UNEMI
UNIVERSIDAD ESTATAL DE MILAGRO

FACULTAD DE CIENCIAS E INGENIERÍA
CARRERA DE INGENIERÍA DE SOFTWARE

TEMA:

HILOS POXIS

AUTORES:

Jaramillo Yepez Ezequiel Cristobal

ASIGNATURA:

Sistemas Operativos

DOCENTE:

Javier Ricardo Bermeo Paucar

FECHA DE ENTREGA:

Viernes, 20 de septiembre de 2024

PERIODO:

2024 - 2024

MILAGRO-ECUADOR

Contenido

1. Introducción

La concurrencia y el paralelismo se han convertido en elementos clave en el desarrollo de software moderno, especialmente en sistemas que requieren la ejecución concurrente de múltiples tareas. En este contexto, los hilos POSIX (Portable Operating System Interface for Unix) han surgido como una de las herramientas más potentes y flexibles para implementar programas concurrentes en sistemas Unix y Linux. Los hilos permiten a un proceso dividir su carga de trabajo en unidades de ejecución independientes y optimizar el uso de recursos del sistema como el procesador y la memoria. A diferencia de los procesos tradicionales, los hilos comparten el mismo espacio de direcciones, lo que facilita la comunicación y sincronización entre hilos, al tiempo que minimiza el coste de creación y gestión de recursos. Esta característica lo hace ideal para aplicaciones en las que se requiere una alta eficiencia, como servidores, sistemas en tiempo real y aplicaciones científicas; como los hilos POSIX están estandarizados, los programas desarrollados sobre este modelo pueden portarse entre distintas plataformas Unix.

2. Desarrollo

3. Hilos Poxis

3.1.Descripción

En los sistemas operativos modernos, los hilos son un componente fundamental para ejecutar tareas en paralelo. Cada hilo puede considerarse como una pequeña unidad de ejecución dentro de un proceso mayor. Los hilos permiten a un programa realizar varias tareas a la vez, como interactuar con un usuario, leer/escribir archivos o calcular resultados complejos en segundo plano, ya que los hilos POSIX están diseñados para ser portables entre diferentes sistemas Unix, El código que utiliza hilos POSIX puede ser reutilizado en diferentes plataformas sin grandes cambios.

3.2.Creación de hilos

En POSIX, los hilos se crean utilizando la función pthread_create(). Esta función toma cuatro argumentos. Un puntero al identificador del hilo, un puntero a los atributos del hilo, una función a ejecutar por el hilo y argumentos a pasar a la función del hilo. Ejemplo:

```
#include <pthread.h>
#include <stdio.h>

void* funcion_hilo(void* arg) {
printf("Hilo creado\n");
return NULL;
}

int main() {
pthread_t hilo;
pthread_create(&hilo, NULL, funcion_hilo, NULL);
pthread_join(hilo, NULL);
return 0;
}
```

3.3.Atributos de los hilos

Los atributos de los hilos como la prioridad, la pila y el modo de desactivación pueden establecerse utilizando el tipo pthread_attr_t. Estos atributos permiten a los hilos personalizar su comportamiento antes de ser creados. Los atributos pueden inicializarse utilizando pthread_attr_init() y atributos específicos pueden establecerse utilizando funciones como pthread_attr_setdetachstate().

3.4.Terminación de hilos

Los hilos se pueden terminar de varias maneras:

- Retornando desde la función main.
- Terminando explícitamente el hilo llamando a **pthread_exit().**
- Cancelado por otro hilo con **pthread_cancel().**

