- ***System Call Handler*** invoked (TRAP) and syscall parameters checked
- ***Generic file and I/O Handler*** called (TRAP function indicates I/O)
- User ***data copied to OS buffer*** or locked to prevent it being swapped out while I/O pending (or overwritten by user if asynchronous I/O)
- Device driver ***start()*** method called - buffer passed as argument
- Device driver ***adds buffer to queue*** of pending I/O requests
  - may order queue e.g. to minimise disk head movement
  - will get processed in due course
- Return to generic I/O handler which ***suspends user process*** and resumes next ready process

Example - Write to Disk - 2

- "Other side" of device driver
- When a disk ***transfer complete interrupt*** occurs (from earlier transfer) interrupt service routine (ISR) removes indicates completion details in buffer and removes blocked indicator from that Ps PCB for the ***scheduler*** to use
- Look on pending request Q for new work, set up DMA registers for transfer and start transfer
- Return from interrupt

- Notes:
  - if no new work the ISR just returns
  - if Q is empty when start() is called the device driver will start the transfer immediately

Interrupt Service Routines

- These are installed by the device initialisation code (boot time)
  - by adding them to the jump table in the generic interrupt handler for the device's interrupt level
- They are called by the device controller (C in diagram) sending an interrupt signal to the processor
- ISR then saves a small part of the current context (OS or User) and jumps to the code for the handler…..
- Which saves any extra state, and deals with it….

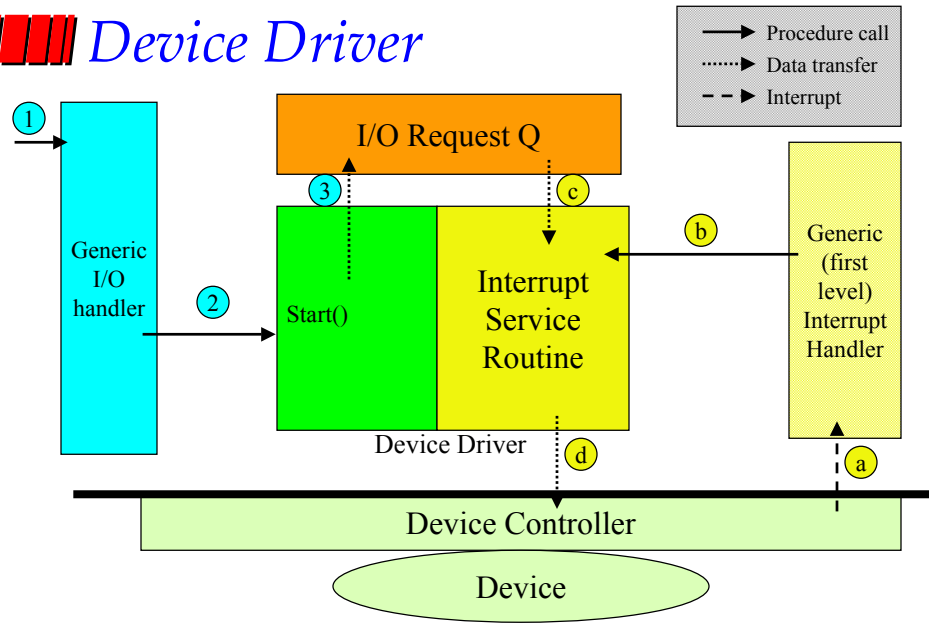
1B11-5 1999 Slide 11

Buffers

- Block mode DMA devices need buffers (for queueing data between the device and the user)
- The operating system keeps and manages a pool of these blocks of memory - typically of several different sizes, for different cases

1B11-5 1999 Slide 12

Device Driver



1B11-5 1999 Slide 13

Notes

- ***File abstraction*** dealt with by generic code (handler)
- ***Device abstraction*** dealt with by device driver - maps to real device
- Q is used to manage ***producer-consumer*** relationship between two halves of device driver

1B11-5 1999 Slide 14

Network Devices

- ***Abstract network device*** is provided in similar way
- Generic network functions call network device drivers
- ∴ Same/similar interface whatever type of links and devices actually used to physically connect computers - another abstraction!!!

Graphic Displays

- ***Bitmap displays*** are usually implemented slightly differently
  - display typically refreshed from a block of dual-ported cheap memory
  - written by the processor, and read by the CRT controller
- So windows systems simply write bitmaps to this area - has the effect of updating the screen (1 scan time later)
- (***Character mode displays*** are driven through serial interface - now virtually obsolete)

Pointers/trackers

- Pointing devices
  - interrupt with the current x/y position of the pointer
  - or delta-x, delta-y since last interrupt
  - i.e. they provide a stream of events
- These may be conveyed to user process via ***software interrupts*** to give the programmer the illusion of ***event streams***
- Fits thread model ***very well***

Summary

- Device abstraction is common in OS to group devices by broad class, and provide a standard API
- Devices are accessed via a table - another piece of book-keeping that the OS must do
- For some groups of devices, further ***layers of abstraction*** are also provided - including:
  - file systems, networks, displays, etc