



# ► PROCESOS

Sistemas Operativos

# PROCESOS – DEFINICIÓN

*Abstracción de un programa en ejecución.*

*Es la entidad mínima de asignación de Recursos. (Actual).*

# PROCESOS – DEFINICIÓN

**PROCESO: Denominación Genérica**

**JOB/TRABAJO: Sinónimo de Proceso, generalmente asociada a procesos tipo BATCH.**

**TASK/TAREA: Sinónimo de Proceso, generalmente asociado a Procesos del Sistema o Procesos en Tiempo Real**

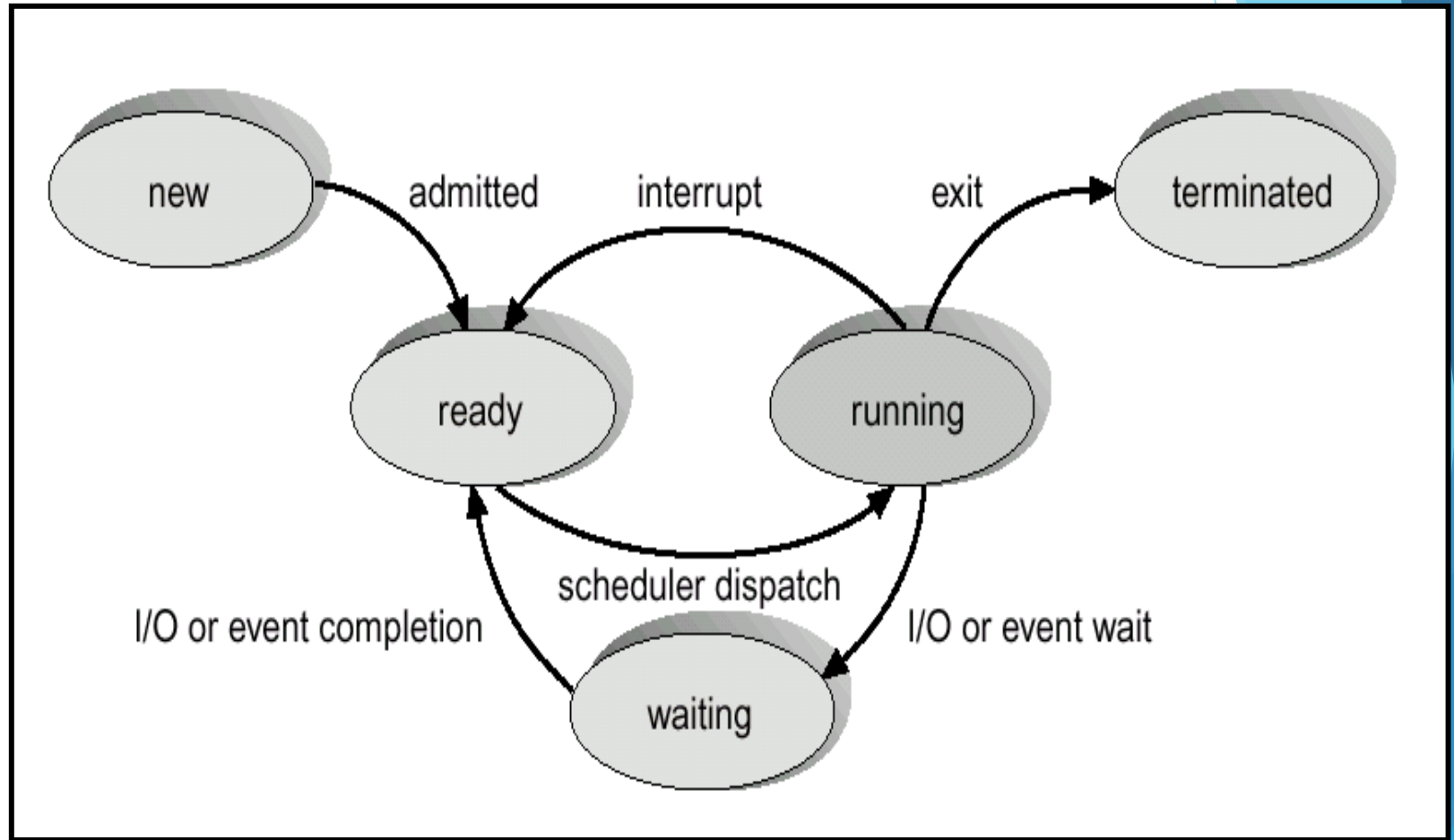
# PROCESOS – DEFINICIÓN

**PROCESO: Denominación Genérica**

**JOB/TRABAJO: Sinónimo de Proceso, generalmente asociada a procesos tipo BATCH.**

**TASK/TAREA: Sinónimo de Proceso, generalmente asociado a Procesos del Sistema o Procesos en Tiempo Real**

# ESTADOS DE UN PROCESO SIN SWAPPING



# CAMBIOS DE ESTADO DEL PROCESO

## **LISTO → EJECUCIÓN**

- El planificador selecciona al proceso para ejecución

## **EJECUCIÓN → LISTO**

- El proceso ha ejecutado demasiado tiempo, se le ha vencido el TimeSlice
- El proceso fue interrumpido y otro proceso de mayor prioridad ha quedado en estado LISTO.  
(Desalojo/Preemptive)

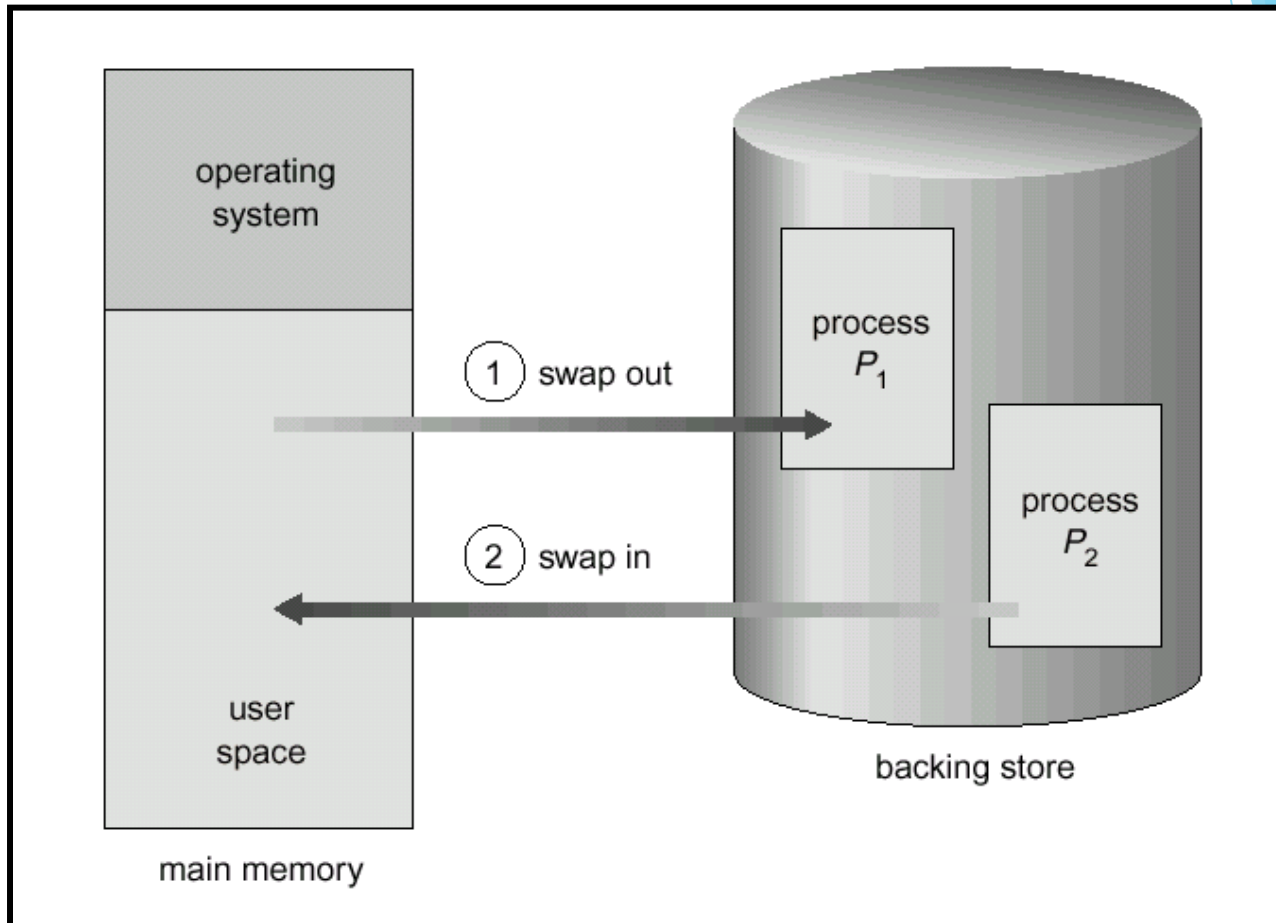
## **EJECUCIÓN → BLOQUEADO**

- El proceso requiere algo que necesita esperar
- Quiere acceder a un recurso de E/S
- Debe esperar por el resultado de una Comunicación

## **BLOQUEADO → LISTO**

- Cuando el evento que estaba esperando que ocurra ha ocurrido.

# SWAPPING

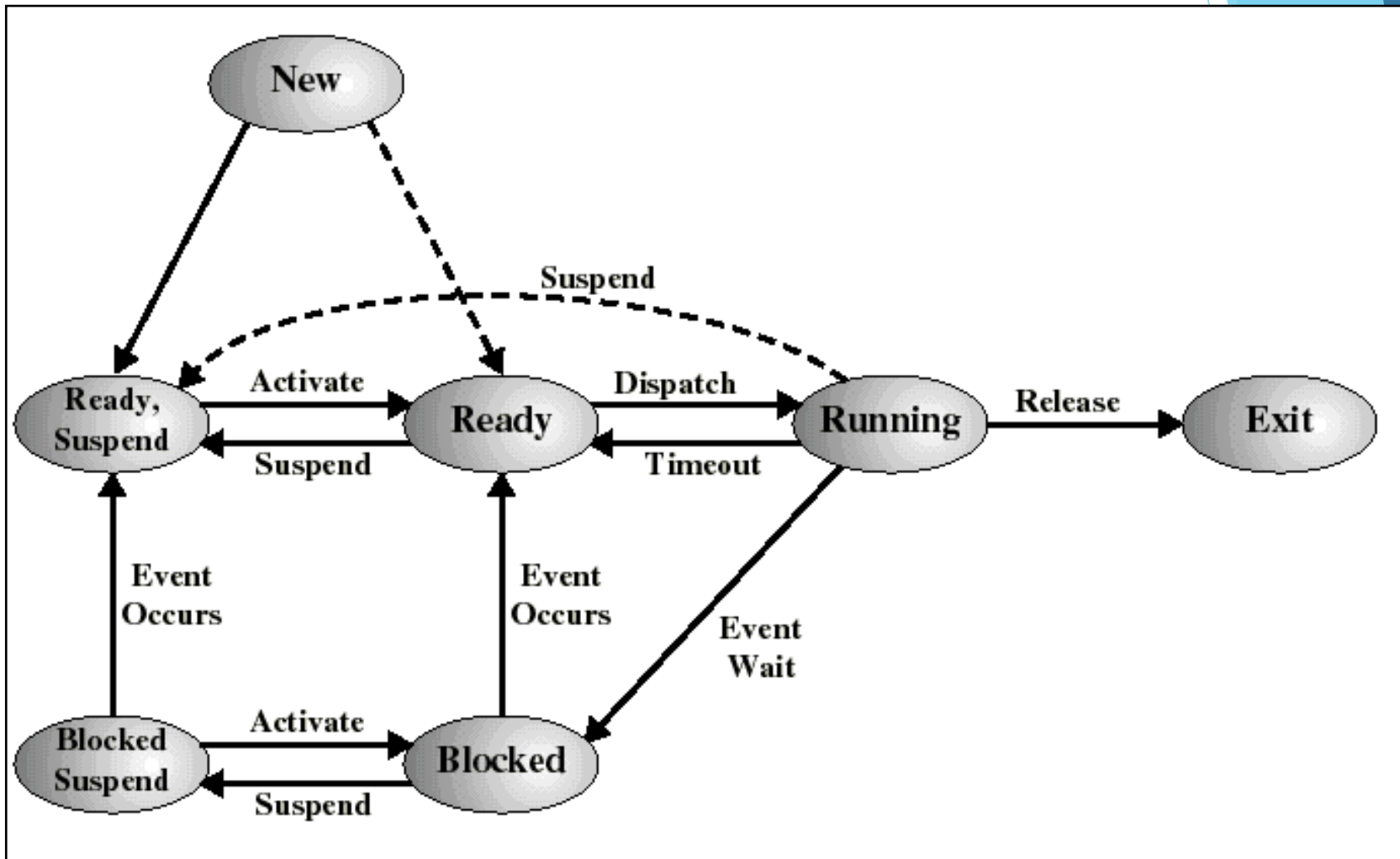


# SWAPPING

- **Para ejecutar los procesos deben estar total o parcialmente en memoria principal**
- **Aún con memoria virtual, mantener muchos procesos en memoria puede deteriorar el rendimiento del sistema**
- **El OS puede necesitar suspender ciertos procesos volcándo su espacio de direcciones a Disco.**
- **Esto añade 2 nuevos Estados a los proceso:**
  - **Bloqueado Suspendido**
  - **Listo Suspendido**



# ESTADOS DE UN PROCESO CON SWAPPING



# NUEVOS CAMBIOS DE ESTADO

## **BLOQUEADO → BLOQUEADO SUSPENDIDO**

El OS selecciona un proceso Bloqueado para Swappearlo y así liberar memoria principal

## **BLOQUEADO SUSPENDIDO → LISTO SUSPENDIDO**

Cuando el evento que estaba esperando que ocurra ha ocurrido pero aún no se encuentra en memoria principal

## **LISTO SUSPENDIDO → LISTO**

Seleccionado por el Planificador de Mediano Plazo para otorgarle memoria principal

## **LISTO → LISTO SUSPENDIDO**

Cuando no hay procesos bloqueados que puedan retirarse de memoria y el planificador decide retirar un proceso listo.

