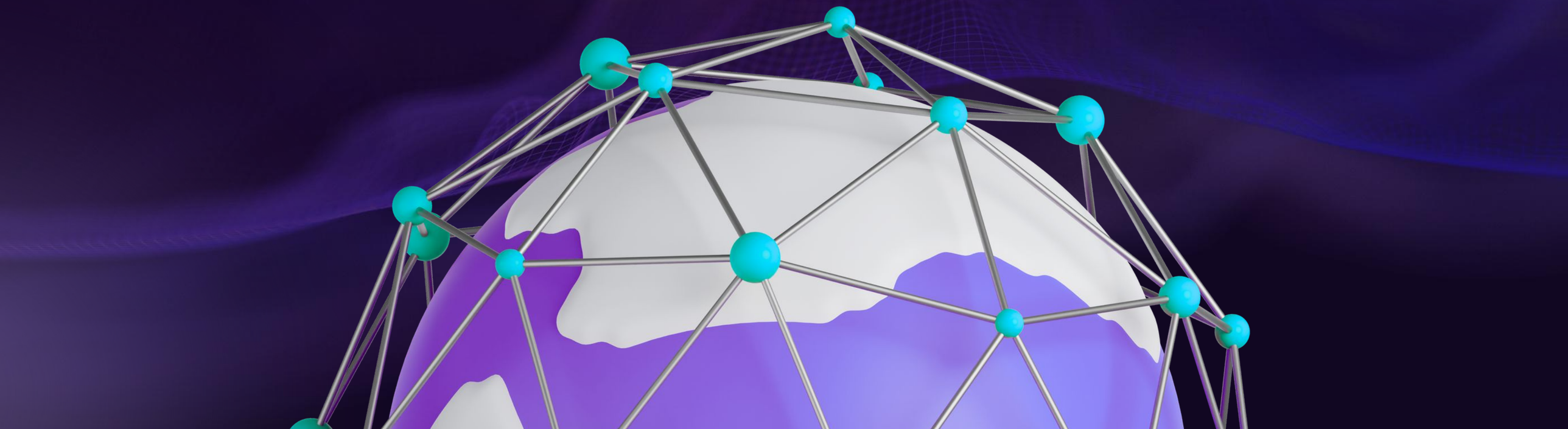
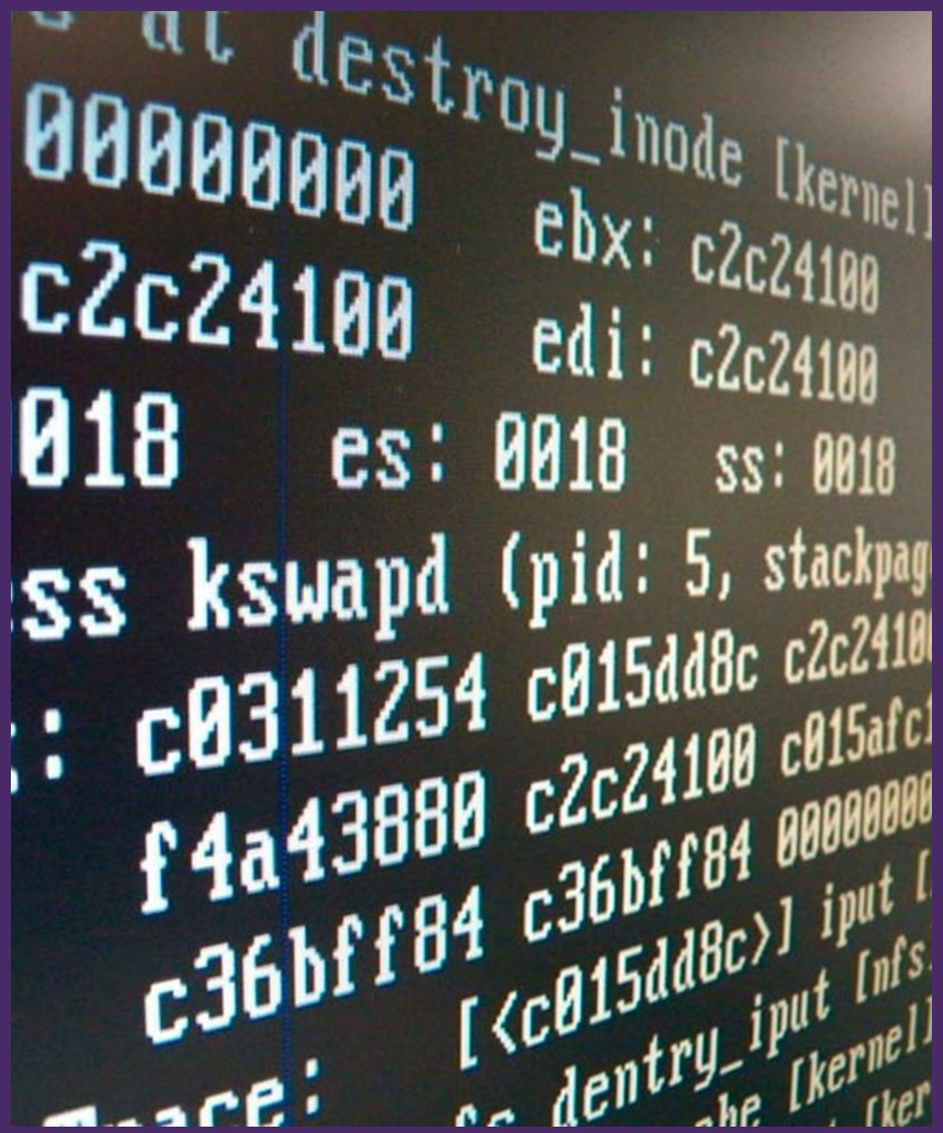
LABORATORIO SISTEMAS OPERATIVOS 2

