Universidad de los Andes

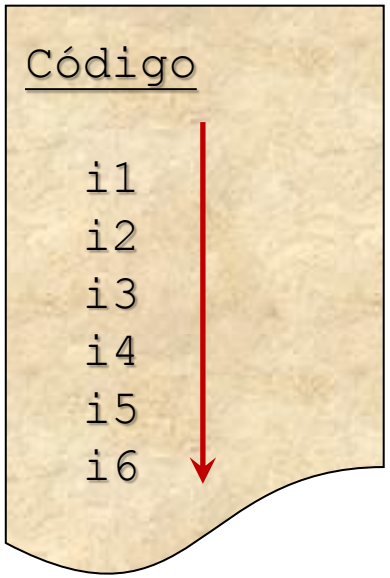
Infraestructura computacional

Concurrencia

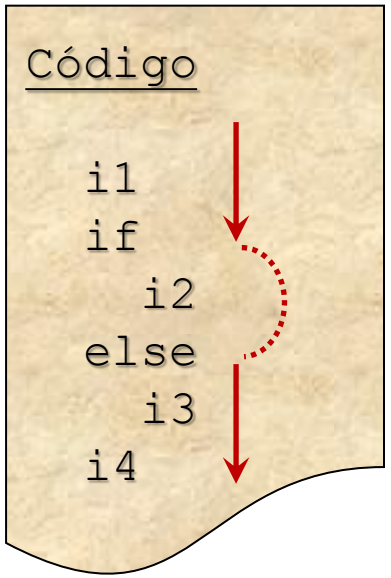
Universidad de los Andes

¿Qué es concurrencia?

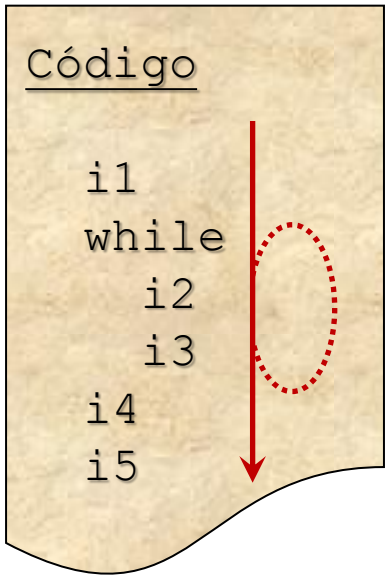
Ejecución secuencial



Ejecución condicional



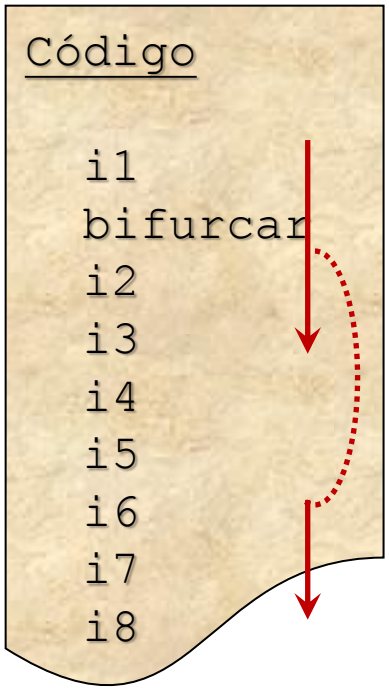
Ejecución iterativa



Universidad de los Andes

¿Qué es concurrencia?

Ejecución concurrente



Universidad de los Andes

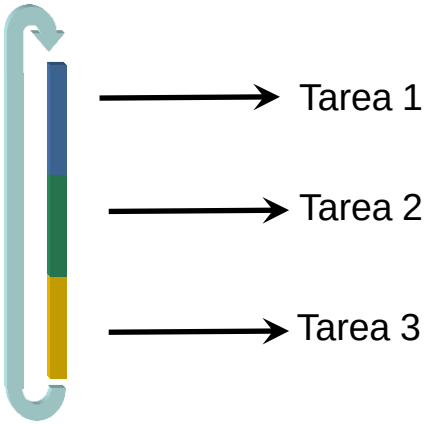
¿Por qué concurrencia?

