Programación de Servicios y Procesos

Programación Multiproceso

Unidad 1: Procesos

INDEX

1. OBJETIVOS

- Aprender los conceptos básicos relacionados con la ejecución de programas.
- Comprender cómo el sistema operativo gestiona los procesos.
- Crear aplicaciones sencillas de múltiples procesos.

2. INTRODUCCIÓN

- **Primeras computadoras:**
  - No tenían sistema operativo.
  - Ejecución de un solo programa a la vez de principio a fin.
  - Los programas tenían acceso directo a los recursos de la máquina.
    - **MUY INEFICIENTE!!!!**

- **Ordenadores modernos:**
  - Sistema operativo multitarea, CPU multicore,....
  - Los procesos se ejecutan simultáneamente en el sistema operativo aislados entre sí y se comunican mediante sockets, memoria compartida, archivos, etc.
  - **ES MÁS CONVENIENTE DIVIDIR UNA GRAN TAREA EN PEQUEÑAS TAREAS COORDINADAS: UTILIZACIÓN ADECUADA DE LOS RECURSOS**

INTRODUCCIÓN

- El **Multithreading** se encuentra en los sistemas operativos multipgramados modernos.
  - Multithreading permite que existan varios subprocesos en el contexto de un proceso.
  - Los Threads comparten recursos de proceso y también se denominan procesos ligeros.
  - Los Threads son capaces de ejecutarse de forma asincrónica e independiente en varias CPU.
  - Los desarrolladores utilizan multithreading para implementar la ejecución simultánea dentro de un proceso.

3. CONCEPTOS BÁSICOS

Concurrencia

- Ejecución de dos o más programas **independientes** que interactúan durante el mismo período de tiempo.
  - *Ejemplo1: Ordenador personal.*

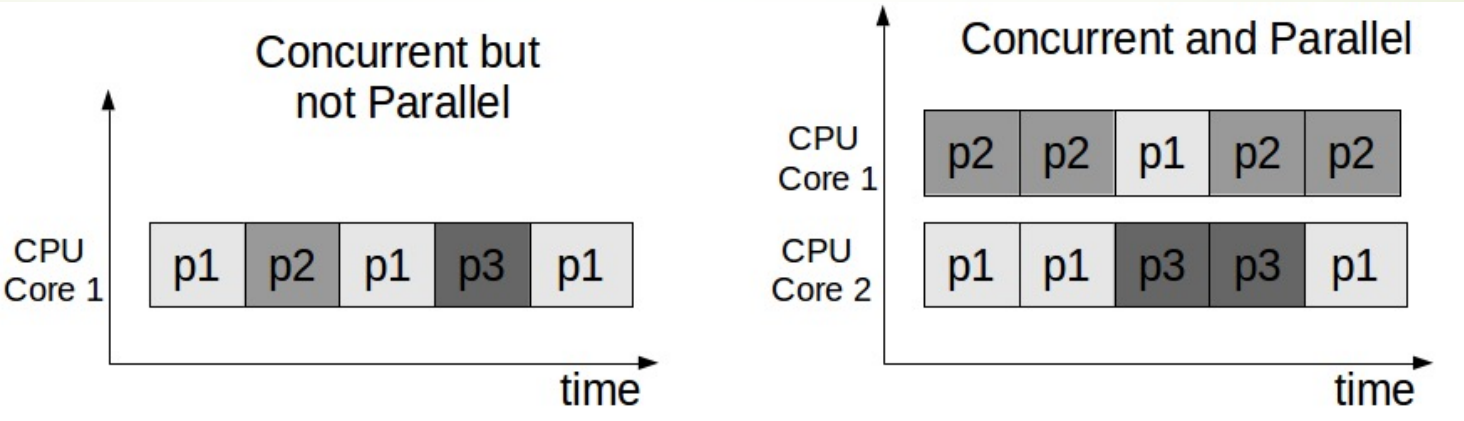
    El usuario abre y utiliza muchas aplicaciones mientras que en segundo plano, el reloj y el escáner de virus también se están ejecutando. El sistema operativo espera a que el usuario pida que se inicien más programas, al tiempo que maneja tareas subyacentes, como resolver qué información de Internet va a qué programa..

  - *Ejemplo2: Reserva de hotel en línea.*

    Este es un ejemplo de sistema distribuido concurrente.

BASIC CONCEPTS

- . La concurrencia puede ser **intercalada** o incluso **simultánea**



- Un sistema concurrente puede ser implementado a través de **procesos** o **hilos**

CONCEPTOS BÁSICOS

PROCESO:

- Programa en ejecución "ejecutándose" en el Sistema operativo.

PROGRAMA:

- Secuencia de instrucciones escritas para realizar una tarea específica.

