Apuntes clase 3

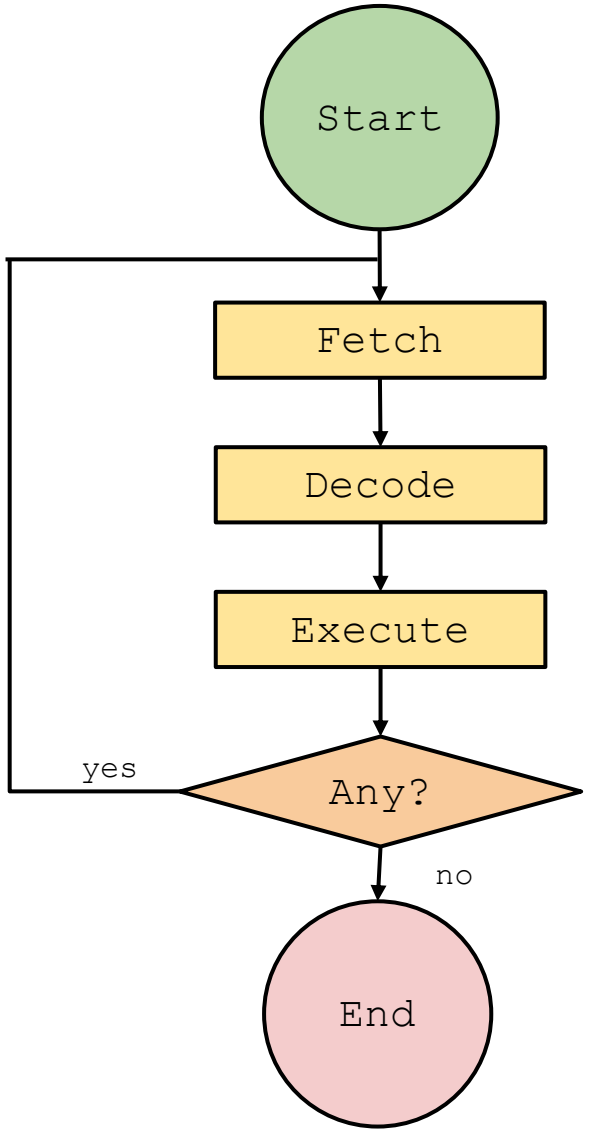
Sistemas Operativos

17/02/2026

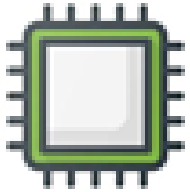
Agenda

- Repaso clase anterior
- Contexto
- ¿Que es un proceso?
- ¿Como se crea un proceso?
- Elementos de los sistemas operativos
- API de procesos
- Estados de un proceso
- Estructuras de datos de sistemas operativos

Instruction fecth

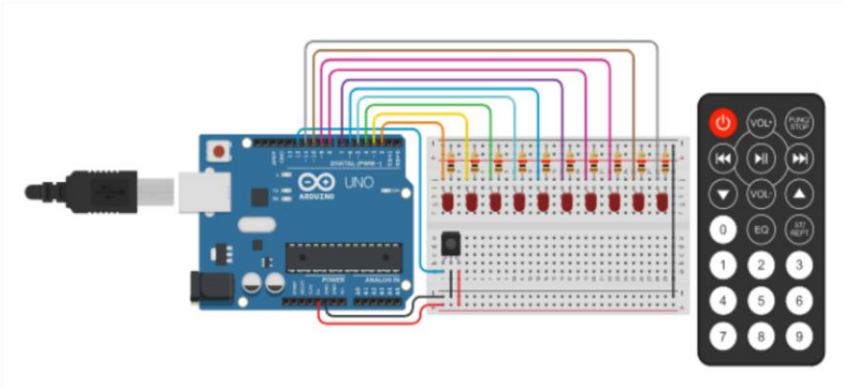


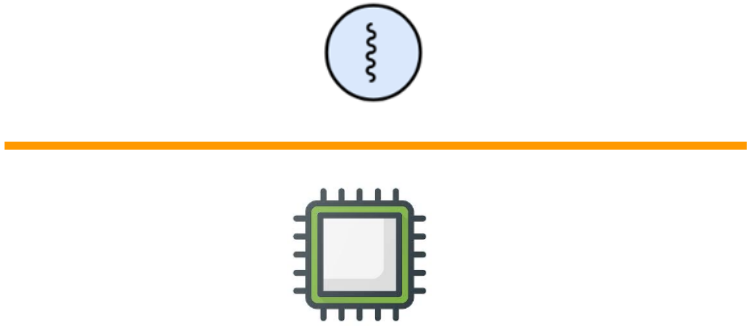

```
.data
    var_x: .word 30
.text
main:
    la $t1, var_x
    lw $t0, 0($t1)
    addi $t0, $t0, 1
    sw $t0, 0($t1)
```



