Sistemas operativos II

**ASPECTOS DEL DISEÑO DE PAQUETES DE HILOS. (190 tarea)**

Un conjunto de primitivas relacionas con los hilos disponibles para los usuarios se llama un paquete de hilos.

El primer aspecto que analizaremos es el manejo de los hilos. Aquí se tienen dos alternativas, los hilos dinámicos y los estáticos. En un diseño estático, se elige el número de hilos al escribir el programa o durante su compilación.

Un método más general consiste en permitir la creación y destrucción de los hilos durante la ejecución. La llamada para la creación de hilos determina el programa principal del hilo y un tamaño de pila, así como otros posibles parámetros; ejemplo una prioridad de planificación. La llamada regresa por lo general un identificador de hilo; para utilizarlo en las llamadas posteriores relacionadas con el hilo. En este modelo, un proceso se inicia (manera implícita) con un hilo, pero puede crear el número necesario de ellos y estos pueden expirar al terminar.

Los hilos pueden concluir de dos maneras. Un hilo puede hacer su salida por su cuenta, al terminar su trabajo, o puede ser eliminado desde el exterior. Los hilos son como los procesos.

Puesto que los hilos comparten una memoria común, pueden y de hecho la utilizan, para guardar datos que comparten los distintos hilos, como los buffers en un sistema de productores y consumidores. El acceso a los datos compartidos se programa por lo general mediante regiones críticas, para evitar que varios hilos intenten tener acceso a los mismos datos al mismo tiempo. La implantación de las regiones críticas es más fácil si se utilizan los semáforos, monitores u otras construcciones similares. Una técnica en común en los paquetes de hilos es el MUTEX. Que es un cierto topo de semáforo moderado. Un MUTEX solo tiene dos estados cerrado y no cerrado. Se definen dos operaciones de MUTEX. La primera LOCK, intenta cerrar MUTEX. Si el MUTEX no está cerrado, LOCK tiene el éxito y el MUTLEX se cierra en una acción atómica. Si dos hilos intentan cerrar al mismo MUTEX en el mismo instante, evento que solo es posible en un multiprocesador, en el cual existen varios hilos en ejecución en distintos CPU, uno de ellos gana y el otro pierde. Si un hilo intenta cerrar un MUTLEX ya cerrado, se bloquea.

La operación UNLOCK elimina la cerradura de un MUTLEX. Si uno o más hilos esperan a un MUTLEX, se libera exactamente uno de ellos. El resto continua su espera.

Otras operaciones que se tienen en ciertos casos es TRYLOCK que intenta cerrar un MUTLEX, si el MUTLEX no está cerrado TRYLOCK regresa un código de estado que indica el éxito. MUTLEX cerrado TRYLOCK no bloque el hilo sino regresa un código de estado que indica la falla.

Otra característica de sincronización que a veces está disponible en los paquetes de hilos es la variable de condición, similar a la variable de condición que se utiliza para la sincronización en el caso de los monitores. Por lo general, se asocia una variable de condición a un MUTEX cuando este se crea. La diferencia entre el MUTLEX y la variable de condición es que el primero se utiliza para una cerradura a corto plazo, principalmente para proteger la entrada a las regiones críticas. Las variables de condición se utilizan para una espera a largo plazo hasta que un recurso esté disponible.

Una solución es el uso de las variables de condición para adquirir el recurso. En este caso, la espera de la variable de condición se define de modo que ejecute la espera y elimine la cerradura en forma atómica. Más adelante, cuando el hilo que conserva el recurso lo libera, llama a WAKEUP que se define de forma que despierte a un hilo o bien a todos los hilos que esperan a la variable de condición especifica. El uso de WHILE en vez de IF, evita que el hilo se despierte pero que alguien más se apropie del recurso antes de que se ejecute el hilo.

La necesidad de poder despertar a todos los hilos en vez de uno, se demuestra en el problema de los lectores y los escritores. Al terminar un escritor, puede adoptar por despertar a todos los escritores pendientes o a todos los lectores pendientes. Si elige los lectores, debe despertarlos a todos y no a uno. La flexibilidad necesaria se obtiene mediante las primitivas de hilos para despertar con exactitud un hilo o despertarlos a todos.

El código de un hilo consta por lo general de varios procedimientos, al igual que un proceso. Puede tener varias variables locales, variables globales y parámetros del procedimiento.

Las variables locales y los parámetros no provocan problema alguno, pero las variables globales de un hilo que no son globales en todo el programa pueden provocar ciertas dificultades.

