TEMA 1
INTRODUCCIÓN

1.1 QUE ES UN SISTEMA OPERATIVO

Es un grupo de programas de proceso con las rutinas de control necesarias para mantener continuamente operativos dichos programas.

Objetivo: Optimizar todos los recursos del sistema. El sistema operativo es el programa fundamental.
Programas de sistema: Controlan la operación de la computadora.
Programas de aplicación: Resuelven problema usuarios.
El s. O. Protege y libera a los programadores de la complejidad del hardware, controlar todas las partes del sistema, presentar maquina virtual.
Microprograma: Es un software que se localiza en la memoria de solo lectura. Busca las instrucciones de lenguaje de maquina para ejecutarlas como una serie de pequeños pasos, define lo que es el lenguaje de maquina.
Lenguaje de maquina: Posee entre 50 y 300 instrucciones, (desplazar datos, hacer operaciones aritméticas y comparar valores.) El S.O. Se ejecuta en modo central o modo de supervisión, Los compiladores, editores y demás programas se ejecutan en modo usuario.
Principales características de los s. O.:

1. Definir la "interfaz del usuario".
2. Compartir el hardware y los datos entre usuarios.
3. Planificar recursos entre usuarios
4. Facilitar la e/s.
5. Recuperarse de los errores.

Principales recursos administrados por los s. O.:
*Procesadores. *Almacenamiento. *Dispositivos de e/s. *Datos.
Los s. O. Son una interfaz con:
*Operadores. *Programadores *Hardware. *Usuarios.

1.2 HISTORIA DE LOS S.O – GENERACIONES

Primera generación cero (década del 1940):
*Carencia de S.O. *Completo acceso al lenguaje de maquina.
Primera generación (1945-1955): bulbos y conexiones:
*Carencia de s. O. *Transición entre trabajos, haciendo la misma mas simple.
Segunda generación (1955-1965): transistores y sistemas de procesamiento por lotes (batch):
Multiprogramación: varios programas de usuarios en el almacenamiento principal, cambiando el procesador
Multiprocesamiento: varios procesadores. Independencia de dispositivo
Tercera generación (1965-1980): circuitos integrados y multiprogramación:

- Difusión de la multiprogramación:
- Partición de la memoria.
- Aprovechamiento del tiempo de espera.
- Aparición de técnicas de spooling (operación simultanea y en línea de periféricos)
- Almacenamiento de trabajos de e/s en discos
- Son sistemas de modos múltiples y de propósitos generales
- Aparecen los lenguajes de control de trabajos
- Soportan timesharing (tiempo compartido)
- Aparecen los sistemas de tiempo real

Cuarta generación (1980-1990): computadoras personales:
Software amigable con el usuario y desarrollo de sistemas operativos de red y sistemas operativos distribuidos.
S.O de red: usuarios conscientes de varias computadoras conectadas donde C/maquina ejecuta su propio S.O. Local. Son similares a los s. O. De un solo procesador
S.O distribuidos: Aparece como un s. O. De un solo procesador. Donde los usuarios no son conscientes del lugar donde se ejecutan sus programas o donde se encuentran sus archivos. Un programa se puede ejecutar en varios procesadores a la vez, maximizando el paralelismo. Además aparecen los emuladores de terminal para el acceso a equipos remotos desde computadoras personales.

1.3 CONCEPTOS DE LOS SISTEMAS OPERATIVOS

Las llamadas al sistema: es la interfaz entre el S.O. y los programas (**instrucciones ampliadas)**
Procesos: es un programa en ejecución que consta del programa ejecutable, sus datos, pila y contador y otros registros. La información de control se almacena en la **tabla de procesos**. Un proceso (suspendido) consta de un espacio de dirección y los datos de la tabla.
Archivos: los S.O. deben brindar independencia de dispositivos. La forma de agrupar archivos es mediante

"directorios jerárquicos", donde cada archivo tiene una ruta de acceso.
Llamada al sistema: Permiten a los programas comunicarse con el s. O. Y solicitarle servicios. A cada llamada le corresponde un procedimiento:

- instrucción tipo "trap".
- s. O. Recupera el control y ejecuta el trabajo solicitado.
- el s. O. Coloca un código de estado y ejecuta "return from trap"
- El procedimiento regresa

1.4 ESTRUCTURA DE LOS SISTEMAS OPERATIVOS (organización interna)

Sistemas monolíticos: Es muy común: no existe estructura, El s. O. Es una colección de procedimientos que se pueden llamar entre si.
Sistemas con capas: Consiste en organizar el S.O. Como una jerarquía de capas:

Estructura en Capas
5- Operador
4- Programas de usuario
3- Control de E/S
2- Comunicación Operador – Proceso
1- Administración de la Memoria y del Disco
0- Asignación del Procesador y Multiprogramación

Máquinas virtuales: existe un elemento central ➔**monitor de la maquina virtual.** Pueden ejecutar cualquier S.O., cada máquina con distintos S.O. y soportan periféricos virtuales.

- **Modelo cliente-servidor**: una tendencia es mover el código a capas superiores y mantener un "Núcleo mínimo". El núcleo controla la comunicación entre los clientes y los servidores. Además se fracciona el S.O. donde cada uno controla una parte y se adapta para su uso en los sistemas distribuidos.

1.5 TENDENCIAS

- multiprocesamiento.
- Distribución del control
- Soporte del paralelismo masivo y soporte de la ejecución concurrente.
- maquinas virtuales y el almacenamiento virtual.
- Compatibilidad.

1.6 HARDWARE Aspectos

Compaginación del almacenamiento: Objetivo: acelerar el acceso al almacenamiento primario (**banco de almacenamiento primario)**
Registro de relocalizacion: Permite relocalizar los programas. Almacena la dirección base de un programa
interrupciones y escrutinio:
Interrupciones: permite obtener la atención de una unidad. Salva el "estado" antes de la interrupción.
Escrutinio: permite verificar el estado de otra unidad independiente.
Utilización del "buffer": es un área de almacenamiento primario.

- "Entradas de buffer simple":
  - No puede haber simultaneidad entre operaciones ➔Afecta la performance.
- "Entradas de buffer doble":
  - Permite la sobre posición de operaciones de e / s ➔Mejora la performance.
  - Es la técnica de "buffer biestable (o en flip flop)".

Dispositivos periféricos:

- almacenamiento de grandes cantidades de información fuera del almacenamiento principal.

Protección del almacenamiento:

- Limita el n° de direcciones que un programa puede referenciar.
  - Se implementa mediante los "registros de limites" o "claves de protección del almacenamiento"

Temporizadores y relojes:

- "Temporizador de intervalos": previene que usuario monopolice el procesador en Sist. Multiusuario.
- genera una interrupción

Operaciones en línea y fuera de línea; procesadores satélite:

- "Operación en línea": conectados al procesador.

- o "operación fuera de línea:" conectados a unidades de control

Canales de entrada / salida:

- o Se dedican al manejo de la e / s con independencia del procesador principal y tienen acceso directo al almacenamiento principal
- o Principales tipos de canales:
  - Selectores.
  - Multiplexores de bytes.
  - Multiplexores de bloques.

Robo de ciclo: Significa que en la competencia entre el procesador y los canales para acceder a un determinado banco de almacenamiento primario (memoria principal), se da prioridad a los canales:

Estado de problema, estado supervisor, instrucciones privilegiadas: Son los distintos "estados de ejecución".

- "estado de problema o de usuario:" acceso a un subconjunto de instrucciones
- "estado supervisor o de núcleo": acceso a todas las instrucciones
- "instrucciones privilegiadas": son aquellas a las que no se tiene acceso en estado de problema.

Almacenamiento virtual:

- o Los sistemas de almacenamiento virtual permiten a los programas referenciar direcciones que no necesitan corresponder con las direcciones reales disponibles en el almacenamiento primario.
- o "Direcciones virtuales" de los programas en ejecución: son traducidas dinámicamente por el hardware a las "direcciones reales"
- o Se utilizan técnicas de:
  - "paginación": bloques de datos de tamaño fijo
  - "segmentación": identifica las unidades lógicas

Multiprocesamiento: Varios procesadores comparten un almacenamiento primario común y un solo s. O. Y es necesario "secuencializar" el acceso.

Acceso directo a la memoria (dma):

- o Requiere una sola interrupción al procesador por cada bloque de caracteres transferidos durante la operación de e / s y mejora el rendimiento.
- o Es como si el procesador, en vez de interrumpido fuera retrasado.
- o "canal dma": es el responsable del robo de ciclos y de la operación de los disp de e / s.

Canalización: Es ka técnica de explotar ciertos tipos de paralelismo durante el procesamiento de instrucciones.

Jerarquía de almacenamiento:

- Almacenamiento "cache": memoria veloz que aumenta la velocidad de ejecución de los programas:
- Almacenamiento primario: memoria principal.
- Almacenamiento secundario: discos, cintas, etc.

1.7 SOFTWARE

Consiste en los programas de instrucciones y datos que definen los algoritmos necesarios para la resolución de problemas.

Programación en lenguaje de maquina:

- o "Lenguaje de maquina":
  - Lenguaje que un computador puede comprender directamente y que es "dependiente de la maquina"
  - Muy poco usado.

Ensambladores y macroprocesadores:

- o Los "lenguajes ensambladores":
  - Incrementar la velocidad de programación.
  - Reducir los errores de codificación.
- o Los "macroprocesadores":
  - Acelerar la codificación de un programa ensamblador.
  - Una "macroinstrucción" indica la ejecución de varias instrucciones en lenguaje ensamblador.

Compiladores: Es un "Lenguajes de alto nivel" que permiten el desarrollo de programas "independientes de la maquina".

- o "compiladores": traducen los lenguajes de alto nivel al lenguaje de maquina.
- o "traductores": denominación para "compiladores" y "ensambladores".

Sistemas de control de entrada / salida (iocs: input / output control system):

- Libera al programador de aplicaciones de la complejidad de la administración de la e / s:

Utilización del spool (operación simultanea de periféricos en línea):

- Evita la demora en la ejecución de programas como consecuencia del uso de periféricos lentos.

Lenguajes orientados hacia el procedimiento *versus* lenguajes orientados hacia el problema:

- O. Hacia el procedimiento: resolver gran variedad de problemas:
- O. Hacia el problema: resolver determinados tipos de problemas:

Compiladores rápidos y sucios *versus* compiladores optimizadores:

- C. Rápidos y sucios: producen rápidamente un programa objeto que puede ser ineficiente respecto de almacenamiento y velocidad de ejecución:
- C. Optimizadotes: mayor lentitud pero altamente eficiente en almacenamiento y ejecución

Interpretadores:

- No producen un programa objeto.
- Ejecutan un programa fuente.

Cargadores absolutos y de relocalizacion:

- Los programas se ejecutan en el almacenamiento principal.
- "asignación": es la asociación de instrucciones y datos
- "cargador": coloca las instrucciones y datos
- Cargador absoluto": instrucciones y datos en las localizaciones especificas
- "cargador de relocalizacion": Un programa en varios lugares

Cargadores de enlace y editores de enlace:

- "cargador de enlace": en el momento de carga, combina cualesquiera programas requeridos y los carga directamente en el almacenamiento primario.
- "editor de enlace": ejecuta la combinación de programas mencionada y además crea una imagen de carga a memoria

1.8 MEMORIA FIJA

- La "microprogramación dinámica": permite cargar fácilmente los nuevos "microprogramas" ("microcodigo") dentro del "almacenamiento de control"
- Los "microprogramas" están formados por "microinstrucciones" son:
  - Más elemental.
  - Función más dispersa.
  - El almacenamiento de control debe ser mucho mas rápido que el almacenamiento primario.

Microcodigos vertical y horizontal:

El microcodigo permite mejorar el rendimiento en la ejecución de un sistema computacional.
El criterio es colocar en la memoria fija (en vez de en el software) las secuencias de instrucciones utilizadas con mas frecuencia.

- M. Vertical:
  - Similar lenguaje de maquina.
  - Especifica el movimiento de uno o varios datos entre registros.
- M. Horizontal:
  - Es más poderoso pero más complejo que el m. Vertical.
  - Las microinstrucciones requieren muchos más bits.

Emulacion: es la técnica que hace que una máquina aparente ser otra, donde el conjunto de instrucciones de lenguaje de maquina que va a ser emulada se microprograma en la "maquina anfitriona".

Microdiagnosticos: efectúa la detección y corrección de errores más amplia a un nivel mas fino.

Computadores personalizados: Moldean el sistema computacional según las necesidades del usuario.

Microprogramacion y sistemas operativos:

- Las Funciones implementadas frecuentemente en microcodigo son:
  - Manejo de interrupciones.
  - Mantenimiento de estructuras de datos.
  - Primitivas de sincronización
  - "intercambio de contexto",

TEMA 2

PROCESOS Y ADMINISTRACION DEL PROCESADOR

2.1 INTRODUCCION Y DEFINICIONES SOBRE PROCESOS

• **"Proceso":** es una abstracción de un programa en ejecución llamada "tarea"; es un programa que se esta ejecutando, una actividad asincrónica (sin bloqueo). O bien una entidad a la cual son asignados los procesadores. La unidad "despachable".

• En sistemas de **multiprogramación** la cpu alterna de programa en programa en un esquema de **seudoparalelismo**. La CPU ejecuta en un instante dado solo un programa, intercambiando rápidamente entre uno y otro.

• El **paralelismo real de hardware** se da en ejecución de instrucciones con mas de un procesador en uso simultáneamente, y con la superposición de ejecución de instrucciones de programa con la ejecución de una o mas operaciones de e / s.

• Un proceso es una actividad de un cierto tipo que tiene un programa, e/s y estado. Un solo procesador puede ser compartido entre varios procesos con "algoritmo de planificación" que determine cuando detener un trabajo en un proceso y dar servicios a otro procesador.

• Los procesos pueden generar procesos hijos mediante llamados al S.O. pudiendo darse ejecución en paralelo.

2.2 ESTADOS DE PROCESOS

cada proceso es una entidad independiente pero a veces interactúa con otros. Los estados que pueden tener un proceso son:

- **En ejecución**: utiliza la cpu en ese momento.
- **Listo**: es ejecutable, se detiene temporalmente porque se ejecuta otros procesos.
- **Bloqueado**: no se puede ejecutar por la ocurrencia de algún evento externo (ej. espera de datos que aún no estan disponibles).

Hay cuatro transiciones entre estos estados:

1. ***Bloqueo:*** el proceso se bloque en espera de datos. E➔B
2. ***Tiempo Excedido***: el planificador elige otro proceso E➔L
3. ***Despacho*:** el planificador elige éste proceso L➔E
4. ***Despertar**:* los datos que precisa están disponibles B➔L

El bloqueo es la única transición de estado iniciada por el propio proceso del usuario, las otras son iniciados por entidades ajenas al proceso.

PBC Bloque de control de proceso: contiene información sobre Estado actual del proceso, Identificación única de proceso, Prioridad de procesos, etc. cuando el S.O. realiza el intercambio de contexto entre los procesos utiliza el PCB para mantener información que necesita para reiniciar el proceso después.

Crear y destruir un proceso: la creación de un proceso implica darle un nombre, determinar la prioridad, crear su PCB, asignarle recursos iniciales. La destrucción de un proceso implica borrarlo del sistema, devolver sus recursos, borrar su PCB. Puede o no significar su destrucción de sus hijos. Un proceso suspendido no sigue hasta que otro proceso lo reanude en el punto donde fue suspendido.

2.3 PROCESAMIENTO DE INTERRUPCIONES

"Interrupción" : evento que altera la secuencia en que el procesador ejecuta las instrucciones. Esto es generado por el hardware del computador. Cuando ocurre una interrupción el S.O. obtiene el control, almacena el estado del proceso interrumpido en su PCB y analiza la interrupción, transfiere el control a la rutina adecuada para la manipulación de interrupciones de acuerdo a su tipo. Una interrupción puede ser iniciada por un proceso en estado de ejecución o por un evento que puede o no estar relacionados con un proceso en ejecución.

Tipos de interrupciones

1. **SVC:** llamada al supervisor, petición del usuario para un servicio del sistema. (ej. obtener más memoria).
2. **E/S:** iniciada por el hardware de E/S, indicando a la CPU que ha cambiada el estado de un canal o dispositivo (Ej. impresora sin papel).
3. **externas:** causadas por distintos eventos (ej. Expiración de un caunto).
4. **De reinicio:** ocurre al presionar la tecla de reinicio.
5. **De verificación de programa:** causada por errores producidos durante la ejecución de procesos. (ej. intentar dividir por cero).
6. **de verificación de maquina:** ocasionado por mal funcionamiento del hardware y se lo comunica al S.O.

El S.O. incluye rutina: "MANIPULADORES DE INTERRUPCIONES" para procesar cada tipo diferente de interrupción. Cuando ocurre una interrupción el S.O. salva el estado del proceso, dirige el control al manipulador y se aplica la técnica de Cambio de Control.
PSW: "Palabra de estado de programa", Los S.O. trabajan con información de control (PSW) que controlan el orden de ejecución de las instrucciones". Contiene información sobre el estado del proceso, existen tres tipos de PSW: actual, nueva, y vieja. La PSW actual almacena la dirección de la próxima instrucción a ejecutar. En un Sistema uni procesador hay un PSW actual, 6 nuevas y 6 viejas. (cada uno para un tipo de interrupción) la PSW nueva tiene la dirección adecuada para este tipo. Cuando ocurre una interrupción el procesador cambia de PSW, la actual en la vieja y la nueva en la actual, y luego sigue el proceso que estaba en ejecución o el de mas alta prioridad según si el proceso es apropiativo o No apropiativo. Un proceso de interrupción NO puede ser interrumpido".

2.4 EL NUCLEO DEL SISTEMA OPERATIVO: Controla todas las operaciones que implican procesos. Es una pequeña porción del código de todo el S.O. Pero es mas importante y de amplio uso. Suele permanece en el almacenamiento primario; si esta activo. El proceso de interrupciones se encuentra en el núcleo, debe ser rápido en sistemas multiusuarios para optimizar el uso de los recursos del sistema y dar buenos tiempos de respuesta a los usuarios interactivos.
Funciones del núcleo del S.O.: crea y destruye procesos, cambia estados de procesos, despacho, suspende y reanuda procesos, sincroniza y comunica a los procesos, manipula los PCB.

2.5 PLANIFICACION DE PROCESOS: El planificador es la porción del S.O. que decide que procesos se va a ejecutar, mediante un algoritmo de planificación.
Características que tienen un buen algoritmo:
Equidad: (cada proceso obtiene su proporción justa de la cpu)
Eficacia (la CPU ocupada).
Tiempo de respuesta mínimo (para el usuario interactivo)
Tiempo de regreso mínimo (que es para el usuario por lote).
Rendimiento máximo: (de tareas procesadas por hora).

Cada procesador es impredecible, puede requerir muchas operaciones de E/S o mucha CPU para cálculo. Para evitar que un proceso se apropie de la CPU los equipos poseen un dispositivo que provoca una interrupción del reloj en forma periódica cada 60 HZ. (60 veces por segundo) en cada interrupción si el proceso sigue o se suspende y cede la CPU a otro.

Planificación de procesos

Planificación Apropiativa: permitir que procesos ejecutables se suspendan temporalmente. Se les quita la CPU	**Planificación no Apropiativa:** permitir que un proceso se ejecute hasta terminar. No se le quita la CPU.	**Planificación del procesador**: determinar cuando asignar los procesadores y a que procesos. (S.O.)

2.6 NIVELES DE PLANIFICACION DEL PROCESADOR
1)- planificación de Alto nivel": es la planificación de trabajos que determina que "trabajos van a competir por los recursos" del sistema.
2)- Planificación de "Nivel Medio": determina que procesos competirán por la CPU. Se efectúan supervisiones y reanudaciones de procesos.
3)- Planificación de "Bajo Nivel": determina que procesos listos se le asignará la CPU cuando quede disponible y se le asigna; despacha la CPU al proceso lo hace "el despachador del S.O." que opera muchas veces por segundo y está siempre en el almacenamiento primario.

2.7 OBJETIVOS DE LA PLANIFICACION: debe ser justa con todos los procesos; Maximizar la capacidad de ejecución; Maximizar el n° de usuarios interactivos; Minimizar la sobrecarga; Equilibrar el uso de recursos; Evitar la postergación indefinida (mediante la estrategia de envejecimiento del proceso que aumenta su prioridad por esperar un recurso).

2.8 CRITERIOS DE PLANIFICACION: Para realizar estos objetivos un mecanismo de planificación debe considerar las características del proceso (prioridad, tiempo de ejecución, etc.). También hay que tener en cuenta que frecuentemente un proceso un proceso genera fallos (carencia) de páginas por que no tiene sus conjuntos de trabajo en el almacenamiento primario.

Otra limitación a tener en cuenta es la limitación de un proceso de E/S y a la CPU si el proceso es por lote o interactivo, la prioridad de un proceso, etc.

2.9 PLANIFICACION APROPIATIVA VERSUS NO APROPIATIVA

La planificación Apropiativa: si se le otorga la CPU al proceso se le puede retirar, es útil cuando los procesos de alta prioridad requieren rápida atención, garantiza buenos tiempos de respuesta en sistemas interactivos de tiempo compartido. Tiene su costo en recursos, el intercambio de contexto implica sobrecarga (hay varios procesos en el almacenamiento primario en espera de CPU).

La planificación no Apropiativa: se le otorga la CPU al proceso. NO se le puede retirar hasta que termine. Los trabajos largos hacen esperar a los cortos, hay más equidad en el tratamiento de los procesos, son más predecibles los tiempos de respuesta.

2.10 TEMPORIZADOR DE INTERVALOS O RELOJ DE INTERRUPCION

El s. O. Posee un "reloj de interrupción" o "temporizador de intervalos" para generar una interrupción en algun tiempo para que atienda la CPU a un proceso de usuario y evitar que no monopolice el sistema, garantizando buenos tiempos de respuesta a usuarios interactivos. El S.O. decide que proceso obtendrá la CPU.