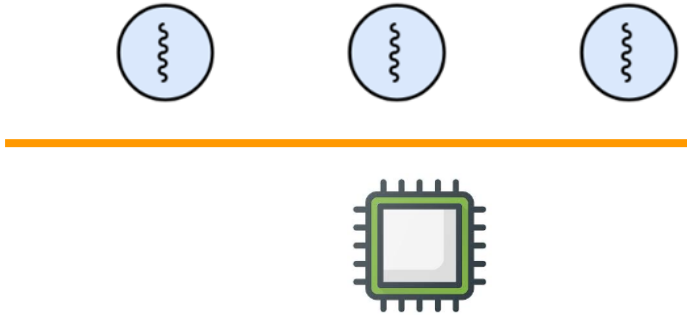
Contraste

Sin sistema Operativo



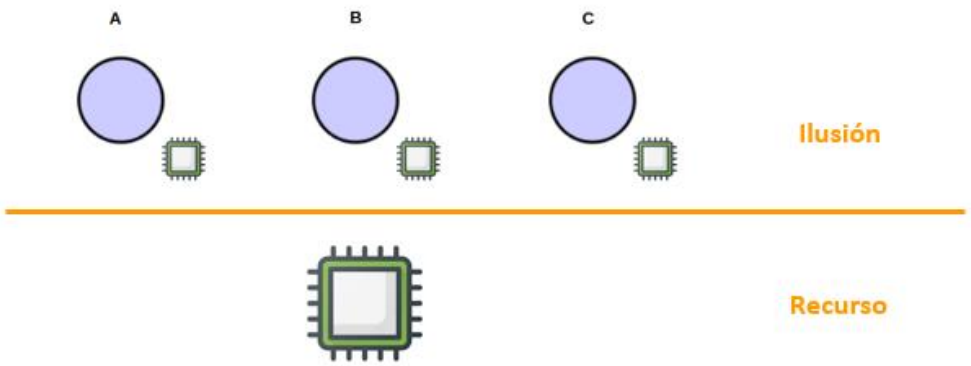


Con sistema Operativo

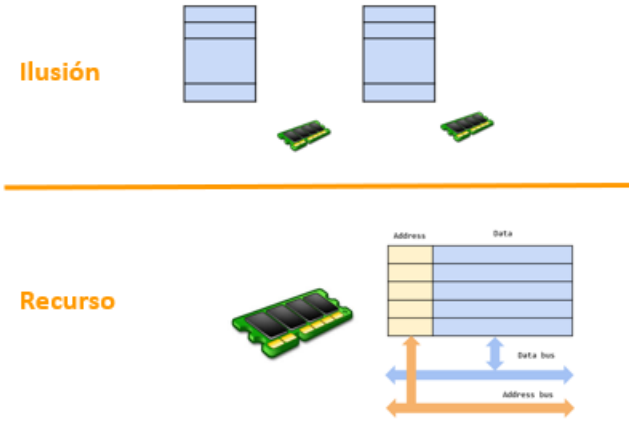


Virtualización

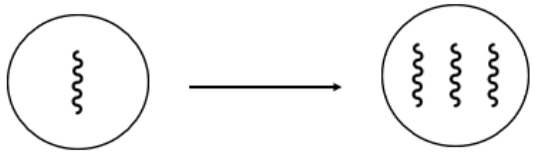
Virtualización de CPU



Virtualización de Memoria



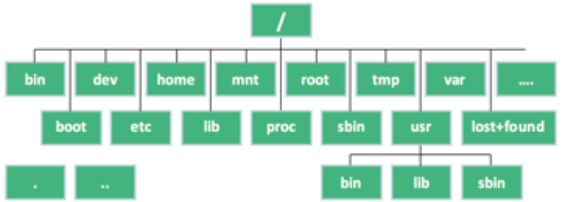
Concurrencia y paralelismo



Un hilo de ejecución

Múltiples hilos de ejecución

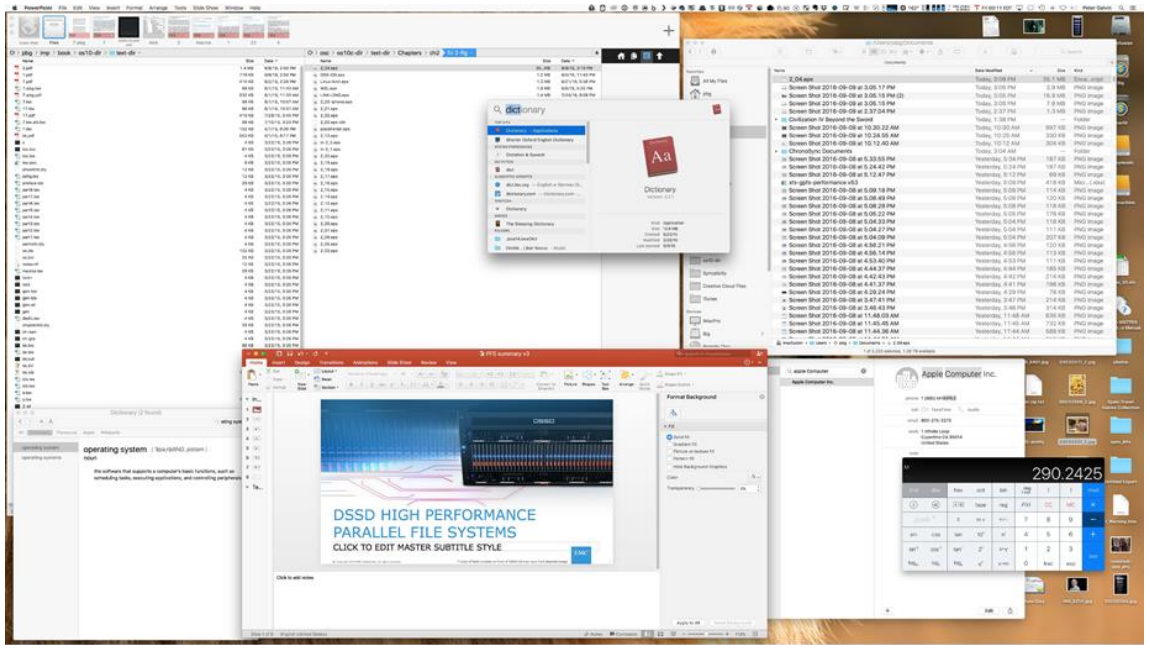
Persistencia

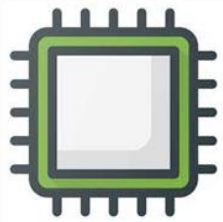


Contextualización

Virtualización de CPU

¿Cómo es que teniendo una sola CPU (en teoría) puedo correr varias tareas a la vez?