Sin embargo, el acceso a las variables globales particulares es un tanto truculento, puesto que la mayoría de los lenguajes de programación tiene forma de expresar las variables locales y las globales, pero no tienen formas intermedias. Es posible asignar un bloque de memoria a las variables globales y transferirlo a cada procedimiento del hilo, como un parámetro adicional. Aunque esta no es una solución elegante, funciona.

En otra alternativa, se pueden introducir nuevos procedimientos de biblioteca para crear, dar valores y leer estas variables globales a lo largo de todo un hilo. La primera llamada se parecería a:

CREATE\_GLOBAL(“BUFPTR”);

Asignan un espacio de almacenamiento a un apuntador llamado BUFPTR en una pila o un área especial de almacenamiento, reservado para el hilo que hizo la llamada. Sin importar la posición del espacio de almacenamiento, el hilo que llamo es el único que tiene acceso a la variable global.

Si otro hilo crea una variable global con el mismo nombre, obtiene un espacio de almacenamiento distinto, que no entra en conflicto con el ya existente.

Se necesitan dos llamadas para tener acceso a las variables globales; una para escribirlas y otras para leerlas. Para escritura, funciona algo como:

SET\_GLOBAL(“BUFPTR”, &BUF);

Guarda el valor de un apuntador en el espacio de almacenamiento creado en la llamada a CREATE\_GLOBAL. Para leer una variable global, la llamada sería algo así como:

BUFPTR=READ\_GLOBAL(“BUFPTR”);

Esta llamada regresa a dirección almacenada en el variable global, de modo que se pueda tener el acceso al valor del dato.

Nuestro último aspecto del diseño relativo a los hilos es la planificación. Los hilos se pueden planificar mediante distintos algoritmos, entre los que se encuentran la prioridad, round robin y otros. Los paquetes de hilos proporcionan a menudo ciertas llamadas para que el usuario pueda especificar el algoritmo de planificación y establecer las prioridades, en su caso.

Modelos de procesadores.

En un sistema distribuido con varios procesadores, un aspecto fundamental en el diseño es como se utiliza a los procesadores que se pueden organizar de varias formas:

* De estación de trabajo.
* De pila de procesadores.
* Hibrido

Estación de trabajo.

Computadoras dispersas, pero conectados entre si mediante una red local.

Algunas estaciones de trabajo tienen lugares fijos, como en una oficina, cada una de ellas asignadas a un único usuario. Algunas otras estaciones de trabajo. Pueden estar en lugares de acceso público y ser utilizadas por diferentes usuarios.

Las estaciones de trabajo pueden contar o no con discos duro en cada una de ellos.

Si las estaciones de trabajo carecen de disco el sistema de archivos puede ser implementado mediante servidores de archivos.

Las estaciones de trabajo carecen de disco el sistema de archivos puede ser implementado mediante servidores de archivos.

Las estaciones de trabajo sin discos fueron populares principalmente por su precio y por su fácil mantenimiento cuando se libera una nueva actualización de algún software, los administradores únicamente los instalan en los servidores.

Los respaldos de información y el mantenimiento son también más sencillos.

Por último, dada que todos los archivos de trabajo se encuentran en el servidor.

Si las estaciones de trabajo tienen sus propios discos.

* Paginación y archivos temporales.
* Paginación, archivos temporales y archivos ejecutables.
* Paginación, archivos temporales y archivos ejecutables y ocultamente.

Paginación y archivos temporales.

Se basa en la idea de que aunque es conveniente mantener todos los archivos de usuario en servidores centrales para facilitar el respaldo y el mantenimiento, también se necesita archivos temporales, no compartidos y que se eliminan al término de la sección.

Paginación, archivos temporales y archivos ejecutables:

* Los discos locales contienen a los archivos ejecutables (aplicación) con el propósito de reducir la carga en la red.
* Los archivos ejecutables pocas veces son modificados y cuando eso sucede basta con transmitir la nueva versión a todas las computadoras.

Paginación archivos temporales y ejecutables y ocultamiento.

* Consiste en utilizar los discos locales como cache y ya que se puede cargar archivos desde el servidor para leerlos y escribir en ellos de manera local y traspasarlos de nuevo al servidor al término de la sección.
* Una desventaja es el tener consistentes todos los caches ¿que pasa si dos usuarios modifican al mismo tiempo el mismo archivo?
* Este problema no es fácil de resolver y requiere un análisis profundo.

Un sistema local y completo de archivos.