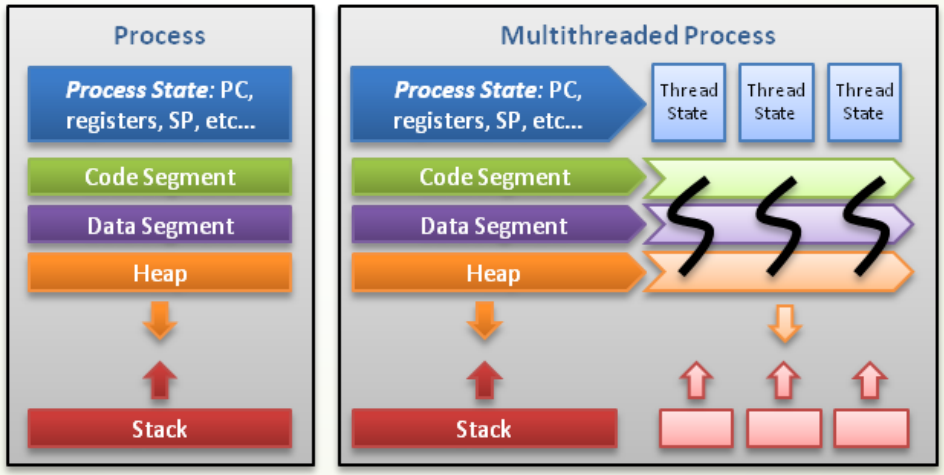
Ejemplo:

Al iniciar dos instancias de un programa de procesador de texto, el sistema operativo asigna dos procesos independientes con su propio espacio de memoria.

BASIC CONCEPTS

Hilo *(Thread)*:

- Es una unidad de trabajo dentro del proceso.
- Puede ser uno o más hilos que se ejecutan dentro del ámbito de un proceso.



Threads contain only necessary information, such as a stack (for local variables, function arguments, return values), a copy of the registers, program counter and any thread-specific data to allow them to be scheduled individually. Other data is shared within the process between all threads.

Conceptos Básicos

Hilo vs Proceso:

- *Un **hilo** tiene acceso al espacio de direcciones de memoria y a los recursos de su proceso, sin embargo, un **proceso** no puede acceder a las variables o recursos asignados para otros procesos.*

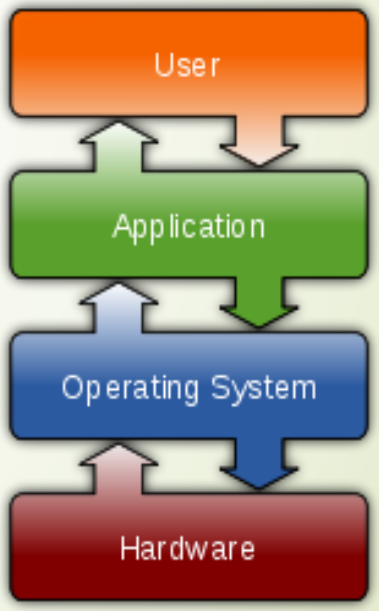
EXECUTABLE FILE

- Archivo de sistema que contiene instrucciones que el sistema operativo de un equipo puede entender y seguir.
- La ejecución de este archivo ejecuta el programa como un proceso.

Conceptos Básicos

Sistema Operativo

- Colección de software que administra los recursos de hardware del equipo y proporciona servicios comunes para programas informáticos:
  - Hace que la computadora amigable.
  - Executar programas de usuario.
  - Optimizar los recursos del sistema.



CONCEPTOS BÁSICOS

Demonio (or Servicio)

- Es un programa informático que se ejecuta como un proceso en segundo plano, en lugar de estar bajo el control directo de un usuario interactivo..
- *El sistema operativo a menudo inicia demonios en el momento del arranque, por ejemplo.:*
  - *para atender las solicitudes de red y la actividad de hardware,*
  - *Para realizar otras tareas, como la configuración de hardware (ex: udevd en algunos sistemas GNU/Linux)*

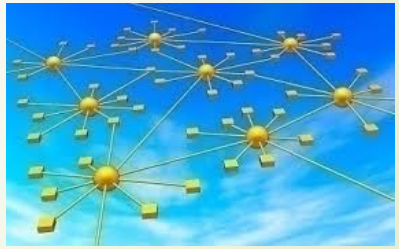
CONCEPTOS BÁSICOS

COMPUTACIÓN PARALELA

- Es una forma de cálculo en la que muchos cálculos se llevan a cabo simultáneamente en un sistema multiprocesador multinúcleo/CPU.

COMPUTACIÓN DISTRIBUIDA

- Estudia los **sistemas distribuidos**: Los componentes de software ubicados en computadoras en red se comunican y coordinan para lograr un objetivo común.
- Un problema se divide en muchas tareas y muchos ordenadores trabajan en paralelo para resolverlo.
  - *Ejemplos de sistemas distribuidos*: Sistemas basados en SOA, juegos en línea multijugador masivos, aplicaciones peer-to-peer, etc.



¿Qué es SOA?

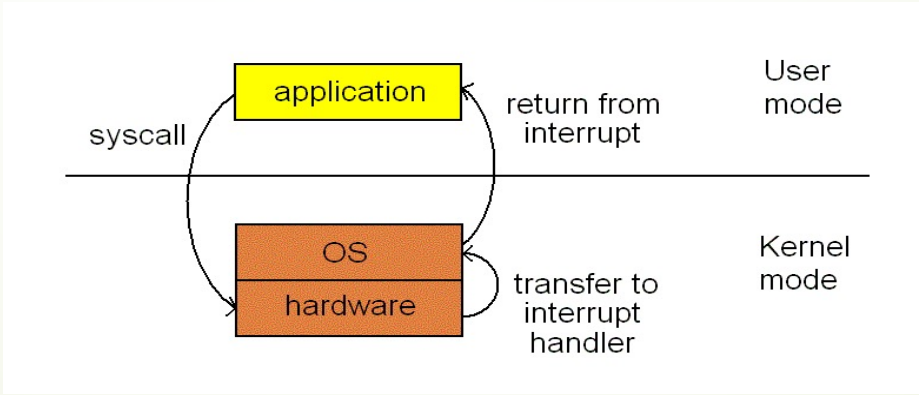
https://www.youtube.com/watch?v=A3_QIYJRVvk(1:16)

