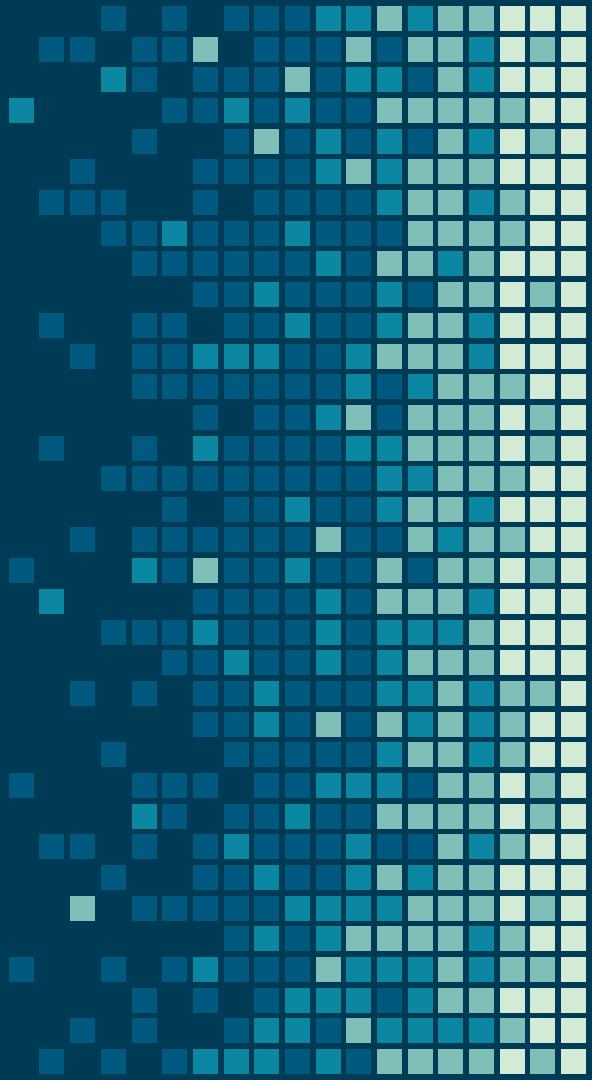
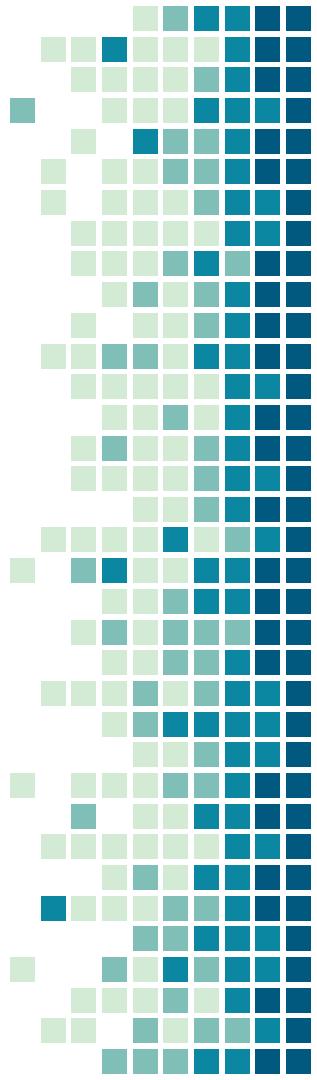


Procesos e Hilos en un Sistema Operativo



Agenda

- Introducción
- Procesos en un SO
- Comunicación entre procesos
- Modelo de multiprogramación
- Hilos
- Condición de carrera
- Soluciones a la condición de carrera



¿Cómo se hace valer la primera garantía de la Multiprogramación?



Introducción



Introducción

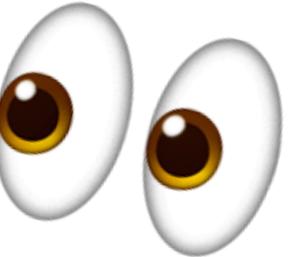
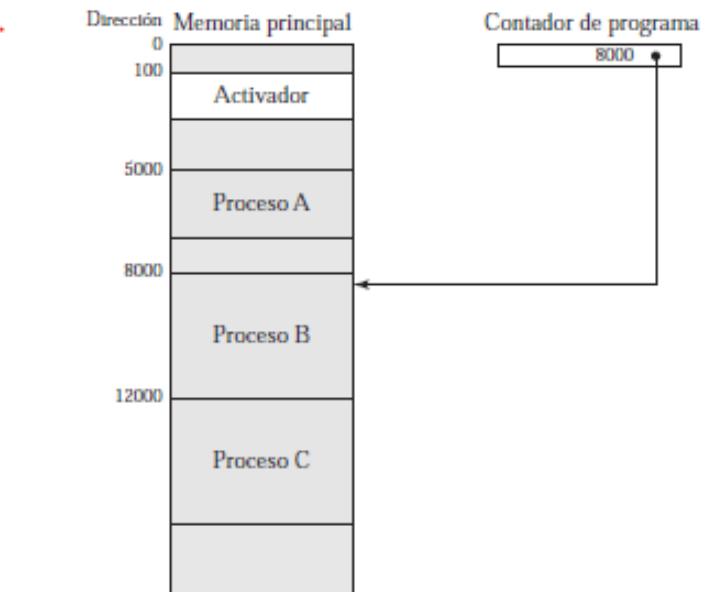


¿Cómo se hace para ejecutar varias aplicaciones al mismo tiempo en un computador?

¿Cómo se hace para comunicar dos aplicaciones?

¿Cómo se asignan los recursos computacionales?

¿Cómo se ve la RAM?



Procesos en un Sistema Operativo



Proceso en un SO

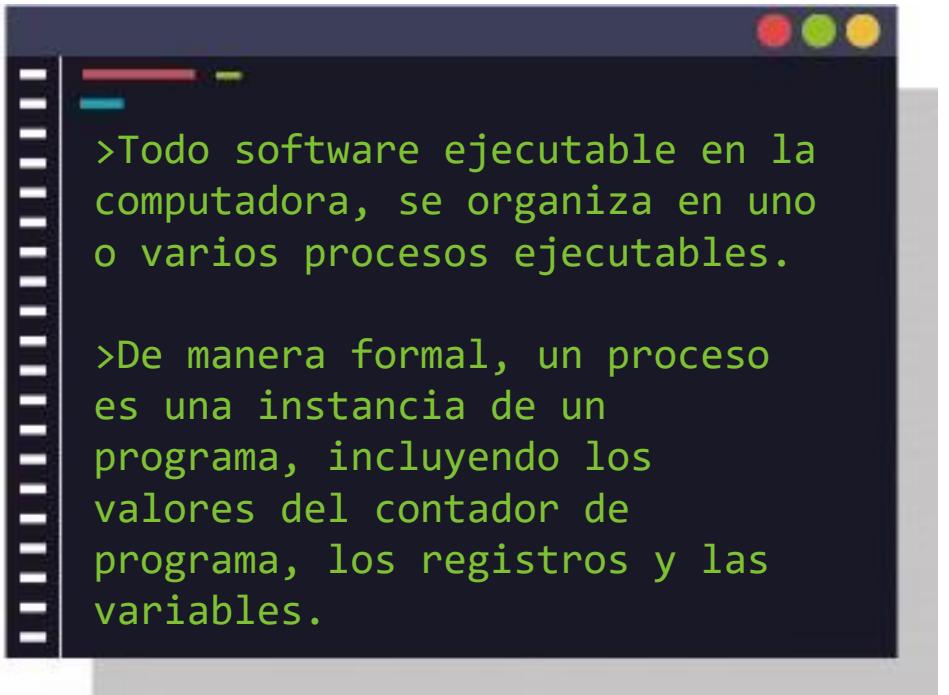


En palabras sencillas es una abstracción de un programa en ejecución.

Proporcionan la capacidad de operar concurrentemente con un único CPU.

- Uno de los conceptos más importantes en sistemas operativo.

Proceso



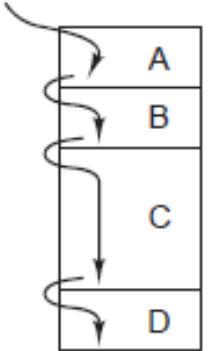
“Cada proceso posee un CPU virtual”.

Reflexión: ¿Es correcto decir que los programas se encuentran en memoria principal?



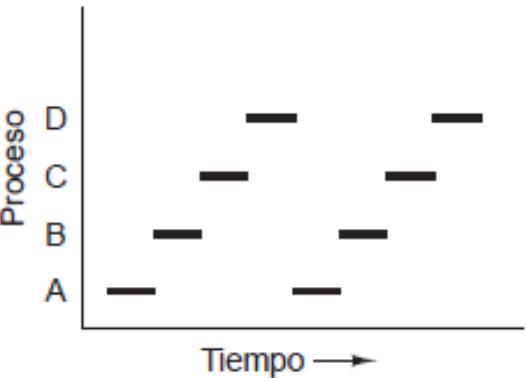
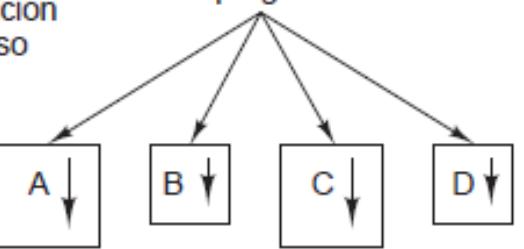
Proceso

Un contador de programa

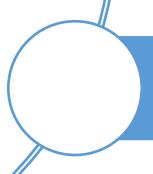


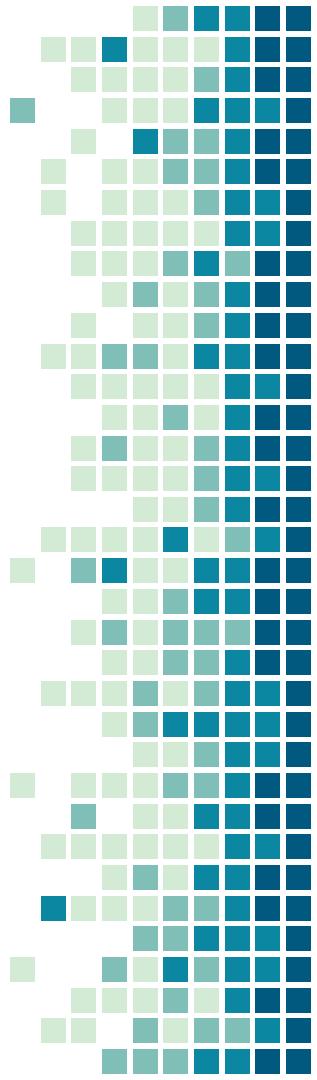
Conmutación
de proceso

Cuatro contadores
de programa



Proceso

-  Se da la conmutación de procesos en el procesador.
-  ¿Cuánto tarda el procesador en conmutar un proceso?
-  ¿Qué se podría decir de la rapidez de los procesos en un procesador en específico?