GESTIÓN DE PROCESOS

¿QUÉ ES UN PROGRAMA?

video: https://www.youtube.com/watch?v=qgK0z08_Is4





CARACTERISTICAS DE UN PROCESO

Todo proceso sin importar el sistema operativo debe de tener las siguientes caracteristicas fundamentales:

- ID de proceso, ID de grupo de proceso, ID de usuario e ID de grupo
- Ambiente
- Directorio de trabajo

Un proceso también proporciona un espacio de direcciones común y recursos comunes del sistema:

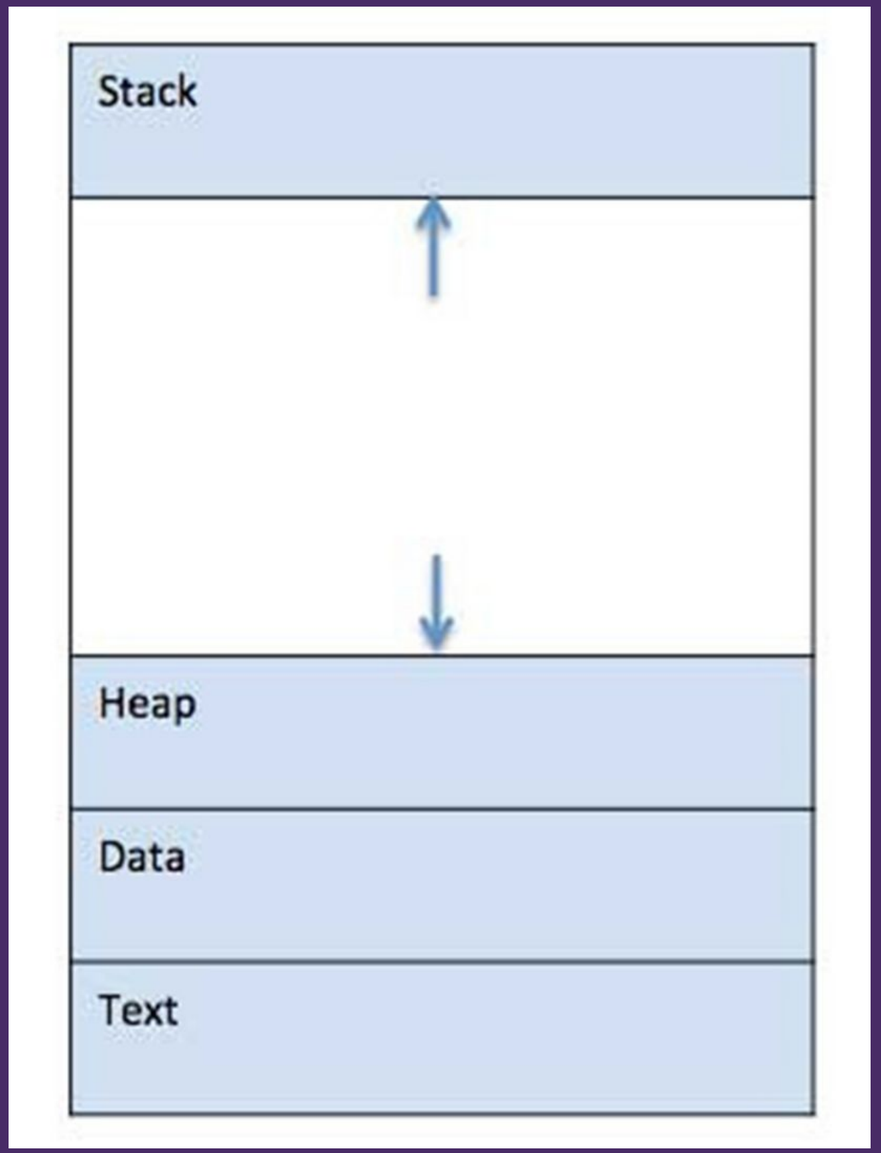
- Descriptores de archivos
- Acciones de señal
- Bibliotecas compartidas
- Herramientas de comunicación entre procesos (como colas de mensajes, pipes, semáforos o memoria compartida)

