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

Práctica No. 3

Concurrencia y Sincronización en la Raspberry Pi

Realización: En parejas.

Fecha de entrega: 03 de octubre del 2025.

Objetivos:

- Comprender el diseño y desarrollo de sistemas concurrentes, la comunicación de sus diferentes procesos e hilos, y su sincronización en el uso de recursos compartidos.
- Comprender el proceso de compilación cruzada para un procesador diferente al procesador local de tu computador.

Preguntas orientadoras:

1. Desde el punto de vista del desarrollo de **Sistemas de Tiempo-Real**, ¿qué utilidad posee el proceso y compilación cruzada (*crosscompilation*)?
2. ¿Qué beneficios y desafios presenta la comunicación y sincronización de procesos e hilos en un **Sistema Concurrente de Tiempo-Real**?
3. Para un desarrollador de software y de sistemas embebidos, ¿presenta alguna utilidad el uso de *scripts* y de archivos **Makefile**? (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.
- Cable de Ethernet.
- Conexión a WiFi.
- Cable USB de suministro de energía, o adaptador de energía propio de la **Raspberry.**

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

I. Introducción

1. Concurrencia y Sincronización en C++:

Estudie cuidadosamente los ejemplos de código C++ analizados y ejecutados en nuestra clase sobre ***Mutual Exclusion and Synchronization***. Recuerde que dichos ejemplos se encuentran en el siguiente repositorio:

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

II. *Cross*-compilación para la Raspberry Pi

1. Instalación de paquetes y herramientas de compilación cruzada

En una instalación de Linux de su preferencia, ejecute los siguientes comandos en el Terminal (*Console*):

sudo apt update && sudo apt upgrade -y

sudo apt install build-essential git -y

sudo dpkg --add-architecture arm64

sudo apt update

sudo apt install crossbuild-essential-arm64

2. Verificación de la instalación de compiladores cruzados (*cross-compilers*)

2.1. Luego de la correcta ejecución de las anteriores instrucciones e instalación de los paquetes respectivos, verifique la versión del compilador cruzado de **GCC** para el **ARM64**:

aarch64-linux-gnu-gcc --version

La salida esperada luego del anterior comando, es el ***Copyright (C)*** del **GCC** para la arquitectura **ARCH64** y para la distribución de **Linux** que usted tiene instalada.

2.2. En la carpeta **/usr/bin/** (o en **/usr/local/bin/**) puede verificar la presencia de todos los compiladores cruzados instalados para la arquitectura **ARCH64**:

cd /usr/bin/

ls -al aarch64-linux*

3. Prueba rápida de los crosscompiladores creados

3.1. En su carpeta de usuario dentro de la distribución Linux escogida para la presente práctica, cree la carpeta **LAB3_RTS**.

3.2. Ingrese a esta carpeta, y cree una nueva carpeta llamada **Pruebas**:

cd $HOME/LAB3_RTS

mkdir Pruebas

3.3. Cree un ejemplo simple de *"Hello World"*:

cat > test.c

#include <stdio.h>

int main() { printf("\nHello, world!\n\n "); return 0; }

3.4. Cierre este archivo con **Ctrl D**.

3.5. Compile y ejecute este ejemplo con el **gcc** convencional de su distribución Linux instalada:

gcc -o exec-x86_64-linux test.c

./exec-x86_64-linux

3.6. Ejecute el siguiente comando, y explique el resultado mostrado:

file exec-x86_64-linux

3.7. Compile ahora el anterior código fuente con el **gcc** creado para el **ARCH64**. Luego intente ejecutarlo:

aarch64-linux-gnu-gcc -o exec-arch64_raspbpi test.c

./exec-arch64_raspbpi

La salida de este último comando debió presentar un error al intentar ejecutar dicho ejecutable. Esto se debe a que este archivo sólo sirve para ejecutarse en el procesador **ARM64** incluido en la **Raspberry Pi 4**.

3.8. Ejecute el siguiente comando, y explique el resultado mostrado:

file exec-arch64_raspbpi

Si este ejemplo sencillo presentó la información y las salidas esperadas en cada caso, entonces el *Toolchain* está listo para *cross*-compilar aplicaciones en **C** y **C++** para el sistema **Raspberry Pi 4**.