Cada computadora puede tener sus propios sistemas de archivos con la posibilidad de tener acceso a los sistemas de archivos de otras computadoras.

La desventaja aquí es que es difícil compartir algunos aspectos y el sistema resultante se parece más aun sistema operativo de red que un sistema distribuido.

Las desventajas del modelo de estación de trabajo son variedades:

* El modelo es fácil de comprender los usuarios tienen una cantidad fija de capacidad de procesamiento exclusiva con lo que tiene un tiempo de respuesta garantizado.
* Cada usuario tiene cierto grado de autonomía y puede asignar los recursos de su computador como juzgue necesario.
* Los discos locales favorecen la autonomía que permite seguir trabajando aunque los servidores fallen.
* El modelo de estación de trabajo tiene dos problemas principales.

Si los microprocesadores siguen haciéndose mas baratos prontos será posible dotar de computadoras de gran potencia a cada usuario haciendo innecesariamente la distribución de las tareas.

Gran parte del tiempo los usuarios no utilizan las estaciones del trabajo, las cuales permanecen inactivas, mientras que algún usuario quizá necesite esa capacidad de computo adicional y no pueda obtenerla.

Actividad pag. 205.

1. ¿Cómo encontrar una estación de trabajo?

Los algoritmos que se utilizan para localizar para localizar las estaciones de trabajo inactivas se pueden dividir en dos categorías: controlados por el servidor y controlados por el cliente.

1. ¿Cómo lograr que un proceso remoto se efectué de forma transparente?

Se necesita la misma ubicación del sistema de archivos, el mismo directorio de trabajo y los mismos variables, del ambiente si que existe.

1. ¿Cómo ocurre si regresa el poseedor de la maquina?

Lo mas fácil es no hacer, nada pero esto tiende a destruir la idea de las estaciones de trabajo “personalmente” si otros personas intentan ejecutar programas en sus estaciones de trabajo el mismo tiempo que usted desea ejecutarla, se dice la respuesta garantizada.

MODELO DE PILA DE PROCESADORES.

Este método consiste en construir una pila de procesadores, repleta de CPU, los cuales se pueden asignar de manera dinámica a los usuarios según la demanda.

En lugar de darle a cada usuario computadoras personales se le da una terminal grafica de alta calidad y alto rendimiento, como las terminales x.

Este método es parecido al tiempo compartido tradicional pero que se construye con tecnología moderna.

Su motivación consiste en dar un paso más adelante en el uso de las estaciones de trabajo sin disco:

* Si el sistema de archivos se puede concentrar en los servidores por qué no hacer lo mismo con los CPU, pues al colocarlos todos en el mismo gabinete se pueden reducir los costos de consumo eléctrico y mantenimiento.
* El modelo también facilita el crecimiento por incrementos. Si la carga de cómputo se incrementa en un 10% se adquiere únicamente los procesadores suficientes para cubrir esa demanda.
* Los usuarios utilizan tantos CPU (procesadores) como les sean necesarios. Lo utilizan por periodos cortos, después los liberan y quedan de nuevo disponibles a la pila.
* No existe el concepto de propiedad.
* El principal argumento para la centralización del poder de cómputo como pila de procesadores proviene de la teoría de colas (estructura de datos).
* Un sistema de colas se tiene en una situación donde los usuarios generan solicitudes de trabajo a un servidor.
* Cuando el servidor está ocupado las solicitudes se forman y se van procesando según su turno.
* Algunos ejemplos de colas de situaciones reales son la fila de una taquilla, la pila de acceso a un evento, ordenar alimentos en algún restaurant.
* El concepto del uso de las estaciones de trabajo inactivas es un débil intento por recuperar tiempo de procesamiento pero es complejo y tiene muchos problemas.
* El modelo de pila de procesadores es una forma más limpia de obtener poder de cómputo adicional que la búsqueda de estaciones inactivas.
* Ya que ningún procesador pertenece a alguien, ni hay maquina origen, no hay peligro de que el poseedor del equipo regrese.
* Seleccionar pila de procesadores o estaciones de trabajo inactivas va a depender del trabajo que se desarrolle.

Modelo de procesador hibrido.

Se puede establecer una mediación al proporcionar a cada usuario una “estación de trabajo” personal y además tener una pila de procesadores. Aunque esta solución es más cara.

Para procesos interactivos sería mejor utilizar estaciones de trabajo, con una respuesta garantizada.

Sin embargo las estaciones inactivas no se utilizan, lo cual hace más sencillo al diseño del sistema. Solo se dejan sin utilizar.