Sistemas Operativos y Distribuidos

Iren Lorenzo Fonseca
iren.fonseca@.ua.es

TEMA 1. Sistemas Operativos.

Gestión de procesos

Tema 1.1 Gestión de procesos

Contenidos

Introducción
Definiciones

Conceptos fundamentales
Procesos

Implementación de procesos
Operaciones y cambios de contexto

Hilos (Threads)
Conceptos

Planificador CPU
Algoritmos

Introducción

Definiciones básicas

Proceso ≠ programa

Proceso: *Programa en ejecución*

Servicios del SO

- **Ejecución concurrente**
- **Sincronización de procesos**
- **Comunicación entre procesos**

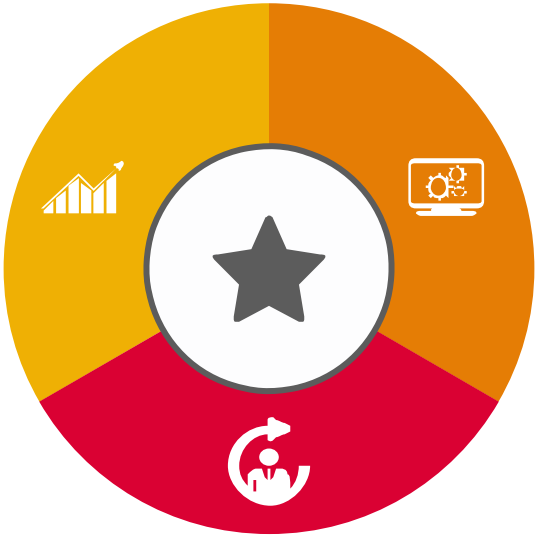
Introducción

Definiciones básicas

- **Ejecución concurrente**

Multiprogramación

Técnica que permite que **múltiples procesos** se mantengan **cargados** en la memoria y **listos** para **ejecutarse**. **Aunque** en los sistemas tradicionales **sólo** hay **un procesador**, el **SO alterna** la **ejecución** de diferentes procesos de forma rápida, lo que da la apariencia de ejecución simultánea



Multiprocesamiento

Capacidad de un sistema para **utilizar más** de **un procesador** físico o **núcleo** de procesador al mismo tiempo para ejecutar varios procesos de manera simultánea. En este caso, hay **varios procesadores** disponibles que pueden **ejecutar** diferentes procesos o partes de un **proceso en paralelo**

Procesamiento Distribuido

Implica la **ejecución** de **procesos** en **múltiples máquinas** o nodos **distribuidos geográficamente** o en una red. En lugar de tener un solo sistema con múltiples procesadores, los procesos se ejecutan en diferentes sistemas interconectados.

Introducción

Definiciones básicas

Proceso ≠ programa

Proceso: *Programa en ejecución*

Servicios del SO

- **Ejecución concurrente**
- **Sincronización de procesos**
- **Comunicación entre procesos**

Conceptos fundamentales

Contenidos

- Relación entre procesos
- Estados de un proceso
- Estructura de un proceso

Relación entre procesos

Tipos

Conceptos

Independientes

01

Los procesos independientes son aquellos que no interactúan ni dependen de otros procesos en el sistema.

Características:

- ✓ No intercambian información con otros procesos.
- ✓ No afectan ni son afectados por el comportamiento o el estado de otros procesos.
- ✓ Son **totalmente autónomos** y pueden ejecutarse en cualquier orden sin afectar el resultado.

Ejemplo:

- ✓ Un proceso que realiza cálculos matemáticos o genera un informe, y no necesita comunicarse con otros procesos durante su ejecución.
- ✓ La ejecución de un editor de texto mientras en otro proceso se ejecuta un reproductor de música. Ambos procesos pueden funcionar independientemente sin necesidad de comunicarse.

Ventajas:

- ✓ Más fácil de gestionar, ya que no requiere coordinación o sincronización con otros procesos.
- ✓ No introduce problemas de interferencia o dependencia entre procesos.

Relación entre procesos

Tipos

Conceptos

Características:

- ✓ Comparten datos o se comunican entre sí.
- ✓ A menudo requieren mecanismos de sincronización.
- ✓ Un proceso puede depender de otro para obtener información o completar una tarea.
- ✓ Los procesos pueden compartir memoria o intercambiar mensajes

Ejemplo:

- ✓ Un navegador web con múltiples pestañas abiertas, donde cada pestaña es un proceso que puede cooperar con el proceso principal del navegador para gestionar el historial o la memoria compartida.

Ventajas:

- ✓ Permiten la comunicación y el trabajo conjunto entre procesos, lo que puede mejorar el rendimiento en ciertas tareas distribuidas.
- ✓ Útil en aplicaciones que requieren compartición de recursos (como memoria o archivos).

Desafíos:

- ✓ Sincronización.
- ✓ Interbloqueo.

Independientes 01

Los procesos independientes son aquellos que no interactúan ni dependen de otros procesos en el sistema.

Cooperativos 02

Los procesos cooperativos son aquellos que interactúan y comparten datos o recursos entre ellos.

Relación entre procesos

Tipos

Conceptos

Características:

- ✓ Los procesos no cooperan entre sí, y deben competir para obtener acceso a los recursos.
- ✓ No comparten información, pero pueden interferir entre sí si compiten por los mismos recursos.
- ✓ Los recursos son asignados por el sistema operativo, que debe decidir cómo distribuirlos de manera justa y eficiente.

Ejemplo:

- ✓ Un proceso que intenta acceder a un dispositivo de impresión mientras otro proceso también intenta utilizarlo. Aquí, el sistema operativo deberá gestionar qué proceso obtiene acceso al dispositivo y cuándo.

Problemas asociados:

- ✓ Interbloqueo (Deadlock)
- ✓ Starvation (Inanición): En sistemas mal diseñados, un proceso puede quedar indefinidamente en espera si otros procesos continúan ocupando los recursos que necesita.

Soluciones comunes:

- ✓ Planificación de CPU.
- ✓ Exclusión mutua.

Independientes 01

Los procesos independientes son aquellos que no interactúan ni dependen de otros procesos en el sistema.

Cooperativos 02

Los procesos cooperativos son aquellos que interactúan y comparten datos o recursos entre ellos.

Competitivos 03

Son aquellos que compiten por recursos limitados del sistema, como la CPU, la memoria, o dispositivos de entrada/salida (E/S).

Estados de un proceso

Estados de un proceso

Ciclo de vida

El **ciclo de vida de un proceso** en un SO describe las diferentes fases por las que pasa un proceso desde su creación hasta su terminación. A lo largo de su vida, un proceso cambia de estado según sus interacciones con la CPU, los recursos del sistema y otros procesos. Estas son las fases principales del ciclo de vida de un proceso:

Estados de un proceso

Ciclo de vida

01 **Creación**

El proceso se encuentra en la fase de creación

El proceso se encuentra en la fase de **creación**. Aquí, el SO está configurando los recursos necesarios para el proceso, como espacio en memoria, tablas de control y otros recursos. Todavía no ha comenzado a ejecutarse.

Acciones:

- Asignación de recursos necesarios (memoria, espacio en disco, etc.).
- Configuración de las estructuras de datos necesarias, como el bloque de control del proceso (**PCB**, Process Control Block).
- Colocación en la **cola de procesos listos**.

Estados de un proceso

Ciclo de vida

El **ciclo de vida de un proceso** en un SO describe las diferentes fases por las que pasa un proceso desde su creación hasta su terminación. A lo largo de su vida, un proceso cambia de estado según sus interacciones con la CPU, los recursos del sistema y otros procesos. Estas son las fases principales del ciclo de vida de un proceso:

Estados de un proceso

Ciclo de vida

02

Listo

Preparado para ejecutarse

En este estado, el proceso está **preparado para ejecutarse** y está esperando ser asignado a la CPU. Todos los recursos necesarios están disponibles, excepto la CPU.

Acciones:

- El proceso espera que el **planificador** del SO le asigne la CPU.
- Puede haber múltiples procesos en este estado, por lo que el planificador utiliza un **algoritmo de planificación** para determinar qué proceso recibe la CPU.
- **Cola de procesos listos**: Los procesos en estado de "listo" se encuentran en la cola de procesos listos, esperando que la CPU esté libre.

Estados de un proceso

Ciclo de vida

El **ciclo de vida de un proceso** en un SO describe las diferentes fases por las que pasa un proceso desde su creación hasta su terminación. A lo largo de su vida, un proceso cambia de estado según sus interacciones con la CPU, los recursos del sistema y otros procesos. Estas son las fases principales del ciclo de vida de un proceso:

Estados de un proceso

Ciclo de vida

03 **Ejecución**

Pasa a este estado cuando el sistema operativo le asigna tiempo en la **CPU**

El proceso pasa a este estado cuando el SO le asigna tiempo en la **CPU**. Aquí, el proceso está ejecutando sus instrucciones.

Acciones:

- La CPU está ejecutando activamente las instrucciones del proceso.
- El proceso puede ejecutar llamadas al sistema, hacer cálculos o manejar interrupciones.

Transiciones:

- **Voluntaria**: Si el proceso necesita realizar una operación de E/S o espera otro recurso, puede ceder voluntariamente la CPU y pasar a un estado de espera (bloqueado).
- **Involuntaria**: El proceso puede ser interrumpido por el SO si su cuanto de tiempo se ha agotado o puede ser suspendido si llega un proceso de mayor prioridad.

Estados de un proceso

Ciclo de vida

El **ciclo de vida de un proceso** en un SO describe las diferentes fases por las que pasa un proceso desde su creación hasta su terminación. A lo largo de su vida, un proceso cambia de estado según sus interacciones con la CPU, los recursos del sistema y otros procesos. Estas son las fases principales del ciclo de vida de un proceso:

Estados de un proceso

Ciclo de vida

04 **Bloqueado**

No puede continuar su ejecución porque está esperando un recurso externo

Un proceso entra en estado de **bloqueo** cuando no puede continuar su ejecución porque está esperando un **recurso externo** (como la finalización de una operación de entrada/salida o la disponibilidad de otro recurso).

Acciones:

- El proceso permanece en este estado hasta que la condición que espera se resuelve, por ejemplo, la finalización de una operación de E/S o la recepción de un mensaje.

Cola de bloqueados: Los procesos en este estado están en una cola de procesos bloqueados, esperando que se complete la operación que los bloquea.

Ejemplo: Un proceso que está esperando la entrada del usuario, como la lectura de un archivo desde un disco duro, se encuentra en este estado

Estados de un proceso

Ciclo de vida

El **ciclo de vida de un proceso** en un SO describe las diferentes fases por las que pasa un proceso desde su creación hasta su terminación. A lo largo de su vida, un proceso cambia de estado según sus interacciones con la CPU, los recursos del sistema y otros procesos. Estas son las fases principales del ciclo de vida de un proceso:

Estados de un proceso

Ciclo de vida

05 **Terminado**

El proceso ha finalizado su ejecución

El proceso ha finalizado su ejecución, ya sea porque ha completado todas sus tareas o porque ha ocurrido un error. El SO libera los recursos que el proceso estaba utilizando y lo elimina de la tabla de procesos

Acciones:

- El sistema libera los recursos asignados al proceso, como la memoria, descriptores de archivos y otros.
- El proceso deja de estar en la tabla de procesos

Causas de la terminación:

- Normal: El proceso ha completado su ejecución de manera exitosa.
- Forzada: El proceso ha sido terminado por el sistema o por otro proceso (a través de una señal o interrupción).
- Error: El proceso ha encontrado un error fatal (como un intento de acceso a memoria no permitida).

