Investigación

Investigue posibles métodos (bibliotecas, apis, etc) para el uso de hilos bajo el sistema operativo GNU Linux (Por ejemplo pthreads)

Bare-Bones Threads	Desarrollado por Christopher Neufeld, es una librería básica de hilos bare-bones. Han sido probados en CPU singulares y contenedores de Linux SMP. Basado en clones e incluye un programa de demostración.
CLthreads	Desarrollado por Pavel Krauz, es una biblioteca pthreads basada en clone(). Incluye libc parcial, soporte POSIX 1003.1b e incluye programas de ejemplo.
DCEthreads	Desarrollado por Michael Peterson, es una librería multihilo para sistemas Intel basados en Linux y está basada en el estándar POSIX 1003.1c. El kit contiene las fuentes de la biblioteca (libpthreads), un entorno de construcción para construir las versiones ELF y A.OUT de la biblioteca, y un completo conjunto de páginas de manual para todas las funciones POSIX .1c.
FSU Pthreads	Desarrollado por Frank Mueller, es una biblioteca de C que implementa hilos POSIX para SunOS 4.1.x, Solaris 2.x, SCO UNIX, FreeBSD y Linux. Usada para GNU Ada Runtime; soporte parcial de libc. Librería pthreads de espacio de usuario; usada para GNU Ada Runtime; soporte parcial de libc, POSIX 1003.4; documentación de pthreads.
JKthread	Desarrollado por Jeff Koftinoff, es un experimento con la llamada clone de Linux 2.0 para implementar hilos del núcleo utilizables en un programa de usuario. Estos jkthreads tienen una API que no tiene nada que ver con pthreads o hilos de Win32 o hilos de BeBox. Incluye envoltorios C++.
LinuxThreads	Desarrollado por Xavier Leroy, es una implementación del paquete de hilos Posix 1003.1c para Linux. A diferencia de otras implementaciones de los hilos Posix, LinuxThreads proporciona hilos en el espacio del núcleo: los hilos se crean con la nueva llamada al sistema clone y toda la programación se hace en el núcleo. Incluye soporte parcial de libc y ejemplos de programas.

LWP	Desarrollado por Stephen Crane, es una pequeña biblioteca de procesos ligeros y portátiles para sun3/4, mips-ultrix, 386BSD, HP-UX y Linux.
NThreads (Numerical Threads)	Desarrollado por Thomas Radke, es un paquete de hilos de espacio de usuario incluido con parches para el kernel de Linux para soportar los hilos del kernel.
PCthreads	Desarrollado por Michael Peterson, es un librería pthreads de espacio de usuario; incluye no-bloqueo select(), read(), y write(). Páginas de manual incluidas. Requiere DCEThreads.
Provenzano Pthreads	Desarrollado por Christopher Provenzano, es una librería pthreads de espacio de usuario con llamadas de sistema que bloquean los hilos (leer, escribir, conectar, aceptar, dormir, esperar, etc.) y una librería C segura para los hilos (stdio, net utilities, etc.) Se distribuye una versión separada con la fuente libc de Linux pero puede que no se construya por defecto.
QuickThreads	Desarrollado por David Keppel, es un conjunto de herramientas para construir paquetes de hilos de espacio de usuario. Tenga en cuenta que está diseñado para facilitar la construcción de paquetes de biblioteca de hilos; por lo tanto, omite muchas funciones de nivel superior.
RexThreads	Desarrollado por Stephen Crane, es una biblioteca de hilos del espacio de usuario.

Tabla 1. Métodos para el uso de hilos en Linux [1].

¿Qué es el concepto de mutex en multiprogramación y qué busca hacer?

El mutex, también conocido como spinlock es la herramienta más simple de sincronización que es utilizado para proteger las regiones críticas y así evitar las race conditions. Es decir, un hilo debe adquirir un bloqueo antes de entrar en una sección crítica. Libera el bloqueo cuando sale de la sección crítica [2].

¿Qué sucede cuando dos hilos quieren utilizar el mismo recurso? ¿Cómo se manejan estos casos?

Cuando más de un hilo compite al mismo tiempo, algunos hilos pueden recibir un estado inválido si otro hilo modifica el recurso al mismo tiempo. Para evitar esta situación, puedes ayudar a proteger secciones críticas de tu código de las run condition usando locks [3].

Ejercicio Hilos - Parte 2

Dentro de la función

1. La función tiene un proceso que contiene un for loop que va desde un entero 0 hasta 99.
2. Antes de comenzar el for loop se realiza un pthread_mutex_lock que restringe el acceso de múltiples threads al loop.
3. Al finalizar el for loop se realiza el pthread_mutex_unlock lo cual permite que el siguiente hilo pueda utilizar el for loop.

Dentro del main

1. Se crea un pthread_mutex_init para inicializar el mutex.
2. Se crea el primer thread y ejecuta la función.
3. Se crea el segundo thread e intenta ejecutar la función.
4. El segundo thread queda en espera debido al mutex.
5. Se termina de ejecutar el primer thread.
6. Se ejecuta el segundo thread.
7. Se termina de ejecutar el segundo thread.
8. Se finaliza el mutex utilizando pthread_mutex_destroy.

Referencias

[1] “Linux Threads Home Page: Are there threading libraries? Where?,” *Tldp.org*, 2020. [Online]. Available: https://tldp.org/FAQ/Threads-FAQ/ThreadLibs.html. [Accessed: 16-Sep-2020].

[2] bmurphy1976, “What is a mutex?,” *Stack Overflow*, 29-Aug-2008. [Online]. Available: https://stackoverflow.com/questions/34524/what-is-a-mutex. [Accessed: 16-Sep-2020].

[3] *Microsoft.com*, 2020. [Online]. Available: https://support.microsoft.com/en-in/help/816161/how-to-synchronize-access-to-a-shared-resource-in-a-multithreading-env#:~:text=Multiple%20threads%20may%20try%20to,resource%20at%20the%20same%20time. [Accessed: 16-Sep-2020].