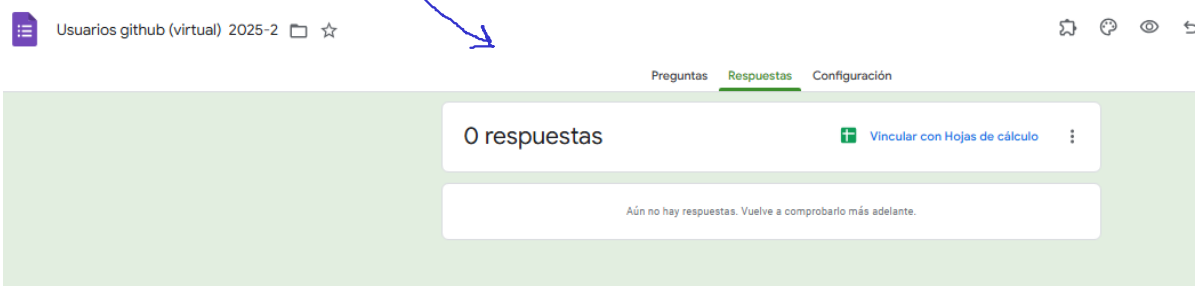
26/08/2025 - Sistemas Operativos (Vde@)

1. Sobre los avisos

Debate	Comenzado por	Último mensaje ↑	Réplicas
Bienvenida	HENRY ALBERTO ... 11 ago 2025	HENRY ALBERTO ... 11 ago 2025	0
Pendientes	HENRY ALBERTO ... 13 ago 2025	HENRY ALBERTO ... 13 ago 2025	0
Apuntes clase 2	HENRY ALBERTO ... 14 ago 2025	HENRY ALBERTO ... 14 ago 2025	0
Red Hat Academy	HENRY ALBERTO ... 18 ago 2025	HENRY ALBERTO ... 18 ago 2025	0
Apuntes clase 3	HENRY ALBERTO ... 19 ago 2025	HENRY ALBERTO ... 19 ago 2025	0
Problemas con el equipo	HENRY ALBERTO ... 21 ago 2025	HENRY ALBERTO ... 21 ago 2025	0
Quiz 2 disponible	HENRY ALBERTO ... 23 ago 2025	HENRY ALBERTO ... 23 ago 2025	0

Ojo

* Tener esto listo para el dia del laboratorio.