Estados de un proceso

Ciclo de vida

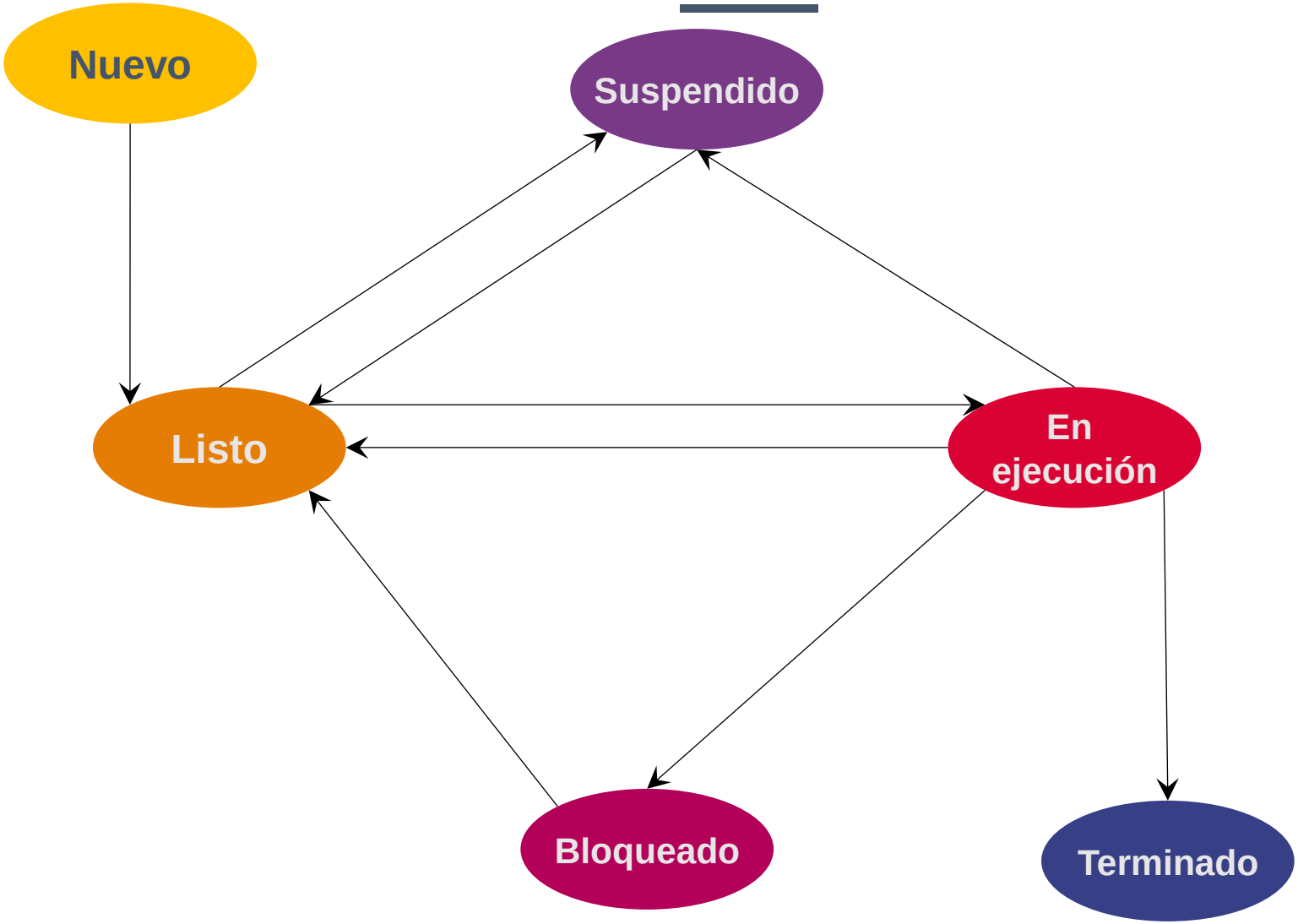
El **ciclo de vida de un proceso** en un SO describe las diferentes fases por las que pasa un proceso desde su creación hasta su terminación. A lo largo de su vida, un proceso cambia de estado según sus interacciones con la CPU, los recursos del sistema y otros procesos. Estas son las fases principales del ciclo de vida de un proceso:

Estados de un proceso

Transiciones

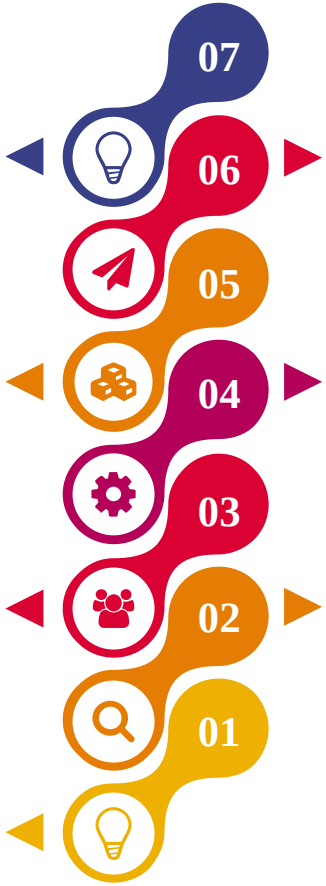
Estados de un proceso

Transiciones



Estados de un proceso

Ejemplo práctico



Terminado
Una vez completado, el proceso termina, y el sistema operativo libera todos los recursos asociados

En Ejecución
El proceso vuelve a obtener la CPU y completa su ejecución

Listo
Cuando el archivo está disponible, el proceso pasa de nuevo al estado listo para continuar

Bloqueado
El proceso necesita leer un archivo, por lo que queda bloqueado esperando a que el archivo esté disponible

En Ejecución
El proceso obtiene la CPU y empieza a ejecutarse. Realiza cálculos y acciones

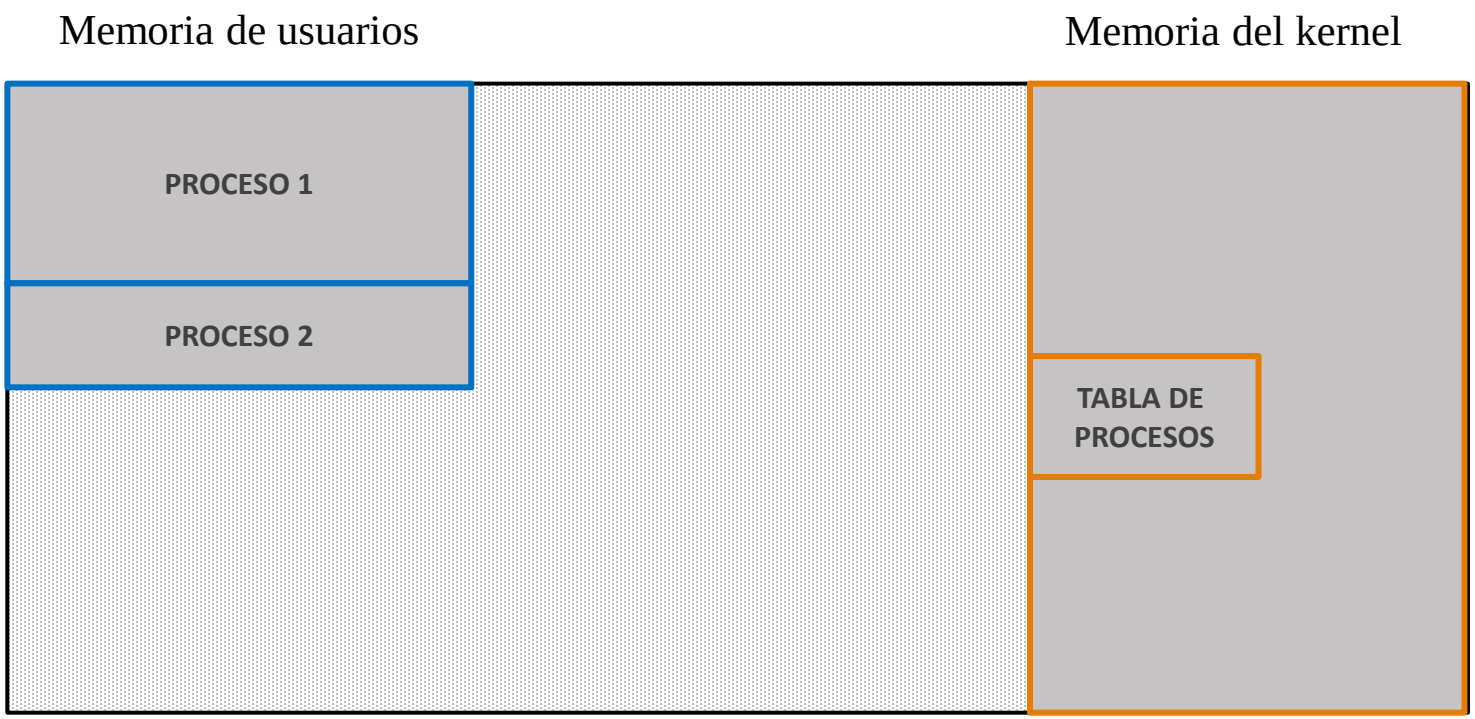
Listo
El proceso está listo para ejecutarse, pero tiene que esperar su turno en la cola de procesos listos

Creación
Imagina que ejecutas un programa desde la línea de comandos. El sistema operativo crea un proceso para este programa

Estructura de un proceso

Estructura de un proceso

Información de procesos



Estructura de un proceso

Tabla de procesos

TABLA DE PROCESOS

Es una estructura de datos que almacena información sobre todos los procesos que están siendo gestionados por el sistema operativo en un momento dado

Estructura de un proceso

Tabla de procesos

Es una estructura de datos que almacena información sobre todos los procesos que están siendo gestionados por el sistema operativo en un momento dado

PCB (PROCESS CONTROL BLOCK)	INFO ADICIONAL DE PROCESOS
PCB (PROCESS CONTROL BLOCK)	INFO ADICIONAL DE PROCESOS
PCB (PROCESS CONTROL BLOCK)	INFO ADICIONAL DE PROCESOS
PCB (PROCESS CONTROL BLOCK)	INFO ADICIONAL DE PROCESOS
PCB (PROCESS CONTROL BLOCK)	INFO ADICIONAL DE PROCESOS

Estática: sistemas antiguos o simples, como algunos sistemas embebidos
Dinámica: stemas operativos modernos como Linux, Windows o Unix, la tabla de procesos tiende a ser dinámica, permitiendo la creación y eliminación de procesos de manera eficiente, dependiendo de las demandas del sistema

Estructura de un proceso

Bloque de Control de Proceso

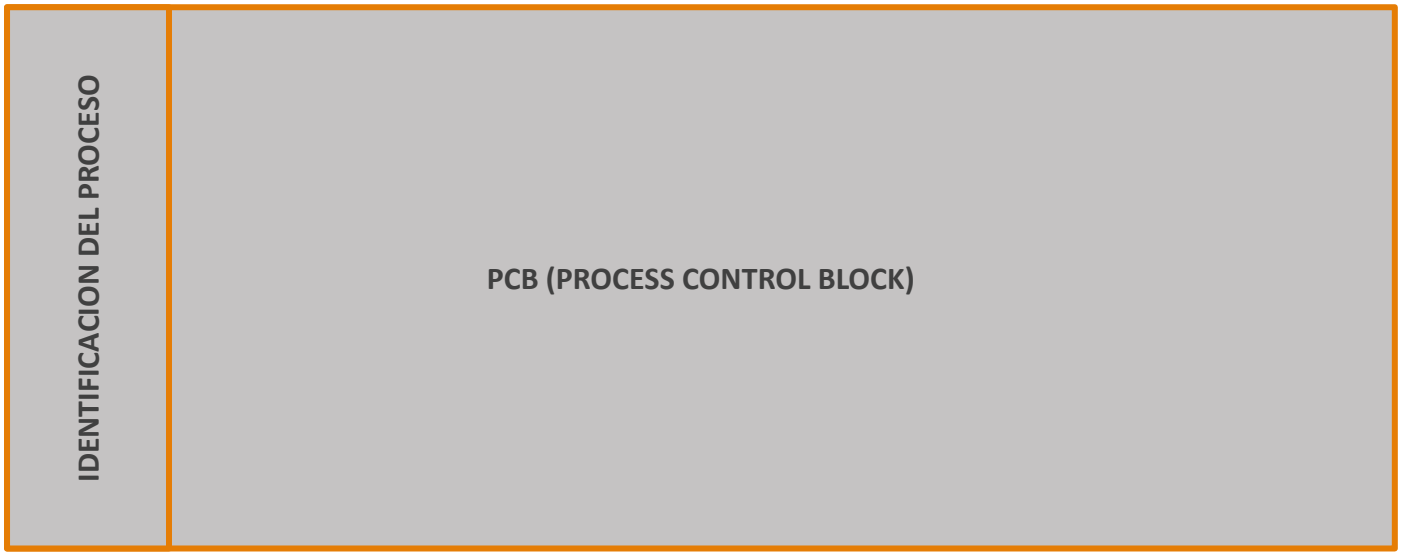
PCB (PROCESS CONTROL BLOCK)

*El **bloque de control de procesos (PCB)** es la estructura de datos del sistema que mantiene toda la información sobre **un** proceso.*

- El PCB de cada proceso se crea cuando se crea un proceso y se destruye cuando el proceso termina.

Estructura de un proceso

Campos del PCB

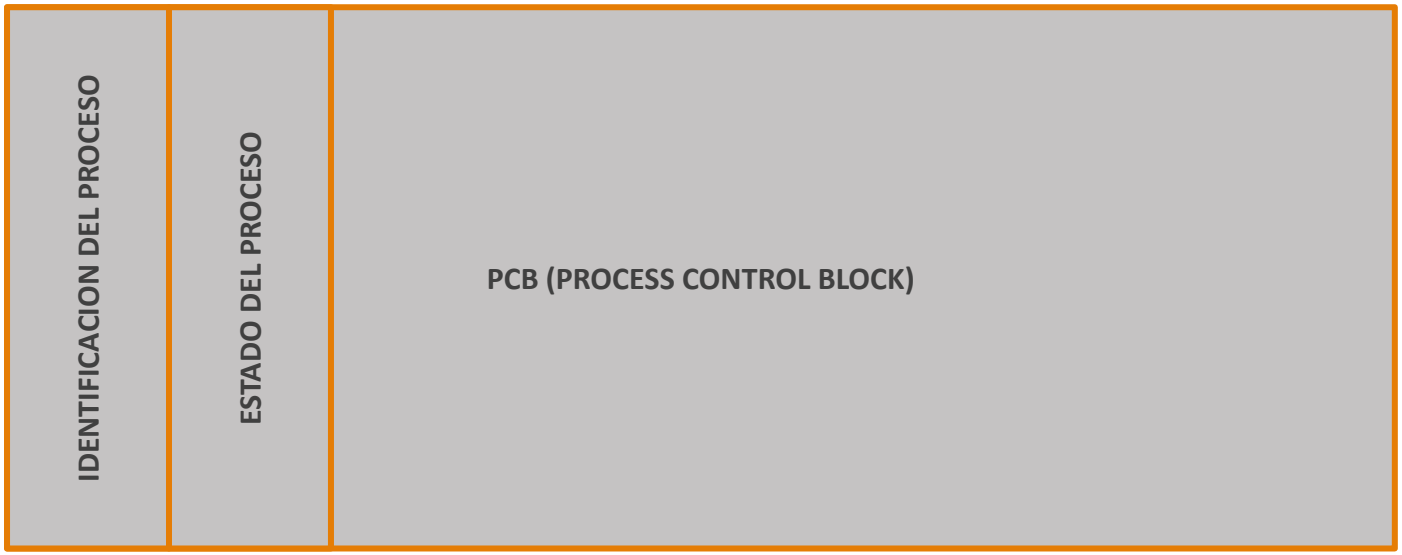


PID (Process ID): Un identificador único que distingue a cada proceso en el sistema.

