NOMBRE DE ALUMNO: VICTOR HUGO LÓPEZ MORENO

NOMBRE DEL PROFESOR (A): ANDRÉS ALEJANDRO REYES MOLINA

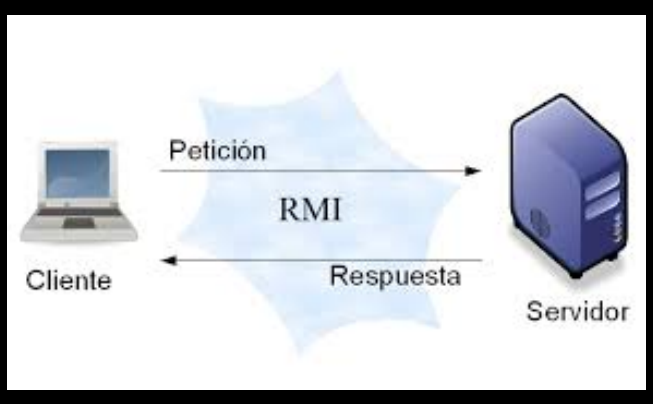
NOMBRE DEL TRABAJO: SUPERNOTA

MATERIA: SISTEMAS OPERATIVOS DISTRIBUIDOS

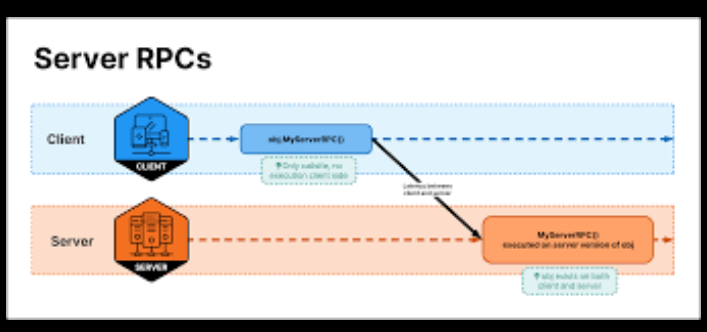
GRADO: 6°

3.1. INTRODUCCIÓN.

Es aquel que está gestionado por un servidor y sus clientes invocan sus métodos utilizando un "método de invocación remota". El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y el resultado se devuelve al cliente en otro mensaje. Tecnologías orientadas a los objetos distribuidos: 1. RMI.- Remote Invocation Method (Sistema de Invocación Remota de Métodos) : Fue el primer fremework para crear sistemas distribuidos de Java. Este sistema permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje. 2. DCOM.- Distributed Component Object Model: El Modelo de Objeto Componente Distribuido, está incluido en los sistemas operativos de Microsoft. Es un juego de conceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios objetos del programa servidores, en otros ordenadores dentro de una red. Esta tecnología está asociada a la plataforma de productos Microsoft.



3.2. RPC.



En computación distribuida, la llamada a procedimiento remoto (en inglés, Remote Procedure Call, RPC) es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas. El protocolo que se utiliza para esta llamada es un gran avance sobre los sockets de Internetusados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando estas encapsuladas dentro de las RPC. Las RPC son muy utilizadas dentro de la comunicación cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando este de vuelta el resultado de dicha operación al cliente.