2.11 PRIORIDADES *Asignadas automáticamente por el sistema. Pueden ser *ESTATICAS (son asignadas literalmente sin importar su verdadero valor, no cambia), DINAMICAS: (cambian, son afectadas por el exterior)

2.12 TIPOS DE PLANIFICACION

1. **Planificación a plazo fijo:** trabajos que se planifican para terminase en un tiempo específico o plazo fijo. Es complejo y se necesita una lista de recursos a utilizar.
2. **Planificación garantizada:** hay un compromiso de desempeño con el proceso del usuario. Si hay tantos procesos cada uno recibiría tanta potencia de CPU. Prioriza los procesos que han recibido menos cpu de la prometida.
3. **Planificación del primero en entrar primero en salir (FIFO):** los procesos se despachan de acuerdo con su tiempos de llegada a la cola de listos y se ejecutan hasta terminar. Es NO APROPIATIVA. Tiempos de respuestas predecibles.
4. **Planificación de asignación en rueda (RR: round robin):** los procesos se despachan en FIFO y disponen de una cantidad limitada de tiempo de CPU (cuanto). Si un proceso no termina antes de que expire su cuanto, la CPU es apropiada por otro proceso, éste va al final de la cola de listos. Hay poca sobrecarga si hay suficiente memoria para los procesos. Es buena en tiempo compartido.
   Tamaño del Cuanto: debe ser lo suficientemente grande como para permitir que la mayoría de las peticiones interactivos requieran de menor tiempo que la duración del cuanto. Se minimiza la sobrecarga de apropiación y se maximiza la utilización del E/S.
5. **Planificación del trabajo mas corto primero (SJF):** NO APROPIATIVA. No recomendable en ambientes de tiempo compartido. El proceso en espera con menor tiempo estimado de ejecución hasta que termine es el siguiente en ejecutarse.
6. **Planificación del tiempo restante mas corto (SRT):** APROPIATIVA. Útil en sistemas de tiempo compartido. El proceso con el tiempo estimado de ejecución menor para terminar es el siguiente en ejecutarse. Hay mas sobrecarga por el cambio de contexto, ya que un proceso en ejecución puede ser apropiado por un nuevo proceso con un tiempo de ejecución menor.
7. **Planificación el siguiente con relación de respuesta máxima (HRN):** NO APROPIATIVO. Corrige el favoritismo del SJF al nuevo trabajo corto. La prioridad esta en función: del tiempo de servicio del trabajo y la cantidad de tiempo que el trabajo ha estado esperando
   Prioridad = (te + ts) / ts.
8. **Planificación por prioridad:** Considera factores externos al proceso. Cada proceso tiene asociada una prioridad. Y se ejecuta la más alta. Los procesos de E/S tiene alta prioridad por ser mucho su tiempo de espera de operaciones de E/S.
9. **Colas de retroalimentación de niveles múltiples:** Favorecer trabajos cortos Y limitados por la e / s para optimizar el uso de los dispositivos de E/S. un proceso entra en la línea de espera al final de la cola superior y se mueve por ésta en "FIFO" hasta obtener la CPU. Si su cuanto expira y no se terminó el proceso se coloca al final de la cola del siguiente nivel inferior.
   Política versus mecanismos de planificación: puede ocurrir que haya procesos hijos ejecutándose, lo cual el planificador no identifica esto. La solución es separar el MECANISMO DE PLANIFICACION de la POLITICA DE PLANIFICACION, para ello se parametriza el algoritmo de planificación y los parámetros pueden determinar por medio de procesos del usuario, así el mecanismo está en el núcleo del S.O. pero la política queda establecida por un proceso del usuario.

10. **Planificación de dos niveles:** Si la memoria principal es insuficiente habrá procesos ejecutables que se mantengan en disco; y Será mayor el tiempo de alternancia del proceso para traerla del disco. Pero esto es más eficiente el intercambio e procesos con un planificador de dos niveles.

2.13 MULTIPROCESAMIENTO "Sistemas de multiprocesamiento": Es configurar un "sistema de computación con varios procesadores con el objetivo: de incrementar la capacidad de ejecución". Si un procesador falla los restantes continúan operando. Es escasa la explotación del paralelismo. Para que haya paralelismo masivo se debe disponer de suficientes procesadores como para que todas las operaciones que puedan ejecutarse en paralelo se asignen a procesadores separados, es una forma de ejecutar un programa en el menor tiempo posible. El S.O. debe percibir que ha fallado un procesador determinado y ya no podrá asignarlo y también debe ajustar sus estrategias de asignación de recursos para evitar la sobrecarga del sistema que esta degradado.
"Regla de nunca esperar": es mejor darle a un procesador una tarea que pueda llegar a no ser utilizada que tenerlo ocioso.

2.14 ORGANIZACION DEL HARDWARE DEL MULTIPROCESADOR: El problema es determinan los medios de conexión de los multiprocesadores y los procesadores de E/S a las unidades de almacenamiento.
Los multiprocesadores contienen 2 o mas procesadores, todos comparten un almacenamiento común, canales de E/S, unidades de control y dispositivos. Todo es controlado por un solo S.O. Las organizaciones más comunes son:

1. **Tiempo compartido o bus común (o conductor común):** usa un solo camino de comunicación con todas las unidades. Si un procesador desea trasferir datos debe verificar la disponibilidad del conductor y de la unidad de destino luego iniciar la transferencia de datos. Es una organización simple, económica y flexible; el sistema falla si falla el bus.
2. **Matriz de barras cruzadas e interruptores:** Existe un camino diferente para cada unidad de almacenamiento. La tasa de transferencia es muy alta por la multiplicidad de caminos de transmisión. La referencia a dos unidades diferentes de almacenamiento es simultánea.
3. **Almacenamiento de interconexión múltiple:** Hay una conexión de almacenamiento por unidad funcional. La conexión es más compleja que en los otros esquemas.

2.15 GRADOS DE ACOPLAMIENTO EN MULTIPROCESAMIENTO: Se clasifican en :

1. **"Multiprocesamiento ligeramente acoplado":** conecta dos o mas sistemas independientes por medio de un enlace de comunicación. Cada sistema tiene su propio S.O. y almacenamiento. Los sistemas funcionan independientemente y se comunican si es necesario para acceder a archivos de otros o intercambiar tareas a otro procesador.
2. **Multiprocesamiento rígidamente acoplado":** utiliza un solo almacenamiento compartido por varios procesadores. Usa un solo S.O. que controla todos los procesadores y el hardware del sistema.
3. **Organización maestro / satélite:** Un procesador es "maestro" y los otros "satélites". El maestro es de propósito general (operaciones de E/S y computaciones). Si falla un satélite se pierde capacidad computacional pero el sistema no falla, si falla el maestro por no efectuar operaciones de E/S.

2.16 S. O. DE MULTIPROCESADORES: Soportan fallas de hardware en procesadores individuales y continúan operando. Algunas funciones de los S.O. de multiprogramación y multiprocesadores son: (asignar y administrar recursos, prevenir el interbloqueo del sistema, equilibrar cargas de E/S, y del procesador y reconfiguración). En sistemas de multiprocesadores es fundamental explotar el paralelismo en el hardware y en los programas, y hacerlo automáticamente.
Las organizaciones básicas de los s. O. Para multiprocesadores son:

1. **Maestro satélite:** fácil de implementar. Solo el procesador maestro ejecuta el S.O.; los satélites ejecutan programas de usuario.
2. **Ejecutivos separados:** cada procesador tiene su S.O. y ejecuta los programas que operan en ese procesador.
3. **Tratamiento simétrico:** para todos los procesadores, complicada de implementar, poderosa y confiable. El S.O. administra un grupo de procesadores idénticos que utilizan cualquier dispositivo de E/S y refencian cualquier unidad de almacenamiento. Todos los procesadores pueden cooperar en la ejecución de un proceso determinado.

2.17 RENDIMIENTO DEL SISTEMA DE MULTIPROCESAMIENTO: **¿**Que pasará si se coloca un nuevo procesador? La adicción de un nuevo procesador no hará que la capacidad de ejecución del sistema aumenta, según la capacidad del nuevo procesador. Debido a que habrá una sobrecarga adicional del S.O. y la productividad tiende a disminuir cierto número de procesadores.

2.18 RECUPERACIÓN DE ERRORES: La capacidad mas importante en estos "sistemas es soportar fallos de hardware" en procesadores individuales y seguir trabajando. Debe haber capacidad de detección y corrección de errores de hardware; detectar fallos antes de que se produzcan y el S.O. dirigir un procesador y haga el trabajo del que falló.

2.19 MULTIPROCESAMIENTO SIMETRICO (MPS): Cada procesador posee capacidades funcionales completas. Los dispositivos de e / s pueden ser conectados a cada uno de los procesadores. Todos los llamados al supervisor pueden ser ejecutados en todos los procesadores. Cada procesador puede ejecutar el planificador para buscar el siguiente trabajo a ejecutar. Utiliza una sola cola de trabajo y cada procesador puede seleccionar trabajos de ella. Las interrupciones de reloj ocurren en diferente momento para minimizar la contención del despacho de procesos.

2.20 TENDENCIAS DE LOS MULTIPROCESADORES

- El uso de multiprocesadores se incrementara en el futuro.
- La confiabilidad es cada vez mayor.
- La reducción de costos por los avances en microelectrónica.
- Desarrollo de lenguajes que expresen el paralelismo.
- El progreso de la detección automática de paralelismo.
- Mayor compactación de los componentes.

TEMA 3
ADMINISTRACION DE LA MEMORIA

3.1 ALMACENAMIENTO REAL: Un factor importante en el diseño de los S.O. es la organización y administración de la memoria principal o real. Los programas y datos deben estar en el almacenamiento principal para poder ejecutarlos y reverenciarlos directamente, si no son necesarios de inmediato se van al almacenamiento secundario o auxiliar que son los soportados en discos. Actualmente los programas crecen en requerimientos de memoria tan rápido como las memorias.
La parte del sistema que administra la memoria es el "ADMINISTRADOR DE LA MEMORIA" que lleva un registro de las partes de la memoria que se están usando y las que no; asigna espacio a los procesos cuando la necesitan y libera espacio de memoria asignada a un proceso que terminó.

3.2 ORGANIZACION Y ADMINISTRACION DEL ALMACENAMIENTO: el almacenamiento se debe optimizar su uso por ser un recurso costoso. La organización es "la manera de considerar este almacenamiento". (si se coloca un programa o varios, se divide en particiones diferentes o iguales, los procesos se ejecutan donde quepan o en un lugar específico, etc.).
Administración del almacenamiento: independientemente del esquema de organización, se decide: las estrategias que se usaran para optimizar el rendimiento (Cuando traer un programa nuevo y colocarlo en memoria, donde se colocaran el programa, qué se ejecutará a continuación, con qué criterios se desplazan los programas, etc).

3.3 JERARQUIA DE ALMACENAMIENTO El almacenamiento principal es mas costoso y menor que el secundario pero es de acceso mas rápido. Los sistemas con varios niveles de almacenamiento destinan recursos para administrar el movimiento de programas y datos con niveles. Un nivel adicional es el "CACHE" o memoria de alta velocidad, es más rápida y más costosa que la memoria principal. Los programas se traspasan de la memoria principal al caché antes de su ejecución y allí se ejecutan mas rápido. Se espera con la caché que baje la sobrecarga de traspaso y mejore el rendimiento al ser mas rápido.

3.4 ESTRATEGIAS DE ADMINISTRACION DEL ALMACENAMIENTO
Son para obtener el mejor uso posible de la memoria. Las estrategias son:
Estrategia de Búsqueda: ¿Cuándo? Obtener el siguiente fragmento de programa o de datos para colocar en la memoria.
Estrategia de búsqueda por demanda: se carga cuando algún programa lo referencia.
Estrategia de búsqueda anticipada. Produce mejor rendimiento del sistema.
Estrategia de colocación: ¿en que lugar?. De la memoria se colocará (cargar) un programa nuevo.
Estrategia de reposición: ¿Qué fragmento?. De programa o de datos desplazar para dar lugar a programas nuevos.

Asignación contigua de almacenamiento y No contigua:
En la asignación contigua: cada programa ocupa un bloque contiguo de localizaciones de almacenamiento.
En la asignación NO contigua: un programa se divide en varios bloques o "segmentos" que se almacenan en direcciones que no son adyacentes, es más compleja pero más eficiente que la contigua.

Asignación contigua de almacenamiento de un solo usuario: el tamaño de un programa esta limitado por la capacidad de memoria principal que necesita.
El almacenamiento se divide en porciones que contienen: el S.O., el programa del usuario, y una porción sin usar. El programa del usuario podría destruir áreas del S.O. ➔ debe estar protegido el S.O. contra el proceso del usuario.

3.5 MULTIPROGRAMACION DE PARTICION FIJA: En los sistemas de Multiprogramación se permite almacenar varios procesos usuarios competir al mismo tiempo por los recursos del sistema. Es necesario que varios procesos residan a la vez en la memoria principal. En la multiprogramación de partición fija: las "particiones" del almacenamiento principal son de tamaño fijo y alojan un proceso cada uno. La CPU se cambia rápidamente con los procesos creando simultaneidad.
Los procesos se traducen con ensambladores y compiladores pasa ser ejecutados solo dentro de una partición específica. El S.O. es de implementación sencilla pero no se optimiza el uso de la memoria. La protección en los sistemas de multiprogramación con asignación contigua de memoria se implementan los "registros de límites", indicando el límite superior e inferior de una partición o región.
La fragmentación en la multiprogramación de partición fija ocurre cuando los trabajos de usuario no llenan sus particiones designados.

3.6 MULTIPROGRAMACION DE PARTICION VARIABLE: Los procesos ocupan tanto espacio como necesitan, pero obviamente no deben superar el espacio disponible de memoria. No hay límites fijos de memoria, la partición de un trabajo es su propio tamaño. Se utiliza el esquema de asignación contigua (el programa ocupa lugares adyacentes de almacenamiento). Los procesos que termina dejan disponibles espacios de memoria principal llamados "AGUJEROS" que pueden usar otros trabajos dejando también agujeros cada vez más chicos y así se genera un desperdicio de memoria principal. Estos agujeros o áreas libres se fusionan para formar uno más grande siendo todos adyacentes o se realiza una compactación de almacenamiento que es cuando los agujeros separados por todo el almacenamiento principal se agrupan al final de las áreas ocupadas dejando un solo agujero grande de memoria libre contigua. Esta técnica es: "RECOGIDA DE RESIDUOS" esta comprensión consume recursos del sistema y este se debe detener mientras la efectúa afectando los tiempos de respuesta. Implica la relocalizacion reubicación) de los procesos que están en la memoria.

Estrategias de colocación del almacenamiento: Se usan para determinar el lugar de la memoria donde se colocará los programas y datos que llegan.

1. *Estrategia de mejor ajuste:* un trabajo nuevo se coloca en el agujero donde quepa más ajustado (deje el menor espacio).
2. *Estrategia de primer ajuste:* un trabajo nuevo se coloca en el primer agujero que quepa.
3. *Estrategia de peor ajuste:* coloca un programa en el agujero que quepa peor (deje el mayor espacio para alojar otro).

3.8 ALMACENAMIENTO VIRTUAL:
Introducción: Es la capacidad de direccional un espacio de memoria mayor que el disponible en el almacenamiento principal" de un sistema de computación.
Los métodos de implementación son mediante:

- Técnica de Paginación
- Técnica de Segmentación.
- Combinar ambas técnicas.

3.9 CONCEPTOS BASICOS DE ALMACENAMIENTO VIRTUAL
La clave del almacenamiento virtual está en la disociación de las direcciones virtuales a la que hace referencia un programa y las direcciones reales disponibles en la memoria real.

- Direcciones virtuales: son los referidos por un proceso en ejecución.
- Direcciones reales: son las disponibles dentro del almacenamiento primario.
- Espacio de direcciones virtuales de un proceso: todas las direcciones virtuales referidas por un proceso.
- Espacio de direcciones reales de un computador: todas las direcciones reales disponibles en el ordenador.

Dat (traducción de espacio de direcciones): las direcciones virtuales se convierten (traducen) en reales al circularse el proceso con mecanismos de traducción muy rápidos.

3.10 ORGANIZACION DEL ALMACENAMIENTO DE NIVELES MULTIPLES:

Se utiliza un esquema de almacenamiento de dos niveles:

1. **Primer nivel "almacenamiento real":** aquí se ejecutan los procesos y en el deben estar los lados para que un proceso se refiera a ellos. Cuando se va a ejecutar un proceso su código y datos se pasan al almacenamiento primarios. El almacenamiento primario es compartido por varios procesos.
2. **Segundo nivel "almacenamiento auxiliar":** son discos de gran capacidad que mantienen los programas y datos que no caben en el almacenamiento principal al mismo tiempo.

3.11 TRANSFORMACION DE BLOQUES: Los mecanismo de traducción dinámica de direcciones (DAT) mantienen información agrupada en bloques sobre ¿Qué? Direcciones del almacenamiento virtual están en el almacenamiento real, y ¿Dónde? Están.
Si el bloque es de tamaño igual se llaman "PAGINAS" su organización➔ "PAGINACION".
Si los bloques son de tamaño diferente se llaman "SEGMENTOS" su Organización se llama "SEGMENTACION".
Si se combinan las dos técnicas, habrá segmentos de tamaño variable compuestos de páginas de tamaño fijo.
Las direcciones son bidimensionales, es decir que una dirección virtual se indica por el número de bloque donde reside y el desplazamiento a partir del inicio del bloque.

3.12 PAGINACION: las páginas se trasfieren del almacenamiento secundario al primario en bloques llamados "marcos de paginas" que tienen el mismo tamaño que las paginas.
Compartir recursos en un Sistema de paginación: en sistemas multiprogramados de tiempo compartido, mas de un usuario ejecutan los mismos programas, para optimizar el uso de la memoria real se comparten las páginas compartibles y no modificables. Habrá marcos de páginas compartidos por varios procesos. El compartir reduce la cantidad de almacenamiento primario para ejecutar bien los procesos pero mantiene el sistema con muchos usuarios (procesos).

3.13 SEGMENTACION: Se transfiere del almacenamiento secundario al primario como unidades completas. En los sistemas de segmentación un programa y sus datos pueden ocupar varios bloques separados de almacenamiento real. Estos bloques no necesitan ser de igual tamaño o ser adyacentes, deben estar compuestos de posiciones contiguas de almacenamiento.
La protección de bloques de memoria se complica y se usan claves de protección del almacenamiento.
Un proceso se ejecuta solo si su segmento actual está en el almacenamiento primario.
Control de acceso en sistemas de segmentación: A cada proceso se le otorgan derechos de acceso a ciertos segmentos. Tipos de acceso:
"acceso de lectura" ➔obtiene información acerca del contenido en ese segmento.
"acceso de escritura" ➔ modifica el contenido del segmento además de destruir.
"acceso de ejecución" ➔puede ejecutar un segmento como si fuera un programa.
"acceso de adición" ➔escribe información solo al final del segmento sin modificarla.
Compartimiento en un sistema de segmentación: una ventaja de la segmentación sobre la paginación es que se trata de un hecho lógico más que físico. En un sistema de segmentación si un segmento se declara compartido, entonces las estructuras que la integran pueden cambiar de tamaño, no cambian el hecho lógico de que están en un segmento compartido.

3.14 SISTEMAS DE PAGINACION / SEGMENTACION

Ofrece las ventajas de las dos técnicas de organizar el almacenamiento virtual.
El tamaño de los segmentos es múltiplo de las páginas. No es necesario que todas las páginas de un segmento estén al mismo tiempo en el almacenamiento primario. Las páginas del almacenamiento virtual que son contiguas no necesitan serlo en el almacenamiento virtual.

3.15 ADMINISTRACION DEL ALMACENAMIENTO VIRTUAL

Las diferentes organizaciones de almacenamiento virtual que se suelen implementar son: paginación, segmentación y segmentación/paginación. "La administración del almacenamiento virtual en sistemas de almacenamiento virtual utilizan estrategias que condicionan la conducta de estos sistemas".

1. **"Estrategia de búsqueda":** ¿Cuándo? Una página o segmento debe traerse del almacenamiento secundario al almacenamiento primario.

Búsqueda por demanda: se espera hasta que se haga referencia a una página o segmento por un proceso antes de traerla al almacenamiento primario.

Búsqueda anticipada: se intenta determinada por adelantado que página o segmento hará referencia un proceso para traerlo al almacenamiento primario.

2. **"Estrategia de colocación":** ¿en que lugar? Del almacenamiento primario se colocará una nueva página o segmento. El sistema decide trivialmente ya que una nueva página se puede colocar dentro de cualquier marco de página disponible.
3. **"Estrategia de reposición":** se decide ¿Cuál página o segmento desplazar? Para dar lugar a una nueva página o segmento cuando el almacenamiento primario esta completamente comprometido. Principales estrategias:

- ♣ El principio de optimización: para obtener un buen rendimiento, la página que se repone es la que no se usará en el futuro por mucho tiempo. el problema es predecir bien.
- ♣ Reposición de página al azar: se escoge al azar la página que se va a reemplazar, teniendo todas las mismas posibilidades. No se suele usar.
- ♣ FIFO (primero en entrar primero en salir): se registra el momento en que ingresa cada página al almacenamiento primario y se reemplaza la página que más tiempo estuvo almacenado sin importar el uso.

Se presenta el inconveniente de que se pueden reemplazar paginas muy usadas: "ANOMALIA FIFO"

- ♣ LRU (menos recientemente usada): se reemplaza la Pág. que no se uso durante el mayor tiempo ya que cada Pág. tiene un sello de tiempo cuando es referenciada. Hay sobrecarga.
- ♣ LFU (menos frecuentemente usada): se reemplaza la Pág. que ha sido usada con menor frecuencia (intensidad). Puede ser la Pág. que recién entro al almacenamiento primario.
- ♣ NRU (no usada recientemente): se reemplaza la página que no se referenció recientemente. Se agregan dos bits de hardware por página: **R➔BIT referenciador **M➔BIT modificado. (SI ➔ 1, NO➔ 0)

3.16 LOCALIDAD: Los procesos tienden a hacer referencia al almacenamiento en patrones no uniformes y muy localizados. La localidad se manifiesta en el tiempo y en el espacio; es una propiedad empírica (observada), nunca esta garantizada pero es altamente probable. "la localidad esta relacionada con la forma en que se escriben los programas y se organizan los datos".

- ♣ Localidad Temporal: es que las localidades del almacenamiento referenciado recientemente son muy probables de que se referencien mas adelante. (Ej. subrutina).
- ♣ Localidad en el espacio: es que las referencias de almacenamiento se acumulan y si se hace referencia a una localidad es probable que las localidades cercanas también lo sean (Ej. arreglo).

El número de fallos de páginas de un proceso depende de la cantidad de almacenamiento primario disponible para sus páginas mientras el subconjunto de paginas de un proceso esta en el almacenamiento primario, el numero de fallos de página no aumenta.

Los conjuntos de trabajo cambian mientras un proceso esta en ejecución.

3.17 CONJUNTOS DE TRABAJO: es una "colección de páginas a las que hace activamente referencia un proceso, para que un programa se ejecute eficientemente su conjunto de trabajo

debe mantenerse en el almacenamiento primario. Los conjuntos de trabajo cambian mientras un proceso esta en ejecución.

3.18 PAGINACION POR DEMANDA Y PAGINACION ANTICIPADA

Paginación por demanda: las páginas se cargan al almacenamiento secundario al primario cuando las referencia un proceso en ejecución. Aumenta la cantidad de almacenamiento que usa un proceso y la cantidad de tiempo que lo usan.
Paginación anticipada: el S.O. intenta predecir. Se precargan las páginas que un proceso necesitará; se reduce el tiempo de ejecución.

3.19 LIBERACION DE PÁGINA Y TAMAÑO DE PÁGINA

Liberación de página: un proceso usuario puede emitir una liberación voluntaria para liberar el marco de página cuando ya no necesitara esa página. Lo ideal será que los compiladores y S.O. detecten automáticamente esta necesidad mediante estrategias de conjuntos de trabajo, para así poder evitar el retraso del desarrollo de las aplicaciones.
Tamaño de Página: es algo complejo de tratar debido que esta influido por varios factores. Por esto debemos saber que lo que estemos ganando por un lado lo estaremos perdiendo por otro. Pero generalmente los tamaños de páginas más utilizados son. 512b, 1kb, 2kb, 4kb.