PPID (Parent Process ID): Identificador del proceso padre que generó este proceso (si existe).

Estructura de un proceso

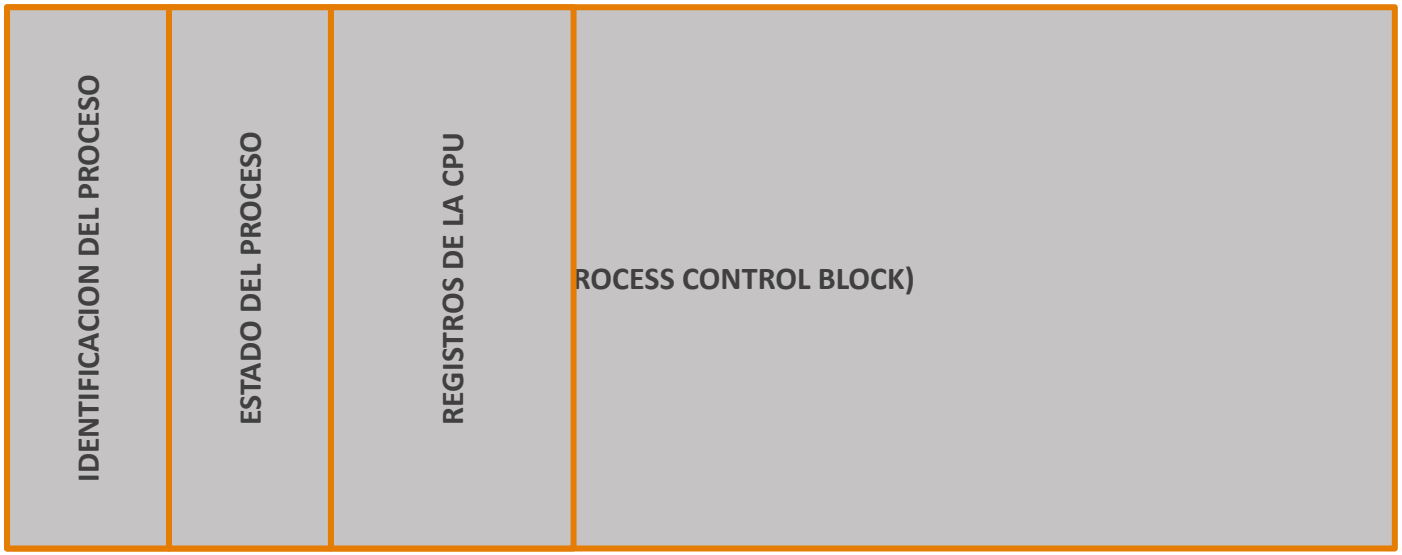
Campos del PCB



Estado del Proceso: El estado actual en el que se encuentra el proceso (nuevo, listo, en ejecución, en espera, terminado, suspendido, etc.).

Estructura de un proceso

Campos del PCB



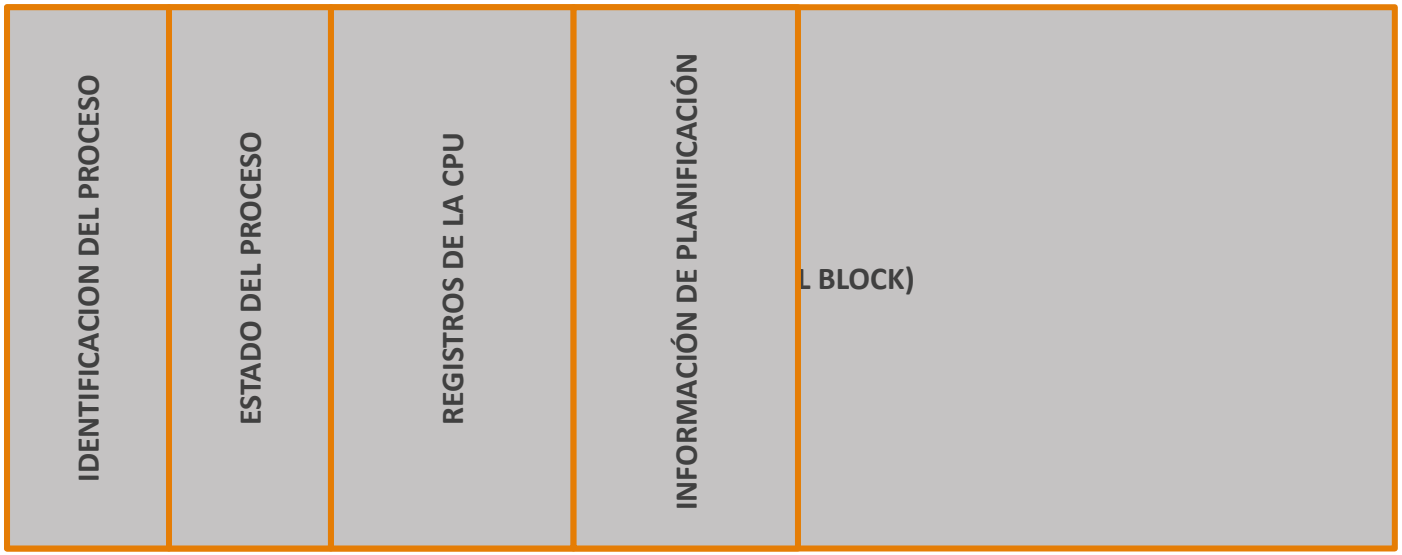
Contador de Programa (PC): Dirección de la próxima instrucción a ejecutar cuando el proceso vuelva a ser ejecutado.

Registros de la CPU: El valor de los registros generales (como acumulador, punteros, contadores, etc.) en el momento en que se interrumpió el proceso.

…

Estructura de un proceso

Bloque de Control de Proceso



Prioridad: Nivel de prioridad del proceso en la planificación del CPU.

Cola de Planificación: Puntero a la cola de procesos listos o en espera, dependiendo de la política de planificación utilizada.

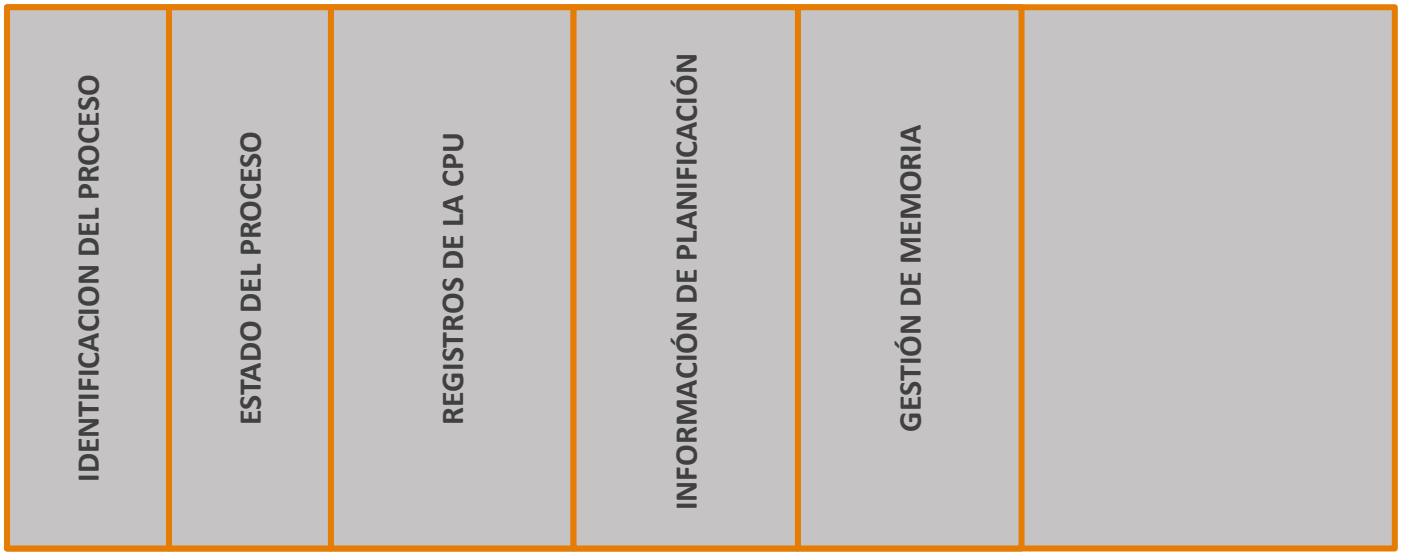
Tiempo de CPU usado: Tiempo de CPU que el proceso ha consumido hasta ahora.

Tiempo de espera: Tiempo que el proceso ha pasado en espera de recursos.

…

Estructura de un proceso

Campos del PCB



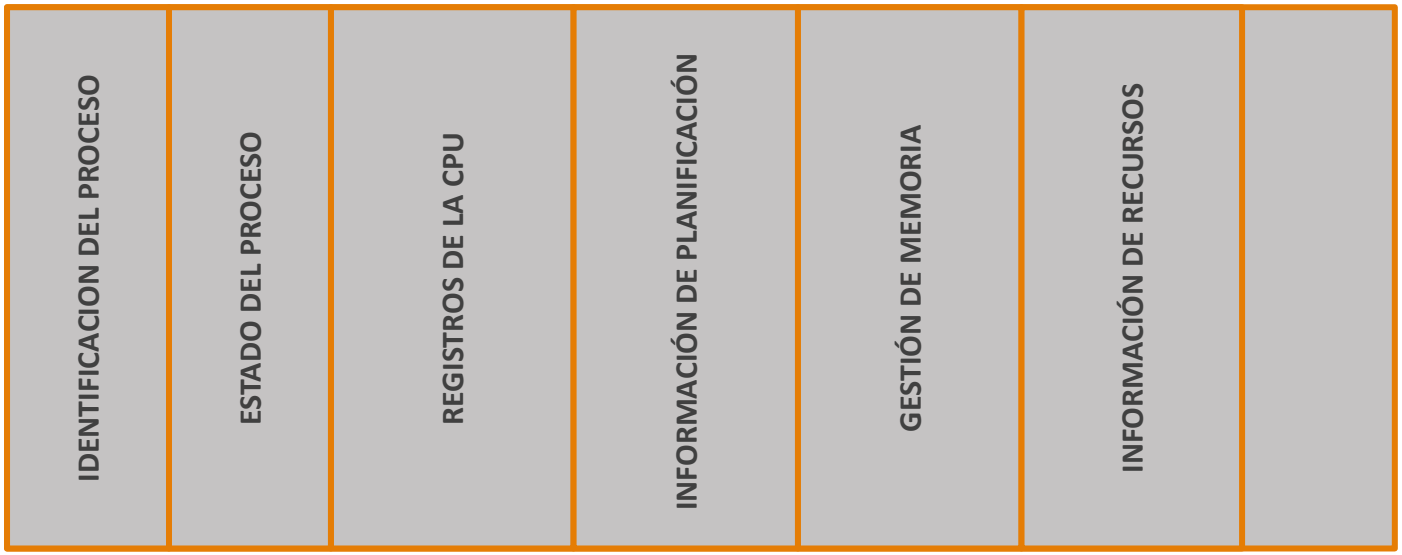
Dirección base y límite: Indica el rango de memoria asignado al proceso (segmento base y su tamaño o límite).

Punteros de la tabla de páginas: En sistemas con memoria virtual, se almacenan punteros a las tablas de páginas o segmentos, que contienen las direcciones de la memoria virtual.

…

Estructura de un proceso

Campos del PCB



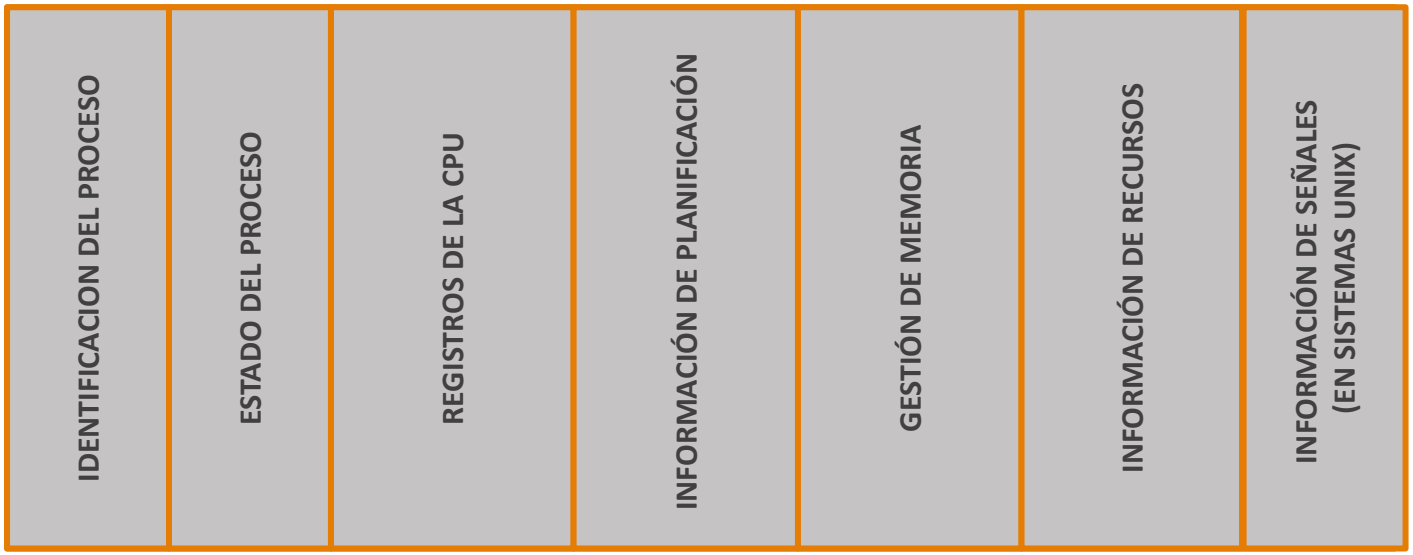
Archivos abiertos: Lista de archivos abiertos por el proceso.