Cuando un programa se carga en la memoria y se convierte en un proceso, se puede dividir en cuatro secciones: pila, montón (heap), texto y datos.

La "stack" es una estructura de datos que almacena información sobre las subrutinas activas de un programa de computadora y se utiliza como espacio temporal para el proceso.

Se distingue de la memoria asignada dinámicamente para el proceso que se conoce como "heap".

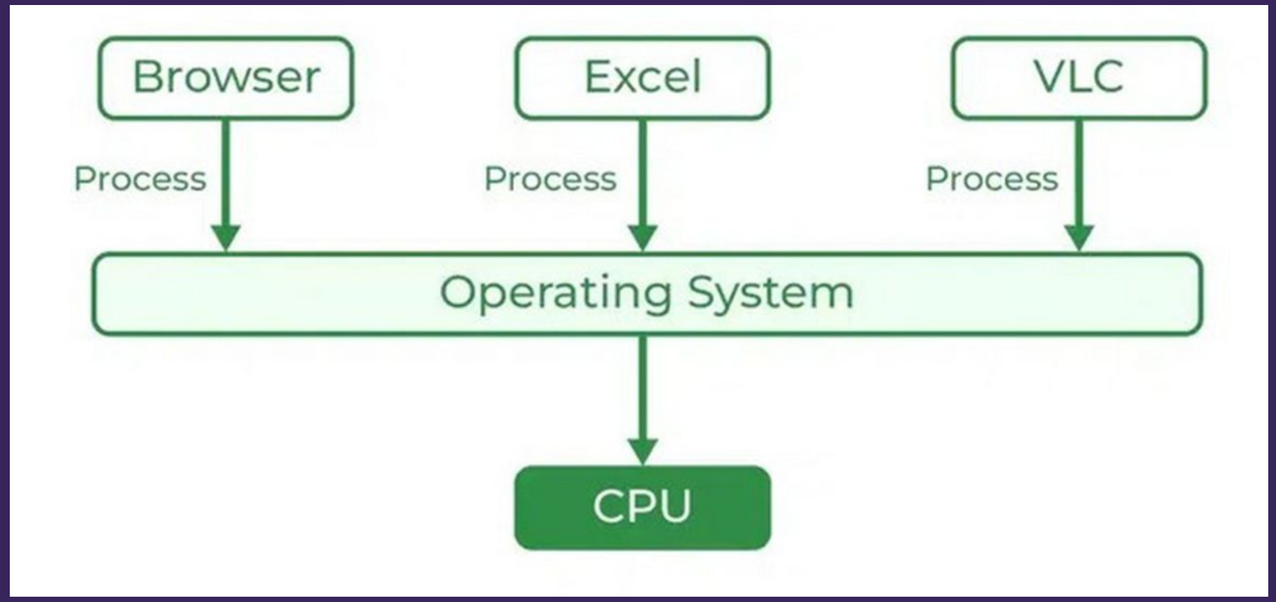
La seccion de texto o codigo incluye la actividad actual representada por el valor del contador de programa y el contenido de los registros del procesador, que pueden contener una instrucción, una dirección de almacenamiento u otro tipo de datos necesarios para el proceso.