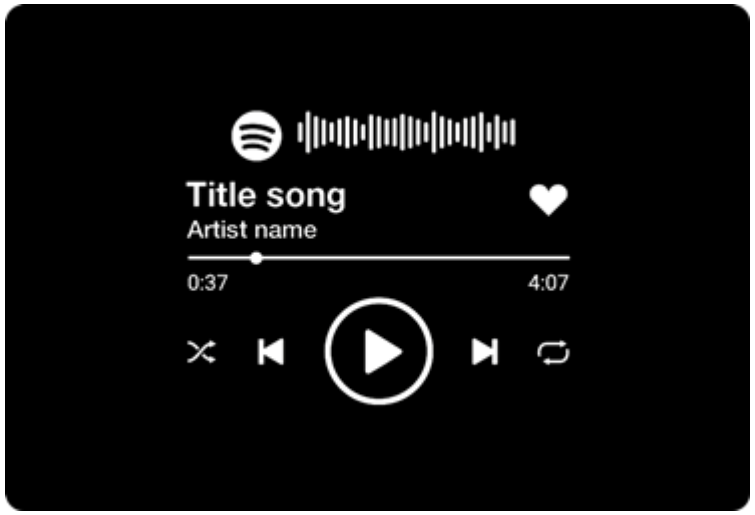




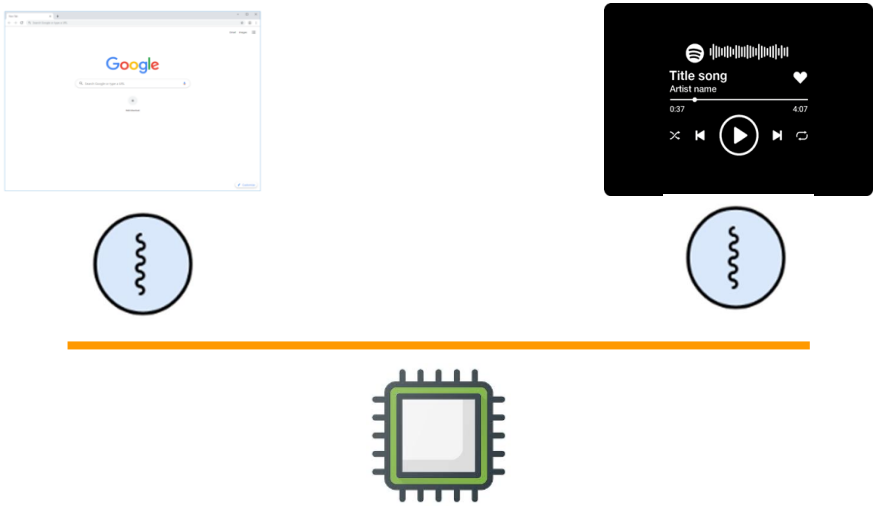
Programa .vs. Proceso

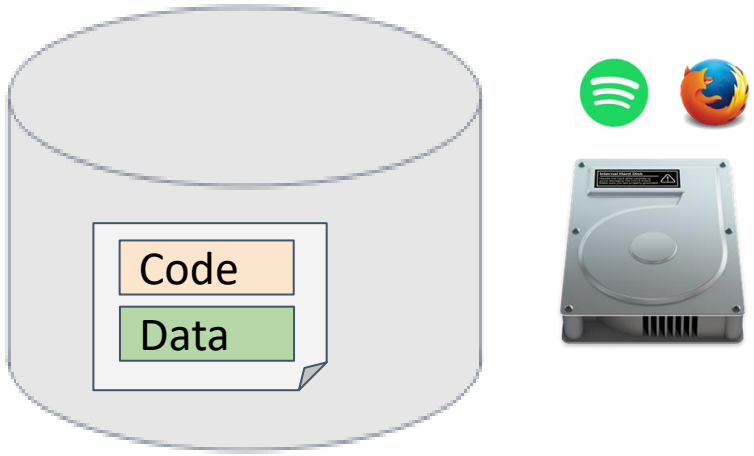
Programa

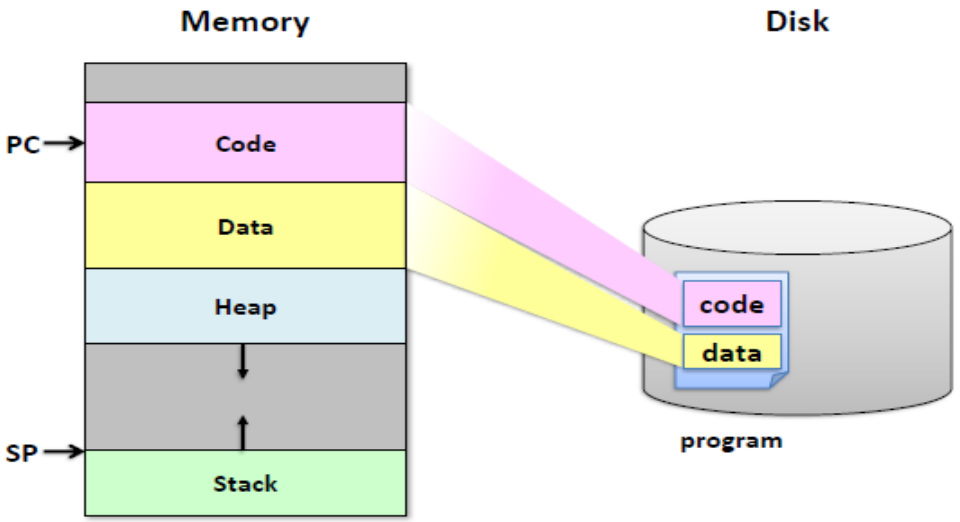
Proceso



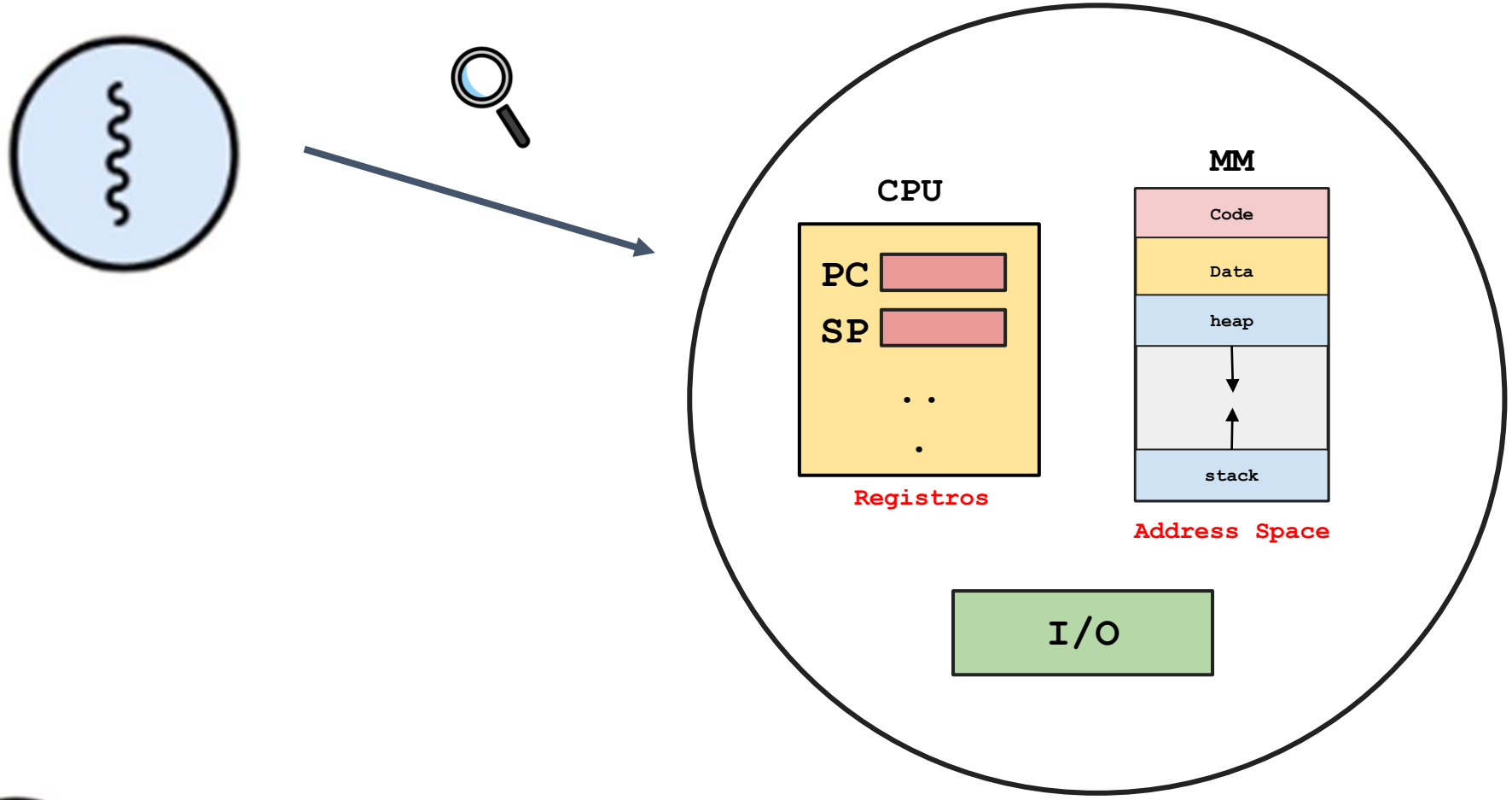
Programa. vs. proceso

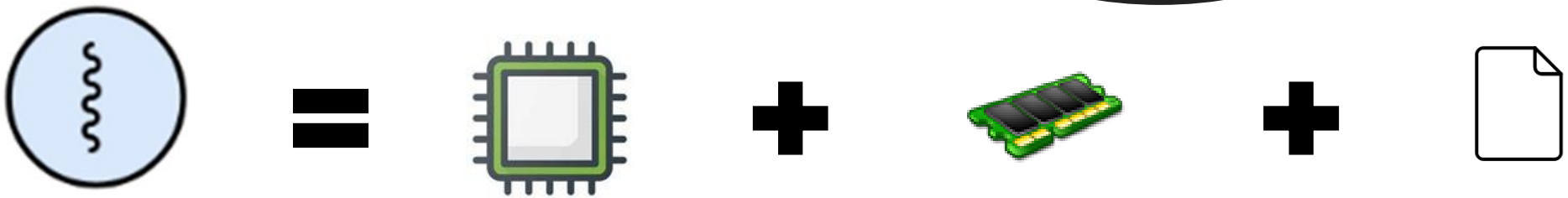




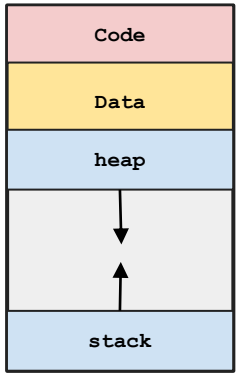


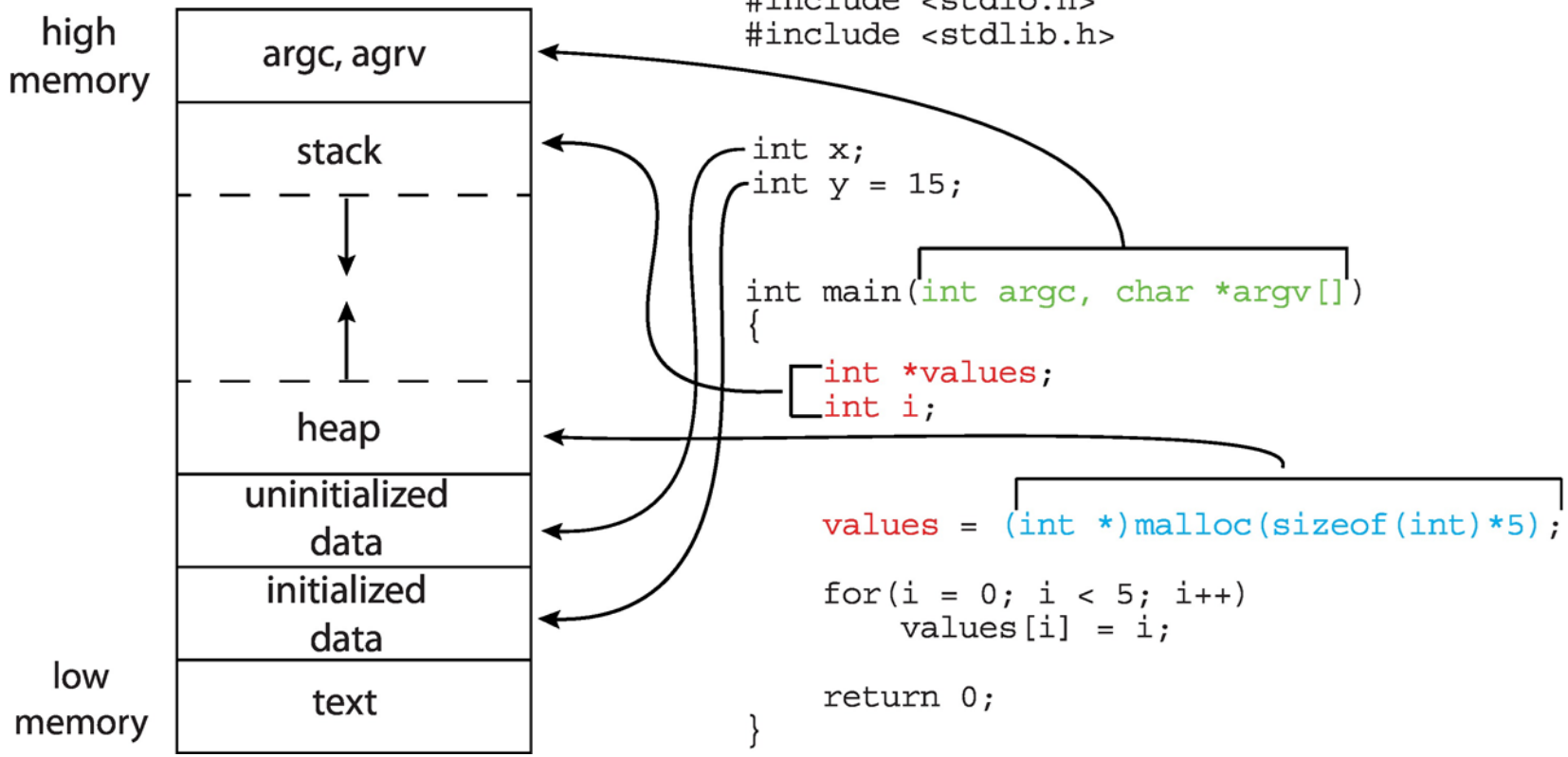
Proceso: Abstracción





Virtualización



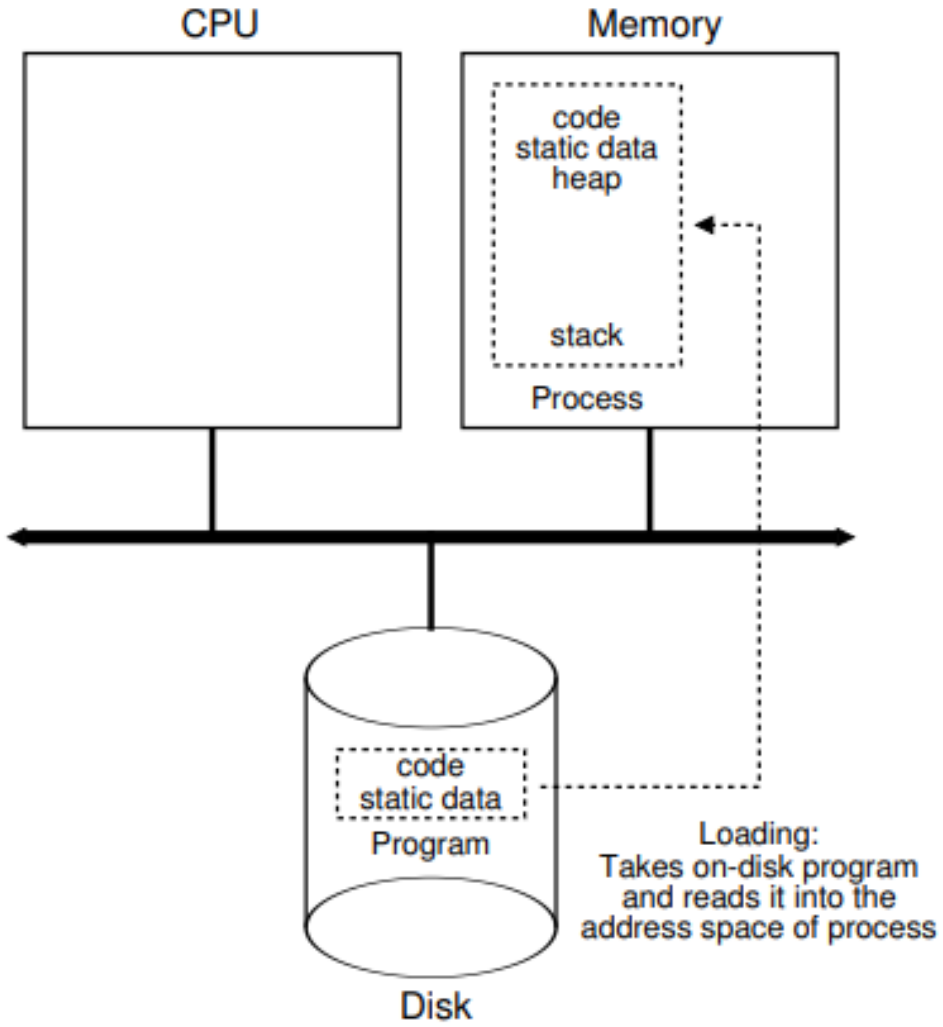


Simulación online (link)