3.20 COMPORTAMIENTO DE UN PROGRAMA EN LA PAGINACION:

un proceso tiende a hacer referencia a una parte significativa de sus páginas inmediatamente después de iniciar su ejecución. Puede conducir sin haber referenciado a algunas de sus páginas. (Ej. rutinas de errores que no se produjeron).

TEMA 4
SISTEMAS DE ARCHIVOS

4.1 INTRODUCCION: Todas las aplicaciones computarizadas necesitan almacenar y recuperar la información. Las condiciones para almacenar información a largo plazo son:

- almacenar gran cantidad de información.
- La información debe sobrevivir a la terminación del proceso que la usa.
- Posibilitar que varios procesos accedan a la información.

➔ La solución es el almacenamiento de la información en discos y otros medios externos en unidades llamadas **archivos.** Los archivos deben ser **persistentes** (no le afecta la creación o terminación de un proceso).

4.2 FUNCIONES DEL SISTEMA DE ARCHIVO

El archivo es una colección de datos con nombre. Se manipulan como una unidad por operaciones como (abrir, cerrar, crear, destruir, copiar); los elementos de datos individuales dentro del archivo se manipulan por operaciones como (leer, escribir, insertar, borrar).

4.4 ARCHIVOS: Un archivo es un conjunto de registros relacionados". Los nombres de los archivos deben ser únicos dentro de un directorio. Algunos sistemas se distinguen con mayúscula y minúscula otro no, el nombre tiene dos partes separados por un punto. *nombre.extensión* (que indica algo del archivo)
La estructura de un archivo puede ser secuencia de bytes, secuencia de registros y árbol (con un campo clave).
La estructuras de archivos pueden ser: secuencia de bytes, secuencia de registros y árbol (con un campo clave).
Los tipos de archivos son:
Archivos regulares con información de los usuarios
Directorios: archivos de sistema para mantener una estructura del sistema de archivos.
Archivos especiales de caracteres: (E/S) y archivos especiales de bloques (modelan discos)
Los tipos de acceso son: **Secuencialmente (de principio a fin) **Aleatoriamente (en cualquier orden)
Con amor flopy
4.3 EL SISTEMA DE ARCHIVOS. Es la parte del S.O. que permite compartir controladamente la información de los archivos, con varios tipos de acceso (lectura, escritura, ejecución).
El sistema de archivos debe proporcionar respaldo y recuperación para prevenir la pérdida accidental o la destrucción de información. Se puede referenciar a un archivo con un nombre, brindando independencia de dispositivo.
El sistema de archivo debe brindar una interface favorable al usuario, suministrando una visión lógica de los datos y las funciones que se van a ejecutar.
El sistema de archivos es un componente importante de un S.O. y suele contener:

- Métodos de acceso: a los datos almacenados en los archivos.
- Administración de archivos: para que sean almacenados, compartidos, etc.
- Administración del almacenamiento auxiliar: para asignar espacio a los archivos
- Integridad de archivo: se relaciona más con la administración del espacio de almacenamiento secundario (discos); se lo organiza con una raíz que indica que comienza el disco, directorios y archivos.
- En sistemas jerárquicos el nombre de un archivo es su trayectoria del directorio raíz al archivo.

4.5 DIRECTORIOS: Los utiliza el S.O. para llevar un registro de los archivos, en muchos sistemas son también archivos. La organización de directorios es muy variada y puede ser:

- Directorio único: un directorio raíz y todos los archivos de los usuarios.
- Directorio por usuario: un directorio por cada usuario, a parte el raíz.
- Árbol de directorios: cada usuario tiene muchos directorios.

El nombre de las rutas de acceso son: ➔ absoluta: trayectoria raíz => archivo
➔ Relativo: se utiliza junto con el concepto de directorio de trabajo o directorio activo.
Las operaciones con directorios: las llamados al sistema permitidos por el manejo de directorios son➔ crear, eliminar (debe estar vacío), abrir, cerrar, leer, ligar, desligar, etc.

4.6 IMPLANTACION DEL SISTEMA DE ARCHIVOS Y SU RELACION CON LA SIGNACION Y LIBERACION DE ESPACIO: Se tiene en mente la forma de almacenamiento de archivos y directorios, la administración de espacio en disco, y la forma de hacerlo bien eficiente y confiable. También hay que tener en cuenta el problema de la fragmentación creciente del espacio en disco, qué trae problemas de performance porqué hace que los archivos se dispersen a través de los bloques separados. Para aliviar esto se utiliza la técnica de recogida de residuos periódicamente a una reorganización de archivos.

Implantación de archivos: el aspecto clave es el registro de los bloques asociados a cada archivo. Los métodos usados son la "asignación contigua o adyacente del almacenamiento secundario, es fácil porque solo necesita el número de bloque del inicio para localizar un archivo y rinde bien en la E/S; el problema es que debe conocer el tamaño máximo del archivo al crearlo y produce fragmentación de discos. En la asignación "NO contigua" el almacenamiento es más dinámico. Los archivos están dispuestos en todo el disco y produce largas búsquedas; el mantenimiento implica sobrecarga. Uno de los esquemas que utiliza este método es La asignación por "Bloques" o por "Nodos-i" (nodos índice) el cual se asocia a cada archivo pequeño tabla o "nodo índice" que contiene los atributos y direcciones en disco de los bloques del archivo, va del disco a la memoria primaria al abrirlo.

Implantación de directorios: para abrir un archivo el S.O. utiliza información del directorio que contiene la dirección del bloque en el disco. La función del sistema de directorios es asociar el nombre del archivo con la información necesaria para localizar los datos.

Archivos compartidos: conviene que estos archivos compartidos aparezcan en distintos directorios de distintos usuarios. La conexión con un directorio y un archivo de otro directorio el cual comparten se llama "ENLACE". El enlace se produce haciendo que el sistema cree un nuevo archivo de tipo "LINK".
El archivo LINK ingresa al directorio del usuario que accede a un archivo de otro directorio y el usuario; solo contiene el nombre de la ruta de acceso del archivo al cual se enlaza, esto se llama "ENLACE SIMBOLICO". En búsqueda genérica se suele encontrar el mismo archivo por distintos rutas.

Administración del espacio en disco: existen dos estrategias para almacenar un archivo. ➔ asignar "N" bytes consecutivos de espacio en el disco. ➔ Dividir el archivo en bloques no necesariamente adyacentes (mas usadas).

El tamaño del Bloque: (unidad de asignación) tienen:

- Sector: es **chico** y con archivo constará de muchos bloques.
- Pista: de ½ K. 1 K. o 2K.
- Cilindro: es **grande** y desperdicia espacio en disco.

DISK CUOTAS: Para evitar que los usuarios se apropien de un espacio excesivo en disco, El S.O. multiusuario proporciona un mecanismo que establece cuotas en el disco. Un administrador del sistema asigne a cada usuario una proporción máxima de archivos y bloques y el S.O. garantice que el usuario no exceda su cuota.

Confiabilidad del sistema de archivos: se debe proteger la información que está en el sistema de archivo efectuando resguardos; así evitando las consecuencias de la pérdida de los sistemas de archivo. Las pérdidas se suelen deber a problemas de hardware, software, hechos externos, etc.
Si hay bloques defectuosos se solucionan por hardware (con una lista de bloques defectuosos) o por software (con un archivo de bloques defectuosos).

Respaldos:

- Copias de seguridad o de back-up: se respalda los archivos con frecuencia que consiste en realizar copias completas del contenido del disco (flexible o rígido).
- Otra estrategia es: El vaciado por incrementos o respaldo incremental: consiste en obtener una copia de respaldo periódicamente llamada copia total y luego obtener una copia diaria solo de aquellos archivos modificados desde la última copia total.

Desempeños de los sistemas de archivos: el acceso al disco es más lento que el acceso a la memoria, los tiempos se miden en Milisegundos para los discos, y nanosegundos para la memoria. Se debe reducir el número de accesos a disco, mediante la técnica del:

- Bloque de Caché (buffer caché): se utiliza el término de « ocultamiento » del francés "cacher = ocultar". Un caché es una colección de bloques que pertenecen al disco pero se mantienen en memoria para mejor rendimiento.

- ♣ Otra técnica para mejorar el rendimiento de un sistema de archivo es reducir la cantidad de movimientos del brazo del disco (mecanismos de acceso): se debe colocar los bloques que probablemente tengan acceso secuencial próximos entre si preferentemente en el mismo cilindro.

4.7 DESCRIPTOR DE ARCHIVOS o bloque de control de archivos: Es un "bloque de control" que contiene información que el sistema necesita para administrar un archivo". Es una estructura dependiente del sistema; y contiene: nombre del archivo, localización del archivo en el almacenamiento secundario, organización del archivo (métodos de acceso); tipo y fechas (creación y destrucción). Los descriptores de archivos suelen mantenerse en el almacenamiento secundario y se pasan al almacenamiento primario al abrir el archivo.
El descriptor de archivo es controlado por el sistema de archivos.

4.8 SEGURIDAD: Los sistemas de archivos contienen información valiosa para el usuario que deben proteger.
La seguridad es la garantía de que los archivos no sean leídos o modificados por personal no autorizado; incluyendo aspectos técnicos, de administración, legales y políticos.
El S.O. utiliza mecanismos de protección para resguardar la información de la computadora.

Dos grandes problemas de la seguridad son:

1. **Pérdida de datos:** ➔*Hechos:* (incendios, inundaciones, guerra, …) ➔*errores de hardware o software* (falla en el disco, CPU, …) ➔*errores humanos:* (pérdida de un disco)
2. **Intrusos:** ➔*pasivos* (solo lee los archivos no autorizados a leer) ➔ *Activos* (hace cambios no autorizados a los datos)
3. **Ataque del caballo de Troya** modificar un programa normal para que haga cosas adversas además de su función usual.
4. **Gusano de Internet:** consta de un programa arrancador y un gusano. Un gusano es un programa completo que hace daño.
5. **Virus:** son fragmentos de programa que se añaden a programas existentes con la intención de infectar (dañar) a otros. Se propaga a través de copias ilegitimas de programas. El virus se ejecuta o intenta reproducirse cuando el programa que la aloja se ejecuta. El virus es más fácil de evitar que curar.

Algunas soluciones para diseñar sistemas mas seguros son:
➔ Añadir contraseñas: se debe prever intentos de penetración consistentes en combinaciones de nombres y contraseñas. Deben almacenarse cifradas (encriptadas).
➔ Identificación física del usuario: (huellas digitales): forma o longitud de los dedos, huellas digitales, vocales, firmas.
➔ Medidas preventivas: (registrar los accesos fallidos) tender trampa a los intrusos.

4.9 MECANISMOS DE PROTECCION: El sistema los utiliza para resguardar información. Muchos objetos del sistema necesitan protección (CPU, memoria, disco, impresora, archivos).
➔ Dominio de protección: cada objeto se referencia por un nombre y un subconjunto de operaciones que se pueden realizar sobre él. Un dominio es un conjunto de parejas (objeto, derechos)
➔ Un derecho es el permiso para realizar algunas de las operaciones.
➔ Listas de control de acceso: asocia a cada objeto una lista ordenada con todos los dominios que pueden tener acceso al objeto, y la forma de dicho acceso (lectura, escritura, ejecución).

4.10 RESPALDOS Y RECUPERACION: La destrucción de la información, ya sea accidental o intencional (falla de hardware y software, fenómenos metereológicos, fallas de energía, robos) debe tenerse en cuenta por el S.O. en general pero por el sistema de archivo en particular. Las técnicas utilizadas son ➔ Respaldos periódicos: hacer copia de los archivos. ➔ Pasar todas las transacciones a un archivo: copiándolo en otro. ➔ Respaldo incremental: se resguarda solo aquellos archivos modificados.

TEMA 5
ENTRADA / SALIDA

5.1 INTRODUCCION: Una de las funciones principales del s. o. es el control de todos los dispositivos de e / s de la computadora; y sus funciones son:

- enviar comandos a los dispositivos.
- detectar las interrupciones.
- controlar los errores.
- proporcionar una interfaz entre los dispositivos y el sistema

El uso inapropiado de los dispositivos de E/S genera ineficiencia del sistema afectando la performance global.

5.2 PRINCIPIOS DEL HARDWARE DE E / S:

1)- **Dispositivos de E/S: se clasifican en:**

➔ Dispositivos de Bloque: la información se almacena en bloques de tamaño fijo, cada bloque con su dirección. Se puede leer y escribir en un bloque de forma independiente (Ej. discos).

➔ Dispositivo de caracteres: la información se trasfiere como un flujo de caracteres sin estructura de bloques, no usan direcciones (Ej. ratón, impresoras de línea).

2)- **Controladores de dispositivos: o "adaptador".** Es un componente electrónico del que constan las unidades de E/S, para manejar más de un dispositivo. El S.O. trabaja con el controlador y no con el dispositivo: el controlador debe efectuar conexiones de errores necesarios, copiar el bloque de la memoria principal. (Ej. Controladores de reloj, teclado, …). Cada controlador posee registros que utiliza para comunicarse con la CPU.

3)- **Acceso directo a memoria: (DMA).** Muchos controladores de dispositivos de bloque permiten el DMA. Si se lee el disco sin DMA el controlador lee en serie el bloque y se desperdicia tiempo de CPU. El DMA libera a la CPU de este trabajo. La CPU le proporciona al controlador la dirección del Bloque en el disco y la dirección en memoria donde debe ir el bloque.

5.3 PRINCIPIOS DEL SOFTWARE DE E/S: Tiene como Objetivo brindar independencia de dispositivo. La idea básica es organizar el software como una serie de capas donde:

- Capa inferior: oculta las peculiaridades del hardware a las capas superiores.
- Capa superior: presenta una interfaz agradable, limpia y regula a los usuarios.

Con respecto a su objetivo, el software debe poder escribir programas que se utilicen con archivos en distintos dispositivos sin modificar el programa para cada tipo de dispositivo. El S.O. debe administrar los dispositivos compartidos (discos) y los de uso exclusivo (impresora). El software de E/S se estructura en capas:

Manejadores de interrupciones: las interrupciones se ocultan en el S.O. Cada procesos que inicien una operación de E/S se bloquea hasta que termina la E/S y ocurra la interrupción, el procedimiento de interrupción realiza lo necesario para desbloquear el proceso que lo inicio.

Manejadores de dispositivos: aquí aparece todo el código que depende de los dispositivos. Este manejador acepta la solicitud que hace el software independiente del dispositivo y verifica su ejecución.

Software de E/S independientes de dispositivo: realiza las operaciones de E/S comunes a todos los dispositivos y proporcionar una interfaz uniforme del software a nivel usuario. Asocia el nombre simbólico de los dispositivos con el apropiado.

Software de E/S en el espacio del usuario: una pequeña parte consta de bibliotecas ligadas entre si con los programas del usuario. Otra categoría de este software es el **sistema de spooling**: es una forma de trabajar con los dispositivos de e /s de uso exclusivo (Ej. la impresora) en un sistema de multiprogramación. El proceso de usuario no abre los archivos de la impresora, se crea un proceso especial que no lo puede hacer, y se crea un directorio de spooling. Entonces un proceso genera todo el archivo por imprimir y lo coloca en el directorio de spooling; luego el proceso especial lo toma y lo tramite. Se evita que su proceso de usuario tenga un recurso mucho tiempo e incrementa la capacidad de usuarios del sistema.

5.4 DISCOS HARDWARE PARA DISCOS

Discos: Las ventajas de utilizar discos como almacenamiento en vez de la memoria principal son:

- Mayor capacidad de espacio de almacenamiento.
- Menor precio por bit.
- La información no se pierde al apagar la computadora.

Hardware para discos: los discos están organizados en cilindros, pistas y sectores. Tiene entre 8 y 32 sectores por pista. Una **búsqueda traslapada** es cuando un controlador realiza búsquedas en una o más unidades al mismo tiempo. los controladores pueden leer o escribir en una unidad y buscar en otra, pero no puede leer o escribir en dos unidades la mismo tiempo.

5.5 OPERACIÓN DE ALMACENAMIENTO DE DISCO DE CABEZA MÓVIL: los datos se graban en una serie de discos magnéticos o PLATOS. Un eje común de un disco gira a una velocidad de 4000 o más revoluciones por minuto. Se lee o escribe mediante la cabeza de lectura/ escritura, hay una por cada superficie del disco. La parte de la superficie del disco de donde se leerá o escribirá debe rotar hasta colocarse debajo o arriba de una cabeza de lectura/escritura, el tiempo de rotación desde la posición actual hasta la adyacente al cabezal se llama TIEMPO DE LATENCIA. Todas las cabezas de lectura - escritura están montadas sobre una *barra* o *conjunto de brazo móvil que se* puede moverse hacia adentro o hacia fuera, llamada operación de BUSQUEDA.

5.6 ALGORITMOS DE PROGRAMACIÓN DEL BRAZO DEL DISCO para mejorar el rendimiento del sistema se busca reducir el tiempo de búsqueda en el disco. Existen esquemas de que utilizan el manejador del disco, como:

- Algoritmo **primero en llegar primero en ser atendido (fcfs)**: mejora muy poco el tiempo de búsqueda, mientras el brazo realiza una búsqueda para una solicitud, otro proceso puede generar otras solicitudes.
- Algoritmo **primero la búsqueda mas corta:** se atiende primero la solicitud más cercana, reduciendo a la mitad el numero de movimientos del brazo que FCFS. El ingreso de nuevas solicitudes demorará a los antiguos. Si hay muchas solicitudes el brazo estará en la mitad y los más lejanos no se atenderán. La solución a éste problema es el algoritmo del elevador.
- **Algoritmo del elevador:** el movimiento del brazo va en la misma dirección hasta no haber más solicitudes y cambia de dirección. Ocasionalmente es mejor que el SSF pero generalmente es peor porque la cantidad de rendimiento del brazo será el doble de número de cilindros.

5. 7 PORQUE ES NECESARIA LA PLANIFICACION DE DISCOS: es necesaria porque en los sistemas de multiprogramación muchos procesos generan peticiones de e / s sobre discos, ya que las peticiones son mas rápidas que las atenciones se genera líneas de espera o colas para cada dispositivo; y para reduciré el tiempo de búsqueda de registros se ordena la cola de peticiones. Al planificar el deseo se examina las peticiones pendientes y se determina la forma más eficiente de servirlo, se analizan las opciones entre las peticiones en espera y se reordena la cola de peticiones. Los tipos más comunes de planificación son: OPTIMIZACION DE LA BUSQEUDA (para poca carga), OPTIMIZACION ROTACIONAL (latencia, para mucha carga).

5.8 CARACTERISTICAS DESEABLES DE LAS POLITICAS DE PLANIFICACION DE DISCOS: Criterios de categorización de las políticas de planificación son:

- *Capacidad de ejecución:* maximizar el número de peticiones servidas por unidad de tiempo.
- *Media del tiempo de respuesta:* minimizar el tiempo de respuesta.
- *Varianza de los tiempos de respuesta (predecibilidad):* minimizar la varianza.

5.9 OPTIMIZACION DE LA BUSQUEDA EN DISCOS: las estrategias más comunes de optimización de la búsqueda son:

- **Planificación fcfs (primero en llegar, primero en ser servido).** No hay reordenación de la cola de peticiones y no se analizan las posiciones entre las peticiones.
- **Planificación sstf (menor tiempo de busqueda primero).** El brazo del disco atiende la petición más cercana, no respeta el orden de llegada, favorece las pistas del centro.
- **Planificación scan.** el brazo del disco se desplaza sirviendo a todas las peticiones que encuentra a su paso y cambia de dirección al final t vuelve a servir.
- **Planificación scan de n-pasos.** Sirve las peticiones en espera cuando comienza igual que la SCAN, si llegan nuevas peticiones durante el recorrido los atiende al regresar.
- **Planificación c-scan (búsqueda circular).:** el brazo va del cilindro exterior al interior, sirviendo a las peticiones, al terminar vuelve al cilindro externo.

5.10 OPTIMIZACION ROTACIONAL EN DISCOS: en cargas pesadas, la probabilidad aumenta de que ocurran referencias al mismo cilindro. La optimización rotacional se utiliza en dispositivos de cabezas fijas y la estrategia utilizada es:
Algoritmo sltf (tiempo de latencia mas corto primero): el brazo del disco se sitúa en un cilindro, examina todas las peticiones sobre el cilindro y sirve primero a la que tiene el retraso rotacional mas corto.

5.11 CONSIDERACIONES DE LOS DISCOS SOBRE LOS SISTEMAS.
Cuando es útil la planificación de deseo? ¿Cuándo puede degradar el rendimiento?
La planificación de disco puede mejorar el rendimiento y eliminar el *embotellamiento* de grandes cargas de peticiones sobre pocos discos.
La planificación es efectiva en sistemas de tiempo compartido con un nivel alto de multiprogramación.
En procesos secuenciales de archivos secuenciales se afectan cilindros adyacentes, las búsquedas son cortas y no necesita planificación.

5.12 MANEJO DE ERRORES EN DISCOS: los errores mas comunes en discos son controlados por el manejador de discos.
- error de programación: Ej. solicitar un sector que no existe.
- error temporal en la suma de verificación: Ej. Provocado por el polvo en la cabeza.
- error permanente en la suma de verificación: Ej. Disco dañado físicamente.
- error de búsqueda: Ej. El brazo se envía al cilindro 6º pero va al 7º
- error del controlador: no acepta los comandos.

5.13 OCULTAMIENTO DE UNA PISTA A LA VEZ EN DISCOS: generalmente el tiempo de búsqueda supera el del rotacional. Una vez que se encuentra en el cilindro no importa si se lee un sector o toda la pista, en especial en dispositivos con **sensibilidad rotacional (rps).** Permite leer una pista en un tiempo de rotación. Hay manejadores que lo hacen por medio de un caché secreto de una pista a la vez o bien en su propio memoria.

5.14 DISCOS EN RAM: utilizan una parte de la memoria principal para almacenar los bloques. Tienen la ventaja del acceso instantáneo, no hay demora rotacional, son adecuados para almacenar programas o datos con accesos frecuentes.

5.16 RELOJES. o cronómetros. Sirven para la operación de sistemas de tiempo compartido. Registran la hora del día y evitan que un proceso monopolice la cpu.
Los **relojes** pueden ser **programables** y su frecuencia de interrupción se controla por el software.

5.17 TERMINALES: la parte independiente del dispositivo del S.O. y los programas del usuario no se tienen que escribir para cada tipo de terminal.

Tienen muchas formas distintas y el manejador de la terminal oculta estas diferencias (Ej. Terminal de impresora, de video). Las terminales pueden operar con una estructura central de buffer o con buffer exclusivos para cada terminal.