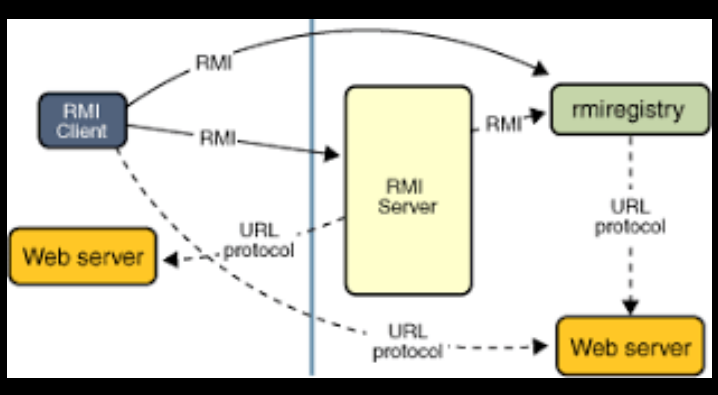
3.3. EJECUCION RPC EN C#

RPC usa un enfoque de contrato primero para el desarrollo de API. Los búferes de protocolo (Protobuf) se usan de forma predeterminada como el lenguaje de definición de interfaz (IDL). El archivo *.proto contiene: La definición del servicio gRPC Los mensajes enviados entre clientes y servidores Para más información sobre la sintaxis de los archivos de protobuf, consulte Creación de mensajes de Protobuf para aplicaciones .NET. Vamos a considerar, por ejemplo, el archivo greeter.proto que se usa en Introducción a un servicio gRPC: Define un servicio Greeter. El servicio Greeter define una llamada a SayHello. SayHello envía un mensaje HelloRequest y recibe un mensaje HelloReply:

ProtoBuf

```
syntax = "proto3";

option csharp_namespace = "GrpcGreeter";

package greet;

// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply);
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings.
message HelloReply {
  string message = 1;
}
```

3.4. RMI



RMI (Java Remote Method Invocation) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI. RMI se caracteriza por la facilidad de su uso en la programación por estar específicamente diseñado para Java; proporciona paso de objetos por referencia (no permitido por SOAP), recolección de basura distribuida (Garbage Collector distribuido) y paso de tipos arbitrarios (funcionalidad no provista por CORBA).

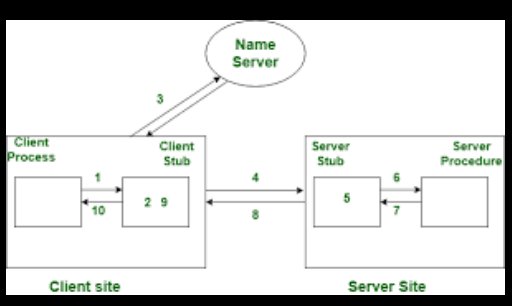
A través de RMI, un programa Java puede exportar un objeto, con lo que dicho objeto estará accesible a través de la red y el programa permanece a la espera de peticiones en un puerto TCP. A partir de ese momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

3.5. EJECUCION EN JAVA DE RMI.

Enviar mensajes no tiene gran utilidad, ¿no sería mejor poder llamar a algunas funciones en el servidor? esto es justamente lo que hace RMI en Java. Declaramos el servidor, el cliente, y la interfaz que servirá como "pegamento" entre estos dos y que se encargará del paso de parámetros. Después, desde el cliente llamamos a las funciones declaradas anteriormente. De esta manera, todos los métodos se ejecutarán en el servidor. En este caso haremos una calculadora, para no hacer el post muy largo, pero podemos hacer miles de cosas; se me ocurre, por ejemplo, conectar a una base de datos. Interfaz remota Comencemos con la interfaz remota, como lo dije hace un momento esta servirá como pegamento. Veamos una cosa interesante, como se puede ver dice "public interface" en lugar de "public class". Esto es porque, aunque suene redundante, es una interfaz y recordemos que éstas sólo sirven para ser sobrescritas más tarde. ¿Y en dónde las sobrescribiremos? en el código del servidor.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