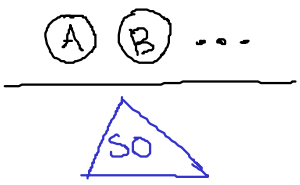


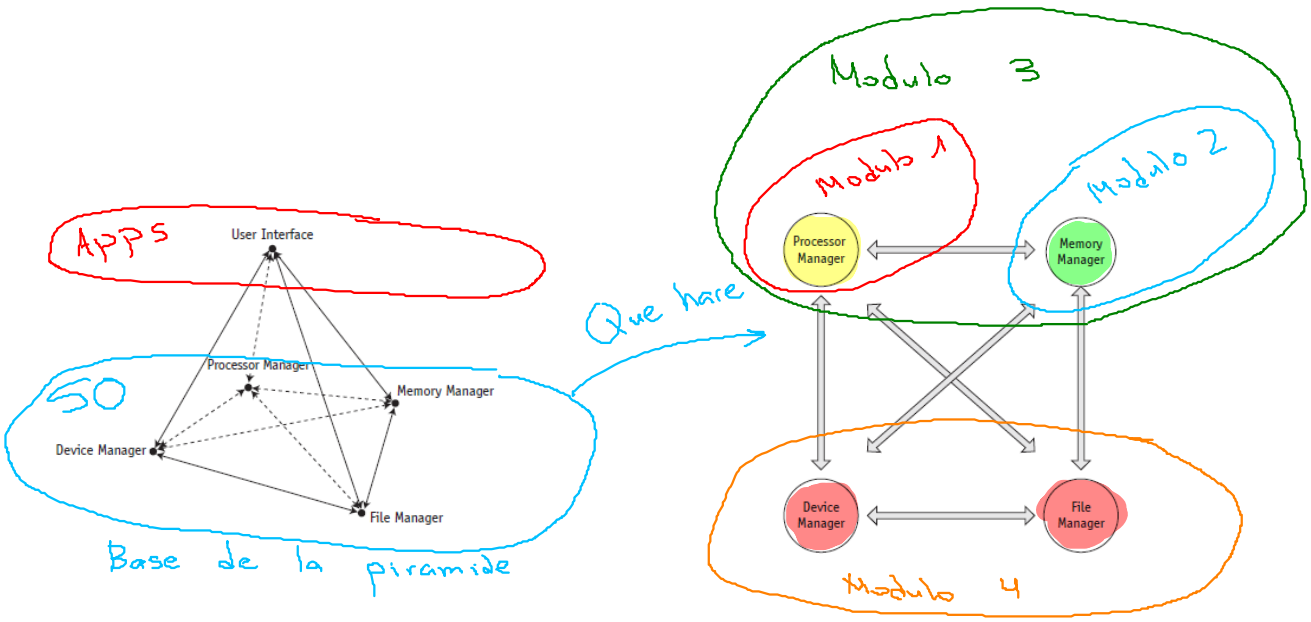
2. Repaso con imagenes

a. Definicion de Sistema Operativo

App: software normal

SO: Software especial

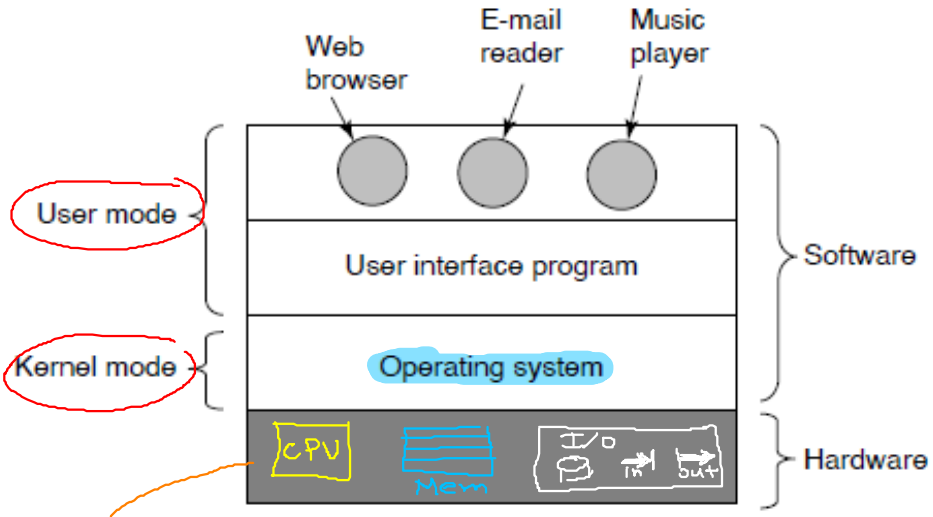




b. Interacción Software – Hardware

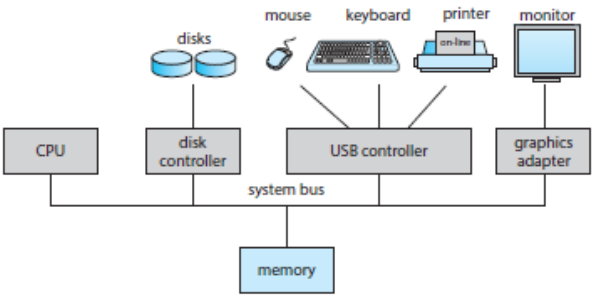
- Capas

Conceptos de la clase de hoy

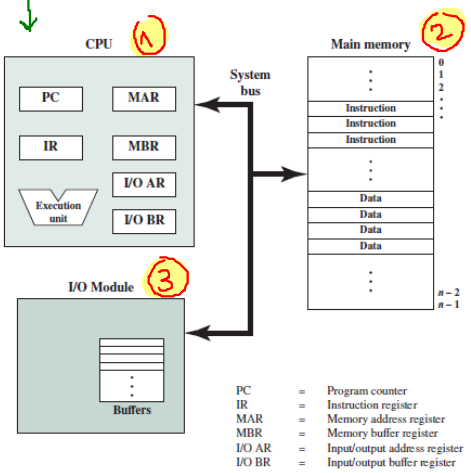


Hw. administrado por el Sistema Operativo

Representación Resumida del Hw como bloques



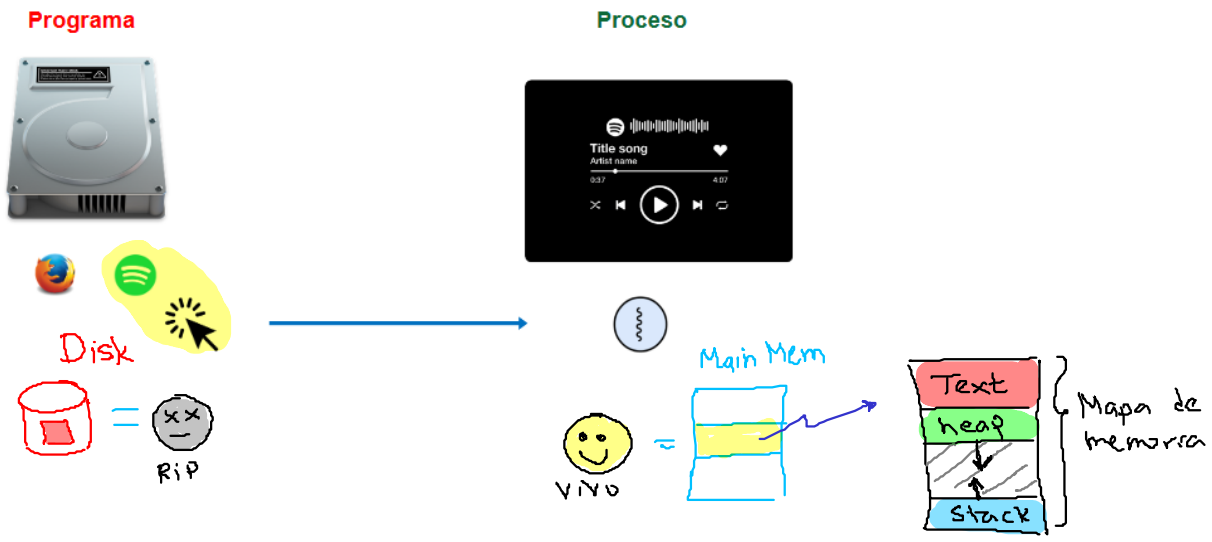
A typical PC computer system.