TEMA 6
BLOQUEOS

6.1 INTRODUCCION Y EJEMPLOS DE BLOQUEO (O INTERBLOQUEO): Un proceso dentro de un sistema de multiprogramación esta *bloqueado* si esta esperando por un evento determinado que no ocurrirá.
Cuando los recursos son compartidos entre usuarios se debe considerar la *prevención*, *habitación*, *detección* y *recuperación* del interbloqueo y la *postergación indefinida.* (un proceso aunque no este interbloqueado espera por un evento que probablemente no ocurrirá). A veces el precio de liberar el interbloqueo en sistemas es muy alto y permitirlo es catastrófico.
Los sistemas de cómputos hay recursos que solo se pueden utilizar por un proceso a la vez (impreso, unidad de cinta). El S.O. tiene la capacidad de otorgar temporalmente a un proceso el acceso exclusivo a un recurso o a varios.
Otro Ejemplo es el interbloqueo de un recurso simple en que "el bloqueo se produce en una espera circular de recursos que detiene completamente el tráfico y es necesario una intervención externa para poner orden y reestablecer la normalidad.
Inanición: envejecimiento de un proceso que no usa el procesador. Una solución es que el S.O. eleva la prioridad mientras espera por un recurso, evitando así la postergación indefinida del proceso, sin estar bloqueado o utilizando FIFO.

6.2 CONCEPTOS DE RECURSOS: el S.O. es un administrador de recursos. Los pueden ser "apropiativos" (Ej. CPU, memoria principal), y "no apropiativos" (unidades de cinta).
La apropiatividad es indispensable para el éxito de los sistemas computacionales multiprogramados. Algunos recursos pueden ser compartidos entre varios procesos o dedicados a procesos individuales.

6.3 BLOQUEOS Y CONDICIONES NECESARIAS PARA EL BLOQUEO: si un proceso necesita utilizar un recurso: lo solicita, después lo utiliza y por último lo libera. Si el recurso no esta disponible cuando lo solicita debe esperar. En algunos S.O. el proceso se bloquea automáticamente y se despierta cuando dicho recurso esta disponible. En otros S.O. la solicitud falla y el proceso debe esperar para luego intentar nuevamente.
Definición formal de bloqueo: un conjunto de procesos se bloquea si cada proceso del conjunto espera un evento que solo puede ser provocado por otro proceso del conjunto. Como todos los procesos estan esperando ninguno realizará un evento que pueda despertar a los demás miembros del conjunto. Todos los procesos esperaran por siempre.
Las condiciones necesarias para el bloqueo son (coffman):
Exclusión mutua: el proceso reclaman control exclusivo del recurso.
Espera por: los procesos mantienen los recursos que le fueron asignados mientras esperan por recursos adicionales.
No Apropiatividad: los recursos no pueden ser extraídos de los procesos que los tienen hasta su completo uso.
Espera circular: cadena circular de procesos en la que c / u mantiene a uno o mas recursos que son requeridos por el siguiente proceso de la cadena.

6.4 MODELACION DE BLOQUEOS: se muestran mediante gráficos dirigidos que tienen dos tipos de modos:
- Procesos (aparecen como círculos).
- Recursos (aparecen como cuadrados).
- R➔P "poseído"
- P➔R "en espera"

- Las *estrategias utilizadas para enfrentar los bloqueos* son:
  - Ignorar todo el problema.
  - Detección y recuperación.
  - Evitarlos mediante una cuidadosa asignación de recursos.
  - Prevenir mediante la negación de una de las condiciones necesarias.

6.5 AREAS PRINCIPALES EN LA INVESTIGACION DE BLOQUEOS: Principales aspectos:
- *Prevención del bloqueo:* se condiciona un sistema para que se elimine toda posibilidad de que se produzcan. Los resultado dan una pobre utilización de los recursos:
- *Evitación del bloqueo:* impone condiciones menos estrictas que en la prevención para lograr utilizar mejor de los recursos. Si el bloqueo aparece este se lo esquiva.

- *Detección del bloqueo:* Se utiliza en sistemas que permiten que estos ocurran, voluntariamente o no. Determinar si ha ocurrido un bloqueo: se debe detectar con precisión los procesos y recursos implicados, y se lo puede ELIMINAR.
- *Recuperación del bloqueo: d*espeja bloqueos de un sistema para continuar operando sin ellos; terminen los procesos estancados y se liberen los recursos. Se logra "extrayendo" (cancelando) procesos bloqueados.

6.6 EL ALGORITMO DEL AVESTRUZ O DE OSTRICH: Se ignora el problema. Algunos s. O. Soportan bloqueos que ni siquiera se detectan, porque que se rompen automáticamente o porque ni siquiera se detectan porque se rompe automáticamente o porque los usuarios prefieren un bloqueo ocasional, en vez de una regla que restringiera a todos los usuarios en el uso de los distintos tipos de recursos. El problema es que se paga un cierto precio para encarar el bloqueo en restricciones para los procesos y en el uso de recursos.

6.7 DETECCION DE BLOQUEOS: El s. O no intenta evitar los bloqueos, intenta detectar cuando han ocurrido y acciona para recuperarse después del hecho.
La detección del bloqueo consiste en determinar si existe o no un bloqueo e identificar que procesos y recursos implicados en el bloqueo.
Los algoritmos de detección implican una sobrecarga en tiempo de ejecución.
Detección de bloqueos de forma:
➔"Un recurso de cada tipo": No hay más de un objeto de cada clase de recurso, si la gráfica contiene uno o mas ciclos existe un bloqueo; y cualquier proceso que forme parte del ciclo está bloqueado.
➔ Varios recursos de cada tipo: se basan en matrices para detectar un bloqueo entre "N" procesos y "M" recursos de cada clase.
E: es el vector de recursos existentes.
A: es el vector de recursos disponibles (no asignados).
MATRIZ C: asignación actual.
MATRIZ R: solicitudes
Se comparan los vectores con las matrices, estarás bloqueado si lo que se solicita es mayor que lo disponible.
Cuando buscar los bloqueos:

- Cada vez que se solicita un recurso (podría sobrecargar el sistema).
- Verificar cada k minutos.
- Verificar cuando el uso de la cpu baje de cierto valor fijo:

6.8 RECUPERACION DE BLOQUEOS: Para romper el bloqueo de un sistema hay que anular una o mas de las condiciones necesarias para el bloqueo. Normalmente, varios procesos perderán algo de lo realizado hasta el momento.
Factores que dificultan la recuperación de bloqueo:

- Puede no estar claro si el sistema está o no bloqueado.
- Muchos sistemas tienen limitaciones para suspender un proceso y luego reanudarlo
- Los procedimientos de suspensión/reanudación implican sobrecarga.

La recuperación se realiza

- Retirando forzosamente a un proceso
- Reclamando sus recursos
- Permitir que los procesos restantes finalicen

Formas de recuperación ante bloqueos son:
Recuperación mediante la apropiación. *T*omar un recurso temporalmente de un proceso y dárselo a otro proceso para que termine.
Recuperación mediante rollback. Se verifica periódicamente los procesos para ver si ocurrió un bloqueo, se graba su estado en un archivo para que más tarde se pueda mirar.
Recuperación mediante la eliminación de procesos. Es la forma mas sencilla de suspender un bloqueo, se elimina un proceso del ciclo o se elimina un proceso que no esta en el ciclo para liberar recursos que necesite un proceso del ciclo. Es preferible eliminar un proceso de compilación que se puede repetir sin problema, que un proceso de actualización de una base de datos.

6.9 EVASION DE BLOQUEOS: se supone que si un proceso solicita un recurso, lo hace solicitando todos al mismo tiempo, pero en la mayoria se solicita uno a la vez. El S.O. decide si otorgar un recurso es seguro o no, y asignarlo si es seguro.

El OBJETIVO es evitar el bloqueo eligiendo correctamente todo el tiempo, pero si necesita información de antemano.
Un ESTADO ES SEGURO si no está bloqueado y existe una forma de satisfacer todas las solicitudes pendientes.
Un algoritmo que intenta evitar los bloqueos es:

- **El algoritmo del banquero (de dijkstra).** Los clientes son los procesos, los créditos son los recursos y el banquero es el S.O.

Consiste en estudiar cada solicitud ver su otorgamiento conduce a un estado seguro y otorga la solicitud, de lo contrario lo pospone. Los procesos deben establecer sus necesidades totales de recursos antes de su ejecución (desventaja) (no utilizado en S.O. de reales)

6.10 PREVENCION DE BLOQUEOS: los bloqueos son imposibles si se grantiza que el menos una de las cuatro condiciones necesarias se satisface (enunciado de Revender).

Estrategias para evitar las condiciones de bloqueo:

- Cada proceso pide todos sus recursos requeridos de una sola vez.
- Si un proceso que mantiene sus recursos, se le niega una petición libera los recursos originales y sei fuera necesario pedirlos de nuevo junto con los recursos adicionales.

Algoritmos de prevención de bloqueos:

Prevención de la condición de exclusión mutua. Si ningún recurso se asignara de forma exclusiva a un solo proceso, no habrá bloqueos. Imposible de aplicar porque hay recursos no compatibles (Ej. la impresora).
Prevención de la condición "detenerse y esperar" o "espera por". Se evita que procesos que conservan recursos esperen mas recursos, no hay bloqueos. Se exige a todos solicitar sus recursos de antemano, pero algunos no saben.
Prevención de la condición de "no apropiación". Los recursos pueden ser retirados de los procesos que los retienen antes de la terminación de los procesos, pero se puede perder información.
Prevención de la condición de "espera circular". Un proceso solo utiliza un recurso en cada momento, si necesita otro recurso debe liberar el primero; los recursos son enumerados globalmente, y se solicita según el orden (creciente).

6.11 OTROS ASPECTOS:

Cerradura de dos fases. (sistema de bases de dato) consiste en:

- Primera fase: el proceso intenta cerrar todos los registros necesarios, uno a la vez.
- Segunda fase: se actualizan y se liberan las cerraduras.

Si durante la primera fase se necesita algun registro ya cerrado, libera todas lñas cerraduras y comienza en la primera nuevamente. No es aplicable en la realidad, podría haber actualizado, enviando mensajes, etc.
Bloqueos sin recursos. Puede ocurrir que dos procesos se bloqueen en espera de que el otro realice cierta acción.
Inanición. En sistemas dinám,icos permanentemente hay solicitudes de recursos, por lo que se necesitará aun criterio para decidir quién obtiene cual recurso y en que momento ya que esto podría ocacionar que ciertos procesos nunca lograran el servicio, aun son estar bloqueados, porque su privilegio en el uso de recurso a otro proceso. Se puede solucionar mediante el criterio de asignación de recursos FIFO.

6.12 TENDENCIAS DEL TRATAMIENTO DEL BLOQUEO: el bloqueo tendrá una consideración mayor en los S.O. porque:

- Orientación hacia la operación asincrónica en paralelo:
- Ignorancia a priori de los procesos de sus necesidades de recursos.
- Considerar los datos como un recurso. Incrementar la capacidad del S.O. para administrar mas recursos.

TEMA 7
INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS

7.1 INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS: A partir de la década de los ´80 aparecen dos avances tecnológicos fundamentales:

- Desarrollo de microprocesadores poderosos y económicos
- Desarrollo de redes de AREA LOCAL (LAN) de alta velocidad.

Esto es lo que permite la aparición de los SISTEMAS DISTRIBUIDOS que necesitan un software distinto al de los sistemas centralizados.
En los SISTEMAS DISTRIBUIDOS los usuarios pueden acceder a gran velocidad a los recursos computacionales de hardware y software distribuidos entre varios sistemas conectados.

7.2 VENTAJAS DE LOS SISTEMAS DISTRIBUIDOS CON RESPECTO A LOS CENTRALIZADOS: los sistemas tienden a la descentralización por la economía. Aquí no se aplica la economía de que si se paga el doble, se obtiene el cuádruplo de desempeño, con la tecnología del microprocesador la solución en costo es: limitarse a un grán número de CPU baratos reunidos en un mismo sistema.
La potencia de un sistema distribuido en "precio/desempeño" es mejor que la de un único sistema centralizado.

- *sistemas distribuidos*: diseñados para que muchos usuarios trabajen en forma conjunta.
- *sistemas paralelos*: diseñados para lograr la máxima rapidez en un único problema.

En los *sistemas distribuidos* existen varias cpu conectadas entre si y trabajan de manera conjunta. (Ciertas aplicaciones son distribuidas en forma inherente).
Otra ventaja es mayor confiabilidad al distribuir la carga de trabajo en muchas máquinas, la falla de uno no afecta a los demás. El sistema sobrevive como un todo.
Otra ventaja es el crecimiento incremental o por incrementos, podría añadirse procesadores al sistema, incrementando el poder de cómputo, pero no en breves lapsos de tiempo.

7.3 VENTAJAS DE LOS SISTEMAS DISTRIBUIDOS CON RESPECTO A LAS PC INDEPENDIENTES: los sistemas distribuidos satisfacen la necesidad de los usuarios de compartir ciertos datos (Ej. reserva de línea aérea). También se puede compartir otros recursos como programas y periféricos costosos (Ej. impresora láser).
Otra ventaja es lograr una mejor comunicación entre las personas (Ej. correo electrónico).
También hay mayor flexibilidad al distribuir la carga de trabajo entre las máquinas disponibles en la forma más eficaz para disminuir los costos. Los equipos distribuidos pueden no ser siempre PC.

7.4 DESVENTAJAS DE LOS SISTEMAS DISTRIBUIDOS: El principal problema es el software (diseño, implantación y uso). El S.O., lenguaje de programación y aplicaciones para estos sistemas es diferente según el criterio de cada uno.
Otro problema tiene que ver con las redes de comunicaciones (pérdida de mensajes, saturación). La ventaja de compartir datos tiene el problema de la seguridad.

7.5 CONCEPTOS DE HARDWARE: Los sistemas distribuidos constan de varias cpu, organizadas de diversas formas respecto de:

- La forma de interconectarlas entre si.
- Los esquemas de comunicación utilizados.

Esquemas de clasificación de los sistemas de cómputos con varias CPU:
Taxonomia de flynn: según el n° de flujo de instrucciones y el n° de flujos de datos (son características especiales).

- **sisd** *(un flujo de instrucciones y un flujo de datos)*. Posee un único procesador.

- **simd** *(un flujo de instrucciones y varios flujos de datos):* ordenar procesadores con una unidad de instrucción.
- **misd** *(un flujo de varias instrucciones y un solo flujo de datos)* Varias instrucciones, un flujo de datos. No se presenta en la actualidad.
- **mimd** *(múltiples instrucciones y múltiples datos)* Un grupo de PCs, cada una con su propio contador de programa, programas y datos. Todos los sistemas distribuidos son de este tipo.

La categoría MIND se divide en:

- **multiprocesadores**: poseen memoria compartida. Los distintos procesadores comparten el mismo espacio de direcciones virtuales.
- **multicomputadoras**: no poseen memoria compartida. (Ej. grupo de PC conectadas mediante red). Cada una de éstas categorías se puede clasificar según la arquitectura de la red de interconexión en:
  - *Esquema de bus:* existe una sola red, bus, cable u otro medio que conecta todas las maquinas:
  - *Esquema con conmutador:* no existe una sola columna vertebral de conexión. Hay múltiples conexiones, los mensajes de mueven a través de los medios de conexión, se decide la conmutación en cada etapa para dirigir el mensaje a lo largo de los cables de salida (Ej. sistema mundial telefonía pública).

Otra clasificación considera el acoplamiento entre los equipos:

- **Sistemas fuertemente acoplados:** el retraso al enviar un mensaje de una computadora a otra es corto y la tasa de transmisión es alta. Se los utiliza como sistemas paralelos. (Ej. multiprocesadores).
- **Sistemas débilmente acoplados:** el retraso de los mensajes entre las maquinas es grande y la tasa de transmisión es baja. Se los utiliza como sistemas distribuidos (Ej. multicomputadoras).

7.6 MULTIPROCESADORES CON BASE EN BUSES: constan de cierto n° de cpu conectadas a un bus común, junto con un modulo de memoria. Todos los elemntos precedentes operan en paralelo. Solo existe una memoria, la cual presenta la propiedad de la COHERENCIA: las modificaciones hachas por una CPU se reflejan de inmediata en las lecturas de la misma o de otra CPU. El problema de este esquema es que el bus tiende a sobrecargarse y el rendimiento a disminuir.
La solución es añadir una **memoria cache** de alta velocidad entre la cpu y el bus. El cache guarda las palabras de acceso reciente y todas las solicitudes pasan a través de la caché, si la solicitud está en la caché, esta responde a la CPU sin solicitarle nada al bus.
Un importante problema debido al uso de cache y otra CPU leer desde su caché la misma palabra y obtener distintas resultados.
Una solución consiste es diseñar la cache para que *cuando una palabra sea escrita al cache, también sea escrita a la memoria* **cache de escritura**.

7.7 MULTIPROCESADORES CON CONMUTADOR: El esquema de multiprocesador con base en buses es bueno hasta con 64 procesadores, pero al superarlo se utiliza otro método de conexión entre CPU y memoria.
1)- Se divide la memoria en módulos y son conectados a las cpu con un **conmutador de cruceta (cross-bar switch)**.
La ventaja es que muchas CPU pueden tener acceso a la memoria al mismo tiempo, aunque no a la mismo memoria simultáneamente. El problema es el alto numero de conmutadores (n CPU X m memorias), ya que en cada intersección está un conmutador del punto de cruce electrónico que el hardware puede abrir o cerrar. Este conmutador se cierra momentáneamente cuando una CPU desea a una memoria en particular.
2)- un esquema que utiliza menos conmutador es la REDOMEGA, ya que posee conmutadores 2X2 (i entradas y i salidas), eligiendo los estados adecuados. Cada CPU podrá tener acceso a cada memoria.
El problema de la red omega es el retraso.
3)- otro esquema es según sistemas jerárquicos, cada CPU tiene asignada cierta memoria local que su acceso será muy rápido a su propia memoria y mas lenta a la memoria de los demás CPU: es el esquema o máquina NUMA (acceso no uniforme a la memoria).

7.8 MULTICOMPUTADORAS CON BASE EN BUSES: Es un esquema sin memoria compartida, cada cpu tiene una conexión directa con su propia memoria local.
El problema es la forma en que las cpu se comuniquen entre si. El tráfico es solo entre una cpu y otra; su volumen de tráfico es menor. Son una colección de estaciones de trabajo en una lan (red de area local).

7.8 MULTICOMPUTADORAS CON CONMUTADOR: Cada cpu tiene acceso directo y exclusivo a su propia memoria particular. Las topologías mas comunes son:
1)- La *retícula:* se basan en las tarjetas de circuitos impresos.
2)- El *hipercubo:* es un cubo *n* – dimensional, donde cada vértice es una CPU y cada arista es una conexión. Ej. un cubo de dimensión 4 se lo puede considerar como dos cubos ordinarios, cada uno de ellos con P vértices y 12 aristas.

7.10 CONCEPTOS DE SOFTWARE: La imagen de un sistema es determinada por el software del S.O. y no por el hardware.
Los S.O. se clasifican en dos tipos:
• El software débilmente acoplado de un sistema distribuido. Permite que las máquinas y usuarios sean independientes entre si y facilita que interactúen cuando sea necesario, los equipos individuales se distinguen bien.
• El software fuertemente acoplado: las máquinas necesitan interactuar siempre, y los equipos individuales no se distinguen bien.
Combinando los distintos tipos de hardware distribuidos con software distribuidos se logran distintas soluciones. Pero no todas son importantes desde el punto de vista del usuario.

7.11 SISTEMAS OPERATIVOS DE REDES: Una forma es el **software débilmente acoplado en hardware débilmente acoplado;** es muy utilizada (Ej. una red de estaciones de trabajo conectados mediante una LAN).
Cada usuario tiene una estación de trabajo para su uso exclusivo, con su propio s. o. y los requerimientos se resuelven localmente. Un usuario puede conectarse de manera remota con otra estación de trabajo.
Cuando una o mas máquinas soportan al sistema de archivo global compartidos, que es accesible desde todas las máquinas, son denominadas SERVIDORES DE ARCHIVOS que aceptan solicitudes de los programas de usuarios.
Los programas se ejecutan en las máquinas clientes, las solicitudes se examinan, se ejecutan y se envía la respuesta.
No es necesario que todas las máquinas ejecuten el mismo S.O.
El s. o. de este tipo debe controlar las estaciones de trabajo, los servidores de archivo y encargarse de la comunicación entre los servidores. Si los clientes y servidores ejecutan diversos S.O., deben coincidir en el formato y significado de los mensajes que intercambian, para comunicarse deben tener el mismo protocolo.
Este esquema es un S.O. de red donde cada máquina ser rige por si mismas y hay pocos requisitos a lo largo del sistema.
NES (network file System): soportan sistemas heterogéneos y los equipos pueden ser también de hardware heterogéneos. Los aspectos más importantes son:
La arquitectura de NFS. Permite que una colección arbitraria de clientes y servidores compartan un sistema de archivo común. A veces todos los clientes y servidores están en una LAN pero esto no es necesario.
Protocolo de NFS; el objetivo del NFS es lograr soportar un sistema heterogéneo en donde los clientes y servidores ejecuten distintos S.O. en hardware diversos, y que la interfaz entre clientes y servidores esten bien definidas. NFS logra esta definiendo dos protocolos: “CLIENTE – SERVIDOR”
A)- Un **protocolo** es un conjunto de solicitudes que de clientes a servidores y las respuestas de servidores a clientes. Dice ¿Qué y cómo se solicita?

Un protocolo de NFS maneja el **montaje** de un directorio en su sistema, o la alternativa del AUTOMONTAJE, según pedido.
Si el cliente utiliza varios servidores en particular obtiene:
° Tolerancia a fallos.
° Mejora del rendimiento.
B)- El otro protocolo es para el acceso a directorios y archivos.
° Implantación de NFS: la implantación del código del cliente y el servidor es independiente de los protocolos NFS:
° NFS: solo trata del sistema de archivos, no hace referencia a otros aspectos, como la ejecución de un proceso.

7.12 SISTEMAS REALMENTE DISTRIBUIDOS: *NFS* es un ejemplo de *software débilmente acoplado en hardware débilmente acoplado*. Cada computadora ejecuta su propio s. o.; solo se dispone de un sistema compartido de archivos; el trafico cliente - servidor debe obedecer los protocolos NFS.
Las *multicomputadoras* son un ejemplo de *software fuertemente acoplado en hardware débilmente acoplado*: crea la ilusión de que toda la red de computadoras es un solo sistema de tiempo compartido, en vez de una colección de máquinas diversas.
Un sistema distribuido es una colección de maquinas sin memoria compartida, pero que aparece ante sus usuarios como una sola computadora.
Características:
- Debe existir un mecanismo de comunicación global entre los procesos, cualquier proceso intercambie información con otro.
- Debe existir un esquema global de protección.
- La administración de procesos debe ser igual en todas partes.
- Se debe tener una misma interfaz de llamadas al sistema en todas partes:
- Es necesario un sistema global de archivos.

7.13 SISTEMAS DE MULTIPROCESADOR CON TIEMPO COMPARTIDO: Corresponde a software fuertemente acoplado en hardware fuertemente acoplado.
Los multiprocesadores operan con un sistema de tiempo compartido, pero son varias CPU en vez de una sola.
° HIPS: mide la velocidad del procesador. 1MPS= 1.000.000 de intrucciones por segundo.
° CARACTERISTICAS: existe una sola cola de ejecución contenido en la memoria compartida.
Los programas de los procesos estan en la memoria compartida y también el S.O. El planificador de procesos del S.O. se ejecutan como una región crítica, con ello se evita que dos CPU elijan el mismo proceso para su ejecución inmediata. Ninguna CPU tiene memoria local, todos los programas se almacenan en la memoria global compartida.