# ETRUCTURA DE UN PROCESO

- Programa de Usuario

- Datos de Usuario

- Pila o Stack(s)

Para llamada a procedimientos y pasaje de parámetros

- Process Control Block (contexto de Ejecución)

Atributos que necesita el OS para controlar el proceso. Esto Incluye:

- Información de Identificación
- Estado del Procesador
- Información de Control

# PCB

Identificador de proceso (*Process Identifier* -PID-, de sus siglas en inglés).

Estado del proceso. Por ej: listo, en espera, bloqueado.

Contador de programa: dirección de la próxima instrucción a ejecutar.

Valores de registro de CPU. Se utilizan también en el cambio de contexto.

Espacio de direcciones de memoria.

Prioridad en caso de utilizarse dicho algoritmo para planificación de CPU.

Lista de recursos asignados (incluyendo descriptores de archivos y *sockets* abiertos).

Estadísticas del proceso.

Datos del propietario (*owner*).

Señales (*Signals*) pendientes de ser servidas. (Almacenados en un mapa de bits).

# PCB

Process management	Memory management	File management
Registers	Pointer to text segment	UMASK mask
Program counter	Pointer to data segment	Root directory
Program status word	Pointer to bss segment	Working directory
Stack pointer	Exit status	File descriptors
Process state	Signal status	Effective uid
Time when process started	Process id	Effective gid
CPU time used	Parent process	System call parameters
Children's CPU time	Process group	Various flag bits
Time of next alarm	Real uid	
Message queue pointers	Effective	
Pending signal bits	Real gid	
Process id	Effective gid	
Various flag bits	Bit maps for signals	
	Various flag bits	

# CAMBIO DE CONTEXTO

**REFIERE AL CAMBIO DEL PROCESO QUE SE ENCUENTRA EN EJECUCIÓN**

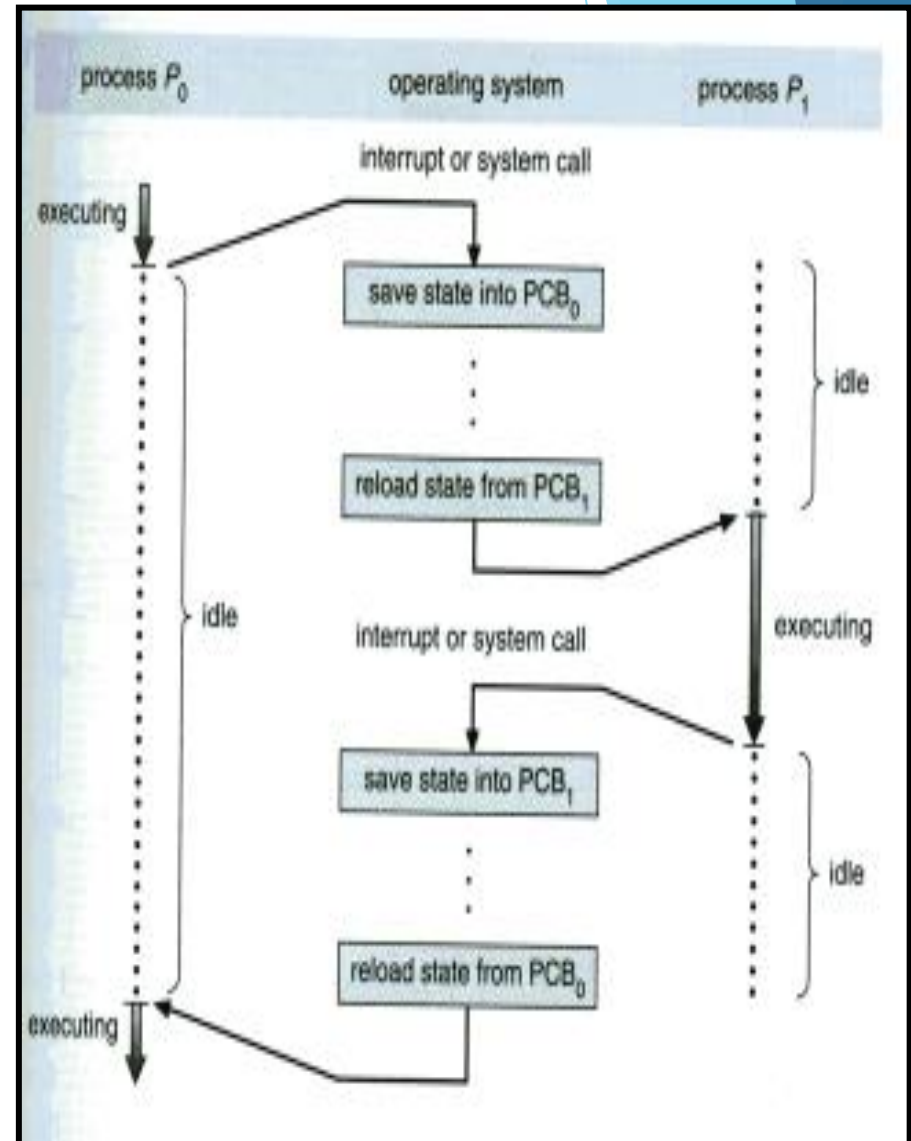
- Un cambio de contexto se debe hacer cuando el OS ha obtenido el control de la CPU
  - System Call/Trap:  
Explícitamente el proceso ha requerido un servicio del OS. Probablemente el proceso quede bloqueado
  - Excepción/Fallo  
Se ha producido un error al ejecutar la última instrucción.
  - Interrupción  
Un evento externo ha causado una interrupción. El control se ha transferido a la ISR.

# ESPACIO DE DIRECCIONES

- Asociado a cada proceso hay un Espacio de Direcciones
- El espacio de direcciones contiene:
  - El programa ejecutable
  - Los Datos del Programa
  - El Stack
- El sistema operativo tiene una Tabla de Procesos es un Vector o una lista enlazada en Memoria

# CAMBIO DE CONTEXTO

- Cuando la OS intercambia procesos debe resguardar el estado del proceso actual y cargar el estado del nuevo proceso.
- Este cambio de Contexto representa un overhead, no se hace trabajo útil mientras se conmuta.
- Es dependiente del soporte de hardware





# CAMBIO DE CONTEXTO

- 1) Salvar el contexto implica resguardar el PC y otros Registros
- 2) Actualizar el PCB del proceso en ejecución con su nuevo estado e información asociada
- 3) Colocar la entrada del PCB en la cola apropiada. LISTO, BLOQUEADO.
- 4) Seleccionar el próximo proceso a ejecutar
- 5) Actualizar el PCB del proceso seleccionado
- 6) Restaurar el contexto de CPU del proceso seleccionado.

# CAMBIO DE CONTEXTO vs CAMBIO DE MODO

## **Cambio de modo:**

- De usuario a sistema (excepción, llamada o interrupción).
- De sistema a usuario (RETI).
- No hay c. de contexto: Mismo proceso en distinta fase.

## **Cambio contexto => Cambio de proceso.**

- Sólo cuando proceso está en fase de sistema.
- El propio proceso (P):
  - cambia su estado.
  - activa el planificador-> selecciona Q.
  - salva su contexto en BCP.
  - restaura contexto de Q del BCP-> ya está ejecutando Q.

# GESTIÓN DE PROCESOS

• **Las funciones claves de la gestión de procesos son:**

- **Creación de Procesos**
- **Terminación de Procesos**
- **Intercambio de Procesos**

# GESTIÓN DE PROCESOS

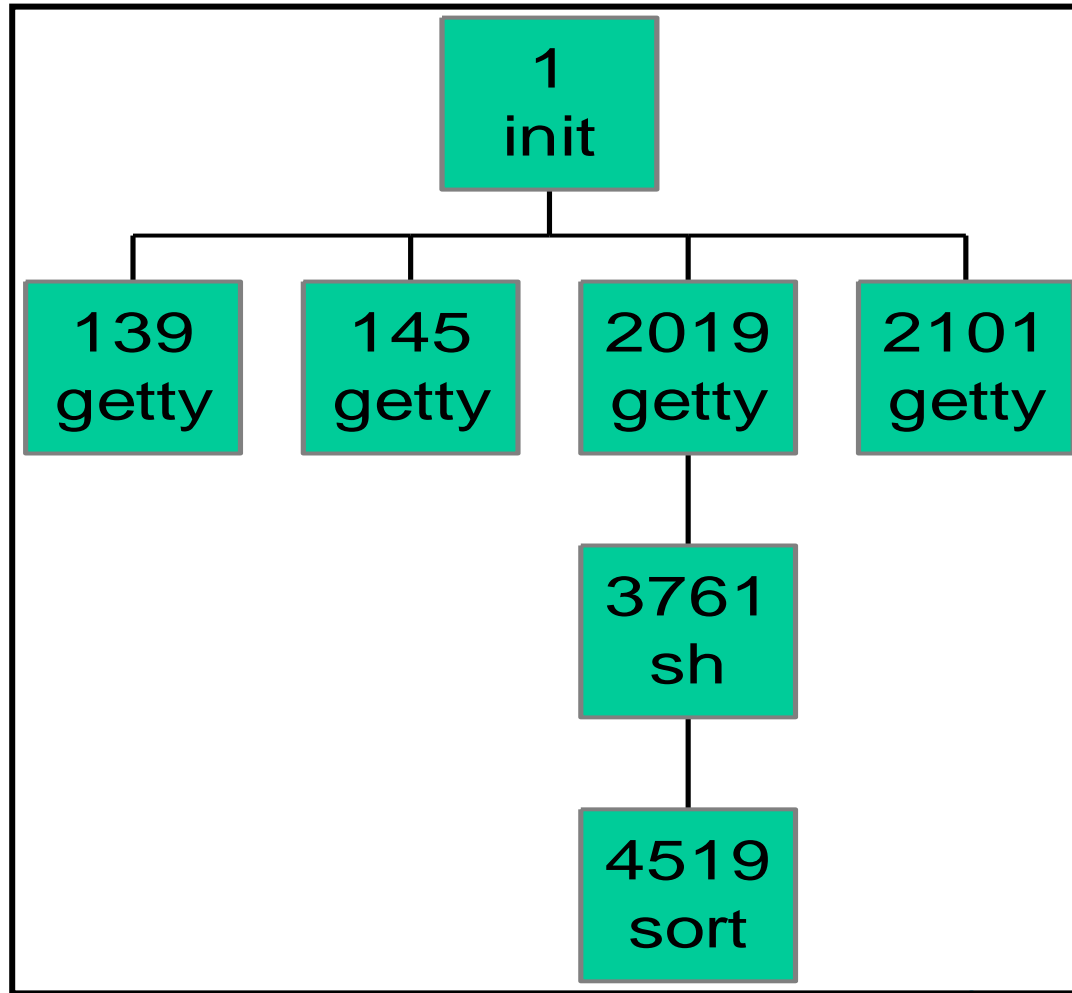
• **Las funciones claves de la gestión de procesos son:**

- **Creación de Procesos**
- **Terminación de Procesos**
- **Intercambio de Procesos**

# JERARQUÍA DE PROCESOS

- En algunos OS un proceso puede crear uno o más procesos hijos (child processes) los cuales pueden crear otros nuevos procesos.
- Los procesos se pueden comunicar entre sí utilizando IPC (interprocess communications).
- Cada proceso tiene asignado un identificador de proceso (PID) que le fue asignado por el OS.
- El PID es un número creciente no repetible de cada proceso.

