Sistemas Operativos 1

Procesos

Hilos

Edwin Salvador

5 de noviembre de 2015

Sesión 6

Contenido I

Contenido I

Programación concurrente

- ¿Qué es la programación concurrente?
- Varios procesos a la vez.
- ¿Podemos tener concurrencia en un computador con que cuenta con un solo CPU?
- Si, gracias a la multiprogramación (multitarea), tenemos procesos que cooperan entre si.
- Los problemas de la concurrencia: sincronización e interbloqueos empezaron con los primeros SO.

Contenido I

Programa vs Proceso

- ¿Cuál es la diferencia entre un programa y un proceso?
- Un programa es un set de instrucciones de alto nivel, un proceso solo se crea cuando un programa empieza a correr.
- Si un programa es **secuencial**, entonces tendremos un solo proceso.
- Si tenemos un programa **concurrente**, tendremos multiples procesos.

Proceso

Un proceso consiste de tres segmentos en memoria:

- **Código** Las instrucciones de máquina en memoria que ejecuta el proceso.
- **Datos** La memoria utilizada por las variables estáticas globales y la memoria asignada en tiempo de ejecución del programa.
- **Stack de ejecución** Son las variables locales y las llamadas a funciones. Cada proceso tiene su propio stack.

Threads

- ¿Qué es un thread o hilo de un proceso?
- Los threads son procesos que comparten el mismo espacio de memoria (código y datos).
- Los threads de un proceso comparten el mismo espacio de memoria pero tienen su propio stack.

