Introducción a los Sistemas Operativos

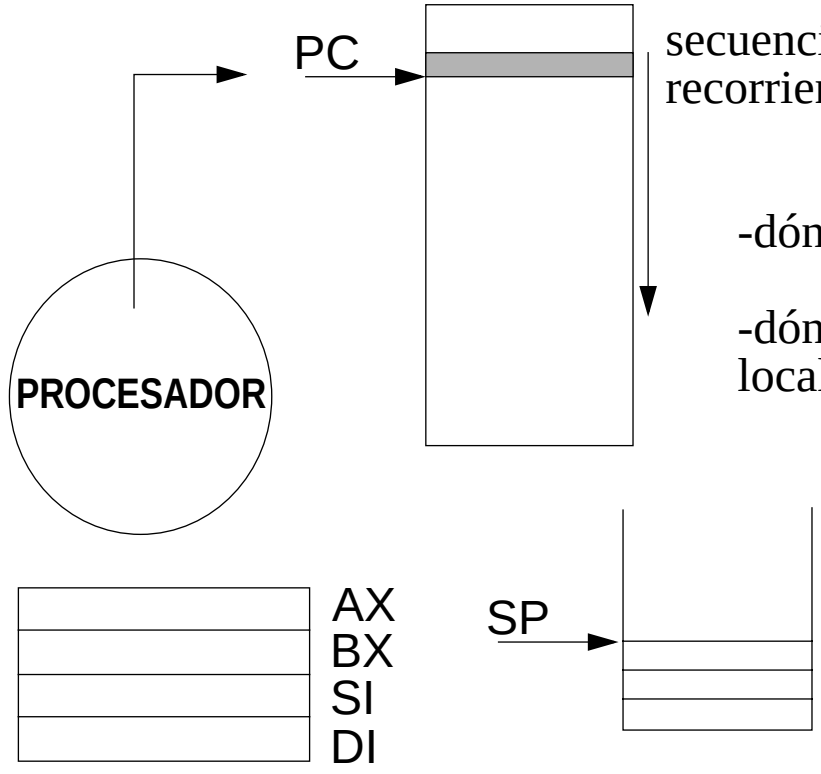
Concurrencia y paralelismo

1. Ejecución de programas. Procesos.
2. Multiprogramación

Bibliografía

Silberschatz and Galvin
Sistemas Operativos. Conceptos fundamentales.
Parte I: Aspectos generales
Tanembaum
Operating Systems. Design and implementation.
Chap. 1: Introduction
Chap. 2.1: Introduction to processes
Stallings
Sistemas Operativos
Cap. 3: Descripción y control del proceso
Birrell
An Introduction to Programming with threads
Digital SRC, 1.989
Thuan Q.Pham and PanKaj K.Garg
Multithreaded Programming with Win32
Prentice-Hall, PTR, 1.999

Ejecución de un programa



secuencia de código que irá recorriendo el procesador

-dónde empieza: PC

-dónde guarda la información local: SP

- a qué memoria se refiere: DS, CS,...

- además, el procesador tiene los registros generales, que utiliza para almacenar datos temporales del programa: AX, BX,... (R0, R1,...)

¿Qué sabe el SO de un programa en ejecución?

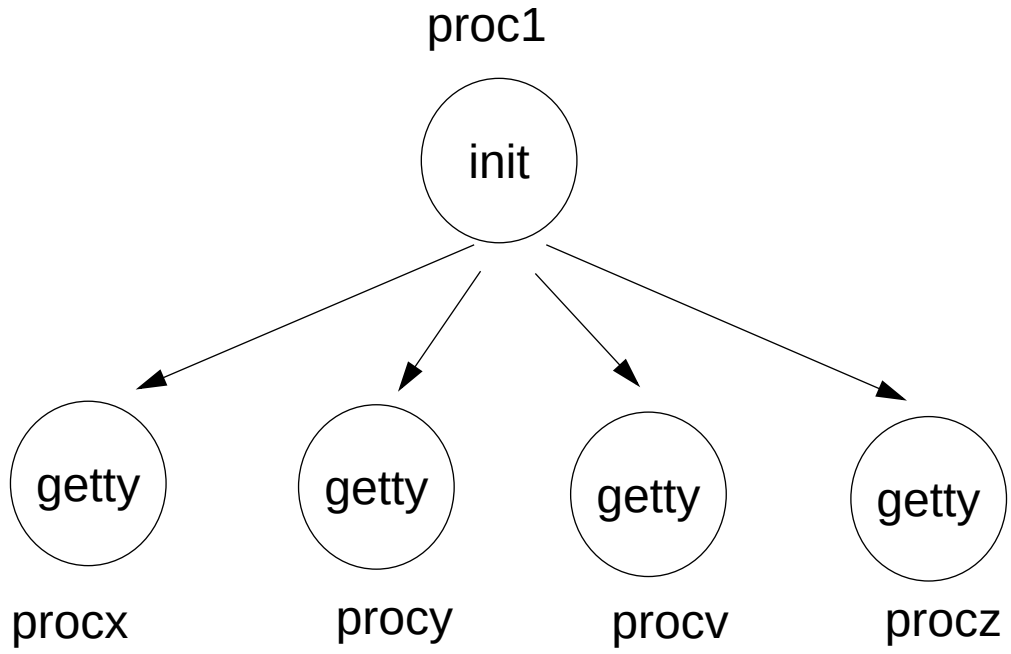
- Usuario que lo ejecuta (asignación justa de recursos)
- Momento en que ha empezado la ejecución
- Tiempo que lleva ejecutándose (contabilidad de recursos)
- Lugar de memoria que ocupa
- Cantidad de memoria que ocupa