# JERARQUÍA DE PROCESOS

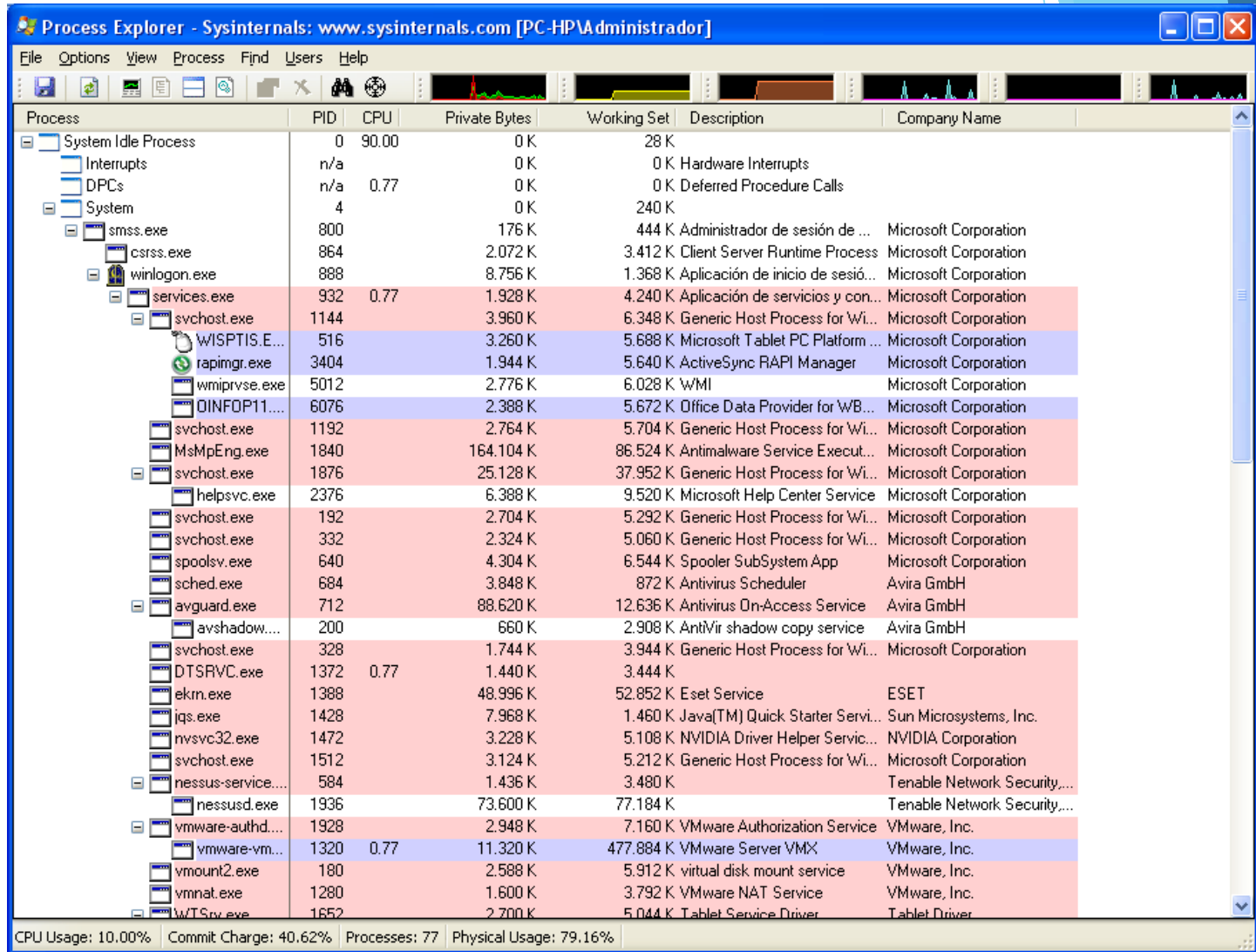


# JERARQUÍA DE PROCESOS

```
nessus:/home/so# ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	Mar12	?	00:00:01	init [2]
root	2	0	0	Mar12	?	00:00:00	[kthrea]
root	3	2	0	Mar12	?	00:00:00	[migrat]
root	4	2	0	Mar12	?	00:00:00	[ksofti]
root	5	2	0	Mar12	?	00:00:00	[watchd]
root	6	2	0	Mar12	?	00:00:00	[events]
root	7	2	0	Mar12	?	00:00:00	[cpuset]
root	8	2	0	Mar12	?	00:00:00	[khelpe]
root	11	2	0	Mar12	?	00:00:00	[netns]
root	14	2	0	Mar12	?	00:00:00	[async/]
root	65	2	0	Mar12	?	00:00:00	[kinteg]
root	67	2	0	Mar12	?	00:00:00	[kblock]
root	69	2	0	Mar12	?	00:00:00	[kacpid]
root	70	2	0	Mar12	?	00:00:00	[kacpi_]
root	115	2	0	Mar12	?	00:00:00	[kserio]
root	146	2	0	Mar12	?	00:00:00	[kondem]
root	163	2	0	Mar12	?	00:00:00	[khungt]
root	164	2	0	Mar12	?	00:00:00	[pdflus]
root	165	2	0	Mar12	?	00:00:00	[pdflus]
root	166	2	0	Mar12	?	00:00:00	[kswapd]

# JERARQUÍA DE PROCESOS



Process Explorer - Sysinternals: www.sysinternals.com [PC-HP\Administrador]

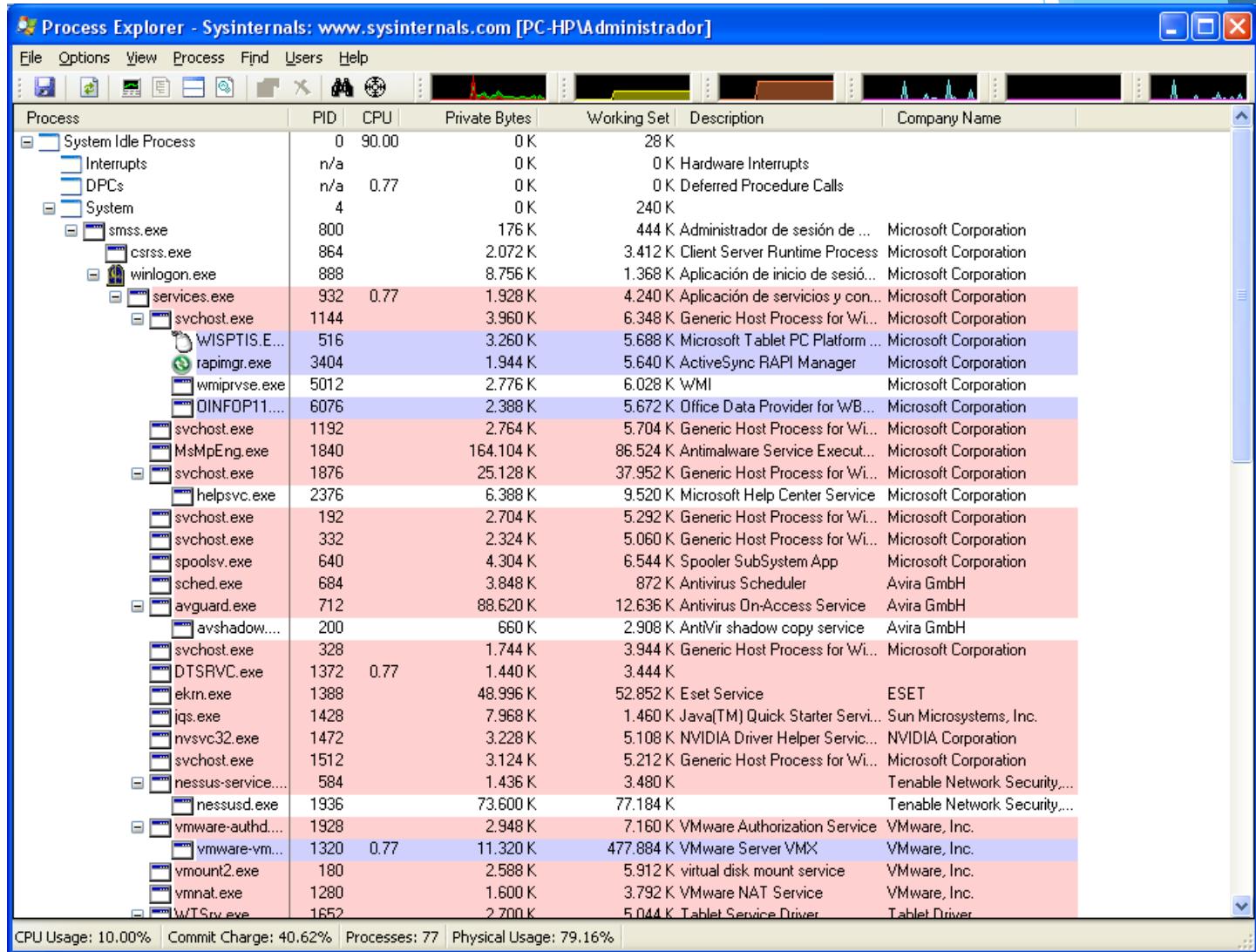
File Options View Process Find Users Help

Process	PID	CPU	Private Bytes	Working Set	Description	Company Name
System Idle Process	0	90.00	0 K	28 K		
Interrupts	n/a		0 K	0 K	Hardware Interrupts	
DPCs	n/a	0.77	0 K	0 K	Deferred Procedure Calls	
System	4		0 K	240 K		
smss.exe	800		176 K		444 K Administrador de sesión de ...	Microsoft Corporation
csrss.exe	864		2.072 K	3.412 K	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	888		8.756 K	1.368 K	Aplicación de inicio de sesi...	Microsoft Corporation
services.exe	932	0.77	1.928 K	4.240 K	Aplicación de servicios y con...	Microsoft Corporation
svchost.exe	1144		3.960 K	6.348 K	Generic Host Process for Wi...	Microsoft Corporation
WISPTIS.E...	516		3.260 K	5.688 K	Microsoft Tablet PC Platform ...	Microsoft Corporation
rapimgr.exe	3404		1.944 K	5.640 K	ActiveSync RAPI Manager	Microsoft Corporation
wmiprvse.exe	5012		2.776 K	6.028 K	WMI	Microsoft Corporation
QINFOP11...	6076		2.388 K	5.672 K	Office Data Provider for WB...	Microsoft Corporation
svchost.exe	1192		2.764 K	5.704 K	Generic Host Process for Wi...	Microsoft Corporation
MsMpEng.exe	1840		164.104 K	86.524 K	Antimalware Service Execut...	Microsoft Corporation
svchost.exe	1876		25.128 K	37.952 K	Generic Host Process for Wi...	Microsoft Corporation
helpsvc.exe	2376		6.388 K	9.520 K	Microsoft Help Center Service	Microsoft Corporation
svchost.exe	192		2.704 K	5.292 K	Generic Host Process for Wi...	Microsoft Corporation
svchost.exe	332		2.324 K	5.060 K	Generic Host Process for Wi...	Microsoft Corporation
spoolsv.exe	640		4.304 K	6.544 K	Spooler SubSystem App	Microsoft Corporation
sched.exe	684		3.848 K	872 K	Antivirus Scheduler	Avira GmbH
avguard.exe	712		88.620 K	12.636 K	Antivirus On-Access Service	Avira GmbH
avshadow...	200		660 K	2.908 K	AntiVir shadow copy service	Avira GmbH
svchost.exe	328		1.744 K	3.944 K	Generic Host Process for Wi...	Microsoft Corporation
DTSRVC.exe	1372	0.77	1.440 K	3.444 K		
ekrn.exe	1388		48.996 K	52.852 K	Eset Service	ESET
iqs.exe	1428		7.968 K	1.460 K	Java(TM) Quick Starter Servi...	Sun Microsystems, Inc.
nvsvc32.exe	1472		3.228 K	5.108 K	NVIDIA Driver Helper Servic...	NVIDIA Corporation
svchost.exe	1512		3.124 K	5.212 K	Generic Host Process for Wi...	Microsoft Corporation
nessus-service...	584		1.436 K	3.480 K		Tenable Network Security...
nessusd.exe	1936		73.600 K	77.184 K		Tenable Network Security...
vmware-authd...	1928		2.948 K	7.160 K	VMware Authorization Service	VMware, Inc.
vmware-vm...	1320	0.77	11.320 K	477.884 K	VMware Server VMX	VMware, Inc.
vmount2.exe	180		2.588 K	5.912 K	virtual disk mount service	VMware, Inc.
vmnat.exe	1280		1.600 K	3.792 K	VMware NAT Service	VMware, Inc.
WTSrv.exe	1652		2.700 K	5.044 K	Tablet Service Driver	Tablet Driver

CPU Usage: 10.00% Commit Charge: 40.62% Processes: 77 Physical Usage: 79.16%



# JERARQUÍA DE PROCESOS



Process Explorer - Sysinternals: www.sysinternals.com [PC-HP\Administrador]

File Options View Process Find Users Help

Process	PID	CPU	Private Bytes	Working Set	Description	Company Name
System Idle Process	0	90.00	0 K	28 K		
Interrupts	n/a		0 K	0 K	Hardware Interrupts	
DPCs	n/a	0.77	0 K	0 K	Deferred Procedure Calls	
System	4		0 K	240 K		
smss.exe	800		176 K		444 K Administrador de sesión de ...	Microsoft Corporation
csrss.exe	864		2.072 K	3.412 K	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	888		8.756 K	1.368 K	Aplicación de inicio de sesi...	Microsoft Corporation
services.exe	932	0.77	1.928 K	4.240 K	Aplicación de servicios y con...	Microsoft Corporation
svchost.exe	1144		3.960 K	6.348 K	Generic Host Process for Wi...	Microsoft Corporation
WISPTIS.E...	516		3.260 K	5.688 K	Microsoft Tablet PC Platform ...	Microsoft Corporation
rapimgr.exe	3404		1.944 K	5.640 K	ActiveSync RAPI Manager	Microsoft Corporation
wmiprvse.exe	5012		2.776 K	6.028 K	WMI	Microsoft Corporation
QINFOP11...	6076		2.388 K	5.672 K	Office Data Provider for WB...	Microsoft Corporation
svchost.exe	1192		2.764 K	5.704 K	Generic Host Process for Wi...	Microsoft Corporation
MsMpEng.exe	1840		164.104 K	86.524 K	Antimalware Service Execut...	Microsoft Corporation
svchost.exe	1876		25.128 K	37.952 K	Generic Host Process for Wi...	Microsoft Corporation
helpsvc.exe	2376		6.388 K	9.520 K	Microsoft Help Center Service	Microsoft Corporation
svchost.exe	192		2.704 K	5.292 K	Generic Host Process for Wi...	Microsoft Corporation
svchost.exe	332		2.324 K	5.060 K	Generic Host Process for Wi...	Microsoft Corporation
spoolsv.exe	640		4.304 K	6.544 K	Spooler SubSystem App	Microsoft Corporation
sched.exe	684		3.848 K	872 K	Antivirus Scheduler	Avira GmbH
avguard.exe	712		88.620 K	12.636 K	Antivirus On-Access Service	Avira GmbH
avshadow...	200		660 K	2.908 K	AntiVir shadow copy service	Avira GmbH
svchost.exe	328		1.744 K	3.944 K	Generic Host Process for Wi...	Microsoft Corporation
DTSRVC.exe	1372	0.77	1.440 K	3.444 K		
ekrn.exe	1388		48.996 K	52.852 K	Eset Service	ESET
iqs.exe	1428		7.968 K	1.460 K	Java(TM) Quick Starter Servi...	Sun Microsystems, Inc.
nvsvc32.exe	1472		3.228 K	5.108 K	NVIDIA Driver Helper Servic...	NVIDIA Corporation
svchost.exe	1512		3.124 K	5.212 K	Generic Host Process for Wi...	Microsoft Corporation
nessus-service...	584		1.436 K	3.480 K		Tenable Network Security,...
nessusd.exe	1936		73.600 K	77.184 K		Tenable Network Security,...
vmware-authd...	1928		2.948 K	7.160 K	VMware Authorization Service	VMware, Inc.
vmware-vm...	1320	0.77	11.320 K	477.884 K	VMware Server VMX	VMware, Inc.
vmount2.exe	180		2.588 K	5.912 K	virtual disk mount service	VMware, Inc.
vmnat.exe	1280		1.600 K	3.792 K	VMware NAT Service	VMware, Inc.
WTSrv.exe	1652		2.700 K	5.044 K	Tablet Service Driver	Tablet Driver

CPU Usage: 10.00% Commit Charge: 40.62% Processes: 77 Physical Usage: 79.16%

# THREADS

programa invocador

