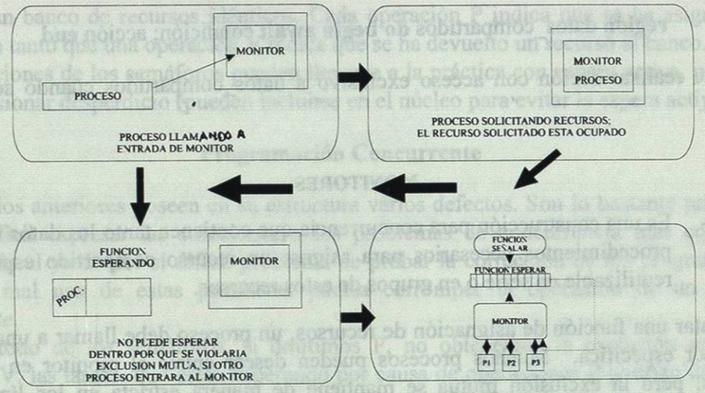
recurso, así que la entrada del monitor invoca a la función *señalar* para permitir que uno de los procesos en espera adquiera el recurso y salga del monitor.
Si un proceso señala la liberación del recurso y ningún proceso esta esperando, la señal no tendrá efecto, pero el monitor habrá recuperado un recurso. Es claro que un proceso que esta en espera de un recurso deberá hacerlo afuera para permitir que otro proceso entre al monitor a devolver el recurso.
Se da mayor prioridad a los procesos en espera para asegurar que un proceso que esta en espera de un recurso termine por obtenerlo, de otra forma un proceso nuevo podría obtener ese recurso y ocasionar un problema de aplazamiento indefinido.



Variables Condicionales.- Las variables condicionales son diferentes de las variables convencionales con las que estamos familiarizadas. Se asocia una variable condicional diferente a cada una de las razones por las que un proceso puede verse obligado a esperar. Las operaciones de *esperar* y *señalar* se modifican para incluir el nombre de la variable condicional que esta esperando o señalando.

ESPERAR (nombre de variable condicional)
SEÑALAR (nombre de variable condicional)

Cuando se define una variable condicional se crea una cola. Un proceso que llama a la función *esperar* se agrega a la cola; un proceso que llama a la función *señalar* hace que un proceso en espera sea sacado de la cola y entre en el monitor.

A continuación se muestra un monitor sencillo para el manejo de la asignación de recursos. El atractivo de la función asignador_de_recursos redica en que funciona exactamente como un semáforo binario; la función obtener_recurso trabaja como la operación P y la función devolver_recurso trabaja como la operación V.


```
monitor asignador_de_recursos;
var recurso_en_uso:boolean;
recurso_libre:condition;

procedure obtener_recurso;
begin
if recurso_en_uso then                    <----------->  Función P
esperar(recurso_libre);
recurso_en_uso := true
end;
procedure devolver_recurso;
begin
recurso_en_uso := false;                  <----------->  Función V
señalar (recurso_libre);
end;
begin
recurso_en_uso := false
end;
```

Buffer circular .-
(Ring Buffer, Bounded Buffer o Buffer Acotado)

Un ejemplo de Monitor. Se utiliza en situaciones en las que un proceso productor pasa datos a un proceso consumidor. Por lo que debemos considerar de vital importancia la sincronización entre los procesos productores y consumidores.
Los sistemas operativos asignan a menudo una cantidad fija razonable de memoria para las comunicaciones mediante buffer entre los procesos productores y consumidores. Esto se puede simular mediante una tabla de tamaño indicado. El productor deposita los datos en elementos sucesivos de la tabla y el consumidor los saca en el orden en que se depositaron. El productor puede haber producido varios elementos que aun no ha utilizado el consumidor. En algún momento el productor llenara el ultimo elemento de la tabla, cuando produzca mas datos deberá dar la vuelta y comenzar a depositar los datos en el primer elemento de la tabla dando por asentado que el consumidor halla sacado los datos previamente colocados ahí por el productor, la tabla se cierra efectivamente en circulo, de ahí el termino de buffer circular.

Debido al tamaño fijo del buffer circular el productor puede encontrar a veces ocupados todos los elementos de la tabla, en este caso, el productor debe esperar hasta que el consumidor vacíe algunos elementos de la tabla, del mismo modo en que habrá ocasiones en que el consumidor deberá esperar hasta que el productor deposite algunos datos en algunos elementos de la tabla.