Carga: De un programa a un proceso

Pasos

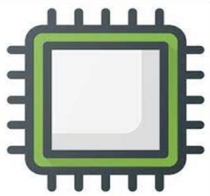
1. del código del programa a memoria en forma diferida
2. Asignación de memoria en la pila (**stack**) en tiempo de ejecución
3. Creación del espacio de memoria **heap**
4. Realización de tareas de inicialización I/O
5. Inicio del programa ejecutando el punto de entrada main()

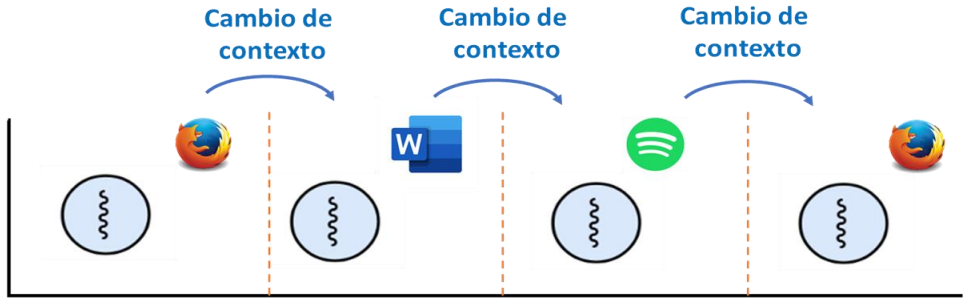


Volvamos a la pregunta

¿Cómo es que teniendo una sola CPU (en teoría) puedo correr varias tareas a la vez?

Técnica de **time sharing**





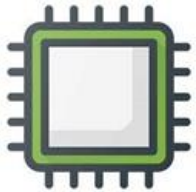
Agenda

- Repaso clase anterior
- Contexto
- ¿Que es un proceso?
- ¿Como se crea un proceso?
- **Elementos de los sistemas operativos**

Elementos de los sistemas operativos

Estos tres conceptos son los pilares que permiten que el software interactúe con el hardware sin caos:

1. **Abstracción (lo que "ve" el usuario)**: Interfaz simplificada que oculta la complejidad física del hardware.
   - process, thread, file, socket, memory page.
2. **Mecanismos (el "como" se hace):** representa las herramientas o funciones de bajo nivel que permiten implementar una operación.
   - create, schedule, open, write, locate.
3. **Políticas (El "quien" y "cuando")**: es el conjunto de reglas o algoritmos que deciden cómo se utilizan los mecanismos para cumplir un objetivo.
   - Last Recently Used (LRU), Earliest Deadline First (EDF), etc.

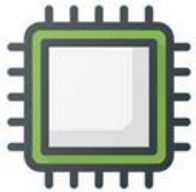


Process management	**Abstracción**	Proceso
	Mecanismos	create, destroy, wait, status,...
	Políticas	FIFO, RR,...

Elementos de los sistemas operativos

Estos tres conceptos son los pilares que permiten que el software interactúe con el hardware sin caos:

1. **Abstracción (lo que "ve" el usuario)**: Interfaz simplificada que oculta la complejidad física del hardware.
   - process, thread, file, socket, memory page.
2. **Mecanismos (el "como" se hace):** representa las herramientas o funciones de bajo nivel que permiten implementar una operación.
   - create, schedule, open, write, locate.
3. **Políticas (El "quien" y "cuando")**: es el conjunto de reglas o algoritmos que deciden cómo se utilizan los mecanismos para cumplir un objetivo.
   - Last Recently Used (LRU), Earliest Deadline First (EDF), etc.

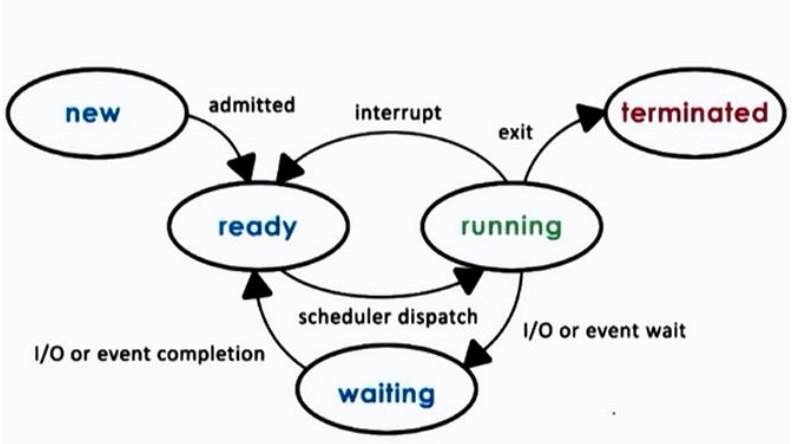


Process management	**Abstracción**	Proceso
	Mecanismos	create, destroy, wait, status,...
	Políticas	FIFO, RR,...

Como implementar la Abstracción

La Abstracción (El "Qué")

- **Definición:** Modelo simplificado que oculta la complejidad del hardware.
- **Objetivo:** Transformar recursos físicos (voltajes, sectores) en objetos lógicos.
- **Ejemplos:** Archivos, Procesos, Espacios de Direccionamiento.



La API (El "Contrato")

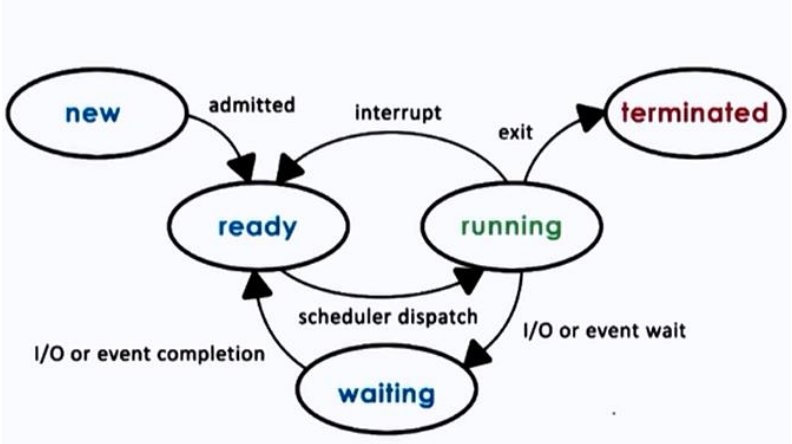
- **Definición:** Conjunto de funciones y protocolos (**punto de entrada**) para interactuar con el Kernel.
- **Rol:** Es la **formalización de la abstracción**. Permite la portabilidad (ej. POSIX).
- **Seguridad:** Única vía legal para pasar de *Modo Usuario* a *Modo Privilegiado*.

Función	**Descripción**
`Crear (create)`	Crea un nuevo proceso para ejecutar un programa.
`Eliminar (destroy)`	Forzar la detención de un programa en ejecución.
`Esperar (wait)`	Espera que un proceso termine su ejecución.
`Operaciones de control`	Ej. Suspender un proceso.
`Estado (status)`	Información de estado del proceso.

Como implementar la Abstracción

La Abstracción (El "Qué")

- **Definición:** Modelo simplificado que oculta la complejidad del hardware.
- **Objetivo:** Transformar recursos físicos (voltajes, sectores) en objetos lógicos.
- **Ejemplos:** Archivos, Procesos, Espacios de Direccionamiento.

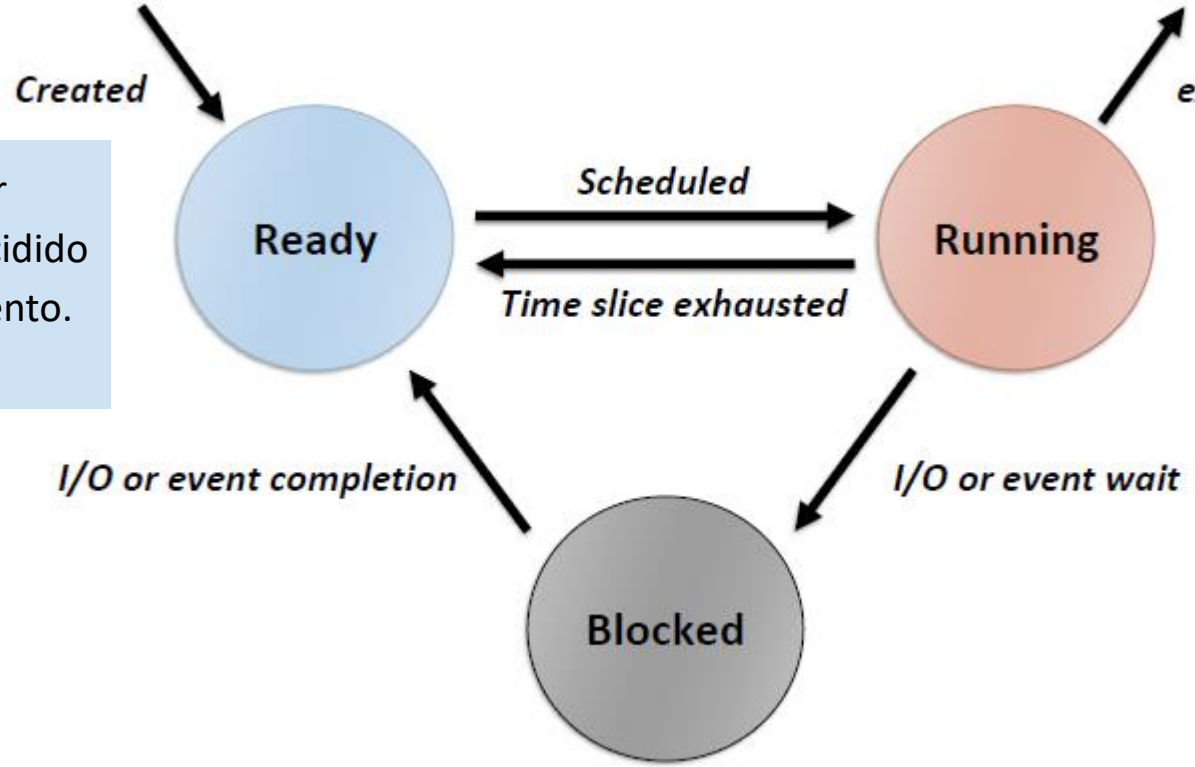


La API (El "Contrato")

- **Definición:** Conjunto de funciones y protocolos (**punto de entrada**) para interactuar con el Kernel.
- **Rol:** Es la **formalización de la abstracción**. Permite la portabilidad (ej. POSIX).
- **Seguridad:** Única vía legal para pasar de *Modo Usuario* a *Modo Privilegiado*.