/*
        Declarar firma de métodos que serán sobrescritos
*/
public interface Interfaz extends Remote {
    float sumar(float numero1, float numero2) throws RemoteException;
    float restar(float numero1, float numero2) throws RemoteException;
    float multiplicar(float numero1, float numero2) throws RemoteException;
    float dividir(float numero1, float numero2) throws RemoteException;
}
```

3.6. DIFERENCIA ENTRE RPC Y RMI

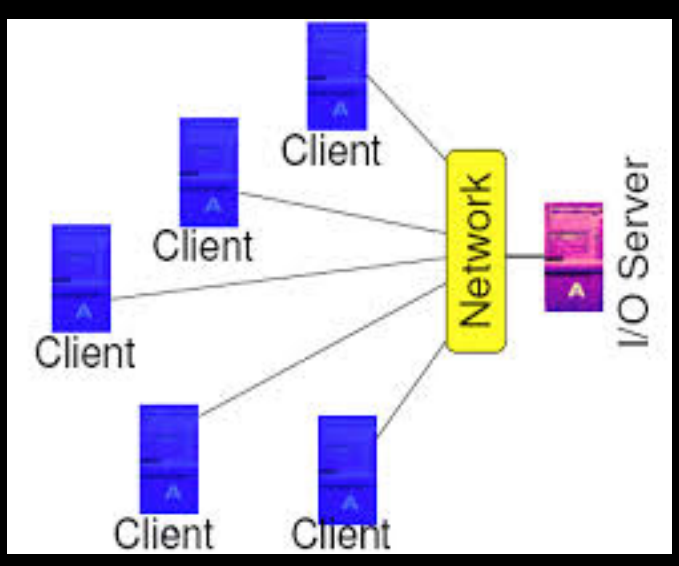


RPC y RMI son los mecanismos que permiten a un cliente invocar el procedimiento o método desde el servidor a través del establecimiento de la comunicación entre el cliente y el servidor. La diferencia común entre RPC y RMI es que RPC solo admite la programación de procedimientos, mientras que RMI admite la programación orientada a objetos. Otra diferencia importante entre los dos es que los parámetros pasados a la llamada a procedimientos remotos consisten en estructuras de datos ordinarias. Por otro lado, los parámetros pasados al método remoto consisten en objetos.
Diferencias clave entre RPC y RMI RPC admite los paradigmas de programación de procedimientos, por lo tanto, está basado en C, mientras que RMI admite paradigmas de programación orientados a objetos y está basado en Java. Los parámetros pasados a procedimientos remotos en RPC son las estructuras de datos ordinarias. Por el contrario, RMI transita objetos como un parámetro al método remoto. RPC se puede considerar como la versión anterior de RMI, y se usa en los lenguajes de programación que admiten la programación de procedimientos, y solo se puede usar el método de paso por valor.

3.7. SOPORTE DEL SISTEMA OPERATIVO.

El sistema operativo es básicamente un programa que controla los recursos del computador, proporciona servicios a los programadores y planifica la ejecución de otros programas. A partir de los µP de 16 bits, las CPU incorporan estructuras de apoyo a los sistemas operativos, por lo que resulta interesante una introducción a dos de las funciones básicas del SO que más inciden en la arquitectura de la CPU: la multiprogramación (o multitarea) y el control de memoria. MULTIPROGRAMACIÓN. La multiprogramación es la tarea central de lossistemas operativos modernos. Permite que múltiples