Procesos, Concurrencia

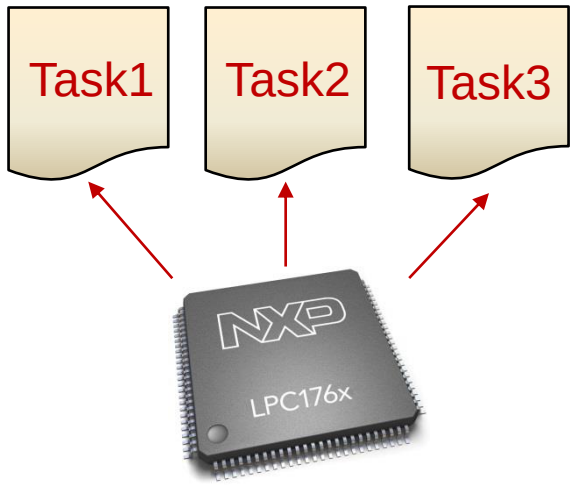
Ing. Eddie Sobrado Malpartida

Objetivos

- Repasaremos los principios de la programación concurrente
- Analizaremos dos alternativas de ejecución de procesos concurrentes
- Veremos cómo crear procesos concurrentes en C/ POSIX

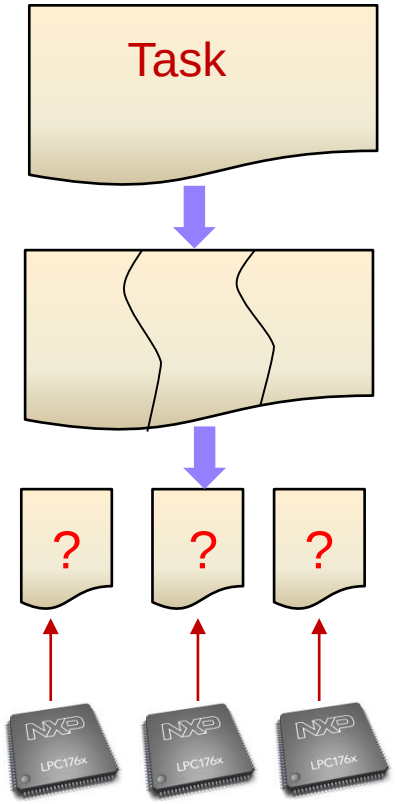
Programación Concurrente

- Programación concurrente es el nombre dado a las técnicas y notación de programación para expresar "paralelismo" potencial y resolver la sincronización y los problemas de comunicación.
- En la programación concurrente solo se cuenta con un procesador. El tiempo de CPU se reparte entre varios procesos.



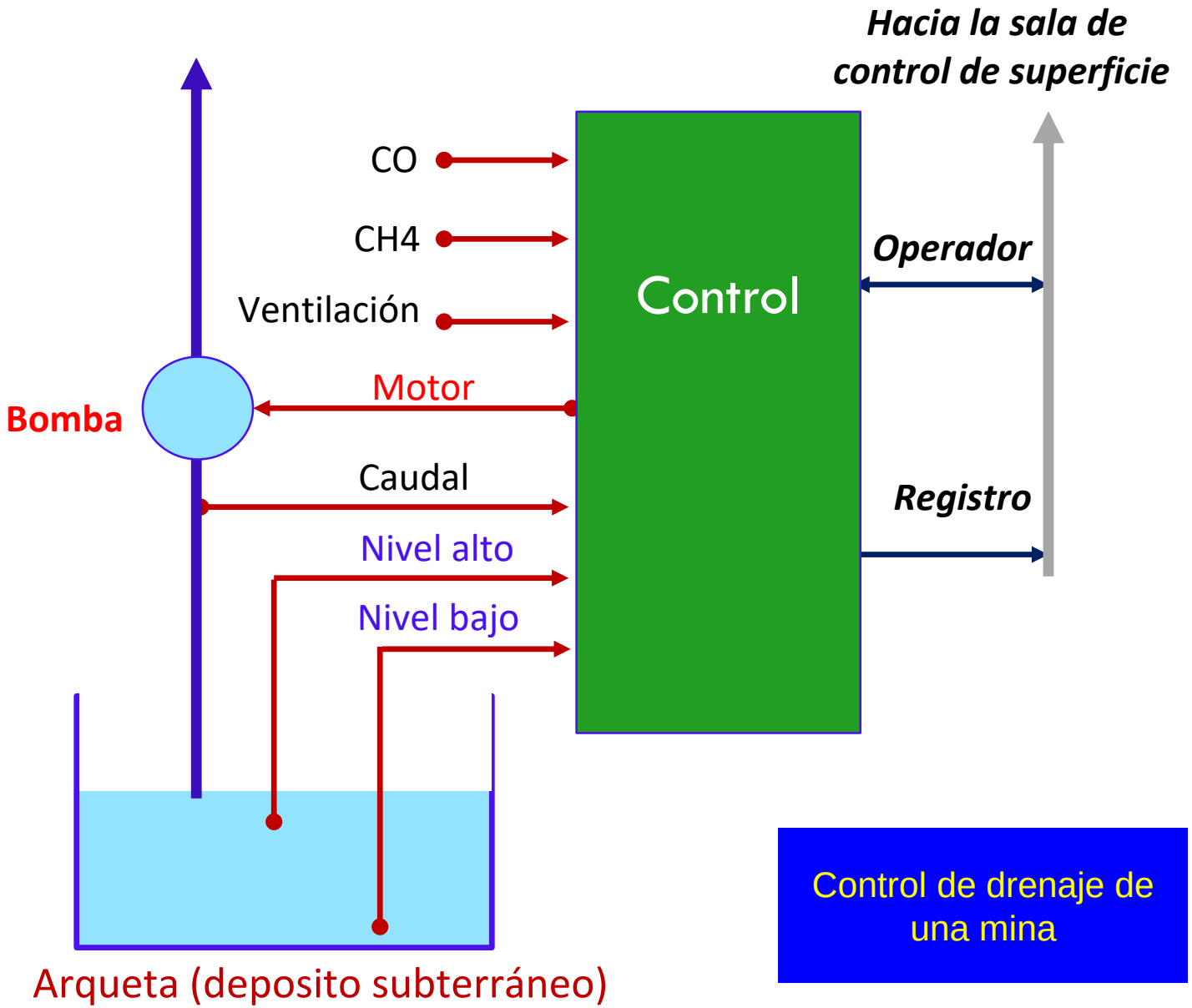
Programación Concurrente

- El paralelismo implica que existen varios procesadores en el sistema.
- La programación paralela implica dividir la ejecución de un programa en distintos módulos los cuales se ejecutaran en distintos procesadores.



