Sistemas de Tiempo Real
Departamento de Ingeniería Electrónica
Universidad de Antioquia
2025-2

Práctica No. 4
Implementación de Tareas Periódicas de Tiempo-Real

Realización: En parejas.

Fecha de entrega: viernes 15 de octubre del 2025.

Objetivos:

- Comprender el uso de los *system calls*, *signals* y más elementos disponibles en el **API POSIX** de Linux para la implementación de **tareas periódicas de tiempo-real**.
- Identificar y utilizar el flujo de implementación y configuración de temporizadores y relojes de tiempo-real basados en el estándar **POSIX-RT**.
- Emular un sistema de control de un automóvil compuesto por diferentes tareas periódicas de tiempo-real basadas en el **POSIX-RT**.

Preguntas orientadoras:

1. Dentro del estándar de **POSIX-RT**, ¿qué diferencias existen entra la implementación de *temporizadores* y *relojes* de tiempo-real?
2. Desde el punto de vista del sistema operativo, ¿cuál es el rol de los *system calls*, eventos y señales en la definición de las diferentes tareas periódicas que posee un determinado **sistema de tiempo-real**? (**Explique**).

Implementos necesarios:

- Un computador PC.
- Una instalación reciente del sistema operativo Linux[1], instalado en una partición física de su disco duro, en el entorno **WSL** de Windows, o en una *máquina virtual*.
- Una tarjeta **Raspberry Pi 4** y una SD card de 16GB ó 32GB.

[1] Se recomienda la instalación de una versión reciente de una distribución Linux basada en **UBUNTU**.

1. Hilos Periódicos en POSIX-RT:

Estudie nuevamente los ejemplos analizados en clase sobre el uso de *System calls* y *signals* en el **API POSIX** de Linux, los cuales se encuentran disponibles en el siguiente repositorio de **Github**:

https://github.com/gpatigno/RTS_2025-2/tree/main/TALLER_UNIDAD2/POSIX-RT

2. Sistema de Control de un Automóvil:

2.1. Diseño del sistema:

Diseñe un sistema de control de un automóvil, compuesto por **una etapa de medición de la velocidad**, un **controlador de frenado ABS**, un **controlador de inyección de combustible**, y un **subsistema de tiempo-real *leve*** escogido por usted, tal que cada etapa sea periódica con los períodos indicados en la **Tabla 1**.

Tarea (τ_i)	Función de la tarea	Período T_i
τ_1	Medida de velocidad	20 ms (50 Hz)
τ_2	Control ABS	40 ms (25 Hz)
τ_3	Inyección de Combustible	80 ms (12.5 Hz)
τ_4	Una tarea de tiempo-real *leve*	Definido por usted

Tabla 1: Tareas de tiempo-real, y sus respectivos períodos.

2.1.1. Para cada tarea indicada, escriba un código fuente en **ANSI C** que implemente en diferentes hilos la funcionalidad requerida, además de su periodicidad mediante instrucciones del **POSIX-RT** de Linux.

2.1.2. Entre las tareas indicadas, implemente dos tareas mediante temporizadores de tiempo-real (**POSIX Timers**), y las otras dos tareas mediante relojes de tiempo-real (**POSIX Clocks**).

2.1.3. Con respecto al *Cuerpo de cada Tarea* (**Job Body**), lleve a cabo algún procesamiento relacionado a la funcionalidad esperada para dichas tareas[2].

2.1.4. Considere además que al menos dos de las tareas descritas comparten el acceso y manipulación de algún arreglo de datos, de manera que sea necesario la definición de sincronización y exclusión mutua para el uso de dicho arreglo.

2.2. Condiciones del código fuente requerido:

Para la implementación y presentación de su código fuente, tenga en cuenta las siguientes indicaciones:

2.2.1. Divida su sistema en diversos archivos ***.c** y ***.h** requeridos para la creación del ejecutable de su programa, y documente su código fuente con comentarios claros y explicativos.

- Indique explícitamente en su código fuente la implementación y configuración de los cuatro hilos relacionados a las tareas de tiempo-real descritas anteriormente.
- Igualmente, explique la implementación del arreglo de datos compartido entre al menos dos hilos de su código fuente.
- Describa su estrategia de implementación de la sincronización requerida para el acceso y manipulación de dicho arreglo de datos.
- Indique las instrucciones de sincronización y comunicación necesarias para la **ejecución concurrente** de cada hilo de su sistema.

2.2.2. Muestre y explique la definición y configuración de las señales de tiempo-real requeridas para la notificación de los eventos requeridos. ¿Cuáles son estos eventos? **Explique**.

2.2.3. Describa la definición y configuración de los temporizadores y relojes de tiempo-real con sus *offset* y períodos respectivos.

- En su código fuente identifique explícitamente los cuatro períodos asignados a cada tarea según lo indicado en la **Tabla 1**.
- Igualmente, muestre en su código el uso de las *System Call*, eventos y señales requeridas para cada temporizador.

[2] ¡No requerimos implementar un verdadero *sistema de control de un automóvil*!

2.2.4. Escriba un **Makefile** que permita compilar fácilmente los anteriores archivos requeridos en su programa, teniendo en cuenta la habilitación de las opciones de compilación multihilo **(-lpthread**) y de instrucciones de tiempo-real (**-lrt**).

2.2.5. Compile su código fuente y ejecútelo en su **PC-Linux** tantas veces como sea necesaria, a fin de observar el buen funcionamiento de su sistema multihilo de tiempo-real para el control de un automóvil.

- Presente pantallazos de la ejecución de su código evidenciando el funcionamiento de su sistema, así como los tiempos de ***offset*** y períodos creados para cada hilo.
- Demuestre mediante gráficos o tablas la correcta ejecución sincrónica de su sistema en relación con el arreglo de datos compartido.

3. Ejecución en la Raspberry Pi 4 (Opcional)[3]

3.1. Siguiente el mismo procedimiento llevado a cabo en la **Práctica No. 3** sobre *Cross*-compilación, realice la respectiva compilación cruzada y ejecución en la **Raspberry Pi 4** de su sistema de control de un automóvil.

3.2. Presente evidencia del funcionamiento de su programa en la **Raspberry Pi 4**, y del uso de los cuatro núcleos de procesamiento de este dispositivo.

3.3. Igualmente, presente evidencia de la ejecución periódica de cada hilo, según los tiempos requeridos en la **Tabla 1**.

3.4. Demuestre mediante gráficos o tablas la correcta ejecución sincrónica de su sistema en relación con el arreglo de datos compartido.

3.5. Elabore un video presentado la correcta ejecución de todo su sistema.

3.6. Presente un análisis comparativo del desempeño observado entre la ejecución de su programa en la **Raspberry Pi 4**, y en su **PC-Linux**.

4. Informe

4.1. Realice su informe con el análisis y respuestas pedidas en la presente guía.

4.2. Presente sus conclusiones generales del trabajo realizado en toda esta Práctica de Laboratorio, e indique la bibliografía utilizada para realizar este informe.

4.3. Adjunte a este informe un archivo zip con el contenido de su carpeta de trabajo, albergando códigos fuente, *Makefile* y pantallazos de las salidas obtenidas en cada ejecución.

[3] La realización exitosa de este ejercicio adicional puede otorgar hasta un **25%** del **Parcial No. 1** del presente curso.