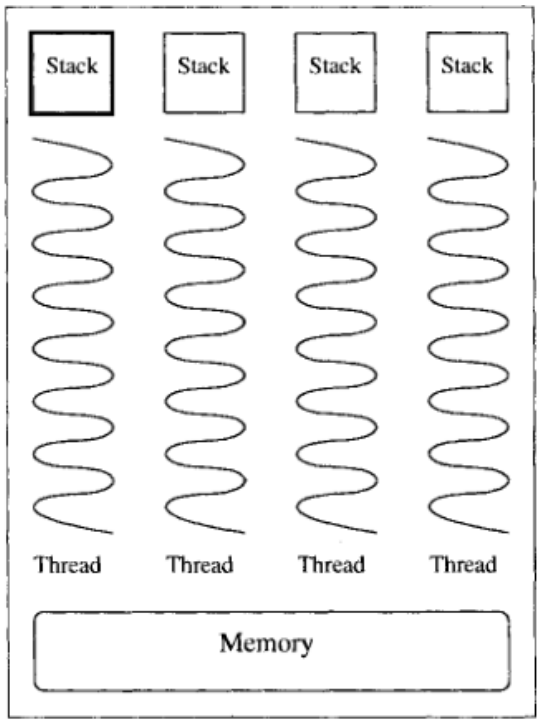


Figura: Un proceso con 4 hilos

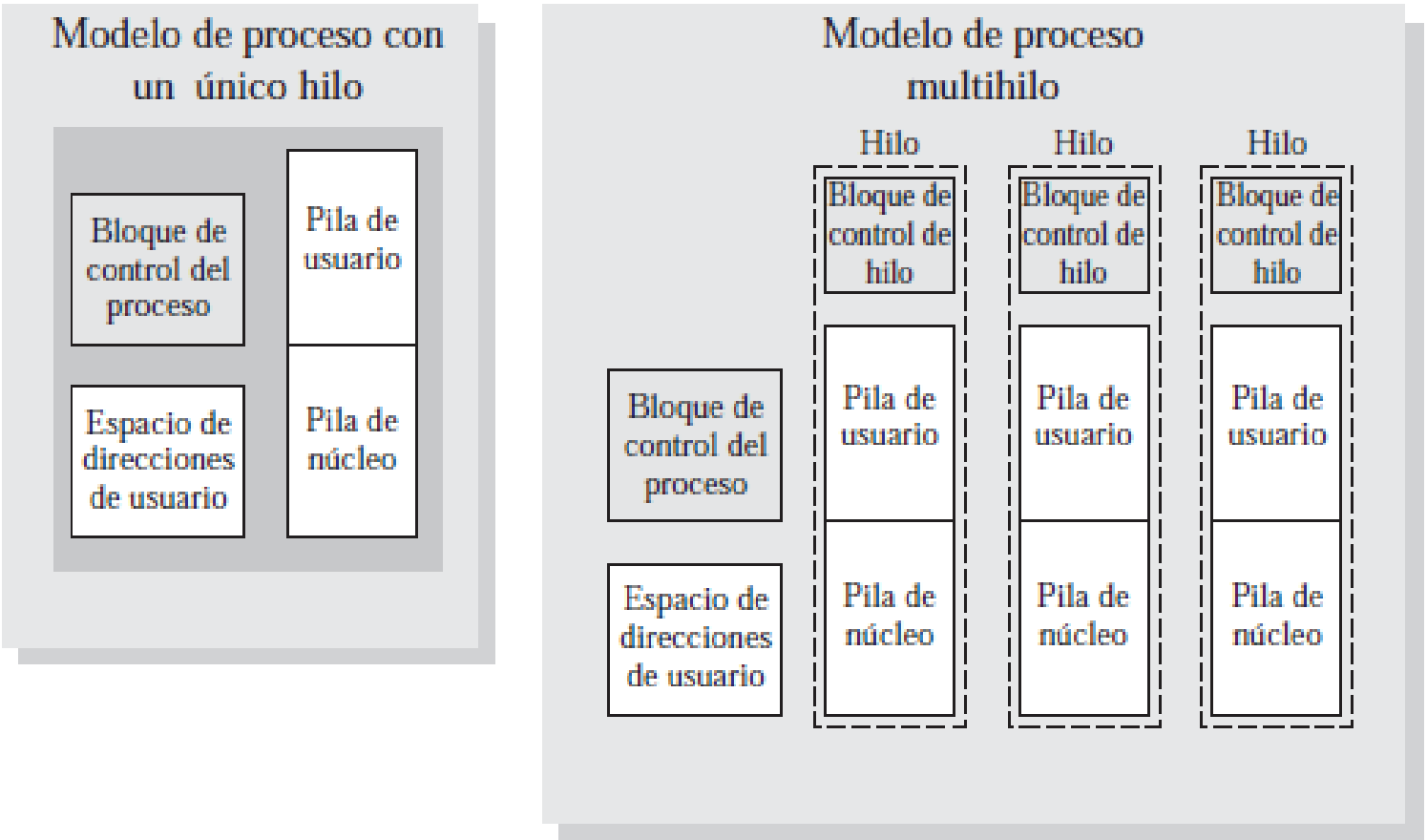
Multihilo

- Multihilo es la capacidad de un SO de dar soporte a múltiples hilos de ejecución en un solo proceso.
- No todos los SO soportan multihilo. Ejemplo:
  - MS-DOS = 1 proceso de usuario y 1 único hilo.
  - Otros SO y algunas versiones de UNIX = múltiples procesos de usuario, pero 1 solo hilo.
- ¿Cuál es la diferencia entre varios procesos de un solo hilo y un proceso de varios hilos?
- Windows, Solaris, Mac OS X, Linux entre otros soportan múltiples procesos y cada uno con múltiples hilos.

Multihilo

- En un entorno multihilo, un proceso es una unidad de asignación de recursos y una unidad de protección.
- Dentro de un proceso puede haber uno o más hilos, cada uno con:
  - Un estado de ejecución,
  - Un contexto de hilo (que se almacena cuando el hilo no está en ejecución).
  - Una pila de ejecución.
  - Espacio de almacenamiento para variables locales.
  - Acceso a la memoria y recursos de su proceso, compartido con todos los hilos de su mismo proceso.

Diferencia entre proceso e hilo



- Multihilo: existe un solo BCP y un espacio de direcciones de usuario para el proceso, pero hay pilas separadas para cada hilo, y un BCH para cada hilo (registros, prioridad, etc).

Multihilo

- Todos los hilos de un proceso:
  - comparten el estado y los recursos de ese proceso.
  - residen en el mismo espacio de direcciones
  - tienen acceso a los mismos datos.

Beneficios de los hilos

Los mayores beneficios provienen de las consecuencias del rendimiento:

- Es más rápido crear un nuevo hilo en un proceso que crear un nuevo proceso (10-100 veces más rápido).
- Un hilo se finaliza más rápido que un proceso.
- Es más rápido cambiar entre dos hilos del **mismo** proceso.
- Mejoran la eficiencia de la comunicación entre diferentes programas.