- Facilita modelado de tareas diferentes pero simultáneas
  - Mejor encapsulamiento
  - Mejor separación de problemas:
- Mayor reactividad
- Mejor desempeño
- Más adecuado para ambientes modernos (red, GUI, multitarea)

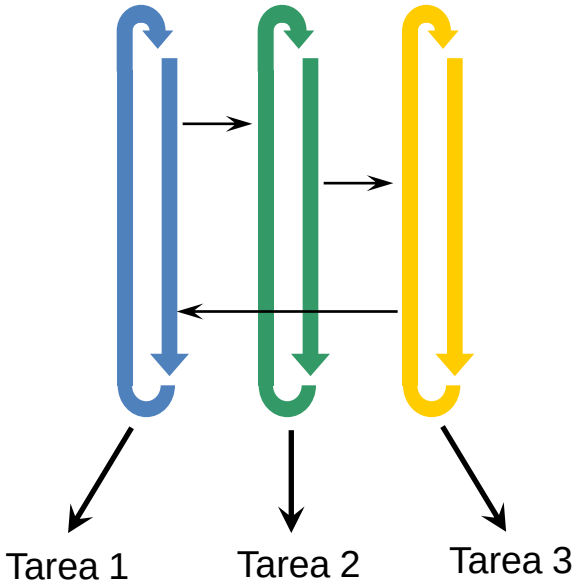
Universidad de los Andes

¿Por qué concurrencia?

Programación monolítica



Programación concurrente



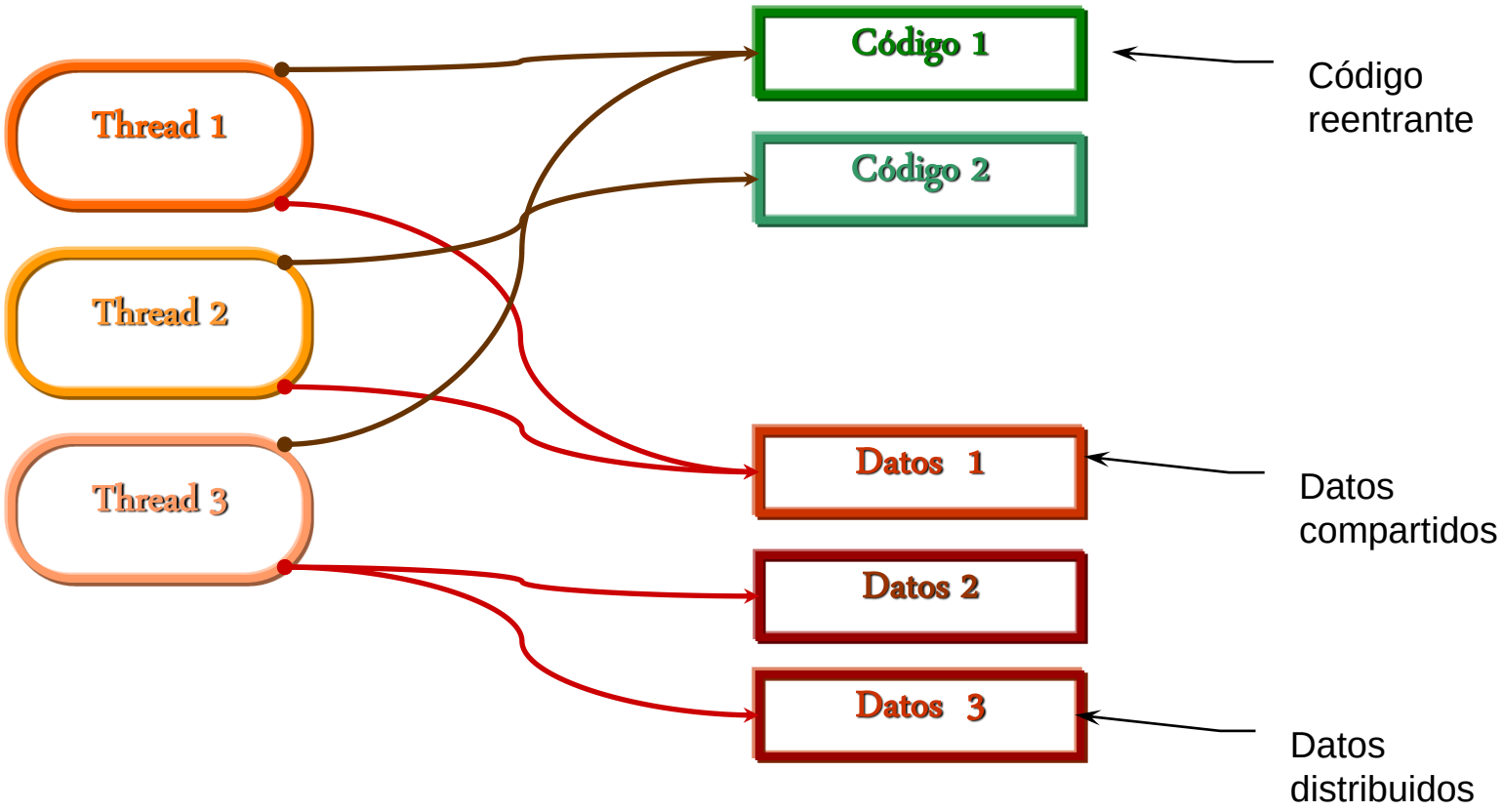
Universidad de los Andes

Niveles de concurrencia

- Circuitos
- Instrucciones (ILP)
- Hilos (*threads*) de un proceso
- Procesos locales
- Procesos remotos