```
main()
```

```
{
```

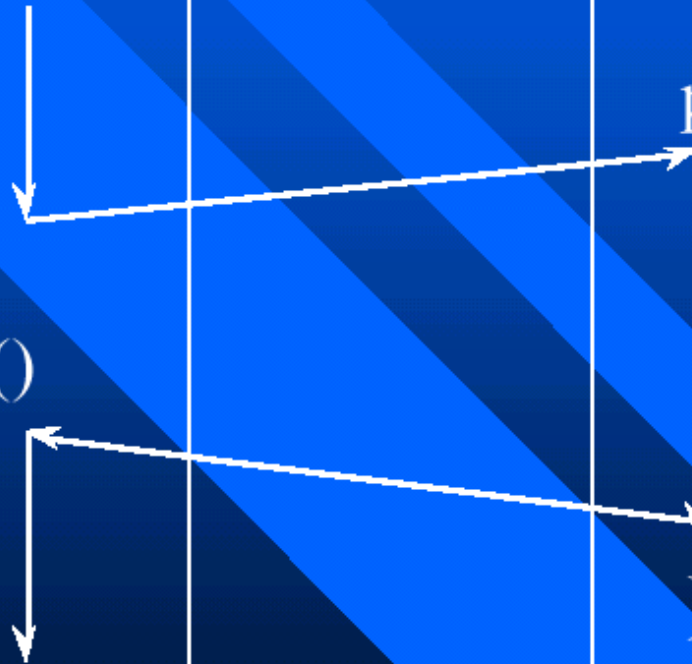
```
    procesa_fd()
```

```
}
```

función invocada

```
procesa_fd() {
```

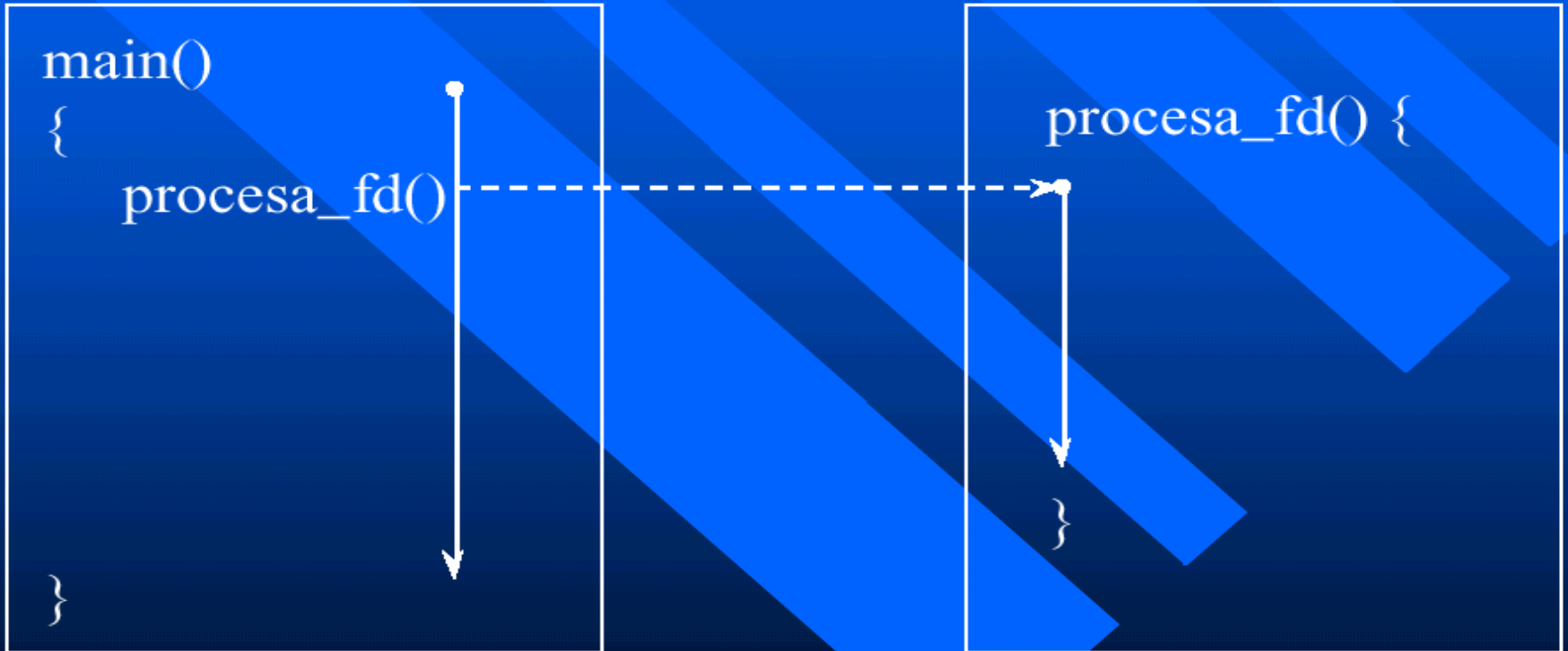
```
}
```



# THREADS

programa invocador

función invocada



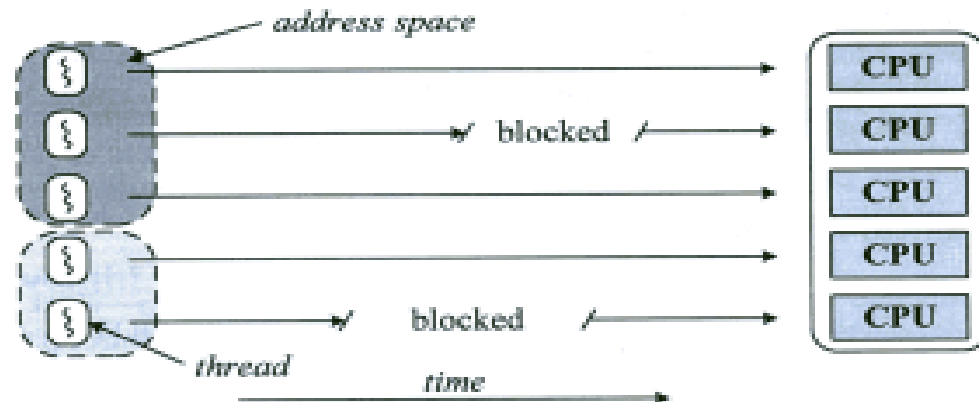
creación thread



ejecución thread

# THREADS: USOS

- Algunos programas tienen mejor rendimiento si utilizan threads para comunicarse entre sí. (Ej: Servidores) que haciéndolo utilizando solo un flujo de instrucciones.



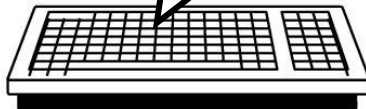
- En sistemas multiprocesador, los threads pueden ejecutarse en paralelismo Real, permitiendo que un programa pueda dividir su trabajo entre distintas CPUs. Su rendimiento es mejor que con único hilo que solo utilizaría una CPU a la vez.

# THREADS: USOS

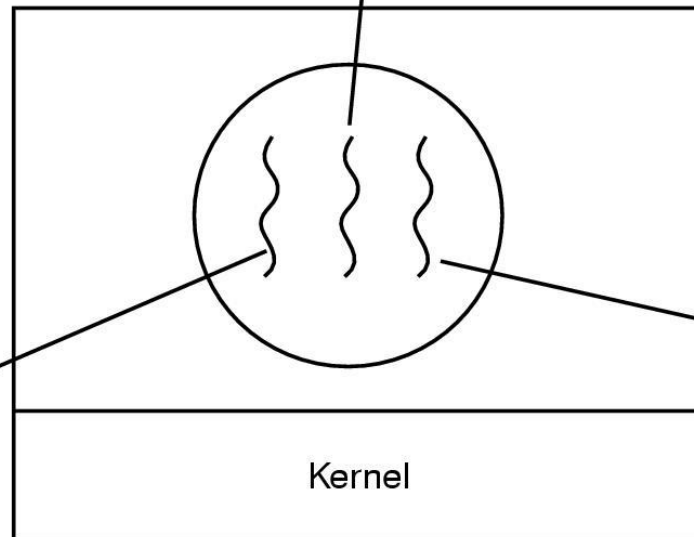
**REFORMATEA EL  
DOCUMENTO**

Four score and seven years ago, our fathers brought forth upon this continent a new nation: conceived in liberty, and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war testing whether that	nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting place for those who here gave their	lives that this nation might live. It is altogether fitting and proper that we should do this. But, in a larger sense, we cannot consecrate we cannot hallow this ground. The brave men, living and dead,	who struggled here have consecrated it far above our poor power to add or detract. The world will little note, nor long