Función	**Descripción**
`Crear (create)`	Crea un nuevo proceso para ejecutar un programa.
`Eliminar (destroy)`	Forzar la detención de un programa en ejecución.
`Esperar (wait)`	Espera que un proceso termine su ejecución.
`Operaciones de control`	Ej. Suspender un proceso.
`Estado (status)`	Información de estado del proceso.

Modelo de tres estados

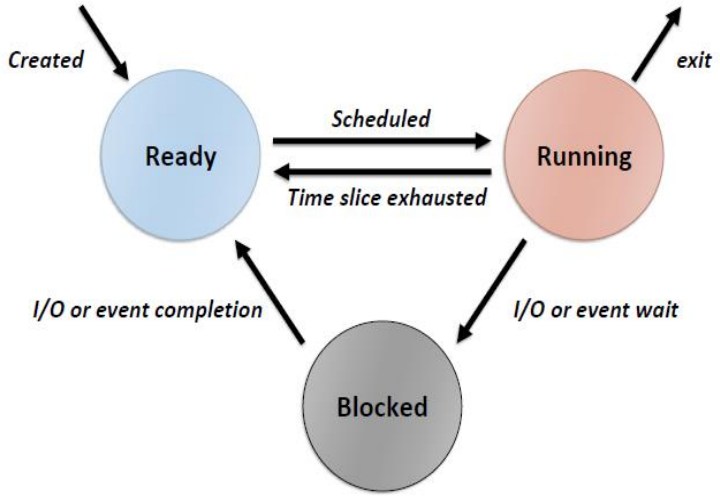


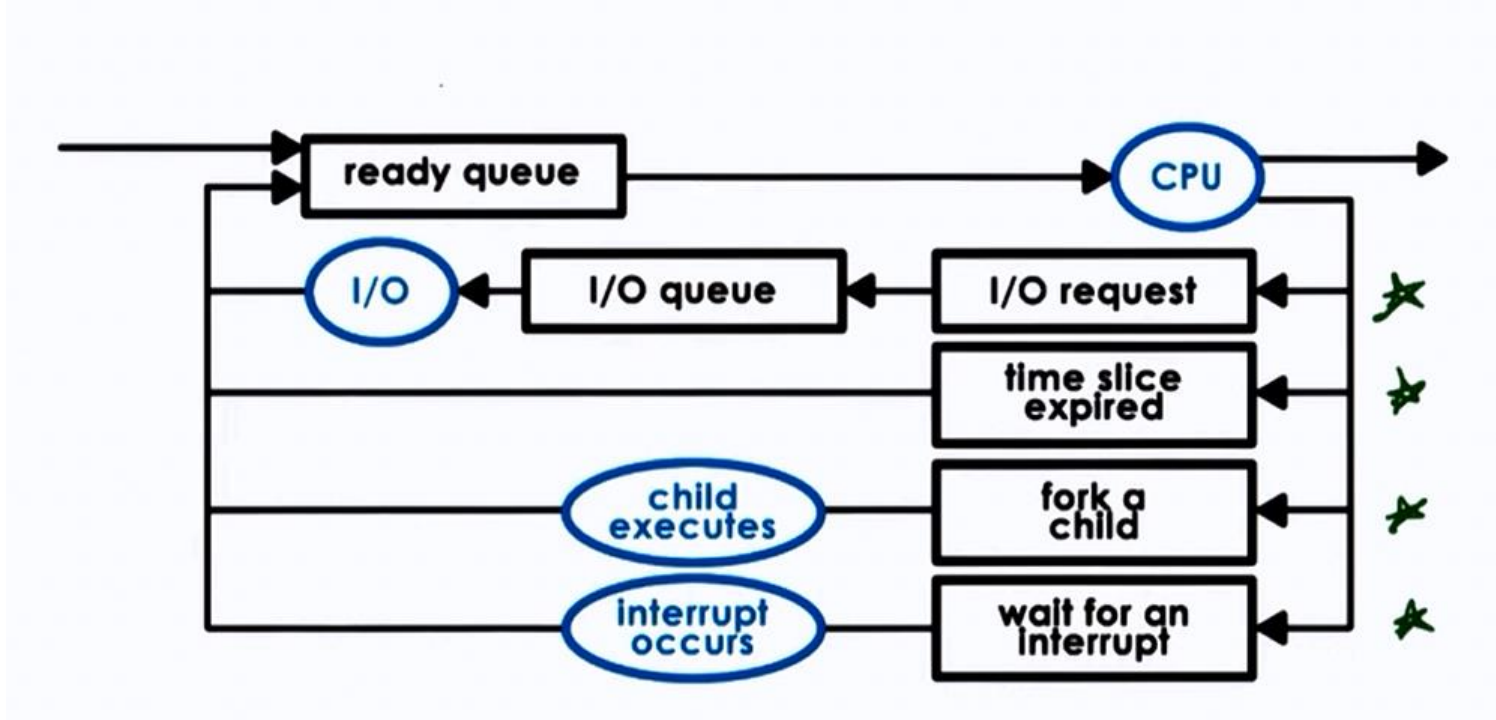
El proceso está listo para ser ejecutado pero el SO ha decidido no ejecutarlo en este momento.

El proceso se ejecuta en el procesador

El proceso desarrolla alguna operación (I/O disco).

Modelo de tres estados

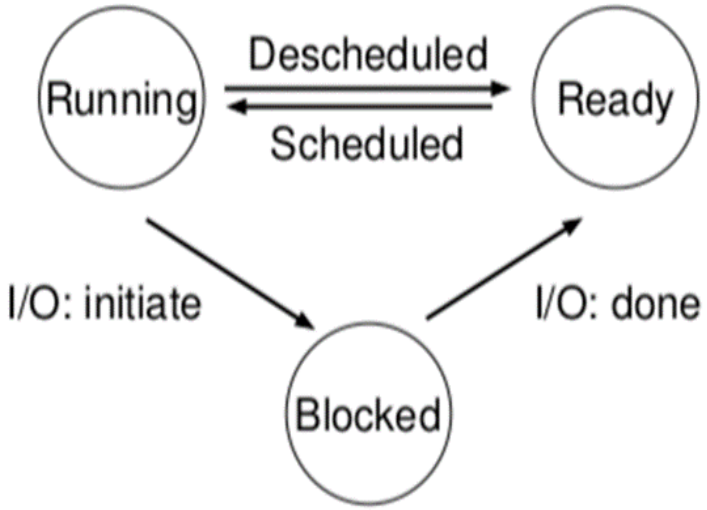




Estados de un proceso – Ejemplo 1

Time	$Process_0$	$Process_1$	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	
4	Running	Ready	$Process_0$ now done
5	–	Running	
6	–	Running	
7	–	Running	
8	–	Running	$Process_1$ now done

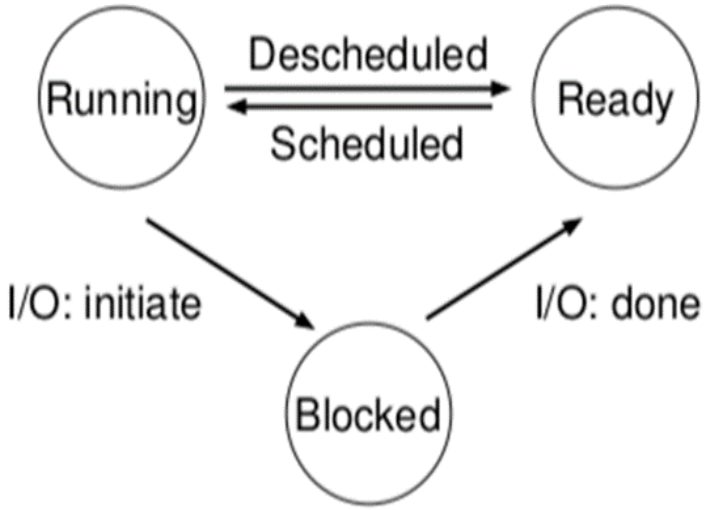
Tracing Process State: CPU Only



Estados de un proceso – Ejemplo 2

Time	$Process_0$	$Process_1$	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	$Process_0$ initiates I/O
4	Blocked	Running	$Process_0$ is blocked,
5	Blocked	Running	so $Process_1$ runs
6	Blocked	Running	
7	Ready	Running	I/O done
8	Ready	Running	$Process_1$ now done
9	Running	–	
10	Running	–	$Process_0$ now done

