Practical 1: Comparative Study of RealTime System

GAHAN SARAIYA (18MCEC10), RUSHI TRIVEDI (18MCEC08)

18mcec10@nirmauni.ac.in,18mcec08@nirmauni.ac.in

I. AIM

Comparative study of any two Real Time Operating System

II. INTRODUCTION

This Lab experiments aims to reflect the comparative study and analysis of two Real Time Operating Systemdescribed below:

- RTLinux (Wind River Linux)
  - a small hard real-time OS microkernel, which is having priority higher than the Linux kernel which works beneath Linux.
  - Real-time kernel executes real-time tasks such as deterministic interrupt latency ISRs and other non-real-time tasks such as in deterministic task processing are moved to Linux kernel.
  - the entire OS runs as a fully preemptive process
  - The Linux kernel is modified by adding a virtual machine layer in between the computer hardware and standard linux kernel.
- RTAI (Real Time Application Interface)

In Linux there are two approaches are considered for real time systems:

1. Improving the Linux kernel preemption.
2. Adding a new software layer beneath Linux kernel with full control of interrupts and processor key features.

Among this approach RTAI and RTLinux uses approach 2

III. COMPARING ON VARIOUS PARAMETERS

I. Architecture

- RTLinux
  - i386
  - ARM
  - PPC
- Real Time Application Interface

  - i386
  - MIPS
  - PPC
  - ARM
  - m68k-nommu

II. Memory Management

- RTLinux
  - **Dynamic memory** – By default, memory must be reserved before the real time thread has started. No dynamic memory allocation (malloc and free functions) are available
  - **Shared memory** – MBUFF module (mbuff_alloc, mbuff_free) (Non-POSIX)
- Real Time Application Interface
  - **Dynamic memory** – Support for dynamic memory allocation (rt_malloc, rt_free). This functions do not have a hard real time behavior. (Non-POSIX)
  - **Shared memory** – MBUFF module (mbuff_alloc, mbuff_free) (Non-POSIX)
    SHMEM module (rtai_malloc, rtai_free, rtai_kmalloc, rtai_kfree). Shmem is the RTAI version, developed by Paolo Mantagazza, which operates in much the same way but is dependant upon RTAI (Non-POSIX)

III. Interprocess communications

- RTLinux
  - FIFO
    UNIX PIPE's like communication mechanism that can be used to communicate real-time process between them and also with normal linux user processes. (non-POSIX)
  - Mailboxes
    Jerry Epplin IPC implementation (rt_mq_open, rt_mq_send etc.) (obsoleted) (Non-POSIX).
- Real Time Application Interface
  - FIFO
    UNIX PIPE's like communication mechanism. Same than in RTLInux. (non-POSIX)
    Also provides a mechanism to create fifos by name.
  - Mailboxes
    RTAI provides mailboxes. Messages are ordered in FIFO order. Different size messages of are allowed. Multiple senders and receivers can read and write messages to the same mailbox. There are several sending an receiving functions that provides a lot of flexibility: blocking, non-blocking, timed and whole/partial message.
  - Message queues
    Provides four different (incompatible) intertask messages facilities

  - net_rpc
    RTAI has extended its API to allow remote (other host) procedure call RPC. New API functions has been added with the following syntax: replace the first two letters of the function name (for example: given the rt_mbx_send(), the new function RT_mbx_send() has been added); and the new function has two new parameters: node and port. This feature do not comply with any communication standard.

Too many incompatible, redundant and non standard features is not desirable. A RTOS should provide a simple API.

IV. Synchronization

- RTLinux
  - Mutex
    POSIX thread mutex variables. Provides PRIORITY_PROTECT protocol to deal with the priority inversion problem.
  - POSIX conditional variables.
  - POSIX semaphores.
- Real Time Application Interface
  - Mutex
    POSIX thread mutex variables. Provides PRIORITY_PROTECT protocol to coping the priority inversion problem.
  - POSIX conditional variables.
  - Semaphores
    POSIX semaphores
    RTAI semaphores (rtf_sem_init, rtf_sem_wait, ...). The underling mechanism used to implement this semaphores is the FIFO's blocking features.

V. Memory Management

- RTLinux
  -
- Real Time Application Interface
  -