Dispositivos asignados: Información sobre los dispositivos de entrada/salida asignados al proceso (por ejemplo, terminales, discos, etc.).

…

Estructura de un proceso

Campos del PCB

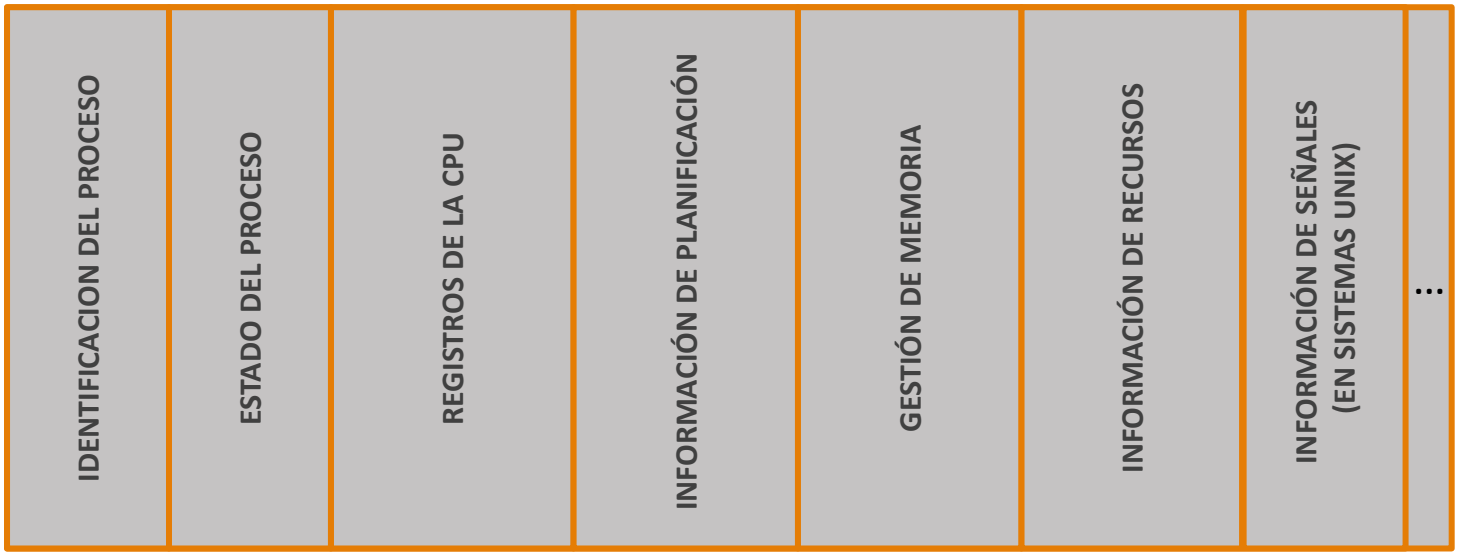


Señal pendiente: Indica si hay señales esperando ser entregadas al proceso.

…

Estructura de un proceso

Campos del PCB



Estructura de un proceso

Conceptos

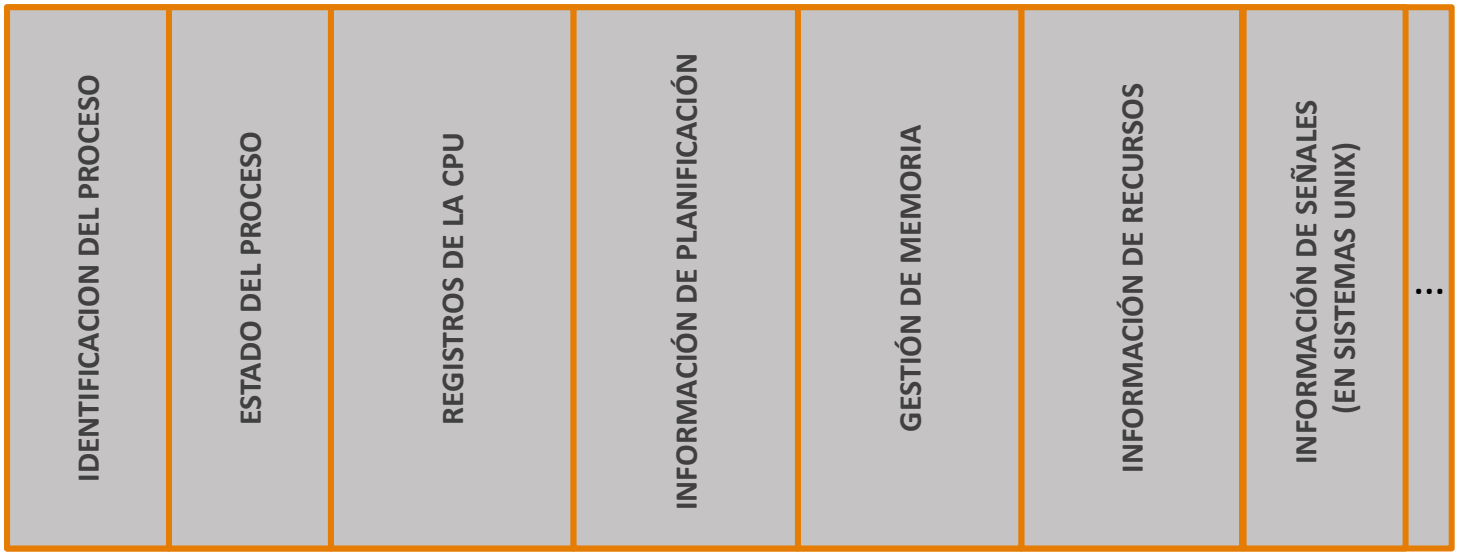
Ejemplo del PCB

PCB (PROCESS CONTROL BLOCK)

Campo	Valor Ejemplo	Descripción
PID (Process ID)	1023	Identificador único del proceso.
PPID (Parent PID)	1001	Identificador del proceso padre (el proceso que lo creó).
UID (User ID)	1000	ID del usuario propietario del proceso.
Estado del Proceso	Listo (Ready)	Estado actual del proceso (Nuevo, Listo, En Ejecución, Bloqueado).
Prioridad	5	Prioridad del proceso (cuanto menor, mayor es la prioridad).
Contador de Programa (PC)	0x0040F123	Dirección de la próxima instrucción a ejecutar.
Registros de CPU	EAX: 0x00000001, EBX: 0x00000005	Estado actual de los registros de la CPU.
Segmento de Código (Code)	0x00100000 - 0x0010FFFF	Direcciones de memoria del segmento de código.
Segmento de Datos (Data)	0x00200000 - 0x0020FFFF	Direcciones de memoria del segmento de datos.
Pila (Stack)	SP: 0x0030FF00	Puntero de pila, dirección actual en la pila del proceso.
Heap	0x00400000 - 0x0040AFFF	Dirección del segmento de memoria heap para asignaciones dinámicas.
Tabla de Páginas	Dirección de tabla de páginas: 0xF0002000	Puntero a la tabla de páginas en memoria virtual.
Archivos Abiertos	/dev/tty1, /home/usuario/documento.txt	Lista de archivos abiertos por el proceso.
Tiempo de CPU usado	30 ms	Tiempo total de CPU consumido hasta ahora.
Dispositivos de E/S	Disco	Dispositivos de entrada/salida asignados al proceso.
Máscara de Señales	0x00000002	Señales bloqueadas o pendientes (en sistemas UNIX).
Cola de Planificación	Listo (Ready Queue)	Cola donde se encuentra el proceso para ser planificado.