Actividad: Busca en Internet SOA y completa la tabla

Significado de SOA:
Características y propósito:
Tipos:
Beneficios:
Inconvenientes:

4. Procesos
Procesos en el sistema operativo

- El sistema operativo incluye el **kernel** y también programas de utilidad, intérpretes de comandos y bibliotecas de programación.
- El kernel responde a **llamadas al sistema**, **interrupciones** de hardware y **excepciones de programas**.
  - Un proceso utiliza llamadas al sistema para solicitar servicios del sistema operativo

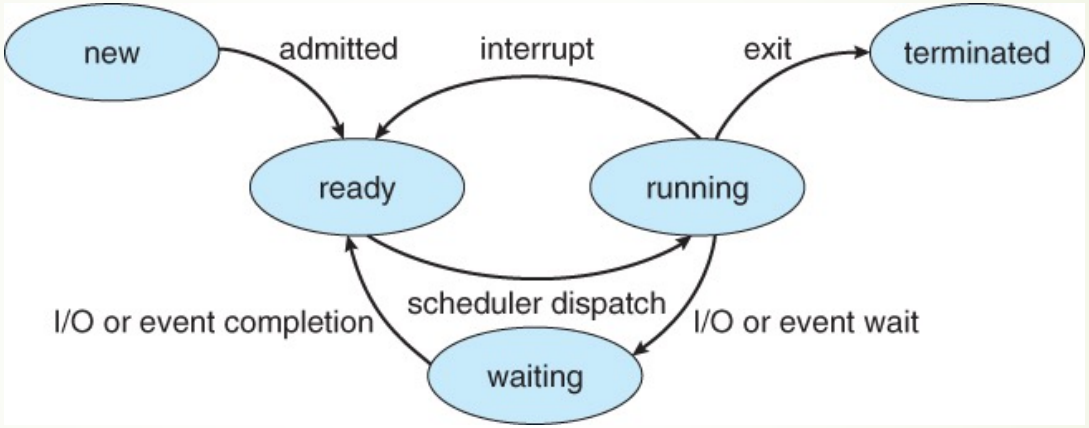


  - *Solo el código del kernel se ejecuta en modo de ejecución de CPU privilegiado/kernel.*
  - *El privilegio permite al kernel controlar HW, ejecutar instrucciones especiales y aislarse de otros procesos.*

4. PROCESOS
Estados de un proceso

- El **sistema operativo** es responsable de ejecutar y gestionar los procesos.
  - Los **componentes** de un proceso son:
    - Instrucciones a ejecutar (código o programa)
    - Datos que se utilizarán durante la ejecución.
    - Recursos necesarios.
    - Estado de la ejecución del proceso
    - Mientras se ejecuta, un proceso pasa por varios **estados**.
- El cambio de estado también se produce a través de la intervención del sistema operativo.

4. Procesos
Estados de un proceso



- **Nuevo** - El proceso está en la etapa de ser creado.
- **Preparado** - El proceso tiene todos los recursos disponibles que necesita para ejecutarse, pero la CPU no está trabajando actualmente según sus instrucciones..
- **Ejecutándose** - La CPU está trabajando en las instrucciones del proceso.
- **A la espera** - El proceso no se puede ejecutar en este momento, porque está esperando que algún recurso esté disponible o que ocurra algún evento. Por ejemplo, el proceso puede estar esperando la entrada del teclado, la solicitud de acceso al disco, los mensajes entre procesos, el apagado de un temporizador o la finalización de un proceso secundario..
- **Terminado -** El proceso ha completado su ejecución.

Actividad: Busque información acerca de las condiciones para la transición de estado del proceso y complete la tabla siguiente:

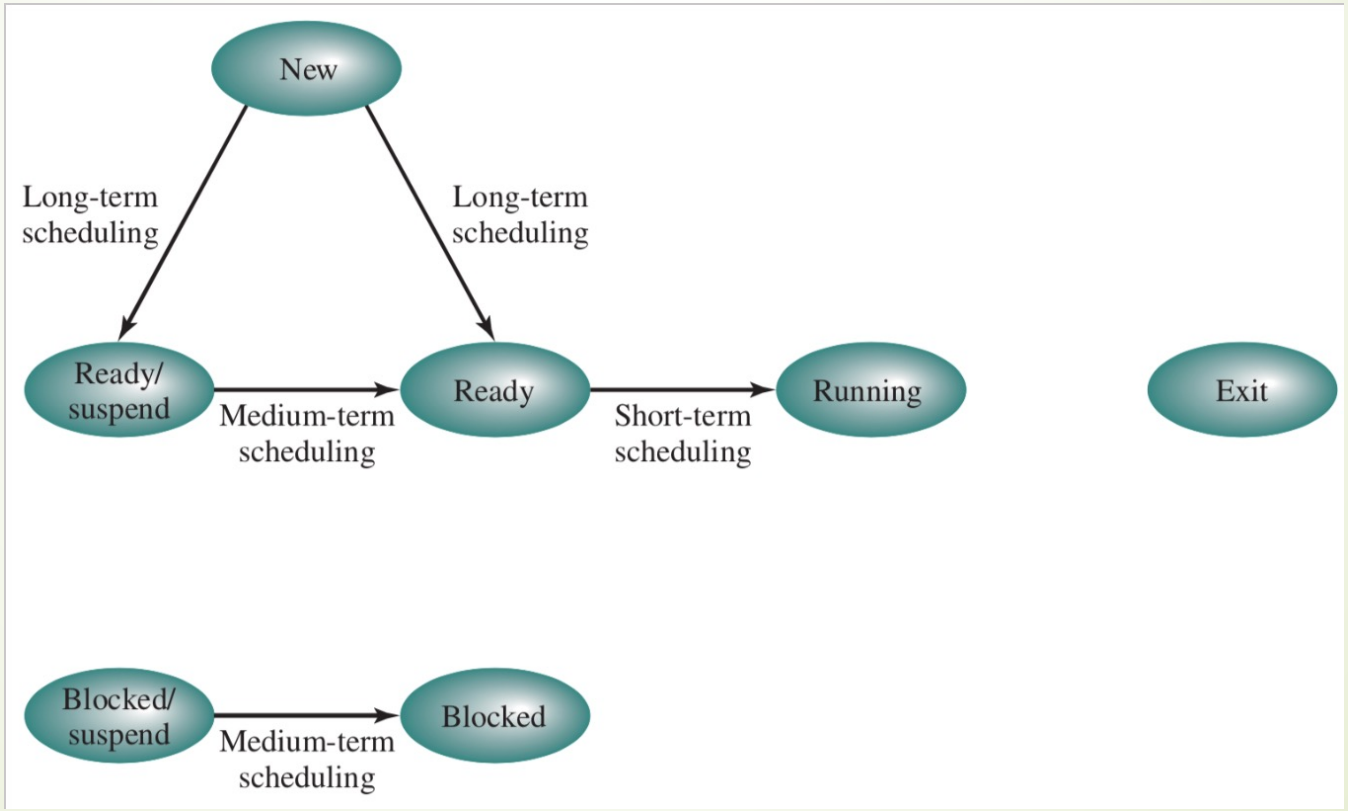
Transición	Descripción de las Condiciones
(Admitido) Nuevo → Preparado	
(Planificador dispatched) Preparado → Ejecutandose	
(Interrumpido, tiempo expirado) Ejecución → Preparado	
(Blocked) Ejecución → Espera	
(Despierta) Espera → Preparado	
(Finaliza, sale) Ejecución→ Terminado	

4. PROCESOS
Planificación de procesos

- La **planificación de procesos** es una parte esencial de un sistema operativo multiprogramación.
- La planificación de procesos maneja la eliminación del proceso en ejecución de la CPU y la selección de otro proceso sobre la base de una **estrategia** particular.
- Los **planficadores** son SW de sistema especial que deciden cuál es el siguiente proceso a ejecutar. Tipos de Planificadores:
  - **Long-term** (Planificador de trabajos): si se debe admitir un proceso en el sistema.
  - **Medium-Term** (intercambio): consideraciones respecto a la memoria. Qué procesos salen de la memoria principal y van al disco?
  - **Short Term** (dispatcher) :! selecciona el siguiente proceso que se ejecutará en la CPU. El **cambio de contexto** de cpu debe ser muy rápido
  - **I/O Scheduler**: decide cuál de los procesos de E/S en espera obtiene acceso al dispositivo de E/S.

4. PROCESOS
Planificación de procesos

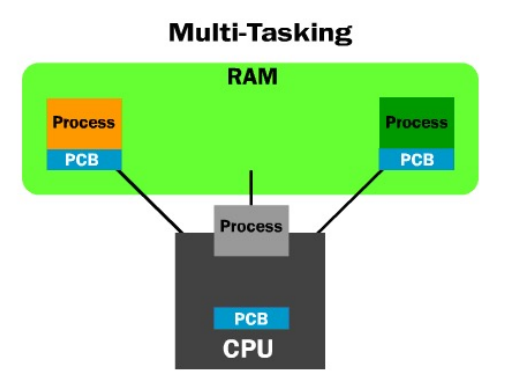
Planificación y Estados de los Procesos



4. PROCESOS
Planificación de procesos

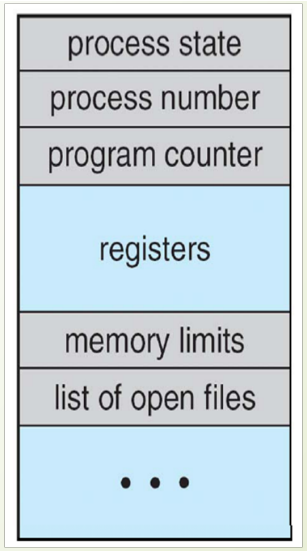
¿Qué es la "multiplexación de Tiempo"?

4. Procesos
Planificación de Procesos



Cambio de Contexto

- Mecanismo que permite que varios procesos compartan una sola CPU.
- Cuando el programador cambia de un proceso para ejecutar otro, el conmutador de contexto guarda el **contexto** del proceso que se está eliminando en su bloque de **control de proceso (PCB).**
- La **Process Table** contiene la PCB para los procesos actuales manejados por el Sistema operativo.