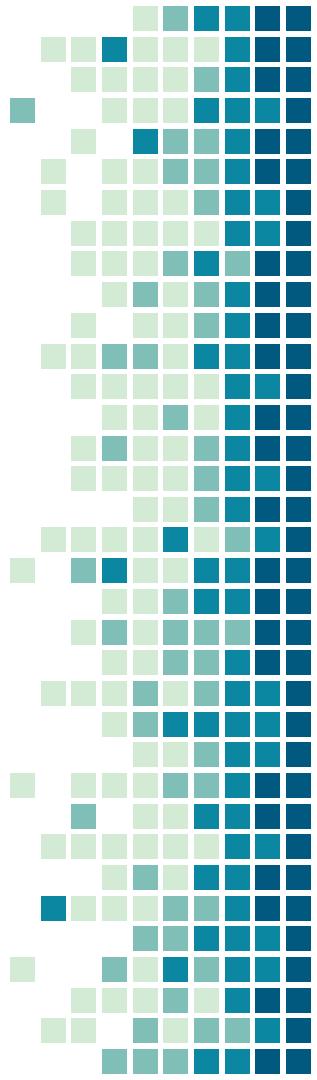
Proceso VS Programa



Un científico computación con mente culinaria hornea un pastel.

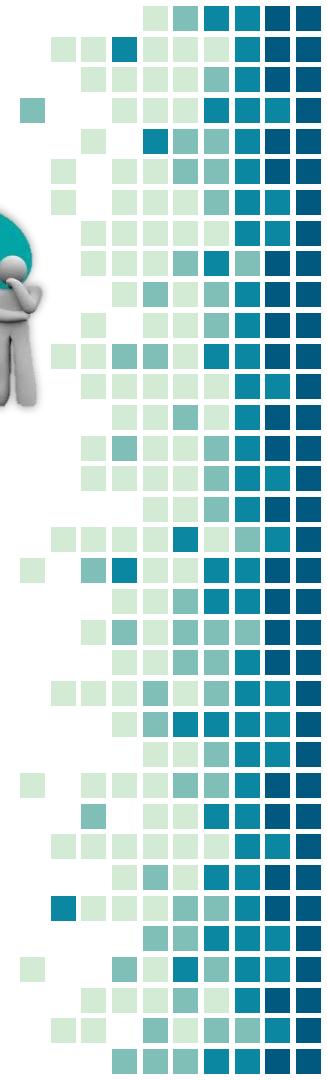
Posee la receta del pastel, así como una cocina muy equipada.

También posee todos los ingredientes necesarios: harina, huevos, azúcar, vainilla, entre otros.



¿Quién es quién ?

- La receta es el programa (algoritmo expresado de cierta manera específica).
- El científico juega el papel del CPU.
- Los ingredientes del pastel son los datos de entrada.
- La cocina es el Hardware necesario.



Eventos

Suponga que el hijo del científico entra gritando que se cortó.

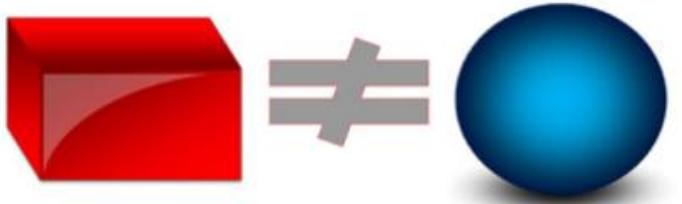
¿Qué debería hacer el científico? Suponiendo que tiene un libro de primeros auxilios.



Diferencia puntual entre proceso y programa

El programa son las instrucciones (algoritmo)

Mientras que...



El proceso es la ejecución con los elementos necesarios (datos de entrada y contador)

Creación de procesos en un SO



1

En el arranque
del sistema.



2

La ejecución,
desde un
proceso, de
una llamada al
sistema para
la creación de
otro proceso.



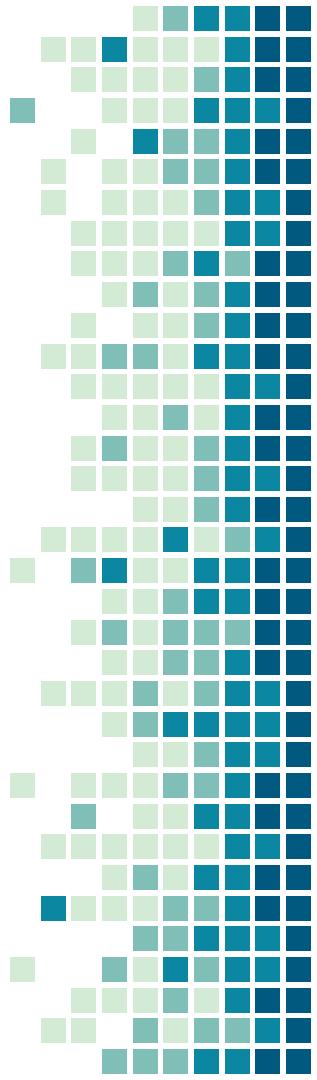
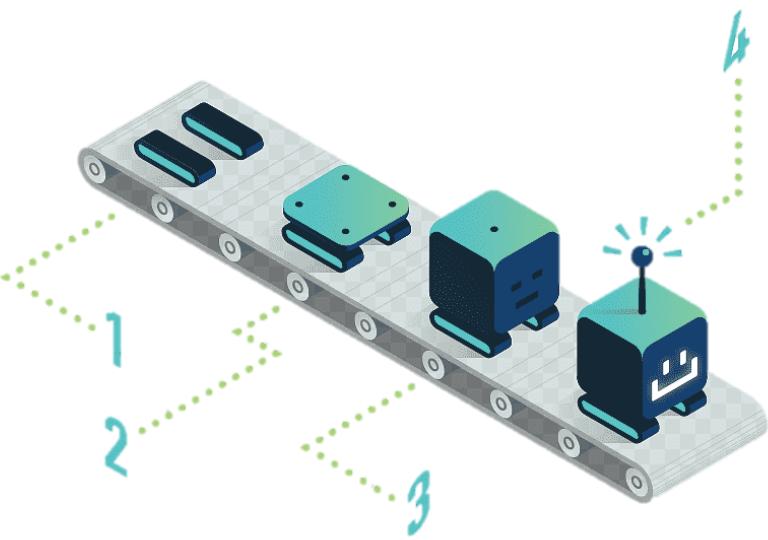
3

Una petición
de usuario
para crear un
proceso.

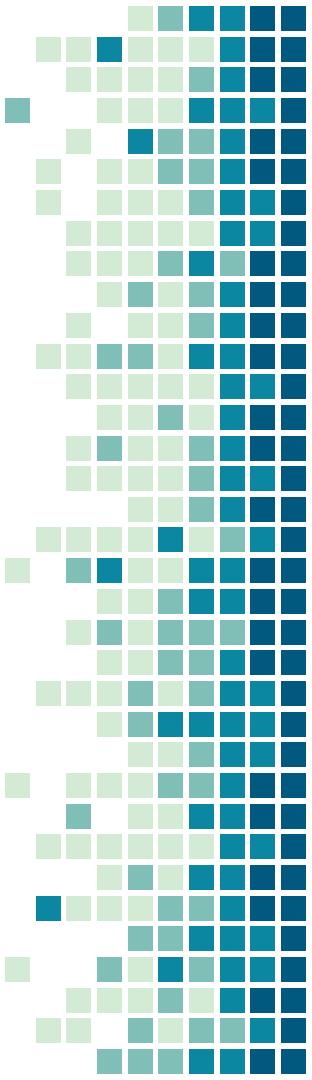
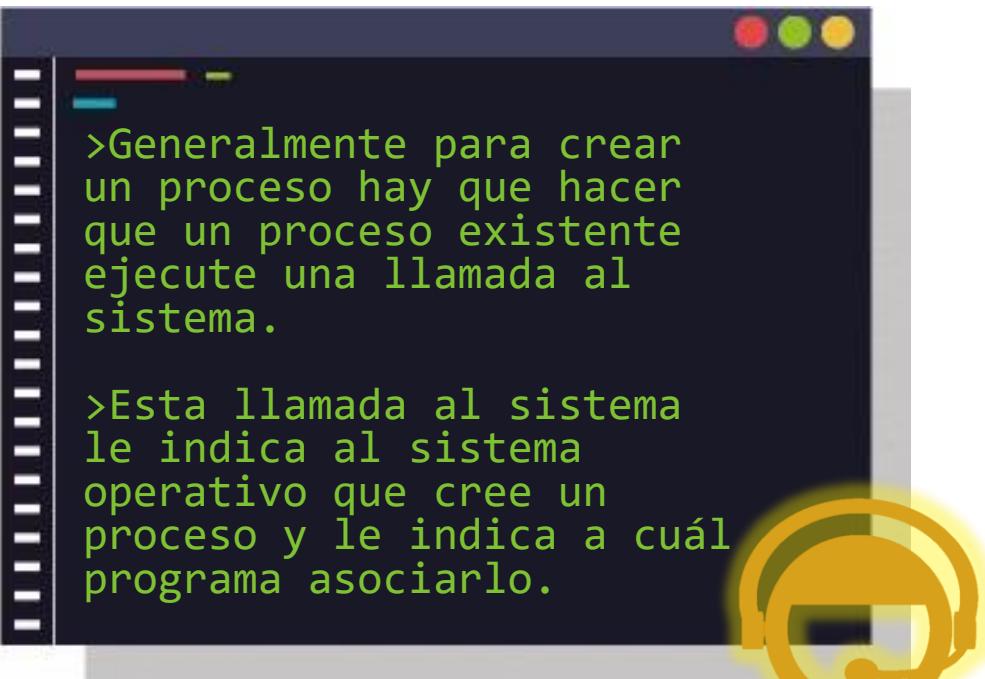


4

El inicio de un
trabajo por
lotes.