Hilos

- Entonces, si se desea crear una aplicación como un conjunto de unidades de ejecución relacionadas, es mucho más eficiente hacerlo con un conjunto de hilos que con un conjunto de procesos independientes.
- Un ejemplo de aplicación que puede hacer uso de hilos es un servidor de archivos. Cada vez que se realiza una petición de archivos, el gestor de archivos puede ejecutar un nuevo hilo. Así en un ambiente **multiprocesador** se pueden ejecutar simultáneamente múltiples hilos del mismo proceso en diferentes procesadores. Y los hilos pueden compartir archivos de datos y coordinar acciones de manera más rápida.
- ¿Ejemplos? Hacer grupos y escribir la mayor cantidad de ejemplos sobre programas que utilizan hilos durante su ejecución.

¿Cómo se utilizan los hilos en un sistema multiprocesador?

- **Trabajo en primer plano y en segundo plano**: Un programa de hojas de cálculo (Excel) puede utilizar un hilo para mostrar menús mientras otro hilo ejecuta los mandatos de usuario y actualiza la hoja de cálculo. Así se puede empezar un nuevo mandato antes de terminar el anterior y se percibe una mayor velocidad de la aplicación.
- **Procesamiento asíncrono:** Un procesador de texto (Word) utiliza **tareas asíncronas** que pueden implementarse como hilos. Se puede crear un hilo cuyo único trabajo sea crear una copia de seguridad cada minuto y guardarla en disco.
- **Velocidad de ejecución:** un proceso multihilo puede computar una serie de datos mientras lee los siguientes datos de un dispositivo. Así mientras un hilo está bloqueado por una operación de E/S, otro hilo puede estar ejecutando.
- **Estructura modular de programas:** Los programas con varias tareas o con varias fuentes y destinos de E/S se pueden implementar más fácilmente usando hilos.

Hilos

- Debido a que todos los hilos de un proceso comparten el mismo espacio de direcciones que su proceso, al suspender el proceso, todos los hilos se suspenden al mismo tiempo. De igual manera cuando se finaliza un proceso, se finalizan todos los hilos de ese proceso.