Hay más de un programa en ejecución

- Identificador único
- Prioridad

El PROCESO

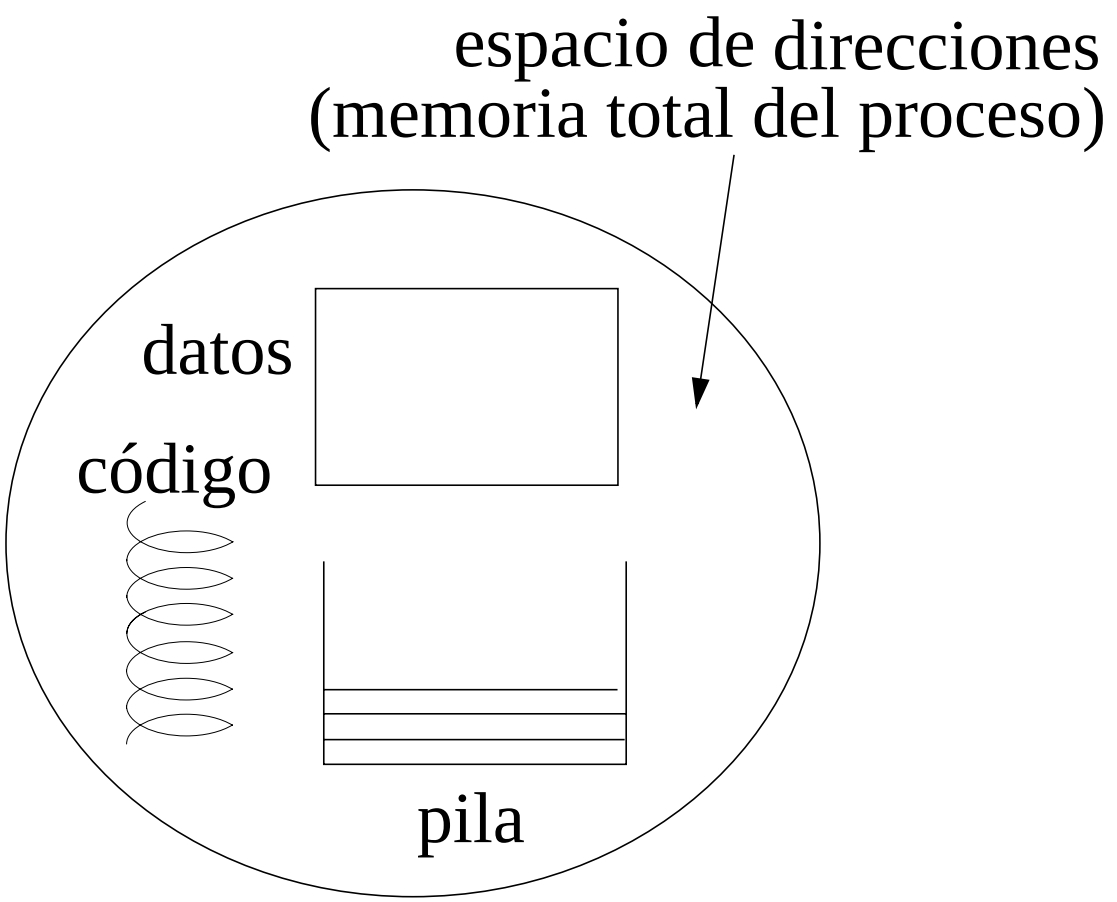
- El proceso es un objeto de sistema
- A cada instancia de programa en ejecución corresponde un proceso



- En el objeto proceso el SO tiene toda la información sobre el entorno de ejecución de una instancia de programa para asignarle el recurso procesador.

El PROCESO

Podemos representar cada objeto proceso por el código que ejecuta, los datos que manipula, los recursos que utiliza y su pila:



Es lo que se conoce también como MÁQUINA VIRTUAL:

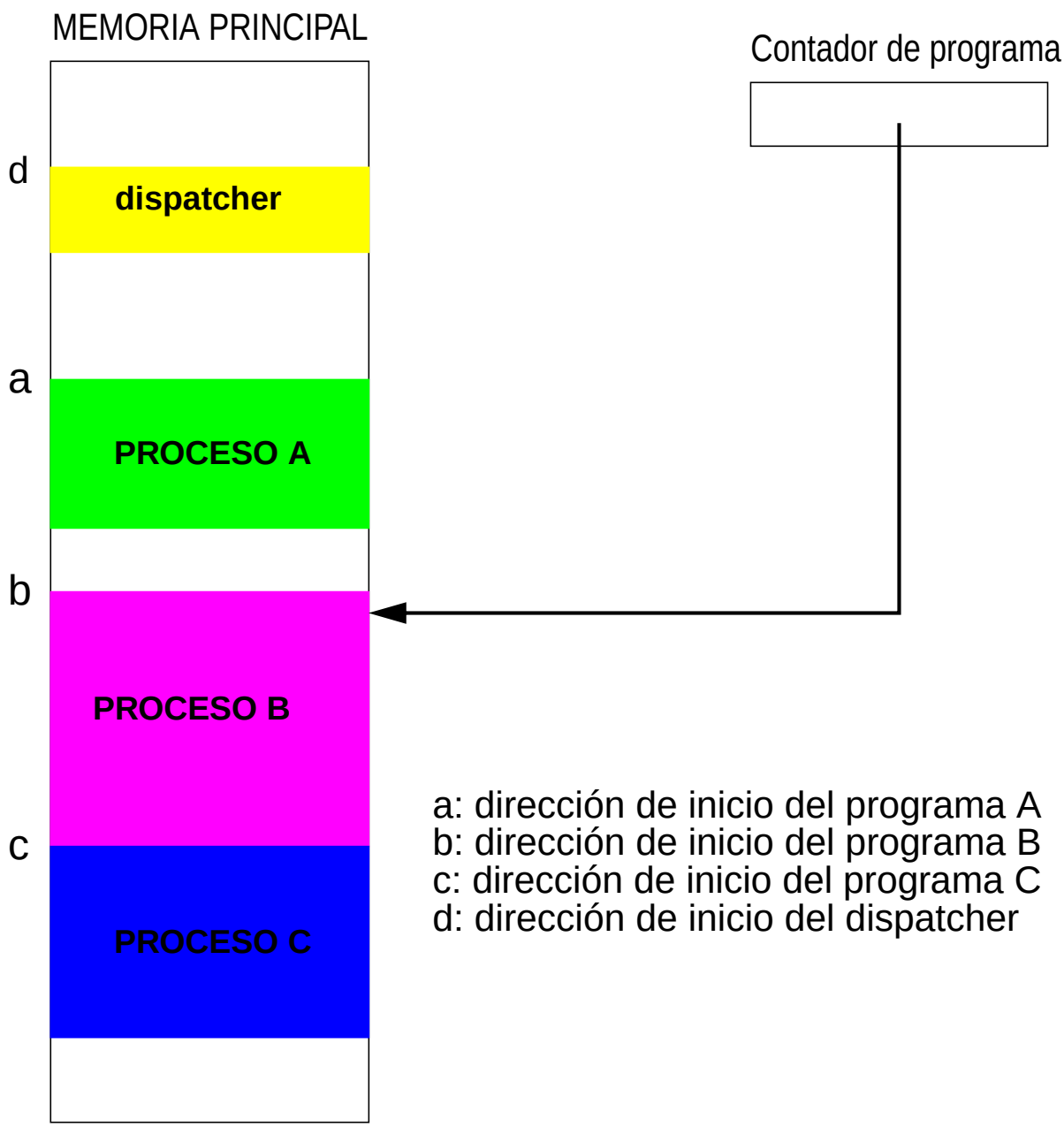
cada programa tiene la “ilusión” de estar trabajando con una máquina “dedicada” a él: procesador, memoria, recursos,...