Creación de procesos



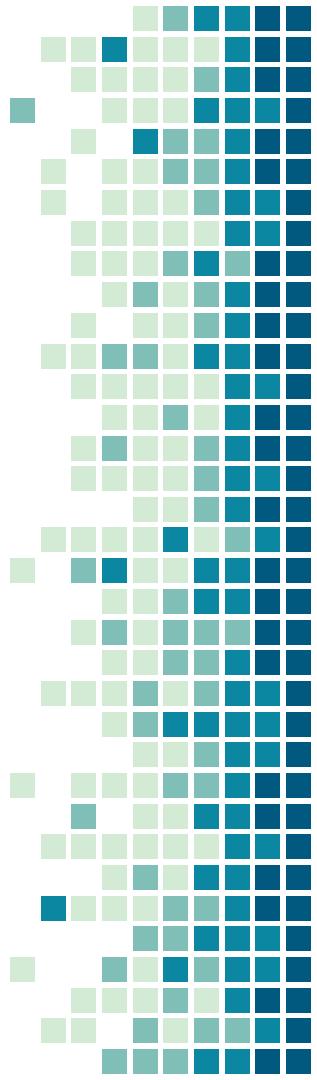
Fork



- Es la llamada al sistema que utiliza UNIX para crear un nuevo proceso.
- Crea un clon exacto del proceso que realizó la llamada.
- Los dos procesos tienen la misma imagen de memoria.
- El proceso creado ejecuta otra llamada (execve) que cambia la imagen de memoria y ejecuta el programa correspondiente.

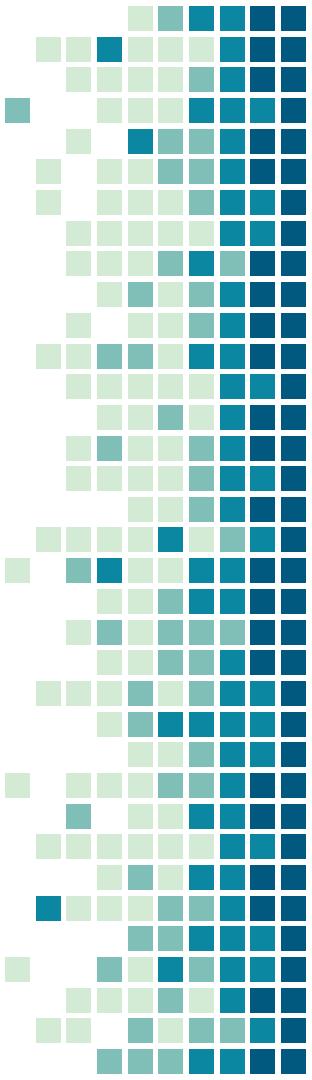
CreateProcess

- Es la llamada al sistema que utiliza Windows para crear un nuevo proceso.
- Crea un clon exacto del proceso que realizó la llamada y carga de una vez el programa correcto.
- Posee más parámetros que fork, alrededor de 10.



CreateProcess Vs Fork

- Una vez que se crea el proceso, los espacios de direccionamiento son distintos.
- En Windows el espacio de direccionamiento es distinto desde el inicio, en Unix está el paso intermedio.
- No se comparte memoria, a excepción del inicio en Unix



Demonios (Daemons)



- Son procesos que pertenecen en segundo plano para administrar ciertas actividades como el correo electrónico, impresiones, antivirus entre otros.
- Generalmente el sistema operativo posee docenas de demonios realizando tareas que son necesarias, pero que no se sabe que se están ejecutando.

Terminación de procesos

Salida normal.

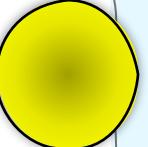
Error fatal.



Salida por error.

Eliminado por otro
proceso.

Salida normal

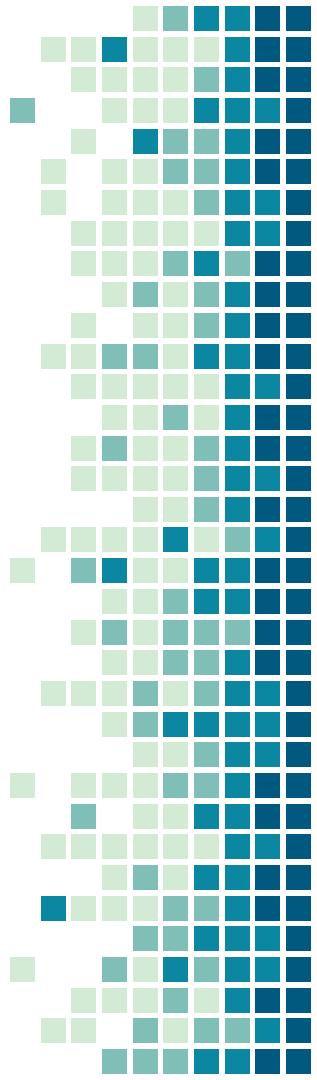
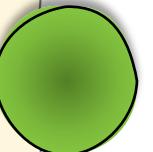


✓ Se refiere cuando los procesos terminan debido a que han concluido su trabajo o que el usuario cierre y termine el proceso.

✓ Cuando un compilador termina de compilar un código en C ejecuta una llamada al sistema para indicar que ya finalizó el trabajo. Exit en UNIX, mientras que en Windows ExitProcess.

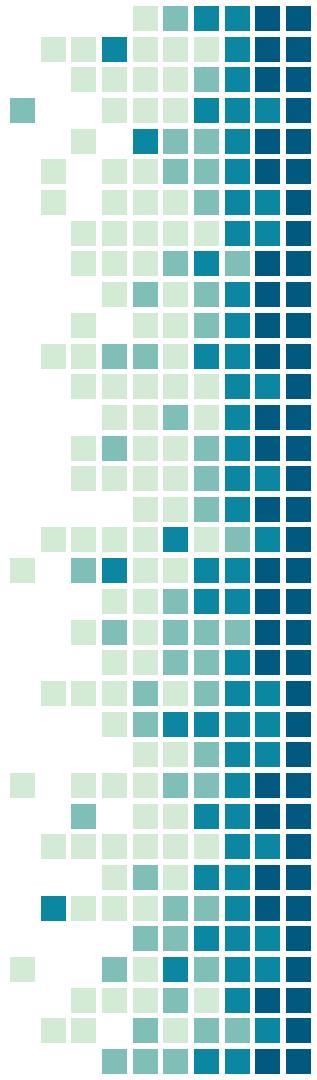
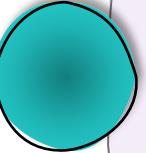
Salida por error de usuario

- ✓ El proceso se termina porque se descubre un error.
- ✓ Por ejemplo cuando se quiere escribir el comando para compilar el archivo test.c y el archivo no existe se produce la llamada al sistema para terminar el proceso.



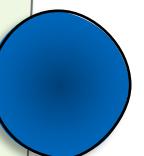
Error fatal

- ✓ Se refiere a un error en el programa.
- ✓ Por ejemplo cuando se ejecuta una instrucción ilegal, hacer referencia a tramo de memoria no existente o realizar una división por cero.



Eliminado por otro proceso

- ✓ Es el caso de cuando un proceso podría ejecutar una llamada al sistema para eliminar otros procesos.
- ✓ En UNIX la llamada es kill, mientras que en Windows es TerminateProcess

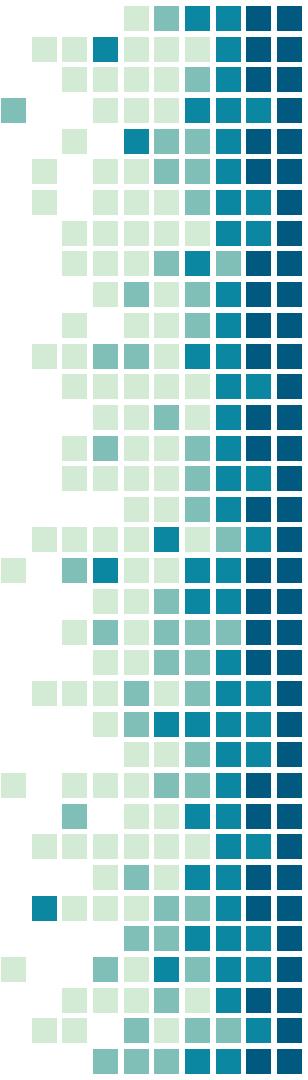


Jerarquía de procesos

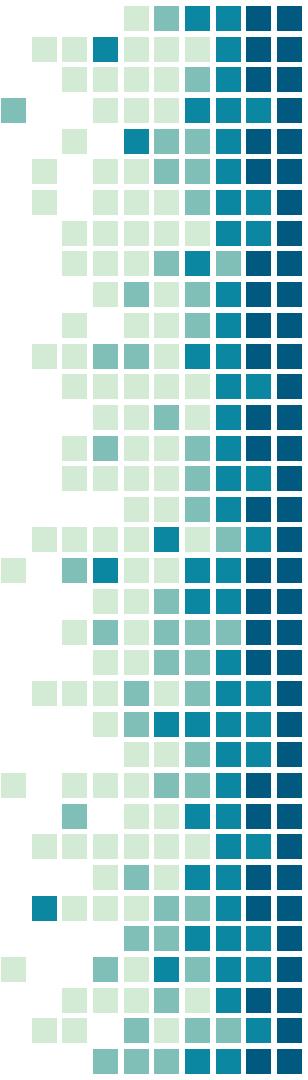
¿Qué pasa cuando se crea un proceso en Linux?