Tracing Process State: CPU and I/O



Modelo Linux

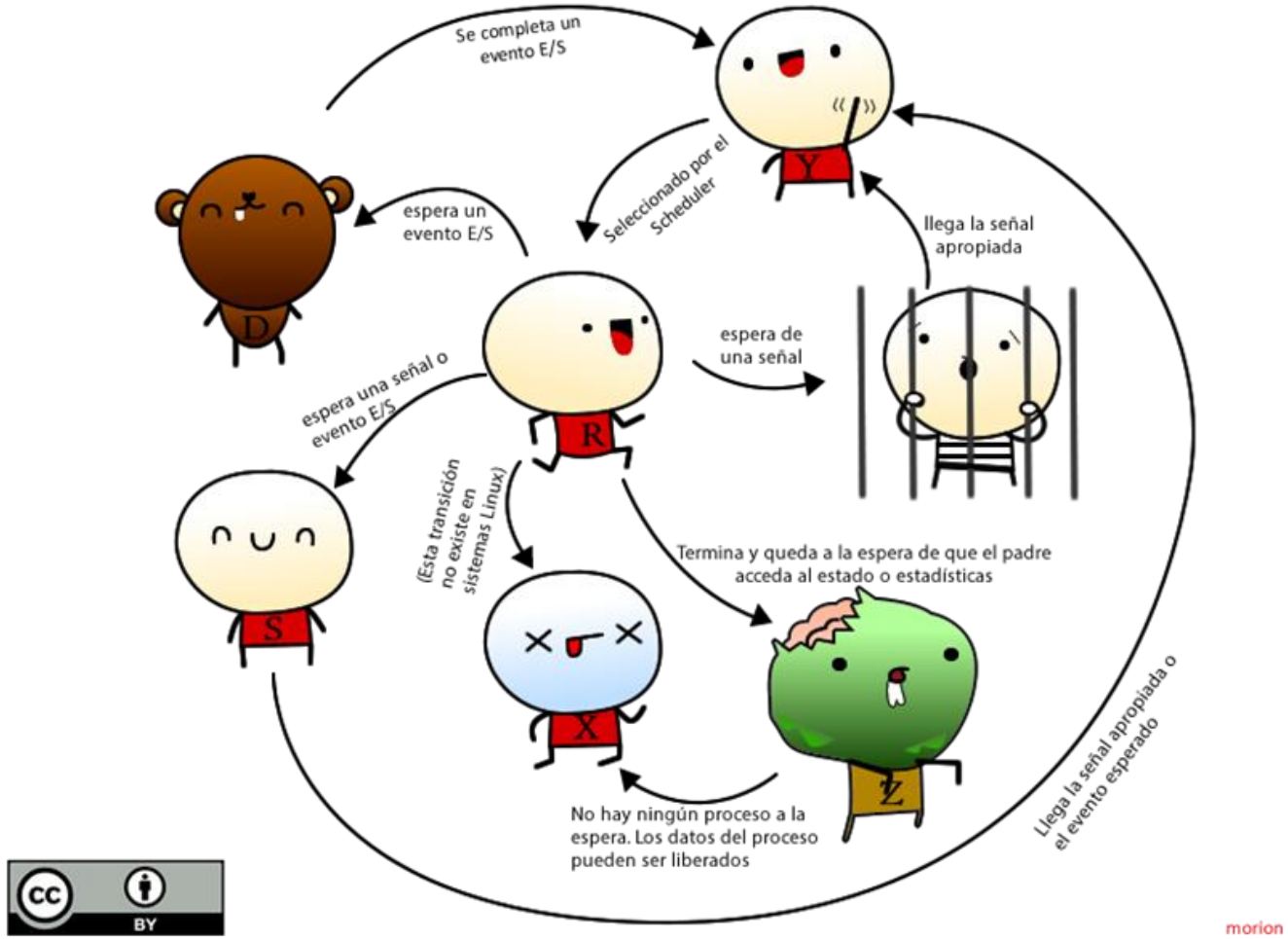
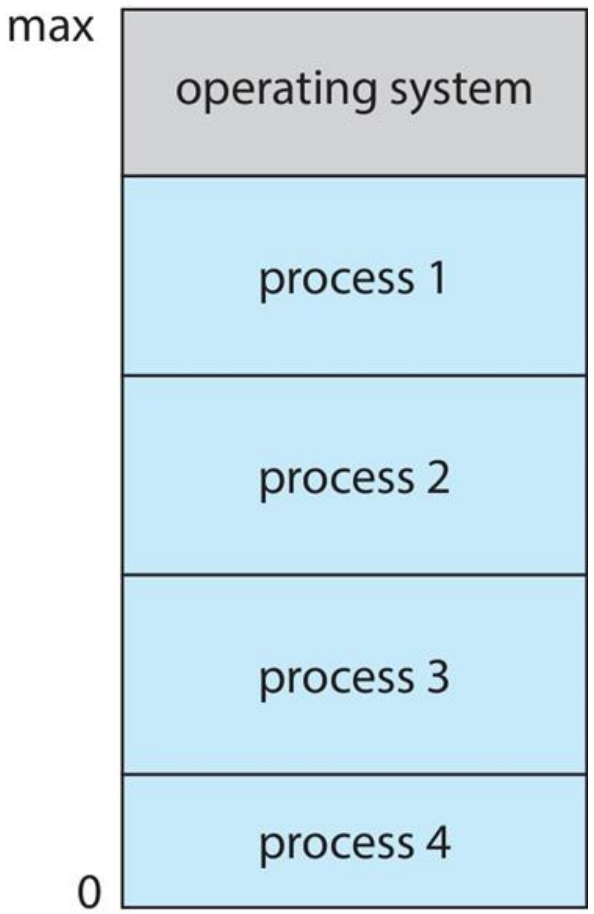


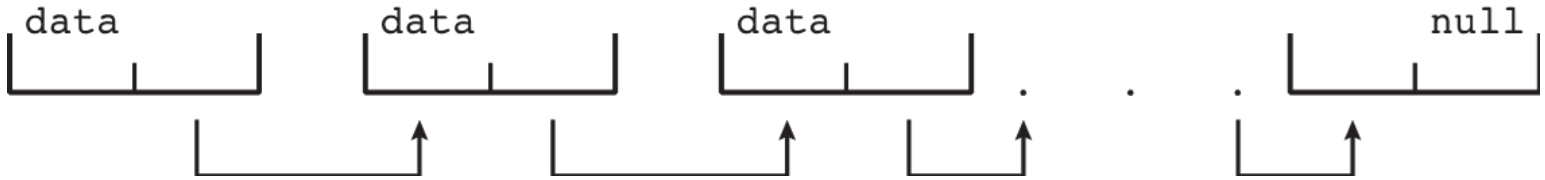
Imagen realizada por Adrián Martínez

Pasando al código – Estructuras de datos

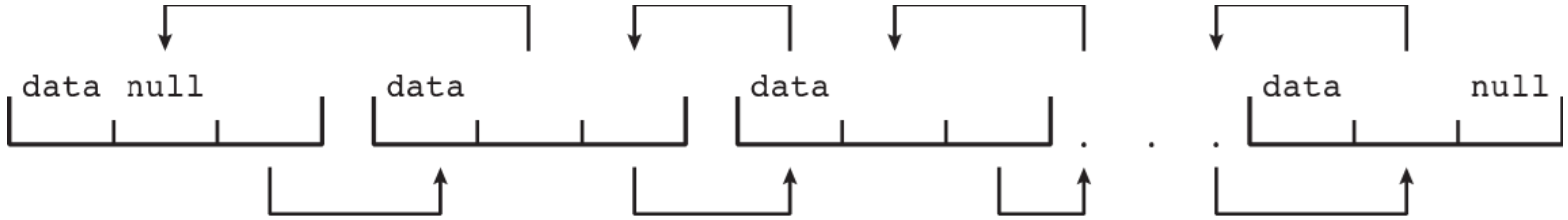
- El sistema operativo requiere **estructuras de datos** para rastrear los elementos claves.