7.14 ASPECTOS DEL DISEÑO: Los aspectos claves en el diseño de s. o. distribuidos son:
° Transparencia: imagen de un único sistema.
° Flexibilidad: en cuanto a la estructura del S.O.
° Confiabilidad: si una máquina falta, otra hará su trabajo.
° Desempeño: muchas actividades en ejecución paralela.
° Escalabilidad: evitar los cuellos de botella con muchos usuarios.

7.15 TRANSPARENCIA: Se debe lograr la imagen de un único sistema. Los usuarios deben percibir que la colección de maquinas conectadas son un sistema de tiempo compartido de un solo procesador, entonces el sistema es **transparente**.
La transparencia (*desde el punto de vista de los usuarios)*, se logra cuando se satisfacen los pedidos de los usuarios con ejecuciones en paralelo en distintas maquinas; se utilizan una variedad de servidores de archivos.

La transparencia *desde el punto de vista de los programas* significa diseñar la interfaz de llamadas al sistema de modo que no sea visible la existencia de varios procesadores.

Tipos de transparencia

- **de localización:** los usuarios no pueden indicar la localización de los recursos.
- **de migración:** los recursos se pueden mover sin cambiar sus nombres.
- **de replica:** usuarios no pueden indicar el n° de copias existentes.
- **de concurrencia:** varios usuarios pueden compartir recursos de manera automática.
- **de paralelismo:** las actividades pueden ocurrir en paralelo sin el conocimiento de los usuarios.

7.16 FLEXIBILIDAD: con respecto a la estructura de los sistemas distribuidos, existen:

- **núcleo monolítico:** proporciona el sistema de archivos, toda la administración de procesos, gran parte del manejo de las llamadas al sistema. cada máquina ejecuta un núcleo que proporciona la mayoría de los servicios.
- **micro núcleo (microkernel):** el núcleo proporciona lo menos posible, es más flexible. Proporciona solo cuatro servicios mínimos: mecanismos de comunicación entre procesos, cierta administración de la memoria, una cantidad limitada de planificación y administración de procesos de bajo nivel y E/S de bajo nivel.

7.17 CONFIABILIDAD: tolerancia a fallos. El objetivo de un sistema distribuido es que si una máquina falla, otra debe encargarse del trabajo. La confiabilidad teórica del sistema es el "OR" voléanos de los componentes, pero a veces se requiere que ciertos servidores estén en servicio simultáneamente para que todo funcione.
Otro aspecto es la DISPONIBILIDAD que es la fracción de tiempo en que se usa el sistema.
Otro es la SEGURIDAD, es decir, que los archivos y otros recursos deben protegerse para evitar el mal uso.
Otro es la TOLERACNIA A FALLOS, la falla de oculta y se hace una recuperación transparente para el usuario, aunque haya degradación de performance.

7.18 DESEMPEÑO: tener muchas actividades en ejecución paralela. Tener en cuenta el desempeño de respuesta, rendimiento (número de trabajos por hora, uso del sistema, velocidad de proceso).

7.19 ESCALABILIDAD: se debe evitar los cuellos de botellas con miles de usuarios. Se utilizan algoritmos y componentes descentralizados.

TEMA 8
COMUNICACION EN LOS SISTEMAS DISTRIBUIDOS

8.1 INTRODUCCION A LA COMUNICACION EN LOS SISTEMAS DISTRIBUIDOS: la diferencia entre un sistema distribuido y un sistema de un único procesador es la comunicación entre los procesos. En un sistema de un único procesador es la comunicación entre los procesos. En un sistema de un único procesador la comunicación es mediante la existencia de la memoria compartida. En un sistema distribuido no existe la memoria compartida. Los procesos para comunicación se operan a reglas "PROTOCOLOS". Estos protocolos en un área amplia toman la forma de varias capas cada una con su meta y reglas. Los mensajes se intercambian de muchas formas. (Ej. llamadas a un procedimiento remoto).

8.2 PROTOCOLOS CON CAPAS: Debido a la ausencia de memoria compartida, toda la comunicación se basa en la *transferencia de mensajes*.

- Este modelo **OSI (modelo de referencia para interconexión de sistemas abiertos)** esta diseñado para permitir la comunicación de los **sistemas abiertos** (con los preparados para comunicarse con otro sistema abierto mediante *reglas estándar);* se establece el formato, contenido y significados de los mensajes recibidos y enviados, constituyen los PROTOCOLOS (son acuerdos en el formato que debe desarrollarse la comunicación).

El modelo OSI distingue dos tipos de **protocolos**:

- **Orientados hacia las conexiones:** antes de intercambiar los datos, el emisor y el receptor establece en forma explícita de antemano la conexión, se dice el protocolo al usar y al terminar su conexión. (Ej. teléfono).
- **Sin conexión:** no se configura de antemano, el mismo transmite el primer mensaje cuando está listo (Ej. depositar una carta en un buzón).

Cada capa proporciona una interfaz con la otra capa por encima de ella, esta interfaz es un conjunto de operaciones que define el servicio que las capas ofrecen a sus usuarios. Cada protocolo utiliza la información de su capa. Cada protocolo de capa se puede cambiar, confiere gran flexibilidad.

8.3 INTRODUCCION AL MODELO CLIENTE - SERVIDOR (C - S): El modelo de la OSI tiene un problema: el costo, y el envío de mensaje. El modelo OSI no estructurado al sistema distribuido; lo hace el modelo c-s (cliente/servidor) como un grupo de procesos en cooperación que ofrecen servicios a los usuarios "SERVIDORES" y un grupo de procesos usuarios "CLIENTES". Este modelo se basa en un protocolo solicitud/respuesta; es sencilla y sin conexión, y no como OSI o TCP/IP (orientado a la conexión).
El cliente envía un mensaje de solicitud el servidor pidiendo un servicio. El servidor ejecuta el requerimiento y regresa los datos solicitados o un código de error. No se establece una conexión seno hasta que se lo solicite. Las capas de protocolo son más cotosas y más eficientes.

8.4 DIRECCIONAMIENTO EN C – S: Para que un cliente pueda enviar un mensaje a un servidor, debe conocer la dirección de este. Algunos esquemas de direccionamiento se basa en:

- La dirección de la máquina donde se envía el mensaje.
- *Identificar los procesos* destinatarios en vez de a las maquinas.
- Se asigna a cada proceso una única dirección que no tiene número de máquina.
- Se deja que cada proceso elija su propio identificador en un espacio de direcciones grande y disperso.
- Se utiliza hardware especial cada proceso elige su dirección en forma aleatoria.

8.5 PRIMITIVAS DE BLOQUEO VS. NO BLOQUEO EN C – S: Las primitivas de transferencia de mensajes son las **primitivas de bloqueo o primitivas sincronas:** El proceso emisor se suspende (se bloquea) mientras se envía el mensaje. El proceso receptor se suspende mientras se recibe el mensaje.
Otras son las **primitivas sin bloqueo o primitivas asíncronas:** el proceso emisor no se suspende, continua su cómputo paralelamente con la transmisión del mensaje, pero no se sale cuando termina la transmisión para poder utilizar el buffer de forma segura.
Generalmente se considera que las desventajas de las primitivas asíncronas no compensan las ventajas del máximo paralelismo que permiten lograr. A las primitivas de envío se las conoce como SEND y a los de receptores RECEIVE.

8.6 PRIMITIVAS ALMACENADAS EN BUFFER VS. NO ALMACENADAS: Las primitivas no almacenadas significan que una dirección se refiere a un proceso específico. Se puede mantener el mensaje un instante en el núcleo (Ej. para mensajes que llegan antes), de esta manera se intenta que los mensajes no se pierdan.
Las primitivas con almacenamiento en buffer utilizan buzón que es una estructura de datos donde se colocan los mensajes que llegan con una dirección perteneciente a un proceso específico. El problema que los buzones son infinitas y se pueden llenar, por esto el núcleo tendrá que decidir si mantener el mensaje por un momento o descártalo. Una solución consistirá en no dejar que un proceso envíe un mensaje si necesita espacio para su almacenamiento en el destino.

8.7 PRIMITIVAS CONFIABLES VS. NO CONFIABLES: se supone que el mensaje enviado será siempre recibido pero se puede perder por muchas causas cuando un cliente envía un mensaje se lo suspende hasta que el mensaje ha sido enviado, pero no hay garantía que el mensaje se ha entregado.
Una solución consistirá en volver a definir la semántica del SEND para hacerlo no confiable, de esta manera el sistema no garantiza la entrega del mensaje, sino que lo deja en manos del usuario. Otro método exige que el núcleo de la maquina receptora envíe un reconocimiento al núcleo de la maquina emisora. Otra solución sería que la misma respuesta funcione como un reconocimiento.

8.8 IMPLANTACION DEL MODELO C – S: Las principales opciones de diseño son:
° Direccionamiento: número de máquina, dirección de procesos entre otros.
° Bloqueo: primitivas con bloqueo. con copia al núcleo o con interrupciones.
° Almacenamiento en buffer: no usar el almacenamiento en buffer, descartar los mensajes inesperados. mantener temporal de los mensajes inesperados o Buzones.
° Confiabilidad: No confiable. //Solicitud - reconocimiento - respuesta - reconocimiento. //Solicitud - respuesta - reconocimiento.

Se combinan todas estas formando diseños distintos.
Otro aspecto de la implementación es el protocolo subyacente utilizado en la comunicación C-S.
Los principales tipos de paquetes son los siguientes:

- **REQ:** Solicitud de cliente a servidor por un servicio.
- **REP:** Respuesta de servidor a cliente.
- **ACK:** Reconocimiento de cualquiera de ellos a algún otro.
- **AYA**: ¿estas vivo?. De cliente a servidor.
- **IAA:** Estoy vivo. De servidor a cliente.
- **TA:** Intenta de nuevo. De servidor a clientes. No hay espacio.
- **AU**: Dirección desconocida. de servidor a cliente.

8.9 LLAMADA A UN PROCEDIMIENTO REMOTO (RPC): es una forma de intercambiar mensajes en un sistema distribuido.
El problema del modelo C-S es que la comunicación es mediante la E/S con los procedimientos SEND/RECEIVE que realizan la E/S.
Otra alternativa fue permitir a los programas que llamasen a procedimiento localizado en otras máquinas.
Cuando un proceso A que llama un procedimiento en otra máquina B se suspende, se ejecuta el procedimiento en esta máquina B. la información se transporta de un lado otro mediante parámetros y regresa en el resultado del procedimiento.
El programador no se preocupa de una transferencia o de la E/S; a esto se llama LLAMADA A PROCEDIEMITNO (REC). El procedimiento que hace la llamada y el que recibe se ejecutan en máquinas diferentes.

8.10 OPERACION BASICA DE RPC: Una llamada convencional a un procedimiento (que opera en una sola máquina) es: el programa llamador se carga en la memoria, después que la lectura termine su ejecución, se coloca el valor de regreso en un registro, se elimina la dirección de regreso, se transfiere de nuevo el control al llamador y éste determina los parámetros de la pila y regresa a su estado original.
La idea es que la RPC se aparezca lo mas posible a una llamada (ser transparente). El procedimiento que hace la llamada no debe ser conciente de que el procedimiento llamado se ejecuta en otra maquina o viceversa.
Pasos de una RPC:

- El procedimiento cliente llama al STUB del cliente de la manera usual.
- El STUB del cliente construye un mensaje y hace un señalamiento al núcleo.

- El núcleo envía el mensaje al núcleo remoto.
- El núcleo remoto proporciona el mensaje al STUB del servidor.
- El STUB del servidor desempaca los parámetros y llama al servidor.
- El servidor realiza el trabajo y regresa el resultado al STUB.
- El STUB del servidor empaca el resultado en un mensaje y hace un señalamiento al núcleo.
- El núcleo remoto envía el mensaje al núcleo del cliente.
- El núcleo del cliente da el mensaje al STUB del cliente.
- El STUB desempaca el resultado y regresa al cliente.

De esta manera se ha convertido la llamada local del procedimiento cliente al STUB del cliente, en una llamada local al procedimiento servidor.

8.11 TRANSFERENCIA DE PARAMETROS EN RPC: El empacamiento de parámetros en un mensaje se llama **ordenamiento de parámetros.** El mensaje también contiene el nombre o nº del procedimiento por llamar. Cuando el mensaje llega al servidor el resguardo STUB le examina para ver cuando el procedimiento necesita y lo lleva a cabo. Los elementos del mensaje son verificados del procedimiento y parámetros.

8.12 CONEXION DINAMICA (DYNAMIC BINDING) EN RPC: El tema es la forma en que el cliente localiza al servidor.
Un método consiste en integrar dentro del código del cliente la dirección (en la red) del servidor. El problema es que si se cambia el servidor se tiene que volver a compilar el programa.
Una solución es la **conexión dinámica** para que concuerden los clientes y los servidores.
Se inicial especificando al servidor, cuando el servidor inicia su ejecución envía un mensaje a un programa CONECTOR para darle a conocer su existencia (registro del servidor).
El servidor puede cancelar su registro con el conector si no va a prestar servicio.
El cliente localiza al servidor enviando un mensaje al conductor especificando lo que necesita. Cuando el cliente llama a un procedimiento por primera vez si existe un servidor adecuado al conector puede ser un cuello de botella para muchas cargas de trabajo.

8.13 SEMANTICA DE RPC EN PRESENCIA DE FALLOS: El objetivo de RPC es ocultar la comunicación al hacer que las llamadas a procedimientos remotos se parezcan a las llamadas locales.
El problema es cuando aparecen los errores porque las diferencias entre las llamadas locales y remotas no son fáciles de encubrir.
Situaciones:
° **El cliente no puede localizar al servidor.** Puede estar inactivo el servidor, puede utilizar nuevas versiones incompatibles con el resguardo del cliente.
° **Perdida de mensajes de solicitud.** El núcleo inicializa un cronometro al enviar la solicitud. Si el tiempo termina y no regresa una respuesta se vuelve a enviar el mensaje, o puede que el servidor este inactivo.
° **Perdida de mensajes de respuesta.** Se utiliza un cronometro, si no llega una respuesta se vuelve a enviar la solicitud, pero el núcleo del cliente no sabe porque no hubo respuesta. Si incrementa las solicitudes para que el núcleo del servidor pueda diferenciar entre original y retransmisión.
° **Fallos del servidor.** Antes de recibir una solicitud (inactivo) o antes de disponer de una solicitud. Soluciones para que el núcleo distinga entre estas dos:
Semántica al menos una: esperar hasta que el servidor vuelva a arrancar o intente realizar de nuevo las operaciones. Garantiza que las RPC se ha realizado al menos una vez, pero es posible que se realice mas veces.
Semántica a lo mas una: no se reintenta y se informa del fallo. Garantiza que la RPC se realiza a lo mas una vez, pero es posible que no se realice ni una vez.
Semántica de no garantizar nada: cuando el servidor falla, el cliente no obtiene ayuda. La RPC se puede realizar un número de veces que va desde "o" hasta "n".
Semántica de exactamente una: Es la solución deseable pero generalmente no existe forma de garantizar esto.
Aquí se diferencian los sistemas con un único procesador y los sistemas distribuidos porque con un único procesador si falla el servidor falla el cliente y la recuperación no es posible ni necesaria, pero con sistemas distribuidos si es posible y necesario.
° **Fallos del cliente:** se genera una solicitud de trabajo o cómputo que al fallar ya nadie espera, "HUERFANO". Esto genera problemas como desperdicios de ciclo de CPU, bloqueo de archivos, apropiación de recursos.
Soluciones a cómputos huérfanos.

Exterminación. Se crea un registro que indica lo que va a hacer el resguardo del cliente antes de que emita la RPC. Luego del re arrancar se elimina el huérfano. Hay sobrecarga de E/S.

Reencarnación: se divide el tiempo en épocas. Cuando un cliente re arranca envía un mensaje a todos las máquinas declarando el inicio de una nueva época. Al recibir este mensaje se elimina todos los cómputos remotos.

Reencarnación sutil: cuando llega un mensaje de cierta época, la máquina verifica si tiene cómputos remotos, si no se localiza al poseedor se elimina el cómputo.

Expiración: si cada RPC se le da un tiempo para realizar su trabajo, si pasa se pasa el tiempo antes de re arrancar todos los huérfanos desaparecidos.

8.14 ASPECTOS DE LA IMPLANTACION EN RPC: el desempeño o performance es fundamental en los sistemas distribuidos y depende de la velocidad de la comunicación y la velocidad depende de la implantación.

- **Protocolos RPC.** Se elige entre un protocolo *orientado a la conexión* o un *protocolo sin conexión* entre cliente y servidor.

Otro aspecto importante es longitud del paquete y el mensaje:

° Realizar una RPC tiene un alto costo por la cantidad de datos que se envían.

° El protocolo y la red deben permitir transmisiones largas para minimizar el mínimo de RPC.

Otra opción es el protocolo estándar de propósito general o alguno especifico en RPC, como el IP:

- **Reconocimientos.** Cuando los RPC son grandes se deben dividir en muchos paquetes pequeños, surge el problema del reconocimiento. Los paquetes se reconocerán individualmente o grupalmente. Estrategias para reconocer:
    - protocolo detenerse y esperar: enviar y esperar el reconocimiento, luego enviar el otro.
    - Protocolo de chorro: envía todo como puede y se reconoce todo el conjunto.

  Cuando el paquete elige a un receptor que no lo puede aceptar hay un ERROR DE EJECUCIÓN y el paquete se pierde.
- **Ruta critica.** Es la serie de instrucciones que se ejecutan con cada RPC. Es importante determinar en que parte de la ruta crítica se ocupa la mayor parte del tiempo que demanda la RPC, que depende del tiempo de RPC y la cantidad de datos.

En RPC con transporte mínimo la mayor parte del tiempo se ocupa en el cambio de contrato, en RPC con transporte de 1Kb o más la mayor parte del tiempo se ocupa en el tiempo de transmisión.

- **Copiado:** El número de veces que se debe copiar un mensaje varía según el hardware, software y el tipo de llamada.
- **Manejo del cronometro:** Se inicializa al enviar un mensaje y se espera una respuesta, si la respuesta no llega se re transmite el mensaje. Para el establecimiento de un cronómetro se requiere construir una estructura de datos que especifique el momento en que el cronómetro debe detener y la acción a realizar, cuando eso no sucede.

8.15 AREAS DE PROBLEMAS EN RPC: La RPC mediante el modelo C-S se utiliza como base de los s. O. Distribuidos. La idea es que la RPC sea ***transparente****:* pero la mayoría no cumplen. Algunos de los problemas son:

- **Variables globales**: no permiten la transparencia en la implementación.
- **Lenguajes débilmente tipificados**: Ej. “c”, no se puede determinar su tamaño.
- No siempre se decide el tipo de parámetros.

8.16 COMUNICACION EN GRUPO: La comunicación de RPC no es solo entre dos procesos (cliente/servidor), a veces es entre varios procesos. Por ejemplo, un grupo de servidores de archivos ofrecen un único servicio de archivo tolerante a fallos. El cliente envía un mensaje a todos los servidores para garantizar la ejecución de la solicitud aunque alguno falle. La RPC no puede controlar la comunicación de un servidor con muchos receptores, a menos que realice RPC con cada uno en forma individual.

Un grupo es una colección de procesos que actúan junto en un sistema o de alguna forma determinada por el usuario.

La propiedad fundamental de todos los grupos es que cuando un mensaje se envía al propio grupo, todos los miembros lo reciben.

La comunicación es UNO-MUCHOS (un emisor, muchos receptores). Los grupos son dinámicos, se crean y se destruyen; un proceso se une a un grupo o deja a otro, un proceso puede ser miembro de varios grupos a la vez.

Implementar la comunicación en grupo depende del hardware y se llama “MULTITRANSMISIÓN”: cuando las máquinas que responden a la dirección.

Otra es la **transmisión simple** para redes que no soportan multitransmisión. Significa que los paquetes se entregan a todas las máquinas y cada una debe verificar si va a dirigirse e ella o lo descarta.

Otra solución es implantar la comunicación en grupo por **paquetes individuales** a c / u de los miembros del grupo por parte del emisor, que es la **uní transmisión**.

8.17 ASPECTOS DEL DISEÑO DE LA COMUNICACION EN GRUPO: Existen posibilidades como:

- Almacenamiento en buffers vs. El no almacenamiento.
- Bloqueo vs. No bloqueo.

Grupos cerrados vs. grupos abiertos. En los **grupos cerrados s**olo los miembros del grupo se envían mensajes entre ellos, los extraños solo pueden enviar a un miembro de grupo en la individual.
En los **grupos abiertos c**ualquier proceso del sistema puede enviar a cualquier grupo.
El grupo cerrado se usa para el procesamiento paralelo. (Ej. un grupo de procesos que trabajan de manera conjunta, tienen su propio objetivo y no interactúan con el mundo exterior).

Grupos de compañeros vs. Grupos jerárquicos. En los GRUPOS DE COMPAÑEROS todos los procesos son iguales, no hay jerarquías. Es simétrico y no tiene un punto de fallo, el grupo puede continuar aunque algunos de los procesos fallen. Al tomar decisiones hay retrasos por la comunicación entre todos.
En los GRUPOS JERARQUICOS un proceso es coordinador y todos los demás trabajadores, cualquier solicitud que se genere externamente o en un trabajador se envía al coordinador y decide cual trabajador es el más adecuado para llevarlo a cabo y se lo envía. La pérdida del coordinador paraliza el grupo. En condiciones normales, toma decisiones sin molestar a los demás procesos.

Membresía del grupo. La comunicación en grupo requiere crear y eliminar grupos. Unir y separar procesos a grupos. Se puede tener un **servidor de grupos** al cual enviar todas las solicitudes, es fácil y eficiente; para la **administración centralizada** de grupo representa un punto de fallo.
Otra posibilidad es la **administración distribuida** de las membresías de grupo, para anunciar un proceso su presencia a un grupo, o al dejarlo debe enviar un mensaje, para esto la E/S al grupo debe sincronizarse.
A veces faltan tantas máquinas que el grupo no puede funcionar, se necesita un protocolo para reconstruir al grupo, algún proceso tomará la iniciación. El protocolo resolverá la situación si dos o mas procesos quieren al mismo tiempo reconstruir el grupo.

Direccionamiento al grupo. Los grupos deben poder direccionarse, al igual que los procesos. A cada proceso se le da una dirección única, o se le pide al emisor una lista de los destinos (Ej. lista de direcciones IP), pero no es transparente.
Otro método es el direccionamiento de predicados, el mensaje se envía al grupo que contiene un predicado a evaluar, si es verdadero se acepta el mensaje y si es falso se descarta el mensaje.

Primitivas Send Y Receive. El envío de un mensaje a un grupo no puede ser una llamada a un procedimiento ya que con la comunicación en grupo existirá en potencia *n* respuestas diferentes. Se utilizan llamadas explicitas para el envío y recepción (modelo de un solo sentido). Uno de los parámetro que indica el destino y el segundo apunta al mensaje por enviar.