¿Qué pasa en el arranque de Linux?

¿Qué ocurre en el caso de Windows?



Estados de un proceso

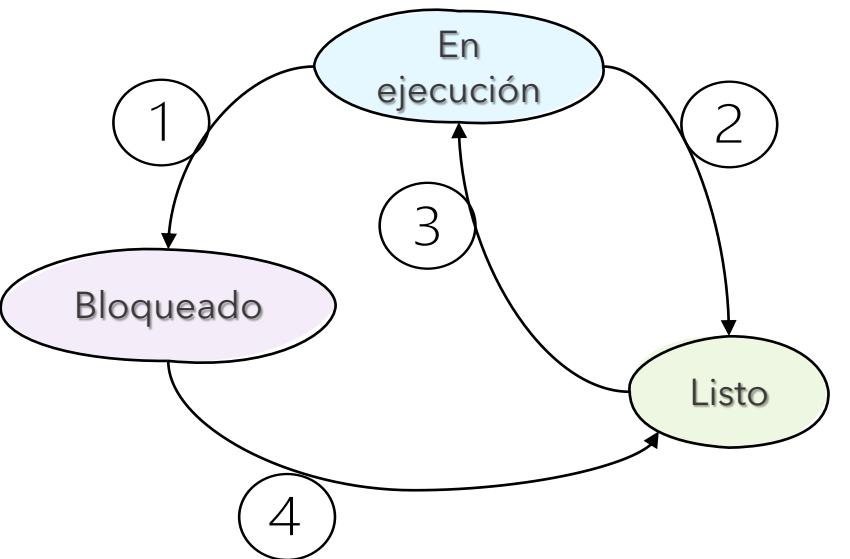


Cada proceso es independiente,
posee su propio contador de
programa y estado interno.

¿Un proceso necesita
comunicación con otro?

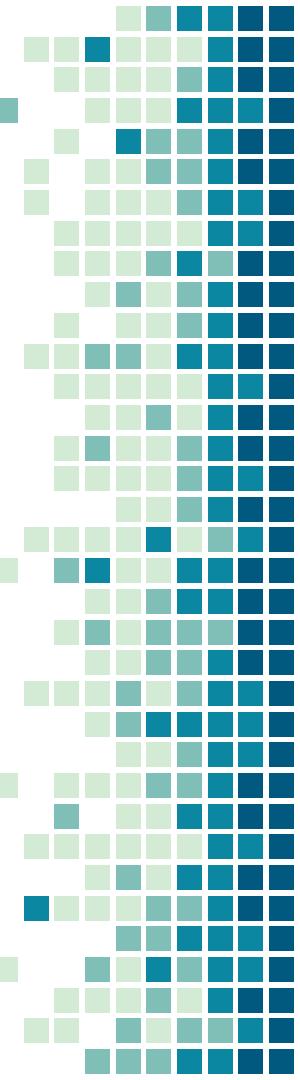
Existen dos modelos:
simplificado y completo.

Modelo simplificado de los estados de un proceso.



- 1: Proceso se bloquea para recibir la entrada o recurso.
- 2: El planificador selecciona otro proceso.
- 3: El planificador selecciona este proceso.
- 4: La entrada o recurso ya está disponible.

Modelo de 5 estados (Completo).



Nuevo: Un proceso que se acaba de crear, pero no está admitido en el grupo de procesos ejecutables.



Listo: Un proceso que se prepara para ejecutar cuando pueda.



Bloqueado: Un proceso que no se puede ejecutar hasta que se cumpla un evento determinado o se complete la entrada.

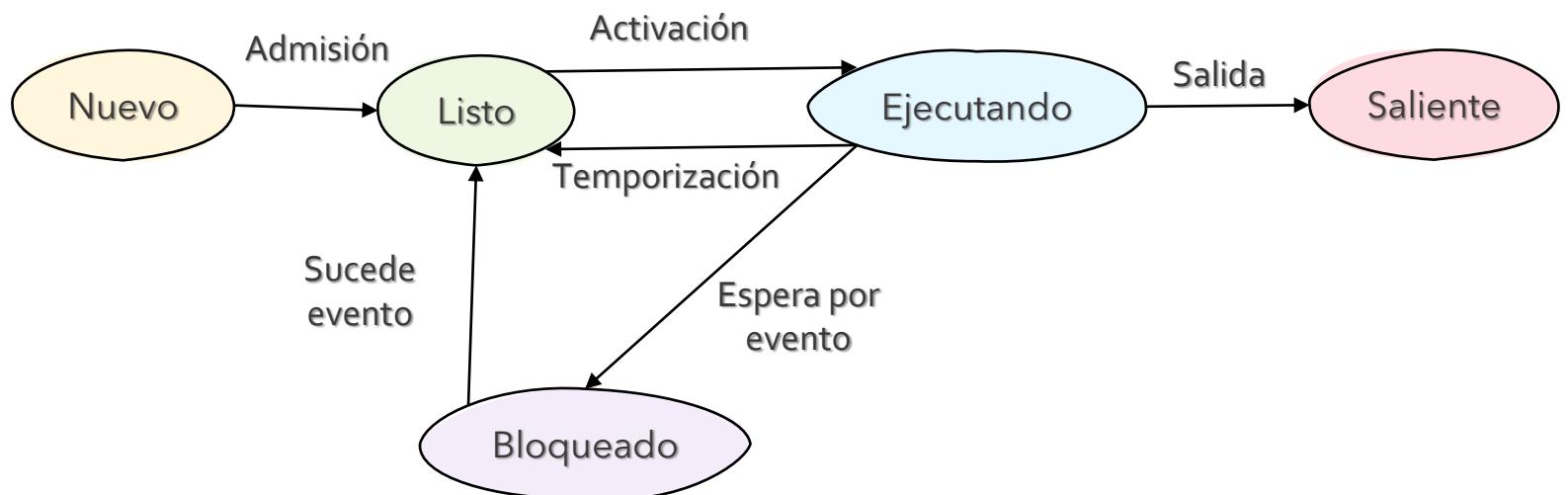


Ejecutando: El proceso se está ejecutando.

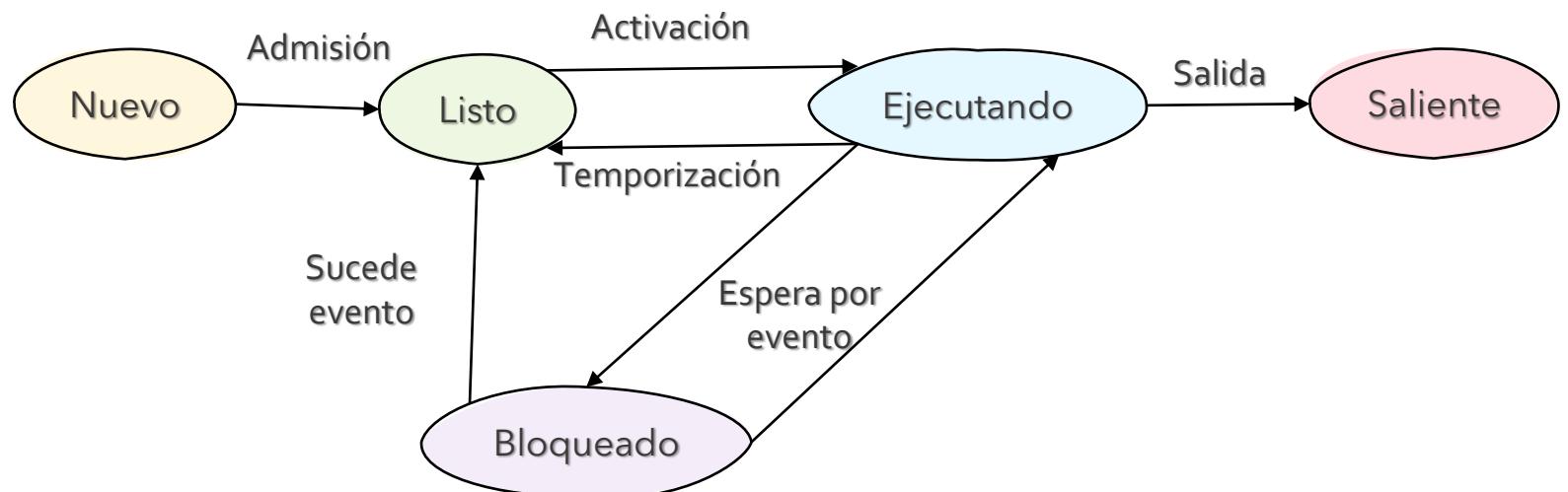


Saliente: Se libera del grupo de procesos ejecutables.