4. PROCESOS
Planifcación de Procesos

Criterios de Planificación

- **Utilización de la CPU:** Mantener la CPU lo más ocupada posible.
- ***Rendimiento***: # de procesos que completan su ejecución por unidad de tiempo.
- **Tiempo de respuesta:** cantidad de tiempo para ejecutar un proceso en particular.
- **Tiempo de espera:** cantidad de tiempo que un proceso ha estado esperando en la cola lista.
- **Tiempo de respuesta:** cantidad de tiempo que se tarda desde que se envió una solicitud hasta que se produce la primera respuesta, no la salida.

OPTIMIZATION CRITERIA

Max utilización CPU

Max rendimiento

Min tiempo de respuesta

Min tiempo de espera

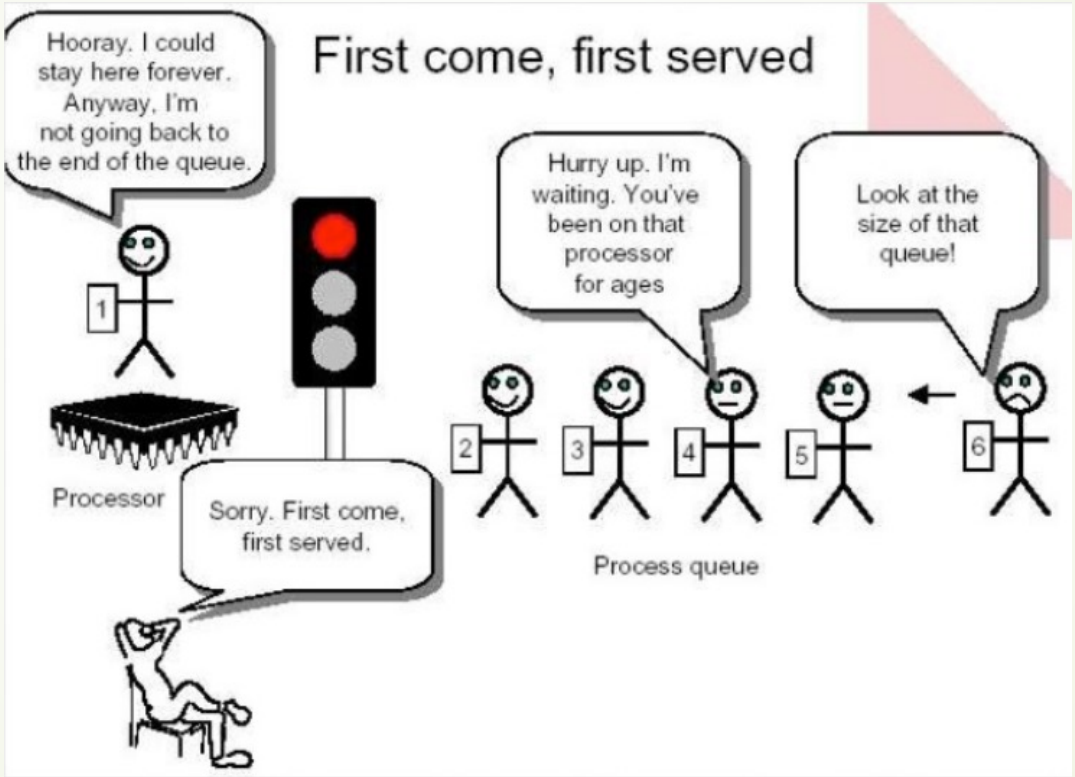
Min tiempo de respuesta

4. PROCESOS
Planificación de Procesos

Algoritmos de Planificación

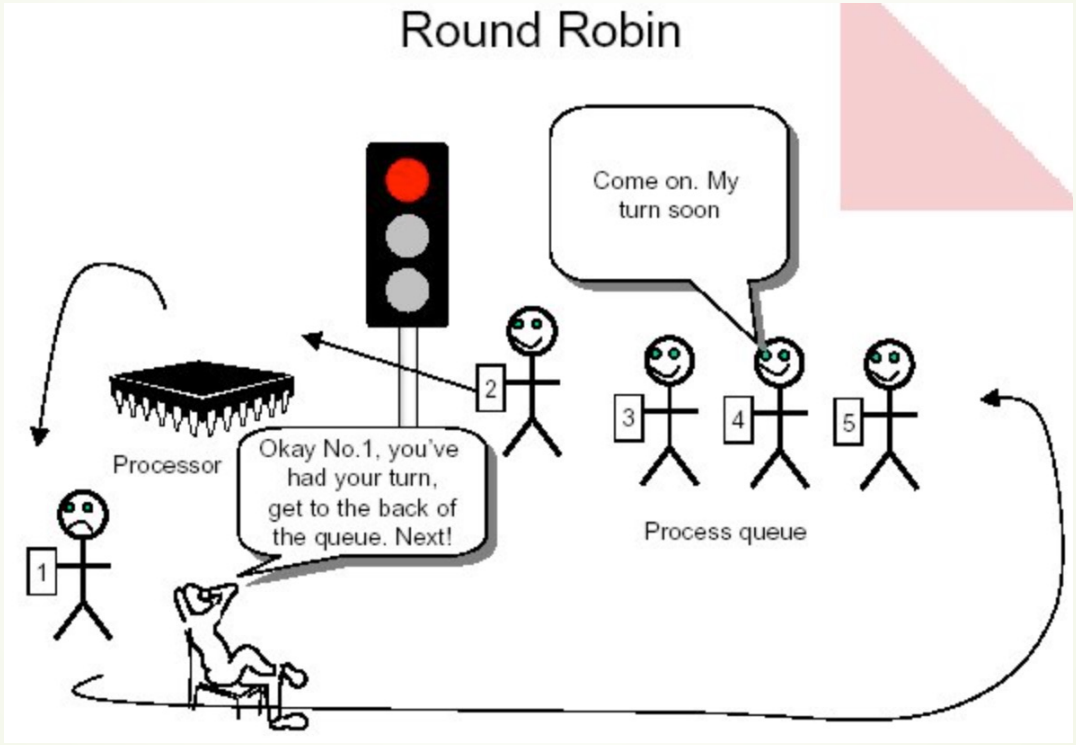
- First Come First Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round Robin (RR) Scheduling
- Multilevel Queue Scheduling