Atomicidad. Los sistemas de comunicación en grupo están diseñados para que los mensajes enviados al grupo lleguen correctamente a todos los miembros o a ninguno de ellos.
La propiedad de "todo o nada" en la entrega se llama **atomicidad** o **transmisión atómica.** Facilita la programación de los sistemas distribuidos. La única forma de garantizar de que cada destino recibió todos sus mensajes es que envíe de regreso su reconocimiento al recibir.

Ordenamiento de mensajes. El ordenamiento de mensajes para el envío de mensajes en la comunicación en grupo es importante. La mejor garantía es la entrega inmediata de todos los mensajes en el orden en que se enviaron.

Grupos traslapados. Un proceso puede ser miembro de varios grupos a la vez. Se puede generar cierta inconsistencia. El problema es que Existe un tiempo global dentro de cada grupo y No existe coordinación entre varios grupos.

Escalabilidad. Se debe considerar como funcionaran los algoritmos cuando:

- Hay Grupos con centenas o miles de miembros.
- Centenas o miles de grupos.

- Utilización de varias LAN y compuertas conectadas
- Diseminación de grupos en varios continentes.

TEMA 9 :"SINCRONIZACION EN SISTEMAS DISTRIBUIDOS"

INTRODUCCION A LA SINCRONIZACION EN SISTEMAS DISTRIBUIDOS.
Además de la comunicación también es importante la forma en que los procesos cooperen y se sincronizan entre si (Ej. forma de implementar las regiones criticas ; forma de asignar recursos en sistemas distribuidos).
Los problemas de las regiones criticas, exclusión mutua y la sincronización, generalmente se resuelven en sistemas de una sola CPU con métodos como los semáforos y los monitores se basan en la memoria compartida, que no se aplican en sistemas distribuidos.
Otro problema es el tiempo y la forma de medirlo, en los modelos de sincronización.

SINCRONIZACION DE RELOJES: para los algoritmos distribuidos, la información se distribuyen entre varias maquinas, los procesos toman las decisiones en forma local .debe evitarse un único punto de fallo en el sistema. No existe un reloj común del tiempo global, es inaceptable reunir toda la información en un solo lugar para su procesamiento
Lograr sincronización sin centralización. En un sistema distribuido no es importante poner de acuerdo a todos las maquinas en la hora.

- La falta de sincronización en los relojes puede ser problemáticas en procesos dependientes del tiempo
- se requieren un acuerdo global del tiempo
-

RELOJES LOGICOS: las computadoras poseen un circuito para el registro del tiempo llamado "DISPOSITIVO RELOJ", en un cronometro consistente en un cristal de cuarzo de precisión sometida a una tensión eléctrica que oscila con una frecuencia definida. Se puede programar un cronometro para que genere una interrupción x veces por segundo, a cada interrupción se llama "MARCA DE RELOJ"
Para varias computadoras cada una con sus relojes es imposible garantizar que los cristales de maquinas distintas oscilan a igual frecuencia.
La diferencia entre los valores del tiempo se llama "DISTORSION DE RELOJ"; y que puede generar fallos en los programas dependientes del tiempo.
Si dos procesos no interactúan no es necesario que los sus relojes este sincronizados lo importante no es que los procesos estén de acuerdo en la hora; si no que coincidan en el orden en que ocurren los eventos.
Para los relojes lógicos lo que importa es la consistencia interna de los relojes, no su cercanía el tiempo real (oficial).

RELOJES FISICOS: Los relojes físicos deben ser iguales (están sincronizados) y no deben desviarse del tiempo real (Ej. en los sistemas de tiempo real) y es importante la hora real del reloj.
En ciertos sistemas se precisa mas de un reloj físico externo y se deben sincronizar entre si y con los relojes del mundo real.
La medición del tiempo real con alta precisión no es sencilla.
Antiguamente el tiempo se media astronómicamente; el día solar es el intervalo entre dos tránsitos consecutivos del sol El segundo 1/86400 de un día solar. el segundo es el tiempo que tarda el átomo de cesio para hacer 9000 millones de transacciones aproximadamente .La oficina internacional de la hora en Paris recibe indicaciones de 50 relojes atómicos en el mundo y calculo el tiempo atómico internacional (TAI) y de esto surge el tiempo coordinado universal (UTC).

El instituto nacional del tiempo estándar de E.E.U.U. y otros países , operan estaciones de radio de onda corta o satélites de comunicaciones, transmiten pulsos UTC con regularidad (Cada segundo) y conociendo la posición del emisor y del receptor se debe compensar el retraso de la propagación de la señal , si se recibe la señal , si se recibe la señal por moden se compensa y se obtiene el tiempo con precisión.

ALGORITMO PARA LA SINCRONIZACION DE RELOJES: si la maquina tiene receptor de UTC todas deben sincronizarse con ella .Si no tiene, cada maquina eleva el registro de su tiempo y se debe mantenerse el tiempo de todas lo mas cercano posible.
ALGORITMO DE CRISTIAN: es para sistemas donde una maquina tiene un receptor de UTC .
El objetivo es sincronizar todas la maquinas con ella cada maquina envía un mensaje Del servidor para solicitar el tiempo actual. El despachador del tiempo responde .Para mejorar la precisión se toman varias mediciones y se promedian.
ALGORITMO DE BERKELEY:en el algoritmo de cristian el servidor de tiempo es pasivo acá es activo. Realiza un muestreo periódico de todas las maquinas para preguntarles el tiempo, y con la respuestas calcula el tiempo promedio- Es adecuado cuando no hay un receptor UTC .Indica a las maquinas que avancen o disminuyan
ALGORITMO POR PROMEDIO: Los anteriores algoritmos centralizados, este es descentralizado .Divide el tiempo en intervalos de resincronizacion de longitud fija .Al inicio de cada intervalo cada maquina transmite el tiempo actual según su reloj y los transmisiones no serán simultaneas porque difieren las velocidades de los relojes, se reciben todas y se promedian los valores.
VARIAS FUENTES EXTERNAS DE TIEMPO: los sistemas que requieren una sincronización los distintos intervalos de tiempo de las distintas fuentes de tiempo, y determinan la intersección, se compensan los retrasos y las velocidades de los relojes.

EXCLUSION MUTUA: cuando un proceso debe leer o actualizar ciertas estructuras de datos compartidas entre una región critica para lograr la exclusión mutua y garantizar que ningún otro proceso utilice las estructuras de datos del mismo tiempo .
En sistemas monoprocesadores las regiones criticas se protegen con semáforos, monitores pero en sistemas distribuidos es mas difícil.
ALGORITMO CENTRALIZADO: se logra la exclusión mutua similar a la forma que se lleva a cabo en un sistema monoprocesador .se elige u proceso coordinador y cuando un proceso sea ingresar a la región critica envía un mensaje de solicitud del coordinador para solicitar servicio, si no hay otro otorga el permiso, el proceso entra.
Si un proceso pide permiso para entrar a una región critica ya asignada a otro proceso, el coordinador no otorga el permiso y encola los pedidos.
En su esquema sencillo y justo con pocos mensajes de control pero el coordinador es un cuello de botella y si falla se bloquean los procesos en espera de una respuesta de acceso
ALGORITMO DISTRIBUIDO: el objetivo es no tener un único punto de fallo (el coordinador central) .se ordenan todos los eventos en el sistema para saber cual ocurrió primero. Cuando un proceso quiere entrar en una región critica, construye un mensaje los envía a todos los procesos y estos enviaran el reconocimiento diciendo OK. Si el receptor ya esta en la región critica no responde y encola la solicitud .Si dos procesos quieren entrar al mismo tiempo se comparan las marcas de tiempo y la menor gana. Un proceso necesita el permiso de todos para entrar a la región critica. Si cualquier proceso falla no responderá a las solicitudes y se interceptara como negación de acceso .S e incrementa la probabilidad de fallo y el trafico en la red. Se sobrecarga el sistema

porque todos los procesos participan en la toma de decisiones para el ingreso a las regiones críticas.
ALGORITMO DE ANILLO DE FICHAS (token ring): los procesos forman un anillo lógico y cada proceso tiene una posición en un anillo, cada proceso sabe quien sigue .Al iniciar el anillo se le da a un proceso la ficha que circula en todo el anillo .Cuando un proceso obtiene la ficha de su vecino verifica si intenta ingresar a una región critica, en caso positivo entra, trabaja, sale y para la ficha, sino vuelve a pasarla .En un instante solo un proceso esta en una región critica.

ALGORITMO DE ELECCION: son los algoritmos para la elección de un proceso coordinador, iniciador, secuenciador, etc.
El objetivo es garantizar que luego de la elección todos los procesos están de acuerdo con que sea así la identidad del nuevo coordinador.
ALGORITMO DE GRANDULON GARCIA-MOLINA: un proceso inicia una elección cuando el coordinador no responde a las solicitudes .El proceso envía un mensaje "elección" a los demás procesos con un numero mayor .Si nadie responde asume que gana y se convierte en el nuevo coordinador .Si responde un proceso con un numero mayor toma el control y el otro proceso termina. Se envía un mensaje para anunciarlo.
ALGORITMO DE ANILLO: los procesos tienen un orden físico o lógico y cada proceso conoce a su sucesor .Cuando un proceso nota que el coordinador no funciona se construye un mensaje " elección" con su numero de proceso y se lo envía a su sucesor y así sucesivamente y cada proceso añade su numero de proceso .Cuando llega nuevamente el mensaje al proceso que lo inicio, se hace circular un mensaje "coordinador" para informar a los demás procesos quien es coordinador (el que tenga el numero mayor) y los miembros del anillo.

TRANSACCIONES ATOMICAS: las técnicas de sincronización son de bajo nivel. El programador se enfrenta directa con:

- exclusión mutua
- manejo de regiones criticas
- prevención de bloqueos
- recuperación de fallos

en la transacción atómica se precisan técnicas de abstracción de mayor nivel para ocultar aspectos técnicas y permitan a los programadores concertarse en los algoritmos y como los procesos trabajen juntos en paralelo.
La propiedad de la transacción atómica el "TODO O NADA", o se hace todo lo que hay que hacer como una unidad o no se hace nada (Ej. retirar dinero de una cuenta y depositar en otra).Se agrupa las dos operaciones en una transacción, o se hacen las dos operaciones o no se hacen ninguna, y se puede regresar al estado inicial sino concluye la transacción.

EL MODELO DE TRANSACCION: si en el sistema hay varios procesos independientes que pueden fallar aleatoriamente.
ALMACENAMIENTO ESTABLE: se implanta con una pareja d discos comunes, cada una es una copia exacta (espejo) de otra unidad .Primero se actualiza uno y luego otro, si falla el sistema y solo actualizo la unidad 1, luego de la recuperación se puede actualizar la unidad 2 en función de la 1.Es un esquema adecuado para aplicaciones que requieren de un alto grado de tolerancia a fallos (ej transacciones atómicas)

PRIMITIVAS DE TRANSACCION: Son proporcionados por el S.O. o por el sistema de tiempo de ejecución del lenguaje .Existen comandos que pueden formar una transacción, terminarla o eliminarla.
PROPIEDADES DE LAS TRANSACCIONES:

- *SEÑALIZACION*: las transacciones que ocurren al mismo tiempo no interfieran entre si.
- *ATOMICIDAD:* la transacción no ocurre si ocurre totalmente e instantáneamente.
- PERMANENCIA: si se compromete la transacción los cambios son permanentes.
- *TRASACCIONES ANIDADAS*: Cuando las transacciones pueden contener subtransacciones(procesos hijos) que se ejecuten en paralelo entre si en procesadores distintos y originan nuevas transacciones
- .

IMPLANTACION DEL MODELO DE TRANSACCION.
existen varios metodos:
ESPACIO DE TRABAJO PARTICULAR: cuando un proceso inicia una transacción se le otorga un espacio de trabajo particular que contienen todos los archivos a los cuales tienen acceso. Las lecturas- escrituras van a este espacio hasta que la transacción se completa o aborte.
Hay un alto consumo de recursos por las copias de los objetos al espacio de trabajo.
BITACORA DE ESCRITURA ANTICIPADA: o lista de intenciones .Los archivos realmente se modifican pero antes de cambiarlos, se graba un registro con los datos necesarios en la bitácora, si la transacción aborta se utiliza la bitácora para respaldo del estado original y se deshacen los cambios hechos .La bitácora permite ir hacia delante o hacia atrás.
PROTOCOLO DE COMPROMISO DE DOS FASES. Uno de los procesos que interviene funcionan como coordinador, este escribe una entrada en la bitácora para indicar que inicia el protocolo, se envía un mensaje a cada procesos para que estén preparados para el compromiso, los que los reciben el mensaje verifican si están listos y lo informan, cuando el coordinador recibe todas las respuestas sabe si se establece el compromiso o aborta.

CONTROL DE CONCURRENCIA EN EL MODELO DE TRANSACCION: se necesitan algoritmos de control de concurrencia cuando se ejecutan varias transacciones simultáneamente: en distintos procesos, en distintos procesadores.
Los principales algoritmos son.
CERRADURA (LOCKING): cuando un proceso debe leer o escribir en un archivo como parte de una transacción, primero cierra el archivo .La cerradura se hace mediante un único manejador centralizado o manejador local en cada maquina las cerradura de lectura se comparten, las cerraduras d escrituras son exclusivas para una transacción .
CONTROL OPTIMISTA DE LA CONCURRENCIA: es sencilla, se sigue adelante y hace todo lo que tiene que hacer, sin prestar atención que tiene que hacer, sin presta atención a lo que hacen los demás se actúa luego si hay algún problema. Se registran los archivos leídos o grabados y el momento del compromiso se verifican las transacciones para ver si algún archivo se modifico desde el inicio de la transacción si ocurre esto se aborta la transacción, sino se completa.
MARCAS DE TIEMPO: a cada transacción se le asocia a una marca d tiempo al iniciarla, las marcas so únicas, cada archivo tendrá una marca para lectura y otra para escritura, que indican la ultima transacción y cuando un proceso intente acceder a un

archivo lo hará si las marcas de tiempo son menores (mas viejas) que la marca de la transacción actual.

BLOQUEOS EN SISTEMAS DISTRIBUIDOS: son peores que en los sistemas monoprocesadores (mas difícil de evitar, prevenir, detectar y recuperan).toda la información relevante esta dispersa en muchas maquinas .Son criticas en sistemas de base de datos distribuidos .Estrategias usuales para el manejo de bloqueos.
ALGORITMO DE AVESTRUZ. Ignora el problema igual que el uso de un solos procesador.
EVITACION. Evitar los bloqueos mediante la asignación cuidadosa de los recursos, pero en los sistemas distribuidos es difícil implementarlos, se requiere información de antemano (proporción de recursos que necesitara cada proceso)
Las más aplicables son:
DETECCION: se permiten que ocurran los bloqueos, se los detecta e intenta recuperarse de ellos.
PREVENCION: se hace que los bloqueos sean imposibles desde el punto de vista estructural.

DETECCION DISTRIBUIDA DE BLOQUEOS: cuando se detecta un bloqueo en un S.O. convencional se eliminan uno o más procesos. Pero en un sistema basado en transacciones se abortan una o mas transacciones y el sistema restaura el estado que tenia antes de iniciar la transacción, pero también la transacción puede volver a comenzar .Las consecuencias de eliminar un proceso son menos severas si se utilizan transacciones.
DETECCION CENTRALIZADA DE BLOQUEOS. Cada maquina tiene la grafica de recursos de sus propios procesos y recursos .Un coordinador central tiene la grafica de recursos todos el sistema, cuando el coordinador detecta un ciclo elimina un proceso para romper el ciclo.
DETECCION DISTRIBUIDA DE BLOQUEOS: los procesos pueden solicita varios recursos al mimo tiempo, en vez de uno cada vez. Un proceso espera uno o mas recursos simultáneamente y pueden ser recursos locales o remotos .Si un proceso se bloquea por otro proceso se envía un mensaje de exploración al procesos que detiene los recursos necesarios, si el mensaje, recorre todo y regresa del emisor, significa que existe un ciclo y el sistema esta bloqueado .Se lo rompe eliminado al proceso que inicio la exploración, o el que tiene el numero mas alto.
PREVENCION DISTRIBUIDA DE BLOQUEOS: consiste en el diseño cuidadoso del sistema para que los bloqueos sean imposibles estructuralmente.
Hay diversas técnicas:
* permitir a los procesos que solo conserven un recurso a la vez
* Exigirle a los procesos que soliciten todos sus recursos al principio.
* que los procesos liberen sus recursos cuando soliciten uno nuevo.
En sistemas distribuidos con el tiempo global y transacciones atómicas : se asocia a cada transacción una marca d tiempo global al indicar la transacción (no hay parejas de transacciones con igual marca).cuando un proceso se va a bloquear en espera de un recurso que esta usando otro proceso , se verifica cual tiene la marca mayor (o mas joven) y se permite que espere solo si el proceso es espera (mas vieja) tiene menor que el otro .al seguir una cadena de procesos en espera, las marca aparecen de forma creciente y los ciclos son imposibles.

TEMA 10
PROCESOS Y PROCESADORES EN SISTEMAS DISTRIBUIDOS

10.1 INTRODUCCION A LOS HILOS (THREADS): Los hilos son como mini procesos que se ejecutan de forma estrictamente secuencial. Tiene su propio contador de programa y una pila para llevar un registro de su posición. Los hilos comparten la cpu igual que los procesos: (secuencial mente en tiempo compartido) en un multiprocesador se pueden ejecutar realmente en paralelo. Los hilos pueden crear hilos hijos. Mientras un hilo esta bloqueado se puede ejecutar otro hilo del mismo proceso. Los distintos hilos de un proceso pueden compartir un espacio de direcciones, el conjunto de un archivo abierto, cronómetros, señales, etc. los hilos pueden tener distintos estados en ejecución bloqueado, listo, terminado.

10.2 USO DE HILOS: Los hilos permiten la combinación del paralelismo con la ejecución secuencial y el bloqueo de las llamadas al sistema.
Posibles métodos de organización para muchos hilos de ejecución:
Modelo servidor / trabajador: Un hilo, el **servidor**, lee las solicitudes de trabajo en el buzón del sistema. Elige a un **hilo trabajador** inactivo (bloqueado) y envía la solicitud, despertándolo. El h. Trabajador verifica si puede satisfacer la solicitud y lo hace. Los hilos ganan un desempeño considerable pero cada uno de ellos se programa en forma secuencial.
Modelo de equipo: todos los hilos son iguales y c / u obtiene y procesa sus propias solicitudes. No hay servidor. Hay una cola de trabajo que contiene todos los trabajos pendientes.
Modelo de entubamiento: El primer hilo genera datos y los transfiere al siguiente para su procesamiento. Los datos pasan de hilo en hilo y en cada etapa se lleva a cabo cierto procesamiento.
Un programa diseñado adecuadamente y que utilice hilos debe funcionar bien en una única CPU y en un multiprocesador.

10.3 ASPECTOS DEL DISEÑO DE UN PAQUETE DE HILOS: un conjunto de primitivas relacionadas con los hilos disponibles para los usuarios se llama un "PAQUETE DE HILOS".
Se tiene hilos estáticos y dinámicos.
En un **diseño estático:** Se elige el n° de hilos al escribir el programa o durante su compilación. C / u de ellos tiene asociada una pila fija. Se logra simplicidad pero también inflexibilidad.
En un **diseño dinámico:** Se permite la creación y destrucción de hilos durante la ejecución. Los hilos comparten una memoria común y el acceso se controla mediante *regiones criticas.*

10.4 IMPLANTACION DE UN PAQUETE DE HILOS: Un paquete de hilos se puede implantar en el *espacio del usuario o del núcleo.*
Implantación de un paquete de hilos en el espacio del usuario: El nucleo no sabe de su existencia y maneja procesos con un único hilo. El intercambio de hilos es mas rápido que si se usara señalamiento al núcleo, cada proceso puede tener su propio logaritmo de planificación de hilos.
Implantación de un paquete de hilos en el espacio del núcleo: para cada proceso el núcleo tiene una tabla con una entrada para cada hilo que contiene información del hilo o un hilo de otro proceso.
Los paquetes de hilos a nivel de usuario tienen el problema que si un hilo se ejecuta no se puede ejecutar ningún otro de ese proceso.

10.5 HILOS Y RPC: en los sistemas distribuidos es común utilizar hilos y RPC. Con un hilos servidor "S" y un hilo cliente "C", para llamar al servidor, el cliente coloca sus argumentos en la pila compartida, hace un señalamiento al núcleo, el núcleo detecta y deduce que es una llamada local, inicia el hilo cliente, el ejecutar el procedimiento del servidor. La llamada se efectúa porque los argumentos están en su lugar y no es necesario su copiado u ordenamiento.
La RPC local se realiza más rápido de esta forma.

10.6 MODELOS DE SISTEMAS: En un sistema distribuido, con varios procesadores, es importante el diseño de cómo se los va a utilizar. Los procesadores distribuidos se pueden organizar de varias formas:

- Modelo de estación de trabajo
- Modelo de pila de procesadores
- Modelo híbrido

10.7 EL MODELO DE ESTACIÓN DE TRABAJO. El sistema consta de estaciones de trabajo (PC) dispersas conectadas entre si mediante una red de área local (LAN). Pueden contar o no con disco rígido en c / u de ellas. Los usuarios tienen una cantidad fija de poder de cómputo exclusivo y un alto grado de autonomía para asignar recursos de estación de trabajo.

10.8 USO DE ESTACIONES DE TRABAJO INACTIVAS: se ordena remotamente la ejecución de procesos en estaciones de trabajo inactivos. Una estación de trabajo estará inactiva cuando nadie toca el ratón o teclado por varios minutos y no se ejecutan algún proceso iniciado por el usuario.
Existen algoritmos para localizar las estaciones de trabajo inactivas:

- *Algoritmos controlados por el servidor:* Cuando una estación de trabajo esta inactiva se convierte en un servidor potencial. Debe anunciar su disponibilidad en la red.
- *Algoritmos controlados por el cliente:* El cliente transmite una solicitud con sus resultados. Si regresa las respuestas se elige una estación inactiva y se la configura.

Para ejecutar un proceso en una estación remota seleccionada se debe lograr el desplazamiento del código y la configuración del proceso remoto para que vea el mismo ambiente que en el área local, y se ejecuta de la misma forma. Si regresa el poseedor de la máquina se elimina el proceso intruso abruptamente u ordenadamente, o bien se podría emigrar el proceso a otra estación.

10.9 EL MODELO DE LA PILA DE PROCESADORES: Se dispone de un conjunto de cpu que se pueden asignar dinámicamente a los usuarios según la demanda. Los usuarios no disponen de estaciones de trabajo sino de terminales graficas de alto rendimiento.
El argumento para la centralización del poder de cómputo como una pila de procesadores proviene de la teoría de colas. Se reúnen todas las CPU y se forma una sola pila de procesadores, y tenemos un solo sistema de colas en vez de "n" colas ejecutándose en paralelo.
Este modelo de pila es más eficiente que el modelo de búsqueda de estaciones inactivas.
También existe el **modelo híbrido** que consta de estaciones de trabajo y una pila de procesadores.