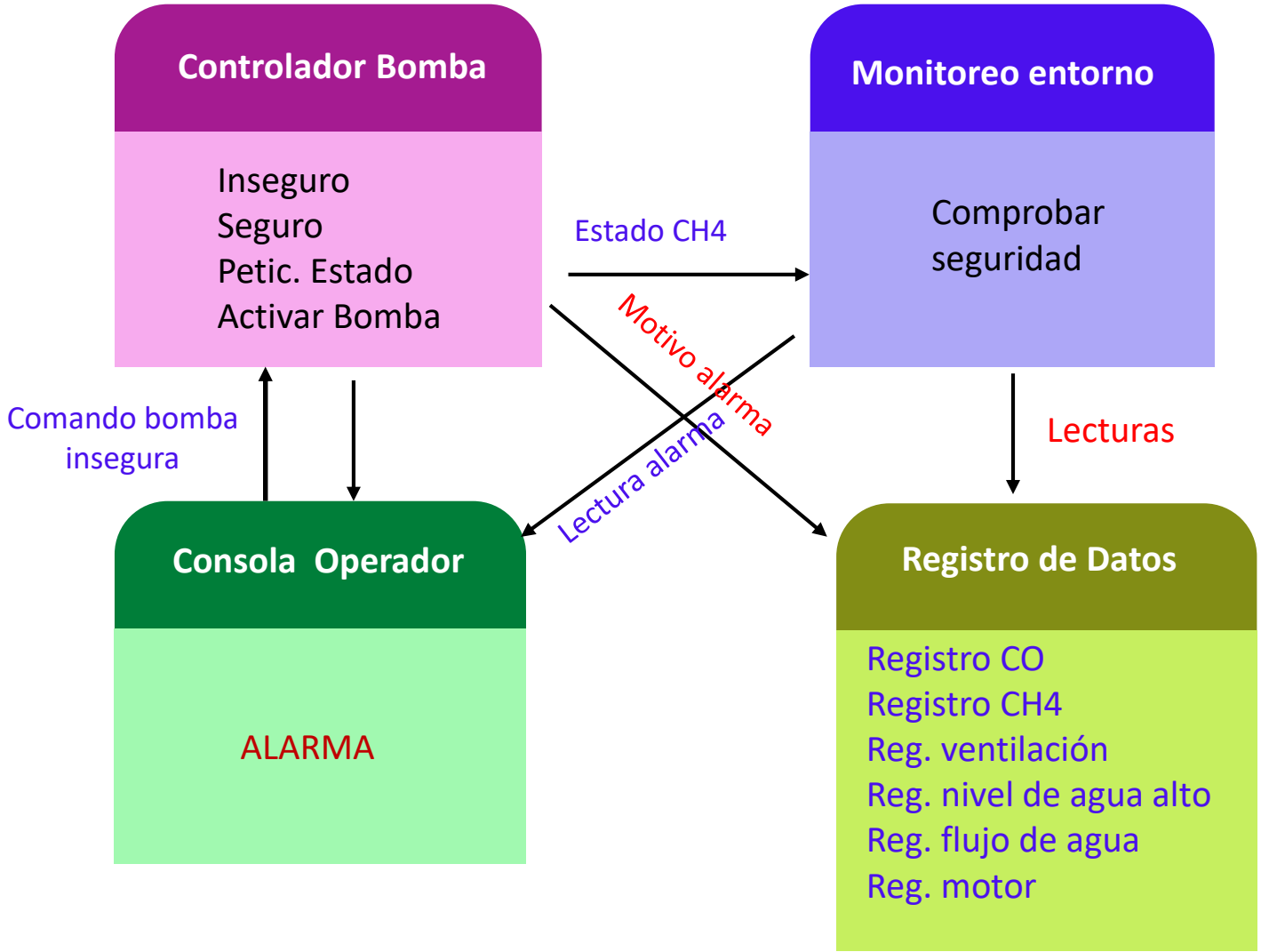
Programación Concurrente: Beneficios

- Virtualmente todos los sistemas de tiempo-real son inherentemente concurrentes: los dispositivos operan en paralelo en el mundo real.
- Proporciona el modelo más simple y natural de concebir muchas aplicaciones. Estructurar un programa como conjunto de procesos concurrentes interactuantes, genera una gran claridad sobre lo que cada proceso debe hacer y cuando debe hacerlo.