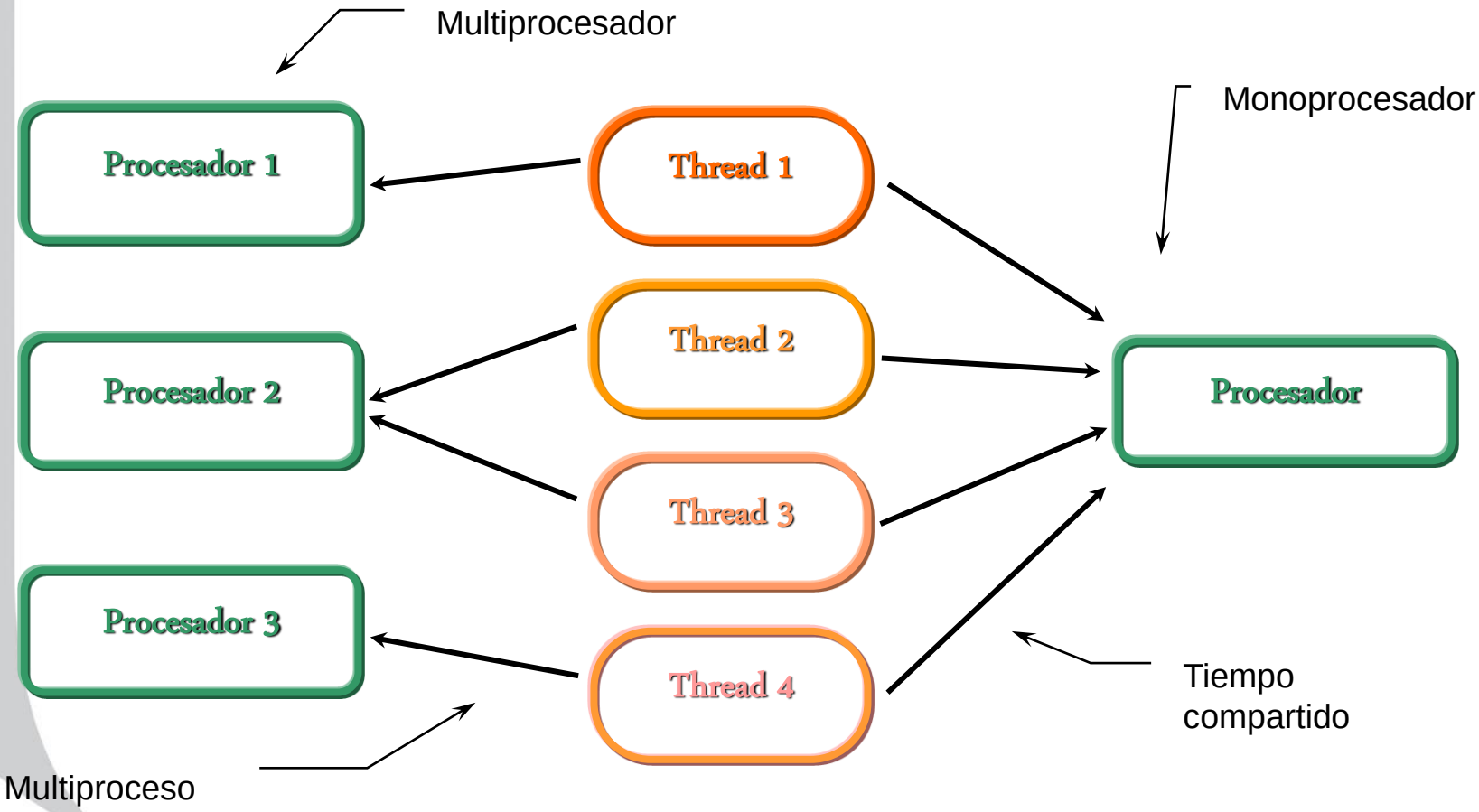
Universidad de los Andes

¿Cómo es la concurrencia?