Modelo de proceso completo.



Tarea Moral: Tiene sentido colocar la siguiente transición



Transición de Null a Nuevo

Se crea un nuevo proceso para ejecutar un programa.

Ocurre con algunas de las razones antes vistas.



Transición de Nuevo a Listo

- El SO realiza esta transición cuando ya se encuentre preparado para ejecutar un nuevo proceso. Contadores de programas listos, tablas, direcciones, registros y todos los elementos necesarios.
- Generalmente se tiene una política para una cantidad máxima de procesos.

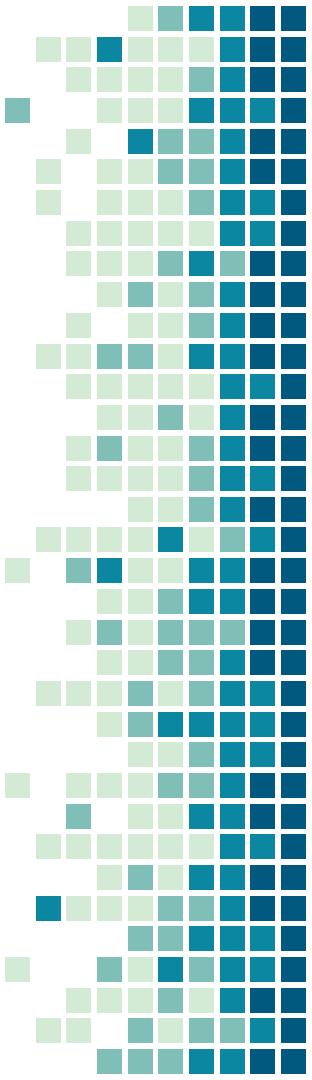
💡 ¿Qué ocurre si no existiera una cantidad máxima?



Transición de Listo a Ejecutando

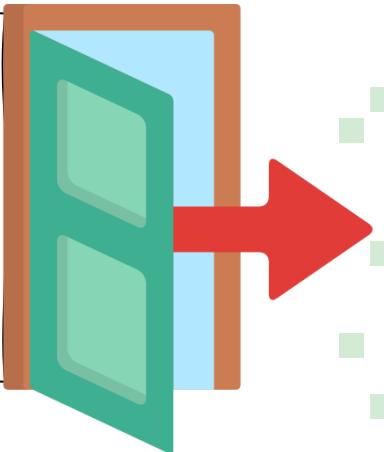


Se refiere al momento de seleccionar un nuevo proceso para ejecutar. El SO selecciona un proceso que se encuentre en estado de listo según las reglas del planificador.



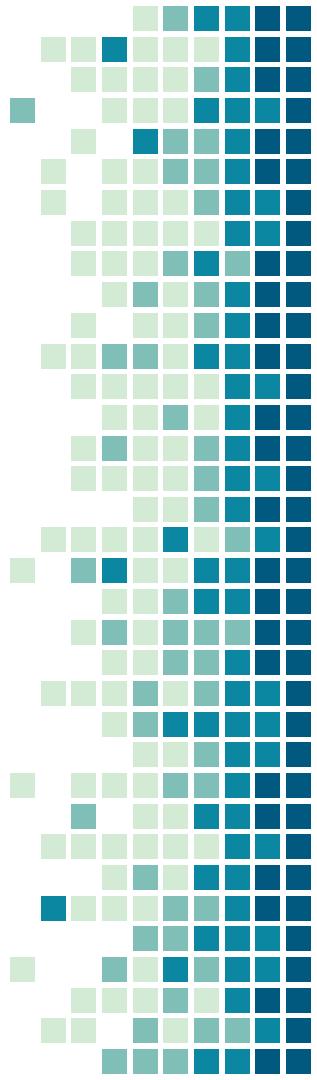
Transición de Ejecutando a Saliente

- El proceso que está en ejecución se finaliza por parte del SO, tanto si el proceso ha terminado con éxito su ejecución como si es por alguna razón que hizo que abortara.



Transición de ejecutando a listo

La principal razón para esta transición es que el proceso en ejecución haya alcanzado el tiempo máximo de pertenencia en el CPU.



Transición de ejecutando a bloqueado

- Esta transición se da porque el proceso solicita algún recurso (datos o hardware) por el cual debe esperar.
- Generalmente se realiza por medio de una llamada al sistema.



Transición de bloqueado a Listo

Se realiza esta transición cuando sucede el evento por el cual estaba esperando.

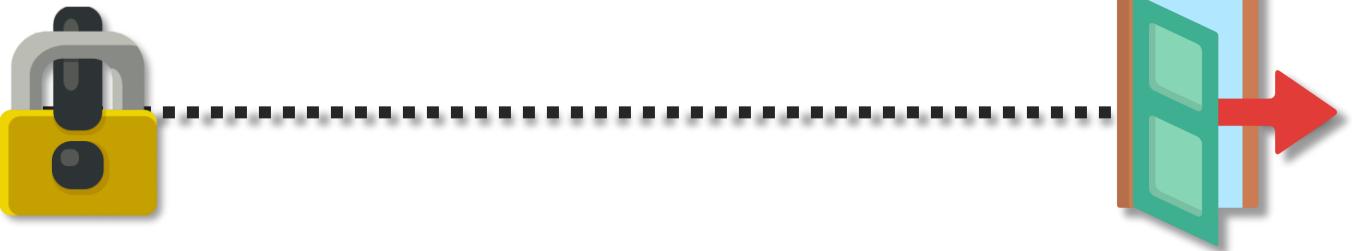


* Transición de Listo a Saliente

- ☞ Esta transición es muy particular, de hecho no está en el modelo, la implementan algunos sistemas operativos.
- ☞ Es cuando el padre puede terminar el proceso del hijo en cualquier momento.

Terminación de procesos

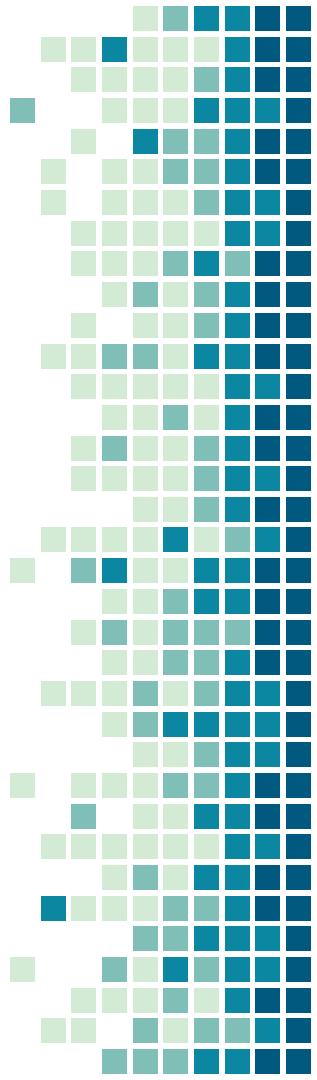
Se da si por alguna razón se debe finalizar el proceso, aplica el mismo concepto que la transición anterior.



Cambio de Proceso

- A primera vista un cambio de proceso parece sencillo. "Al final de cuentas es sólo decidir cuál proceso sigue y ponerlo a ejecutar y listo".

¿Qué se podría tomar en cuenta para realizar este procedimiento?

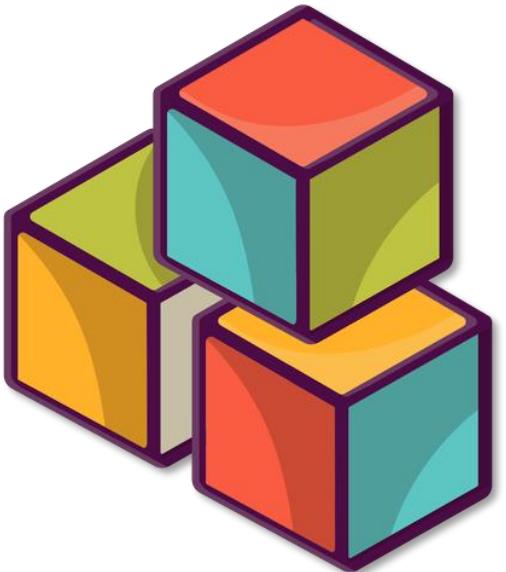


Datos generales de un proceso



- Identificador.
- Estado.
- Prioridad.
- PC.
- Punteros a memoria.
- Datos de contexto.
- Información de auditoría.
- Información de E/S.

Bloque de control de proceso (PCB)

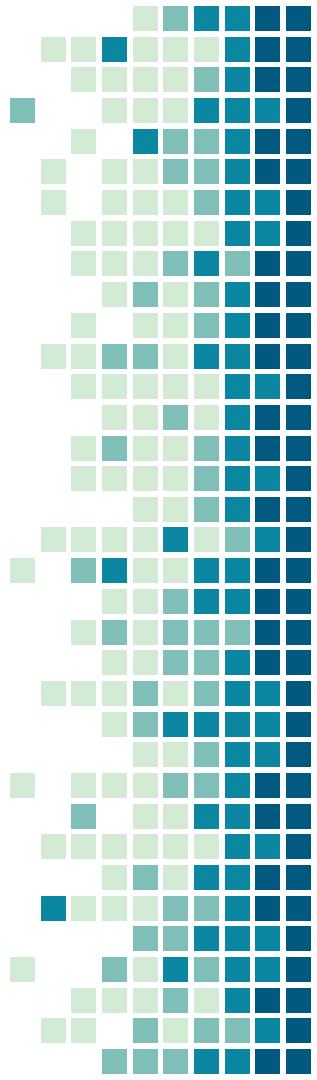


- Una de las estructuras más importantes del sistema operativo, contiene toda la información sobre un proceso que necesita saber el sistema operativo.
- Entonces para hacer el cambio de proceso se necesita tener una tabla de PCB y con ello tener el control de todos las variables correspondientes a los procesos.