Un mecanismo de buffer circular es adecuado para implantar un mecanismo de *spool* en los sistemas operativos.
Cuando un proceso genera líneas que se deben imprimir en un dispositivo de salida relativamente lento, como una impresora. Como el proceso puede producir las líneas mucho

más rápidamente de lo que la impresora puede imprimirlas y como también se desea que el proceso se ejecute lo mas rápidamente posible, las líneas de salida del proceso se envían a un mecanismo de buffer circular. El buffer circular puede estar en el almacenamiento primario, pero es más probable que este en disco. A este proceso se le llama *spooler*.
Otro proceso lee las líneas del buffer circular y las escribe en la impresora trabajando a la velocidad de la impresora a este proceso a menudo se le llama *despooler*.
El buffer circular tiene suficiente espacio para absorber el desajuste resultante de la diferencia de velocidad entre *spooler y despooler*. Desde luego suponemos que el sistema no genera indefinidamente, líneas más rápido de lo que la impresora pueda imprimirlas; si así fuera el buffer siempre estaría lleno y sería de poco valor mantener la continuidad de la operación de impresión.

Llamadas anidadas a monitores.- Es posible que un sistema que utilice monitores permita a los procedimientos de un monitor llamar a procedimientos que estén en otro, con lo que se originan las llamadas anidadas a monitores.

Expresiones de ruta.- La noción de expresiones de ruta se introdujo para poder especificar el orden en que se ejecutarán los procedimientos al tiempo que se mantienen separados de esta especificación los procedimientos.
Ejemplos:
path leer **end**
path comenzar_lectura, leer, lectura_terminada **end**
path C:\DOS\WINDOW\F_PROT96

Cuando una expresión de ruta especifica una secuencia de llamadas, no implica que el mismo invocador tenga que hacer cada una de las llamadas.

Paso de mensajes.- Las dos formas existentes de comunicación entre procesos son:

- La memoria compartida
- El paso de mensajes.

Los procesos envían y reciben mensajes mediante llamadas como las siguientes:

```
enviar (proceso_receptor, mensaje);
recibir(proceso_transmisor, mensaje);
```

Las llamadas enviar y recibir se realizan normalmente como llamadas al sistema operativo accesibles desde muchos ambientes de lenguajes de programación.

La llamada a enviar puede señalar explícitamente el nombre del proceso receptor, o puede omitir el nombre, indicando así que el mensaje será difundido a todos los procesos.

Envío con bloqueo.- Debe esperar a que el receptor reciba el mensaje (comunicación sincrónica). Una llamada a recibir con bloqueo obligara al receptor a esperar.

Envío sin bloqueo.- Permite al transmisor continuar con otro procesamiento aunque el receptor no haya recibido aún el mensaje (comunicación asincrona). Este tipo de envíos requiere un mecanismo de buffer para almacenar el mensaje hasta que lo reciba el receptor.
La comunicación asíncrona sin bloqueo aumenta el nivel de concurrencia.
Una llamada a recibir sin bloqueo permite que el receptor continúe con otro procesamiento antes de volver a intentar la recepción.

Cuando se habló de comunicación entre procesos en el mismo computador siempre damos por asentado una transmisión sin errores, sin embargo, en los sistemas distribuidos la transmisión puede tener errores, incluso puede perderse, por ello los transmisores y receptores cooperan muchas veces utilizando un *protocolo de reconocimiento* para confirmar la recepción correcta de cada transmisión.
El transmisor que espera un reconocimiento del receptor puede utilizar un mecanismo de tiempo, cuando expira el tiempo si es que no ha llegado el reconocimiento el transmisor puede retransmitir el mensaje.
Una complicación de los sistemas distribuidos con paso de mensajes es la necesidad de dar nombres en forma no ambigua a los procesos, de forma que las llamadas explícitas a enviar y recibir hagan referencia a los procesos correctos.
La creación y destrucción de procesos se puede coordinar mediante algún mecanismo centralizado de asignación de nombres pero tal cosa puede introducir una sobrecarga considerable de transmisión ya que cada maquina deberá solicitar permiso para utilizar nuevos nombres.
Una alternativa es asegurar que cada computador asigne nombres únicos a sus propios procesos; de esta manera puede hacerse referencia a los procesos combinando el nombre del computador con el nombre del proceso. Lograrlo, requiere un control centralizado para determinar un nombre único para cada computador de un sistema distribuido, pero esto, no representa una sobrecarga importante, pues los computadores se añaden o eliminan de las redes con muy poca frecuencia.
Las comunicaciones a base de mensajes en los sistemas distribuidos presentan serios problemas de seguridad, uno de ellos es el problemas de comprobar la autenticidad ya que siendo receptor o transmisor en realidad no se sabe si son impostores los que transmiten o los que reciben los mensajes.