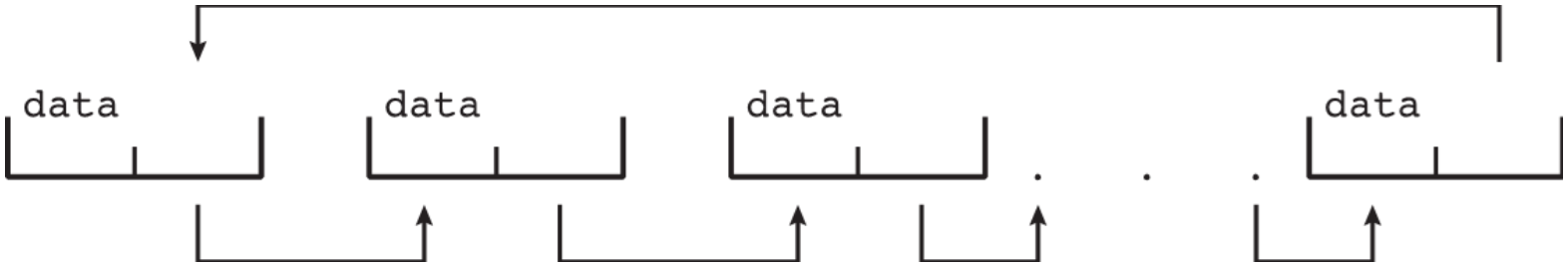
Single linked list



Double linked list

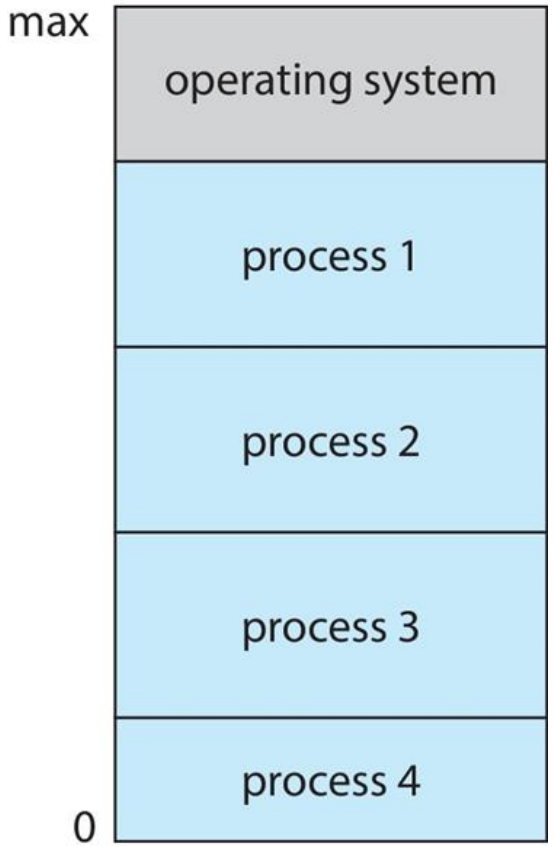


Circular linked list

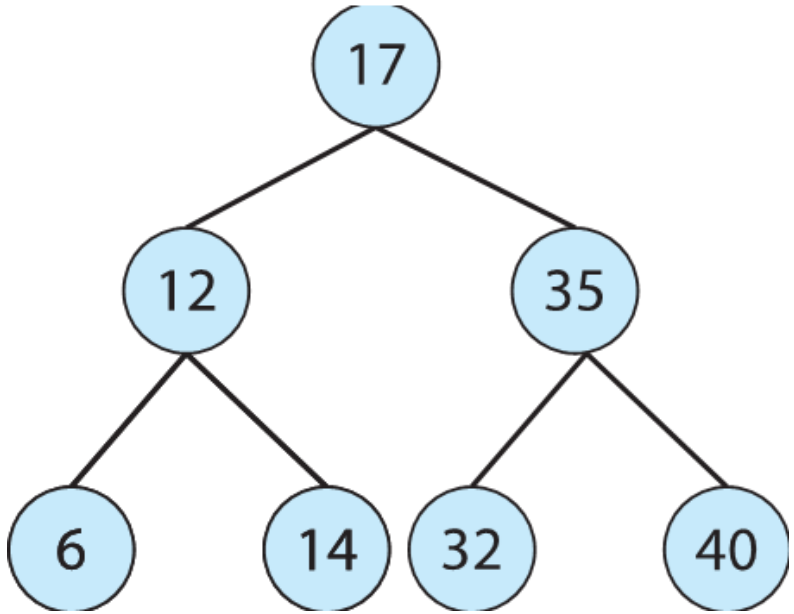


Pasando al código – Estructuras de datos

- El sistema operativo requiere **estructuras de datos** para rastrear los elementos claves.

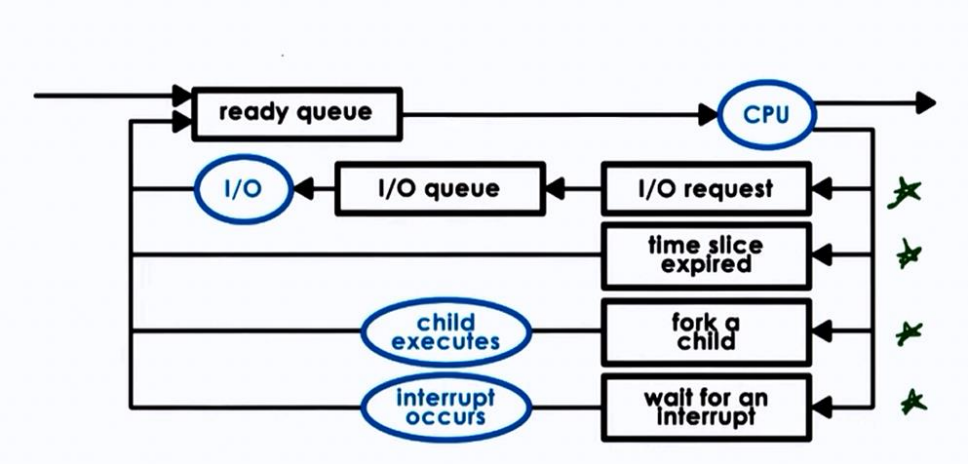


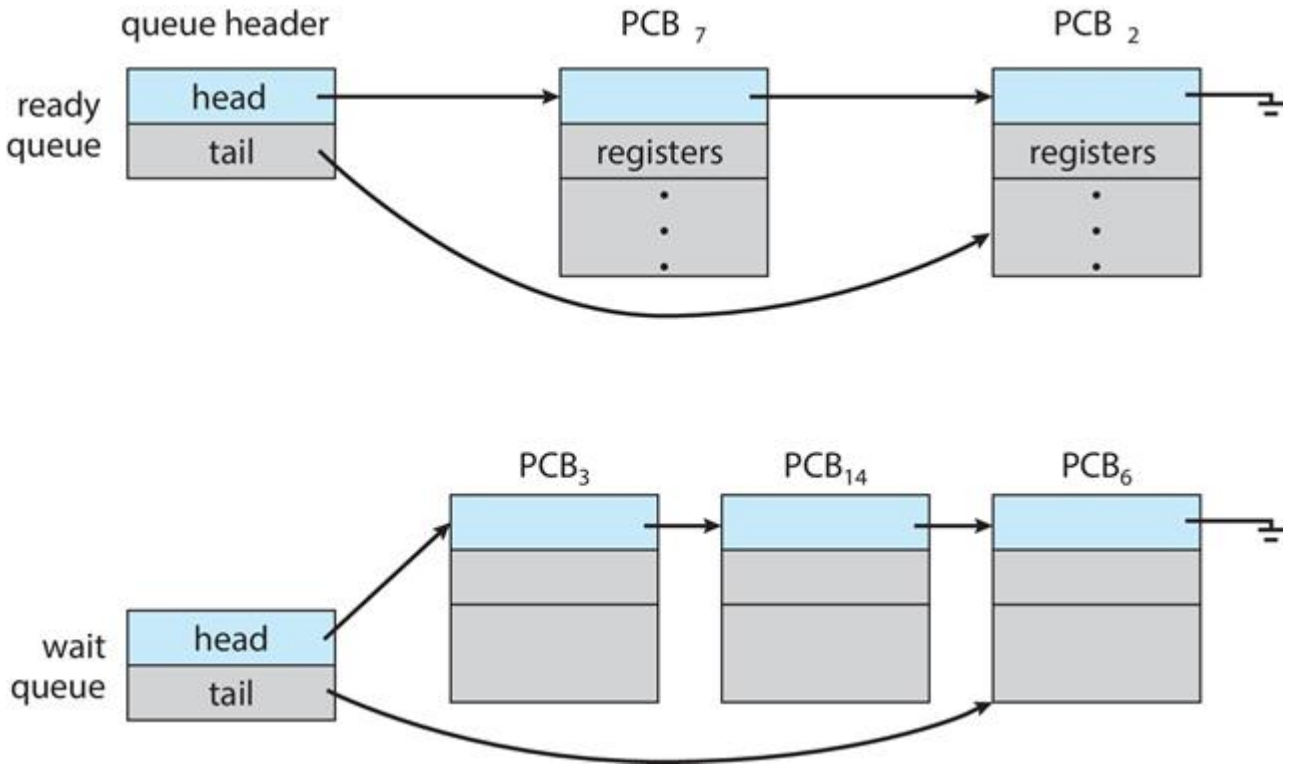
Binary search three



Lista de procesos

- Procesos listos
- Procesos bloqueados
- Procesos en ejecución

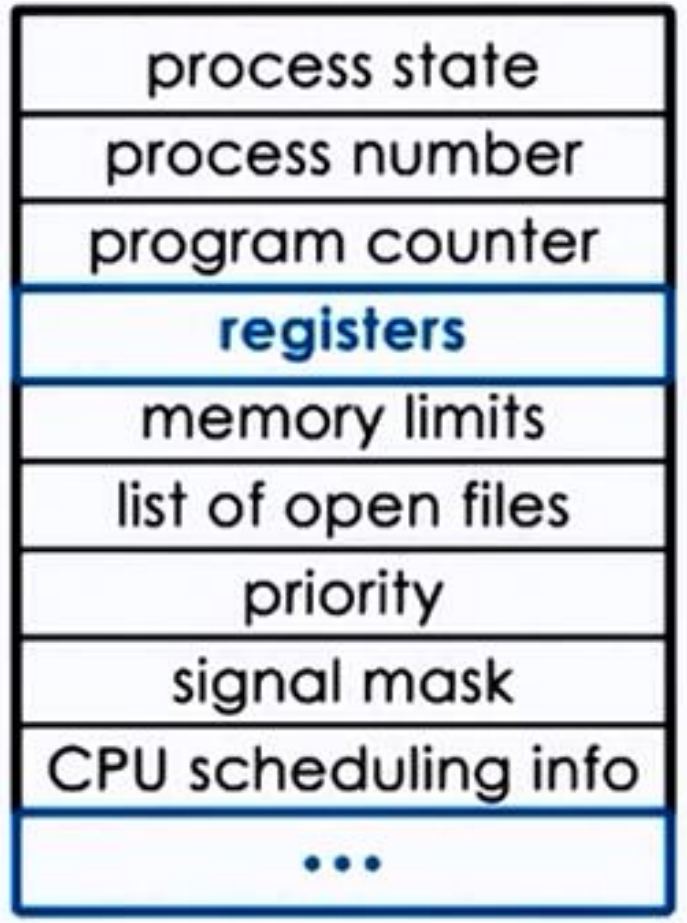




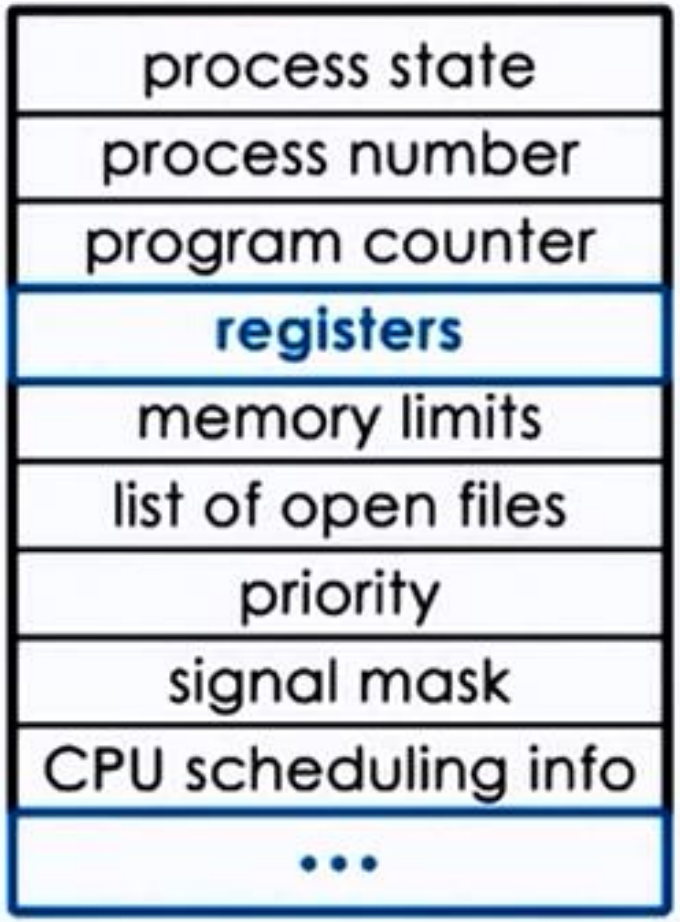
Process Control Block (PCB)