Universidad de los Andes

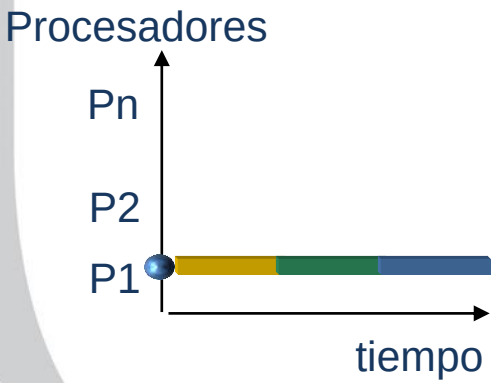
¿Cómo es la concurrencia?

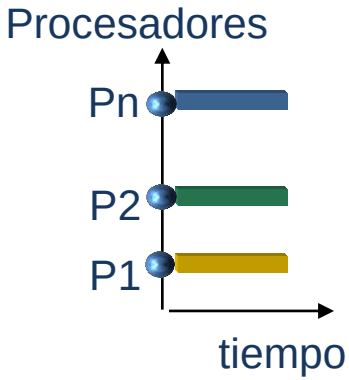


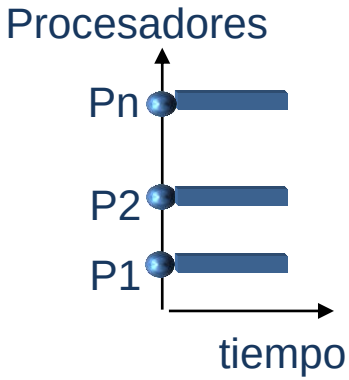
Universidad de los Andes

Tipos de concurrencia

- Multiprogramación
- Multitarea (tiempo compartido)
  - S.O. apropiativo (*preemptive*)
- Multiproceso
- Paralelismo





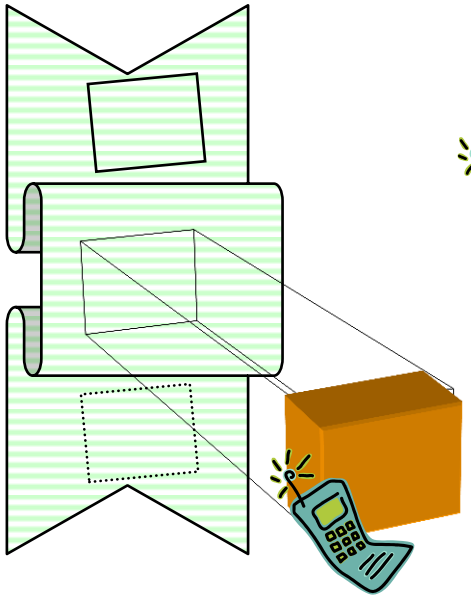


Universidad de los Andes

¿Cómo expresar la concurrencia?

Programa

¿Cómo sincronizar procesos?



¿Cómo generar procesos?

¿Cómo comunicar procesos?

Proceso

Universidad de los Andes

¿Cómo expresar la concurrencia?

- Incorporar construcciones específicas a un lenguaje
  - Compilador nuevo
  - Requiere entrenamiento
  - Puede no ser “natural”
- Directivas al compilador
  - Programas paralelos a partir de programas secuenciales
  - Restrictivo
- Usar un API
  - Solo se necesita disponer de la librería respectiva
  - Programación poco, o nada, estructurada