programas de usuario o usuarios que se hallan en memoria se alternen entre la utilización de la CPU y los accesos a I/O, de manera que el procesador siempre se mantenga ocupado con un proceso mientras los demás esperan. PLANIFICACION (SCHEDULING) DE ALTO NIVEL. Determina qué programas son admitidos por el sistema para ser procesados. El planificador (proyecta) controla pues el grado de multiprogramación (número de procesos en memoria). Una vez admitido, un programa se convierte en un proceso y es añadido a la cola para ser tratado por el distribuidor. El planificador de alto nivel puede limitar el grado de multiprogramación para dar un servicio satisfactorio al conjunto actual de procesos

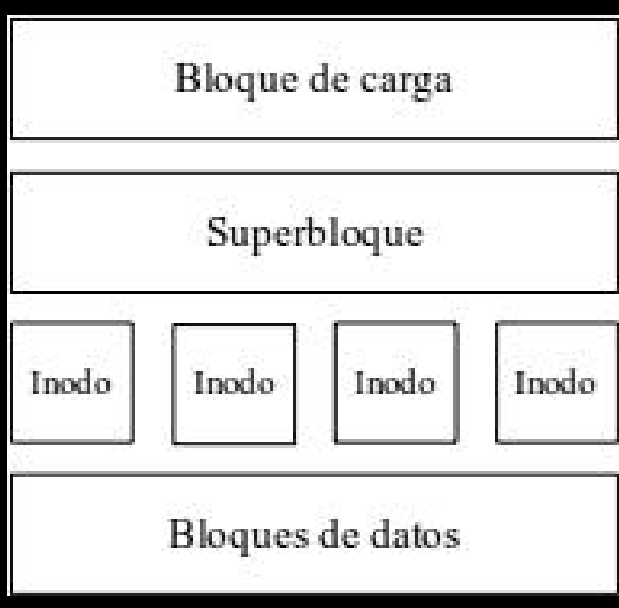
3.8. SISTEMAS DE ARCHIVOS DISTRIBUIDOS.



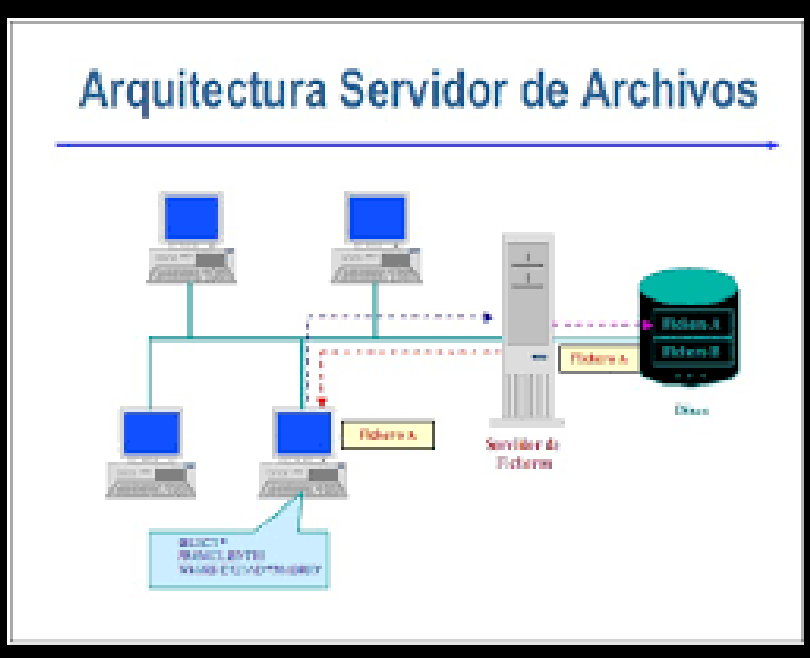
Un sistema de archivos distribuido nos va a permitir almacenar y acceder a archivos remotos como si fueran locales, sin que notemos pérdidas en el rendimiento. Vamos a analizar dos sistemas de archivos distribuidos muy conocidos (NFS y AFS), comentando sus principales ventajas e inconvenientes. Para poder montar nuestro propio sistema distribuido, antes hay que tener claro cuál vamos a usar. Se comentarán sobre todo 2 aspectos fundamentales: el primero la consistencia, es decir, diferentes máquinas que quieran acceder a un mismo archivo, deben de ver el mismo contenido en él, a pesar de que otro usuario haya realizado modificaciones. El segundo será el rendimiento. Si se desea un mayor nivel de consistencia, el rendimiento se verá penalizado

3.9. CARACTERÍSTICAS DE LOS SISTEMAS DE ARCHIVOS.

Tu disco duro externo, el disco duro interno de tu ordenador, un USB o una tarjeta SD. Todos ellos son unidades de almacenamiento, lo que quiere decir que cuando los formateas estás creando la infraestructura en las que van a alojarse los datos que después le quieras meter. Es aquí donde entra en juego el sistema de archivos, un componente del sistema operativo que se encarga de administras la memoria de cada unidad. Se encargan de asignarle a los archivos el espacio que necesiten, ordenarlos, permitir el acceso a ellos y administrar el espacio libre de las unidades de almacenamiento. Es como un bibliotecario, que ordena y registra