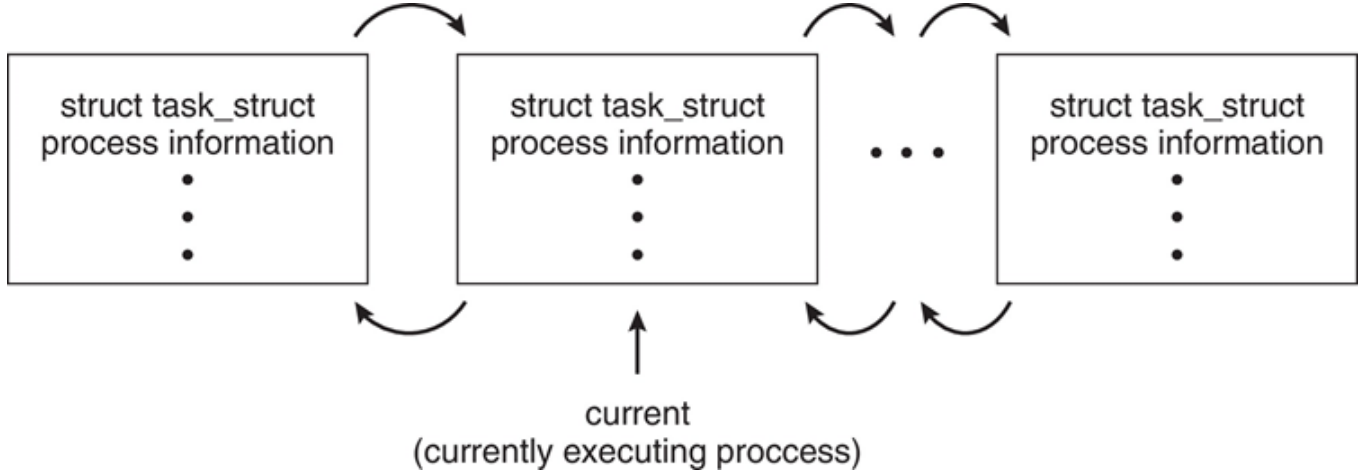
Información acerca de cada proceso (estructura en C).

- **Estado de ejecución del programa**: running, ready, etc.
- **Program counter**: Localización de la próxima instrucción a ejecutar.
- **Registros de la CPU**: Contenido de todos registros asociados al proceso.
- **Información de scheduling de la CPU**: Prioridades, etc.
- **Información de manejo de memoria**: Memoria asignada por el proceso.
- **Información (Accounting information)**: CPU usada, tiempo gastado desde el inicio, limites de tiempo.
- **Información de estado de I/O**: Dispositivos I/O asociados al proceso, lista de archivos abiertos.



Process Control Block (PCB)

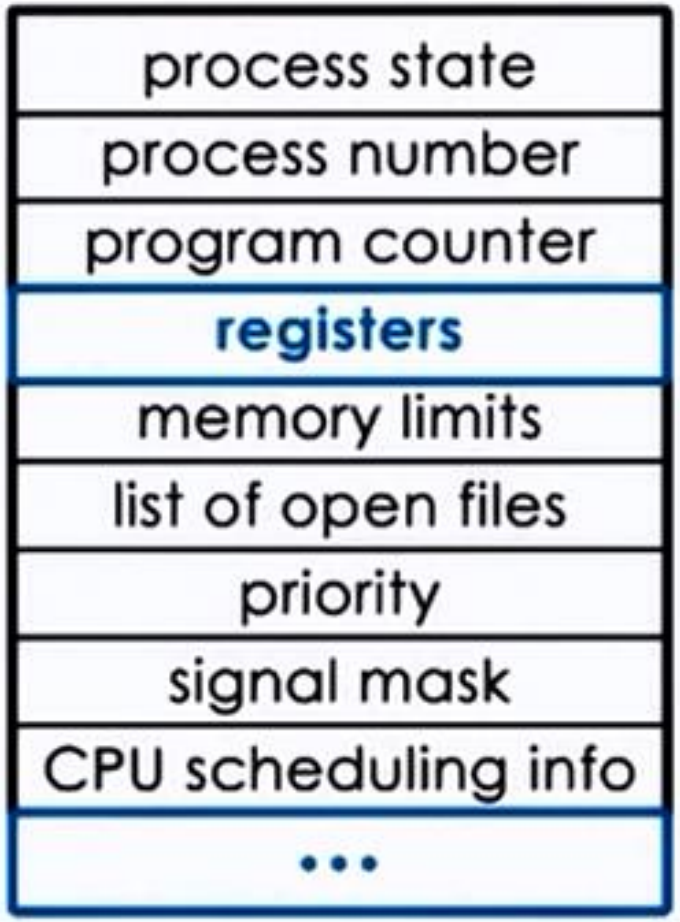


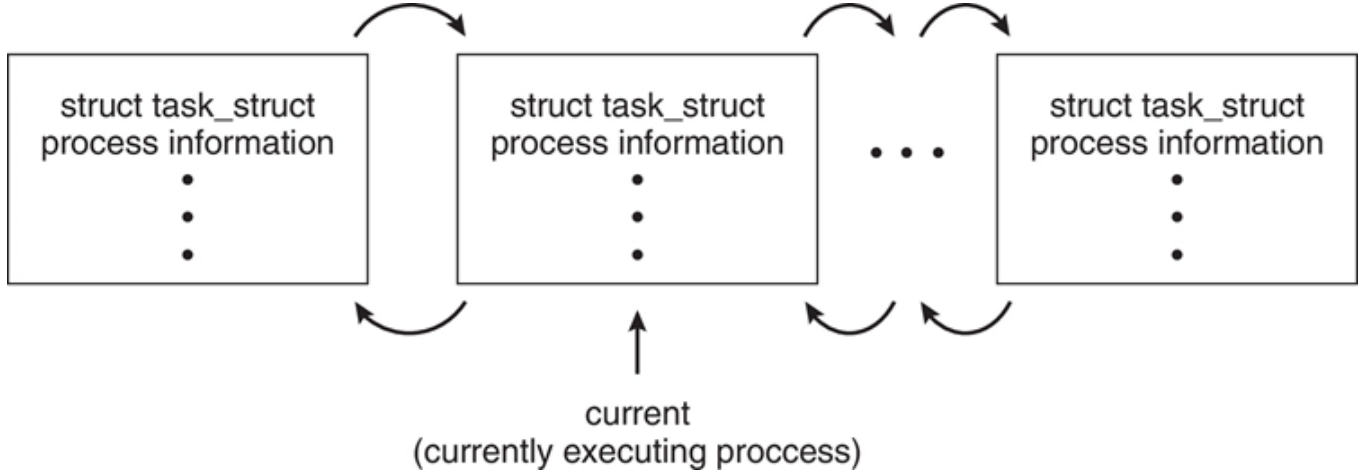



```
pid t_pid;    /* process identifier */
long state;    /* state of the process */
unsigned int time_slice    /* scheduling information */
struct task_struct *parent;/* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files;/* list of open files */
struct mm_struct *mm;    /* address space of this process */
```

The xv6 Proc Structure (link)

Process Control Block (PCB)






```
pid t_pid;    /* process identifier */
long state;    /* state of the process */
unsigned int time_slice    /* scheduling information */
struct task_struct *parent;/* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files;/* list of open files */
struct mm_struct *mm;    /* address space of this process */
```

The xv6 Proc Structure (link)

Process Control Block (PCB)

```
// the registers xv6 will save and restore
// to stop and subsequently restart a process
struct context {
  int eip;
  int esp;
  int ebx;
  int ecx;
  int edx;
  int esi;
  int edi;
  int ebp;
};

// the different states a process can be in
enum proc_state { UNUSED, EMBRYO, SLEEPING,
                  RUNNABLE, RUNNING, ZOMBIE };
```

The xv6 Proc Structure (link)

Process Control Block (PCB)

```
// the information xv6 tracks about each process
// including its register context and state
struct proc {
  char *mem;                          // Start of process memory
  uint sz;                            // Size of process memory
  char *kstack;                       // Bottom of kernel stack
                                      // for this process
  enum proc_state state;              // Process state
  int pid;                            // Process ID
  struct proc *parent;                // Parent process
  void *chan;                         // If !zero, sleeping on chan
  int killed;                         // If !zero, has been killed
  struct file *ofile[NOFILE];         // Open files
  struct inode *cwd;                  // Current directory
  struct context context;             // Switch here to run process
  struct trapframe *tf;               // Trap frame for the
                                      // current interrupt
};
```

The xv6 Proc Structure (link)