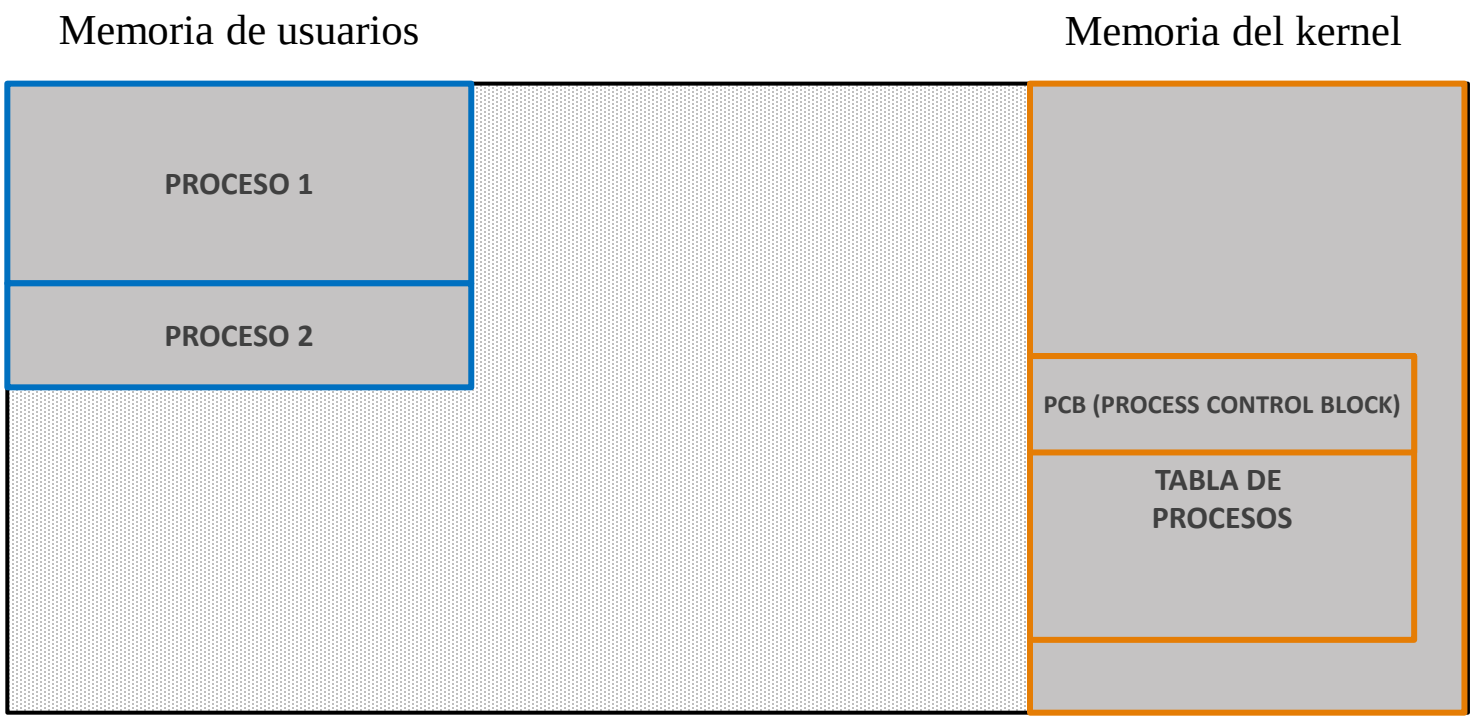
Estructura de un proceso

Campos del PCB



Estructura de un proceso

Información de procesos



Estructura de un proceso

Información de procesos

Estructura de un proceso

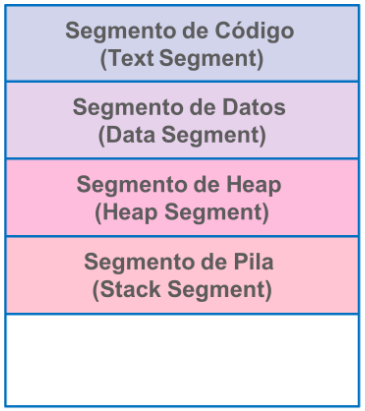
Información de procesos

PROCESO 1

Conceptos de los SO

Memoria

Espacio de direcciones



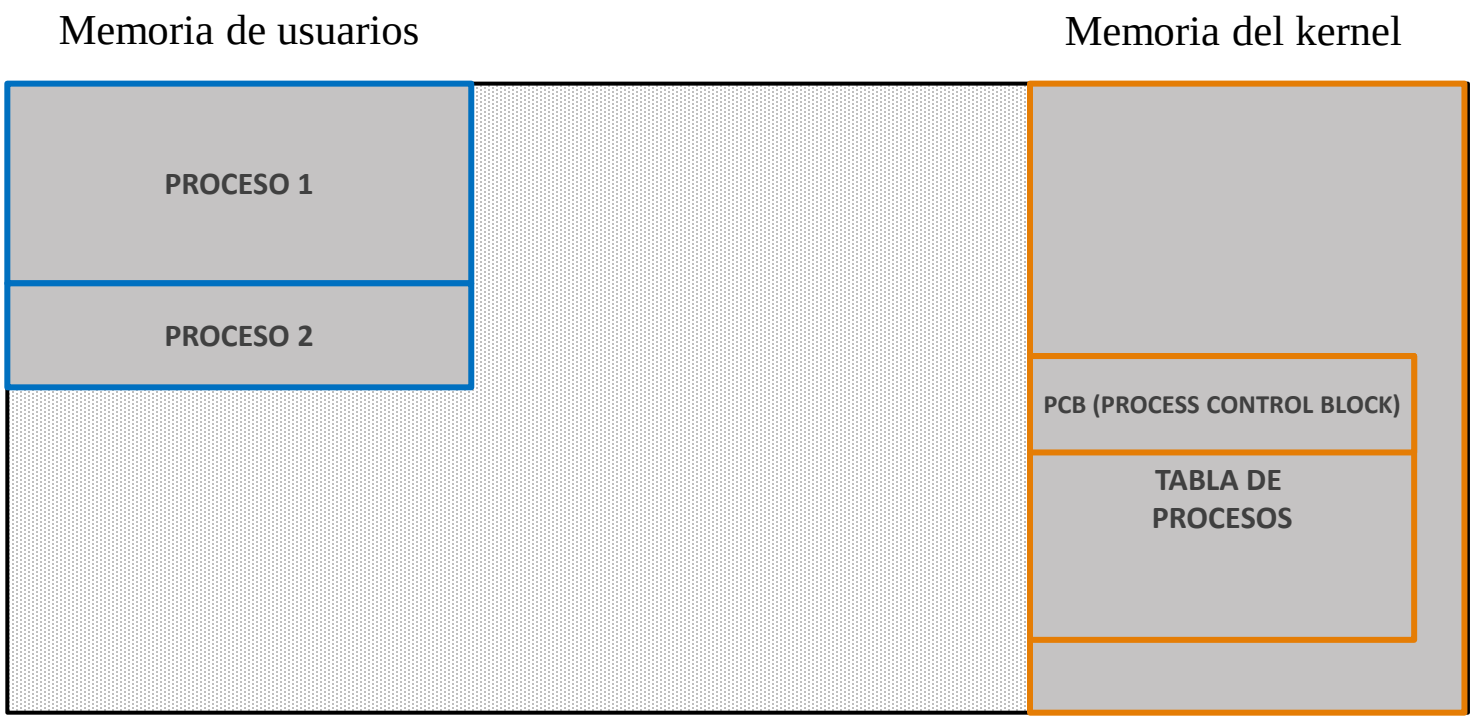
- Contiene las **variables locales** y las **direcciones** de **retorno** de las **funciones**.
- La pila (stack) es utilizada para almacenar información sobre las funciones activas, como los parámetros de función, las direcciones de retorno y las variables locales. La pila crece y decrece conforme se llaman y se devuelven las funciones.

Estructura de los SO

Conceptos fundamentales

Estructura de un proceso

Información de procesos



Implementación de Procesos

Contenidos

- Operaciones con procesos
- Cambio de contexto

Operaciones con procesos

Creación de procesos Sistemas UNIX

Llamada al sistema para crear un proceso (fork()).

1. Asignación de un PID único para el nuevo proceso.
2. Creación del PCB que almacena información del proceso.
3. Asignación de espacio de memoria para el proceso.
4. Copia del contexto del proceso padre.
5. Inicialización del proceso configurando el contador de programa y la pila.
6. Colocación en la cola de planificación para ser programado.
7. Transición al estado listo para comenzar su ejecución.

Operaciones con procesos

Terminación de procesos Sistemas UNIX

Terminación de procesos: *exit*, *kill*

Terminación de procesos: exit, kill

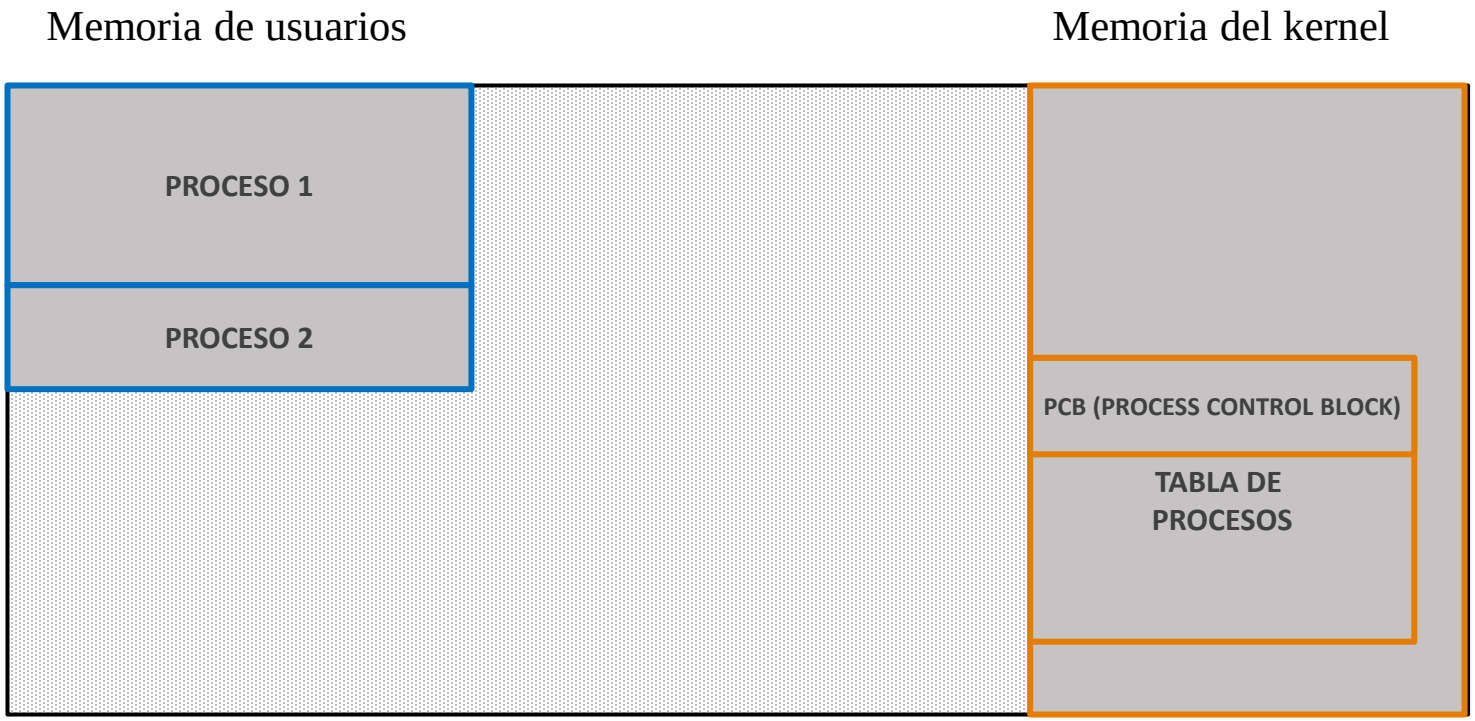
1. **Notificación:** El SO puede notificar al proceso que va a ser terminado (en el caso de terminación controlada) o puede finalizarlo de inmediato (en el caso de terminación forzada).
2. **Liberación de Recursos**: El sistema operativo libera todos los recursos asignados al proceso .
3. **Actualización del PCB**: Para reflejar su estado como terminado. La información de estado, los recursos liberados y otros datos relevantes se registran.
4. **Notificación a Procesos Padres**
5. **Eliminación del PCB**: Finalmente, el PCB del proceso se elimina de la tabla de procesos del sistema operativo, liberando espacio para otros procesos.

Cambio de contexto

Cambio de contexto

Contexto

El **contexto** de un proceso se refiere a toda la información necesaria que el sistema operativo necesita para gestionar y ejecutar un proceso. Este contexto incluye tanto los datos que describen el estado del proceso en un momento dado como la información necesaria para reanudar su ejecución en el futuro



Cambio de contexto

Contexto

Datos del PCB y del espacio de direcciones del proceso

PROCESO 1

- Segmento de código
- Segmento de datos
- Heap
- Pila (Stack)

PCB (PROCESS CONTROL BLOCK)

- Identificación del proceso (PID
- Estado del proceso
- Contador de programa
- Registros de CPU
- Prioridad
- Punteros de memoria
- …

Cambio de contexto

Cambio de Contexto

Cambio de contexto de un proceso

Retirar de la CPU el proceso en ejecución y asignarla a un proceso en estado de listo.

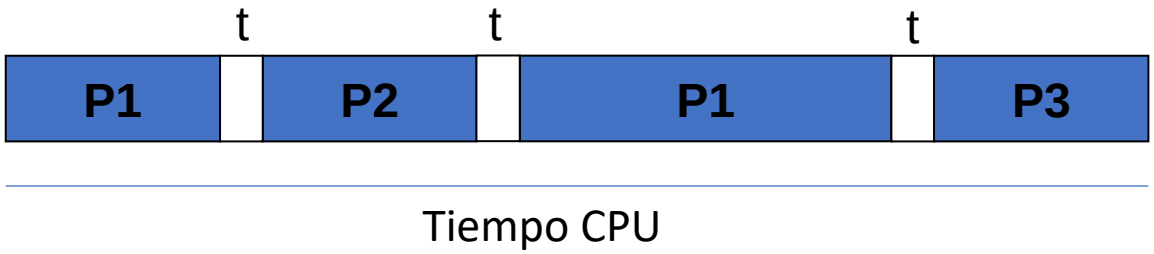
- Acciones necesarias
  - Salvar el contenido del proceso en ejecución a su PCB
  - Restaurar el contexto del nuevo proceso desde su PCB

Cambio de contexto

Costo

Costos del Cambio de Contexto

Tiempo: El cambio de contexto no es gratuito. Implica overhead debido a la necesidad de guardar y restaurar el contexto, lo que puede consumir tiempo de CPU. El tiempo perdido en cambios de contexto excesivos puede impactar negativamente el rendimiento del sistema.

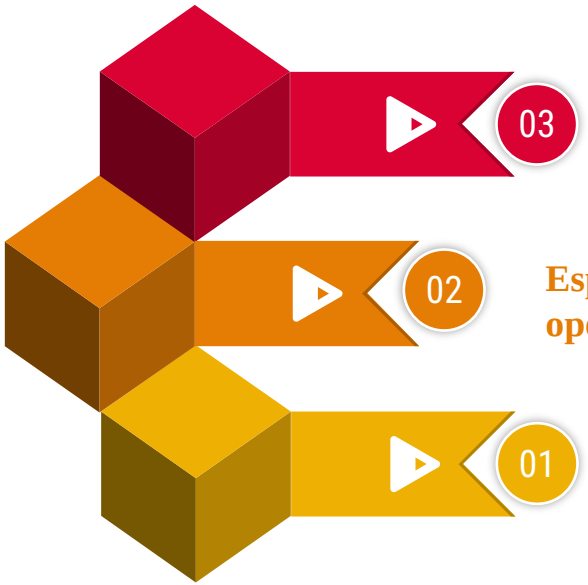


$$\text{Utilización} = \frac{T(P1) + T(P2) + T(P3)}{T(P1) + T(P2) + T(P3) + 3t}$$

Cambio de contexto

Motivos

Motivos que provocan el cambio de contexto



Existencia de una interrupción

Espera de un proceso al realizar una operación de E/S

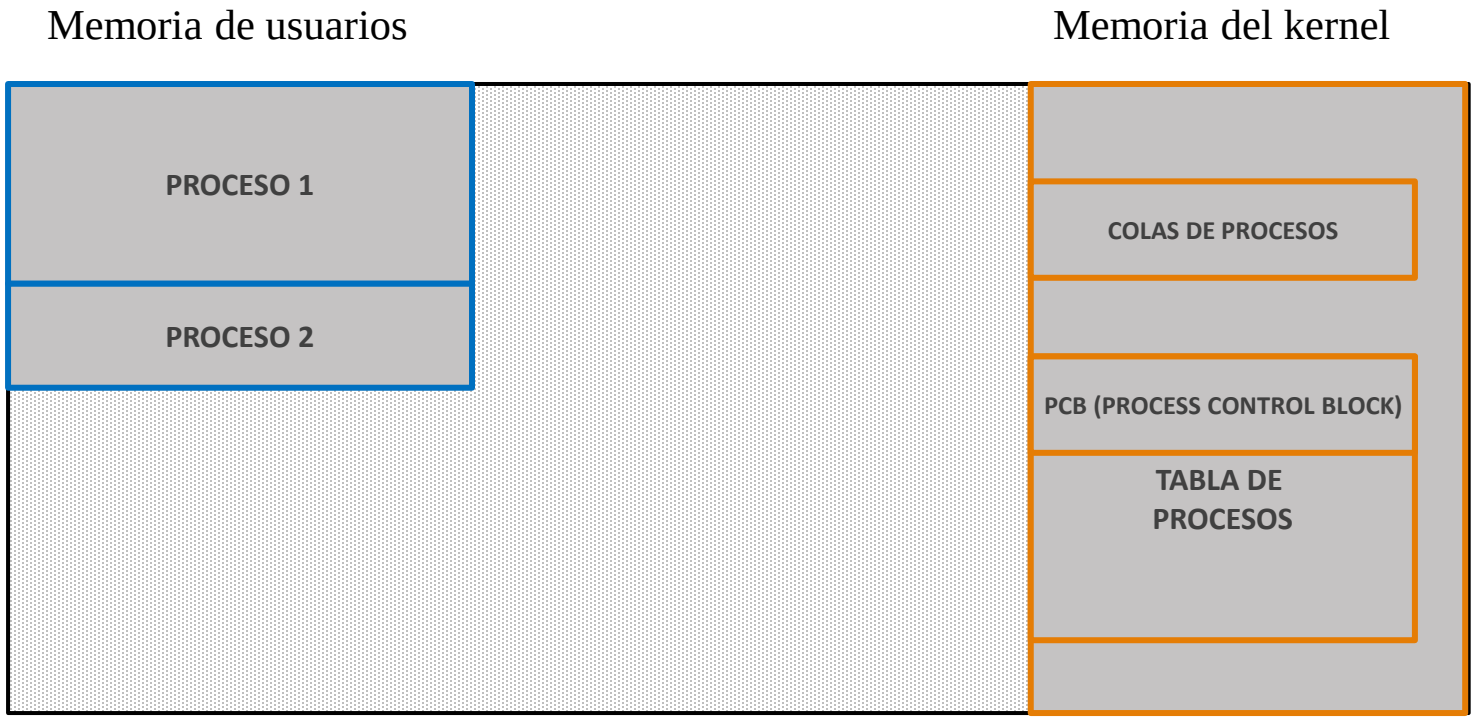
Finalización normal de un proceso en ejecución

Implementación de procesos

Colas de procesos

Colas de procesos: *Tipo Abstracto de Datos que mantiene los PCB de los procesos en una estructura dinámica de lista.*

- Se mantiene una cola de procesos por cada estado de los procesos.