remember, what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated	here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us, that from these honored dead we take increased devotion to that cause for which	they gave the last full measure of devotion, that we here highly resolve that these dead shall not have died in vain that this nation, under God, shall have a new birth of freedom and that government of the people, for the people
---	--	---	--	---	---

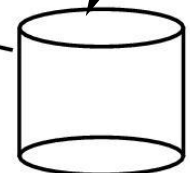
**ATIENDE  
AL USUARIO**



Keyboard



**GUARDA EL  
DOCUMENTO**



Disk

# VENTAJAS DE THREADS

- Mientras un thread está Bloqueado esperando un evento, otro thread del mismo proceso puede seguir ejecutando
- La Cooperación de múltiples threads en un mismo proceso le da mejor rendimiento y throughput.
- Las Aplicaciones que utilizan buffers en común obtienen mayores beneficios en la utilización de threads

# THREADS: CARACTERÍSTICAS

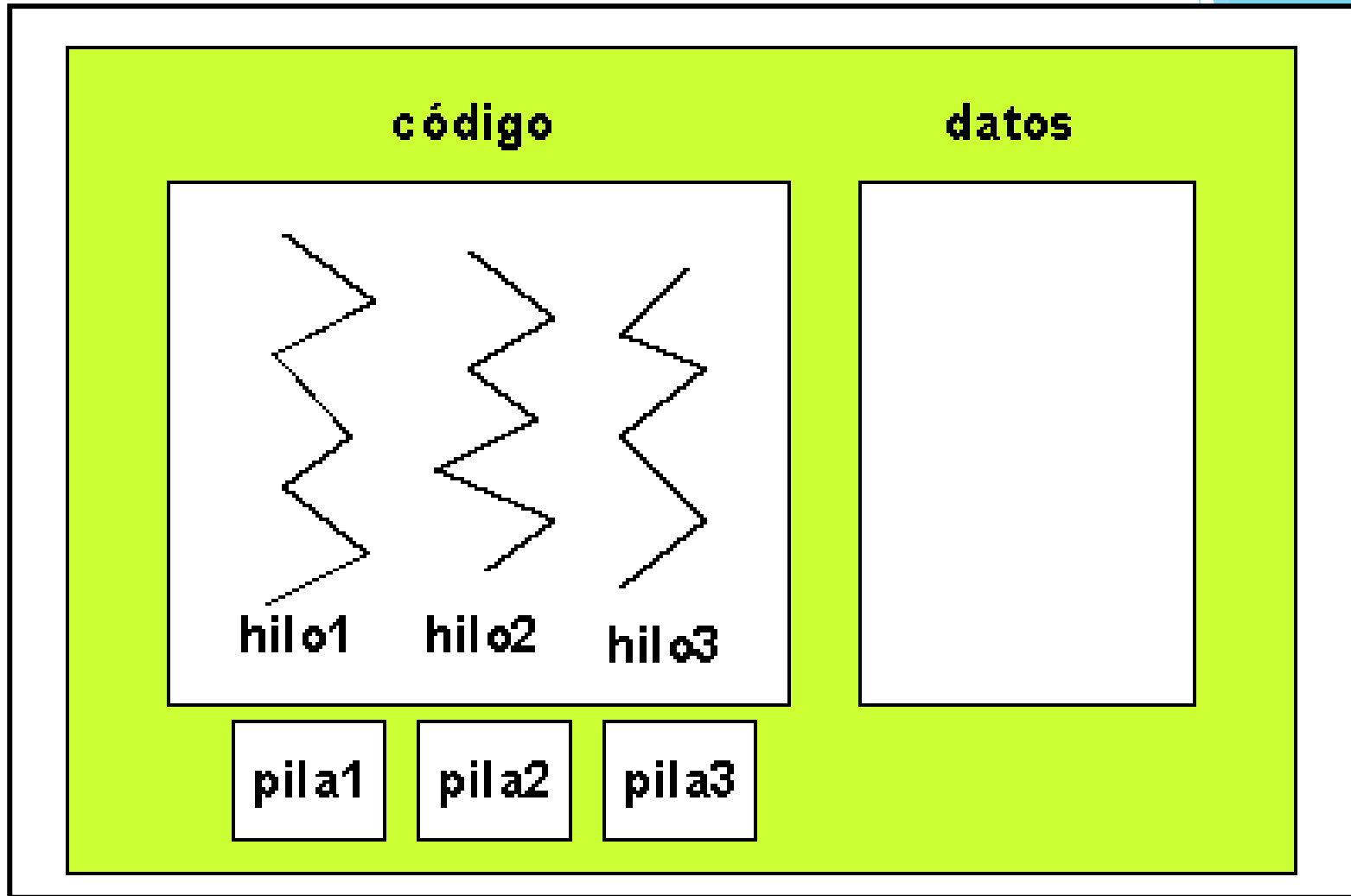
- Los threads se ejecutan independiente y simultáneamente
- Comparten las instrucciones del proceso y casi todos sus datos
- Si un thread realiza un cambio en un dato de los datos compartidos este será percibido por los otros threads en el mismo proceso
- Su ejecución puede afectar al proceso de forma sorprendente
- Pueden proporcionar concurrencia dentro de un mismo proceso
- Si dos threads comparten recursos durante su ejecución, deben tener cuidado de que no se interfieran uno con el otro