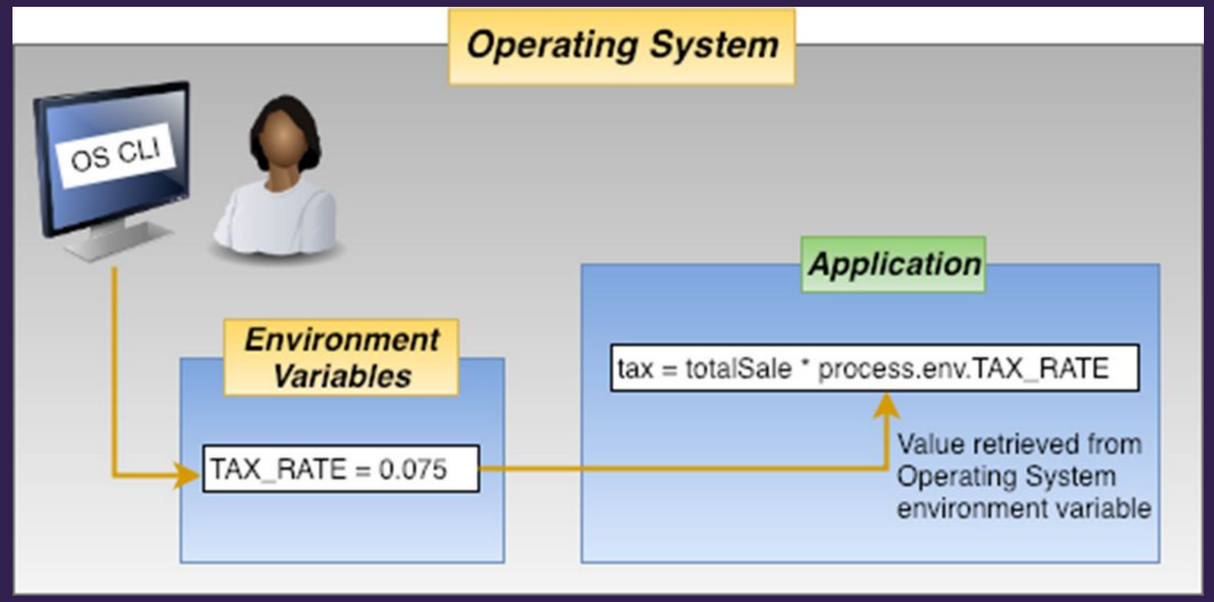
Puede haber varias instancias de un solo programa y cada instancia de ese programa en ejecución es un proceso.

Cada proceso tiene un espacio de direcciones de memoria independiente, lo que significa que un proceso se ejecuta de forma independiente y está aislado de otros procesos. No puede acceder directamente a datos compartidos en otros procesos.

Esta independencia de los procesos es valiosa porque el sistema operativo hace todo lo posible por aislarlos procesos para que un problema con un proceso no corrompa ni cause estragos en otro proceso.



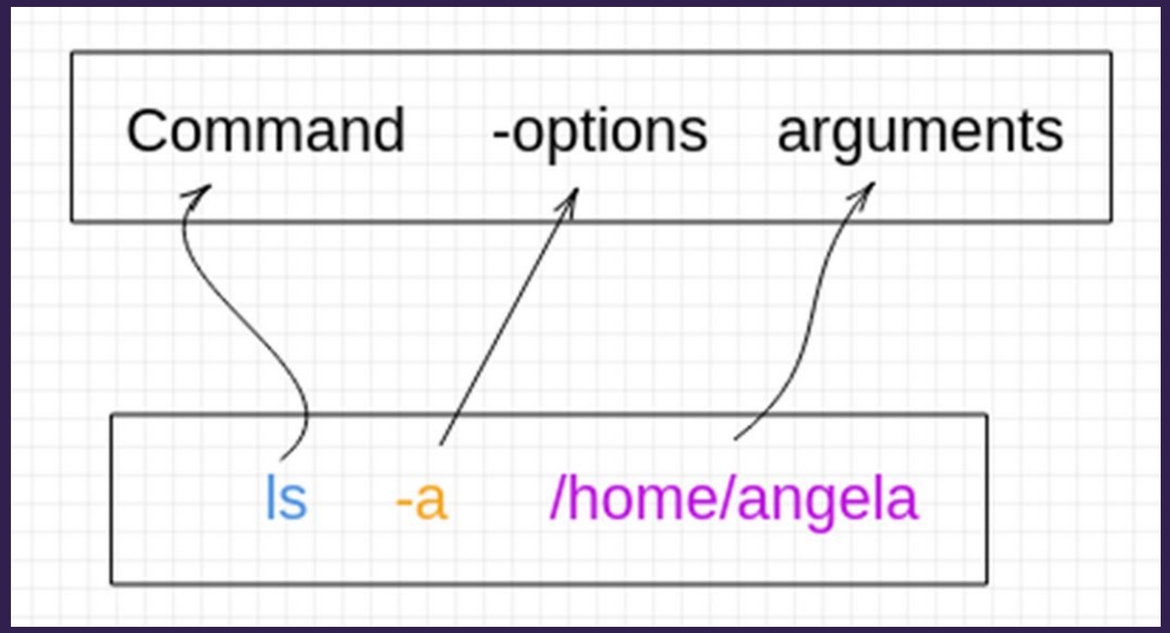
ENVIROMENT



El entorno o enviroment es una lista de variables de entorno heredadas del proceso principal que ejecuta una función exec.