Buzones y puertos
La comunicación a base de mensajes se puede establecer directamente entre receptor y transmisor, o pueden utilizar colas intermedias. En el caso de las colas intermedias, el transmisor y receptor especifican la cola, en lugar del proceso, con que se van a comunicar.

Buzón.- Es una cola de mensajes que puede ser utilizada por múltiples transmisores y receptores.
En un sistema distribuido, los receptores pueden estar situados en muchos computadores, por lo cual requieren dos transmisiones para la mayor parte de los mensajes, una del transmisor al buzón y otra del buzón al receptor. El problema se resuelve dando a cada receptor un puerto.

Puerto.- Se define como un buzón utilizado por múltiples transmisores y un solo receptor.
En los sistemas distribuidos, cada despachador tiene normalmente un puerto en el cual recibe solicitudes de muchos clientes.

Conductos.- Unix introdujo la noción de conductos o pipes se como un esquema para manejar comunicaciones entre procesos. Conducto es en esencia un buzón que permite extraer un número especificado de caracteres a la vez. No tiene la noción de mensaje, de modo que,si el usuario desea simular un buzón de mensajes con un conducto, deberá leer cada vez el número apropiado de caracteres (el tamaño del mensaje). Los conductos se utilizan con frecuencia para manejar las comunicaciones entre un proceso productor único y un solo consumidor.

Llamadas a procedimientos remotos.- La noción de llamadas a procedimientos remotos o RPC (remote procedure calls) se introdujo con el objeto de ofrecer un mecanismo estructurado de alto nivel para realizar comunicaciones entre procesos en sistemas distribuidos.
Con una llamada a un procedimiento remoto, un proceso de un sistema puede llamar a un procedimiento de un proceso de otro sistema. El proceso que llama se bloquea esperando el retorno desde el procedimiento llamado en el sistema remoto y después continua su ejecución desde el punto que sigue inmediatamente a la llamada.
El procedimiento llamado y el que llama residen en procesos distintos, con espacios de direcciones distintos, por lo cual no existe la noción de variables globales compartidas, como en los procedimientos normales dentro de un solo proceso. Por lo tanto, las RPC transfieren la información estrictamente a través de parámetros de la llamada.
Una RPC puede realizarse aprovechando un mecanismo existente del tipo enviar/recibir. Las llamadas al procedimiento remoto sería:

```
enviar ( proceso_remoto,  parámetros_de_entrada);
recibir (proceso_remoto,  parámetros_de_salida);
```


PLANIFICACION DE TRABAJOS Y DEL PROCESADOR

En la planificación del procesador se estudian los problemas de cuándo asignar procesadores y a cuáles procesos asignarlos, también se considera la admisión de nuevas tareas en el sistema, la suspención y reactivación de procesos para ajustar la carga del sistema.

Niveles de planificación:

Planificación de Alto Nivel .- **(Planificación de trabajo)** Determina cuáles trabajos podrán competir activamente por los recursos del sistema o bien cuales trabajos deben admitirse por lo que también la llamamos planificación de admisión.

Planificación de Nivel Intermedio.- Determina que procesos pueden competir por el CPU. Responde a las fluctuaciones temporales en la carga del sistema mediante la suspensión temporal y la activación (reanudación) de procesos para lograr una operación mas fluída del sistema y ayuda a alcanzar ciertas metas globales del rendimiento del sistema. Actúa como amortiguador entre la admisión de los trabajos y la asignación del CPU a esos trabajos.

Planificación de Bajo Nivel.- Determina a cuál proceso listo se le asignará el CPU cuando éste se encuentre disponible y se encargará de asignar el CPU a ése proceso. Se lleva a cabo mediante el despachador y este debe residir en la memoria principal.

Objetivos de la planificación

En el diseño de una disciplina de planificación deben ser considerados varios objetivos, mismos que suelen caer en conflicto haciendo esto un tanto más complejo. Sin embargo a continuación se listan algunos objetivos:

Ser justa.- Una disciplina de planificación es justa si todos los procesos se tratan de la misma forma y ningún proceso se queda en aplazamiento indefinido.

Elevar al máximo la producción y el rendimiento.- Una disciplina de planificación debe de tratar de atender el mayor número posible de procesos por unidad de tiempo.

Aumentar al máximo el número de usuarios interactivos que reciben respuesta en tiempos aceptables (segundos).