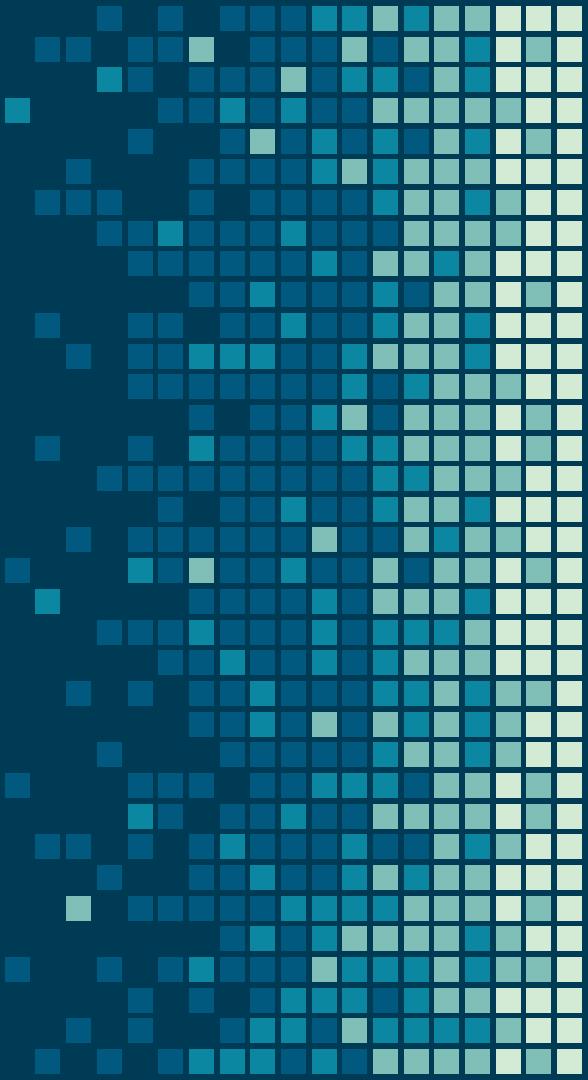
¿Qué contiene el PCB?

Identificador
Estado
Prioridad
Contador de programa
Punteros de memoria
Datos de contexto
Información de estado de E/S
Información de auditoría
⋮

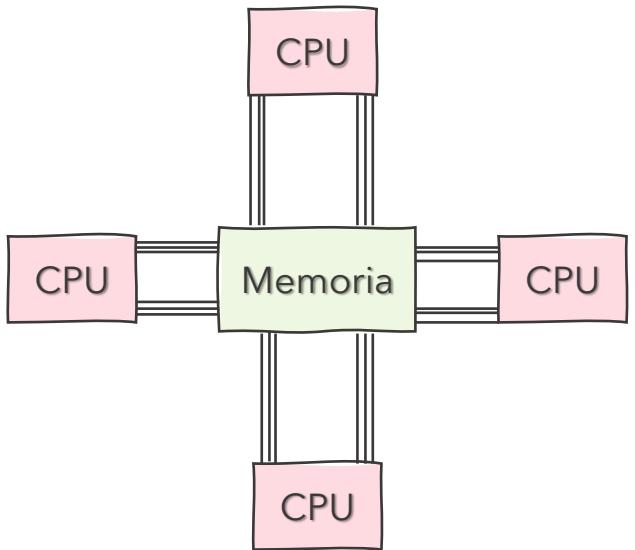
- Contador de programa
- Estado del proceso.
- Apuntadores de pila.
- Asignación de memoria.
- Información sobre archivos abiertos.
- Información de contabilidad y planificación.
- Identificadores y demás elementos necesarios.



Comunicación entre procesos



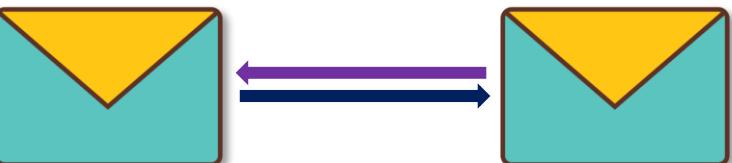
1. Memoria Compartida



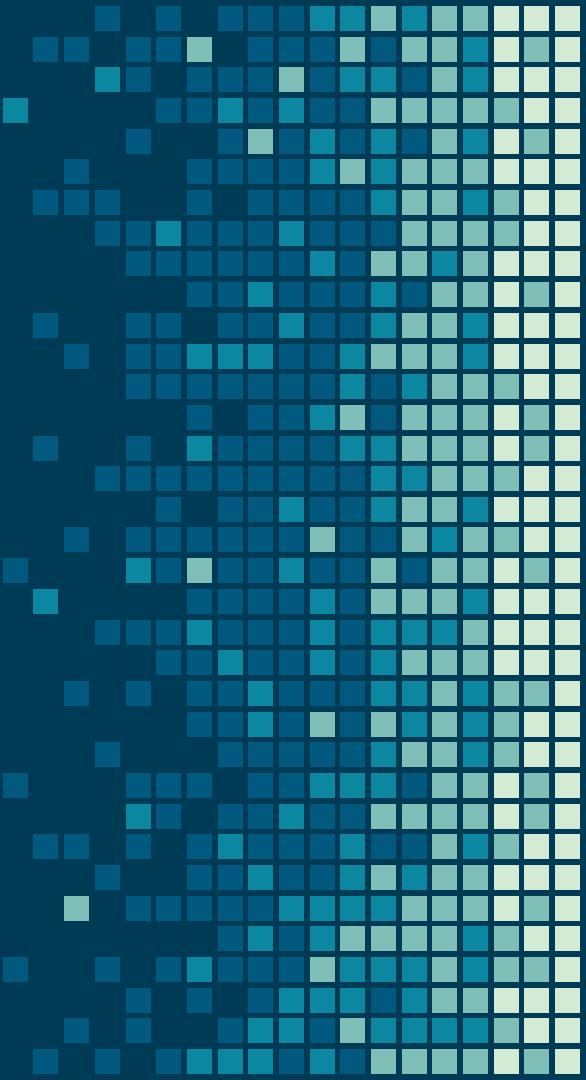
- Una región de memoria a la cual ambos procesos tienen acceso.
- Ambos procesos deben aceptar la restricción, con el fin de que ambos puedan leer y escribir.

2. Paso de mensajes.

- **Dos operaciones:** enviar y recibir datos en ambos procesos.
- Se debe establecer un enlace entre ambos.
- Comunicación directa e indirecta.
- Comunicación sincrónica o asincrónica.
- Bufferización.



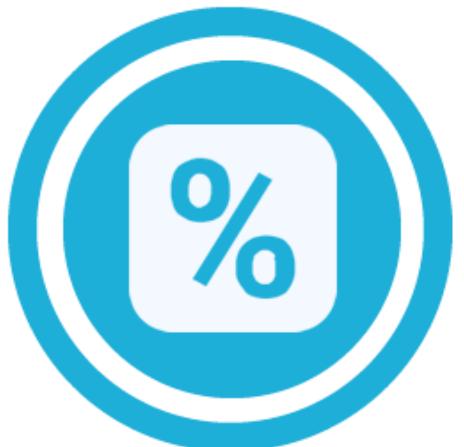
¿A qué se refiere el término Quantum?



Modelo de Multiprogramación



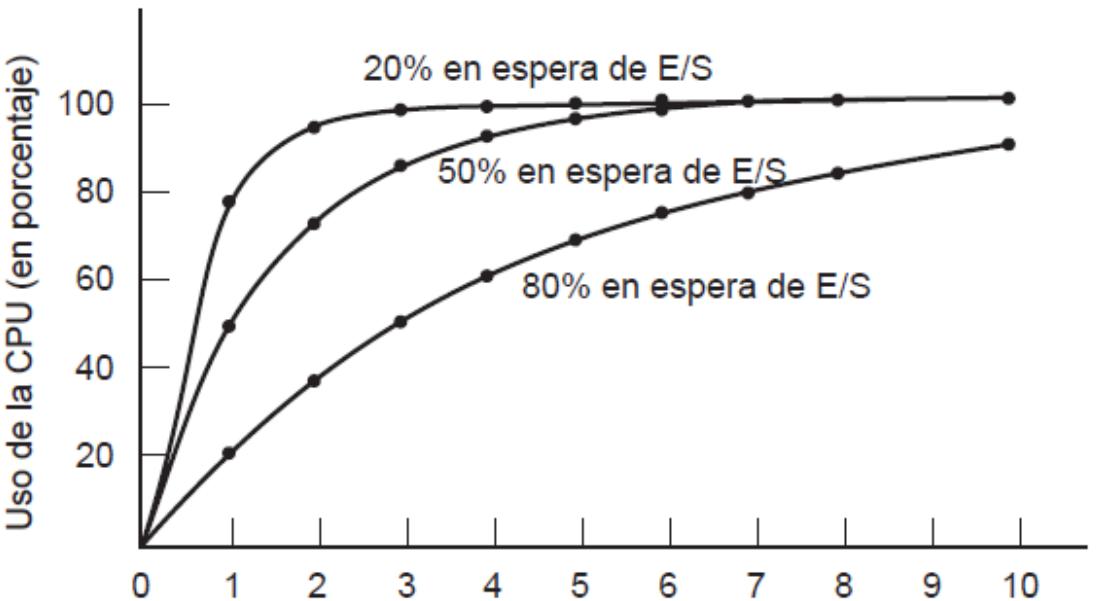
Recordando Probabilidad y Estadística



En Costa Rica, se ha detectado que existen 400 000 tipos de pieles en las personas, si en el 2021 hay 3 650 000 personas ¿ Cuál es la probabilidad de encontrar 2 personas con el mismo tipo de piel ?