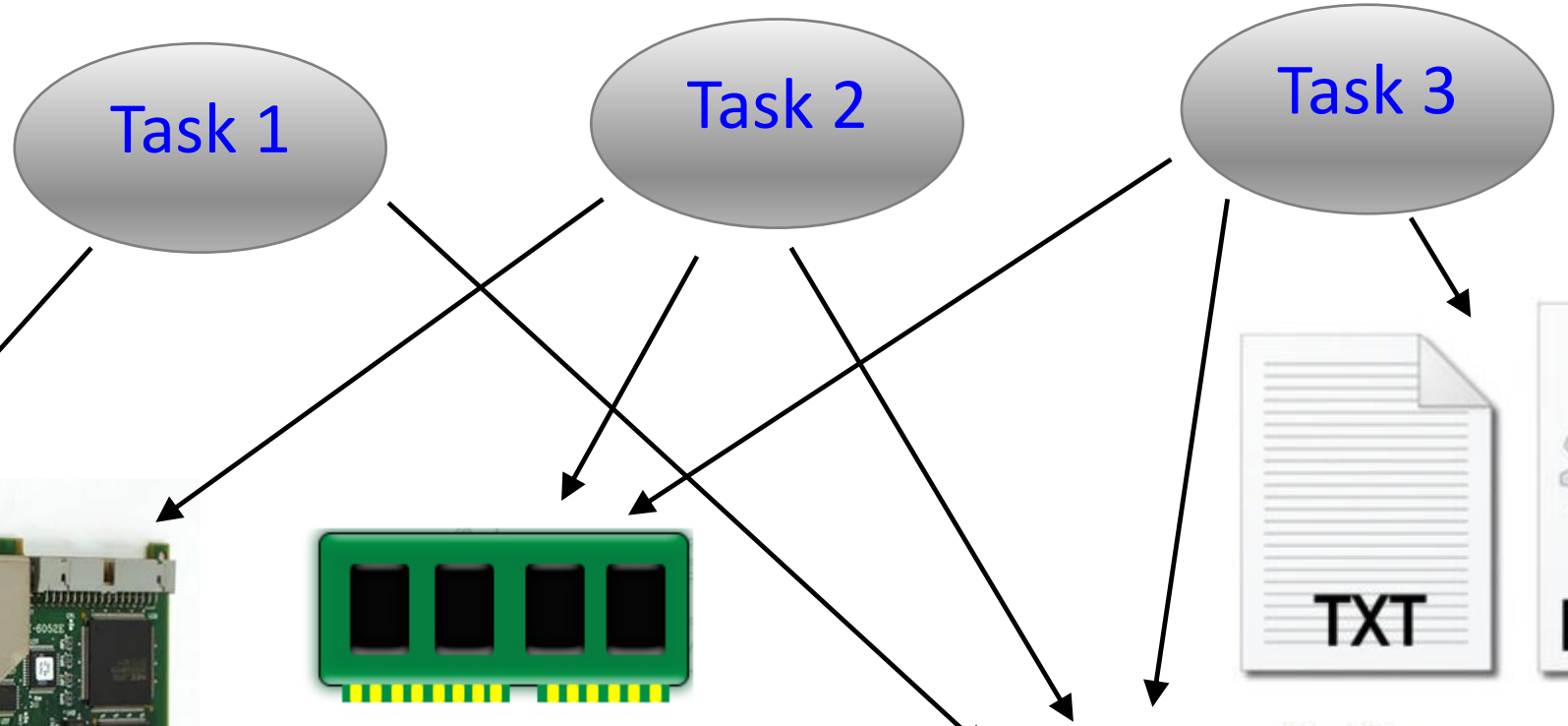
Programación Concurrente: Beneficios

- Facilita el diseño orientado a objeto de las aplicaciones, ya que los objetos reales son concurrentes.



Programación Concurrente: Beneficios

- Hace posible compartir recursos y subsistema complejos.



Tarjeta de adquisición de datos

Memoria datos RAM

Archivos

Tarjeta de RED

Programación Concurrente: Beneficios

- En sistemas monoprocesador permite optimizar el uso de los recursos. Permite solapar los tiempos de entrada /salida o de acceso al disco de unos procesos con los tiempos de ejecución de CPU de otros procesos.
- Facilita la programación de tiempo real, ya que se pueden concebir como procesos cuya ejecución se planifican de acuerdo con la urgencia.
- Cuando el entorno es multiprocesador, la ejecución de los procesos es realmente simultánea en el tiempo, y esto reduce el tiempo de ejecución del programa
- Facilita la realización de programas fiables por despliegue dinámico de los procesos en los procesadores

Programación Concurrente

Resumen:

El programa concurrente es un modelo mas natural en la mayoría de las aplicaciones del mundo real, y que refleja mejor la concurrencia que se produce en el dominio del problema. Un diseño que haga énfasis en la simultaneidad de los procesos, conduce a una mayor simplicidad y facilidad de compresión.

Programas y Procesos

Programa

- Objeto estático que existe en un archivo
- Una secuencia de instrucciones
- Existencia estática en espacio y tiempo

```
main() {
Int i, prod=1;
for (i = 0; i < 100 ; i++)
prod = prod * i ;
}
```

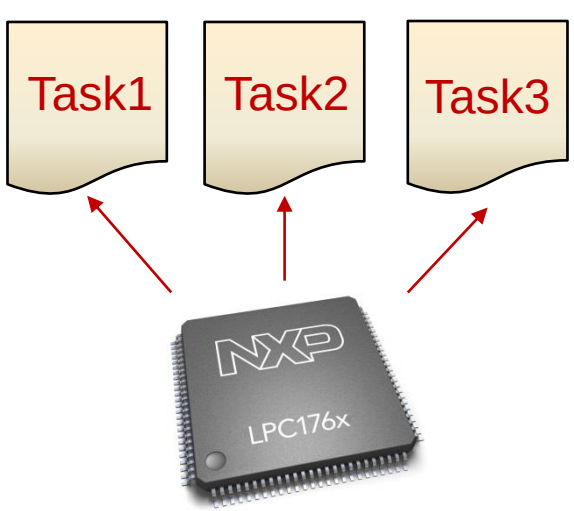
Proceso

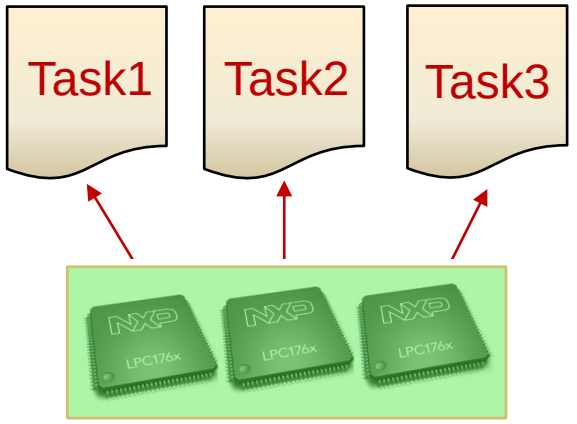
- Objeto Dinámico: Programa en ejecución
- Una secuencia de instrucciones en ejecución
- Existe en espacio limitado de tiempo
- Procesos iguales se pueden ejecutar

```
prod = prod*i
; Este Proceso se ejecuta
 100    veces
```