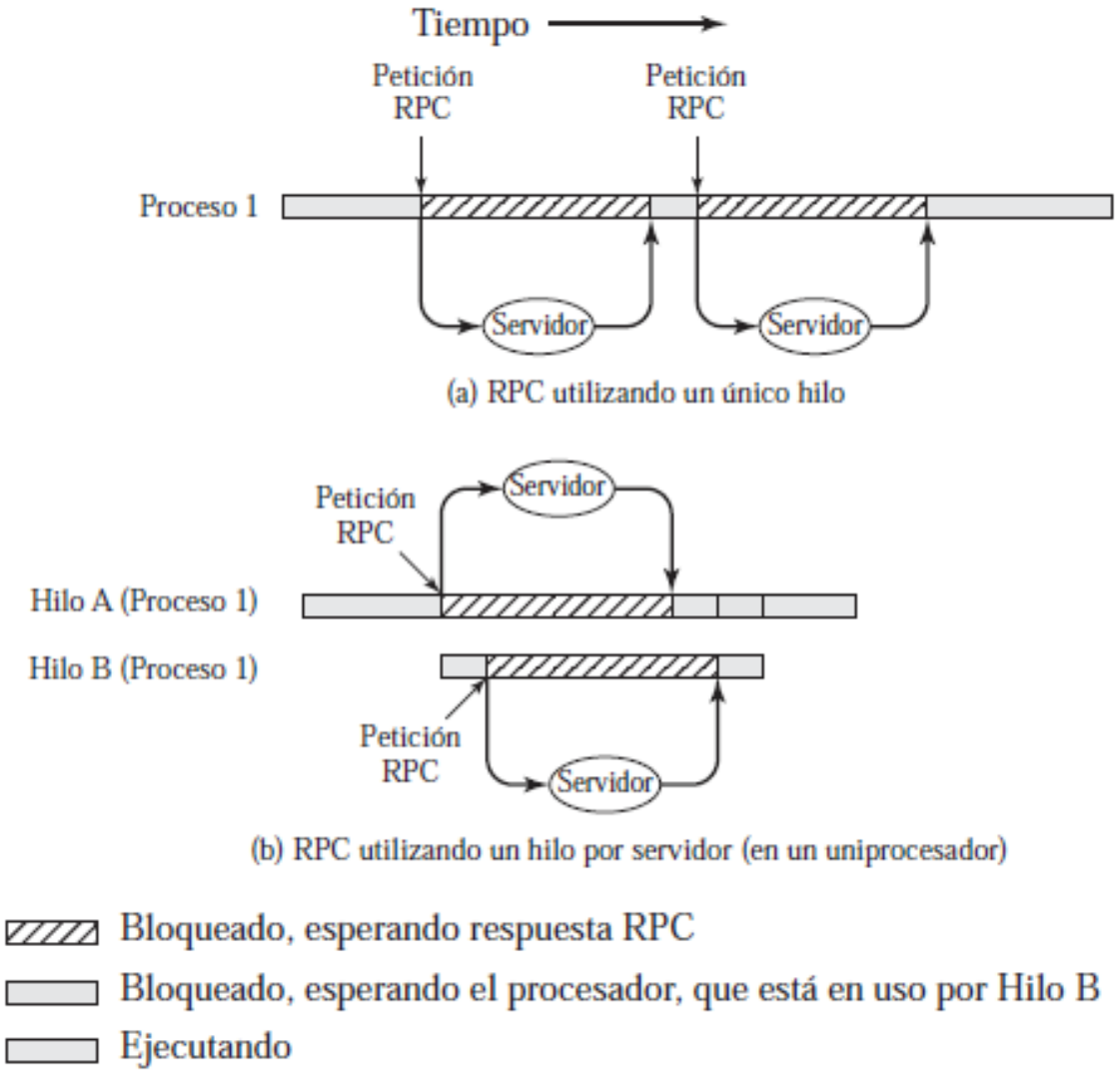
Contenido I

Funcionalidad de los Hilos

Estados de los hilos

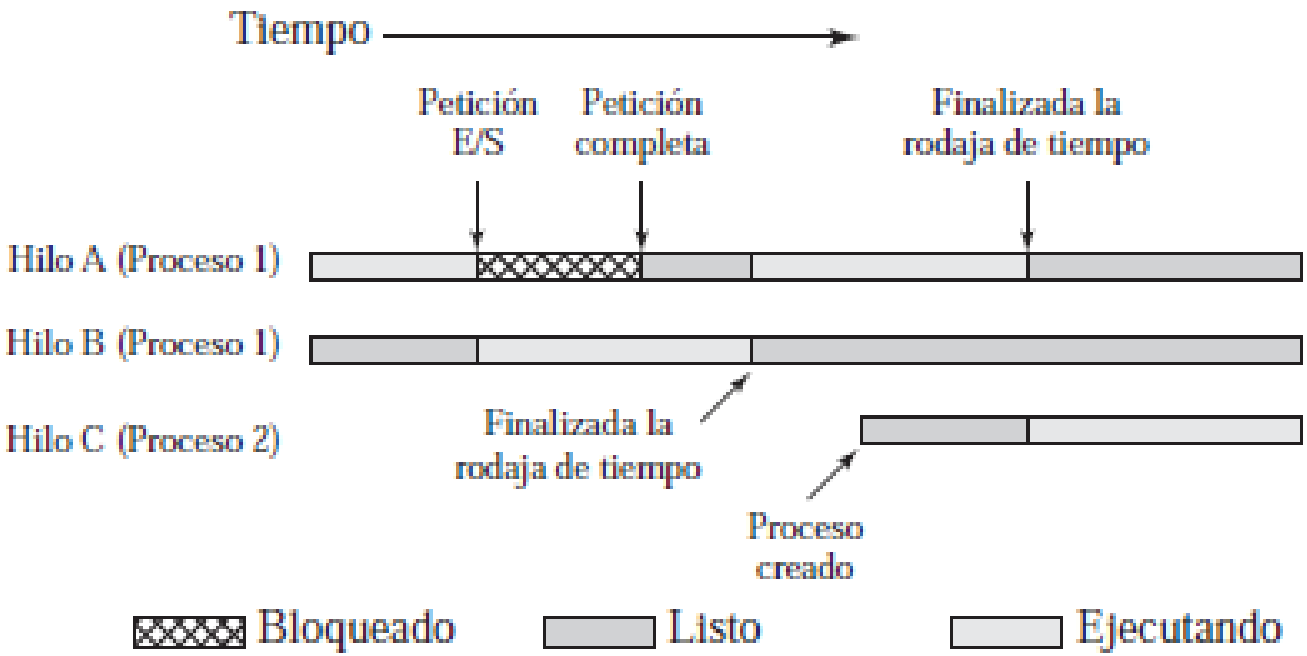
- Los principales estados de los hilos son Ejecutando, Listo y Bloqueado.
- El estado Suspendido se aplica solo a nivel proceso ya que si se expulsa un proceso de memoria, todos sus hilos deben también ser expulsados. Porqué? porque comparten el mismo espacio de direcciones.