Modelo de multiprogramación

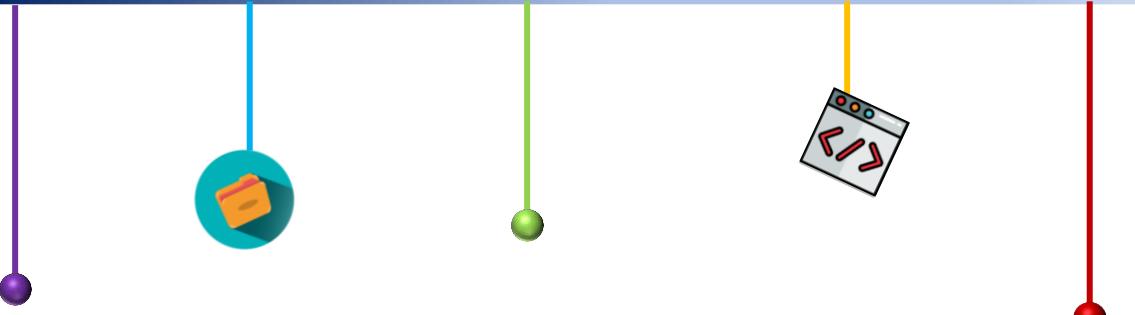
$$\text{Uso de la CPU} = 1 - p^n$$



Hilos: Los primos de los procesos



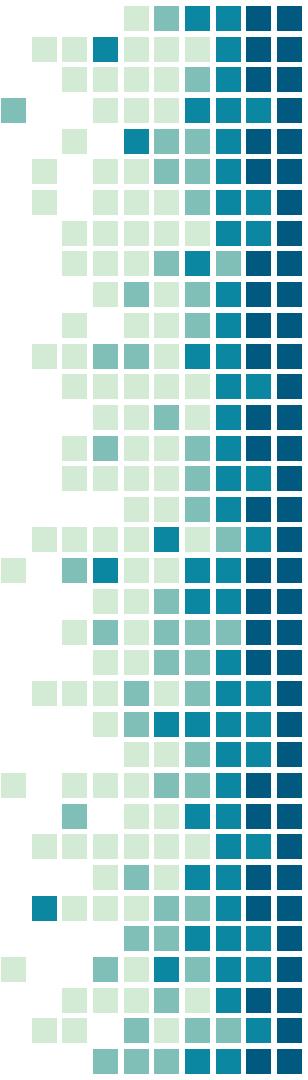
Hilos



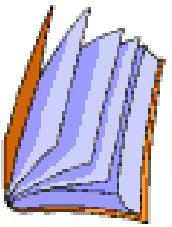
- En los sistemas tradicionales, cada proceso tiene un espacio de direccionamiento y un solo flujo de control.
- Son conocidos como procesos ligeros y corresponden a un programa en ejecución que comparten la imagen de memoria y otra información como código, archivos, dispositivos abiertos entre otros.
- Cada hilo posee un contador de programa y pila, pero son estructuras menos complejas que el PCB, y por ende livianas.

¿Por qué utilizar hilos?

- 👉 Habilidad de compartir espacios de direcciones y datos entre los diferentes procesos ligeros.
- 👉 Son más ligeros que los procesos, por lo que es más sencillo eliminarlos y crearlos.
- 👉 Útiles en sistemas multinúcleo.



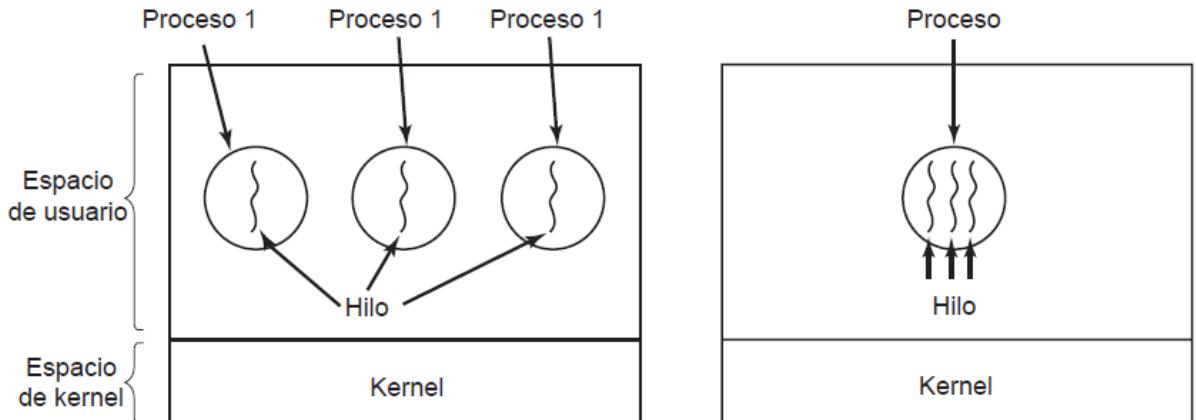
Ejemplo de la utilidad de los hilos



- Un procesador de palabras, por ejemplo Word.
- Suponga que se está escribiendo un libro.
- ¿Qué ocurre si hay que cambiar la palabra "ay" por "Hay"?
- ¿Qué ocurre si hay que dar formato específico a todas las viñetas?
- ¿Si se implementa con dos hilos mejora ?

¿En cuál caso hay paralelismo?

Estrictamente, en ninguno hay paralelismo real.



¿Qué se comparte y qué no?



A la izquierda los que se comparten por los hilos en un proceso y a la derecha lo que es propio de un hilo.

Elementos por proceso	Elementos por hilo
Espacio de direcciones	Contador de programa
Variables globales	Registros
Archivos abiertos	Pila
Procesos hijos	Estado
Alarmas pendientes	
Señales y manejadores de señales	
Información contable	

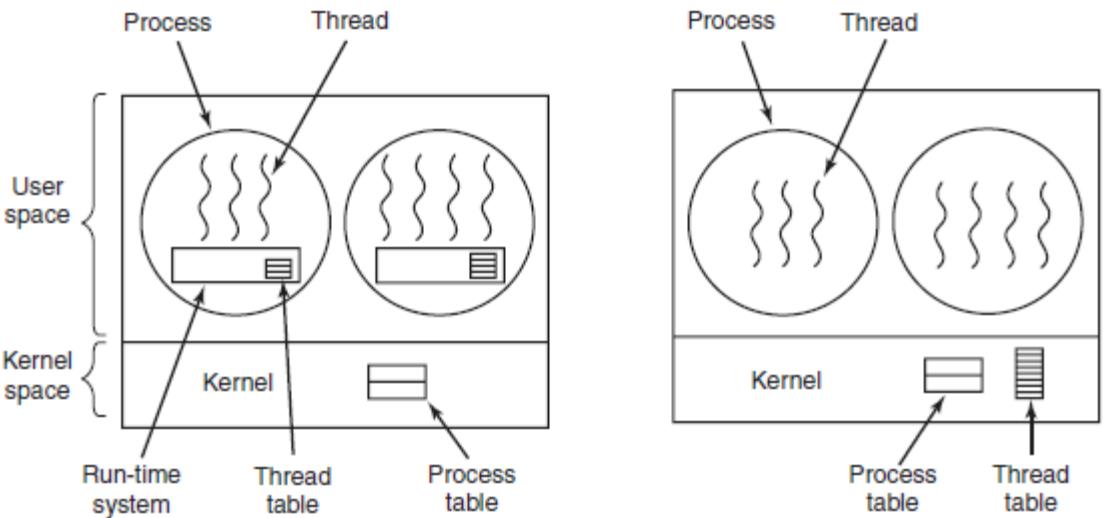
Hilos en POXIS

Llamada de hilo	Descripción
Pthread_create	Crea un nuevo hilo
Pthread_exit	Termina el hilo llamador
Pthread_join	Espera a que un hilo específico termine
Pthread_yield	Libera la CPU para dejar que otro hilo se ejecute
Pthread_attr_init	Crea e inicializa la estructura de atributos de un hilo
Pthread_attr_destroy	Elimina la estructura de atributos de un hilo

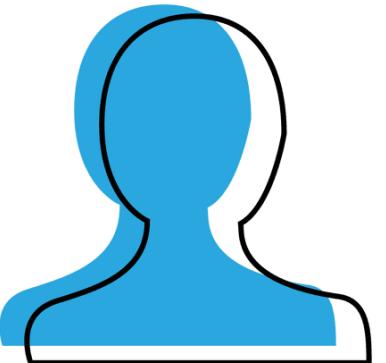
¿Opciones para implementar hilos?



Dos maneras de implementar hilos



Espacio de Usuario



- Generalmente se implementa como una biblioteca.
- Los hilos necesitan su propia "Tabla de PCB".
- Invocar el calendarizador o rutinas de ejecución es más sencillo.
- Permite configurar el calendarizador por proceso.

Espacio de Kernel

- ✓ El kernel debe utilizar un tabla de threads donde se almacene información de todos los hilos del sistema.
- ✓ Las llamadas que bloquean hilos, son implementadas como llamadas al sistema.
- ✓ Uno de los problemas es el manejo de las señales de los procesos.
- ✓ Modificar el sistema operativo no es muy recomendable.

¿Qué ocurre con recursos compartidos, qué se debe de realizar?



Condición de carrera



Condición de carrera.

Situación en donde dos o más procesos están leyendo o escribiendo algunos datos compartidos y el resultado final depende de quién se ejecuta y exactamente cuándo lo hace.



¿Cómo solucionar el problema de condición de carrera?