la posición exacta en la que se ha escrito un fichero dentro de la unidad, y así tu sistema operativo puede acceder rápidamente a ellos y saber dónde empieza y acaba cada uno. Siguiendo con la analogía bibliotecaria, de la misma manera que cada bibliotecario puede tener su método para organizar los libros, cada sistema de archivo hace lo mismo, organizando y gestionando los datos de maneras diferentes.



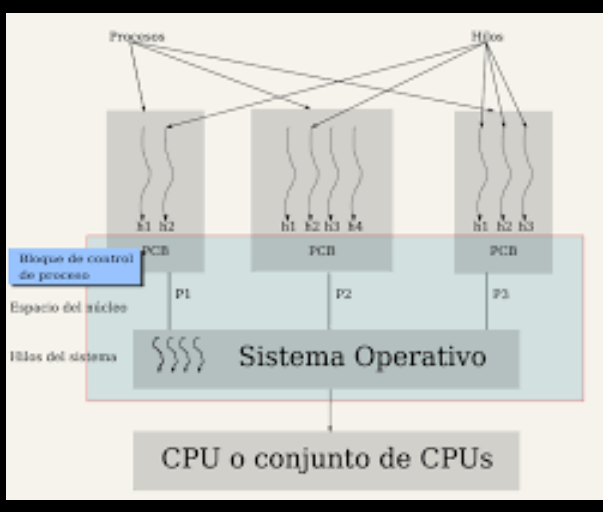
3.10. ARQUITECTURA DEL SERVICIO DE ARCHIVO



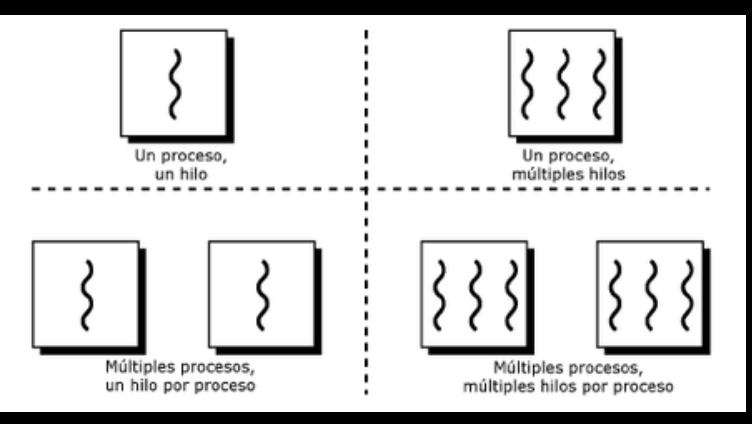
La computación desde sus inicios ha sufrido muchos cambios, desde los grandes ordenadores que permitían realizar tareas en forma limitada y de uso un tanto exclusivo de organizaciones muy selectas, UNIVERSIDAD DEL SURESTE 111 hasta los actuales ordenadores ya sean personales o portátiles que tienen las mismas e incluso mayores capacidades que los primeros y que están cada vez más introducidos en el quehacer cotidiano de una persona. Los mayores cambios se atribuyen principalmente a dos causas, que se dieron desde las décadas de los setenta: * El desarrollo de los microprocesadores, que permitieron reducir en tamaño y costo a los ordenadores y aumentar en gran medida las capacidades de los mismos y su acceso a más personas. * El desarrollo de las redes de área local y de las comunicaciones que permitieron conectar ordenadores con posibilidad de transferencia de datos a alta velocidad.

3.11. PROCESOS E HILOS

En un sistema multiprogramado o de tiempo compartido, un proceso es la imagen en memoria de un programa, junto con la información relacionada con el estado de su ejecución. Un programa es una entidad pasiva, una lista de instrucciones; un proceso es una entidad activa, que empleando al programa– define la actuación que tendrá el sistema. En contraposición con proceso, en un sistema por lotes se habla de tareas. Una tarea requiere mucha menos estructura, típicamente basta con guardar la información relacionada con la contabilidad de los recursos empleados. Una tarea no es interrumpida en el transcurso de su ejecución. Ahora bien, esta distinción no es completamente objetiva y se pueden encontrar muchos textos que emplean indistintamente una u otra nomenclatura. Si bien el sistema brinda la ilusión de que muchos procesos se están ejecutando al mismo tiempo, la mayor parte de ellos típicamente está esperando para continuar su ejecución en un momento determinado sólo puede estar ejecutando sus instrucciones un número de procesos igual o menor al número de procesadores que tenga el sistema.