Definición de proceso

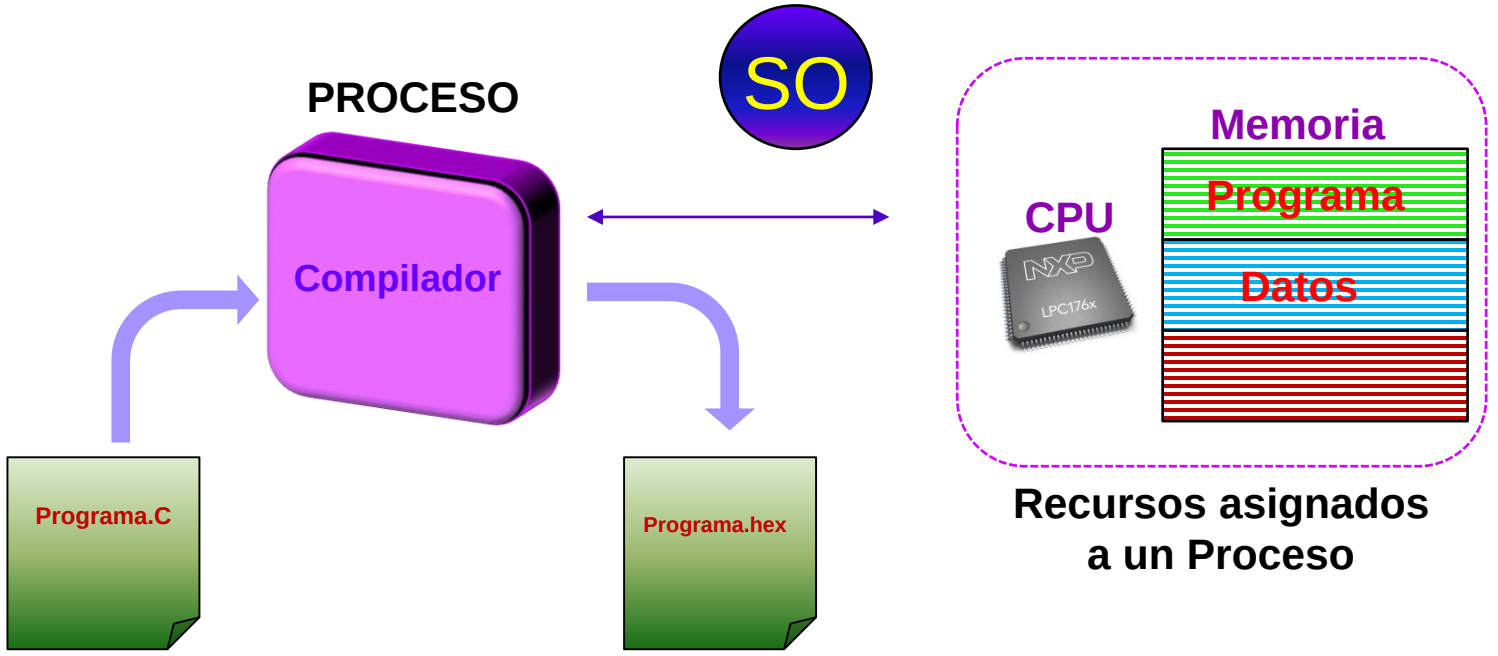
- El SO crea un numero de computadoras virtuales
- La ejecución de un programa sobre una de esos computadoras virtuales es llamada un proceso secuencial
- La computadora virtual da la ilusión que cada proceso esta ejecutándose sobre una CPU dedicada con una memoria dedicada



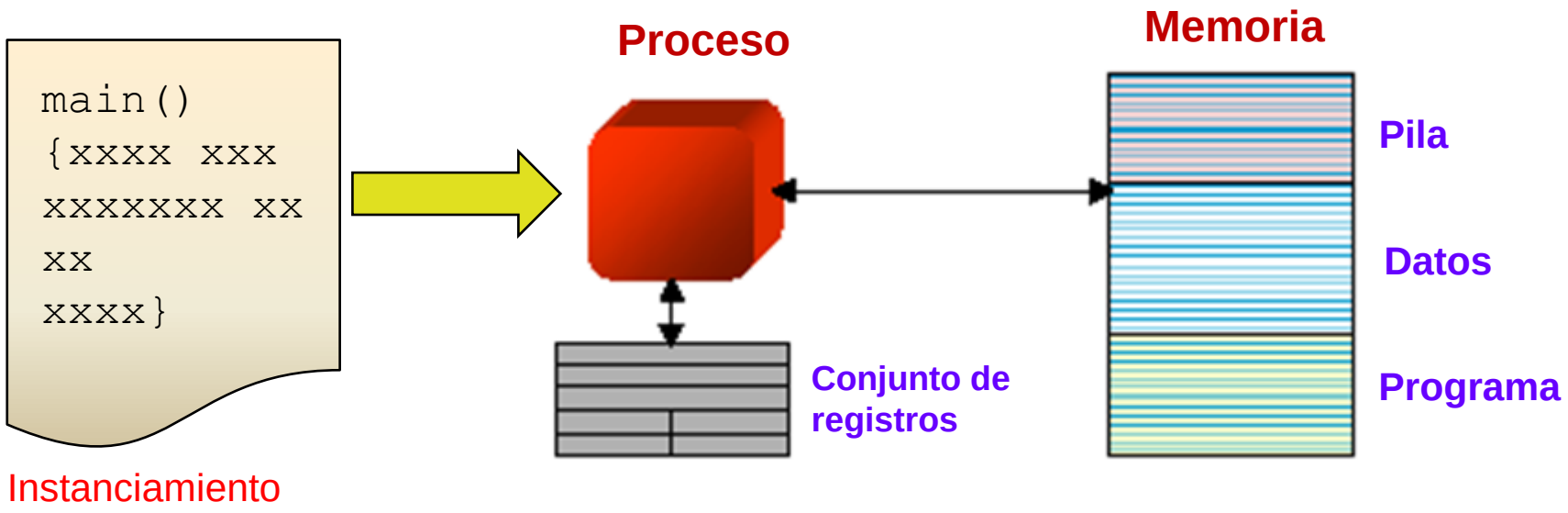


Definición de proceso

- Un proceso es un programa en ejecución
- Componentes: Un código y una estructura de datos
- Ejemplo:
  - ✓ Un compilador C ejecutándose será un proceso para el sistema operativo
  - ✓ El SO le asigna recursos (procesador, memoria, dispositivos, etc.) y controla su ejecución