PROCESO Y FLUJO

Sistemas Multiprogramados

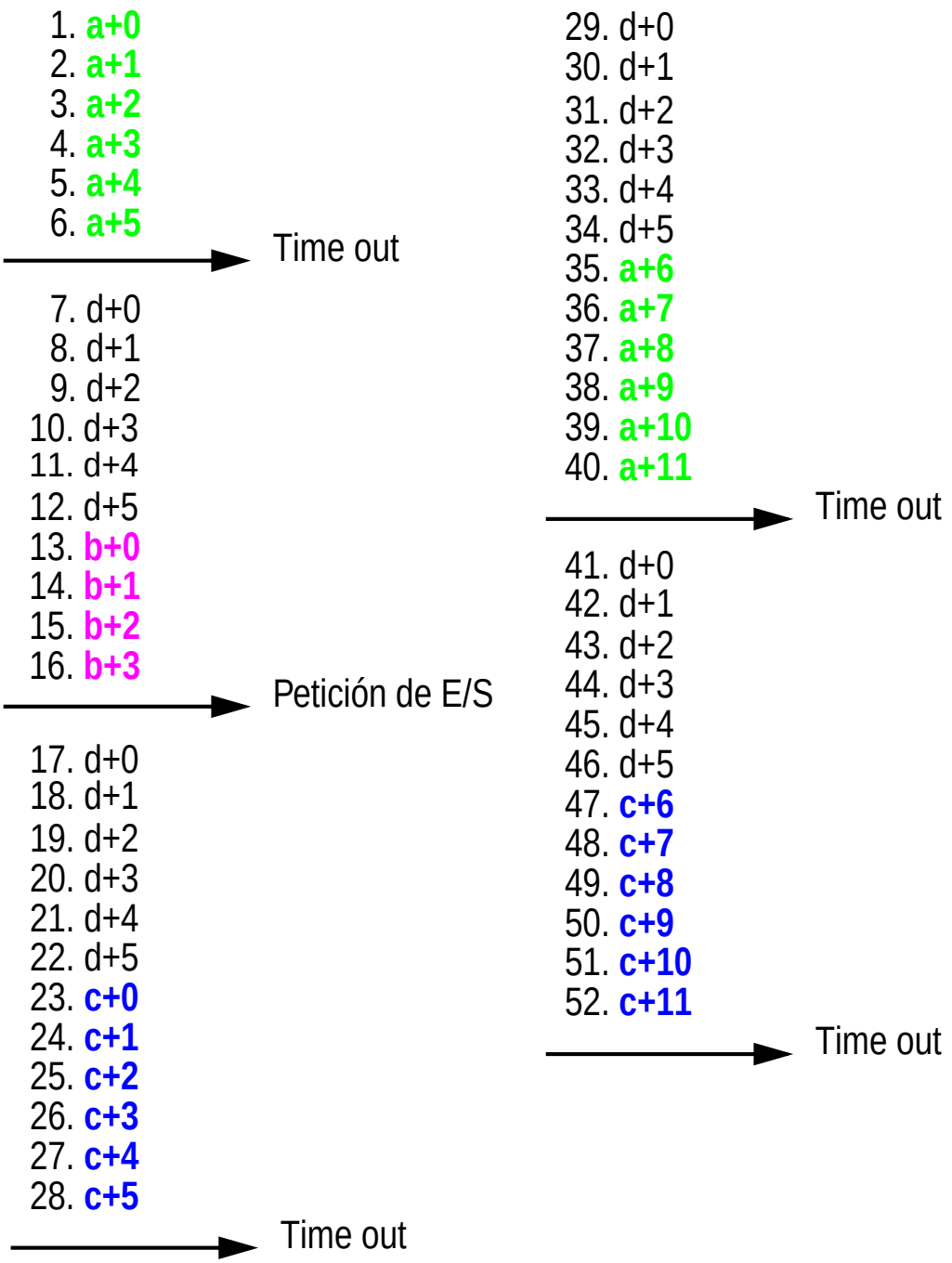
(dibujo extraído de Stallings, pág. 115)