Monohilo vs Multihilo



- Programa que realiza 2 RPC a dos máquinas diferentes y poder combinar resultados.
- en a los resultados se obtienen en secuencia.
- en b se mejora la velocidad de ejecución del programa.

Multihilo en uniprocesador con multiprogramación



- Multiprogramación permite **intercalado** de hilos de varios procesos pero nunca se llegan a ejecutar en paralelo.

Sincronización de hilos

- Debido a que todos los hilos de un proceso comparten recursos y espacio de direcciones, cualquier modificación de un recurso (archivo) por parte de un hilo, afecta el entorno de todos los hilos del mismo proceso.
- Por este motivo es necesario sincronizar las actividades de los hilos para que no interfieran entre ellos o corrompan estructuras de datos.

Interferencia entre Hilos

- Ejecutar el ejemplo Counter.java que está dentro del paquete de ejemplos y observar que los resultados.
- Se puede observar que los resultados son impredecibles. A que se debe esto?
- Una razón para este comportamiento es:
  - Thread A: Retrieve c.
  - Thread B: Retrieve c.
  - Thread A: Increment retrieved value; result is 1.
  - Thread B: Decrement retrieved value; result is -1.
  - Thread A: Store result in c; c is now 1.
  - Thread B: Store result in c; c is now -1.

Ejercicio

- Considere la siguiente clase:

```
class Schedule  {
  static int x  =  0;
  static int y  =  0;
  public static int op1(){x  =  1;  return y;)
  public static int op2(){y  =  2; return  3*x;}
}
```

Si un thread llama a op1 y otro thread llama a op2, entonces que valores podrían ser devueltos por op1 y op2 tomando en cuenta que los hilos se ejecutan concurentemente?

Contenido I

Hilos de nivel de usuario y de nivel núcleo

- **Hilos de nivel usuario (ULT):** la aplicación gestiona todo el trabajo de los hilos y el núcleo no es consciente de la existencia de los mismos. Se utilizan bibliotecas de hilos para gestionarlos (creación, suspensión, destrucción).
- **Hilos de nivel núcleo (KLT)**: El núcleo realiza todo el trabajo de gestión de hilos. No hay código de gestión de hilos en la aplicación, solamente una API para acceder a las utilidades de hilos del núcleo. Windows utiliza este enfoque.

Contenido I

Programación Concurrente en UNIX

- En UNIX, los procesos son organizados a manera de árbol donde cada proceso tiene su propio PID.
- UNIX proporciona las llamadas al sistema `fork` y `wait` para la creación y sincronización de procesos.
- **`fork`** cuando un proceso ejecuta una llamada `fork`, se crea un hijo con una copia del espcio de memoria del proceso padre. La única diferencia es el valor de retorno de la llamada `fork`.
  El proceso padre obtiene el PID del hijo como valor de retorno de `fork`, en cambio el hijo obtiene 0 como el valor de retorno.

```
pid = fork();
if (pid == 0) {
  // child process
  cout << "child process";
}
else {
  // parent process
  cout << "parent process";
}
```

`wait`

- La llamada `wait` la utiliza el proceso padre para espera la terminación de un proceso hijo.
- Un proceso termina cuando ejecuta la última instrucción del código o ejecuta una llamada del sistema `exit`.
- Cuando un hijo termina su ejecución, si el proceso padre está esperando, entonces es restaurado y recive el PID del proceso hijo que terminó desde la llamada `wait`.

