MÓDULO 2 MF0964_3: DESARROLLO DE ELEMENTOS DE SOFTWARE PARA GESTIÓN DE SISTEMAS

UNIDAD 3 UF1288: DESARROLLO DE COMPONENTES SOFTWARE PARA SERVICIOS DE COMUNICACIONES

1 Programación concurrente

core networks

1 Programación concurrente

1.1 Introducción

core networks

Computación concurrente

La computación concurrente es la simultaneidad en la ejecución de múltiples tareas interactivas. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa.

Las tareas se pueden ejecutar en una sola unidad central de proceso (multiprogramación), en varios procesadores o en una red de computadores distribuidos.

1 Programación concurrente

1.2 Programación de procesos e hilos de ejecución

core networks

Hilos (threads)

Un hilo o proceso ligero es una unidad básica de ejecución, con su propio: contador de programa registros de CPU pila (stack).

Los hilos dentro de una misma aplicación comparten: código y datos recursos del S.O. (ficheros, E/S, etc.).

La creación de un nuevo hilo es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

Hilos (threads) (II)

Un hilo es básicamente una tarea que puede ser ejecutada en paralelo con otra tarea.

Los hilos de ejecución que comparten los mismos recursos, sumados a estos recursos, son en conjunto conocidos como un proceso. El hecho de que los hilos de ejecución de un mismo proceso compartan los recursos hace que cualquiera de estos hilos pueda modificar éstos. Cuando un hilo modifica un dato en la memoria, los otros hilos acceden a ese dato modificado inmediatamente.

Lo que es propio de cada hilo es el contador de programa, la pila de ejecución y el estado de la CPU (incluyendo el valor de los registros).

Hilos (threads) (III)

El proceso sigue en ejecución mientras al menos uno de sus hilos de ejecución siga activo. Cuando el proceso finaliza, todos sus hilos de ejecución también han terminado. Asimismo en el momento en el que todos los hilos de ejecución finalizan, el proceso no existe más y todos sus recursos son liberados.

Algunos lenguajes de programación tienen características de diseño expresamente creadas para permitir a los programadores lidiar con hilos de ejecución. Otros (la mayoría) desconocen la existencia de hilos de ejecución y éstos deben ser creados mediante llamadas de biblioteca especiales que dependen del sistema operativo en el que estos lenguajes están siendo utilizados.

1 Programación concurrente

1.3 Programación de eventos asíncronos

core networks

Programación secuencial

Tradicionalmente, el modelo de ejecución que utilizan la mayoría de los lenguajes y paradigmas de programación se corresponde con un tratamiento secuencial del cuerpo del programa.

Un programa es entendido como una secuencia de instrucciones que se ejecutan ordenadamente y donde la ejecución de cada operación no da comienzo hasta que no termina la anterior.

Programación asíncrona

La programación asíncrona establece la posibilidad de hacer que algunas operaciones devuelvan el control al programa llamante antes de que hayan terminado mientras siguen operando en segundo plano.

Esto agiliza el proceso de ejecución y en general permite aumentar la escalabilidad, pero complica el razonamiento sobre el programa.

Modelos de programación asíncrona

- Modelo de paso de continuadores. Cada función recibe información acerca de cómo debe tratar el resultado –de éxito o error– de cada operación.
- Modelo de eventos. Se utiliza una arquitectura dirigida por eventos que permite a las operaciones no bloqueantes informar de su terminación mediante señales de éxito o fracaso.
- Modelo de promesas. Se razona con los valores de retorno de las operaciones no bloqueantes de manera independiente del momento del tiempo en que dichos valores –de éxito o fallo– se obtengan.
- Modelo de generadores. Se utilizan generadores para devolver temporalmente el control al programa llamante y retornar en un momento posterior a la rutina restaurando el estado en el punto que se abandonó su ejecución.

1 Programación concurrente

1.4 Mecanismos de comunicación entre procesos

core networks

Comunicación entre procesos

La comunicación entre procesos (comúnmente IPC, del inglés Inter-Process Communication) es una función básica de los sistemas operativos. Los procesos pueden comunicarse entre sí a través de compartir espacios de memoria, ya sean variables compartidas o buffers, o a través de las herramientas provistas por las rutinas de IPC.

La IPC provee un mecanismo que permite a los procesos comunicarse y sincronizarse entre sí, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la red subyacente.

Tipos de comunicación entre procesos

La comunicación puede ser:

- Síncrona o asíncrona
- Persistente (persistent) o momentánea (transient)
- Directa o indirecta
- Simétrica o asimétrica
- Con uso de buffers explícito o automático
- Envío por copia del mensaje o por referencia
- Mensajes de tamaño fijo o variable

1 Programación concurrente

1.5 Sincronización

core networks

Sincronización

En muchos casos, los procesos se reúnen para realizar tareas en conjunto, a este tipo de relación se le llama procesos cooperativos. Para lograr la comunicación, los procesos deben sincronizarse, de no ser así pueden ocurrir problemas no deseados.

La sincronización es la transmisión y recepción de señales que tiene por objeto llevar a cabo el trabajo de un grupo de procesos cooperativos.

Sincronización (II)

Para que los procesos puedan sincronizarse es necesario disponer de servicios que permitan bloquear o suspender bajo determinadas circunstancias la ejecución de un proceso. Los principales mecanismos de sincronización que ofrecen los sistemas operativos son:

- Señales.
- Tuberías.
- Semáforos.
- Mutex y variables condicionales.
- Paso de mensajes.

1 Programación concurrente

1.6 Acceso a dispositivos

core networks

Puertos lógicos

El Puerto Lógico es una zona, o localización, de la memoria de un ordenador que se asocia con un puerto físico o con un canal de comunicación, y que proporciona un espacio para el almacenamiento temporal de la información que se va a transferir entre la localización de memoria y el canal de comunicación.

Un puerto lógico es una salida de bits, que pueden ser 1 o 0, o sea, un puerto es el valor que se usa en el modelo de la capa de transporte para distinguir entre las múltiples aplicaciones que se pueden conectar al mismo host, o puesto. Entonces un puerto lógico de Internet es una interface de software que permitirá el ingreso y salida de data por aplicaciones que usan Internet.

Puertos lógicos (II)

Los puertos en general tienen un rango que van de 0 hasta 65534, y son divididos en tres categorías:

Well-known port: 0-1013

Registered ports: 1024-49151

Dynamic and private ports: 49152-65534

Puertos lógicos (III)

Well-known port: Aquí se encuentra el rango de puerto usado por las aplicaciones más comunes como el FTP (20-21), HTTP (80), SSH (22), TELNET (23), etc.

Registered ports: Los puertos de registro son asignados por IANA o ICANN, los cuales hacen referencia a una determinada aplicación o protocolo. Un ejemplo clásico es el uso de webmin que utiliza el puerto 10000 para su conexión

Dynamic and private ports: Son los puertos que están disponibles para cualquier aplicación a utilizar en la comunicación.

Recurso

https://es.wikipedia.org/wiki/Anexo:N%C3%BAmeros_de_puertos_de_red

1 Programación concurrente

1.7 Resumen

core networks

Resumen

- Computación concurrente.
- Hilos.
- Programación asíncrona.
- Comunicación entre procesos.
- Sincronización.
- Puertos lógicos.