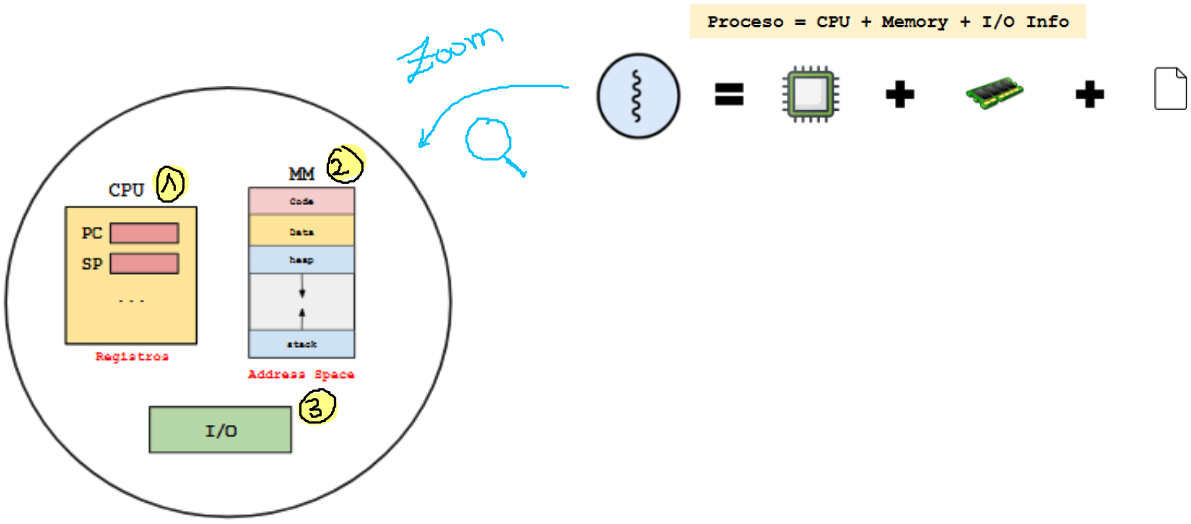
Computer Components: Top-Level View

c. Proceso

* Distinción importante : Programa ≠ Proceso



Abstracción: El Proceso



* Estados de un proceso: De alguna manera un proceso es como un ser vivo.

Modelo de cinco estados (Silverchatz, Stallings)

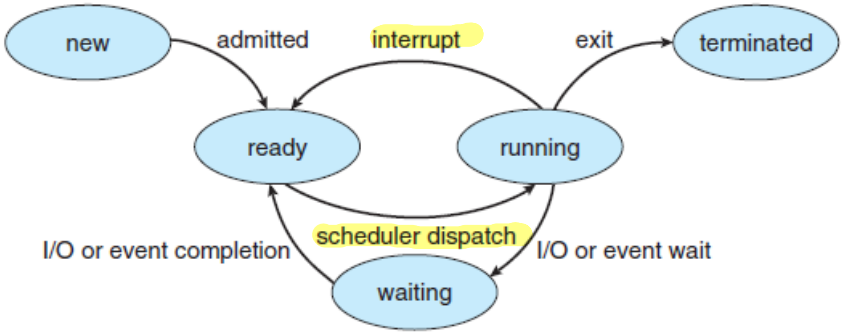
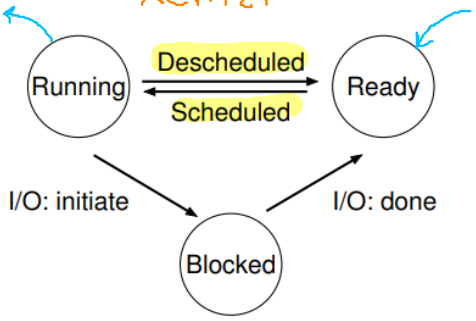


Diagram of process state.

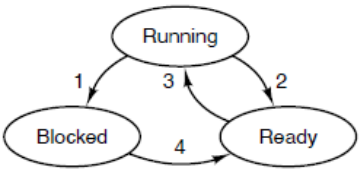
Modelo de tres estados

Remzi



Process: State Transitions

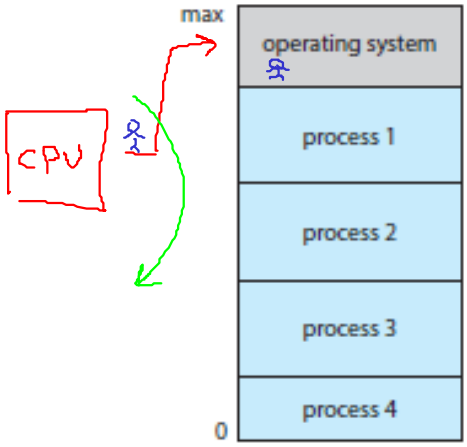
Tanembaum



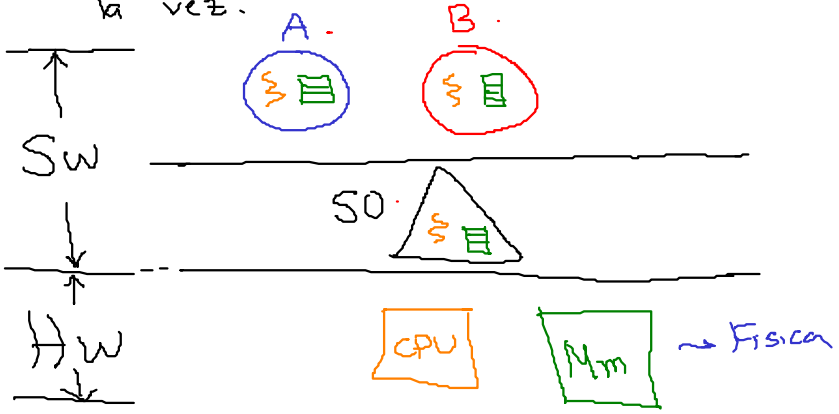
1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

A process can be in running, blocked, or ready state. Transitions between these states are as shown.