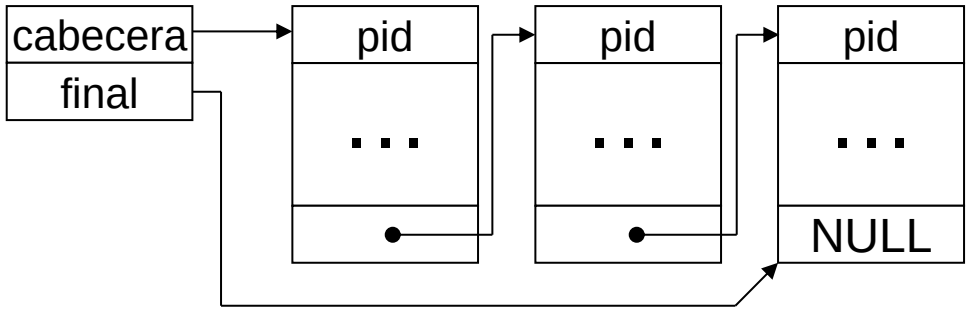


Implementación de procesos

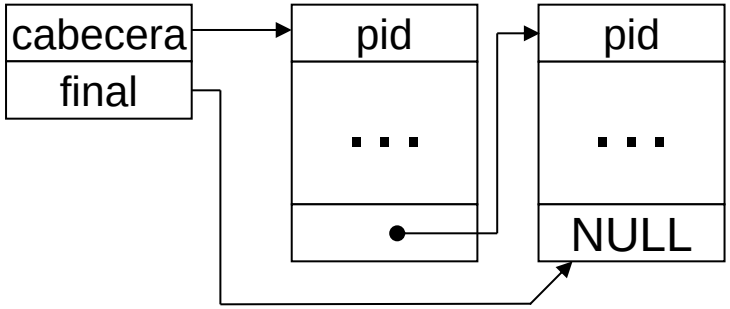
Colas de procesos

COLAS DE PROCESOS

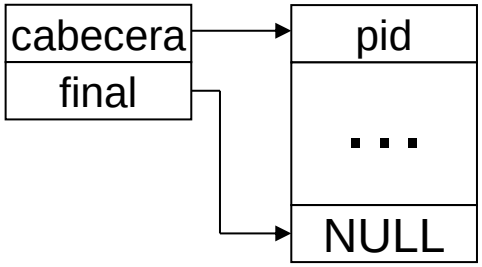
Cola de procesos listos



Cola de procesos bloqueados



Cola de procesos en ejecución

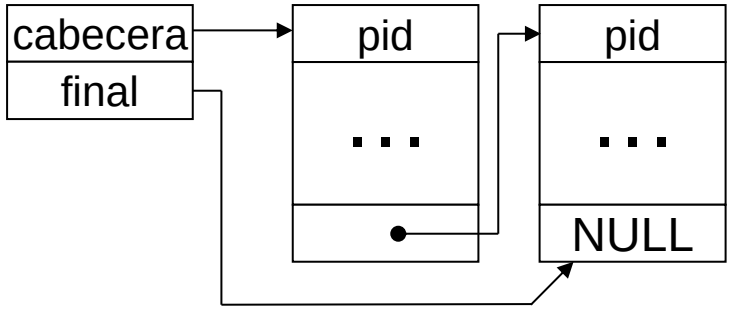


Implementación de procesos

Colas de procesos

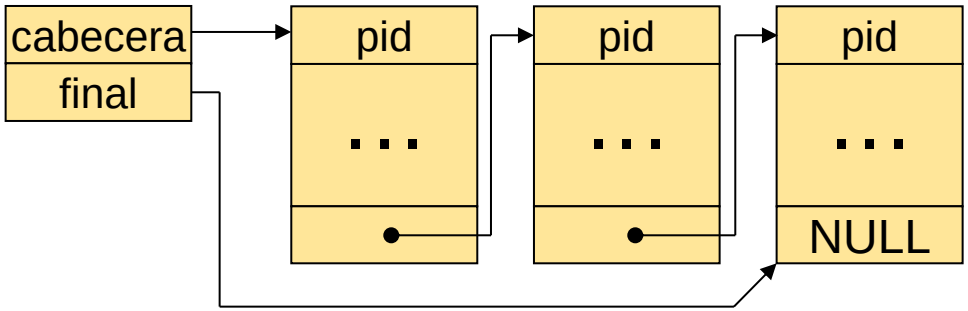
COLAS DE PROCESOS

Cola de procesos bloqueados

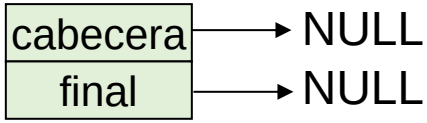


Lista de espera por recursos

Cola de procesos Impresora



Cola de procesos BD

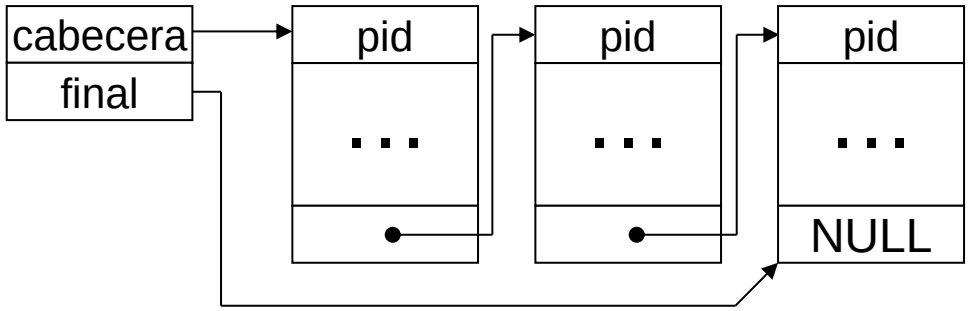


Implementación de procesos

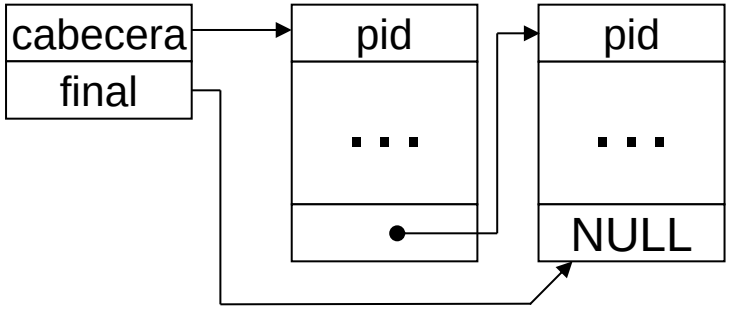
Colas de procesos

COLAS DE PROCESOS

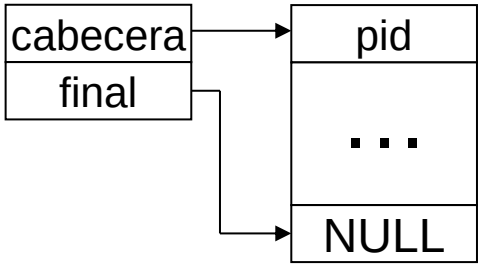
Cola de procesos listos



Cola de procesos bloqueados



Cola de procesos en ejecución

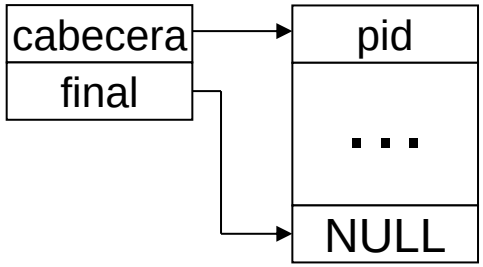


Implementación de procesos

Colas de procesos

COLAS DE PROCESOS

Cola de procesos en ejecución



Sistemas con varios procesadores

Cola de Listos Global:

•Todos los procesadores/núcleos comparten una única **cola global de listos**.

•Los procesadores extraen procesos de esta cola cuando están disponibles para ejecutar.

•Este enfoque es sencillo, pero puede causar cuellos de botella debido a la contención cuando varios procesadores intentan acceder a la misma cola simultáneamente.

Colas de Listos por Procesador/Núcleo:

•Cada procesador o núcleo tiene su propia **cola de listos local.**

•Los procesos se distribuyen entre las diferentes colas, lo que reduce la contención, pero introduce un desafío para equilibrar la carga entre los diferentes procesadores.

•Algunos sistemas usan políticas de **balanceo de carga** (load balancing)

Hilos

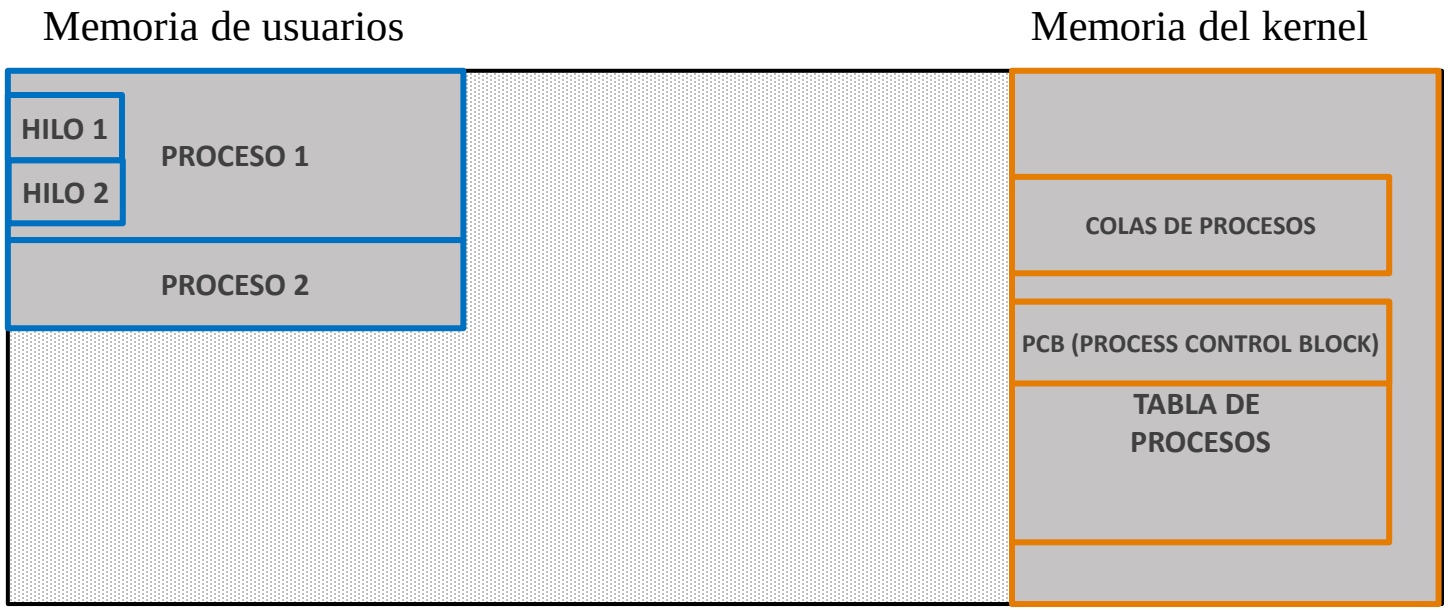
Contenidos

- [x] Introducción
- [x] Hilos en Python

Introducción

Hilos

- ✓ Un hilo es una unidad de ejecución dentro de un proceso.
- ✓ Todos los hilos de un mismo proceso comparten el mismo espacio de direcciones y recursos, pero tienen su propio contador de programa, pila y registros.
- ✓ Esto hace que la comunicación entre hilos dentro de un proceso sea más eficiente y sencilla en comparación con la comunicación entre procesos.