- Varios programas en memoria, se reparten el uso del procesador



PROCESO Y FLUJO

Sistemas Multiprogramados

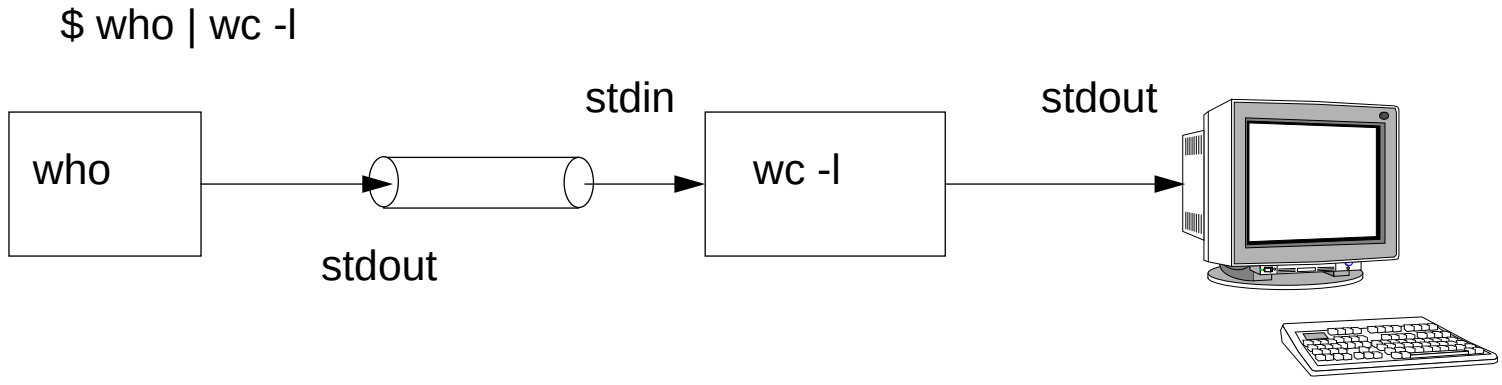


Sistemas multiprogramados

Varios procesos coexisten en el mismo tiempo en el sistema.

- Comparten recursos del sistema
  - impresora
  - red
  - ficheros
- Participan en la realización de un trabajo
  - comandos con pipes
  - procesos padre/hijos/hermanos

Cooperan en realizar un trabajo: filtros



Concurrencia y paralelismo

• Hay **concurrencia** entre varios procesos cuando **existen** al mismo tiempo.
PROCESOS CONCURRENTES

• Hay **paralelismo** entre varios procesos cuando se **ejecutan** al mismo tiempo.
PROCESOS PARALELOS

• El paralelismo requiere un soporte físico: varios procesadores.
• La concurrencia es el caso general y el paralelismo un caso particular.
• La concurrencia (y el paralelismo) se refiere a la ejecución de código:
Hay procesos concurrentes y flujos concurrentes.

• Hablaremos en general de **flujos concurrentes**:
Pueden ser del mismo proceso o diferentes procesos.
Pueden correr en un único procesador o varios.

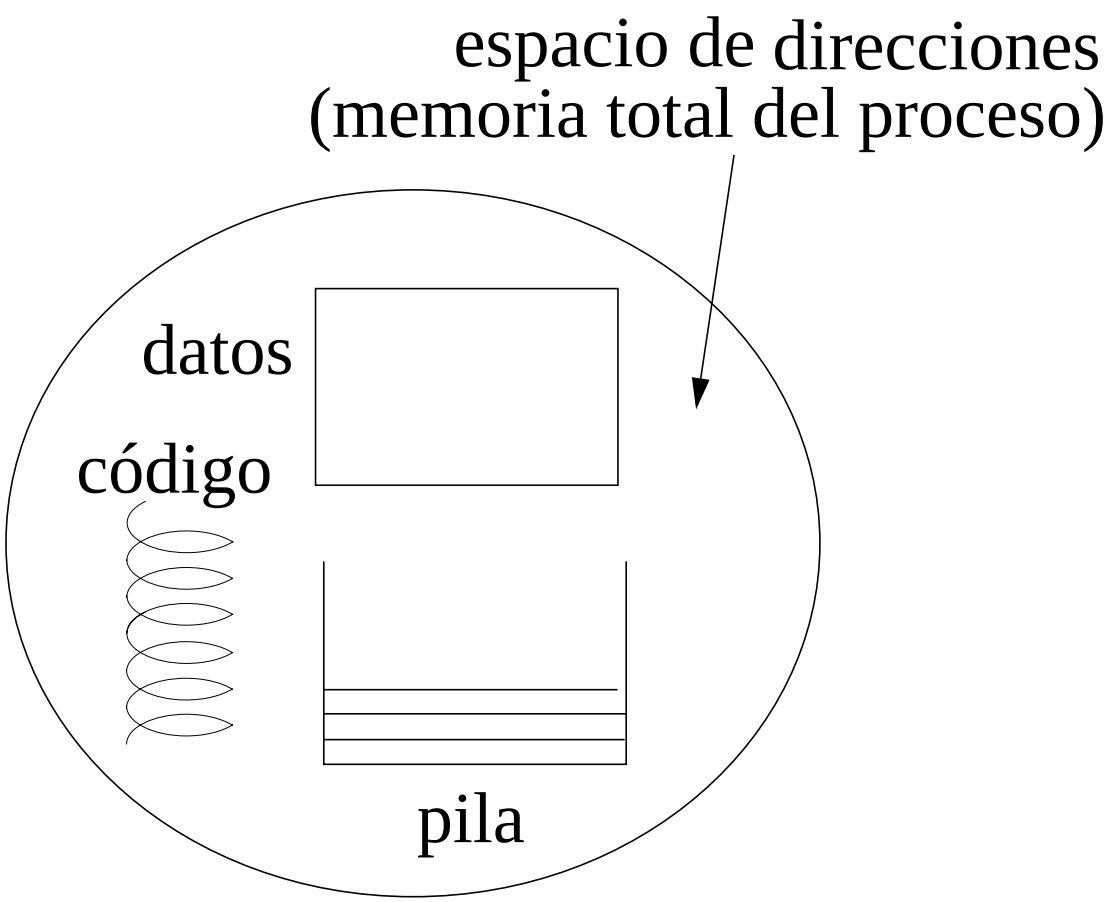
El PROCESO

DEF(1): “Definimos proceso como **todas aquellas acciones ejecutadas por una CPU con un segmento dado como segmento descriptor**, desde el primer momento que dicho segmento se convierte en segmento descriptor hasta el último momento en que el segmento deja de serlo. Así, un proceso tiene un **principio muy definido** y si acaba, tiene un **final igualmente definido**” (VYSSOTSKY et al. , 1965, MULTICS).