Esquema de un Proceso



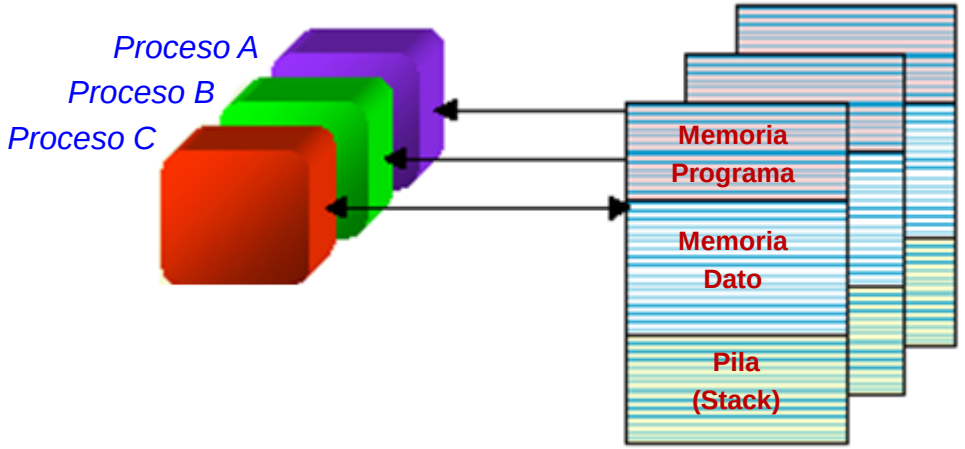
- Varios procesos pueden ejecutar el mismo programa
- Contexto de un proceso: información requerida para su ejecución: código, datos, pila, registros

Estructura de un programa Unix

```
int  debug = 1;
char *progname;
main (int argc,char *argv[])
{
    int i;
    char *ptr,*malloc();
    progname = argv[0];
    printf("arg = %d\n",argc);
    for (i=1; i<argc; i++){
        ptr = malloc(strlen(argv[i])+1);
        strcpy(ptr,argv[i]);
        if (debug) printf("%s\n",ptr);
    }
}
```

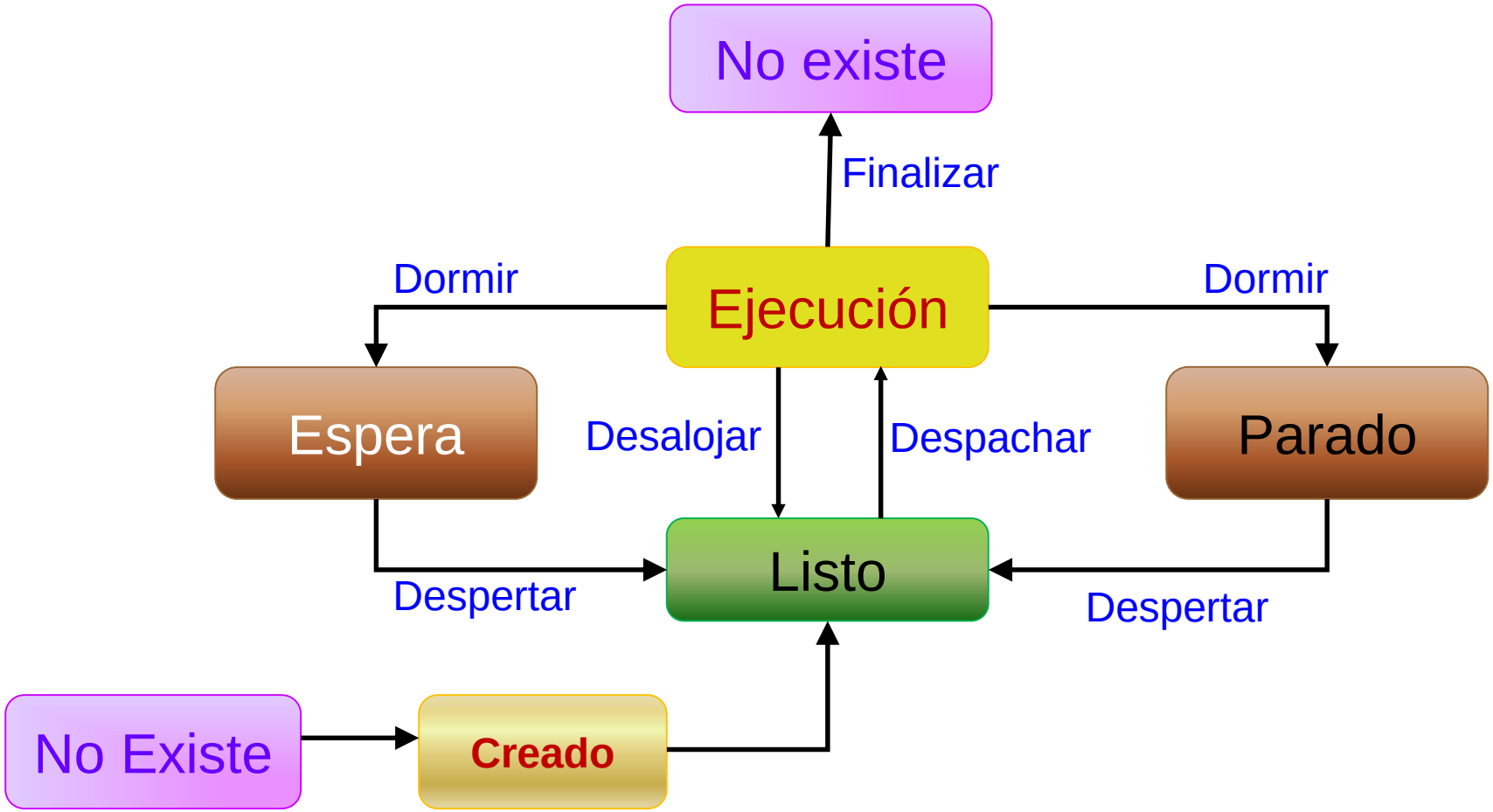
Proceso: Espacio de direcciones

- Cada proceso tiene un espacio de direcciones independiente
- Los cambios de contexto pueden consumir mucho tiempo debido a:
  - ✓ Volumen del contexto
  - ✓ Reprogramación de la unidad de gestión de Memoria (MMU)