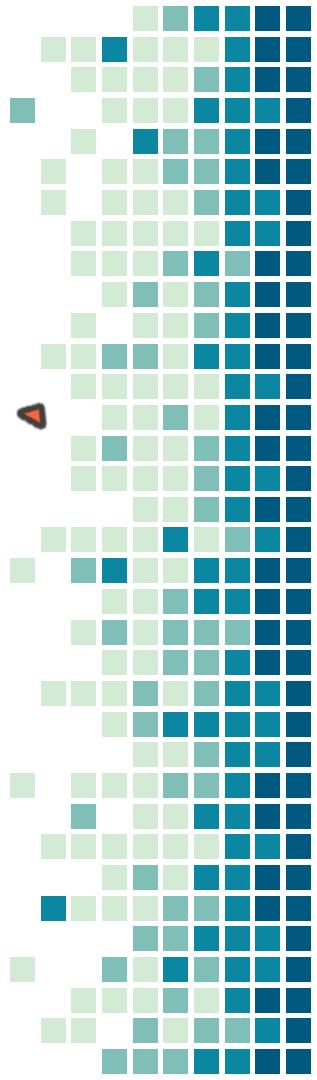


De alguna manera se deben sincronizar los procesos e hilos, con el fin de garantizar la manipulación de datos correctamente.

- La idea fundamental es que de cierta manera prohibir que más de un proceso lea y escriba al mismo tiempo un dato, es decir, exclusión mutua.

Región crítica de un programa.

- Es la parte del programa en la que se accede a la memoria o recurso compartido.
- Entonces si dos procesos nunca están en sus secciones críticas al mismo tiempo no habría condiciones de carrera.

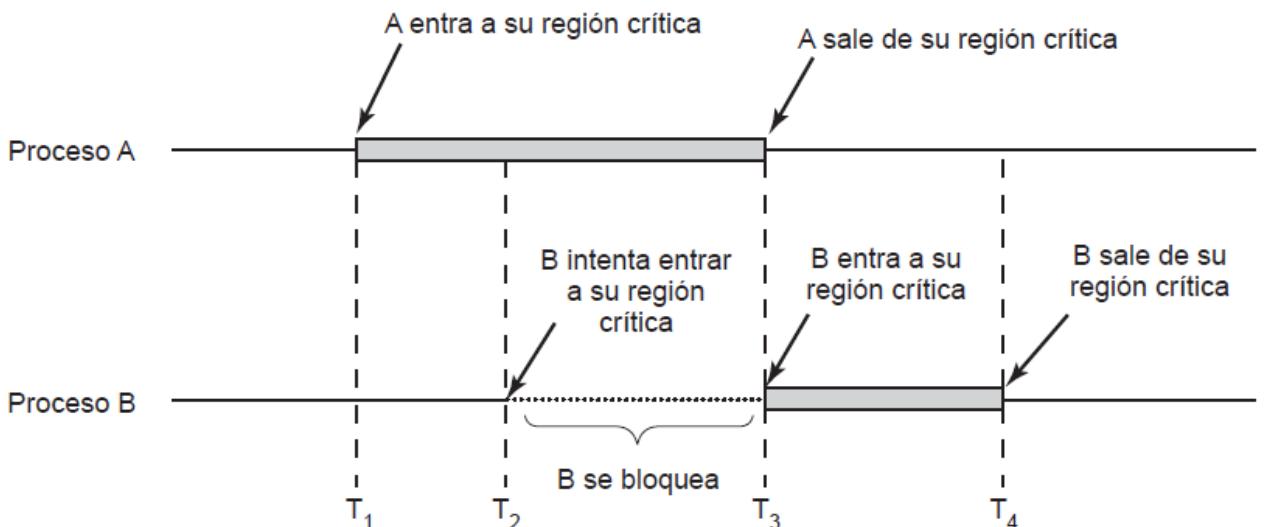


Condiciones para solucionar el problema de una manera eficiente.



- No puede haber dos procesos de manera simultánea dentro de sus regiones críticas.
- No pueden hacerse suposiciones acerca de las velocidades o el número de CPU.
- Ningún proceso que se ejecute fuera de su región crítica puede bloquear otros procesos.
- Ningún proceso tiene que esperar para siempre para entrar a su región crítica.

Exclusión mutua.



Solución 1: Deshabilitar interrupciones.

Esta solución no es muy atractiva.



☞ **Deshabilita las interrupciones justo después de entrar a la región crítica y luego las habilita justo después que sale.**

Solución 2: Variables de candado.

Consiste en tener una variable global compartida.

Cuando un proceso desea entrar a su región crítica primero evalúa el candado, si es 0, entra ,en caso contrario debe esperar.

Es una solución por software.



Solución 3: Alternancia estricta.

Se refiere a que debe tomar turnos

```
while (TRUE) {  
    while (turno != 0) /* ciclo */;  
    region_critica();  
    turno = 1;  
    region_nocritica();  
}
```

```
while (TRUE) {  
    while (turno != 1) /* ciclo */;  
    region_critica();  
    turno = 0;  
    region_nocritica();  
}
```

Solución 4: Peterson.

```
#define FALSE 0
#define TRUE 1
#define N 2                                /* número de procesos */

int turno;                                /* ¿de quién es el turno? */
int interesado[N];                         /* al principio todos los valores son 0 (FALSE) */

void entrar_region(int proceso);           /* el proceso es 0 o 1 */
{
    int otro;                               /* número del otro proceso */

    otro = 1 – proceso;                   /* el opuesto del proceso */
    interesado[proceso] = TRUE;          /* muestra que está interesado */
    turno = proceso;                     /* establece la bandera */
    while (turno == proceso && interesado[otro] == TRUE) /* instrucción nula */;
}

void salir_region(int proceso)              /* proceso: quién está saliendo */
{
    interesado[proceso] = FALSE;         /* indica que salió de la región crítica */
}
```

Semáforos.

Se utilizan para controlar el acceso a un recurso por parte de un número finito de instancias



Proceso A



Boqueado



Proceso B



Despierto



Existen dos operaciones down y up (Proberen y Verhogen).

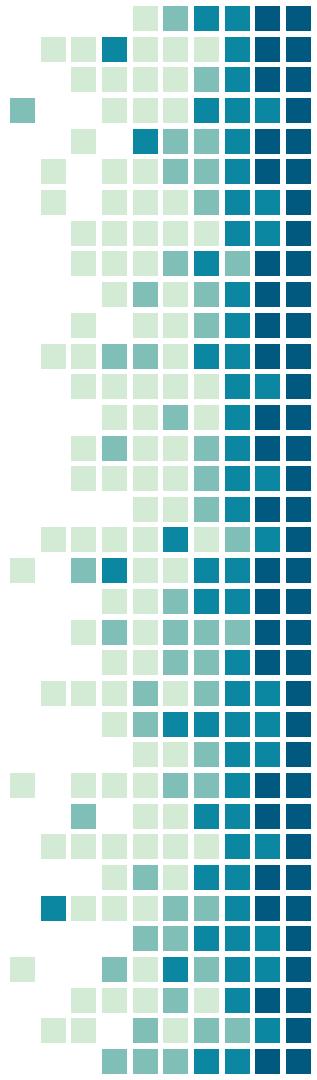
Operación down.



Comprueba si el valor es mayor que cero.

Si es mayor que cero, disminuye el valor del semáforo y continúa el proceso.

Si el valor es cero, el proceso es dormido sin completar la operación down por el momento.

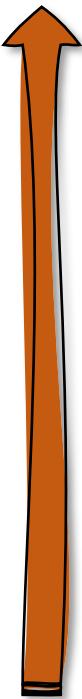


Operación up.

Incrementa el valor del semáforo.

Si uno o más procesos estaban inactivos en ese semáforo, sin poder completar una operación down anterior, el sistema selecciona uno al alzar.

Al igual que down, es una operación indivisible.



Elementos de un semáforo.



El valor actual del semáforo.



El identificador del último proceso que operó con el semáforo



Cantidad de procesos en espera que el valor del mismo sea mayor.

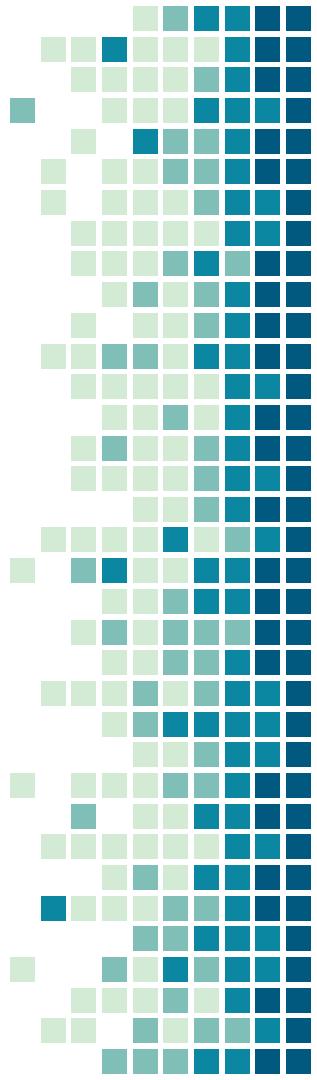


Cantidad de procesos en espera que el valor del mismo sea cero.



Colas de proceso que están asociados a dicho semáforo.

¿Habrá algún problema con los semáforos?



Referencias

- Tanenbaum, A. S. (2022).
Sistemas operativos modernos. Pearson Educación.

- Stallings, W. (2003).
Sistemas operativos. Martin Iturbide

- King, S. T., Dunlap, G. W., & Chen, P. M. (2003, June).
Operating System Support for Virtual Machines. In USENIX Annual Technical Conference, General Track (pp. 71-84).

¿Preguntas?

Realizado por: Jason Leitón Jiménez.

Tecnológico de Costa Rica

Ingeniería en Computadores

2023

TEC