Contenido I

Clase Thread

- Java proporciona la clase predefinida `Thread`.
- Una clase puede extender la clase `Thread`, sobreescribir el método `run` y luego llamar a `start()` para lanzar el thread.

```
public class HelloWorldThread extends Thread  {
  public void run()  {
    System.out.println("Hello World");
  }
  public static void main( String[] args)  {
    HelloWorldThread t = new HelloWorldThread();
    t.start();
  }
}
```

Ejemplo del uso de Hilos

- ¿Cuál es el algoritmo para calcular el enésimo número de la serie de Fibonacci F_n?
- Podemos utilizar la relación recursiva:

$$F_n = F_{n-1} + F_{n-2}$$

Para $n \geq 2$. Los casos bases son:

$$F_0 = 1$$

y

$$F_1 = 1$$

Ejercicio

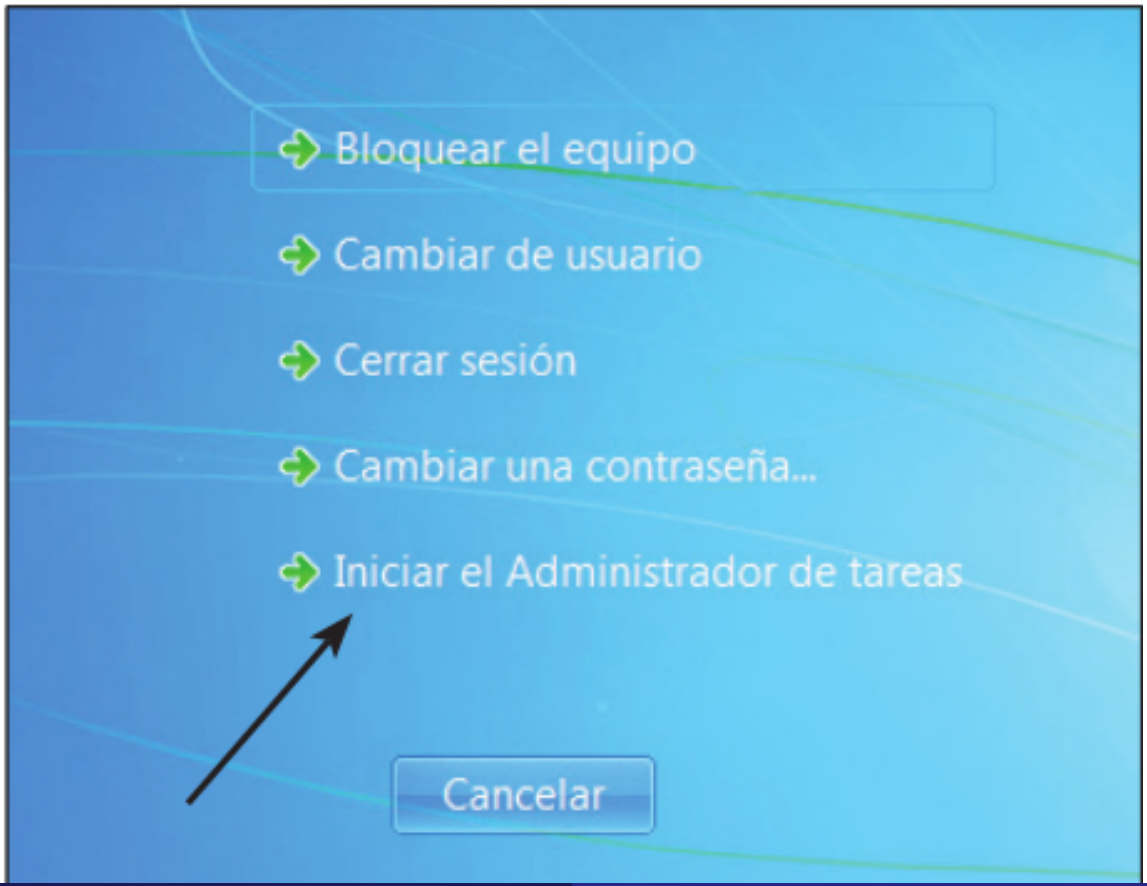
Threads y join

- Podemos implementar el programa anterior utilizando `threads` y el constructor `join`.
- Para computar F_n, el método `run` se bifurca (`fork`) en dos threads que computan $F_n - 1$ y $F_n - 2$ recursivamente.
- El thread principal espera a que finalicen estos dos threads utilizando la función `join`.