* Proceso y Virtualización

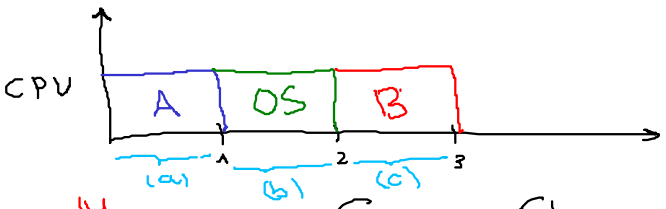


Multiprogramación: Ilusión de que varios programas se ejecutan a la vez.

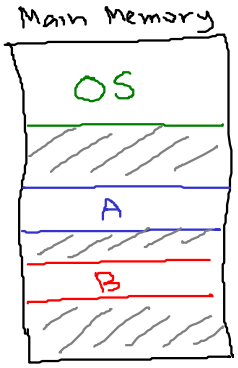


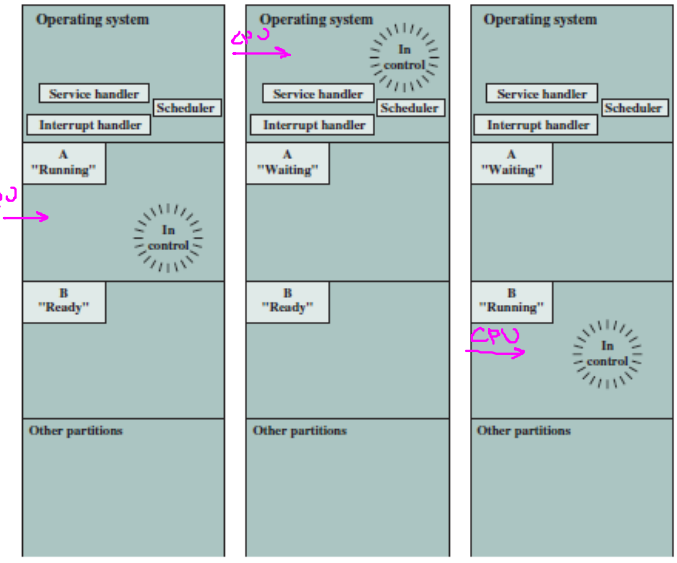
Memory layout for a multiprogramming system.

CPU: Time Sharing

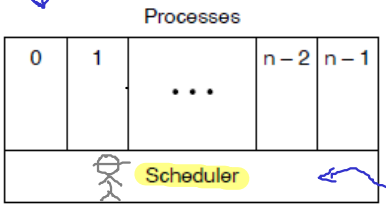


Memoria: Space Sharing





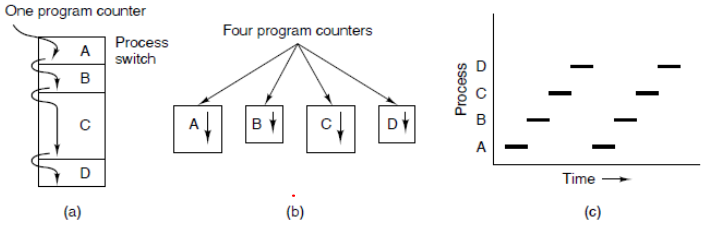
Scheduling Example



Parte del SO

The lowest layer of a process-structured operating system handles interrupts and scheduling. Above that layer are sequential processes.

* Ilusión . vs. Realidad

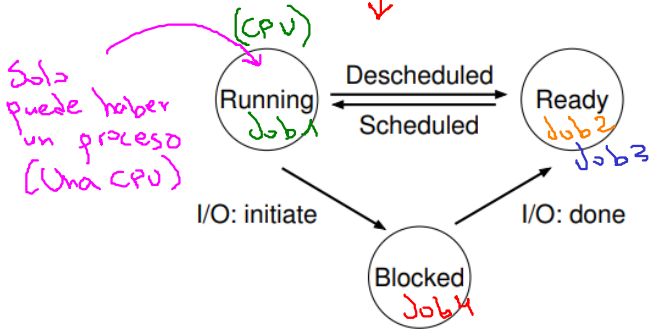


(a) Multiprogramming four programs. (b) Conceptual model of four independent, sequential processes. (c) Only one program is active at once.

CPU

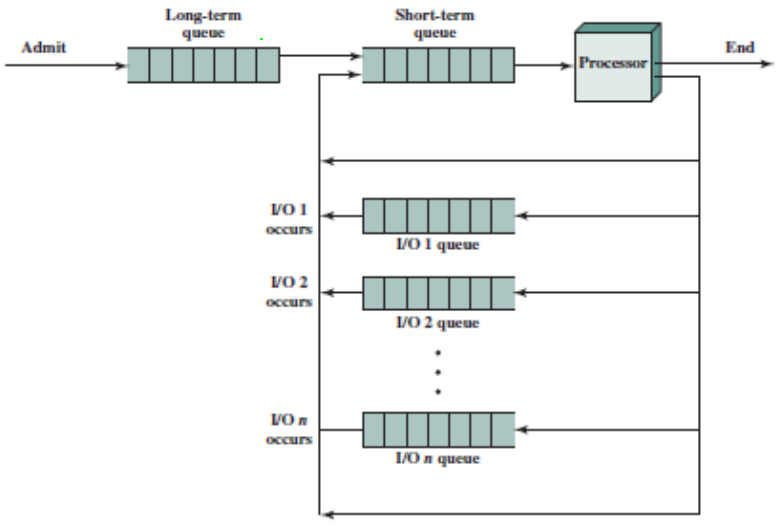
Modelo de 3 estados

Solo puede haber un proceso (Una CPU)