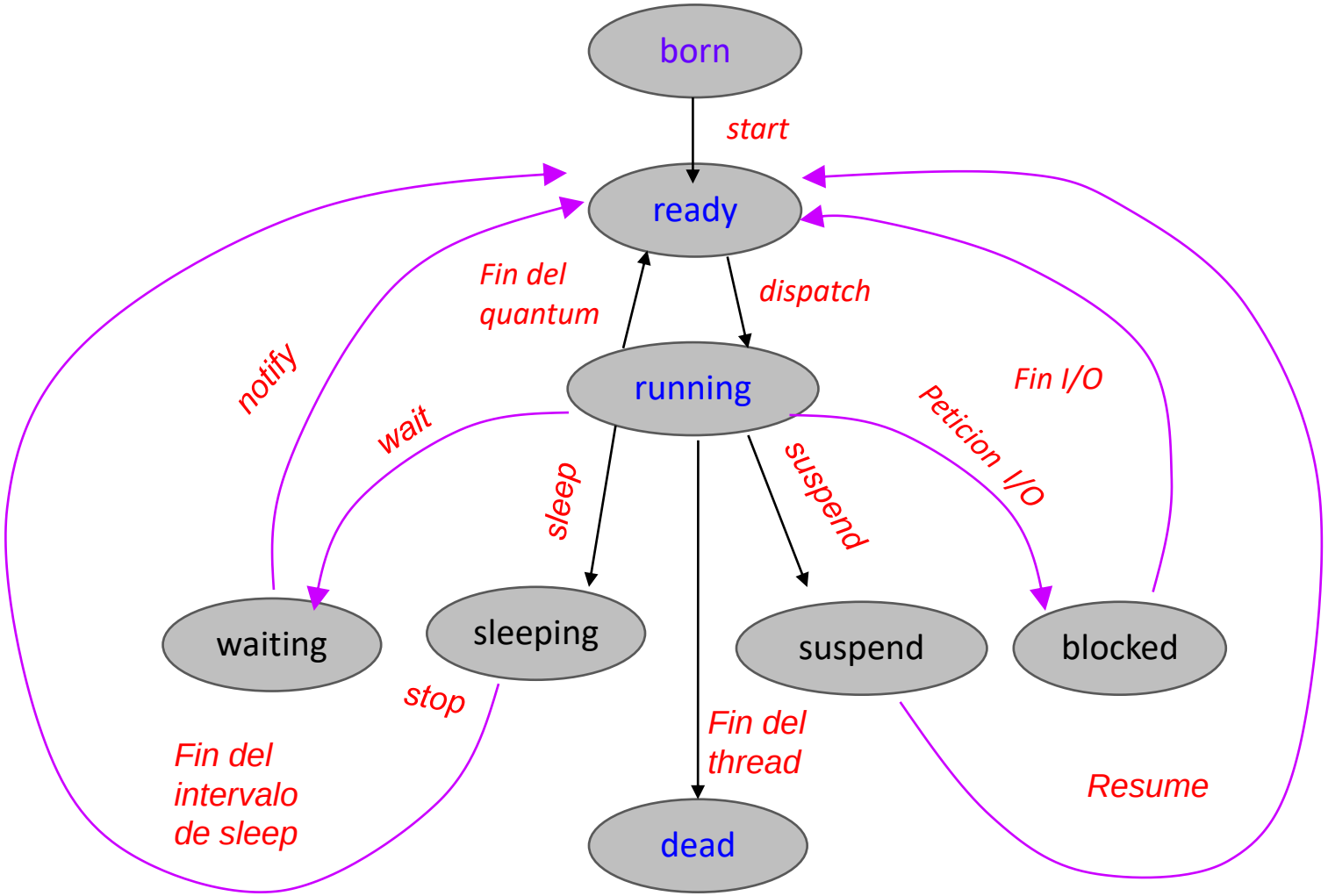


Estados de un Proceso

- Estados de Transición: Cambios en estado determinado por:
  - ✓ Planificador (Scheduler)
  - ✓ Recursos que llegan a estar disponibles



Ciclo de vida de un Proceso (THREAD) en Java



Estados de un Proceso

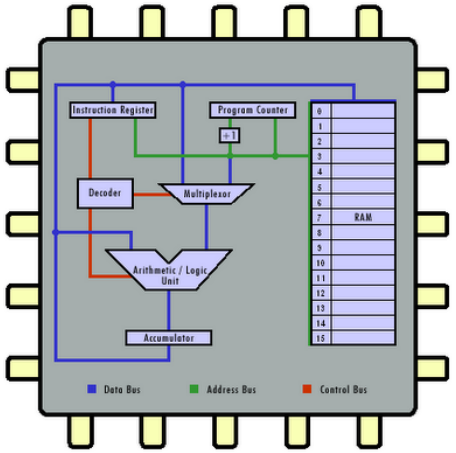
- Creación*:* Cuando se crea un proceso se crea un hilo para ese proceso. Luego, este hilo puede crear otros hilos dentro del mismo proceso, proporcionando un puntero de instrucción y los argumentos del nuevo hilo. El hilo tendrá su propio contexto y su propio espacio de la columna, y pasará al final de los Listos.
- Ready (listo): esperando por la CPU
- Running (ejecutándose) esta usando la CPU

Estados de un Proceso

- Bloqueo*:* Cuando un hilo necesita esperar por un recurso que estará disponible, se bloquea (salvando sus registros de usuario, contador de programa y punteros de pila). Ahora el procesador podrá pasar a ejecutar otro hilo que esté en la final de los Listos mientras el anterior permanece bloqueado.
- Desbloqueo*:* Cuando el suceso por el que el hilo se bloqueó se produce, el mismo pasa a la final de los Listos.
- Terminación*:* Cuando un hilo finaliza se libera contexto

CPU

- El CPU tiene una estructura que contiene información referente a la situación del programa en ejecución (registro de instrucción, contador de programa, registros generales, etc.)