Una variable de entorno contiene una cadena de caracteres de longitud variable. Por ejemplo, PATH es una variable del sistema exportada en /etc/environment que contiene una lista de directorios que utiliza el shell para buscar archivos ejecutables binarios al intentar ejecutar un programa.

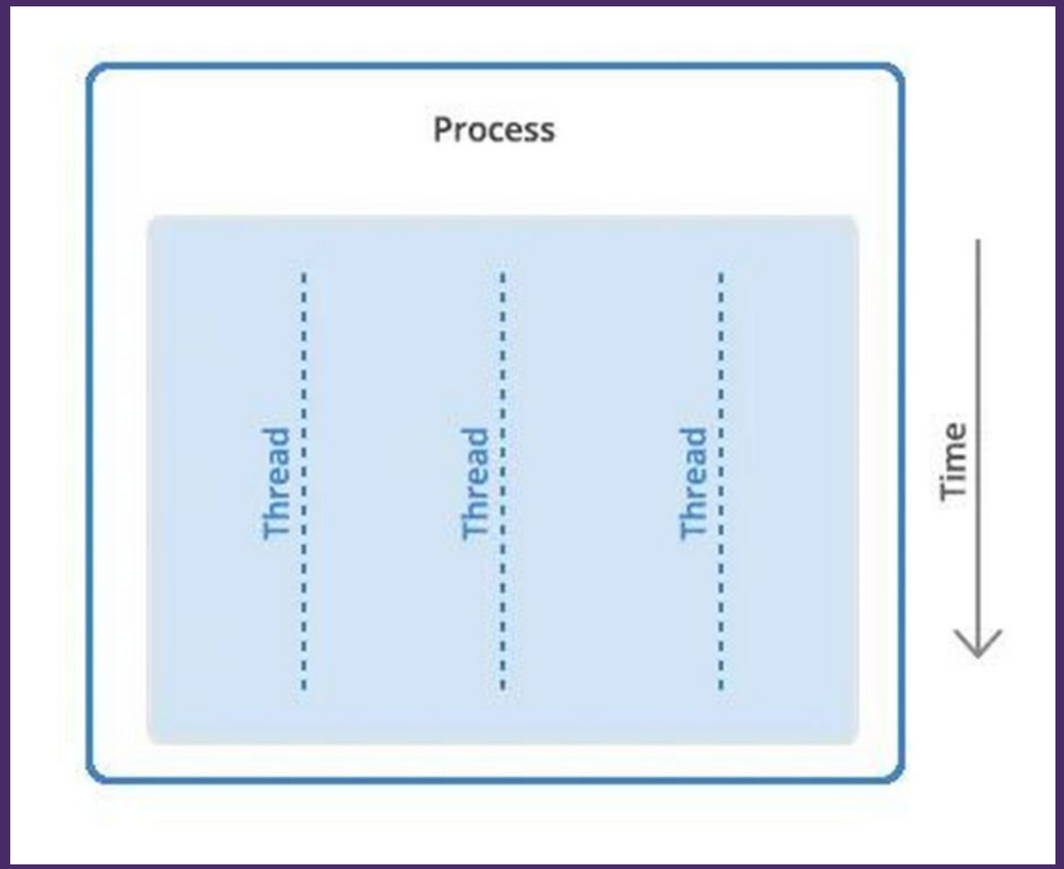
ARGUMENTS



Los argumentos son una lista de valores de datos pasados a un nuevo proceso.

Los argumentos se ingresan en la terminal o consola después de ingresar el comando. Se puede escribir más de un argumento juntos, estos se procesarán en el ordenen que se escriban.

HILOS (THREADS)



Un hilo de ejecución es la secuencia más pequeña de instrucciones que un programador puede gestionar de forma independiente.

El sistema operativo crea y administra hilos, estos comparten la misma memoria y recursos que el programa que los creó, permitiendo que múltiples threads colaboren y trabajen de manera eficiente dentro de un solo programa.