Process: State Transitions

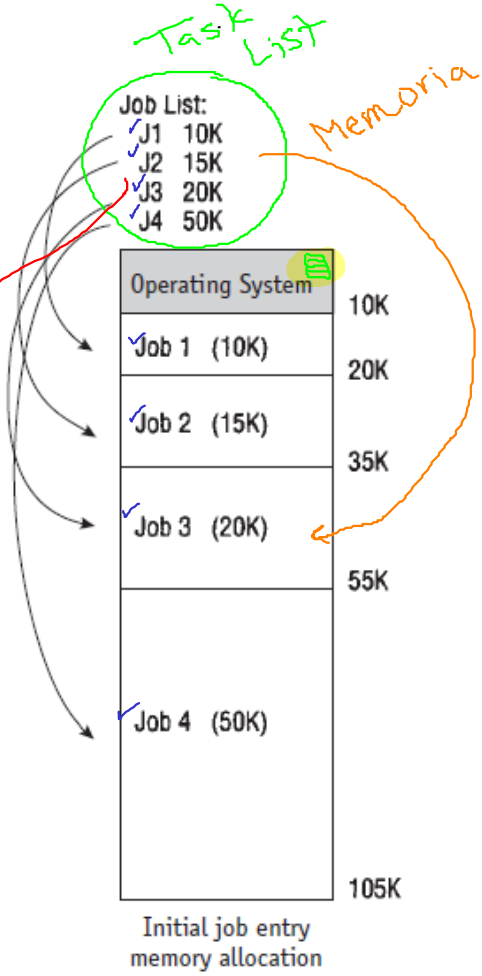
Como se implementa este modelo (Aprox).



Queuing Diagram Representation of Processor Scheduling

Task List

Memoria

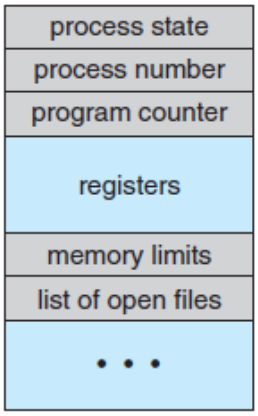


Initial job entry memory allocation

* Estructuras de datos asociadas

- Process control Block (PCB)

Zoom ?

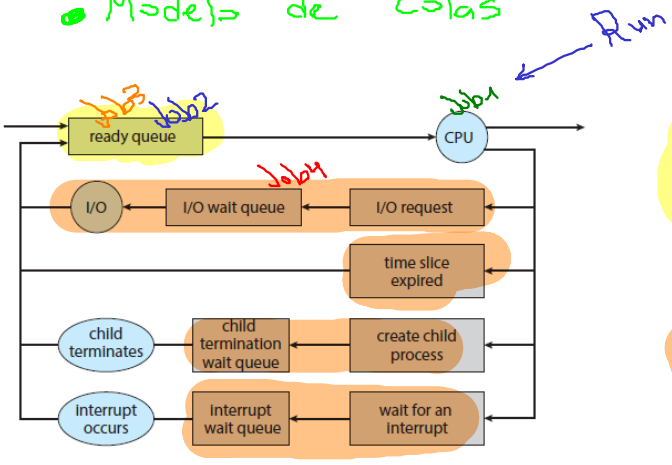


Process control block (PCB).

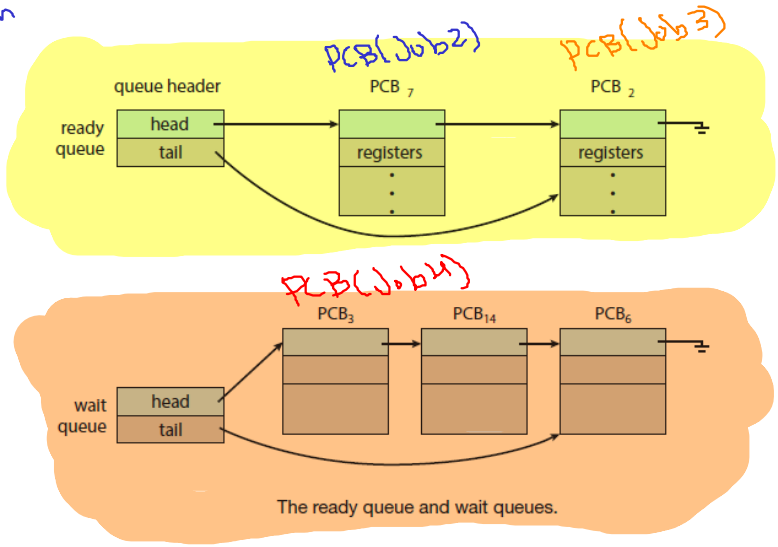
Process management	Memory management	File management
Registers	Pointer to text segment info	Root directory
Program counter	Pointer to data segment info	Working directory
Program status word	Pointer to stack segment info	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Some of the fields of a typical process-table entry.