Bloques de Control de un Proceso

- El SO mantiene una tabla de proceso con información de cada proceso, llamado Process Control Block (PCB)
- Almacena:
  - ✓ ID del Proceso, ID Usuario, ID Grupo
  - ✓ Estado actual del proceso (Running, Ready, Blocked)
  - ✓ Directorio de trabajo
  - ✓ Contexto de Hardware: copia los Registros (Contador Programa CP, Stack pointer SP, palabra de estado del programa PSW y otros)
  - ✓ Puntero a segmentos de memoria asignada (Stack, Data, Text)
  - ✓ Parámetros de Prioridad/Planificación
  - ✓ Tabla de archivos Abiertos
  - ✓ Registro de los recursos asignados

Sistemas Concurrentes: MODELOS

- La implementación actual (la ejecución) de una colección de procesos usualmente toma una de estas tres formas:
- Multiprogramación

  Ejecución de múltiples procesos en un solo procesador. Los procesos multiplexan sus ejecuciones en un solo procesador
- Multiprocesamiento

  Ejecución de múltiples procesos en un sistema multiprocesador donde hay acceso a memoria compartida. Los procesos multiplexan sus ejecuciones sobre un sistema.
- Programación Distribuida

  Ejecución de múltiples procesos en varios procesadores los cuales no comparten memoria. Los procesos multiplexan sus ejecuciones en varios procesadores.

En todos los casos hay que multiplexar el uso del procesador o procesadores entre los procesos.

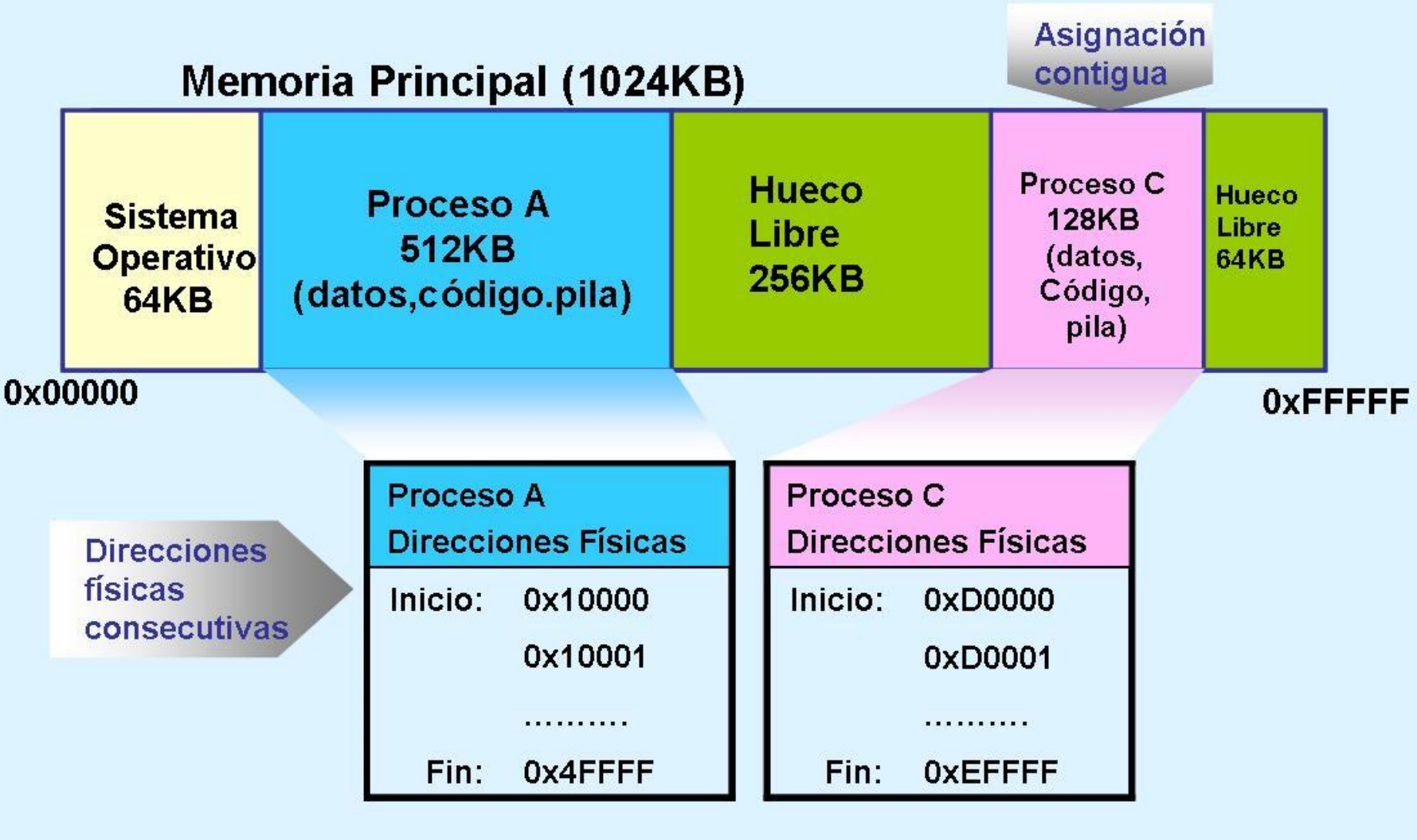
Sistemas Concurrentes

- Razones
  - ✓ Compartir recursos físicos
  - ✓ Compartir recursos lógicos
  - ✓ Acelerar los cálculos
  - ✓ Modularidad
  - ✓ Comodidad