Introducción

Procesos vs Hilos

Características	Proceso	Hilo
Creación	Más costosa en tiempo y recursos.	Más rápida y eficiente.
Administración	Independiente, tiene su propio espacio de memoria.	Comparten espacio de memoria con otros hilos del mismo proceso.
Finalización	Cuando un proceso termina, se liberan todos sus recursos.	Cuando un hilo termina, los recursos compartidos permanecen.
Cambio de contexto	Más costoso debido a la separación de espacios de memoria.	Más eficiente, ya que comparten el espacio de memoria.

Introducción

Tipos de Hilos

Hilos a nivel de usuario

Los hilos a nivel de usuario son gestionados enteramente por una biblioteca en el espacio de usuario, sin intervención del kernel del sistema operativo.

Estos hilos son fáciles de crear y administrar porque no requieren intervención del sistema operativo, lo que hace que la conmutación de hilos sea extremadamente rápida.

Desventajas:

- Si uno de los hilos realiza una operación bloqueante (como una lectura de disco), todo el proceso se bloquea.
- El sistema operativo no sabe de la existencia de estos hilos, por lo que no puede optimizar su ejecución en múltiples CPUs.

Hilos a nivel de kernel

En los hilos a nivel de kernel, el sistema operativo es responsable de la gestión de los hilos. Aquí, el kernel es quien maneja la creación, finalización y cambio de contexto de los hilos

Ventajas:

- Los hilos del mismo proceso pueden ejecutarse en diferentes núcleos de CPU de forma simultánea, lo que mejora el rendimiento en sistemas multiprocesadores.
- Si un hilo se bloquea, otros hilos del mismo proceso pueden seguir ejecutándose.

Desventajas:

- La gestión de hilos a nivel de kernel es más costosa en términos de recursos del sistema, ya que involucra más operaciones y llamadas al sistema.

Hilos en Python

Hilos en Python

Como funciona

- ✓ El multithreading permite la ejecución de **varios hilos** dentro de un solo proceso.
- ✓ Todos los hilos comparten el mismo espacio de memoria, lo que permite una comunicación eficiente entre ellos.
- ✓ Sin embargo, debido al **GIL** (Global Interpreter Lock) en algunas implementaciones de Python, sólo **un hilo puede ejecutar código Python a la vez**, lo que limita el rendimiento en tareas intensivas en CPU.

  - CPython: Sí tiene GIL.
  - Jython: No tiene GIL.
  - IronPython: No tiene GIL.
  - PyPy: Sí tiene GIL (pero tiene proyectos experimentales para eliminarlo).

Hilos en Python

Implementación de python

```
import platform

implementation = platform.python_implementation()

print(f"Estás usando: {implementation}")

if implementation == "CPython":
    print("Esta implementación usa GIL.")
elif implementation == "PyPy":
    print("Esta implementación usa GIL, pero tiene optimizaciones.")
else:
    print("Esta implementación no usa GIL.")
```

Hilos en Python

Paralelismo con GIL

¿Cuándo usar multithreading?

- El multithreading es útil cuando la mayor parte del tiempo de ejecución se dedica a **operaciones de entrada/salida (I/O)**, como leer y escribir archivos, realizar solicitudes de red, o interactuar con bases de datos.
- En estas tareas, el GIL se libera mientras se espera la respuesta, permitiendo que otros hilos continúen ejecutándose.
- Ejemplos de tareas I/O intensivas:
  - Servidores web que manejan múltiples conexiones simultáneas.
  - Programas que descargan múltiples archivos de la red.
  - Aplicaciones que procesan grandes cantidades de datos de entrada/salida (como logs o consultas a bases de datos).

Menor consumo de memoria: Dado que todos los hilos comparten el mismo espacio de memoria, el uso de **memoria** es más eficiente en comparación con múltiples procesos, lo que puede ser importante si se ejecutan muchos hilos al mismo tiempo.

Hilos en Python

Ejemplo con thread

```python
import threading
import time

# Función que ejecutará cada hilo
def worker(number):
    print(f"Hilo {number} empezando")
    time.sleep(2)  # Simula una operación de I/O con un retardo de 2 segundos
    print(f"Hilo {number} terminando")

# Crear hilos
thread1 = threading.Thread(target=worker, args=(1,))
thread2 = threading.Thread(target=worker, args=(2,))

# Iniciar los hilos
thread1.start()
thread2.start()

# Esperar que ambos hilos terminen
thread1.join()
thread2.join()

print("Todos los hilos han terminado")
```

Hilos en Python

Paralelismo con GIL

MULTIPROCESSING

- El **multiprocessing** crea **múltiples procesos** independientes, cada uno con su propio espacio de memoria.
- La gran ventaja es que cada proceso tiene su propio intérprete de Python y no está limitado por el GIL, lo que permite que varios procesos ejecuten código Python simultáneamente en **múltiples núcleos de CPU**

Hilos en Python

Paralelismo con GIL

¿Cuándo usar multiprocessing?

- **Tareas intensivas en CPU**: Si el programa está realizando operaciones que consumen mucha CPU, como cálculos matemáticos complejos, entrenamiento de modelos de IA o procesamiento de grandes volúmenes de datos, **multiprocessing** es generalmente la mejor opción.
- Cada proceso se ejecuta en su propio núcleo de CPU, lo que permite un verdadero **paralelismo**.
- Ejemplos de tareas CPU intensivas:
  - Entrenamiento de redes neuronales profundas.
  - Algoritmos de procesamiento de imágenes o videos.
  - Simulaciones científicas o financieras.

Escalabilidad en sistemas multicore: Dado que cada proceso puede correr en su propio núcleo de CPU, el multiprocessing permite aprovechar completamente los sistemas **multicore**, lo que es clave para tareas computacionales intensivas.

Hilos en Python

Ejemplo con multiprocessing

```python
import multiprocessing
import time

# Función que ejecutará cada proceso
def worker(number):
    print(f"Proceso {number} empezando")
    result = sum(i * i for i in range(10**6))  # Simula una tarea intensiva en CPU
    print(f"Proceso {number} terminando con resultado {result}")

if __name__ == "__main__":
    # Crear procesos
    process1 = multiprocessing.Process(target=worker, args=(1,))
    process2 = multiprocessing.Process(target=worker, args=(2,))

    # Iniciar los procesos
    process1.start()
    process2.start()

    # Esperar que ambos procesos terminen
    process1.join()
    process2.join()

    print("Todos los procesos han terminado")
```

Hilos en Python

Paralelismo con GIL

Hilos

Características	Multithreading	Multiprocessing
Paralelismo	No hay verdadero paralelismo debido al GIL (solo un hilo puede ejecutar código Python a la vez).	Proporciona verdadero paralelismo; cada proceso puede ejecutarse en su propio núcleo.
Tareas adecuadas	Tareas I/O intensivas (operaciones de red, E/S, bases de datos).	Tareas CPU intensivas (cálculos complejos, procesamiento de datos).
Overhead	Menor overhead, ya que los hilos comparten el mismo espacio de memoria.	Mayor overhead debido a la creación de procesos independientes y la duplicación de memoria.
Uso de memoria	Más eficiente en cuanto a memoria, ya que todos los hilos comparten el mismo espacio de memoria.	Cada proceso tiene su propio espacio de memoria, lo que consume más recursos.
Facilidad de uso	Más fácil compartir datos entre hilos (porque comparten memoria).	Compartir datos entre procesos es más complejo y puede requerir mecanismos de IPC (memoria compartida, colas, pipes).
Escalabilidad	No escala bien en sistemas multicore debido al GIL.	Escala muy bien en sistemas multicore, aprovechando todos los núcleos de la CPU.

Planificador CPU

Contenidos

- [x] Introducción
- [x] Algoritmos de planificación CPU

Introducción

Planificador CPU

Decidir qué proceso en el sistema en estado **listo** debe ser asignado a la **CPU** para su ejecución

Criterios de planificación

Rendimiento

Cantidad de trabajo que un sistema puede realizar en un tiempo determinado.

Rendimiento = nº procesos terminados/t

Tiempo de Espera

Tiempo total que un proceso pasa en la cola de listos, esperando a ser ejecutado. No incluye el tiempo de ejecución

Utilización de la CPU

Proporción del tiempo en que la CPU está activa y ejecutando procesos en comparación con el tiempo total disponible.

Utilización = $t_{CPU\_ocupada}/t_{CPU\_TOTAL}$

Tiempo de Retorno

Tiempo total que transcurre desde que se envía un proceso hasta que se completa. Incluye el tiempo de espera, el tiempo de ejecución y el tiempo de I/O

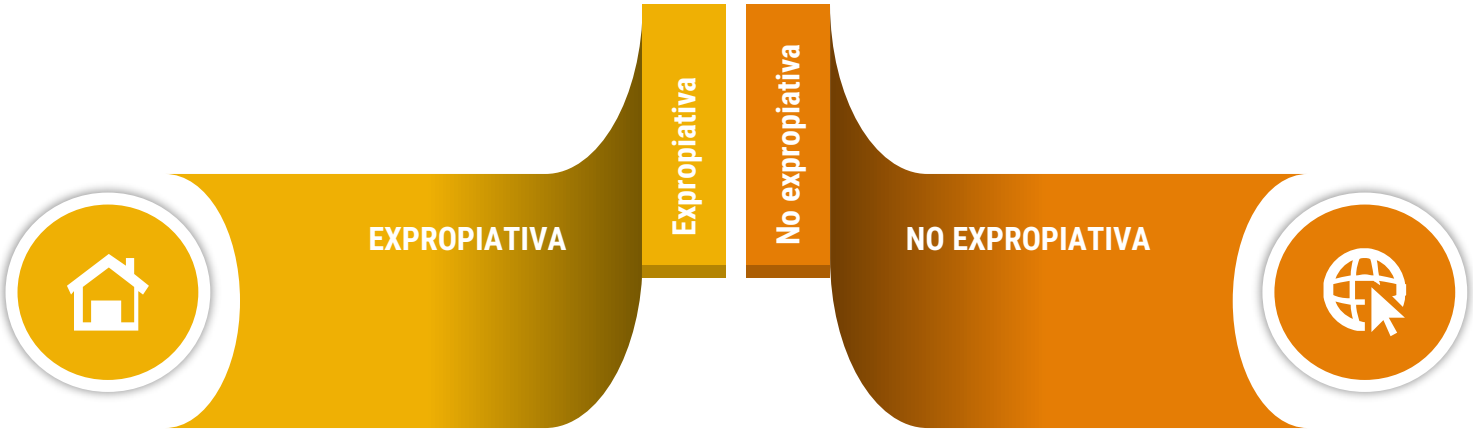
Tiempo de Respuesta

Tiempo transcurrido desde que se inicia un proceso hasta que el sistema comienza a devolver la salida. Es un criterio crucial para sistemas interactivos.

Algoritmos de planificación

Algoritmos de planificación

Tipo de planificación



Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

Enfoque donde una vez que un proceso ha comenzado a ejecutar en la CPU, debe completarse o entrar en el estado de espera antes de que se le pueda quitar la CPU.
Esto significa que un proceso no puede ser interrumpido arbitrariamente por otros procesos.

Ventajas

- **Simplicidad**: La planificación no expropiativa es más fácil de implementar, ya que no se requiere un sistema de prioridades complejo ni la gestión de interrupciones.
- **Menor Sobrecarga**: Al no haber cambios de contexto frecuentes, hay menos sobrecarga en la CPU, lo que puede llevar a un mejor rendimiento en ciertas aplicaciones.

Desventajas

- **Ineficiencia**: Puede ser ineficiente en entornos donde los procesos de mayor prioridad necesitan atención inmediata, ya que un proceso de baja prioridad puede monopolizar la CPU.
- **Latencia**: Puede aumentar el tiempo de espera de los procesos críticos, lo que afecta la capacidad de respuesta del sistema.

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

01

FIFO (First In First Out)

Los procesos se ejecutan en el orden en que llegan, sin interrupciones

02

SJF (Shortest Job First) No Expropiativo

Los procesos más cortos se ejecutan primero, pero una vez que un proceso comienza a ejecutarse, no se puede interrumpir

03

Planificación con Prioridad No Expropiativa

Los procesos se ejecutan según su prioridad, pero no pueden ser interrumpidos hasta que terminen su ejecución o entren en espera.

Planificador CPU

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

01

FIFO (First In First Out)

Los procesos se ejecutan en el orden en que llegan, sin interrupciones

Los procesos se ejecutan en el orden en que llegan a la cola de listos. El primero que llega es el primero que se ejecuta.

Ventajas: Fácil de implementar y entender.

Desventajas: No tiene en cuenta la duración del proceso. Un proceso largo puede hacer que los demás procesos esperen demasiado tiempo (*problema del convoy*).

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

01

FIFO (First In First Out)

Los procesos se ejecutan en el orden en que llegan, sin interrupciones

02

SJF (Shortest Job First) No Expropiativo

Los procesos más cortos se ejecutan primero, pero una vez que un proceso comienza a ejecutarse, no se puede interrumpir

03

Planificación con Prioridad No Expropiativa

Los procesos se ejecutan según su prioridad, pero no pueden ser interrumpidos hasta que terminen su ejecución o entren en espera.

Planificador CPU

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

02

SJF (Shortest Job First) No Expropiativo

Los procesos más cortos se ejecutan primero, pero una vez que un proceso comienza a ejecutarse, no se puede interrumpir

El proceso con el tiempo de ejecución más corto se ejecuta primero

Ventajas: Minimiza el tiempo promedio de espera.

Desventajas: Difícil de predecir el tiempo exacto de cada proceso, y puede ocurrir el problema de inanición, donde procesos largos nunca obtienen CPU.