CARACTERISTICAS DE LOS HILOS

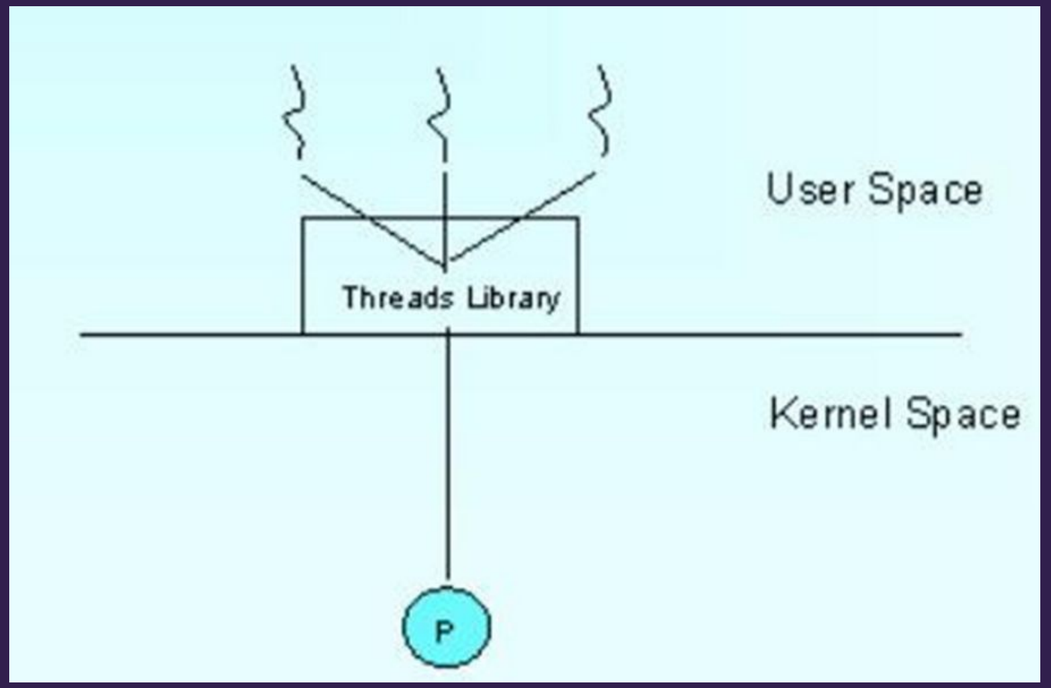
Un hilo es la entidad programable. Sólo tiene aquellas propiedades que se requieren para garantizar su control independiente del flujo. Estos incluyen las siguientes propiedades:

- Pila
- Propiedades de programación (como política o prioridad)
- Conjunto de señales pendientes y bloqueadas.
- Algunos datos específicos del hilo como indicadores de error

Los hilos dentro de un proceso no deben considerarse como un grupo de procesos. Todos los hilos comparten el mismo espacio de direcciones. Esto significa que dos punteros que tienen el mismo valor en dos hilos hacen referencia a los mismos datos.

Además, si algún hilo cambia uno de los recursos compartidos del sistema, todos los hilos dentro del proceso se ven afectados. Por ejemplo, si un hilo cierra un archivo, el archivo se cierra para todos los hilos.