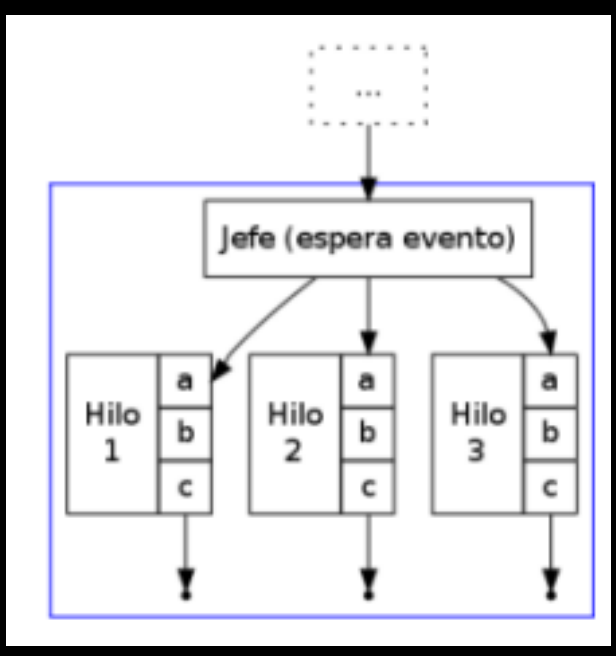
3.12. LOS HILOS Y EL SISTEMA OPERATIVO



Como se vio, la cantidad de información que el sistema operativo debe manejar acerca de cada proceso es bastante significativa. Si cada vez que el planificador elige qué proceso pasar de Listo a En ejecución debe considerar buena parte de dicha información, la simple transferencia de todo esto entre la memoria y el procesador podría llevar a un desperdicio burocrático2 de recursos. Una respuesta a esta problemática fue la de utilizar los hilos de ejecución, a veces conocidos como procesos ligeros (LWP, Lightweight processes). Cuando se consideran procesos basados en un modelo de hilos, se puede proyectar en sentido inverso que todo proceso es como un solo hilo de ejecución. Un sistema operativo que no ofreciera soporte expreso a los hilos los planificaría exactamente del mismo modo.

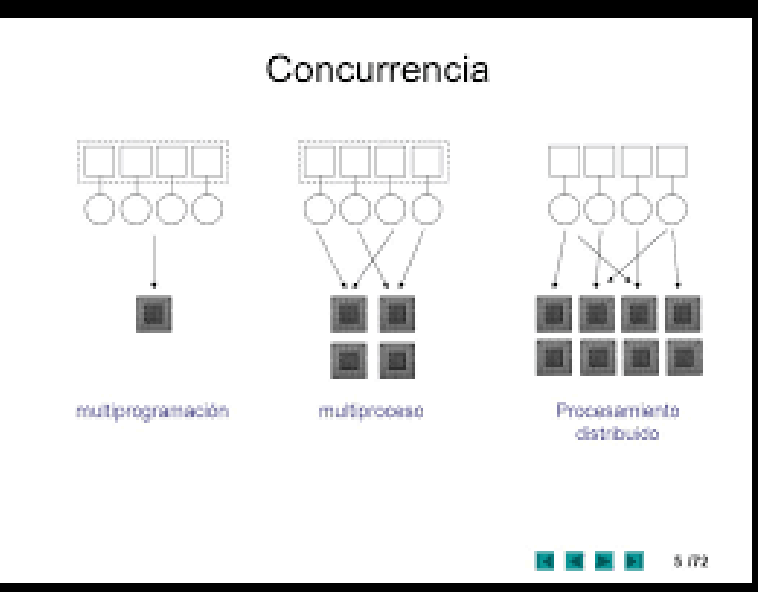
3.13. PATRONES DE TRABAJO CON HILOS

Un hilo tiene una tarea distinta de todos los demás: el hilo jefe genera o recopila tareas para realizar, las separa y se las entrega a los hilos trabajadores. Este modelo es el más común para procesos que implementan servidores (es el modelo clásico del servidor Web Apache) y para aplicaciones gráficas (GUI), en que hay una porción del programa (el hilo jefe) esperando a que ocurran eventos externos. El jefe realiza poco trabajo, se limita a invocar a los trabajadores para que hagan el trabajo de verdad; como mucho, puede llevar la contabilidad de los trabajos realizados. Típicamente, los hilos trabajadores realizan su operación, posiblemente notifican al jefe de su trabajo, y finalizan su ejecución.



Equipo de trabajo Al iniciar la porción multihilos del proceso, se crean muchos hilos idénticos, que realizarán las mismas tareas sobre diferentes datos. Este modelo es frecuentemente utilizado para cálculos matemáticos (p. ej.: criptografía, render, álgebra lineal).

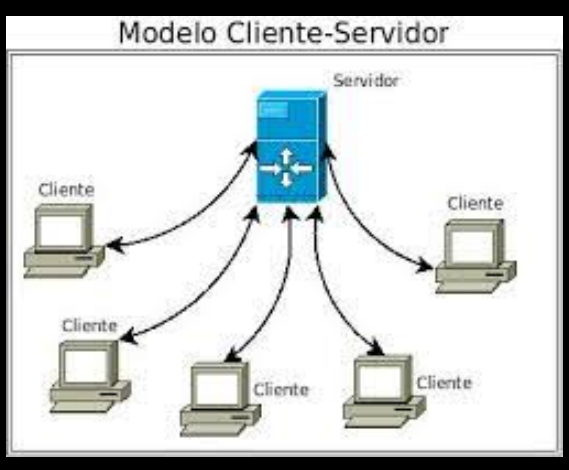
