Tema 1. Administración de Sistemas

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Índice

Objetivos

- Definir el **concepto** de **Administrador**.
- Identificar las **tareas** asociadas a la **administración** de un **sistema**.
- Estudiar las **responsabilidades** del **Administrador** en un entorno **multiusuario**.
- Analizar los **temas** que conforman la **filosofía** de la **administración** de sistemas.
- Estudiar los **comandos** básicos de **administración**.
- Conocer algunas **herramientas** administrativas de los **SSOO**.

El Administrador

- Persona(s) **responsables** del **correcto** funcionamiento de un **sistema informático** (**sysop** (**root**)).
- ¿Correcto Funcionamiento? $\rightarrow$ "**Los sistemas cumplan las perspectivas de los usuarios**".
- **Crecimiento** de **tecnologías** en la empresa $\rightarrow$ Administrador de Sistemas con **conocimientos** sobre muy **diversos** aspectos.

Problemas a resolver por el administrador de sistemas

- Realizar **cableados** de instalaciones o reparar cables.
- **Instalar** sistemas operativos o software de aplicaciones.
- **Corregir** problemas y errores en los sistemas, tanto hardware como software.
- **Formar** a los usuarios.
- **Evaluar** económicamente **compras** de equipamiento de hardware y software.
- **Automatizar** un gran número de **tareas** comunes.
- **Incrementar** el **rendimiento** general del trabajo en su organización.

Subperfiles de la tarea del administrador

- Admin. de **Sistemas** Operativos.
- Admin. de Bases de **Datos**.
- Admin. de **Copias** de Seguridad.
- Admin. de **Seguridad** Informática.
- Admin. de **Red**.
- Admin. de **Soporte** Técnico.

Axioma

Mientras más **pequeña** sea la **empresa** mayor **número** de **responsabilidades** asumirá el **Administrador** del Sistema.

Índice

El **Administrador** de un entorno multiusuario:

- Debe **conocer** perfectamente tanto la **administración de la red** como la **administración de los sistemas operativos** conectados a ella.
- Ha de **suministrar** al resto de usuarios un entorno **fiable**, **seguro** y **estable** en el cual puedan desarrollar sus trabajos de forma efectiva.
- El papel del **Administrador** suele ser también **normativo**: tiene que definir unas reglas de **convivencia** y **aprovechamiento** del sistema para regular su uso (No Confundir Roles).

Para asegurar el correcto funcionamiento del sistema se debe:

- Hacer el sistema **fácil de usar**.
- **Anticiparse** a los **problemas** para evitarlos (mantenimiento proactivo).
- **Arreglar** problemas **rápidamente** (si no se pueden evitar).
- Implementar sistemas de **respaldo** y de **seguridad** (plan de contingencias).
- Asegurar el **funcionamiento correcto** del hardware y software.
- **Mantener** el software **actualizado**.
- Administrar las cuentas de los **usuarios**.

La necesidad de un Administrador se justifica:

- **No todos** pueden ser **expertos**: usuarios han de ser usuarios, coste en tiempo para la empresa, ...
- Incremento de **complejidad** de los sistemas: redes y comunicaciones.
- **Productividad** de la empresa depende **críticamente** de los sistemas informáticos.
- Problema: Mantenimiento.

Índice

Las tareas del administrador de un sistema informático:

- Son **cambiantes** y requieren de una **formación continua**.
- Requieren de **mano izquierda** para tratar con los usuarios.
- Aunque existan varios administradores debe haber un **coordinador general**.

Instalación de Servidores

El servidor es el elemento hardware más complejo del sistema y debe:

- Soportar **grandes** cantidades de **memoria**.
- Soportar y garantizar **altas** prestaciones de **HDD** de gran **capacidad** (RAID).
- Sistemas **multiprocesador**.
- **Tolerancia** a fallos: **UPS** (Uninterruptible Power Supply), **RAID** (Redundant Array of Independent Disks),...

Gestión de Servidores

El servidor debe mantenerse en funcionamiento correctamente:

- Crear un sistema de **generación** automático de **estadísticas**.
- Disponer de un criterio teórico para la **comparación** de los datos **estadísticos**.
- Detectar **necesidades** de **actualización** del software a partir de dichos datos.
- Detectar **insuficiencias** hardware que permitan establecer **políticas** de **reparto** de carga.

Instalación y Mantenimiento de la Red

El administrador debe conocer la configuración de la red:

- El **cableado** y la **instalación** de dispositivos puede **no** ser **responsabilidad** del administrador.
- La instalación de **dispositivos** de **red** y del **cableado** es una tarea de alta **complejidad**.
- El administrador se suele **responsabilizar** de su mantenimiento.
- El administrador debe realizar **cambios** menores de **configuración**.