Procesos y contexto

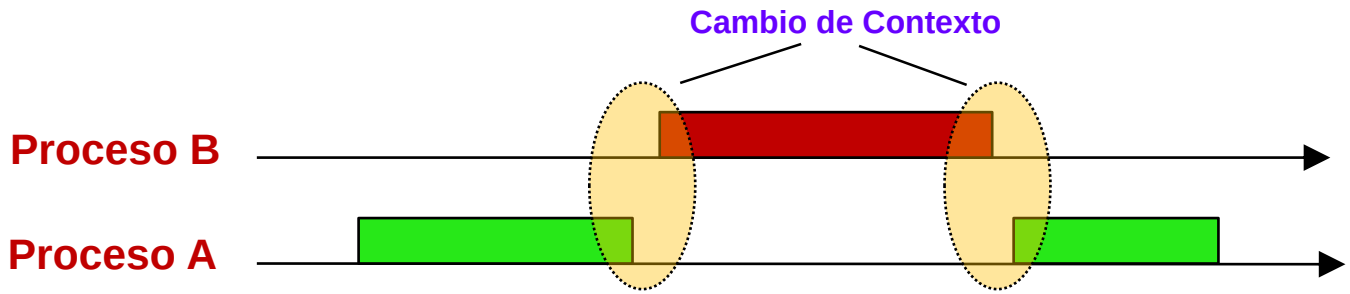
- En sistemas concurrentes observamos que se multiplexa los múltiples procesos sobre una sola CPU. En cualquier instante de tiempo solo un proceso esta en ejecución (*running).*
- Cuando un proceso sale de ejecución para dejar los recursos a otro proceso, el sistema debe:
  - ✓ Salvar el estado actual del proceso
  - ✓ Copiar todos los registros al BCP. Estos registros son útiles para restaurar el estado en la próxima ejecución. Es decir permite el retorno de la situación en ese punto cuando el proceso que fue retirado de ejecución sea nuevamente activado
  - ✓ Copia todos los valores de los registros del nuevo proceso desde los registros del BCP

Procesos y contexto

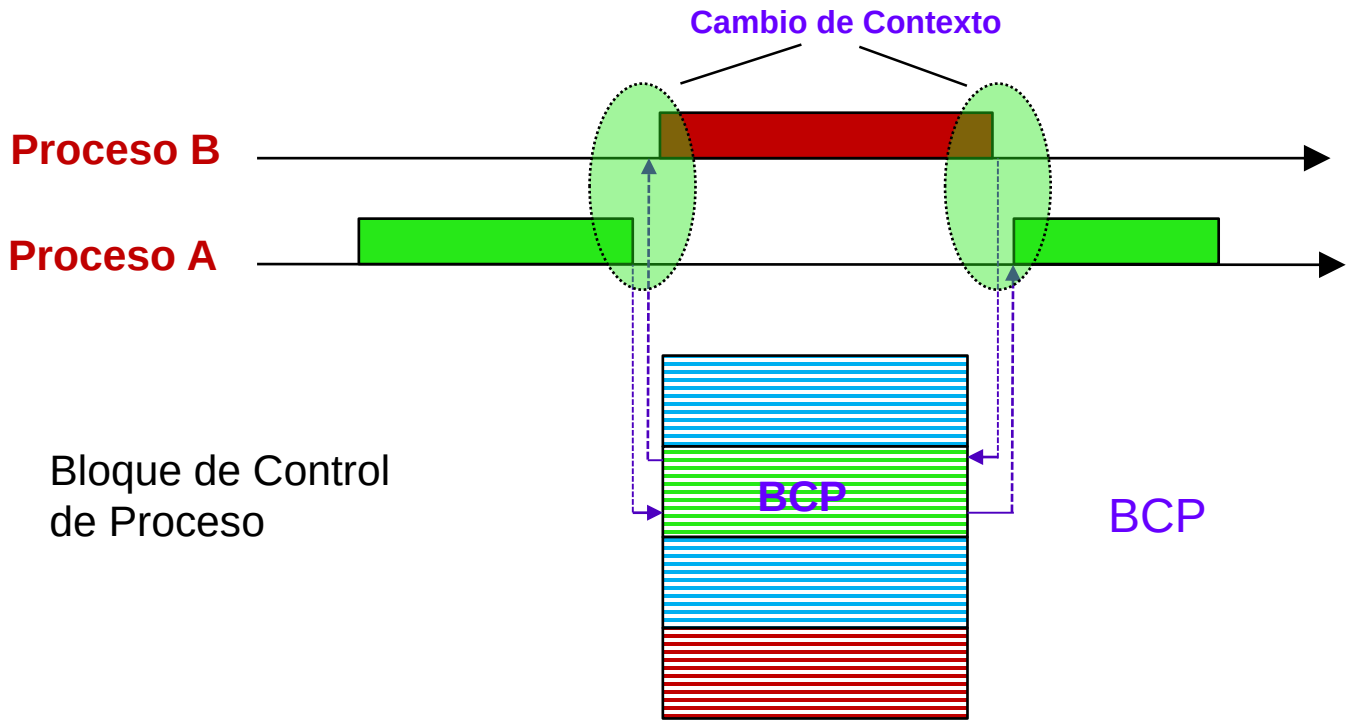


Cambio de Contexto

- Estos pasos mencionados se conoce como: *Cambio de contexto*.
- El cambio de contexto es *overhead*; el sistema no realiza cómputo útil durante el cambio.
- El tiempo de realización del cambio de contexto es dependiente del soporte de hardware.



Cambio de Contexto



- BCP: contiene toda la información sobre un proceso
  - ✓ Para reactivarlo: conjunto de registros
  - ✓ Administrativa: identificación, prioridad, dirección de inicio, mascara de señales, etc

Ejecución concurrente

- Los procesos concurrentes se ejecutan con ayuda de un núcleo de ejecución (*Run- Time Kernel*) que es el planificador (scheduler) en un sistema operativo
- El núcleo se encarga de la *creación, terminación y el multiplexado de los procesos.*
- El núcleo puede tomar varias formas:
  - ✓ Núcleo desarrollado como parte de la aplicación
  - ✓ Núcleo incluido en el entorno de ejecución del lenguaje
  - ✓ Núcleo de un sistema operativo de tiempo real
  - ✓ Núcleo microprogramado dentro del procesador para mayor eficiencia.

Ejecución concurrente

- El algoritmo (método) de planificación utilizado por el núcleo de ejecución afectará al comportamiento temporal del programa (sistema).
- Desde el punto de vista del programador, el núcleo de ejecución planifica los procesos de una forma no determinista.