4. Automatización del proceso de compilación cruzada

4.1. Estudie algunos de los videos del siguiente tutorial sobre el comando **Make** y la escritura de **Makefile**, y haga uso de esta herramienta tanto como considere útil y conveniente, a fin de automatizar el proceso de compilación y crosscompilación en la presente Práctica:

https://www.youtube.com/playlist?list=PLTd5ehIj0goOrqKZPvq1Np-8PUFcQSSm-

4.2. Seleccione ahora alguno de los ejemplos compartidos en el repositorio de **Github** indicado anteriormente, y en su nuevo archivo **Makefile** agregue las instrucciones requeridas para llevar a cabo sobre el ejemplo escogido, el mismo procedimiento de compilación y compilación cruzada realizado anteriormente con el ejemplo de *"Hello World"*.

4.3. Verifique el buen funcionamiento de su **Makefile**, y del programa compilado para su distribución local (**host**) de Linux.

4.4. Igualmente con el ejemplo escogido, realice los pasos indicados en los items **3.7** y **3.8**.

5. Análisis y compilación cruzada de una Aplicación Concurrente Real

5.1. Realice ahora una copia del siguiente repositorio referente a la implementación en C++ de un sistema para el sensado y registro de temperatura y humedad:

https://github.com/gpatigno/RTS_2025-2/tree/main/TempHumidLogger

5.2. Estudie y analice el código fuente suministrado, e identifique las fallas de comunicación y sincronización que este código presenta. En su informe de esta Práctica de laboratorio explique las fallas detectadas.

5.3. Modifique dicho código fuente a fin de agregar instrucciones de comunicación adecuadas, así como instrucciones de sincronización de los recursos compartidos por los diferentes hilos que componen la aplicación.

5.4. Realice tantas mejoras en este código como usted y su compañero consideren adecuado.

5.5. Agregue comentarios explicativos en todo el código fuente, y en sus nuevas instrucciones agregadas.

5.6. Cree una copia del **Makefile** creado anteriormente, y en su nueva copia elabore el proceso de compilación y compilación cruzada de los diferentes archivos que componen este sisterma concurrente.

5.7. Compile su código modificado, y ejecute el programa resultante en su distribución **host** de Linux.

5.8. Verifique el buen funcionamiento de su nueva versión del sistema analizado, colocando especial atención en la ejecución y comunicación de cada hilo, y en la sincronización de los recursos compartidos.

5.9. Realice capturas de pantalla de la ejecución de su código, y presente su análisis respectivo en el informe de la Práctica.

III. Ejecución de Programas en la Raspberry Pi

1. Ejemplo 1: *Hello World*

1.1. Siga el **Anexo I** compartido con la presente guía, a fin de ejecutar en la **Raspberry Pi 4** el archivo de prueba compilado en el numeral **II.3.7** de la presente guía.

1.2. Comparta un breve video mostrando la ejecución de este ejemplo en la **Raspberry Pi 4**.

2. Ejemplo 2: Aplicación Concurrente Real

2.1. Siguiente el mismo procedimiento llevado a cabo en el paso anterior, envíe a la **Raspberry Pi 4** el archivo ejecutable obtenido en la *cross*-compilación de su versión modificada del programa para el sensado y registro de temperatura y humedad.

2.2. Ejecute su programa en la **Raspberry Pi 4** y elabore un breve video describiendo la correcta ejecución de su sistema concurrente.

IV. Informe

1. Realice el informe con su respuesta a las preguntas orientadoras indicadas al comienzo de esta guía de laboratorio.
2. Además, presente la respuesta a las preguntas y análisis pedidos en esta guía.
3. Presente los detalles de configuración y compilación cruzada (*cross-compilation*) de los ejemplos analizados en esta práctica de laboratorio. **Explique cada procedimiento.**
4. Presente los detalles de ejecución de los ejemplos analizados en esta práctica de laboratorio. **Explique cada procedimiento.**
5. Presente conclusiones y bibliografía utilizada para realizar este informe.
6. Adjunte a este informe un archivo **zip** con el contenido de su carpeta **LAB3_RTS** y los archivos de salida obtenidos en cada caso.