Mantenimiento y Configuración de las Estaciones de Trabajo

- Configuración de las WS para **acceder** al **sistema**.
- Puede que **no** tengan porque tener el **mismo SO**, o que tengan versiones diferentes del mismo.
- PROBLEMA: **Preferencias** y **costumbres** de los usuarios pueden chocar con aspectos normativos de la entidad (criterio unificado).
- PROBLEMA: Coincidencia horaria.

Añadir y Eliminar Usuarios

- **Asignar** cuentas a **nuevos** usuarios.
- **Eliminar** cuentas de usuarios que causen **baja** (seguridad). ¿Archivos?
- Establecer **roles** de acceso para los **distintos** usuarios. ¿Usuario "experto" en informática?

Añadir y Eliminar Hardware

- La **reconfiguración** de las WS es un proceso **crítico**.
- Hardware **mal configurado** funcionaría por **debajo** de sus posibilidades.
- La **eliminación** de Hardware también requiere de una **reconfiguración**.

Realizar Copias de Seguridad

- Proceso que se debe **automatizar** ya que puede **consumir** mucho **tiempo**.
- **Complejidad** en la búsqueda de **horarios**, acciones que lleven a una necesidad de **actualizar** una copia.
- Es una de las **tareas más importantes** del Administrador. Debe ser siempre **supervisada**.
- Tiene que existir una **política de seguridad** para las copias.

Otras Tareas

- **Instalar** software nuevo.
- **Monitorizar** el sistema (log).
- Solución de **problemas**.
- Mantenimiento de **documentación**.
- Reparar **errores**.
- Mantenimiento de la **seguridad**.
- **Enseñar** y **ayudar** a los usuarios.
- **Contabilidad** del sistema.

Índice

Temas que conforman la filosofía de la administración de sistemas:

- **Automatizar** todo.
- **Documentar** todo.
- **Comunicar** tanto como sea posible.
- **Conocer** sus **recursos**.
- **Conocer** sus **usuarios**.
- **Conocer** el **negocio**.
- La **seguridad no** puede ser una ocurrencia **posterior**.
- **Planifique**.
- Espere lo **inesperado**.

Documentar Todo

- Más **tarde** lo hago.
- ¿Para qué escribirlo? Yo **me acuerdo**.
- Si lo mantengo en mi memoria, no me despedirán, así ¡Tendré **seguridad laboral**!

¿Qué hay que documentar?

- Políticas.
- Procedimientos.
- Cambios.

Temas que conforman la filosofía de la administración de sistemas:

- **Automatizar** todo.
- **Documentar** todo.
- **Comunicar** tanto como sea posible.
- **Conocer** sus **recursos**.
- **Conocer** sus **usuarios**.
- **Conocer** el **negocio**.
- La **seguridad no** puede ser una ocurrencia **posterior**.
- **Planifique**.
- Espere lo **inesperado**.

Comunica tanto como sea posible

- **Pequeños** cambios pueden **confundir** completamente al personal de la empresa.
- Hay que **elegir** el medio de **comunicación** en función de los siguientes criterios.
  - **Política** general de la **empresa**.
  - **Número** de **usuarios** al que va dirigido.
  - Estructura **departamental** de la empresa.
  - El correo electrónico puede ser muy peligroso.
- Hay que **advertir** a los usuarios con la **antelación** justa.
- Una vez **realizadas** las modificaciones hay que **indicarlo**.

Temas que conforman la filosofía de la administración de sistemas:

- **Automatizar** todo.
- **Documentar** todo.
- **Comunicar** tanto como sea posible.
- **Conocer** sus **recursos**.
- **Conocer** sus **usuarios**.
- **Conocer** el **negocio**.
- La **seguridad no** puede ser una ocurrencia **posterior**.
- **Planifique**.
- Espere lo **inesperado**.

La seguridad no puede ser una ocurrencia posterior.

- **Nunca** asumir la seguridad como algo **garantizado**.
- Los **peligros no** vienen todos de **Internet**.
- **Nunca** enfocar la seguridad desde el punto de vista **forense**.
- Establecer **políticas de acceso** estrictas.

Temas que conforman la filosofía de la administración de sistemas:

- **Automatizar** todo.
- **Documentar** todo.
- **Comunicar** tanto como sea posible.
- **Conocer** sus **recursos**.
- **Conocer** sus **usuarios**.
- **Conocer** el **negocio**.
- La **seguridad no** puede ser una ocurrencia **posterior**.
- **Planifique**.
- Espere lo **inesperado**.

Conclusiones

- **Pequeños cambios** pueden tener graves consecuencias.
- Problemas **técnicos** mínimos comparados con problemas de **trato personal**.
- Lo difícil es conocer las posibilidades de los **recursos humanos**.
- Limitación del uso del **correo electrónico**.
- **Limitaciones** de descripción de problemas por parte del usuario (**teléfono**).
- **No minimizar** las consecuencias en el personal de **pequeños cambios** ni de la **complejidad** de cualquier **tarea**.