- Son sinónimos de proceso: entorno de ejecución, unidad de asignación de recursos, task,...

El PROCESO

Podemos representar cada objeto proceso por el código que ejecuta, los datos que manipula, los recursos que utiliza y su pila:

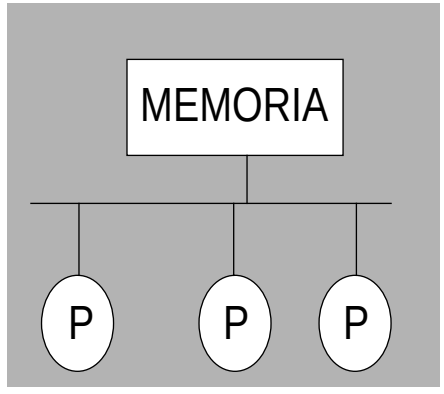


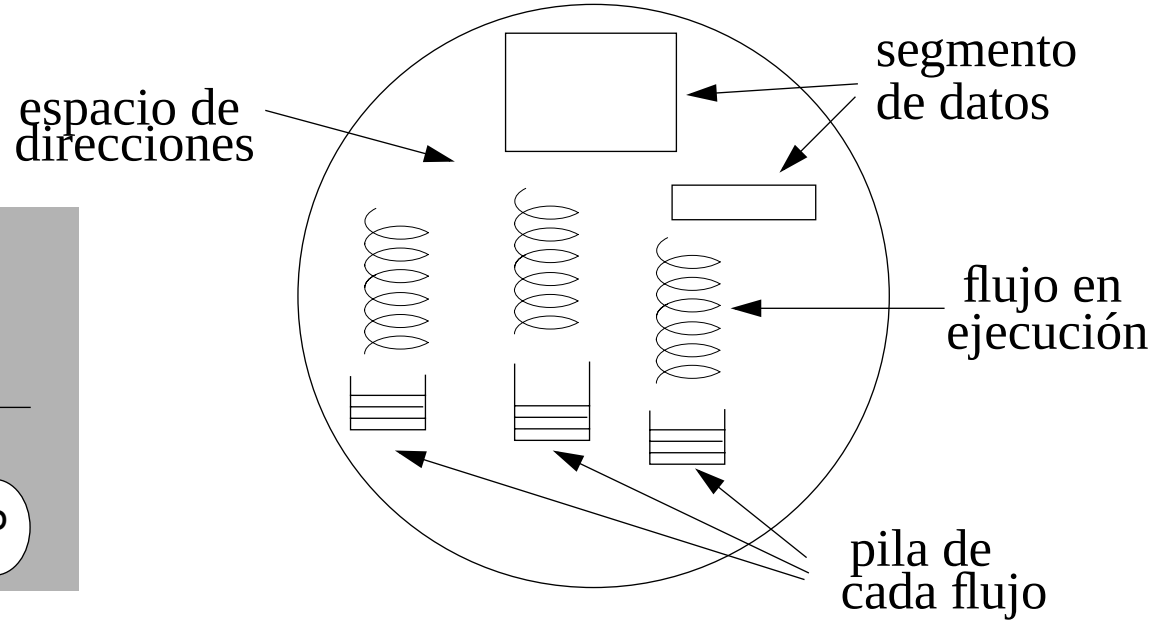
Es lo que se conoce también como **MÁQUINA VIRTUAL**:

cada programa tiene la “ilusión” de estar trabajando con una máquina “dedicada” a él: procesador, memoria, recursos,...

El FLUJO (*Thread)*

- Para que un programa pueda ejecutar diferentes partes en paralelo, en una máquina multiprocesador:
  - buscar un modelo de MÁQUINA VIRTUAL MULTIPROCESADOR
  - el SO asigna a un mismo proceso, varios procesadores al mismo tiempo.





- Se diferencia el objeto proceso en **task** y **thread**

El FLUJO

DEF (2): “Una *task* es un **entorno de ejecución** en el cual pueden correr *threads*. Como **unidad básica de asignación de recursos**, una *task* incluye un (...) espacio de direcciones y acceso protegido a los recursos del sistema (como procesadores,..., y memoria). La noción de proceso en UNIX se representa en Mach por una *task* que tiene un único *thread* de control.

Un *thread* es una **unidad básica de utilización de la CPU**. Es totalmente equivalente a un contador de programa independiente operando en una *task*. Todos los *threads* de una *task* comparten el acceso a todos los recursos de la *task*”. (RASHID et al., 1988, diseñadores de Mach)

El proceso y el flujo

- Proceso (o *task*): unidad de asignación de recursos:
  - espacio de direcciones (registros de la MMU)
  - tabla de canales (y por tanto, dispositivos asignados)
  - etc....
- Flujo: unidad de ejecución
  - código
  - pila
  - registros del procesador
- En UNIX cada proceso sólo puede tener un flujo:
  - el concepto “proceso” engloba todo.
  - Los signals permiten que el proceso sea avisado de acontecimientos.
  - A través del sistema de ficheros, los procesos pueden pasarse datos.
  - El **espacio de direcciones** es **privado** para cada proceso (para cada flujo).