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

01

FIFO (First In First Out)

Los procesos se ejecutan en el orden en que llegan, sin interrupciones

02

SJF (Shortest Job First) No Expropiativo

Los procesos más cortos se ejecutan primero, pero una vez que un proceso comienza a ejecutarse, no se puede interrumpir

03

Planificación con Prioridad No Expropiativa

Los procesos se ejecutan según su prioridad, pero no pueden ser interrumpidos hasta que terminen su ejecución o entren en espera.

Planificador CPU

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

03

Planificación con Prioridad No Expropiativa

Los procesos se ejecutan según su prioridad, pero no pueden ser interrumpidos hasta que terminen su ejecución o entren en espera.

Cada proceso tiene una prioridad asignada. Los procesos con prioridad más alta se ejecutan antes que los de menor prioridad. Puede ser estática o dinámica

Ventajas: Permite dar preferencia a procesos más importantes o críticos.

Desventajas: Puede ocurrir inanición si los procesos de baja prioridad nunca reciben CPU. Para mitigar esto, se puede implementar envejecimiento (aging), donde los procesos que esperan mucho tiempo incrementan su prioridad.

Algoritmos de planificación

Tipo de planificación

NO EXPROPIATIVA

01

FIFO (First In First Out)

Los procesos se ejecutan en el orden en que llegan, sin interrupciones

02

SJF (Shortest Job First) No Expropiativo

Los procesos más cortos se ejecutan primero, pero una vez que un proceso comienza a ejecutarse, no se puede interrumpir

03

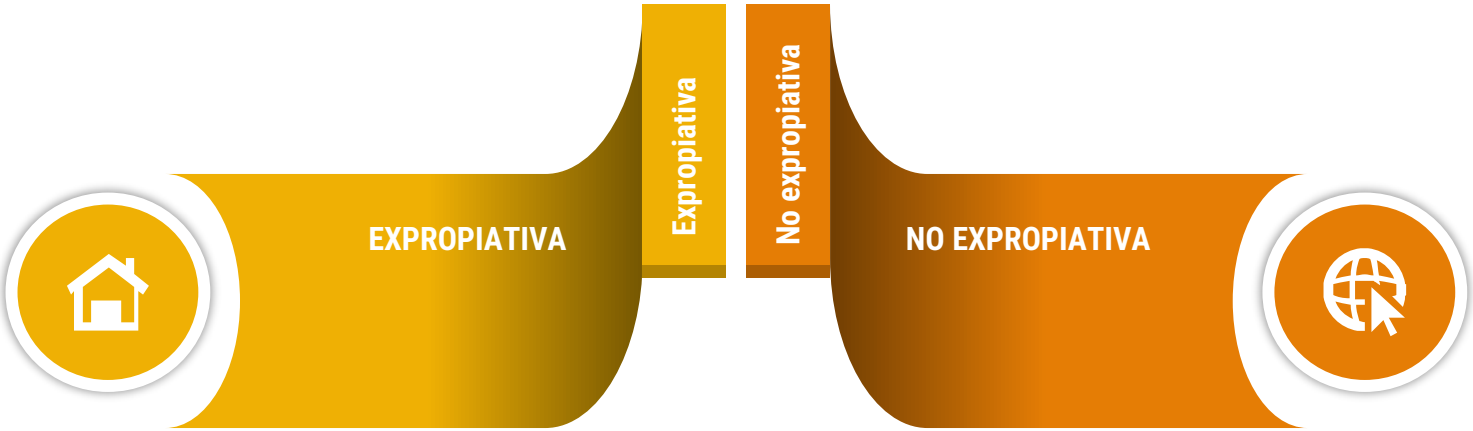
Planificación con Prioridad No Expropiativa

Los procesos se ejecutan según su prioridad, pero no pueden ser interrumpidos hasta que terminen su ejecución o entren en espera.

Planificador CPU

Algoritmos de planificación

Tipo de planificación



Algoritmos de planificación

Tipo de planificación

EXPROPIATIVA

Enfoque donde un proceso puede ser interrumpido y despojado de la CPU, incluso si no ha terminado su tiempo asignado.
Esto permite que otros procesos, que pueden tener mayor prioridad o que son más críticos, obtengan acceso a la CPU.

Ventajas

- Mejora la capacidad de respuesta del sistema, especialmente en entornos interactivos donde los procesos deben responder rápidamente a las entradas del usuario.
- Permite que el sistema operativo utilice la CPU de manera más eficiente al dar preferencia a los procesos que requieren atención inmediata.

Desventajas

- Los cambios frecuentes de contexto pueden llevar a una sobrecarga, ya que cada cambio de proceso implica guardar y cargar estados.
- Los procesos de baja prioridad pueden verse continuamente desplazados y no recibir tiempo de CPU, lo que puede llevar a situaciones de inanición (Starvation).

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Planificación con Prioridad

Los procesos se ordenan según su prioridad, y el de mayor prioridad es asignado a la CPU, interrumpiendo a los de menor prioridad si es necesario

02

SJF Expropiativo (Shortest Job First)

Similar al SJF, pero permite que los procesos más cortos interrumpan a los que están en ejecución si llegan a la cola

03

Planificación con Colas Multinivel (MLQ)

Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Planificador CPU

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

- La **planificación por turno circular** o **Round Robin (RR)** es uno de los algoritmos de planificación más simples y comunes en sistemas multitarea.
- Está diseñado para entornos **multiusuario** donde se requiere equidad y buena respuesta en tiempos compartidos

Algoritmos de planificación

Algoritmos expropiativos

Planificador

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Características

Expropiativo

Si el tiempo de quantum de un proceso se agota, la CPU se expropia y se le da al siguiente proceso en la cola

03

02

Ciclo de Procesos

Los procesos se colocan en una cola circular y se ejecutan por turnos. Si un proceso no termina durante su quantum, es interrumpido y colocado al final de la cola, y la CPU se asigna al siguiente proceso

Quantum de Tiempo

Se asigna un pequeño intervalo de tiempo (llamado "quantum" o "cuanto") a cada proceso

01

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Ventajas

- Ofrece equidad: Todos los procesos reciben un tiempo de CPU justo, por lo que es adecuado para sistemas interactivos.
- Buena respuesta en sistemas donde muchos procesos cortos e interactivos están en ejecución

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Desventajas

- Si el quantum es demasiado corto, puede haber un exceso de cambios de contexto, lo que disminuye la eficiencia.
- Si el quantum es demasiado largo, se aproxima a un algoritmo no expropiativo como FIFO.

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Planificación con Prioridad

Los procesos se ordenan según su prioridad, y el de mayor prioridad es asignado a la CPU, interrumpiendo a los de menor prioridad si es necesario

02

SJF Expropiativo (Shortest Job First)

Similar al SJF, pero permite que los procesos más cortos interrumpan a los que están en ejecución si llegan a la cola

03

Planificación con Colas Multinivel (MLQ)

Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Planificador CPU

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación con Prioridad

02

Los procesos se ordenan según su prioridad, y el de mayor prioridad es asignado a la CPU, interrumpiendo a los de menor prioridad si es necesario

En la planificación con prioridad, a cada proceso se le asigna una prioridad, y la CPU se asigna al proceso con la prioridad más alta. En el caso expropiativo, si un proceso con mayor prioridad llega mientras otro de menor prioridad está ejecutándose, el proceso actual es interrumpido. Las prioridades pueden ser fijas (estáticas) o pueden cambiar durante la ejecución (dinámicas)

Ventajas: Permite dar preferencia a procesos más importantes o críticos.

Desventajas: Puede ocurrir inanición si los procesos de baja prioridad nunca reciben CPU. Para mitigar esto, se puede implementar envejecimiento (aging), donde los procesos que esperan mucho tiempo incrementan su prioridad.

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Planificación con Prioridad

Los procesos se ordenan según su prioridad, y el de mayor prioridad es asignado a la CPU, interrumpiendo a los de menor prioridad si es necesario

02

SJF Expropiativo (Shortest Job First)

Similar al SJF, pero permite que los procesos más cortos interrumpan a los que están en ejecución si llegan a la cola

03

Planificación con Colas Multinivel (MLQ)

Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Planificador CPU

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

SJF Expropiativo (Shortest Job First)

Similar al SJF, pero permite que los procesos más cortos interrumpan a los que están en ejecución si llegan a la cola

03

Versión expropiativa del algoritmo SJF (Shortest Job First). Si un proceso con un tiempo restante más corto llega, se interrumpe el proceso actual y el proceso con menor tiempo restante toma la CPU.

Ventajas: Minimiza el tiempo de espera promedio, ya que se da preferencia a los trabajos más cortos

Desventajas: Similar a SJF, tiene el riesgo de inanición para procesos largos. Puede ser difícil de implementar, ya que requiere conocer o estimar con precisión el tiempo restante de cada proceso.

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Planificación con Prioridad

Los procesos se ordenan según su prioridad, y el de mayor prioridad es asignado a la CPU, interrumpiendo a los de menor prioridad si es necesario

02

SJF Expropiativo (Shortest Job First)

Similar al SJF, pero permite que los procesos más cortos interrumpan a los que están en ejecución si llegan a la cola

03

Planificación con Colas Multinivel (MLQ)

Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Planificador CPU

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación con Colas Multinivel (MLQ)

Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Características

- **Colas de Prioridades**: Los procesos de cada cola tienen diferente prioridad. Por ejemplo, los procesos interactivos pueden tener una prioridad más alta que los procesos batch.
- **Algoritmos en Cada Cola**: Cada cola puede utilizar un algoritmo de planificación diferente. Por ejemplo, la cola de procesos interactivos puede usar Round Robin, mientras que la cola de procesos batch puede usar FIFO.
- **Expropiativo entre Colas**: Un proceso en una cola de mayor prioridad puede interrumpir a un proceso en una cola de menor prioridad.

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación con Colas Multinivel (MLQ)

Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Ventajas:

Flexibilidad para manejar diferentes tipos de procesos, ya que cada grupo de procesos puede tener un algoritmo de planificación óptimo.

Desventajas:

Inanición: Los procesos en colas de baja prioridad pueden ser ignorados si los procesos en colas de mayor prioridad ocupan constantemente la CPU.

Algoritmos de planificación

Algoritmos expropiativos

EXPROPIATIVA

Planificación por Turno Circular (Round Robin, RR)

Cada proceso recibe un cuanto de tiempo, y después de usarlo, se mueve al final de la cola

01

Planificación con Prioridad

Los procesos se ordenan según su prioridad, y el de mayor prioridad es asignado a la CPU, interrumpiendo a los de menor prioridad si es necesario

02

SJF Expropiativo (Shortest Job First)

Similar al SJF, pero permite que los procesos más cortos interrumpan a los que están en ejecución si llegan a la cola

03

Planificación con Colas Multinivel (MLQ)

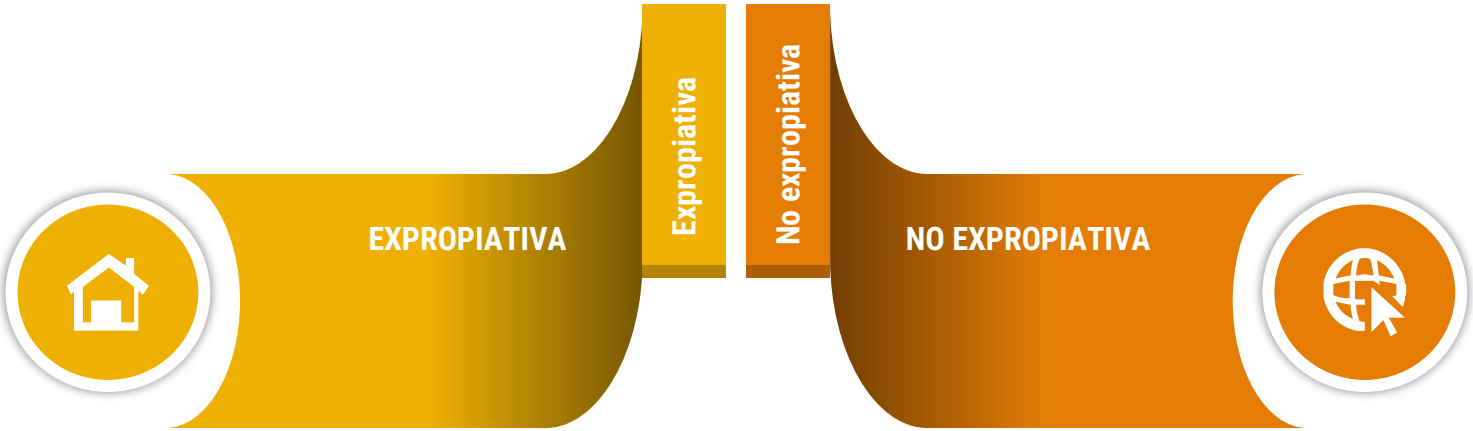
Los procesos se clasifican en diferentes colas de acuerdo con ciertas características

04

Planificador CPU

Algoritmos de planificación

Tipo de planificación



Tema 1.1 Conceptos básicos

Contenidos

Introducción
Definiciones

Conceptos fundamentales
Procesos

Implementación de procesos
Operaciones y cambios de contexto

Hilos (Threads)
Conceptos

Planificador CPU
Algoritmos

Sistemas Operativos y Distribuidos

Iren Lorenzo Fonseca
iren.fonseca@.ua.es

TEMA 1. Sistemas Operativos.

Gestión de procesos