Ser predecible.- Una tarea debe ejecutarse aproximadamente en el mismo tiempo y casi al mismo costo sea cual sea la carga del sistema.

Reducir al mínimo el gasto extra.- El gasto extra se considera por lo común un desperdicio de recursos, pero la inversión de cierta porción de los recursos del sistema como gasto extra puede mejorar en gran medida el rendimiento total del sistema.

Equilibrar el aprovechamiento de los recursos.- Los mecanismos de planificación deben mantener ocupados los recursos del sistema. Deben favorecerse los procesos que requieren los recursos poco utilizados.

Lograr un equilibrio entre la respuesta y el aprovechamiento.- La mejor manera de garantizar tiempos de respuesta adecuados es tener suficientes recursos disponibles en el momento en que son necesarios. El precio que debe pagarse por esta estratégia es que el aprovechamiento global de los recursos será pobre. En los sistemas de tiempo real, las respuestas rápidas son esenciales y el aprovechamiento de los recursos es menos importante. En otros tipos de sistemas, la economía exige un aprovachamiento efectivo de recursos.

Evitar el Aplazamiento Indefinido.- La mejor manera de evitarlo es el empleo del *envejecimiento*, es decir, mientras un proceso espera un recurso, su prioridad debe crecer. En algún momento, la prioridad será tan alta que el recurso se asignará al proceso.

Imponer prioridades.- En los ambientes en que se asignan prioridades a los procesos, los mecanismos de planificación deben favorecer a los procesos de alta prioridad.

Dar preferencia a los procesos que ocupan recursos decisivos.- Aunque un proceso tenga baja prioridad, podría estar ocupando un recurso decisivo, y el recurso puede ser requerido por un proceso de alta prioridad. Si el recurso no es apropiable, el mecanismo de planificación debe dar al proceso un trato mejor del que se le daría comunmente, de tal modo que libere el recurso con más rapidez.

Dar un mejor trato a los procesos, muestra un comportamiento deseable. Ejemplo: bajas tasas de paginación.

Degradarse paulatinamente con las cargas pesadas.- Un mecanismo de planificación no debe desplomarse bajo el peso de una carga fuerte en el sistema. Debe evitar la carga excesiva impidiendo la creación de procesos nuevos cuando la carga es pesada, o bien debe dar servicio a la carga mayor con una reducción moderada del nivel de

atención a todos los procesos.

Criterios de planificación

La limitación de un proceso por E/S.- Cuando un proceso obtiene el CPU ¿lo usará en forma breve antes de generar una petición de E/S?

La limitación de un proceso por el CPU.- Cuando un proceso obtiene el CPU, ¿tiende a usarlo hasta que expira su cuanto de tiempo?

Si un proceso es por lotes o es interactivo.- Los usuarios interactivos suelen hacer peticiones "triviales" que deben atenderse de inmediato para garantizar tiempos de respuesta adecuados. Los usuarios por lotes, por lo general pueden tolerar retrasos razonables.

Qué tan urgente es la respuesta.- Un proceso por lotes que tarda toda la noche no requerirá una respuesta inmediata. Un sistema de control de procesos en un tiempo real que supervise a una refinería de petróleo necesita una respuesta rápida, quizá para evitar una explosión.

Las prioridades de los procesos.- Los procesos de alta prioridad deben recibir mejor tratamiento que los de baja prioridad

La frecuencia con la que un proceso está generando fallas de página.- Teoricamente, un proceso que genera pocas fallas de página tiene acumulados sus conjuntos de trabajo en el almacenamiento principal. Pero los procesos que experimentan muchas fallas de página no han establecido aún sus conjuntos de trabajo, y lo convencional es favorecer a los procesos que ya los han establecido. Otro punto de vista es que los procesos con altas tasas de fallas de página deben tener preferencia, ya que usan poco el CPU antes de generar una petición de E/S.

Las frecuencias con las que los recursos de un proceso son apropiados por otro de mayor prioridad.- Los procesos cuyos recursos son apropiados muy a menudo deben recibir un tratamiento menos favorable. La cuestión es que cada vez que el sistema operativo invierte trabajo extra en echar a andar este proceso, el breve lapso de ejecución que se logra antes de la apropiación no justifica el gasto extra necesario para echar a andar el proceso en primer lugar.

Cuanto tiempo real de ejecución ha recibido el proceso.- Algunos diseñadores opinan que deben favorecerse los procesos con poco tiempo de ejecución. Otro punto de vista es que un proceso que ha recibido mucho tiempo