Librerías de flujos

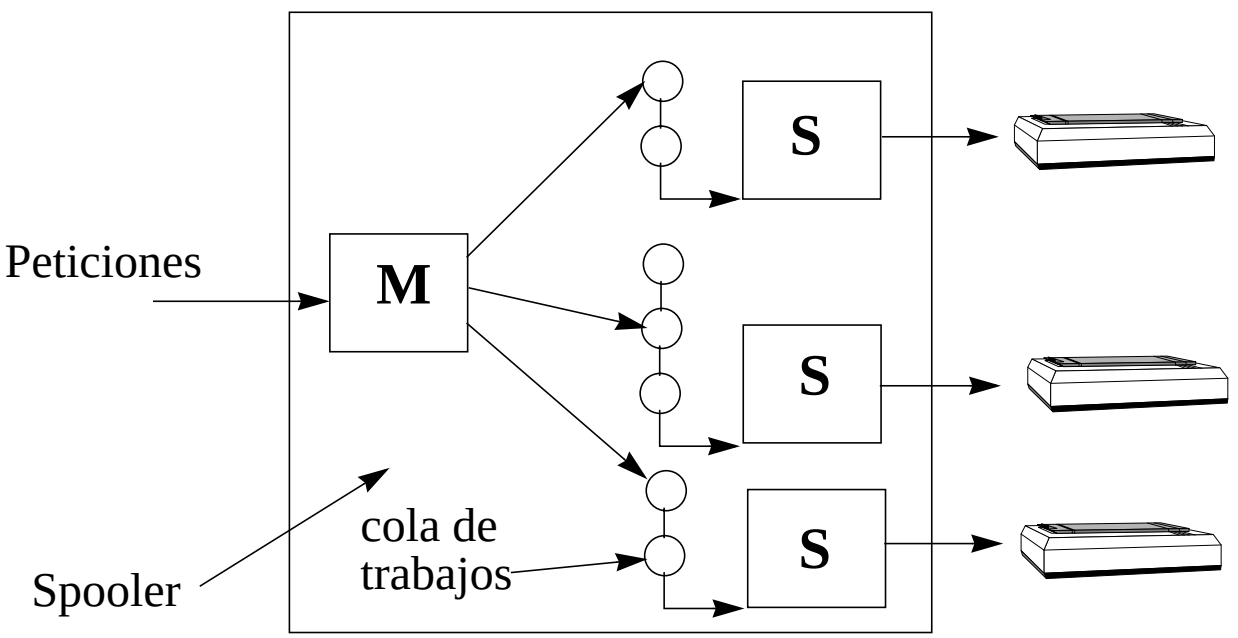
- Se ofrece la posibilidad de cambio de contexto, entidades de flujo y primitivas de sincronización
- Se ofrece un interfaz de trabajo, pero se puede acceder a cualquier rutina pública.
- La planificación de flujos es totalmente transparente al SO
    - No sabe la concurrencia que hay en la librería.
    - Sólo podrá haber tanto paralelismo como “procesadores virtuales” ofrezca.
    - Un bloqueo de un servicio del SO, impide la planificación de cualquier flujo.

Ventajas de la Programación Concurrente

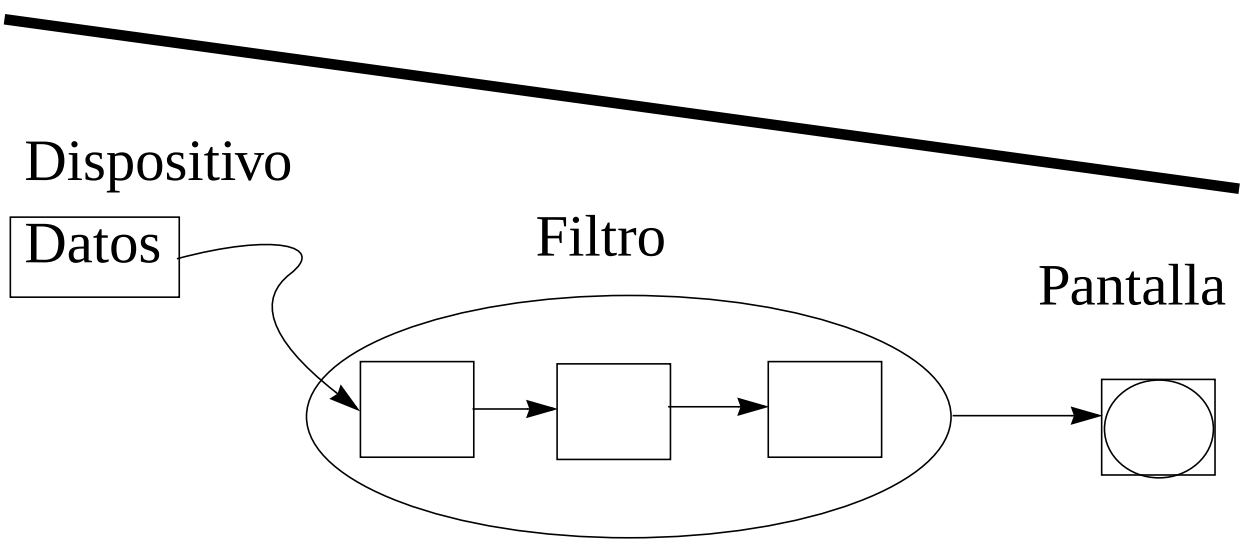
- Programación modular
  - Considerar aisladamente cada tarea
- Aprovechar desde un programa el paralelismo (multiprocesadores).
- Tratar el trabajo con E/S lenta con flujos dedicados.
  - el flujo principal del programa puede seguir trabajando
- Atención a varias peticiones simultáneas (sistemas distribuidos)

