Embedded Operating System

Agenda

- Hardware abstraction
- I / O Management
- Computer structure
- Computer IO (Input Output)
- Interrupt Processing
- Maskable vs Non-maskable interrupt
- Interrupt Controller
- CPU Scheduling

Hardware abstraction

- OS hides hardware details/intricasies from end users/user programs.
- Same program can work on different hardwares (e.g. different disks, different monitors, ...).
- Computer hardware may have IO mapped IO or Memory mapped IO.
- Typically lowest layer in OS is hardware abstraction layer.
- OS also have various device drivers -- which handles working/execution of different IO devices.

Computer structure

- CPU: Genral purpose processor for program/OS execution
- Memory: RAM
- Storage: Disk
- IO: Keyboard, Monitor
- All these Peripheral Connected by "bus".
- Each IO device has a "dedicated" "internal" processing unit -- IO device controller

Computer IO (Input Output)

- Synchronous IO: CPU is waiting for IO to complete.
  - Hw technique: Polling

- Asynchronous IO: CPU is not waiting for IO to complete (doing some other task).
  - Hw technique: Interrupts
  - OS maintains a device status table to keep track of IO devices (busy/idle) and processes waiting for those IO devices.

Interrupt Processing

- IO event is sensed by IO device controllers.
- It will be conveyed to CPU as a special signal - Interrupt.
- CPU pause current execution and execute interrupt handler.
- "Interrupt handler" will get address of "ISR" (from IVT) and execute ISR.
- When ISR is completed, execution resumes where it was paused

Maskable vs Non-maskable interrupt

- Maskable -- Interrupts can be disabled temporarily i.e. their processing (ISR) can be delayed.
  - Lower priority (w.r.t. Non-Maskable)
  - Most of hardware interrupts are Maskable.
  - e.g. 8085/86 -- INTR pin, ARM7 -- irq/fiq

- Non-Maskable
  - Interrupts cannot be disabled i.e. processing must be done immediately.
  - High priority Special hardware interrupts are Non-Maskable.
  - Usually these interrupt indicate severe failure. e.g. 8085/86 -- NMI pin, ARM CM3 -- NMI

Hardware vs Software interrupt

- Hardware -- interrupts from hardware peripherals.
- Software interrupt
  - Special instructions (Assembly/Machine level) when executed, current execution is paused, interrupt handler is executed and then the paused execution resumes.
  - Arch specific:
    - 8085/86: INT
    - ARM 7: SWI
    - ARM Cortex: SVC
    - Also called as "Trap" in few architecture

Interrupt Controller

- Convey the interrupts from various peripherals to the CPU.
- Also manage priority of the interrupt (when multiple interrupts arrives at same time).
- e.g. 8085/86 <-- 8259 (pic), Modern x86 processors (apic), ARM-7 (VIC), ARM-CM3 (NVIC), ...
- In few arch, interrupt controllers are programmable (e.g. ARM), while in few arch it is not programmable (e.g. AVR).
  - PIC can config priorities of the interrupts.
  - Non-programmable IC priorities of interrupts are fixed.

CPU Scheduling

- Modern OS are time-sharing systems i.e. CPU time is allocated to each process and after that time the next process is scheduled on the CPU.
- Timer Hardware (PIT/SysTick) is used periodically generate the interrupt.
- When interrupt is arrived, interrupt handler is executed, it stores current process execution context (in its PCB) and then invokes ISR.
- Then interrupt handler invokes scheduler.
- Scheduler check if time allocated to current process is completed and if completed then decide the next process to be executed (using some scheduling algorithm).
- This process execution context is then restored into CPU (from its PCB) by the dispatcher and the next process continues to execute.
- The whole process is also referred as "Context Switch".

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com