HILOS DE NIVEL DE USUARIO

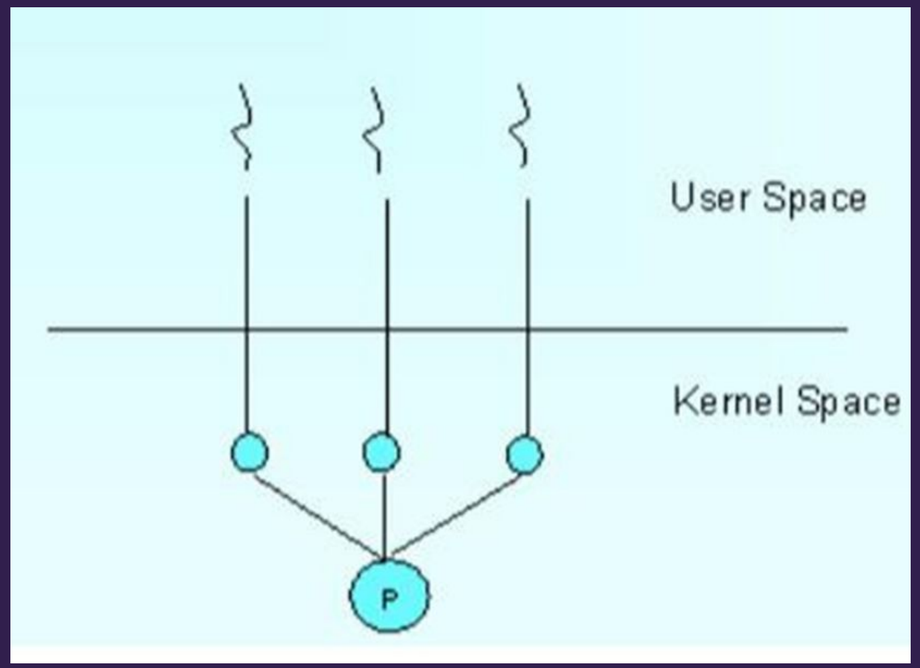


El hilo de nivel de usuario es un tipo de hilo que no se crea mediante llamadas al sistema. El kernel no interviene en la creacion ni gestión de estos hilos, si no que es el usuario quien se encarga de gestionarlos.

Estos suelen ser livianos porque se administran completamente en el espacio del usuario, sin involucrar al kernel. Esto significa que las operaciones de creación de hilos, cambio de contexto y sincronización pueden ser más rápidas en comparación con los subprocesos a nivel de kernel.

Al no estan vinculados con el kernel, estos hilos funcionan de manera consistente en diferentes sistemas operativos y plataformas.

HILOS DE NIVEL DE KERNEL



A diferencia de los hilos a nivel de usuario que se administran completamente en el espacio del usuario mediante una biblioteca de hilos o un entorno de ejecución, los hilos a nivel de kernel son visibles y administrados por e lpropio kernel.

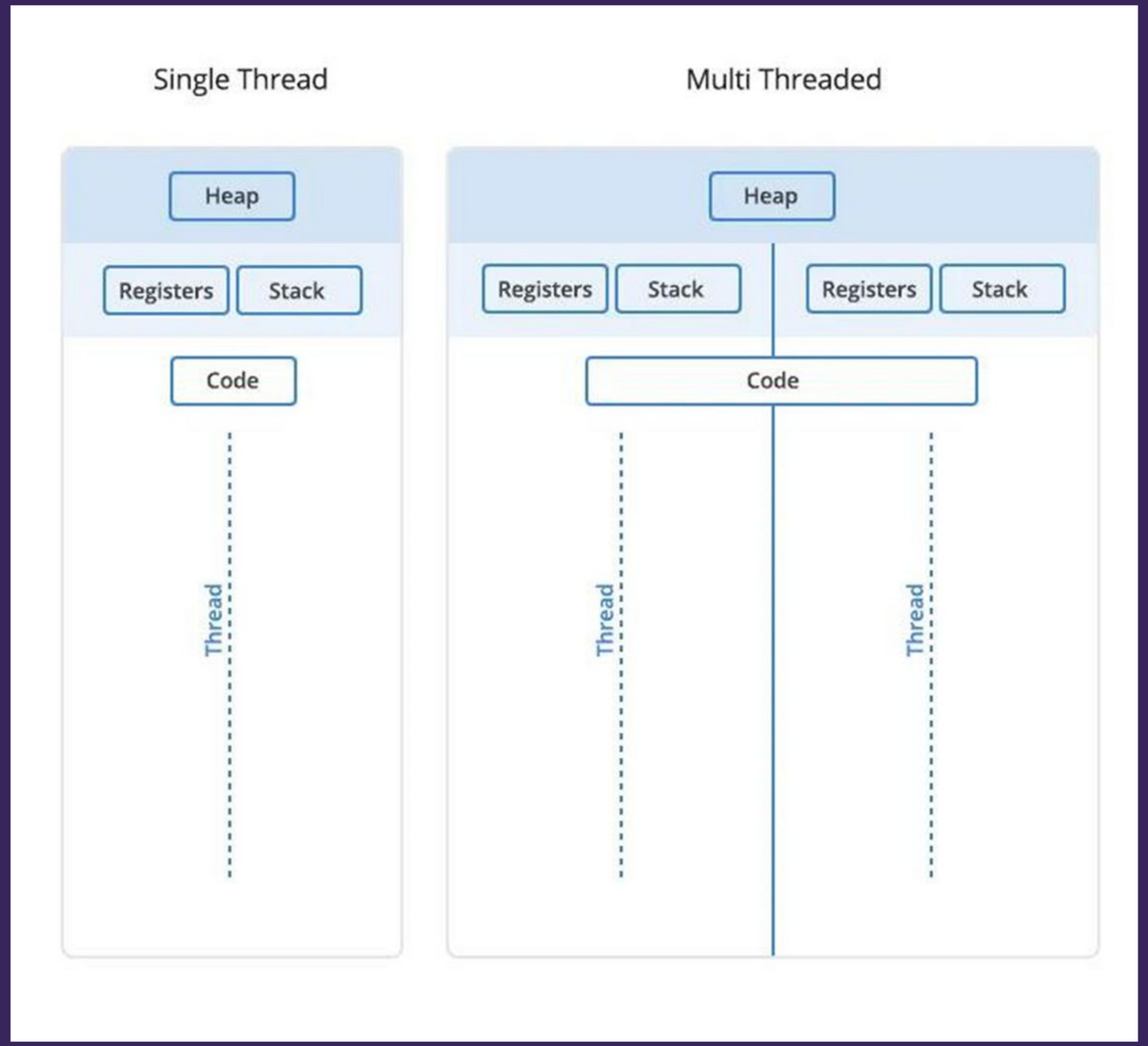
Los hilos a nivel de kernel son compatibles directamente con el kernel del sistema operativo, que proporciona scheduling, cambio de contexto y otras funcionalidades de gestión de subprocesos por medio de llamadas de sistema.