Algunos ejemplos de concurrencia

Master/Slave



Productor/Consumidor



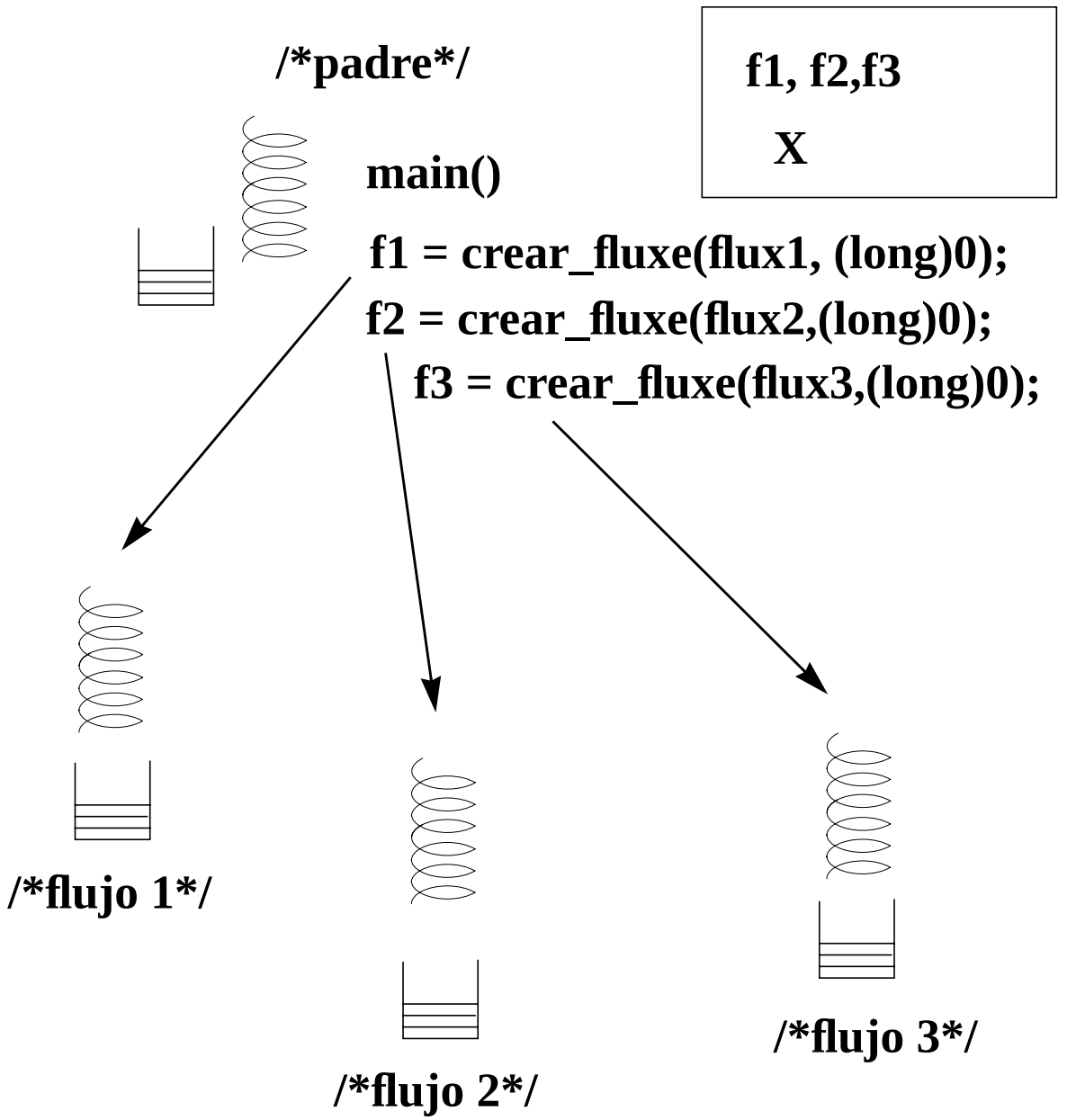
Programación Concurrente

- Avance de una tarea con **múltiples puntos de ejecución**.
- Cada punto de ejecución es un "*thread*".
- El trabajo de cada *thread* "se ve" **simultáneo** desde el programa.
- Son parte de la misma tarea y hay **puntos de encuentro**
    - Sincronización
    - Comunicación
- La gestión de la concurrencia se hace **por software**
    - Sistema operativo
    - Librerías
    - No consideramos hardware "especializado"