4. PROCESOS
Algoritmos de Planficación



- FIFO cola donde el primer trabajo obtendrá la primera vez.
- Puede ser muy malo si los trabajos cortos vienen después de los trabajos largos.

4. PROCESOS
Algoritmos de Planificación



- Cada proceso tomará turnos durante un tiempo denominado "quanto". Cuando el quantum expira, el proceso actual se quita y se elige otro para ejecutarlo.
- Es un algoritmo muy eficiente, pero no es necesario el mejor. Es malo cuando todos los trabajos son de la misma longitud.

4. PROCESOS
Creación de Procesos

- Principales eventos que provocan la creación de procesos:
- **Arranque** del sistema**.**
  - Una **llamada al sistema** desde un proceso para crear un **nuevo proceso**. (En Unix, la llamada al Sistema **fork** crea un nuevo proceso (hijo) idéntico al proceso padre)
  - **Solicitud del usuario** para crear un proceso.
  - Inicio de **un trabajo por lotes**.
- La mayoría de los lenguajes concurrentes ofrecen alguna variante de los siguientes **mecanismos de creación de procesos**:
  - Co-routines.
  - Fork and Join.
  - Cobegin/coend.

4. PROCESOS
Terminación de Procesos

- **Salida Normal:** return or exit
  - El proceso (**voluntario**) termina su declaración final de main y esto provoca una llamada al sistema que eliminará el proceso y desaslocalará sus recursos.
- **Terminación anormal:** El proceso finaliza cuando se indica:
  - **Salida de error** (**voluntaria**): Un error causado por un error del programa (ejecutar una instrucción ilegal, hacer referencia a memoria inexistente o dividir por cero).

    En algunos sistemas (por ejemplo, **UNIX**), un proceso puede decirle al sistema operativo que desea manejar ciertos errores por sí mismo (excepciones), en cuyo caso, el proceso se señala (interrumpe) en lugar de terminar cuando ocurre uno de los errores.

    Fatal error (**involuntary**): factores externos como un error en la máquina virtual o API de Java, un error en un archivo del sistema, etc.
  - **Matado por otro proceso (involuntario)**: Un proceso puede causar la terminación de otro proceso a través de una llamada al sistema apropiada.

Contesta a la pregunta…

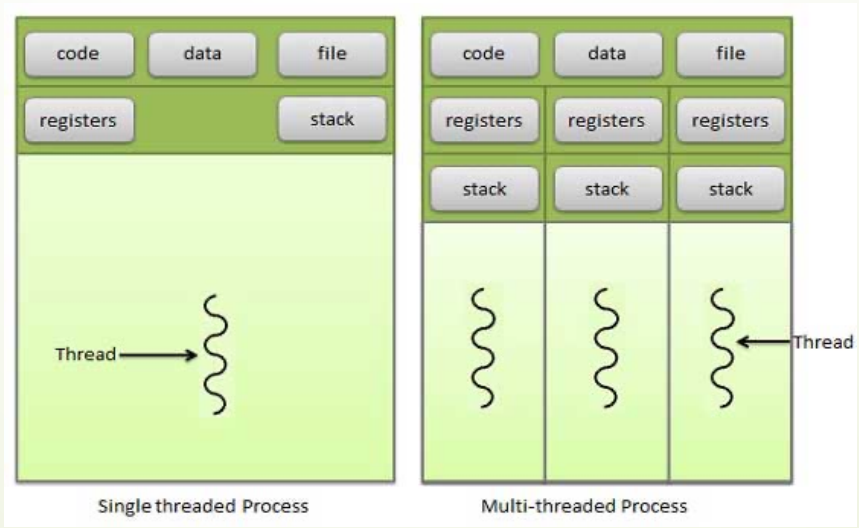
What is the operating system command to **terminate** a process …

…in Linux?