10.10 ASIGNACION DE PROCESADORES: Son necesarios algoritmos para decidir cual proceso hay que ejecutar y en que maquina.

- Para el modelo de estaciones de trabajo: Decidir cuando ejecutar el proceso de manera local y cuando buscar una estación inactiva. Para el modelo de la pila de procesadores: Decidir donde ejecutar cada nuevo proceso.

10.11 MODELOS DE ASIGNACIÓN. Se piensa que todas las máquinas son idénticas (o compartibles con el código), difieren a lo sumo en la velocidad, y que cada procesador se puede comunicar con los demás.
Estrategias de asignación de procesadores:

- **No migratorias:** si se colocó un proceso en una maquina permanece ahí hasta que termina.
- **Migratorias:** Un proceso se puede trasladar aunque haya iniciado su ejecución. Permiten un mejor balance de la carga pero son más complejas.

Los algoritmos de asignación intentan optimizar algo:

- **Uso de las cpu**
- **Tiempo promedio de respuesta**
- **Tasa de respuesta**

10.12 ASPECTOS DEL DISEÑO DE ALGORITMOS DE ASIGNACION DE PROCESADORES:

- **Algoritmos deterministas vs. Heurísticas.** Los deterministas son adecuados cuando se sabe de antemano todo sobre el comportamiento de los procesos. Los heurísticos cuando la carga es imprescindible.
- **Algoritmos centralizados vs. Distribuidos.** Los centralizados reúnen toda la información en un lugar y toman una mejor decisión, pero la máquina central se puede sobrecargar.
- **Algoritmos óptimos vs. Suboptimos.** Los óptimos consumen más recursos que los subóptimos. En los sistemas reales y buscan solución subóptimos, heurísticos y distribuidos.
- **Algoritmos locales vs. Globales.** Cuando se crea un proceso se divide si se va a ejecutar en la máquina que lo generó o en otra. "POLITICA DE TRANSFERENCIA". Los locales son sencilla pero no óptimos y los globales son mejores pero consumen muchos recursos.
- **Algoritmos iniciados por el emisor vs. Iniciados por el receptor.** Cuando una máquina se deshace de de un proceso debe decidir donde enviarlo: "POLITICA DE LOCALIZACIÓN". Se necesita de la información de la carga en todas las partes, obteniéndola de un emisor sobrecargado que busca una máquina inactiva o de un receptor desocupado que busca trabajo.

10.13 ASPECTOS DE LA IMPLANTACION DE ALGORITMOS DE ASIGNACION DE PROCESADORES: los algoritmos suponen que las maquinas conocen su propia carga y pueden informar su estado.

- **La *medición de la carga* no es tan sencilla**. No es sencilla. Algunos métodos: *contar el numero de procesos o solo los procesos en ejecución o listos, *Ó medir el tiempo que la cpu esta ocupada.
- **El *costo excesivo* en consumo de recursos** para recolectar medidas y desplazar procesos, ya que se debería considerar el tiempo de CPU, ancho de banda y el uso de memoria.
- **La complejidad del software:** desempeño, correctéz y robustez del sistema.
- **Estabilidad del sistema:** las máquinas ejecutan sus algoritmos en forma asincrónica por lo que el sistema nunca se equilibra.

10.14 EJEMPLOS DE ALGORITMOS DE ASIGNACION DE PROCESADORES:

- **Un algoritmo determinista según la teoría de graficas.** Precisa información de antemano como los requerimientos de cpu y de memoria de los procesos; si el núcleo de proceso supera al número de CPU, se asignará varios procesos a la misma CPU, la asignación debe minimizar el tráfico en la red.
- **Un algoritmo centralizado.** No necesita información de antemano, es un algoritmo arriba-abajo porque un coordinador mantiene la tabla de usos. No se minimiza el uso de CPU, se procura otorgar a cada usuario una parte justa del poder de cómputo. Cuando la máquina donde se crea un proceso decide que lo va a ejecutar en otra, pide al coordinador que le asigne un procesador.
- **Un algoritmo jerárquico.** En el centralizado el modo central se convierte en un cuello de botella y en un único punto de fallo. Se organizan los procesadores en jerarquías lógicas independientes de las estructuras. Para cada grupo de máquinas hay una máquina administradora que registra las máquinas ocupadas y las inactivas. En caso de falla de un equipo, lo puede reemplazar un subordinado.
- **Un algoritmo distribuido heuristico (eager).** Al crearse un proceso, la maquina donde se origina envía mensajes de prueba a una maquina para preguntar si su carga es menor; si responde que sí, el proceso se envía a ese lugar, sino se elige otra máquina y la prueba.

Luego de "n" pruebas negativas el algoritmo termina y el proceso se ejecuta en la máquina de origen, por ser la que tiene la menor carga.

- **Un algoritmo de remates.** Utiliza un modelo económico con:
  - Compradores (procesos) y vendedores (procesadores) de servicios.
  - Precios establecidos por la oferta y la demanda.

Los procesos deben comprar tiempo de cpu. Cada procesador anuncia su precios, reúnen ofertas de los procesos, informa la ganadora, lo ejecuta y actualiza los precios.

10.15 PLANIFICACION EN SISTEMAS DISTRIBUIDOS: cada procesador hace su planificación local. La planificación independiente no es eficiente cuando se ejecutan en distintos procesadores un grupo de procesos relacionados entre sí, con gran atracción entre ellos.

Se debe garantizar que los procesos con comunicación frecuente se ejecuten de manera simultánea. Un grupo de procesos relacionados entre sí se iniciaran juntos, en muchos casos.

La comunicación dentro de los grupos debe prevalecer sobre la comunicación entre los grupos.

El algoritmo de "Austerhout" utiliza la planificación. Debe garantizar que todos los miembros del grupo se ejecuten al mismo tiempo. Cada procesador utiliza un algoritmo de planificación ROUND ROBIN. Se mantienen sincronizados los intervalos de tiempo. Todos los miembros del grupo se colocan en el mismo número de espacio de tiempo pero en procesadores distintos.

TEMA 11
SISTEMAS DISTRIBUIDOS DE ARCHIVOS

11.1 INTRODUCCION A LOS SISTEMAS DISTRIBUIDOS DE ARCHIVOS: En un sistema distribuido se distinguen entre los conceptos de *servicio de archivos* y el *servidor de archivos*.

- El **servicio de archivos**: especifica los servicios que el sistema de archivos ofrece a sus clientes sin decir nada de su implantación.
- El **despachador (servidor) de archivos**: Es un proceso que se ejecuta en una maquina y ayuda a implantar el servicio de archivos. Puede haber uno o varios en un sistema.

11.2 DISEÑO DE LOS SISTEMAS DISTRIBUIDOS DE ARCHIVOS: Los componentes de un sistema distribuido de archivos son:

- El verdadero servicio de archivos: realiza operaciones en los archivos como lectura, escritura.
- El servicio de directorios: crea y maneja directorios, añade y elimina archivos de los directorios.

11.3 LA INTERFAZ DEL SERVICIO DE ARCHIVOS: La protección en los sistemas distribuidos usa las mismas técnicas de los sistemas uniprocesador.

Posibilidades: permiso de cada usuario para acceder a un objeto.

Listas para control de acceso: cada archivo tiene una lista de usuarios que pueden acceder al archivo.

Los servicios de archivos se pueden clasificar en:

- **Modelo carga / descarga:** Las principales operaciones son la lectura y escritura en un archivo. Los archivos se pueden almacenar en memoria o en un disco local.
- **Modelo de acceso remoto:** El sistema de archivos se ejecuta con todas las funciones en los servidores y no en los clientes.

11.4 LA INTERFAZ DEL SERVIDOR DE DIRECTORIOS: Proporciona operaciones para crear y eliminar directorios, nombrar y cambiar el nombre de archivos y mover archivos de un directorio a otro. Se utiliza un **sistema jerárquico de archivos**, representado por un árbol de directorios.

En sistemas que utilizan varios servidores de archivos mediante el montaje remoto, los clientes tienen una visión distinta del sistema de archivo, el sistema no se comporta como un único sistema de tiempo compartido.

La **transparencia con respecto a la posición** significa que el nombre de la ruta de acceso no sugiere la posición del archivo.

Resumiendo, los Métodos usuales para nombrar los archivos y directorios en un sistema distribuido son:

- Nombre maquina + ruta de acceso.
- Montaje de sistemas de archivos remotos en la jerarquía local de archivos.
- Un único espacio de nombres que tenga la misma apariencia en todas las maquinas.

11.5 SEMANTICA DE LOS ARCHIVOS COMPARTIDOS: se necesita definir una semántica de lectura-escritura cuando se comparten archivos.

En sistemas monoprocesador que permiten a los procesos compartir archivos (ej.: unix) la semántica es:

- Si una lectura sigue a un escritura, la lectura debe regresar el valor recién escrito.
- Si dos escrituras se realizan en serie y luego se ejecuta una lectura, el valor que se debe regresar es el almacenado en la última escritura.

En un sistema distribuido esta semántica se logra fácil si:

- Solo existe un servidor de archivos.
- Los clientes no ocultan los archivos.

Problemas que se presentan:

- Retrasos en la red (un read llega antes que un write, se obtiene un valor viejo)
- Desempeño pobre de un sistema distribuido, donde todas las solicitudes pasan a un único servidor.

Otra solución para el uso de archivo compartidos en un sistema distribuido es usar las "TRANSACCIONES ATOMICAS", se garantiza que todas las llamadas contenida en la transacción se lleven a cabo en órden y no habrá interferencia de otras transacciones concurrentes.

11. 6 IMPLANTACION DE UN SISTEMA DISTRIBUIDO DE ARCHIVOS :

- **Uso de archivos**
- **La estructura del sistema**

- **El ocultamiento**
- **La duplicación o replica**
- **El control de la concurrencia**

11.7 USO DE ARCHIVOS. Antes de implantar un sistema de archivos se deberá analizar los "*patrones de uso*". Las principales propiedades observadas son: la mayoría de los archivos son pequeños, la lectura es mas común que la escritura, los accesos son secuenciales, los archivos son de corta vida, es poco usual compartir archivos.

11.8 ESTRUCTURA DEL SISTEMA: en algunos sistemas no existe distinción entre el software de un cliente y el de un servidor, todas las maquinas ejecutan lo mismo.
En otros sistemas el servidor de archivos y el de directorios son solo programas del usuario (se ejecuta el software de cliente y servidor en la misma máquina).
También los clientes y servidores pueden ser máquinas distintas de hardware o software.
Formas de estructurar el servicio a directorios:

- Combinar el servicio a directorios y archivos en un único servidor que administre todas las llamadas.
- Separar el servicio a directorios y archivos utilizando un servidor de directorios y un servidor de archivos.

Otro aspecto estructural es si los servidores de archivos, directorios o de otro tipo deben contener la *información de estado de los clientes*.
Si los servidores no contienen los estados, el servidor lleva a cabo la solicitud y no guarda información en relación a los clientes y las solicitudes. La longitud del mensaje es mayor porque la solicitud incluye información.
Si los servidores incluyen información de estado de los clientes entre las solicitudes. Al abrir un archivo el servidor guarda información que relaciones a los clientes con los archivos abiertos. Al abrir un archivo el cliente recibe un descriptor de archivo que existe para identificar el archivo.

11.9 OCULTAMIENTO: En un sistema cliente-servidor, cada uno con su memoria principal y un disco, Existen cuatro lugares donde se pueden almacenar los archivos o partes de ellos:

- El disco del servidor.
- La memoria principal del servidor.
- El disco del cliente (si existe).
- La memoria principal del cliente.

Cuando el cliente desea leer un archivo se transfiere del disco del servidor a la memoria principal del servidor, y luego recién a la memoria principal del cliente a través de la red.
Esto trae problemas de desempeño y se lo mejora "OCULTANDO" (conservando): los archivos de uso más recientes en la memoria principal del servidor. El cliente lee el archivo desde el cache del servidor y elimina la transferencia de archivo.

11.10 REPLICA: en los sistemas distribuidos de archivo se proporciona la réplica de archivos como un servicio:

- Hay varias copias de archivos
- Cada réplica esta en un servidor de archivo independiente
- La réplica se realiza para:
  - Aumentar la confiabilidad al disponer de respaldos independientes de cada archivo
  - Permite el acceso a archivos aun cuando falle un servidor de archivos.
  - Reparten la carga de trabajo entre varios servidores.

Un sistema de transporte de la réplica si la misma se administra sin intersección del usuario.

11.11 CONCLUSIONES IMPORTANTES RESPECTO DE LA IMPLANTACION DE UN SISTEMA DISTRIBUIDO DE ARCHIVOS: Los principios del diseño de un sistema distribuido de archivos son:

- Las estaciones de trabajo tienen ciclos que hay que utilizar: (si hay que elegir para hacer algo entre una estación de trabajo o un servidor). Se elige una estacion de trabajo porque sus ciclos de CPU son menos costosos que los ciclos de un servidor.
- Utilizar el cache el máximo posible porque ahorra tiempo de cómputo y ancho de banda de la red.
- Explotar las propiedades de uso (eficiencia y sencillos)
- Minimizar el conocimiento y modificación a lo largo del sistema.
- Crear lotes de trabajo mientras sea posible para el mejor desempeño.

11.12 TENDENCIA EN LOS SISTEMAS DISTRIBUIDOS DE ARCHIVOS: Los cambios en el hardware tendrán un efecto importante en los sistemas distribuidos de archivo futuro.

También el cambio de expectativa del usuario.

11.13 CONSIDERACIONES RESPECTO DEL HARDWARE: el abaratamiento de la memoria principal permitirá disponer de servidores con memorias cada vez mayores. Se podrá alojar directamente en memoria el sistema de archivo logrando mayor sencillez y mejor desempeño. También se podrá obtener respaldos continuos ante el corte de energía eléctrica.
La disponibilidad de redes de fibra óptica de alta velocidad permitirá una simplificación del software.
La posible construcción de interfaces de red que resuelvan por hardware problema difícil de soportar por software.

11.14 ESCALABILIDAD: la tendencia de los sistemas distribuidos es hace sistemas cada vez más grandes. Algunos sistemas distribuidos que operan bien para cientos de máquinas.
Los algoritmos centralizados no se escalan bien porque el servidor puede convertirse en un cuello de botella.

11.15 REDES EN UN AREA AMPLIA. Los sistemas distribuidos se asocian con redes de área local (LAN), pero cada vez será mayor la necesidad de conectar entre si cubriendo grandes áreas (nacionales, regionales, continentales, etc.).
Los sistemas de archivo deberán soportar esta necesidades teniendo en cuanta la heterogeneidad de los equipos, formatos y códigos. Deberá atenderse a los cambios de tendencia de los requerimientos de las aplicaciones.
El ancho de banda de la red en los sistemas distribuido puede ser insuficiente para el desempeño esperado.

11.16 USUARIOS MOVILES: son usuarios de equipos móviles. Están gran parte del tiempo desconectados del sistema de archivos de su organización.
Lo deseable sería un sistema distribuido totalmente transparente para su uso simultáneo por millones de usuario móviles que frecuentemente se desconectan.

11.17 TOLERANCIA DE FALLOS. La difusión de los sistemas distribuidos incrementa la demanda de sistemas que esencialmente nunca fallen.
Los sistemas tolerantes a fallos requerirán cada vez más redundancia en: Hardware, comunicaciones, software, datos, etc.
La replica de archivos seria un requisito; También que los sistemas funcionen aun con la carencia de parte de los datos.
Los tiempos de fallo aceptables por los usuarios serán cada vez menores.

TEMA 12
RENDIMIENTO

12.1 INTRODUCCION A LA MEDICION, CONTROL Y EVALUACION DEL RENDIMIENTO.

El S.O. es un administrador de recursos, y debe poder determinar la efectividad con que lo administra.
En muchas de las instalaciones de los recursos se realizan poco o ningún control, y cuando se hacen controles específicos se generan datos que a veces no se sabe como interpretar. Las intalaciones arar vez cuentan con personal instruido en técnicas de análisis del rendimiento.
Los primeros años del desarrollo de las computadoras, el rendimiento se controlaba en el hardware porque era mas costoso.
Actualmente, la tendencia es el software por sus presupuestos. Un software diferente y/o mal utilizado puede ser causa de un rendimiento del hardware y software.

12.2 TENDENCIAS IMPORTANTES QUE AFECTAN A LOS ASPECTOS DEL RENDIMIENTO

Los avances tecnológicos del hardware son costos han bajado y seguirán bajando.
Los costos del trabajo del personal van aumentando.
El microprocesador permitió bajar los costos de los ciclos de CPU, y poner la atención en otro foco, como la utilización de disposición de entrada/salida. Otros aspectos también influyen en la evaluación, construcción de redes, procesamientos distribuidos. Las conexiones se hacen con redes y no con computadores específicos.

12.3 NECESIDAD DEL CONTROL Y DE LA EVALUACION DEL RENDIMIENTO. Los objetivos corrientes en la evaluación del rendimiento son:

- *Evaluación de selección:* El evaluador debe decidir si la adquisición de un sistema de computación es apropiada.
- *Proyección del rendimiento:* El evaluador debe estimar el rendimiento de un sistema inexistente (nuevo sistema o nuevo componente de hardware o software).
- *Control del rendimiento:* El evaluador acumula datos del rendimiento de un sistema o componente existente para Asegurar que el sistema cumple con sus metas de rendimiento, para ayudar a estimar los cambios planeados y para proporcionar los datos necesarios para tomar decisiones estratégicas.

Al desarrollar un sistema nuevo se predecían:

- Las aplicaciones que correrán en el sistema.
- Las cargas de trabajo que las aplicaciones deberán manejar.

Durante el desarrollo e implementación de un nuevo sistema se determina:

- La mejor organización del hardware.
- Las estrategias de administración de recursos.
- Si el sistema cumple o no con sus objetivos de rendimiento.

12.4 MEDICIONES DEL RENDIMIENTO: El rendimiento expresa la manera o la eficiencia con que un sistema de computación cumple sus metas.
El rendimiento es una *cantidad relativa* pero suele hablarse de *medidas absolutas de rendimiento:* Ej.: n° de trabajos atendidos.

Las mediciones de rendimiento pueden estar:

- *Orientadas hacia el usuario*: Ej.: tiempos de respuesta.
- *Orientadas hacia el sistema:* Ej.: utilización de la cpu.

Algunas mediciones comunes son:

- *Tiempo de regreso:* desde la entrega del trabajo hasta su regreso
- *Tiempo de respuesta:* Tiempo de regreso de un sistema interactivo.
- *Tiempo de reacción del sistema:* Tiempo desde que el usuario presiona “enter” hasta que se da la primera sección de tiempo de servicio.

Otras medidas del rendimiento utilizadas son:

- *Varianza de los tiempos de respuesta:* Es una medida de dispersión y de la predecibilidad.
- *Capacidad de ejecución:* medida de la ejecución de trabajo por unidad de tiempo.
- *Carga de trabajo:* medida de la cantidad de trabajo.

- *Capacidad:* de rendimiento máxima que un sistema puede tener

- *Utilización:* fracción de tiempo que un recurso esta en uso. En cuanto a la CPU se debe distinguir entre uso de trabajo productivo y uso en sobrecarga.

12.5 TECNICAS DE EVALUACION DEL RENDIMIENTO

Tiempos. Proporcionan los medios para realizar comparaciones rápidas del hardware. Una posible unidad de medida es el "mips" (millón de instrucciones por segundo).
Mezclas de instrucciones. Son útiles para comparaciones rápidas del hardware. Se usa una promedio de varios tiempos de las instrucciones mas apropiadas para una aplicación.
Programas del núcleo. es un programa típico que puede ejecutarse en una instalación. Se utilizan los tiempos que suministran la fabricación para cada máquina para calcular su tiempo de ejecución. Se corre el programa típico a cada maquina y se evalúan su tiempo de ejecución.
Modelos analíticos. Son representaciones matemáticas de sistemas de computación o de componentes de sistemas de computación. Generalmente se utilizan los modelos de: *Teoría de colas. O Procesos de markov.*
Puntos de referencia Son programas reales que el evaluador ejecuta en la maquina que se esta evaluando. Es un *programa de producción* (que se ejecuta con regularidad)
Programas sintéticos. Combinan las técnicas de los núcleos y los puntos de referencia. Son programas diseñados para ejercitar características especificas de una maquina.
Simulación. Técnica con la cual el evaluador desarrolla un modelo computarizado del sistema que se esta evaluando. Se preparar un programa inexistente y se ejecuta para ver como se comportaría en ciertas circunstancias. Se evita la construcción de sistemas mal diseñados.
Control del rendimiento. Es la recolección y análisis de información relativa al rendimiento del sistema existente. Permite localizar embotellamientos y decidir la mejor forma para el rendimiento. Útil para distribuir los trabajos de varios tipos.
El control del rendimiento puede hacerse por medio de técnicas de:
Software: con monitores económicos pero consumen recursos en el sistema.
Hardware: con monitores costosos pero no influyen en el sistema.
Monitores: Producen grandes cantidades de datos que deben ser analizados manualmente o por sistema.
Indican con precisión como esta funcionando un sistema.

12.6 EMBOTELLAMIENTOS Y SATURACION:

Los recursos administrados por los s. O. Se acoplan interactúan de maneras complejas la cual puede producir embotellamiento que limitan el rendimiento del sistema.
Un embotellamiento tiende a producirse en un recurso cuando el tráfico de trabajos o procesos de ese recurso *alcanza su capacidad límite*. El recurso se encuentra *saturado* y los procesos que compiten por el recurso comienzan a interferirse unos a otros.
Un ejemplo es el problema de la hiper paginación que ocurre cuando el almacenamiento primario está lleno y los conjuntos de trabajo de los procesos activos no se pueden mantener simultáneamente en el almacenamiento primario.
El embotellamiento se detecta controlando cada cola de peticiones de los recursos. La cola no debe crecer porque habrá mas peticiones esperando que peticiones servidas.

12.7 CICLOS DE RETROALIMENTACION:

es una situación en la cual la información del estado actual del sistema se pone a disposición de las peticiones entrantes.
La ruta de las peticiones puede modificarse, si la retroalimentación indica que puede haber dificultad de darle servicio.
Retroalimentación negativa. Las nuevas peticiones puede decrecer como resultado de la información que se esta retroalimentando. Implica que las colas crezcan indefinidamente.
Retroalimentación positiva. La información retroalimentada provoca un incremento pero puede deteriorar el rendimiento.

TEMA 13
MODELADO ANALITICO EN RELACION AL RENDIMIENTO

13.1 INTRODUCCION AL MODELADO ANALITICO Y TEORIA DE COLAS: Los modelos analíticos son las representaciones matemáticas de los sistemas y permiten al avaluador del rendimiento sacar conclusiones acerca del comportamiento del sistema.
Si no hubiera líneas de espera se recibirá servicio de inmediato (que es lo deseable) pero el costo de tener suficiente capacidad de servicio para que no haya espera es muy alto.
Si hay clientes que demandan servicios prestada por servidores, algunos ingresan a la red de colas u esperan a que un servicio quede disponible.
Algunas colas son:

- *Limitadas*: contienen un número fijo de clientes.
- *Ilimitadas*: crecen lo suficiente para mantener a todos los clientes.