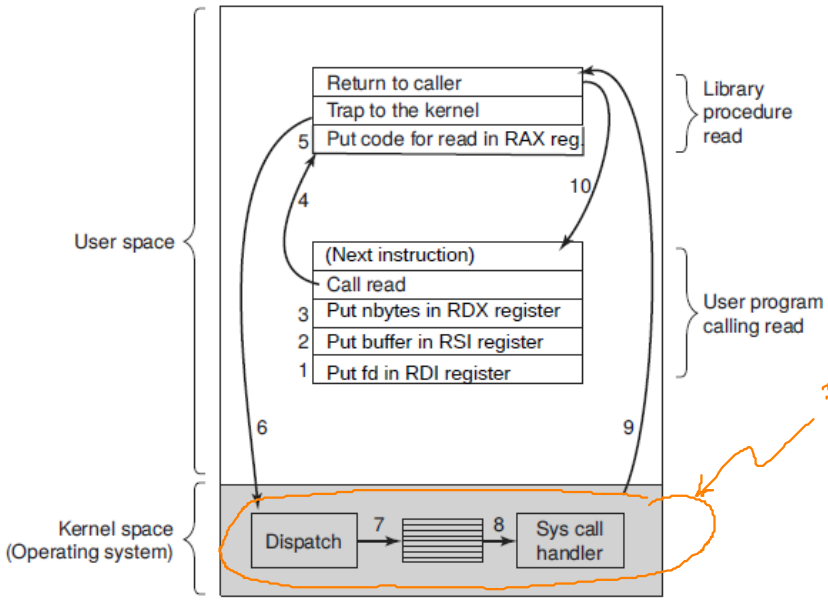
- Modelo de Colas



Queueing-diagram representation of process scheduling.



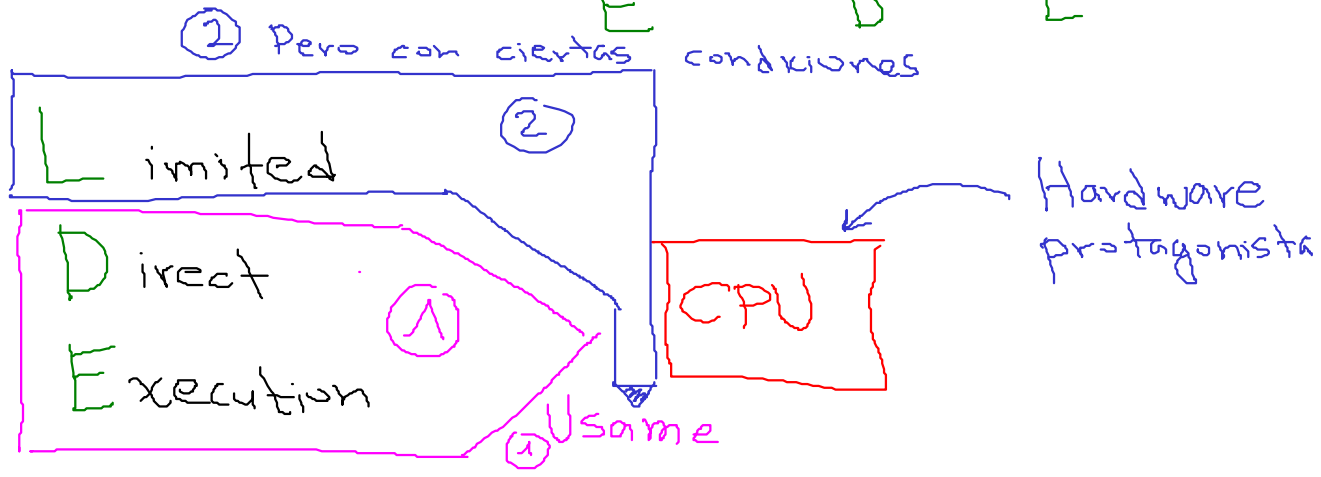
The ready queue and wait queues.



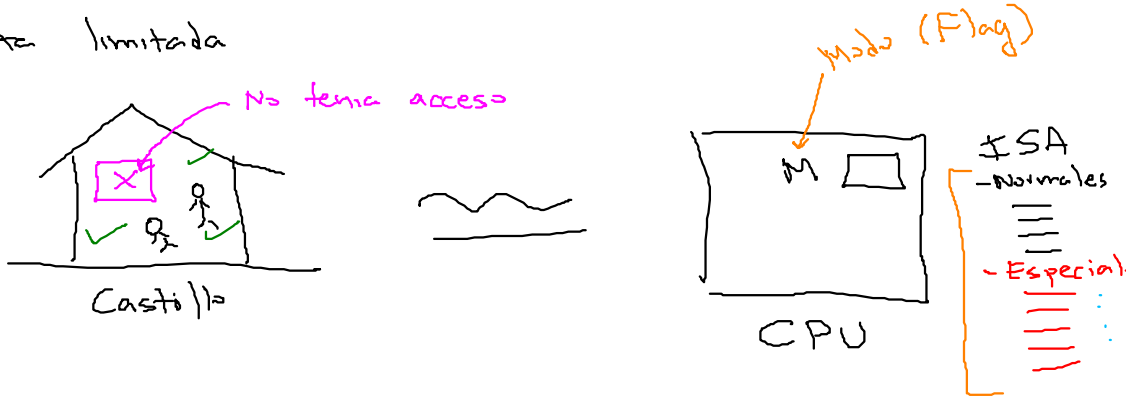
Estas estructuras de datos estan en espacio kernel.