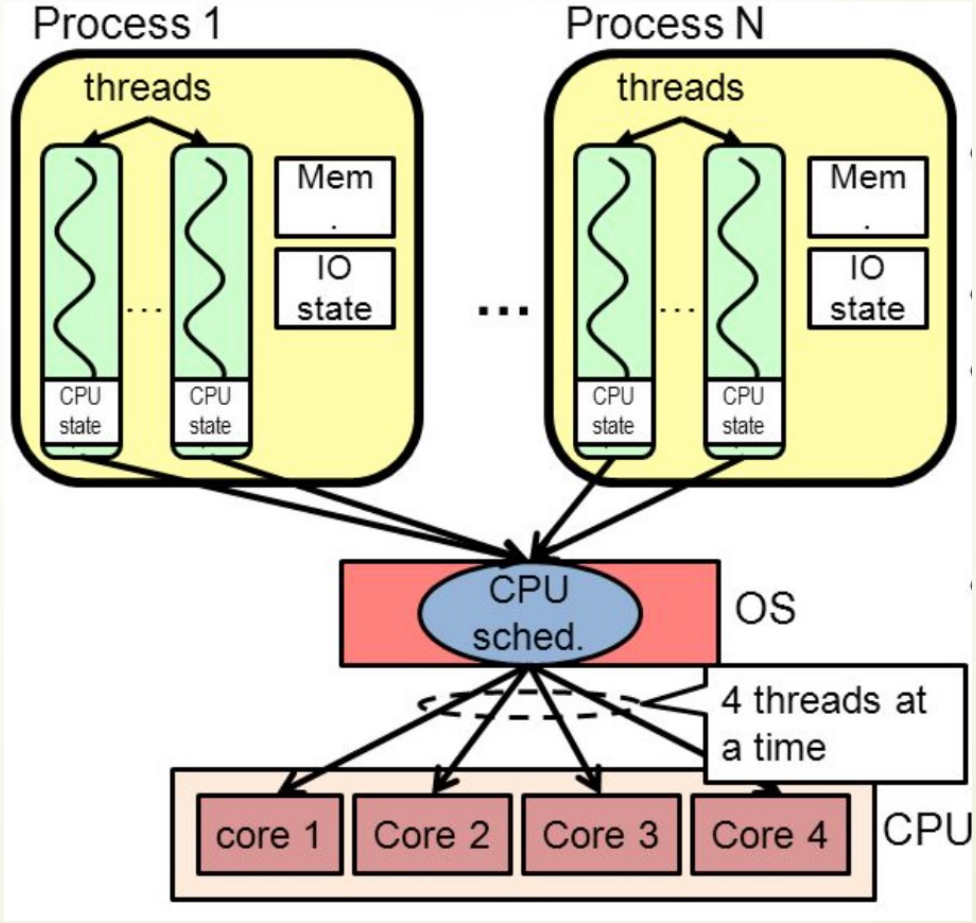
…in Windows?

4. PROCESOS
Procesos e hilos

- Un hilo es un flujo de ejecución a través del código del proceso con su propio contador de programa, registros del sistema y pila.
- Un hilo también se denomina **proceso ligero**
- *Cada hilo existe y pertenece exactamente a un proceso*



4. PROCESSES
Processes and threads



4. PROCESSES
Procesos e Hilos

Diferencias

Procesos	Hilos (Threads)
El proceso es pesado o requiere muchos recursos.	Thread es ligero, requiere menos recursos que un proceso.
La conmutación de procesos necesita interacción con el sistema operativo, ya que tiene un espacio de direcciones separado.	El cambio entre subprocesos en el mismo proceso es mucho más rápido porque el sistema operativo solo cambia los registros, no la asignación de memoria.
En múltiples entornos de procesamiento, cada proceso ejecuta el mismo código, pero tiene su propia memoria y recursos de archivo.	Todos los subprocesos pueden compartir el mismo código, datos y conjunto de archivos abiertos.
Si un proceso está bloqueado, ningún otro proceso puede ejecutarse hasta que se desbloquee el primer proceso.	Mientras un subproceso está bloqueado y en espera, el segundo subproceso de la misma tarea puede ejecutarse.
Múltiples procesos sin usar subprocesos usan más recursos.	Múltiples procesos con subprocesos utilizan menos recursos.
En múltiples procesos, cada proceso opera independientemente de los demás.	Un hilo puede leer, escribir o cambiar los datos de otro hilo.

4. PROCESOS
Procesos e hilos

VENTAJAS de hilos:

- Minimizar el **tiempo de cambio de contexto**.
- El uso de hilos proporciona **concurrencia** dentro de un proceso.
- Comunicación **eficiente**, ya que comparten memoria y recursos.
- **Economía**: más económico para crear y hacer el cambio de contexto, ya que el kernel no actúa.
- Utilización de **arquitecturas multiprocesador** a una mayor escala y eficiencia.

4. PROCESOS
Administrando procesos

- El sistema operativo es responsable de **crear** y **ejecutar** procesos.
- Siempre se crea un **nuevo proceso** cuando es requerido **por otro proceso** o por el **usuario**.
  - Cuando se inicia el sistema operativo y se carga el kernel, se crea el proceso inicial del sistema y luego se crean jerárquicamente los otros procesos: padres, hijos, abuelos, etc.
  - *Cuando se crea un proceso, se le da un **PID (Process Identifier**)*

Práctica:
Gestión de Procesos en Linux

Lea el documento y realice las tareas indicadas.