# THREADS A NIVEL KERNEL

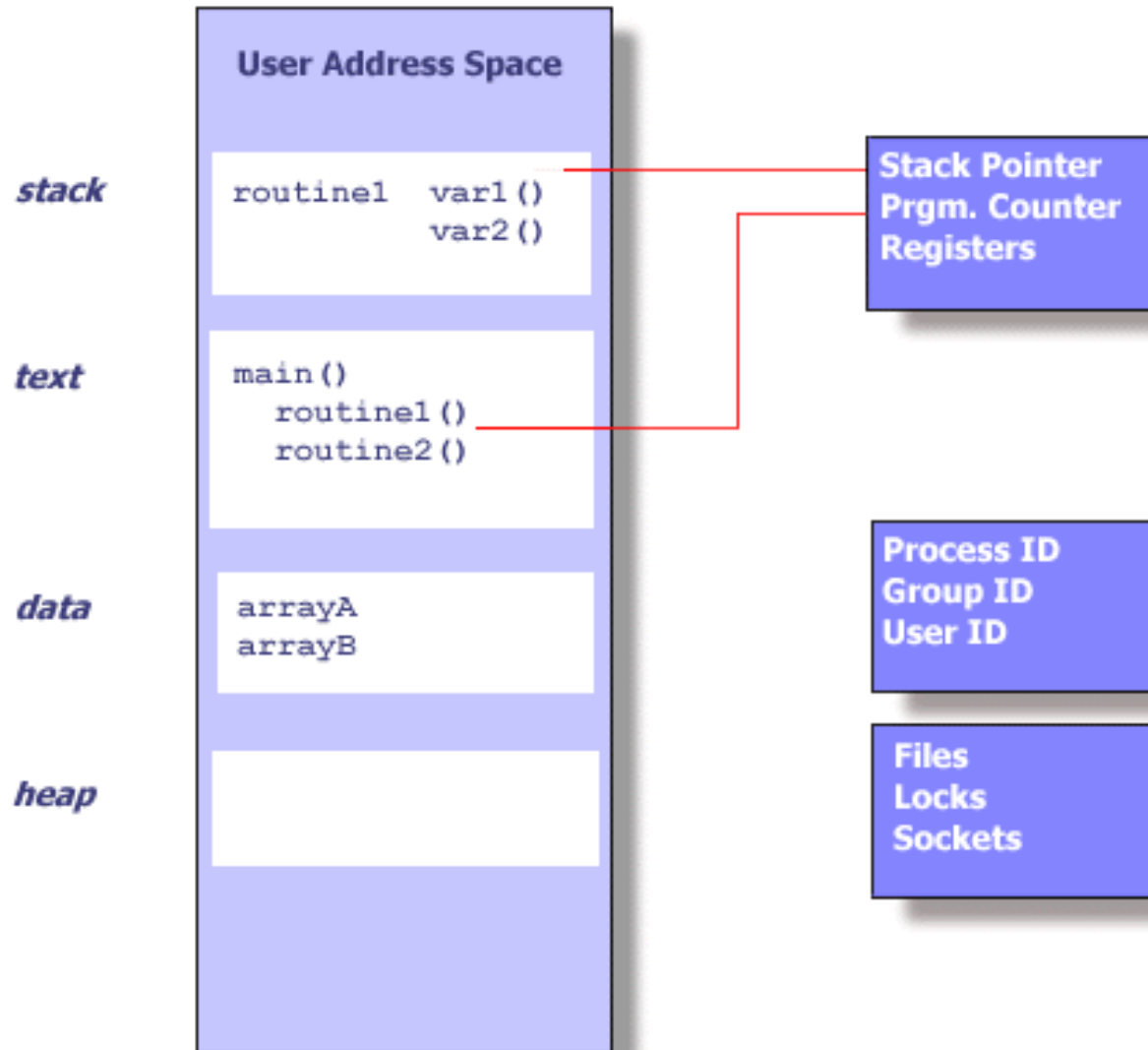
- Un thread (o proceso liviano) es la entidad mínima planificable.
- Consiste de:
  - PC
  - Set de Registros
  - Su propio Stack
- Un thread comparte con los threads del mismo proceso:
  - Área de Código (Ej: subrutinas, procedimientos)
  - Área de Datos
  - Usuario propietario
  - Recursos del OS (Ej: archivos abiertos)



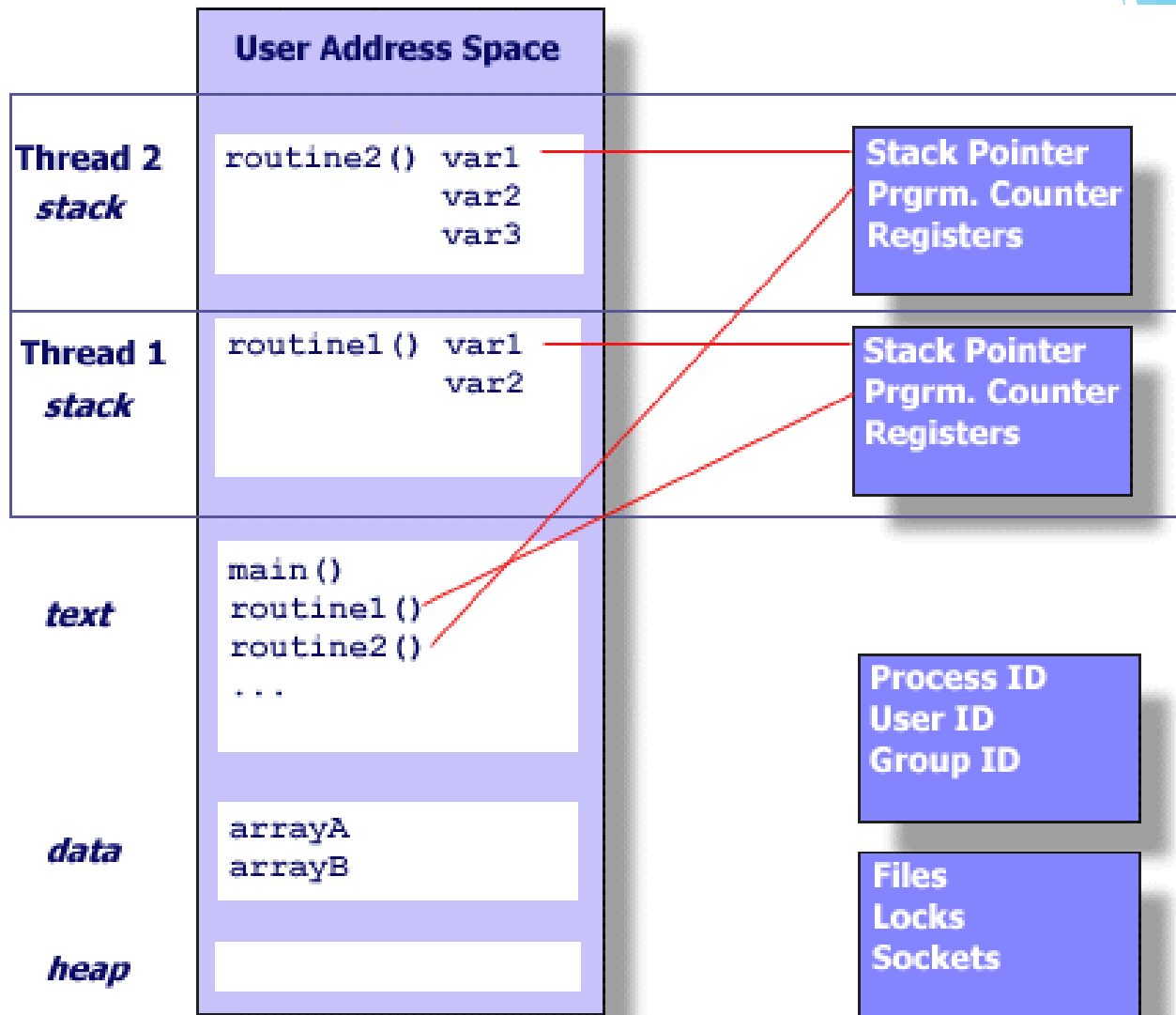
# THREADS A NIVEL KERNEL



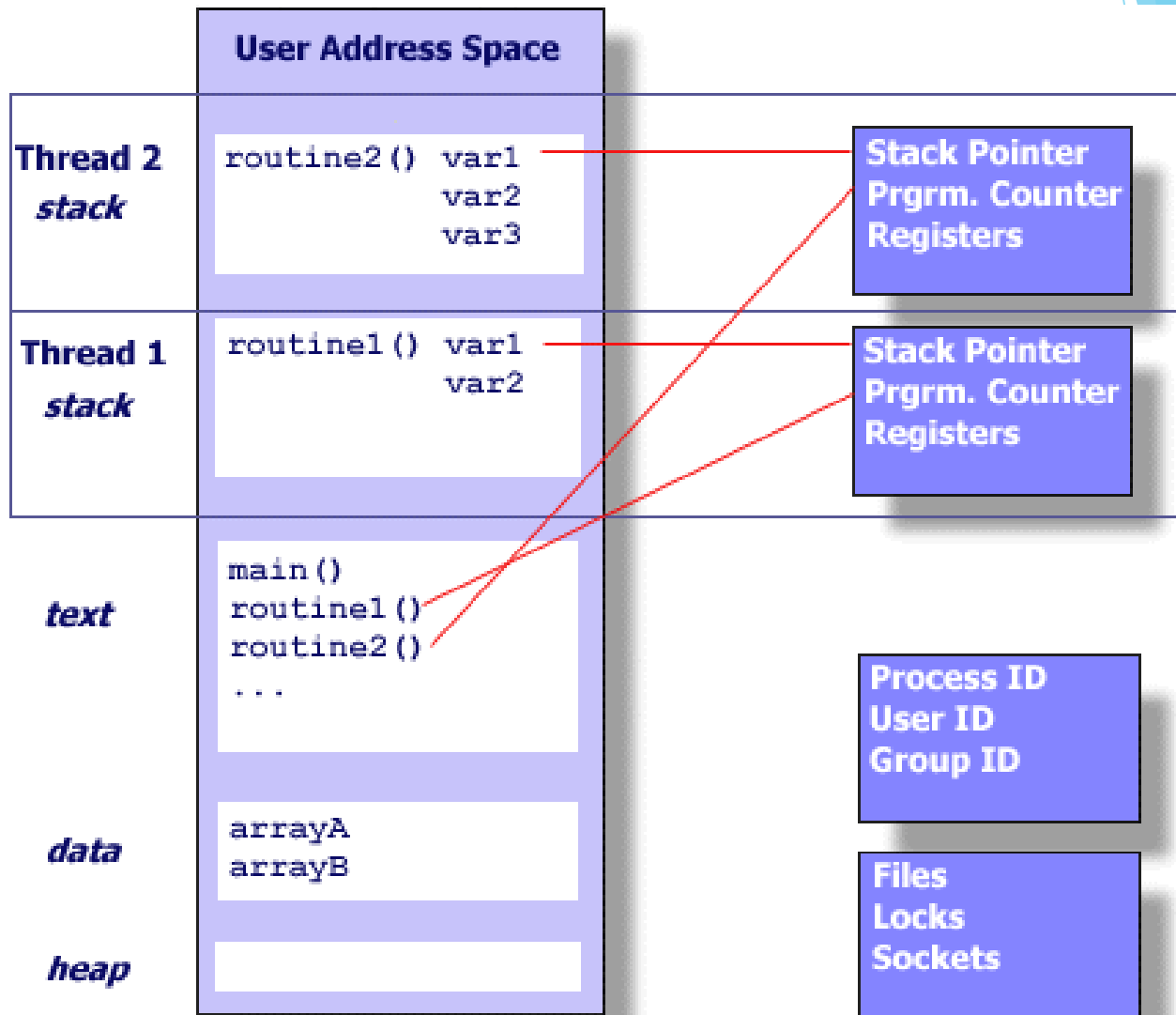
# THREADS A NIVEL KERNEL



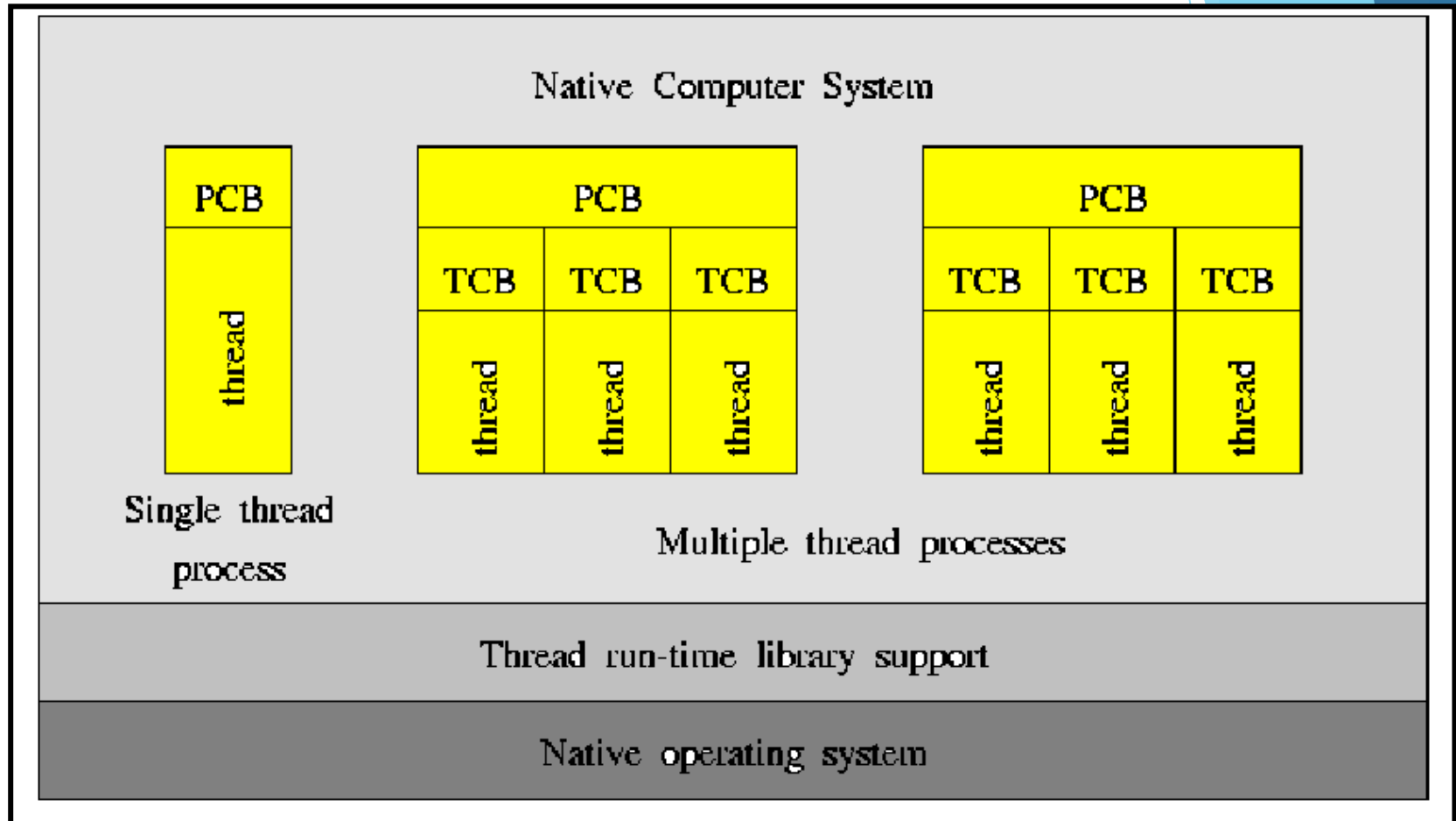
# THREADS A NIVEL KERNEL



# THREADS A NIVEL KERNEL



# THREADS A NIVEL LIBRERÍA



# THREADS A NIVEL LIBRERÍA

## ■ Ventajas

- Thread switching no involucra al Kernel => Se evita un cambio de Modo.
- Se puede hacer planificación Específica de acuerdo al mejor algoritmo
- Son portables

## • Desventajas

- La mayoría de los System calls son bloqueantes y el Kernel bloquea al proceso entero
- El kernel solo puede asignar CPU a los procesos => no pueden usar eficientemente multiprocesadores

# THREADS A NIVEL LIBRERÍA

## ■ Ventajas

- Thread switching no involucra al Kernel => Se evita un cambio de Modo.
- Se puede hacer planificación Específica de acuerdo al mejor algoritmo
- Son portables

## • Desventajas

- La mayoría de los System calls son bloqueantes y el Kernel bloquea al proceso entero
- El kernel solo puede asignar CPU a los procesos => no pueden usar eficientemente multiprocesadores

# THREADS EJEMPLO

# gcc -o threads -pthread threads.c