Índice

Diagnosis del Sistema I

top: muestra los procesos en ejecución en tiempo real (Figura 1)

Uso:

top [-] [d delay] [p pid] [q] [c] [C] [S] [s] [i] [n iter] [b]

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1255	root	20	0	348444	109084	37696	S	1,3	5,3	2:31.03	Xorg
3096	alumno	20	0	460228	15724	11000	S	0,7	0,8	0:00.69	xfce4-scre+
3095	**alumno**	**20**	**0**	**32228**	**1776**	**1244**	**R**	**0,3**	**0,1**	**0:00.09**	**top**
1	root	20	0	33768	3156	1476	S	0,0	0,2	0:03.07	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.02	kthreadd

Figura 1: top

Diagnosis del Sistema II

Valores de salida de *top* I

- **PID**: Identificador de proceso.
- **USER**: Usuario propietario del proceso.
- **PR**: Prioridad del proceso.
- **NI**: El valor nice del proceso, un valor que asigna prioridad al proceso. Un valor negativo de nice indica prioridad alta mientras que positivo prioridad baja.

Diagnosis del Sistema III

Valores de salida de *top* II

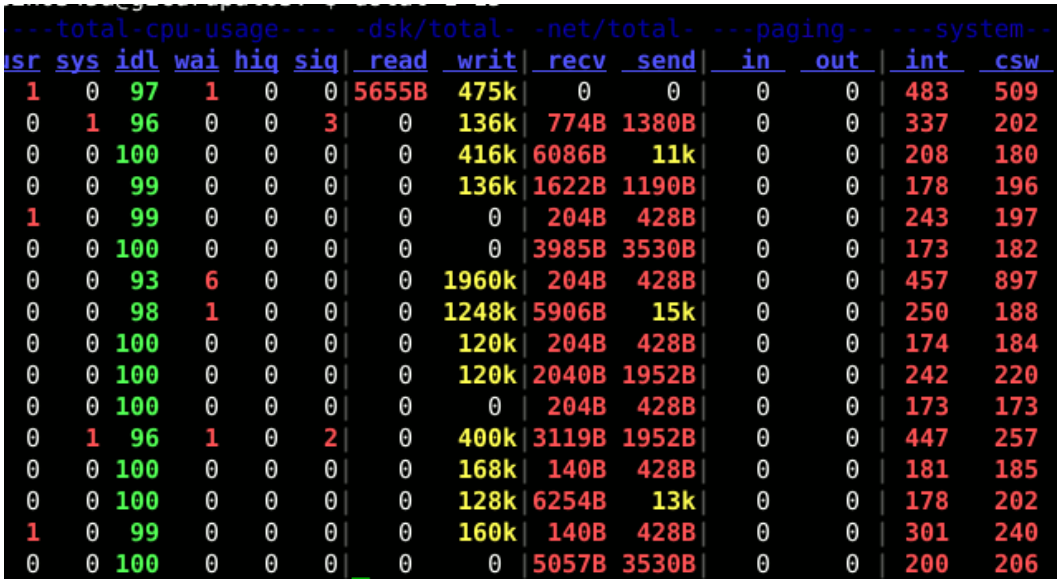
- **VIRT**: Memoria virtual usada por el proceso.
- **RES**: Memoria física usada por el proceso.
- **SHR**: Memoria compartida del proceso.
- **S**: Indica el estado del proceso: S=sleep R=running Z=zombie.
- **%CPU**: Porcentaje de CPU utilizada por el proceso.
- **%MEM**: Porcentaje de RAM utilizada por el proceso.
- **TIME+**: Tiempo de actividad total del proceso.
- **COMMAND**: Nombre del proceso. Ej: envice.

Diagnosis del Sistema IV

dstat: informes sobre el uso de recursos del sistema (Figura 2)

Uso:

dstat [-a] [-f]



---total-cpu-usage----						-dsk/total-		-net/total-		---paging--		---system--	
usr	sys	idl	wai	hiq	siq	read	writ	recv	send	in	out	int	csw
1	0	97	1	0	0	5655B	475k	0	0	0	0	483	509
0	1	96	0	0	3	0	136k	774B	1380B	0	0	337	202
0	0	100	0	0	0	0	416k	6086B	11k	0	0	208	180
0	0	99	0	0	0	0	136k	1622B	1190B	0	0	178	196
1	0	99	0	0	0	0	0	204B	428B	0	0	243	197
0	0	100	0	0	0	0	0	3985B	3530B	0	0	173	182
0	0	93	6	0	0	0	1960k	204B	428B	0	0	457	897
0	0	98	1	0	0	0	1248k	5906B	15k	0	0	250	188
0	0	100	0	0	0	0	120k	204B	428B	0	0	174	184
0	0	100	0	0	0	0	120k	2040B	1952B	0	0	242	220
0	0	100	0	0	0	0	0	204B	428B	0	0	173	173
0	1	96	1	0	2	0	400k	3119B	1952B	0	0	447	257
0	0	100	0	0	0	0	168k	140B	428B	0	0	181	185
0	0	100	0	0	0	0	128k	6254B	13k	0	0	178	202
1	0	99	0	0	0	0	160k	140B	428B	0	0	301	240
0	0	100	0	0	0	0	0	5057B	3530B	0	0	200	206