5. PROGRAMACIÓN MULTIPROCESO EN JAVA

- Como sabemos, el sistema operativo se encarga de proporcionar multiprogramación.
- Si los procesos cooperan, el desarrollador debe implementar toda la **comunicación** y **sincronización**.
- Escribir una aplicación de múltiples procesos generalmente implica los siguientes pasos:
  - Descomposición funcional: identificar diferentes tareas .
  - Partición: asignar cada tarea a un proceso.
  - Implementación: programar la aplicación.
    - *Vamos a utilizar la Máquina Virtual Java multiplataforma.*
    - *Java solo permite métodos muy simples para implementar la comunicación y sincronización entre procesos.*

5. PROGRAMACIÓN MULTI-PROCESO EN JAVA
Creando procesos en Java

- Class ***Process*** → proporciona métodos para:
  - Obtener entrada desde el proceso hijo,
  - Enviar la salida al proceso hijo,
  - Esperar hasta que la ejecución del hijo finalice,
  - Comprobar el estado de salida del proceso hijo,
  - Destruir (kill) el proceso hijo.

- *El subproyeso no tiene su propio terminal o consola y toda la comunicación con el padre se realiza a través de* ***flujos de datos***.

5. PROGRAMACION MULTIPROCESO EN JAVA
Creando un proceso en Java

```
import java.io.IOException;
import java.util.Arrays;


public class ProcessCreating  {

    public static void main(String[] args) throws IOException {
        if (args.length <= 0) {
            System.err.println("I need a command/program to execute!!");
            System.exit(-1);
        }

        ProcessBuilder pb = new ProcessBuilder(args);
        try {
            Process process = pb.start();
            int exitValue = process.waitFor();
            System.out.println("Execution of " +
            Arrays.toString(args) + "returns exit value: " + exitValue);
        }
        catch (IOException ex) {
            System.err.println("IO Exception !!");
            System.exit(-1);
        }
        catch (InterruptedException ex) {
            System.err.println("The child process exited with error");
            System.exit(-1);
        }

    }
}
```

5. PROGRAMACION MULTIPROCESO EN JAVA Destruyendo un proceso en Java

```
import java.io.IOException;

public class ProcessDestroy {

    public static void main(String[] args) throws IOException {
        if (args.length <= 0) {
            System.err.println("I need a command/program to execute!!");
            System.exit(-1);
        }

        Runtime runtime = Runtime.getRuntime();

        try {
            Process process = runtime.exec(args);
            process.destroy();
        }
        catch (IOException ex) {
            System.err.println("IO Exception !!");
            System.exit(-1);
        }
    }
}
```

5. PROGRAMACION MULTIPROCESO EN JAVA
Comunicación de Procesos

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Arrays;

public class ProcessCommunication {

    public static void main(String[] args) throws IOException {

        Process process = new ProcessBuilder(args).start();
        InputStream inStream = process.getInputStream();

        InputStreamReader inStreamReader = new InputStreamReader(inStream);
        BufferedReader bufReader = new BufferedReader(inStreamReader);


        System.out.println("Ouput for the child process "
                + Arrays.toString(args) + " : ");

        String line;
        while ((line = bufReader.readLine()) != null) {
            System.out.println(line);
        }
    }
}
```

5. PROGRAMACION MULTIPROCESO EN JAVA
Communicating processes

- La salida del proceso **ls –la** puede ser:

Output - ProcessCommunication (run)

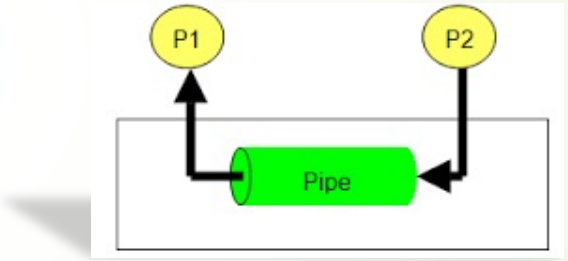
```
run:
Ouput for the child process [ls, -la] :
total 16
drwxr-xr-x  7 empar  staff   238 Aug 26 17:40 .
drwxr-xr-x  6 empar  staff   204 Aug 26 17:34 ..
drwxr-xr-x  3 empar  staff   102 Aug 26 17:40 build
-rw-r--r--  1 empar  staff  3681 Aug 26 17:34 build.xml
-rw-r--r--  1 empar  staff    82 Aug 26 17:34 manifest.mf
drwxr-xr-x  7 empar  staff   238 Aug 26 17:34 nbproject
drwxr-xr-x  3 empar  staff   102 Aug 26 17:34 src
BUILD SUCCESSFUL (total time: 1 second)
```

5. PROGRAMACION MULTIPROCESO EN JAVA
Sincronización

- El proceso primario puede bloquearse a sí mismo y esperar a que finalice el proceso hijo.
- El hijo devuelve un valor de salida con el resultado de la ejecución.

```
import java.util.Arrays;
import java.io.IOException;

public class ProcessSynchronize {

    public static void main(String[] args) {
        try {
            Process process = new ProcessBuilder(args).start();
            int exitValue = process.waitFor();

            System.out.println("The exit value for " + Arrays.toString(args)
                               + " is: " + exitValue);
        } catch (IOException ex) {
            System.out.println("There was an error while running "
                               + Arrays.toString(args)
                               + " --> " + ex.getMessage());
        } catch (InterruptedException ex) {
            System.out.println("The current process was interrupted -->"
                               + ex.getMessage());
        }

    }
}
```

Actividad:
Redirección de la I/O: piping



Lee la descripción de esta actividad en la documentación de la unidad y realiza las tareas indicadas.