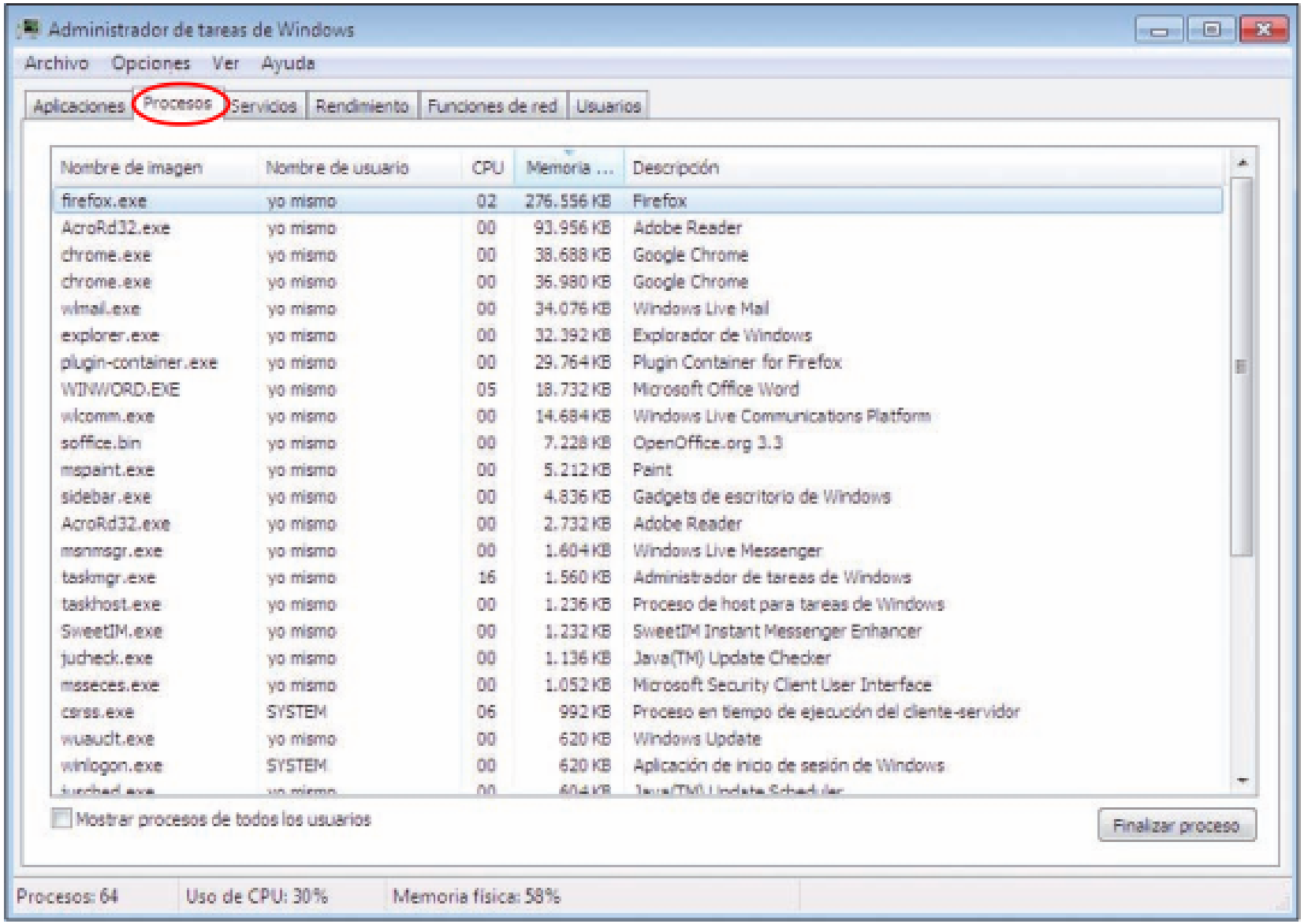
Contenido I

Acceder a procesos en Windows 7

- ¿Cómo accedemos a ver los procesos ejecutándose en Windows 7? `ctrl + alt + supr` **Administrador de tareas**.
- ¿Otras opciones en Windows 7?
- ¿Opciones en Windows 8?