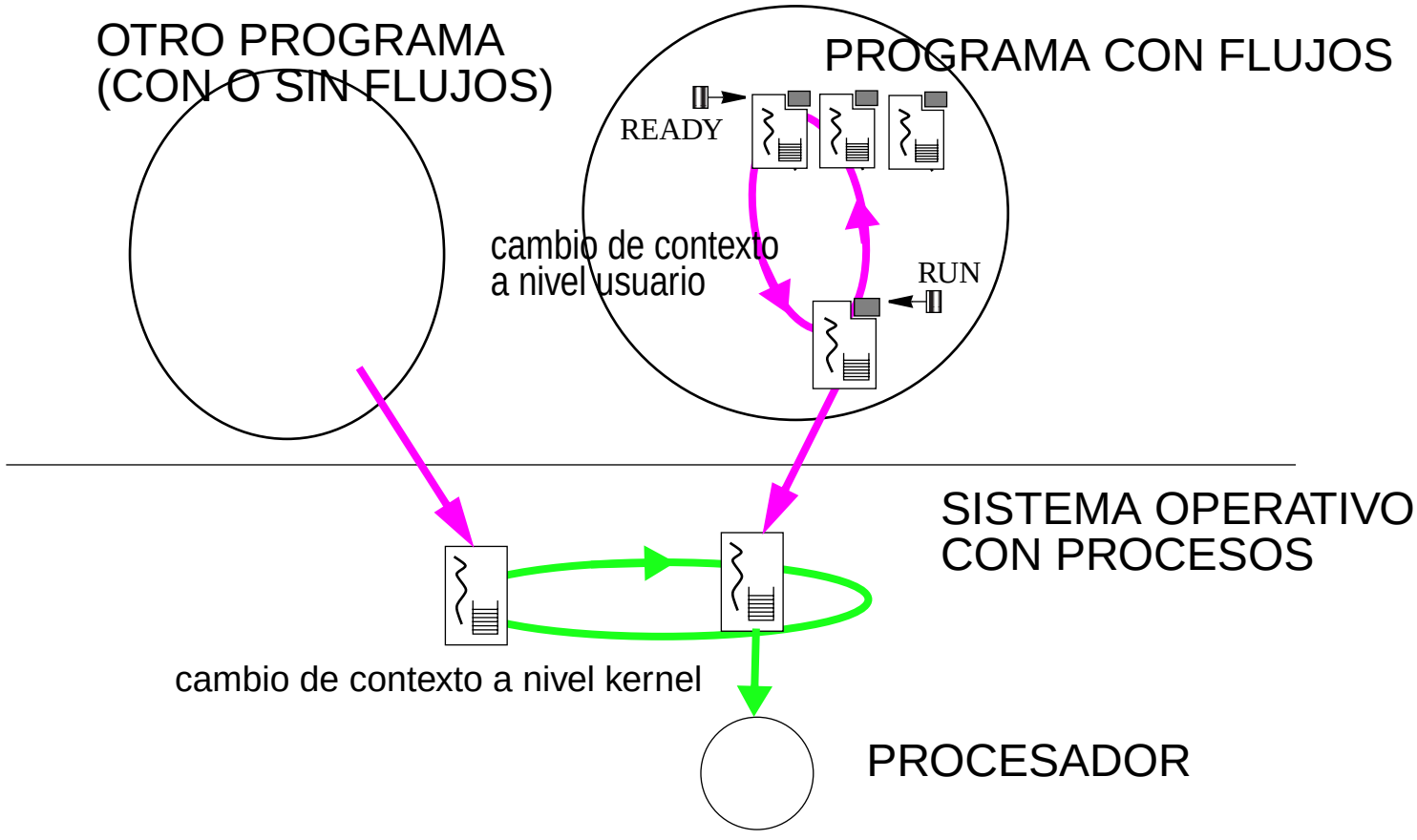
CONCURRENCIA

EJEMPLO: Calcular concurrentemente

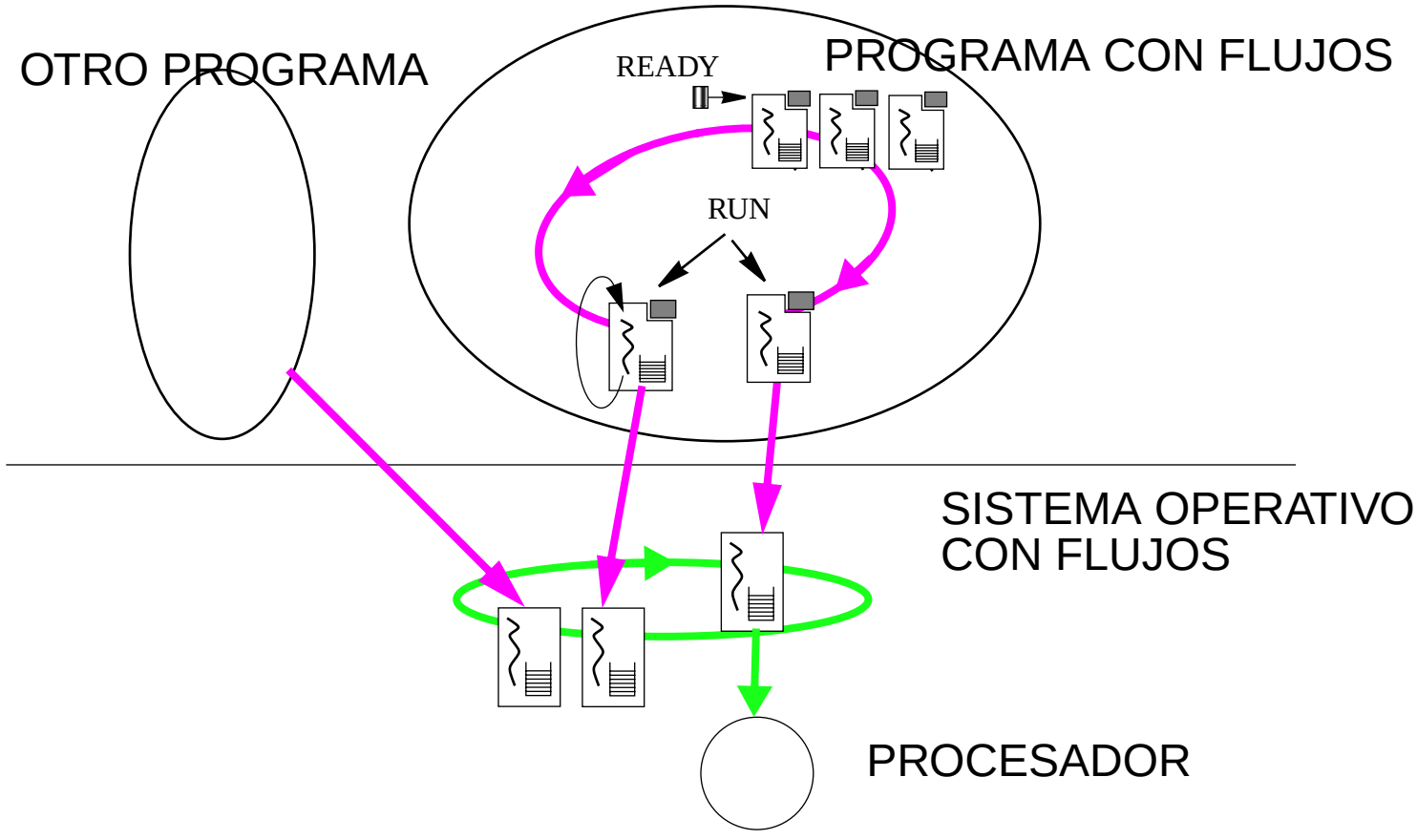
f(x) = 2(3x + 4)



Relación entre la librería de flujos y el SO (I)



Relación entre la librería de flujos y el SO (II)



Relación entre la librería de flujos y el SO (III)