Useful Documentation

Before the start of your lab, please review the following resources:

- Monitors: "course notes" => "Process Description and Control" => "Conncurrency" => "Monitors"
- IPC: "course notes" => "Process Description and Control" => "Concurrency" => "IPC"

Provided Files

- reader-writer.c - the reader-writer test application.
- reader-writer-monitor.c - contains helper functions for the reader-writer application.
- liblist.a - list library from Assignment 1. You may choose to use your own instead.
- list.h - header file

Description

Overview

In this lab, you will learn how to implement a monitor using IPC from the RT Threads library. This will help with your understanding of Assignment 2 - Part B, but is also a likely candidate for an ***exam question***, so please make sure you understand the relationship between IPC, monitors, and semaphores. Ask plenty of questions! You *may* work with your partner on this implementation.

Monitors

Monitors provide a method for synchronizing between threads or processes. Similar to assignment 2, you are to implement 6 functions:

- **RttMonInit()** - called at the start of the program to initialize the monitor.
- **RttMonEnter() / RttMonLeave()** - called by a process entering / exiting the monitor.
- **RttMonWait(int CV)** - causes the calling process to wait on some condition variable (CV). The effect is that the calling process exits the monitor, is added to the associated CV queue, and waits until it is signalled by another process.

- **RttMonSignal(int CV) -** signals the process at the head of the CV queue, to resume running. The effect is that the calling process must exit the monitor, is added to an urgent queue, and waits until the other process leaves the Monitor; this ensures that only one process is running in the Monitor at a time.
- **MonServer() -** Server process that handles the coordination by putting processes on lists according to the semantics of Monitors. Receives messages from the client processes and determines which list they are moved onto and off of and who to reply to such that the proper coordination is achieved.

Deliverables

A single tar file containing your `Makefile`, `reader-writer-monitor.h`, `monitor.c`, `monitor.h` and `gitlog.txt`. Name the tar file lab4.tar.

Submission

Create a .tar file with the 5 files and submit to Lab 4. Your name, NSID and Student number must be at the beginning of every individual file that you hand in.

Grading

- `Makefile`: 1 mark
- `monitor.c`: 1 mark
- `monitor.h`: 1 mark
- `reader-writer-monitor.h`: 1 mark
- `gitlog.txt`: 1 mark