3.14. CONCURRENCIA



Desde un punto de vista formal, la concurrencia no se refiere a dos o más eventos que ocurren a la vez sino a dos o más eventos cuyo orden es no determinista, esto es, eventos acerca de los cuales no se puede predecir el orden relativo en que ocurrirán. Si bien dos procesos (o también dos hilos) completamente independientes entre sí ejecutándose simultáneamente son concurrentes, los temas que en la presente sección se expondrán se ocupan principalmente de procesos cuya ejecución está vinculada de alguna manera (p. ej.: dos procesos que comparten cierta información o que dependen uno del otro). Aunque una de las tareas principales de los sistemas operativos es dar a cada proceso la ilusión de que se está ejecutando en una computadora dedicada, de modo que el programador no tenga que pensar en la competencia por recursos, a veces un programa requiere interactuar con otros: parte del procesamiento puede depender de datos obtenidos en fuentes externas, y la cooperación con hilos o procesos externos es fundamental.

3.15. MECANISMOS DE SINCRONIZACIÓN

En la presente sección se enumeran los principales mecanismos que pueden emplearse para programar considerando a la concurrencia: candados, semáforos y variables de condición. Regiones de exlcusión mutua: candados o mutexes Una de las alternativas que suele ofrecer un lenguaje concurrente o sistema operativo para evitar la espera activa a la que obliga el algoritmo de Peterson (o similiares) se llama mutex o candado (lock). La palabra mutex nace de la frecuencia con que se UNIVERSIDAD DEL SURESTE 121 habla de las regiones de exclusión mutua (mutual exclusion). Es un mecanismo que asegura que cierta región del código será ejecutada como si fuera atómica. Sincronización: la exclusión de las secciones críticas entre varios procesos se protegen por medio de regiones de exclusión mutua. La figura ilustra varios procesos que requieren de una misma sección crítica; las flechas punteadas representan el tiempo que un proceso está a la espera de que otro libere el paso para dicha sección, y las punteadas verticales indican la transferencia del candado.



REFERENCIAS

TODA LA INFORMACION DE ESTE TRABAJO SE TOMO DE LA ANTOLOGÍA CORRESPONDIENTE A LA MATERIA.