Real Time Operating Systems for Networked Embedded Systems

Miss. Priyanka R. Gulhane

Miss. Monika G. Pawar

Computer Science & Engineering,

PRMIT & R Badnera

2014-15

Presentation Outline

- Definitions
- Role of an OS in Real Time Systems
- Features of Real Time Operating Systems
  - Scheduling
  - Resource Allocation
  - Interrupt Handling
  - Other Issues
- Linux for Real Time Systems and RTLinux
- Other RTOS's

Real Time System

- A system is said to be **Real Time** if it is required to complete it’s work & deliver it’s services on time.
- Example – Flight Control System
  - All tasks in that system must execute on time.
- Non Example – PC system

Hard and Soft Real Time Systems

- Hard Real Time System
  - Failure to meet deadlines is fatal
  - example : Flight Control System
- Soft Real Time System
  - Late completion of jobs is undesirable but not fatal.
  - System performance degrades as more & more jobs miss deadlines
  - Online Databases

Hard and Soft Real Time Systems

(Operational Definition)

- Hard Real Time System

  The hard real-time system is to ensure that all deadlines are met.

- Soft Real Time System

  Demonstration of jobs meeting some statistical constraints suffices.

- Example – Multimedia System
  - 25 frames per second on an average

Role of an OS in Real Time Systems

- Standalone Applications
  - Often no OS involved
  - Micro controller based Embedded Systems
- Some Real Time Applications are huge & complex
  - Multiple threads
  - Complicated Synchronization Requirements

Features of RTOS’s

- Scheduling.
- Resource Allocation.
- Interrupt Handling.
- Other issues like kernel size.

Other RTOS issues

- Interrupt Latency should be very small
  - Kernel has to respond to real time events
  - Interrupts should be disabled for minimum possible time
- For embedded applications Kernel Size should be small
  - Should fit in ROM
- Sophisticated features can be removed
  - No Virtual Memory
  - No Protection

Other Problems with Linux

- Processes are non preemtible in Kernel Mode
  - System calls like fork take a lot of time
  - High priority thread might wait for a low priority thread to complete it’s system call
- Processes are heavy weight
  - Context switch takes several hundred microseconds

Linux Importance

- Coexistence of Real Time Applications with non Real Time system can be done.
  - Example http server
- Device Driver Base
- Stability

RTLinux

- Real Time Kernel at the lowest level
- Linux Kernel is a low priority thread
  - Executed only when no real time tasks
- Interrupts trapped by the Real Time Kernel and passed onto Linux Kernel
  - Software emulation to hardware interrupts
    - Interrupts are queued by RTLinux
    - Software emulation to disable_interrupt()

RTLinux (*contd*)

- Real Time Tasks
  - Statically allocate memory
  - No address space protection
- Non Real Time Tasks are developed in Linux
- Communication
  - Queues
  - Shared memory

Peripheral devices and protocols

- Interfacing

  Serial/parallel ports, USB

- Communication

  Serial, Ethernet, Low bandwidth radio.

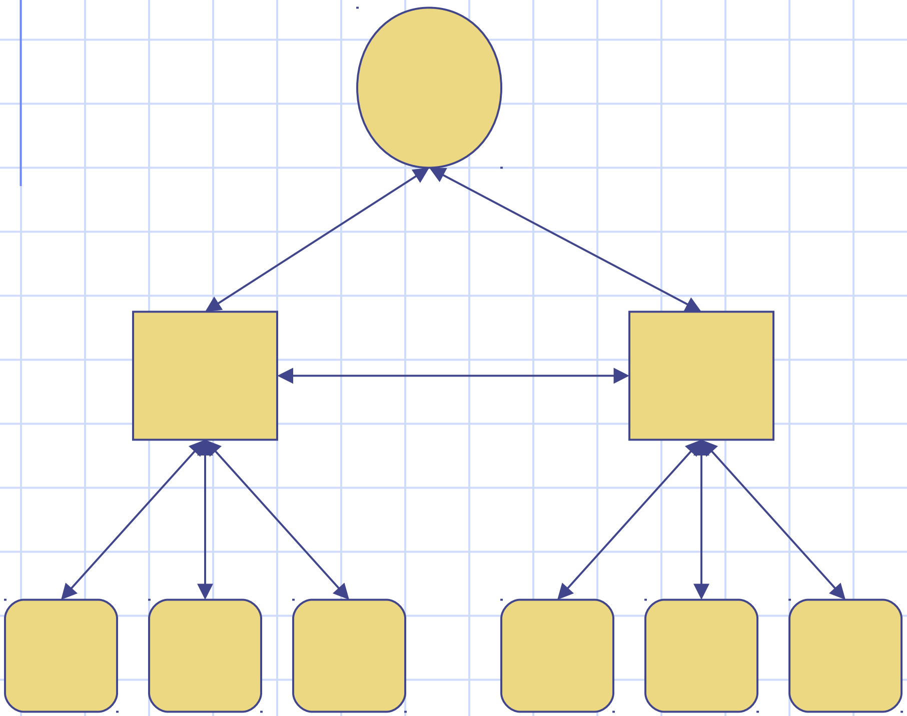
- User Interface

  LCD, Keyboard, Touch sensors, Sound, Digital pads, Webcams

- Sensors

  A variety of sensors using fire, temperature, water level, sound, vision

Fire alarm system: an example



Central server

Controller based

Low bandwidth radio links

Sensors: microcontroller based

- THANK YOU FOR LISTINING….