3. Tema de Hoy (Ejecución directa limitada)

E D L



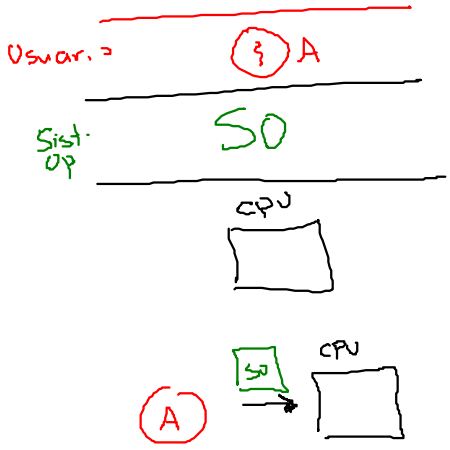
Ejecución directa limitada

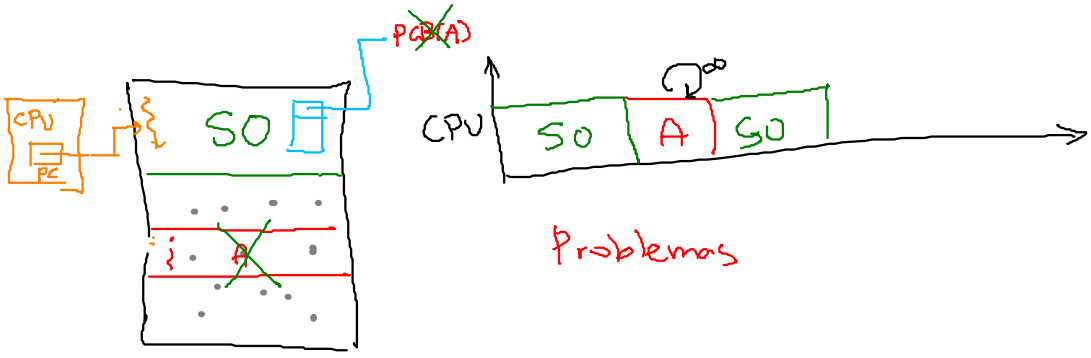


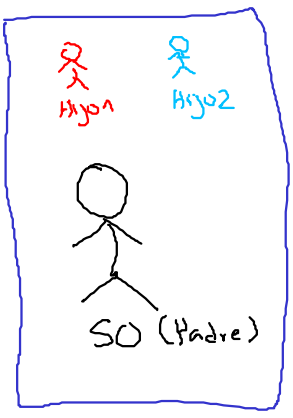
Como se usa la CPU entre los diferentes proceso?

1 Ejecución Directa

Sistema Operativo	Proceso de usuario
1. Crea una entrada en la **lista de procesos** ✓ 2. Asigna memoria al proceso ✓ 3. Carga el programa a memoria ✓ 4. Inicializa pila (*stack*) con `argc/argv` ✓ 5. Reinicia registros de CPU ✓ 6. Ejecuta llamada a `main()` ✓	
	7. Ejecuta `main()` 8. Ejecuta `return` de `main()` ↻ ∞
9. Libera la memoria del proceso 10. Elimina la entrada de la **lista de procesos**	