```
int main(int argc, char *argv[])
{
    pthread_t t[MAXTHREADS];
    void *res;
    int i, j, s;
    char car;

    for( j = 0; j < MAXTHREADS; j++)
    {
        car = 65+j;
        s = pthread_create(&t[j], NULL, threadFunc, &car);
        if (s != 0)
        {
            printf("pthread_create %d\n",j);
            exit(1);
        }
        else
            printf("pthread_create success %d\n",j);
    }
}
```



# THREADS EJEMPLO

```
for( j = 0; j < MAXTHREADS; j++)
{
    s = pthread_join(t[j], &res);
    if (s != 0)
    {
        printf("pthread_join %d\n",j);
        exit(1);
    }
    else
        printf("pthread_join success %d\n",j);
}

exit(EXIT_SUCCESS);
}
```

# THREADS EJEMPLO

```
static void *threadFunc(void *arg)
{
    int i;
    char car, *ptr;

    ptr = (char *) arg;
    car = *ptr;
    sleep(1);
    for(i =0; i < MAXLOOP; i++)
    {
        putchar(car);
        sleep(1);
        fflush(stdout);
    }
    r = (int)car;
    return((void*)&r);
}
```

# THREADS EJEMPLO

```
nessus:/home/so# ./threads
pthread_create success 0
pthread_create success 1
pthread_create success 2
pthread_create success 3
pthread_create success 4
ABCDEDCBAECBDAEDBCAEDCBAECBADEABDCEBDCAEDCBAECBDAEBDACEDACBEACBDECBADEBADCEBCADE
ADCBECDBEDBCAEBDCAE
pthread_join success 0
pthread_join success 1
pthread_join success 2
pthread_join success 3
pthread_join success 4
```

# PLANIFICACIÓN CPU

*La planificación del procesador se refiere a la manera o técnicas que se usan para decidir cuanto tiempo de ejecución y cuando se le asignan CPU a cada proceso del sistema. Esto es crucial para el buen funcionamiento del sistema.*

# NIVELES DE PLANIFICACIÓN

- ***DE LARGO PLAZO:***

Decide que trabajos (proyectos de procesos) son candidatos a convertirse en procesos compitiendo por los recursos del sistema.

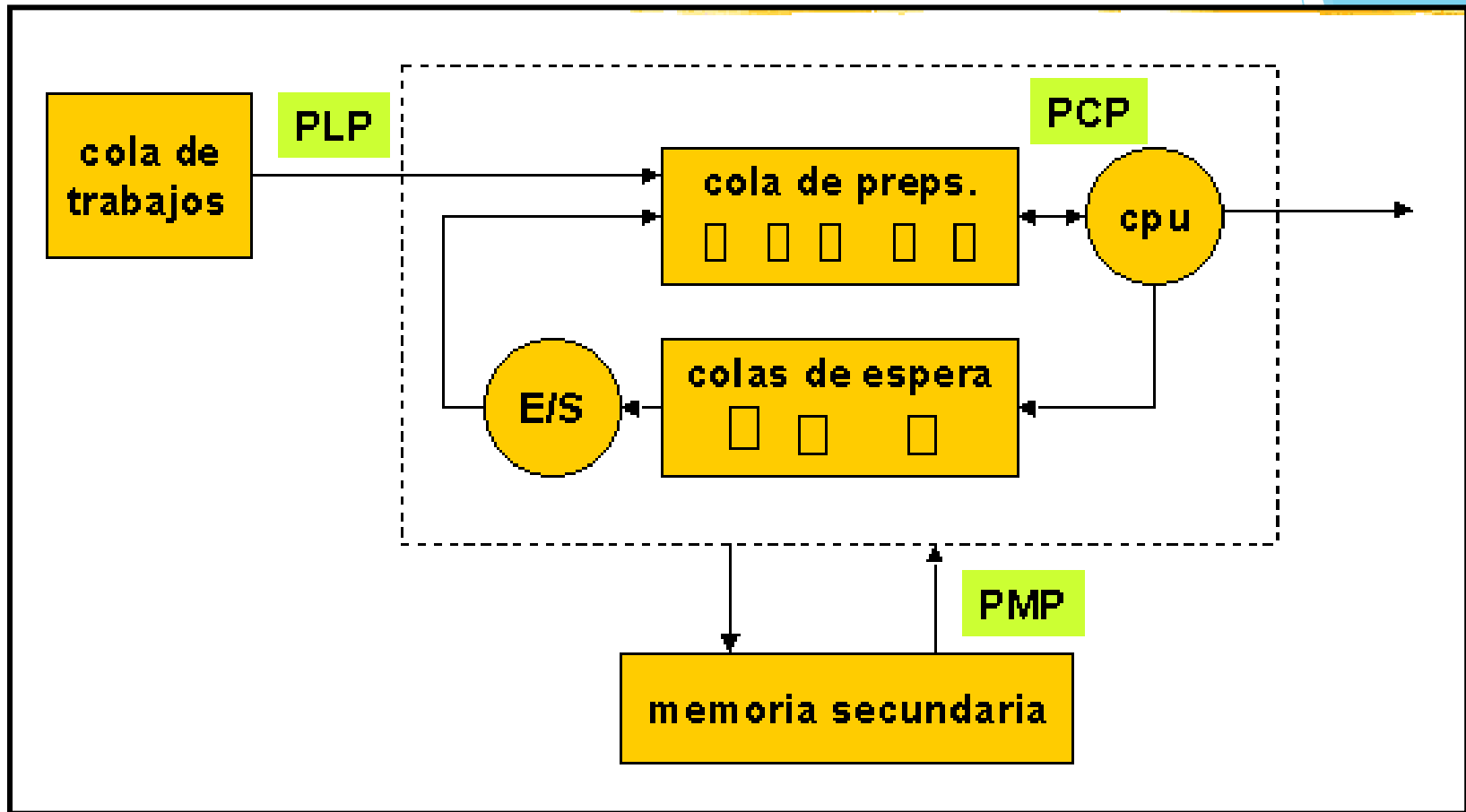
- ***DE MEDIANO PLAZO:***

Decide a que procesos se le otorga memoria principal para poder ejecutar

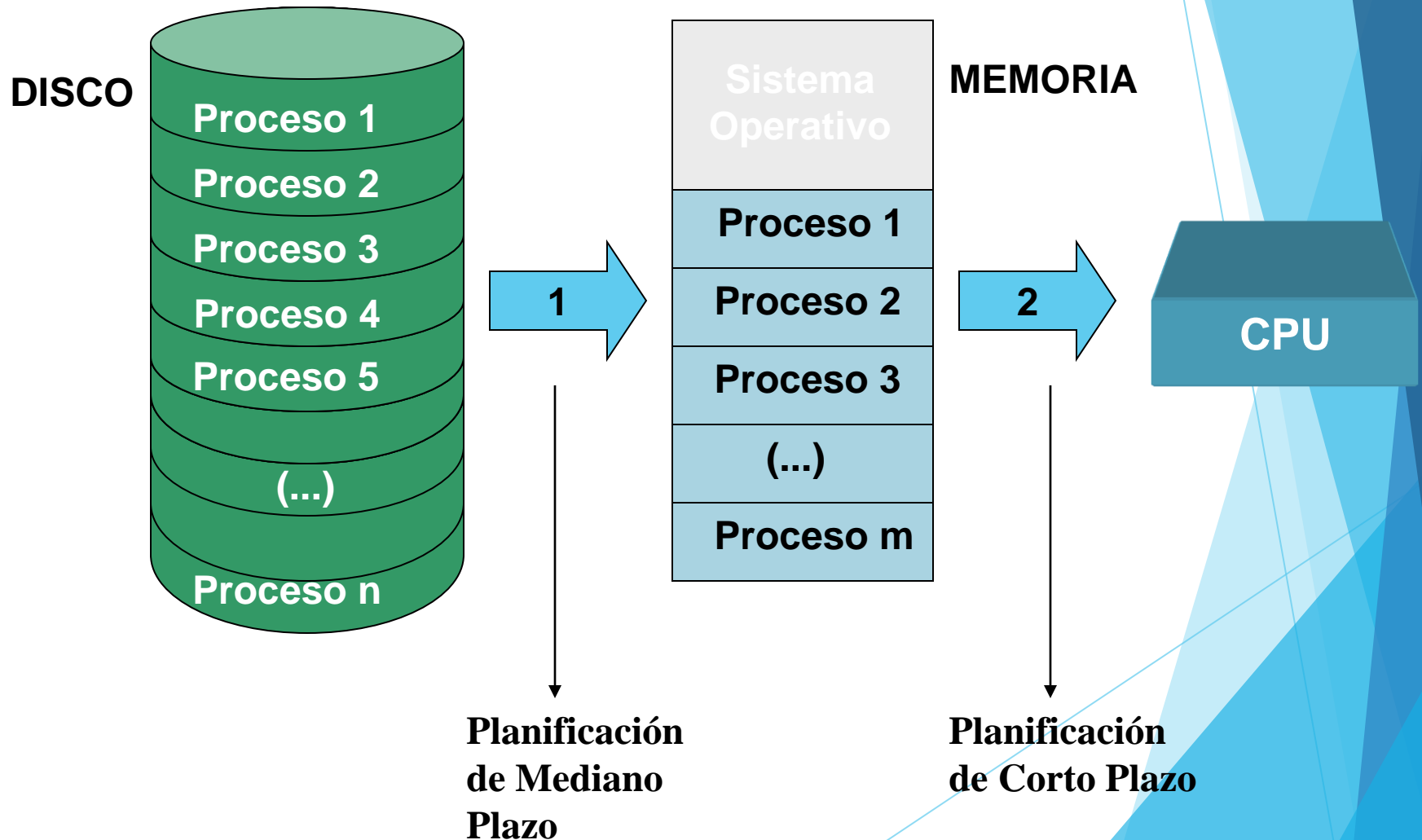
- ***DE CORTO PLAZO:***

Decide que proceso, de los que ya están listos es al que le toca ahora estar utilizar la CPU

# NIVELES DE PLANIFICACIÓN



# NIVELES DE PLANIFICACIÓN



# PLANIFICACIÓN CPU: MODELOS DE SISTEMAS

- Podemos considerar que la vida activa de un proceso es una sucesión de:

- ráfagas de CPU -> el proceso ejecuta instrucciones
- ráfagas de E/S -> el proceso utiliza o espera por la E/S

- Según la utilización de los recursos, se observan:

- procesos intensivos en CPU (ej. cálculos numéricos)
- procesos intensivos en E/S (ej. interactivos)



# PLANIFICACIÓN CPU: OBJETIVOS

## 1. Justicia o Imparcialidad:

Todos los procesos son tratados de la misma forma, y en algún momento obtienen su turno de ejecución o intervalos de tiempo de ejecución hasta su terminación exitosa.

## 2. Maximizar la Producción:

El sistema debe de finalizar el mayor numero de procesos en por unidad de tiempo.

## 3. Minimizar el Tiempo de Respuesta:

Cada usuario o proceso debe observar que el sistema les responde consistentemente a sus requerimientos.

# PLANIFICACIÓN CPU: OBJETIVOS

## 4. Evitar el aplazamiento indefinido:

Los procesos deben terminar en un plazo finito de tiempo.

## 5. El sistema debe ser predecible:

Ante cargas de trabajo ligeras el sistema debe responder rápido y con cargas pesadas debe ir degradándose paulatinamente. Otro punto de vista de esto es que si se ejecuta el mismo proceso en cargas similares de todo el sistema, la respuesta en todos los casos debe ser similar.

# PLANIFICACIÓN CPU: OBJETIVOS

## 4. Evitar el aplazamiento indefinido:

Los procesos deben terminar en un plazo finito de tiempo.

## 5. El sistema debe ser predecible:

Ante cargas de trabajo ligeras el sistema debe responder rápido y con cargas pesadas debe ir degradándose paulatinamente. Otro punto de vista de esto es que si se ejecuta el mismo proceso en cargas similares de todo el sistema, la respuesta en todos los casos debe ser similar.

# PLANIFICACIÓN CPU: OBJETIVOS

## **All systems**

Fairness - giving each process a fair share of the CPU