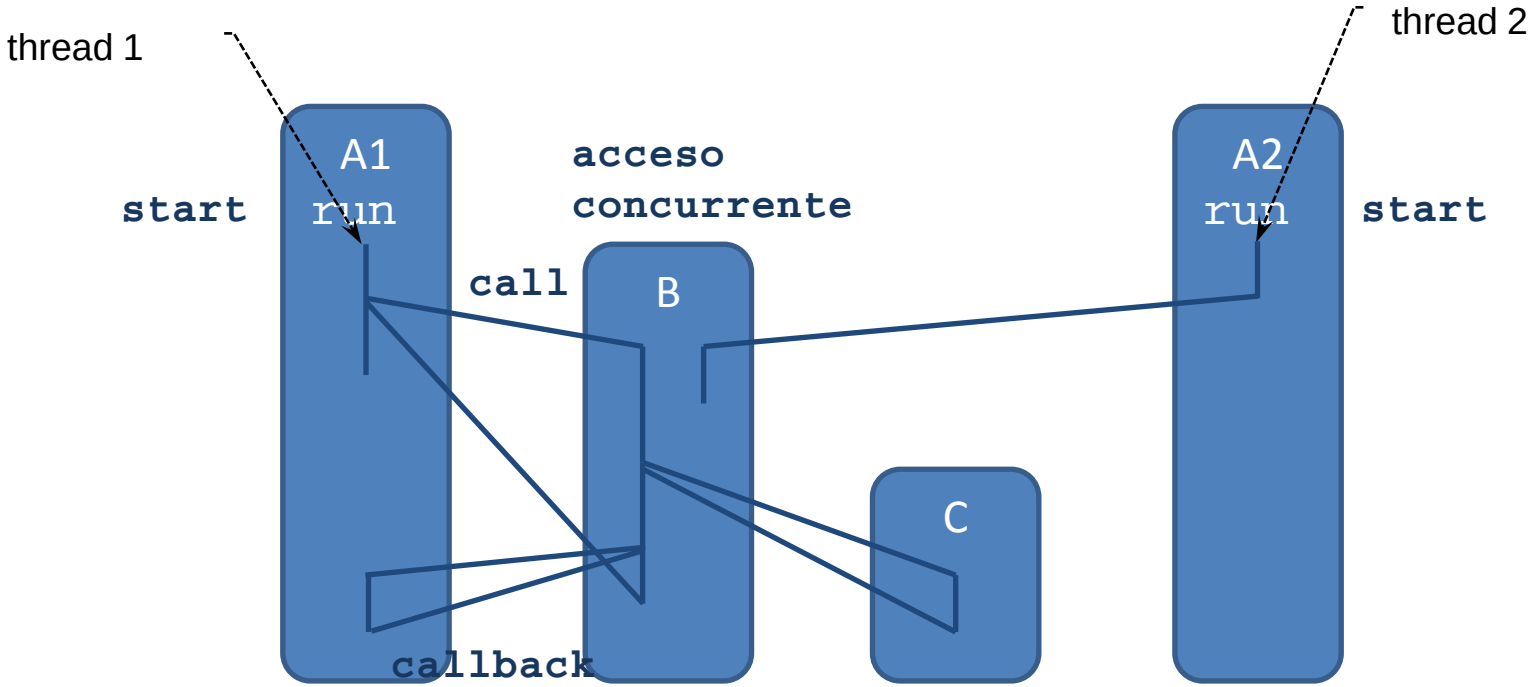
Universidad de los Andes

Concurrencia en Java

- Basada en threads
- Threads de lenguaje
- Creación de threads:
  - Los threads son representados por objetos
  - … por ende existe una clase **`Thread`**
  - Los threads del programador son clases derivadas de `Thread`
  - La ejecución empieza en el método **`run`**
  - El `new` no activa al thread, solo crea al objeto
  - Para activar un thread (para que empiece a ejecutar su `run`) se invoca el método **`start`**
  - También se puede implementar la interfaz **`Runnable`**
  - Un thread no es un objeto: es un flujo de control sobre el código

Universidad de los Andes

Concurrencia en Java



Universidad de los Andes

Concurrencia en Java

Declaración

```
public class T extends Thread {
   …
   public void run( ) {
      acciones del thread
   }
}
```

Creación y activación

```
T t = new T( );   //Creación
   …
t.start( );   //Activación
```

Universidad de los Andes

Concurrencia en Java

Generar n threads con identificación

```
public class T extends Thread {
  private int id;

  public T ( int n ) {
    id = n;
  }
  …
  public void run( ) {
    puede usar id
  }
  …
for ( i = 0; i < nThreads; i++) {
    new T( i ).start( );
}
```

Universidad de los Andes

Concurrencia en Java

Buscar un valor en una matriz

```
public class T extends Thread {
  private static int valor;
  private static int n;
  private static int [][] M;

  private int id;

  public T ( int i ) {
    id = i;
  }

  private static void inicializar( ) {
    inicializa valor, n, M
  }
```

Universidad de los Andes

Concurrencia en Java

Buscar un valor en una matriz

```
public void run (){
  int nElementos = M[id].length;

  for ( int j = 0; j < nElementos; j++) {
    if ( M[id][j] == valor ) {
      System.out.println( id );
    }
  }
}
public static void main(String[] args) {
  inicializar( );
  for ( int i = 0; i < n; i++)
    new T( i ).start();
}
```

Universidad de los Andes

Concurrencia en Java

Interfaz runnable

```
public class R implements Runnable{
   …
   public void run( ) {
      acciones del thread
   }
}
```

Creación y activación

```
R r = new T( );  //Creación del objeto
Thread t = new Thread( r );  //Creación del thread
   …
t.start( );  //Activación
```