Procesos en Windows

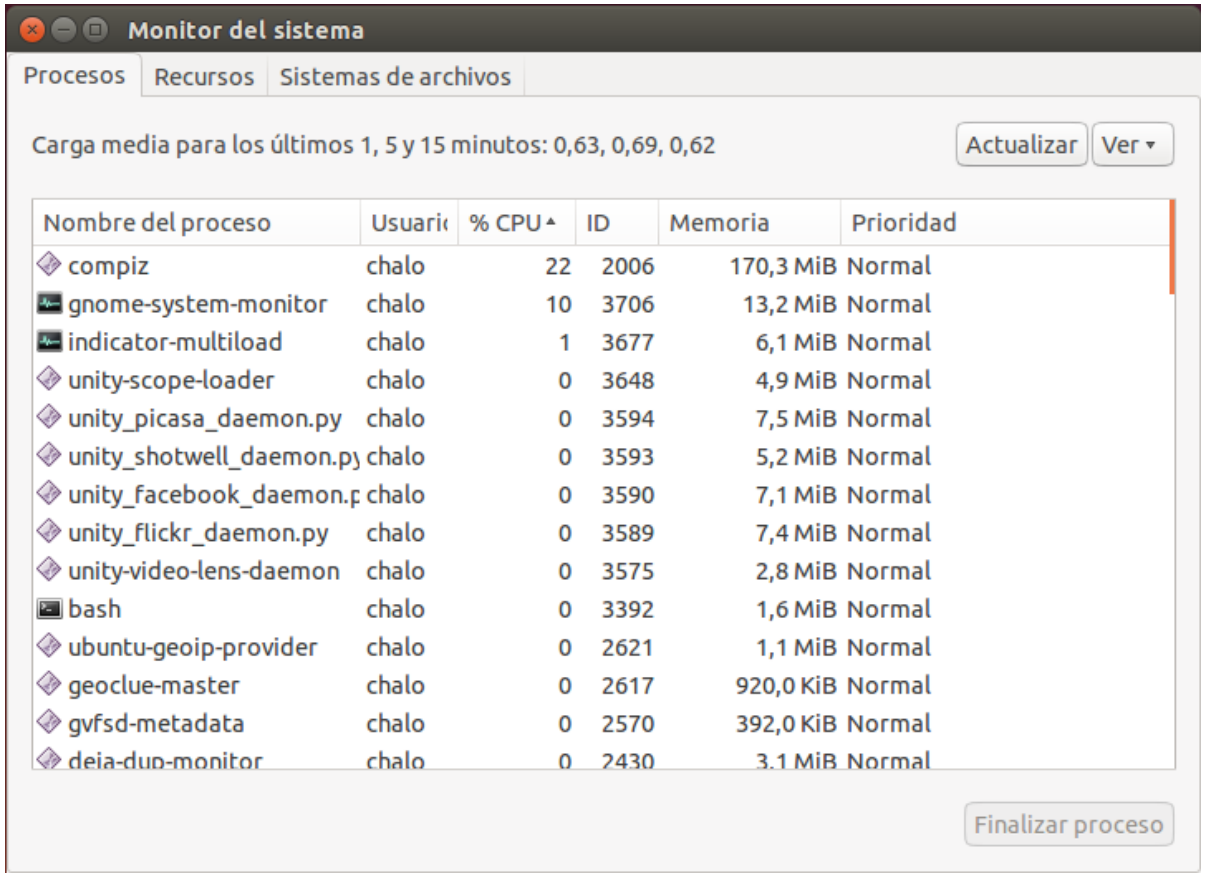


Procesos en Windows

- La pestaña de procesos nos mostrará:
  - los procesos en ejecución.
  - el usuario propietario
  - el trabajo del procesador
  - la cantidad de memoria
  - descripción
  - clic derecho en la cabecera permite seleccionar que aspectos mostrar.
  - Total de procesos activos (inferior izquierda)
  - clic derecho sobre proceso mostrará:
    - Terminar proceso
    - Establecer prioridad
    - Buscar en línea
    - Propiedades

Procesos en Ubuntu

- ¿Cómo abrimos el administrador de procesos en Ubuntu? Varias opciones: Información similar que en Win.
  - Buscar: Monitor del sistema
  - Buscar/Icono de aplicaciones/monitor del sistema
  - Instalar "indicador de carga del sistema"



Procesos en Ubuntu en la línea de comandos

- ps
- top
- htop
- kill
- renice

`http://ss64.com/bash/` lista de comandos.

Contenido I

Trabajo en clase

Revisar el documento “Aprendiendo Linux (100 Ejercicios).html”

Deber

Responder las preguntas del archivo “100 Ejercicios Linux.pdf”

Entrega 19 de noviembre 2015