Policy enforcement - seeing that stated policy is carried out

Balance - keeping all parts of the system busy

## **Batch systems**

Throughput - maximize jobs per hour

Turnaround time - minimize time between submission and termination

CPU utilization - keep the CPU busy all the time

## **Interactive systems**

Response time - respond to requests quickly

Proportionality - meet users' expectations

## **Real-time systems**

Meeting deadlines - avoid losing data

Predictability - avoid quality degradation in multimedia systems

# PLANIFICACIÓN APROPIATIVA

- **La planificación No apropiativa (Non-Preemptive):**

Es aquella en la cual un proceso puede ser retirado de su estado de Ejecución si lo hace por sí mismo o por vencimiento de su TimeSlice.

- **La planificación Apropiativa (Preemptive):**

Es aquella en la cual un proceso puede ser retirado de su estado de Ejecución si lo hace por sí mismo o por vencimiento de su TimeSlice o porque un proceso de Mayor prioridad se encuentra en estado de LISTO.

# PLANIFICACIÓN APROPIATIVA

- **La planificación No apropiativa (Non-Preemptive):**

Es aquella en la cual un proceso puede ser retirado de su estado de Ejecución si lo hace por sí mismo o por vencimiento de su TimeSlice.

- **La planificación Apropiativa (Preemptive):**

Es aquella en la cual un proceso puede ser retirado de su estado de Ejecución si lo hace por sí mismo o por vencimiento de su TimeSlice o porque un proceso de Mayor prioridad se encuentra en estado de LISTO.

# FCFS

- Es muy simple, los procesos reciben su turno conforme llegan.
- La ventaja de este algoritmo es que es justo y no provoca aplazamiento indefinido.
- La desventaja es que no aprovecha ninguna característica de los procesos y puede no servir para un proceso de tiempo real.
- El tiempo promedio de respuesta puede ser muy malo comparado con el logrado por el del trabajo más corto primero.

# ROUND ROBIN

- También llamada por turno, consiste en darle a cada proceso un timeslice.
- Los procesos están ordenados en una cola circular.
- La ventaja de este algoritmo es su simplicidad, es justo y no provoca aplazamiento indefinido.



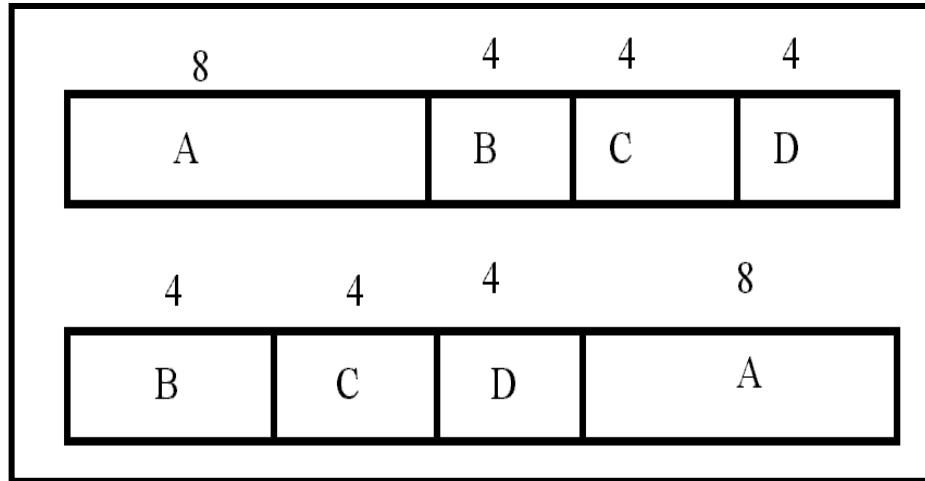
# PRIMERO EL TRABAJO MÁS CORTO

- Se requiere saber o tener una estimación de cuánto tiempo necesita el proceso para terminar.
- Si se sabe, se ejecutan primero aquellos trabajos que necesitan menos tiempo y de esta manera se obtiene el mejor tiempo de respuesta promedio para todos los procesos. .
- A=4; B=12; C=18;D=24;E=26

Proceso	Espera desde	Termina	Tiempo de Espera
A	0	4	4
B	0	16	16
C	0	34	34
D	0	58	58
E	0	84	84

Tiempo promedio =  $(4 + 16 + 34 + 58 + 84) / 5 = 39$  unidades

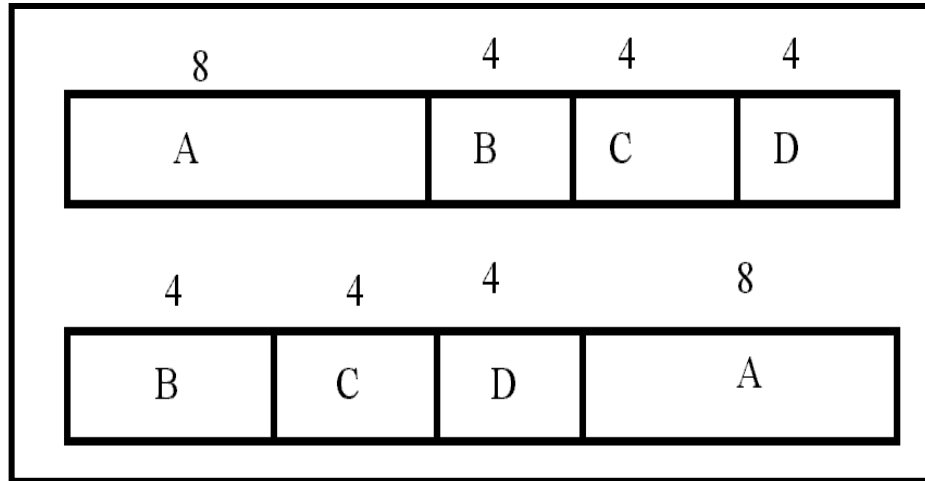
# PRIMERO EL TRABAJO MÁS CORTO



$$T_{\text{RESP1}} = (8 + 12 + 16 + 20) / 4 = 14$$

$$T_{\text{RESP2}} = (4 + 8 + 12 + 20) / 4 = 11$$

# PRIMERO EL TRABAJO MÁS CORTO



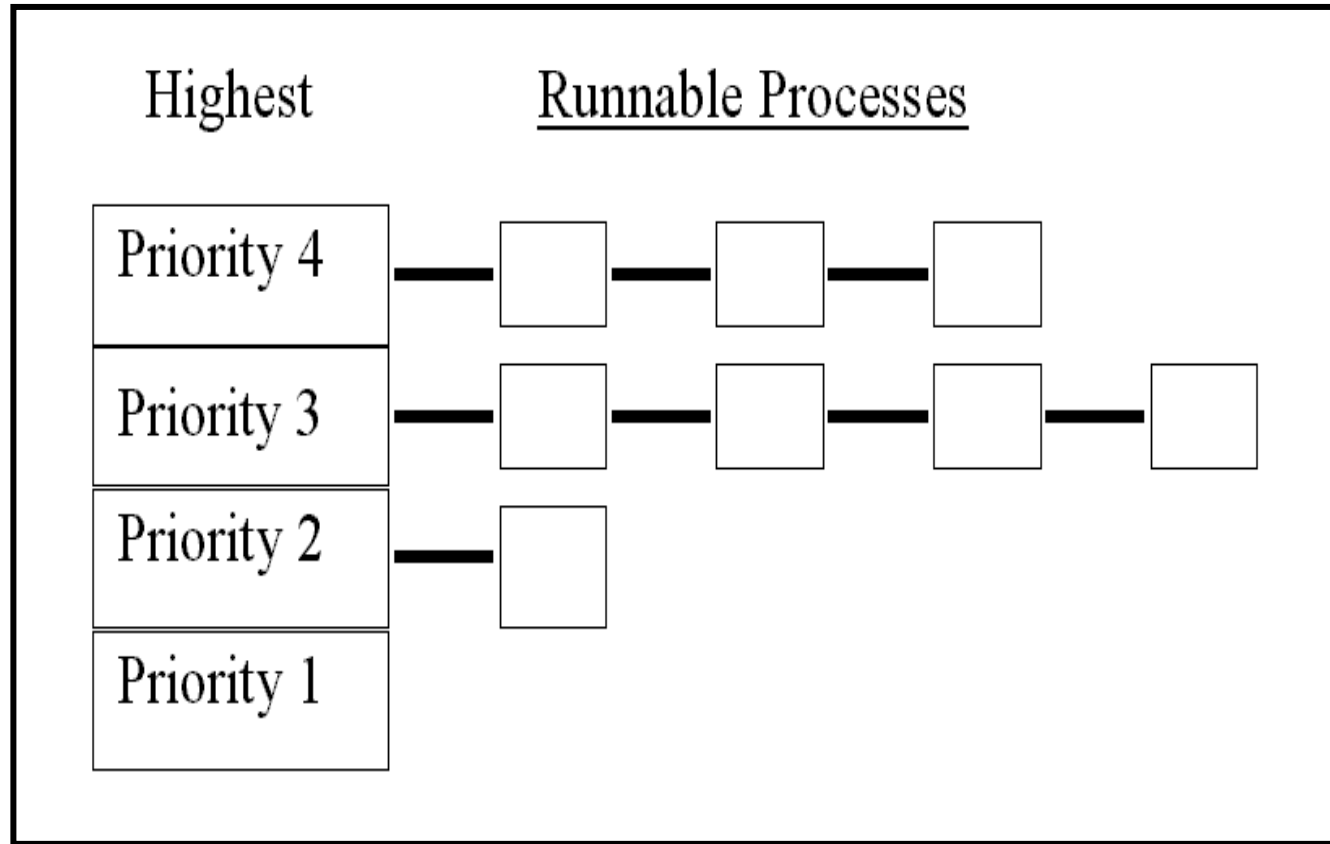
$$T_{\text{RESP1}} = (8 + 12 + 16 + 20) / 4 = 14$$

$$T_{\text{RESP2}} = (4 + 8 + 12 + 20) / 4 = 11$$

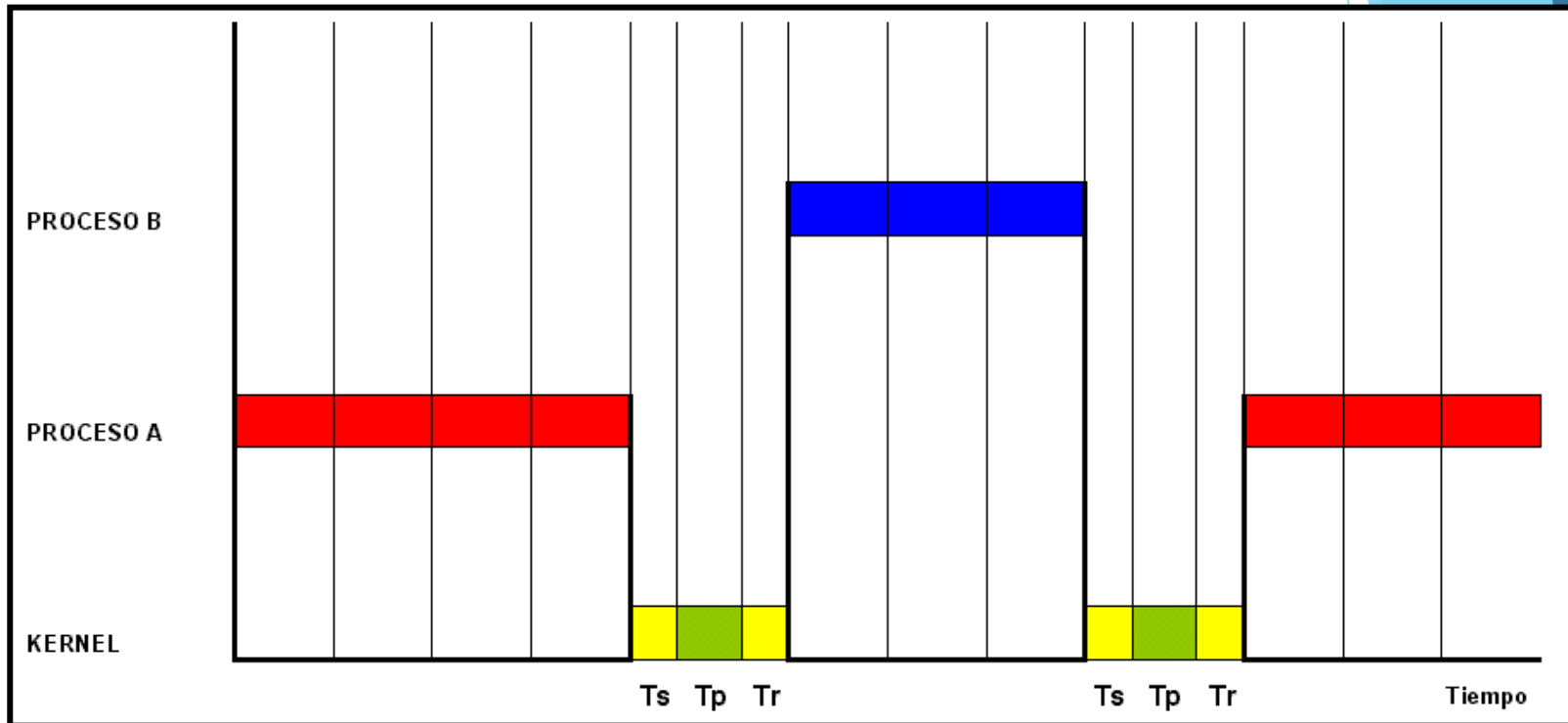
# PLANIFICACIÓN POR PRIORIDADES

- Los procesos de mayor prioridad se ejecutan primero.
- Si existen varios procesos de mayor prioridad que otros, pero entre ellos con la misma prioridad, pueden ejecutarse estos de acuerdo a su orden de llegada o por 'round robin'.
- La ventaja de este algoritmo es que es flexible en cuanto a permitir que ciertos procesos se ejecuten primero e, incluso, por más tiempo.
- Su desventaja es que puede provocar aplazamiento indefinido en los procesos de baja prioridad.

# PLANIFICACIÓN POR PRIORIDADES



# RENDIMIENTO



**Ts = Tiempo de Save**

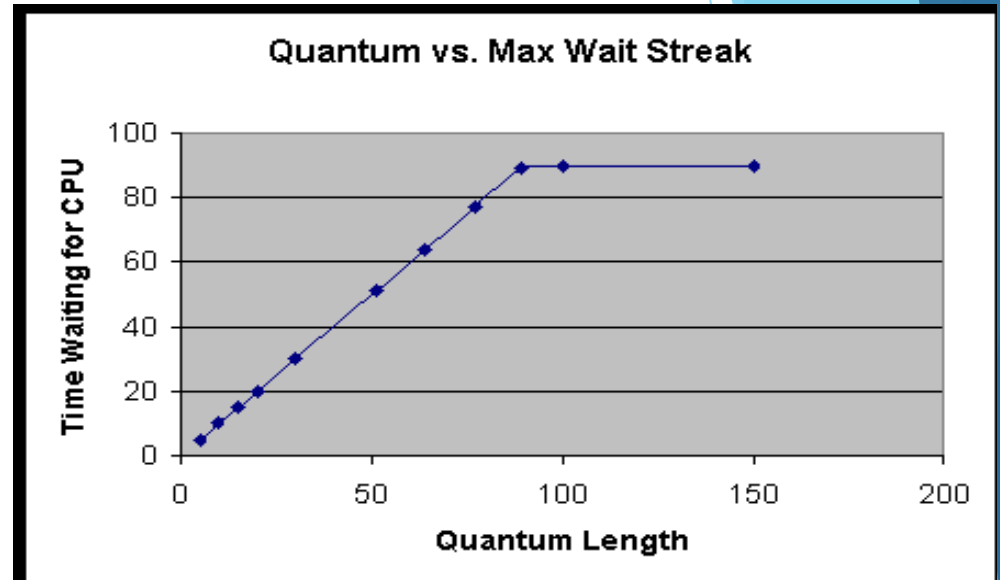
**Tr = Tiempo de Restore**

**Tp= Tiempo de Planificación**

# RENDIMIENTO: TIMESLICE

## **TimeSlice Grande:**

Disminuyen las interrupciones provocando tiempos de respuesta mas largos.



## **TimeSlice Pequeño:**

Se producen muchas interrupciones, esto implica cambios de contexto lo que significa una reducción en el rendimiento. En la fórmula de rendimiento es  $n$ .