Las técnicas mas conocidas de modelado analítico son TEORIA DE COLAS y PROCESOS DE MARKOW
Se deben tener en cuenta *variables aleatorias* que son descritas:

- La variable aleatoria "q" representa el tiempo que emplea un cliente esperando en la cola a ser servido.
- La v. a. "s" representa la cantidad de tiempo que emplea un cliente en ser servido.
- La v. a. "w" representa el tiempo total que emplea un cliente en el sistema de colas: "w = q + s". (total de espera + servicio).

13.2FUENTE, LLEGADAS Y LLEGADAS DE POISSON:
Fuente: Los clientes son proporcionados a un sistema de colas desde una *fuente* que puede ser infinita o finita.

- *fuente infinita* la cola de servicio puede ser muy grande.
- *fuente finita* la cola de servicio es limitada.

Llegadas: son los momentos en que los clientes llegan a un sistema de colas. Los clientes llegan de uno en uno y no hay colisión.
Las llegadas según Poisson se trabaja con una tasa promedio de llegada expresada en clientes por unidad de tiempo" y tienen una distribución de Poisson.

13.3TIEMPOS DE SERVICIO, CAPACIDAD DE LA COLA Y NUMERO DE SERVIDORES EN EL SISTEMA: Se supone que los tiempos de servicio son aleatorios, que el cliente que requiera el sistema.
La capacidad de las colas puede ser:

- *Capacidad infinita:* Cada cliente que llegue puede entrar en el sistema de colas y esperar, sin importar cuantos clientes hay en espera.
- *Capacidad cero (o sistemas de pérdidas):* Cada cliente que llegan cuando la instalación de servicios está ocupada no podrán ser admitidas.
- *Capacidad positiva:* Los clientes que llegan solo esperan si hay lugar en la cola. Los sistemas de colas se pueden categorizar según el número de servidores.

NUMERO DE SERVIDORES EN EL SISTEMA.

- *Sistemas de un solo servidor:* Tienen un solo servidor y solo da servicio a un solo cliente a la vez.
- *Sistemas de servidores múltiples:* Tienen varios servidores con igual capacidad y dan servicio a varios clientes a la vez.

13.4 DISCIPLINAS DE COLAS: Son las reglas usadas para elegir al siguiente cliente de cola que va a ser servido. La mas conocida es la "FCFS" o primero en llegar, primero en ser servido.

13.5 INTENSIDAD DE TRAFICO Y UTILIZACION DEL SERVIDOR
INTENSIDAD DE TRÁFICO. Es una medida de la capacidad del sistema para dar servicio efectivo a sus clientes.
Se define como la razón de la media del tiempo de servicio y la media del tiempo entre llegadas.
Se utiliza para determinar el número de servidores iguales y la medida que necesitará un sistema para dar servicios a sus clientes en que las colas sean largas o se rechazan clientes.
UTILIZACION DEL SERVIDOR. se define como la intensidad del tráfico por servidor. es la probabilidad de que un servidor determinado se encuentre ocupado. En sistemas de un solo servidor es igual a la intensidad de tráfico.

13.6 ESTADO ESTABLE EN FUNCION DE SOLUCIONES TRANSITORIAS: los sistemas de colas se suponen que están operando en estado estable.
La posición inicial de un sistema no indica su comportamiento periódico. Los sistemas de colas pasan por un periodo inicial de operación antes de tener un funcionamiento UNIFORME, PREDECIBLE.
La solución y estudio de un sistema de colas es mas fácil si esta en estado estable. Las situaciones transitorias o dependientes del tiempo, son más complejas.

13.7 RESULTADO DE LITTLE: es una de las medidas más sencilla y útiles del rendimiento de un sistema de colas relaciona los clientes en la cola y en el sistema, da el promedio de operadores en espera por minuto.

13.8 RESUMEN DEL PROCESO DE POISSON: se define como la probabilidad de exactamente "M" llegadas en un intervalo de tiempo de longitud "L".

13.9 ANALISIS DE UN SISTEMA DE COLAS M/M/1: utiliza la intensidad de trafico y la Utilización del servidor. es la Probabilidad de que todos los servidores estén en uso, por lo que un cliente que llega debe esperar. También usa el Tiempo promedio en la cola y en el sistema mediante formulas. *Formulas de estado para el sistema de colas m/m/1* para modelar el uso de un equipo especial.
En una situación dada, se puede determinar que un solo equipo no es suficiente para hacer frente a las necesidades que se plantaban sin provocar esperas excesivas.

13.10 ANALISIS DE UN SISTEMA DE COLAS M/M/c: para analizar si las decisión es adquirir más equipos en un sistema para ser efectivo y no provocar mucha espera. Se mantendrán todos los equipos en un lugar central o se distribuirán. Si se colocan los equipos en diferentes lugares, cada equipo es analizar como un sistema de colas M/M/1. Si se desea centralizar, se considera un sistema de colas M/M/c sencillo.

13.11 PROCESOS DE MARKOV: son modelos adecuados para describir el comportamiento de sistemas: es sistema esta situado en uno de un conjunto de *estados discretos mutuamente excluyentes y colectivamente exhaustivos.*
El *estado presente* del sistema y las *probabilidades de transición* entre varios estados del sistema hacen el comportamiento futuro del sistema.
Dado que un proceso de markov se encuentra en un estado determinado, su comportamiento futuro no depende de su historia anterior a su entrada a ese estado.
Un cambio de estado en un *proceso de markov de transición continua* puede producir cambios de estado en cualquier instante de una escala de tiempo continua.

13.12 PROCESOS DE NACIMIENTO Y MUERTE: Son un caso importante de los procesos de markov. Particularmente aplicables al modelado de sistemas de computación.
Un *proceso de markov de nacimiento y muerte continuo* utiliza la tasa a la cual ocurren las transacciones de un estado a otro, con la tasa promedio de nacimiento desde el estado y la de muerte.

TEMA 14
SEGURIDAD DE LOS SISTEMAS OPERATIVOS

14.1 INTRODUCCION A LA SEGURIDAD DE LOS SISTEMAS OPERATIVOS: La evolución de la computación y de las comunicaciones hasta hoy, ha hecho más accesibles a los sistemas informáticos, y se incrementado los riesgos con respecto a la seguridad.
la importancia de la seguridad de los sistemas informáticos es cada vez mayor por la expansión de las redes de computación y la comunicación de datos.
Los datos administrados por el sistema informático cada vez son mas confidenciados y críticos (Ej. trasferencia de fondos). Los sistemas deben funcionar sin interrupciones y sus problemas.
El S.O. como administrador de los recursos del sistema es importante en el aspecto de la seguridad pero se debe complementar con métodos externos al S.O.
El nivel de seguridad a proporcionar depende del valor de los recursos que hay que asegurar.

14.2 REQUISITOS DE SEGURIDAD: los requisitos de la seguridad de un sistema define lo que significa la seguridad para el sistema. los requisitos son la base para determinar si el sistema implementado es seguro.

14.3 UN TRATAMIENTO TOTAL DE LA SEGURIDAD: incluye aspectos de la seguridad del computador distinto a los de la seguridad de los S.O.
La *seguridad externa* debe asegurar la instalación computacional contra intrusos y desastres como incendios e inundaciones. Luego del acceso físico al S.O. se debe identificar al usuario antes de que acceda a los recursos "*Seguridad de la interfaz del usuario*".
La *seguridad interna* trata de los controles incorporados al hardware y al s. O. Para asegurar que son confiables, operable y la integridad de los programas y datos.

14.4 SEGURIDAD EXTERNA Y SEGURIDAD OPERACIONAL:
Seguridad externa. Consiste en:
- *Seguridad física. Proteccion contra desastres e intrusos.*
- *Seguridad operacional.*

La seguridad física incluye: la protección contra desastres (costoso y no se analiza en detalle, depende de las consecuencia de la pérdida) y la protección contra intrusos (usando sistemas de identificación física, tarjeta, voz…)
En la seguridad física importan los mecanismos de detección de humo, Sensores de calor, detección de movimiento.
Seguridad operacional. Consiste en las diferentes políticas y procedimientos implementados por la administración de la instalación computacional.
La *autorización* determina que acceso se permite y a quien.
La *clasificación* divide el problema en sub problemas: Los datos del sistema y los usuarios se dividen en clases y a cada clase se conceden diferentes derechos de acceso.
Un aspecto crítico es la selección y asignación de personal: que sea confiable, pero lo principal es la división de responsabilidades.
Para diseñar medidas efectivas de seguridad se debe:
- Enumerar y comprender las amenazas potenciales.
- Definir que grado de seguridad se desea (y cuanto se esta dispuesto a gastar en seguridad).
- Analizar las contramedidas disponibles.

14.5 VIGILANCIA, VERIFICACION DE AMENAZAS Y AMPLIFICACION
Vigilancia. Tiene que ver con: La verificación y la auditoria del sistema; La autentificación de los usuarios.
Los sistemas sofisticados de autentificación de usuarios es difíciles de evitar por los intrusos. El problema es que se puede rechazar a usuarios legítimos (ej. en un sistema de reconocimiento de voz que se puede llegar a rechazar a un usuario resfriado).
Verificación de amenazas. Es una técnica por lo que los usuarios no pueden tener acceso directo a un recurso. Solo lo tiene rutinas del S.O. llamados "PROGRAMAS DE VIGILANCIA". El usuario solicita el proceso al S.O., el S.O. niega o permite el acceso. El acceso lo hace un programa de vigilancia que pasa los resultados al programa del usuario, y permite detectar los intentos de penetración, el momento que se producen y advertir.
Amplificación. Se producen cuando un programa de vigilancia necesita para cumplir su meta mayores derechos de acceso de los que disponen los usuarios.

14.6 PROTECCION POR CONTRASEÑA: Las clases de elementos para establecer la identidad de una persona son:
Algo sobre la persona: Ej.: huellas digitales, registro de la voz, fotografía, firma, etc.
Algo poseído por la persona: Ej.: insignias especiales, tarjetas de identificación, llaves, etc.
Algo conocido por la persona: Ej.: contraseñas, combinaciones de cerraduras, etc.
El esquema más común de autentificación es la *protección por contraseña:* donde el usuario elige una palabra clave, la amenaza, la teclea para que la admita el sistema (la clave no se ve en pantalla).
Desventajas: los usuarios tienden a elegir contraseñas fáciles de recordar (DNI, nombres) y esto puede ser conocida por quien intente violar la seguridad con actos repetidos debiendo limitar la cantidad de fallos de aciertos para el ingreso de la contraseña. La contraseña no debe ser muy corta para no facilitar la probabilidad de aciertos; pero tampoco muy larga para que se pueda recordar.

14.7 AUDITORIAS Y CONTROL DE ACCESO:
Auditoria. Suele realizarse e posteriori en sistemas manuales, se examinan las recientes transacciones de una organización para determinar si hubo hechos ilícitos.
La auditoria de un sistema informático implica un procesamiento inmediato: se verifican las transacciones que se acaban de hacer.
Un registro de auditoria es un registro permanente de acontecimientos importante que tiene un sistema. Se realiza automáticamente cuando ocurre un evento; se almacena en un área protegida del sistema. en un mecanismo importante de detección, debe ser revisado cuidadosamente y con frecuencia.
Controles de acceso. El control de accso a los datos almacenados es fundamental para la seguridad interna.
Los derechos de acceso son los tipos de acceso que tiene varios sujetos u objetos. Los sujetos acceden a los objetos. Los objetos son entidades que contienen información (discos, cintas, procesadores, almacenamiento, etc).
El objeto esta protegido contra los sujetos, las autorizaciones a un sistema se conceden a los sujetos. Los sujetos pueden ser varios tipos de entidades como usuarios, procesos, etc.
Los derechos de acceso mas comunes son acceso a lecturas, acceso a escrituras o acceso a ejecuciones. Una forma de implementar es mediante una MATRIZ de control de acceso que es protegida por el S.O. con filas para los sujetos, columnas para el objeto y celdas de las matriz para los derechos de acceso que un usuario tiene a un objeto.

14.8 NUCLEOS DE SEGURIDAD Y SEGURIDAD POR HARDWARE:
Núcleos de seguridad. es mas fácil hacer un sistema seguro si la seguridad se incorpora desde el principio al diseño del sistema. Las medidas de seguridad deben ser implementadas en todo el sistema. Un sistema de alta seguridad requiere que el núcleo del S.O. sea seguro. Las medidas de seguridad, mas decisivas están en el núcleo y se lo mantiene lo mas pequeña posible. La seguridad de un sistema depende de asegurar las funciones que realiza:

- El control de acceso.
- La entrada al sistema.
- La verificación.
- La administración del almacenamiento real, del almacenamiento virtual y del sistema de archivos.

Seguridad por hardware. Existe una tendencia a incorporar al hardware funciones del s. O. porque resultan mas seguras que cuando se acceden como instrucción de software que pueden ser modificadas y también porque pueden operar mas rápido que en el software. Mejora la performance, permite controlarlos con más frecuencia.

14.9 SISTEMAS SUPERVIVIENTES: El diseño de sistemas de alta seguridad debe asegurar: Su operación de manera continua y confiable y también su disponibilidad.
Un *sistema de computación superviviente* es aquel que continúa operando aun después de que uno o más de sus componentes falle. Generalmente continúan operando con una degradación suave en los niveles de prestaciones. Los componentes fallidos se deben poder reemplazar sin interrumpir el funcionamiento del sistema.
Una clave para la capacidad de supervivencia es la REBUNDANCIA, si un componente falla otro equivalente toma su puesto. Esto se puede implementar como un conjunto de recursos idénticos que funcionen en paralelo o un conjunto de accesos separados redundantes que se activan cuando se produce un fallo.
Características de supervivencia:

- La incorporación de mecanismos contra fallos en el hardware en vez de en el software.
- El uso de *multiprocesamiento transparente* para permitir mejorar el rendimiento sin modificar el software.
- El uso de subsistemas múltiples de e/s.
- La incorporación de mecanismos de detección de fallos en el hardware y en el software.

14.10 CAPACIDADES Y SISTEMAS ORIENTADOS HACIA EL OBJETO: Un *derecho de acceso* permite a un *sujeto* acceder a un *objeto* de una manera preestablecida.
Los sujetos son los usuarios de los sistemas de computación o entidades que actúan por los usuarios o por el sistema como trabajo, procesos y procedimientos, etc. Los objetos son los recursos del sistema: Ej.: archivos, programas, directorios, terminales, pistas de discos, bloques de almacenamiento primario, etc. Pero los sujetos son objetos del sistema y acceden a otros sujetos. Los sujetos son entidades activas y los objetos son pasivos.
Una *capacidad* es una señal. La posesión de una capacidad por un sujeto le confiere derechos de acceso a un objeto. Las capacidades no suelen ser modificadas pero suelen ser reproducidas.
Un *dominio de protección* define los derechos de acceso que un sujeto tiene a los distintos objetos del sistema: Es el conjunto de capacidades que pertenecen al sujeto.
La creación de capacidades es una función de rutinas de los S.O. cuidadosamente guardadas. Se crea el objeto y luego una capacidad para ese objeto.

14.11 CRIPTOGRAFIA: el uso creciente de redes de computadoras y la importancia del tráfico cursado hace necesario proteger los datos. Se cifran los datos para la transmisión de información detectadas.
La criptografía es el uso de las transformaciones de datos para que sean incomprensibles a todos (excepto el usuario destinado).
Problemas:
- De Intimidad: como evitar la obtención no autorizada de información de un canal de comunicaciones.
- De la autentificación: evitar que modifiquen una transmisión.
- De la disputas: como proporcionar al receptor, de un mensaje, pruebas legales de la identidad del emisor.

Un sistema de intimidad criptográfica. El *remitente* transmite un mensaje que pasa por una unidad de codificación que lo transforma de texto simple o cifrado, o incomprensible para el espía y se transmite por un canal inseguro pero de forma segura.
El receptor legítimo pasa el texto cifrado por una unidad de descifrado para regenerar el texto simple.
Criptoanálisis. Es el proceso de intentar regenerar el texto simple a partir del texto cifrado, pero sin conocer la clave de ciframiento. Esta tarea lo hace el espía o CRIPTOANALISTA, si no lo logra el sistema es seguro.
Sistemas de clave publica. Las distribuciones de claves de un sistema criptográfico se debe hacer por canales muy seguros:
Los sistemas de clave pública ayudan ala distribución de claves porque se separan las funciones de cifrados y descifrados utilizando distintas claves. Una parte se hace pública (ciframiento) y la otra privada (desciframiento).
Firmas digitales. Para que una firma digital sea aceptado como sustituos de una forma escrita debe ser fácil de autentifica (reconocer) por cualquiera, pero producible solo por el autor.
En los criptosistemas de clave pública el procedimiento es:
  - El remitente usa la clave privada para crear un mensaje firmado.
  - El receptor: Usa la clave publica del remitente para descifrar el mensaje y queda el mensaje firmado para usarlo en caso de disputas.

Aplicaciones. La criptografía es útil en los sistemas multiusuario y en las redes de computadoras. Se la utiliza para proteger contraseñas, almacenándolas cifradas, o para proteger datos almacenados en sistemas de computación.
En el *cifrado de enlace* la red cifra y descifra en cada nodo.
En el *cifrado punto a punto* un mensaje se cifra en sus fuentes y se descifra solo una vez, en su destino. La dirección de destino no puede cifrarse.

14.12 PENETRACION AL SISTEMA OPERATIVO: La penetración cambia el estado de las máquinas, del estado problema al sistema supervisor.
Los estudios de penetración están diseñados para determinar si las defensas de un sistema contra ataques de usuarios no privilegiados son adecuadas.
El control de E/s es un área favorita para intentar penetrar a un sistema. los canales de E/S tienen acceso al almacenamiento primario y pueden modificar información importante.

Principales fallos genéricos funcionales de los sistemas.
- *Autentificación:* Los usuarios no pueden determinar si el hardware y el software con que funcionan son los que deben ser.
- *Cifrado:* No se almacena cifrada la lista maestra de contraseñas.
- *Implementación:* que no proviene de un buen diseño de seguridad
- *Confianza implícita:*

    - Una rutina supone que otra esta funcionando correctamente cuando, de hecho, deberia examinar los parametros suministrados por la otra rutina.
- *Compartimiento implicito:*
    - El s. O. Deposita inadvertidamente informacion importante del sistema en un espacio de direcciones del usuario.
- *Comunicacion entre procesos:*
    - Usos inadecuados de los mecanismos de send / receive que pueden ser aprovechados por los intrusos.
- *Verificacion de la legalidad:*
    - Validacion insuficiente de los parametros del usuario.
- *Desconexion de linea:*
    - Ante una desconexion de linea el s. O. Deberia:
        - Dar de baja al usuario (o los usuarios) de la linea.
        - Colocarlos en un estado tal que requiera la re - autorizacion para obtener nuevamente el control.
- *Descuido del operador:*
    - Un intruso podria engañar a un operador y hacer que le habilite determinados recursos.
- *Paso de parametros por referencia en funcion de su valor:*
    - Es mas seguro pasar los parametros directamente en registros que tener los registros apuntando a las areas que contienen los parametros.
    - El paso por referencia puede permitir que los parametros, estando aun en el area del usuario, puedan ser modificados antes de ser usados por el sistema.
- *Contraseñas:*
    - No deben ser facilmente deducibles u obtenibles mediante ensayos repetidos.
- *Entrampamiento al intruso:*
    - Los s. O. Deben tener mecanismos de entrampamiento para atraer al intruso inexperto.
- *Privilegio:*
    - Cuando hay demasiados programas con demasiados privilegios se viola el *principio del menor privilegio*.
- *Confinamiento del programa:*
    - Un programa “prestado” de otro usuario puede actuar como un “caballo de troya”:
- *Prohibiciones:*
    - Se advierte a los usuarios que no utilicen ciertas opciones porque los resultados podrian ser “indeterminados”, pero no se bloquea su uso.
        - Puede robar o alterar datos.
- *Residuos:*
    - Un intruso podria encontrar una lista de contraseñas con solo buscar en lugares tales como una “papelera”:
        - Del sistema o fisica.
        - La informacion delicada debe ser sobrescrita o destruida antes de liberar o descartar el medio que ocupa.
- *Blindaje:*
    - Los intrusos pueden conectarse a una linea de transmision sin hacer contacto fisico:
        - Utilizan el campo inducido por la circulacion de corriente en un cable.
        - Se previene con un adecuado blindaje electrico.
- *Valores de umbral:*
    - Si no se dispone de valores umbral, no habra:
        - Limites al n° de intentos fallidos de ingreso.
        - Bloqueos a nuevos intentos.
        - Comunicaciones al supervisor o administrador del sistema.

Ataques genericos a sistemas operativos.

- *Asincronismo:*
    - Se tienen procesos multiples que progresan asincronicamente.
    - Un proceso podria modificar los parametros ya validados por otro proceso pero aun no utilizados.

    - Un proceso podria pasar valores malos a otro aun cuando el segundo realice una verificacion extensa.
- *Rastreo:*
    - Un usuario revisa el sistema intentando localizar informacion privilegiada.
- *Entre lineas:*
    - Se utiliza una linea de comunicaciones mantenida por un usuario habilitado que esta inactivo.
- *Codigo clandestino:*
    - Se modifica el s. O. Bajo una presunta depuracion pero se incorpora codigo que permite ingresos no autorizados.
- *Prohibicion de acceso:*
    - Un usuario escribe un programa que bloquea el acceso o servicio a los usuarios legitimos mediante:
        - Caidas del sistema, ciclos infinitos, monopolio de recursos, etc.
- *Procesos sincronizados interactivos:*
    - Se utilizan las primitivas de sincronizacion del sistema para compartir y pasarse informacion entre si.
- *Desconexion de linea:*
    - El intruso intenta acceder al trabajo de un usuario desconectado:
        - Luego de una desconexion de linea.
        - Antes de que el sistema reconozca la desconexion.
- *Disfraz:*
    - El intruso asume la identidad de un usuario legitimo luego de haber obtenido la identificacion apropiada por medios clandestinos.
- *Ataque "nak":*
    - Si el s. O. Permite a un usuario:
        - Interrumpir un proceso en ejecucion mediante una "tecla" de "reconocimiento negativo".
        - Realizar otra operacion.
        - Reanudar el proceso interrumpido.
    - Un intruso podria "encontrar" al sistema en un estado no protegido y hacerse con el control.
- *Engaño al operador:*
    - Con un engaño se hace realizar al operador una accion que comprometa la seguridad del sistema.
- *Parasito:*
    - Mediante equipamiento especial el intruso:
        - Intercepta los mensajes entre un usuario habilitado y el procesador.
        - Los modifica o reemplaza totalmente.
- *Caballo de troya:*
    - El intruso coloca un codigo dentro del sistema que luego le permita accesos no autorizados.
    - Puede permanecer en el sistema.
    - Puede borrar todo rastro de si mismo luego de la penetracion.
- *Parametros inesperados:*
    - El intruso suministra valores inesperados a una llamada al nucleo.
    - Intenta aprovechar una debilidad de los mecanismos de verificacion de la legalidad del s. O.