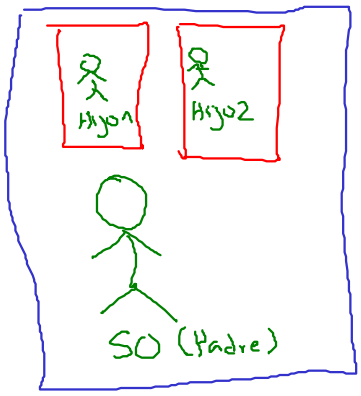






② Ejecución Directa Limitada

Restricciones



Modos

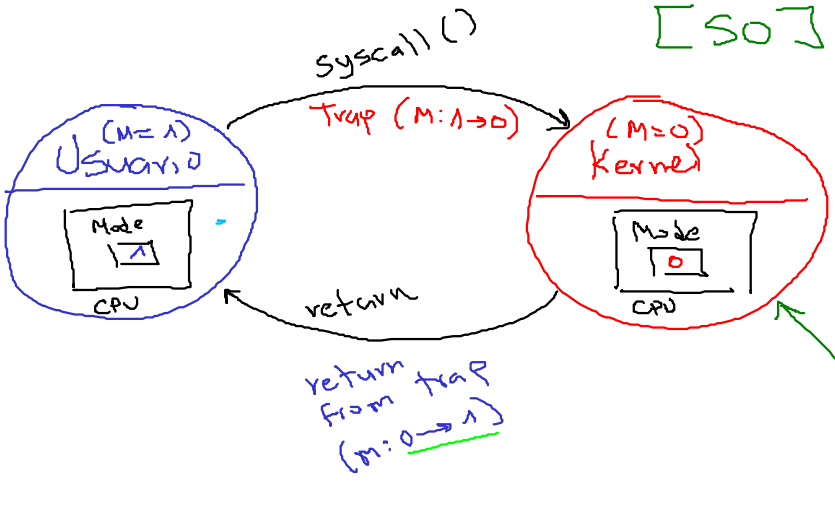
→ CPU { Modo: 0 → K ; Modo: 1 → U

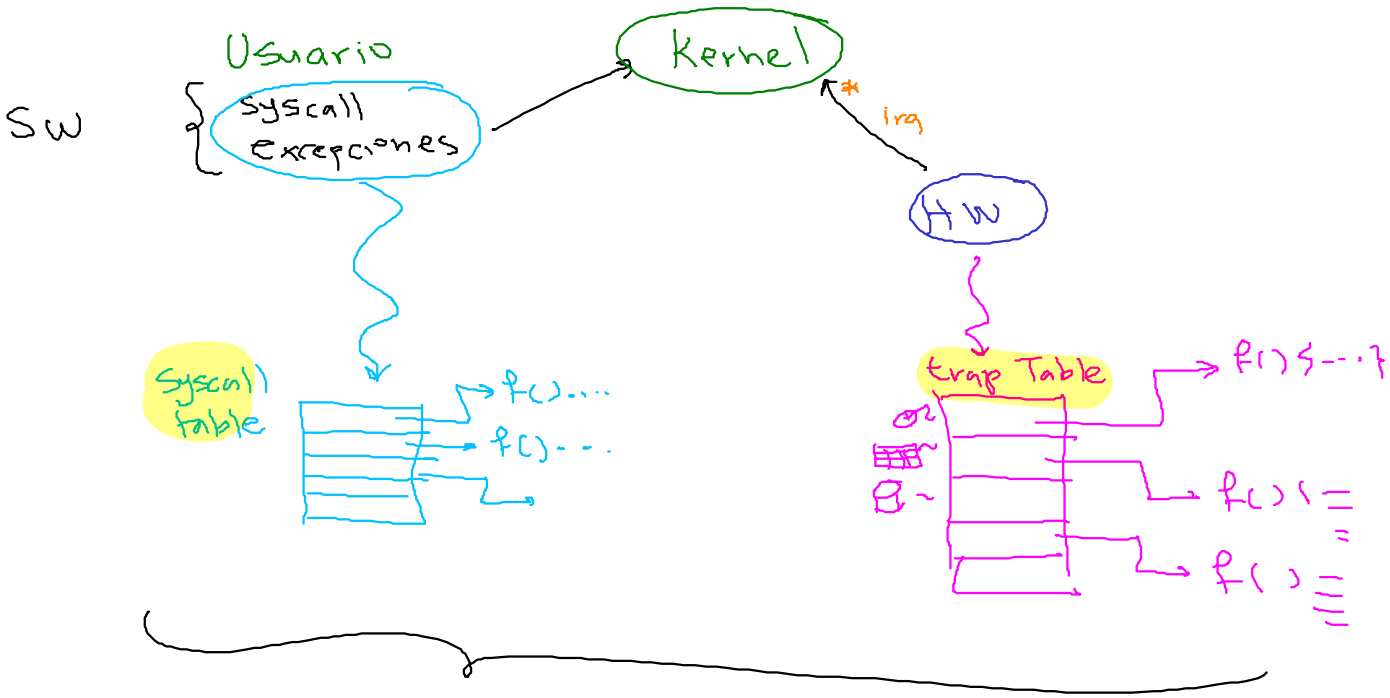
Kernel
- Acceso total
- HW
- Todas las instrucciones del ISA

[SO]

User
- Acceso Rest
- ~~HW~~
- Solo algunas instrucc del ISA

[Apps]





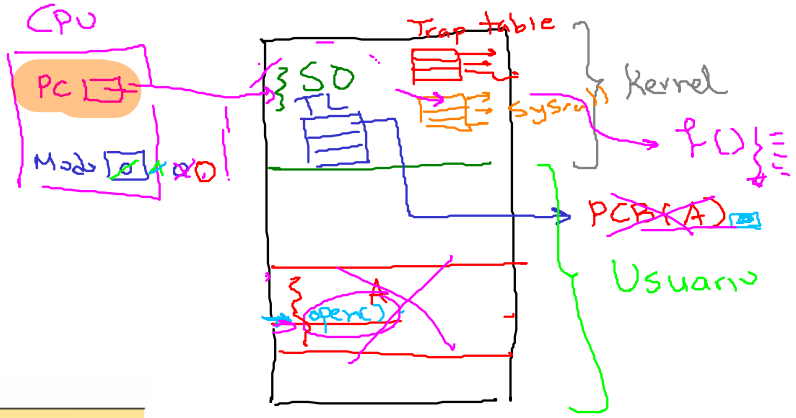
Estructuras del Kernel

1. Lista de procesos
2. Syscall table
3. trap table

Protocolo

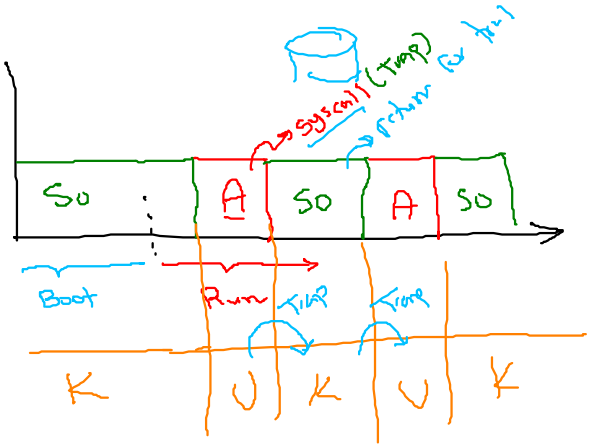
- SO @ Boot

SO @ Boot (Modo kernel)	Hardware
• Inicializa la trap table	• Almacena la dirección del Syscal handler • Ejecuta `return` de `main()`



- SO @ Run

SO @ Run (Modo kernel)	Hardware (CPU)	Programa (Modo usuario)
• Crea entrada en la lista de procesos • Asigna memoria al proceso • Carga el programa a memoria • Inicializa pila (stack) con `argc/argv` • Almacena valores de registros en el `kernel stack` • `Return-from-trap`	• Inicializa registros desde el kernel stack • Pasa a modo usuario • Salta a `main()`	• Ejecuta `main()` • ... • Llamado al sistema (`trap`)



SO @ Run (Modo kernel)	Hardware	Programa (Modo usuario)
	• Almacena valores de registros en el kernel stack • Pasa a modo kernel • Salta al trap handler	
• Atiende el evento `trap` • Realiza el trabajo solicitado por la llamada al sistema • `Return-from-trap`		
	• Restaura valores de registros desde el kernel stack • Pasa a modo usuario • Salta a PC después de `trap`	
		• ... • return de `main()` • Llamado al sistema `exit` (`trap`)
• Libera la memoria del proceso • Elimina la entrada de la lista de procesos.