3.5.Esperar la terminación de los hilos

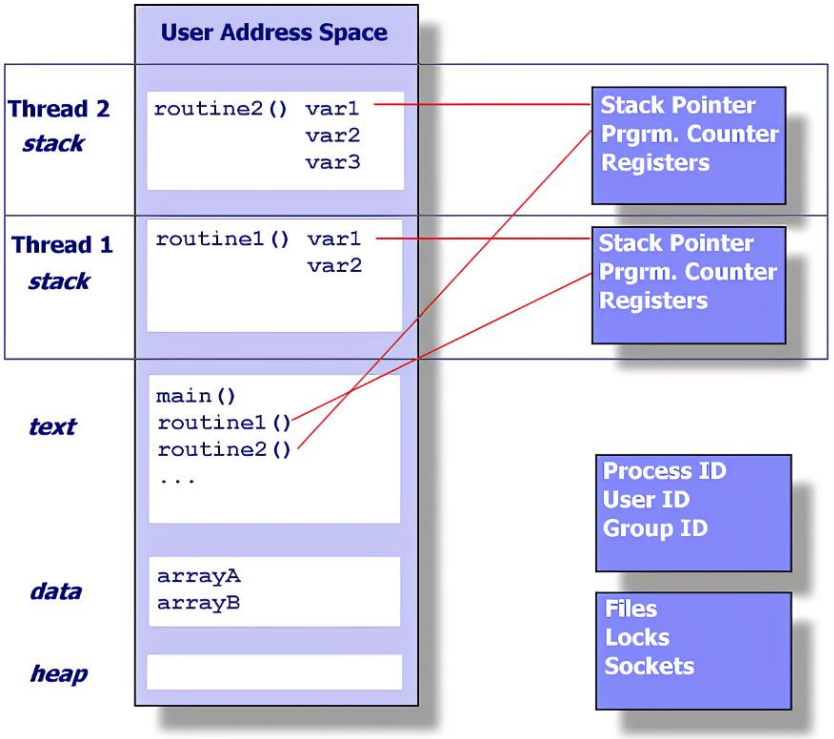
En la mayoría de las aplicaciones concurrentes, es importante que el hilo principal (por ejemplo, el proceso padre) espere a que el hilo secundario termine de ejecutarse. El estándar POSIX proporciona una función específica para que un subproceso espere a que otro salga, permitiendo una coordinación eficiente entre subprocesos y errores como la pérdida de datos o la corrupción de información compartida. evitando la pérdida de datos y la corrupción de la información compartida.

3.6.Cancelación de hilos.

Los hilos POSIX pueden ser terminados por otros hilos o por el propio proceso principal. La cancelación es útil cuando una tarea ya no es necesaria o cuando se detecta un problema que requiere la finalización de una operación concreta. La terminación puede ser controlada por el

hilo receptor, permitiendo al programador diseñar mecanismos para asegurar que el hilo termina de forma segura, los recursos son liberados y los datos no se pierden.

3.7.Ejemplo:



El ejemplo ilustra cómo los hilos POSIX permiten **concurrencia** dentro de un proceso. Cada hilo puede ejecutar una rutina o función diferente, pero todos ellos comparten los mismos recursos del proceso. La separación de pilas y registros para cada hilo asegura que puedan ejecutar sus tareas sin interferir con los demás, mientras que el uso de un espacio de direcciones compartido facilita la comunicación entre hilos, sin la sobrecarga de gestionar procesos separados.

4. Conclusión

Los hilos POSIX son una potente herramienta de programación concurrente y paralela, que permite a los desarrolladores maximizar el rendimiento del sistema dividiendo las tareas en unidades más pequeñas y ejecutándolas simultáneamente dentro de un único proceso. A diferencia de los procesos con su propio espacio de direcciones, los hilos comparten el mismo espacio de memoria, lo que facilita la comunicación entre hilos y reduce la sobrecarga del sistema al no ser necesario duplicar recursos. Esta característica es especialmente valiosa en aplicaciones que manejan grandes cantidades de datos o requieren tiempos de respuesta elevados, como servidores web, sistemas en tiempo real y simulaciones científicas.

Dado que los hilos pueden crearse, sincronizarse y cancelarse de forma eficiente, el modelo de hilos POSIX es muy flexible y adaptable a diferentes necesidades. Sin embargo, también plantea retos, especialmente cuando se trata de gestionar recursos compartidos y sincronizar entre hilos. Si los mecanismos de sincronización, como los mutex y los semáforos, no se implementan correctamente, pueden producirse problemas como las condiciones de carrera y bloqueos.

En conclusión, el uso adecuado de hilos **POSIX** puede permitir el procesamiento paralelo y mejorar significativamente la eficiencia de las aplicaciones. Sin embargo, los programadores necesitan hacer un buen uso de las técnicas de sincronización y gestión de recursos compartidos para evitar errores y asegurar la integridad de los datos. En última instancia, los hilos POSIX proporcionan un equilibrio entre flexibilidad, rendimiento y complejidad, lo que los convierte en una opción ideal para aplicaciones que requieren una alta concurrencia y un uso eficiente de los recursos del sistema.

5. Bibliografía

Butenhof, D. R. (1993). Programming with POSIX threads. Addison-Wesley Professional.

Barney, B. (2009). POSIX threads programming. National Laboratory. Disponível em:< https://computing. llnl. gov/tutorials/pthreads/> Acesso em, 5, 46.

GonzaLez,A.J.(s. f.).*POSIXthreads*. http://profesores.elo.utfsm.cl/~agv/elo330/2s07/lectures/POSIX_Threads.html

Herrera, J. (2024, 19 enero). *Estándar POSIX, ¿qué es y para qué sirve?* Guía Hardware. https://www.guiahardware.es/estandar-posix/

Gabriel Gerónimo, C. Un Vistazo a los Sistemas Distribuidos _.