Sin embargo, pueden generar una mayor sobrecarga en comparación con los hilos a nivel de usuario y requieren consideraciones específicas de la plataforma con respecto a la portabilidad y la compatibilidad.

MULTITHREADING

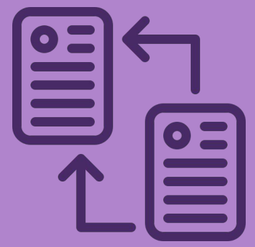
Multithreading es una técnica utilizada en los sistemas operativos para mejorar el rendimiento y la capacidad de respuesta de los sistemas informáticos al permitir que un solo proceso ejecute múltiples tareas al mismo tiempo.

El multithreading permite que varios hilos compartan los mismos recursos de un único proceso, como la CPU, la memoria y los dispositivos de E/S.

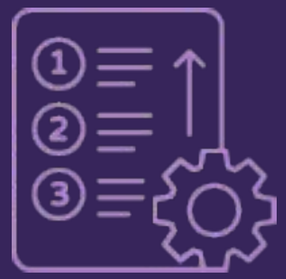


¿POR QUÉ USAR MULTITHREADING?

Los hilos se ejecutan en paralelo mejorando el rendimiento de la aplicación. Cada uno de estos subprocesos tiene su propio estado de CPU y pila, pero comparten el espacio de direcciones del proceso y el entorno.

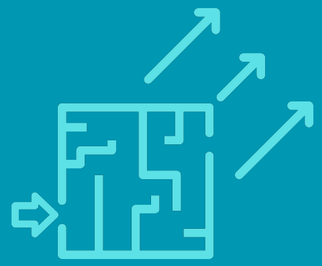


Los hilos pueden compartir datos comunes, por lo que no necesitan utilizar comunicación entre procesos. Al igual que los procesos, los hilos también tienen estados como listo, en ejecución, bloqueado, etc.

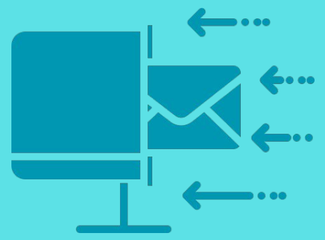


Se puede asignar prioridad a los hilos al igual que los procesos, de esta manera los hilos de mayor prioridad seran programados y ejecutados primero.

¿CUANDO NO USAR MULTITHREADING?



Para programas pequeños y sencillos con flujos de ejecución lineal, la introducción de multiples hilos puede agregar complejidad innecesaria sin proporcionar mejoras significativas en el rendimiento. En tales casos, la sobrecarga de administrar subprocesos puede superar cualquier beneficio potencial.



Si la aplicación está principalmente vinculada a E/S (por ejemplo, comunicación de red, operaciones de archivos), agregar más hilos no necesariamente mejorará el rendimiento. De hecho, podría incluso degradar el rendimiento debido a la sobrecarga del cambio de contexto entre hilos.

No todas las plataformas y lenguajes de programación admiten multithreading por igual. Si la aplicación necesita ser portátil en diferentes plataformas o lenguajes que carecen de soporte sólido para multithreading, podría ser mejor evitar depender en gran medida de este.

¡GRACIAS POR LA ATENCIÓN!

¿Dudas?