Figura 2: dstat

Diagnosis del Sistema V

Estadísticas *dstat*

- **CPU**: Estadísticas varias sobre CPU. Las más interesantes las secciones de usuario, sistema o idle.
- **Disco**: Actividad de lectura y escritura en disco, el total de todos los discos si hay varios.
- **Red**: Datos enviados y recibidos, el total de todas las redes si hay varias.
- **Paginación**: Actividad de paginación del sistema.
- **Sistema**: Interrupciones (int) y cambios de contexto (csw).

Diagnosis del Sistema VI

Obtener información sobre procesos

ps -auxw:permite obtener la información sobre los procesos

- -a Muestra todos los procesos menos "lider" del grupo de procesos
- -u Ordenados por usuario efectivo
- -x Junto con opción -a, muestra todos los procesos.
- -w salida ensanchada (ocupa todo el ancho del terminal)

Ejemplo de uso Figura(3)

ps -auxw

Diagnosis del Sistema VII

```
avahi      419  0.0  0.0  25756   196 ?        S    10:57   0:00 avahi-daemon: chroot helper
root       422  2.8  1.7 748716 143180 tty1     Ssl+ 10:57   2:16 /usr/bin/Xorg.bin :0 -auth /v
root       442  0.1  0.2 118352 16440 ?        S    10:57   0:08 /usr/bin/python2 -O /usr/shar
root       450  0.0  0.0      0     0 ?        S<   10:57   0:00 [ttm_swap]
root       453  0.0  0.0      0     0 ?        S<   10:57   0:00 [radeon-crtc]
root       454  0.0  0.0      0     0 ?        S<   10:57   0:00 [radeon-crtc]
root       455  0.0  0.0      0     0 ?        S<   10:57   0:00 [radeon-crtc]
root       456  0.0  0.0      0     0 ?        S<   10:57   0:00 [radeon-crtc]
root       457  0.0  0.0      0     0 ?        S<   10:57   0:00 [radeon-crtc]
root       458  0.0  0.0      0     0 ?        S<   10:57   0:00 [radeon-crtc]
sddm       466  0.0  0.0  35652  4132 ?        Ss   10:57   0:00 /usr/lib/systemd/systemd --us
```

Figura 3: ps -auxw /

Diagnosis del Sistema VIII

Obtener módulos del kernel cargados (cat /proc/modules)

lsmod

Salida de *lsmod*

Module	Size	Used by
vboxsf	42558	0
rfcomm	53664	4
bnep	18895	2
bluetooth	342208	10 bnep,rfcomm
...	...	...

Utilización de memoria: *sar -B* I

Sar permite organizar y ver datos sobre la actividad del sistema, acceder a esos datos de actividad y generar informes automáticos para supervisar el rendimiento.

Estadísticas sobre paginación.

sar *[opciones] [delay[tiempo]]*

Estadísticas de *sar -B 1*

Linux 3.13.0-45-generic (nombre-PC) 04/02/15 -x86-64 (8 CPU)

13:02:44	pgpgin	pgpgout	fault	majflt	pgfre	pgscank	pgscand	pgsteal	vmeff
13:02:45	0,00	4,00	14,33	0,00	59,33	0,00	0,00	0,00	0,00
13:02:46	0,00	4,00	10,33	0,00	393,33	0,00	0,00	0,00	0,00

Utilización de memoria: *sar -B* II

Descripción de estadísticos de **sar -B**:

- pgpgin: kilobytes que el sistema cargó de disco por segundo.
- pgpgout: kilobytes que el sistema cargó en disco por segundo.
- fault: fallos de página (mayores+menores) por segundo.
- majflt: fallos/s que han necesitado la carga de una página.
- pgfree: páginas colocadas en la lista por segundo.
- pgscank: páginas escaneadas por *kswapd daemon* por segundo.
- pgscand: páginas escaneadas directamente por segundo.
- pgsteal: páginas que el sistema ha recuperado de la memoria caché (memoria intermedia de páginas y swapcache) por segundo para satisfacer sus demandas de memoria.
- vmeff: calculado como pgsteal / pgscan. Métrica de la eficiencia de la página de recuperación.

Utilización de memoria: *free* I

Free muestra la cantidad de memoria libre y usada por el sistema, tanto física como swap. También memoria caché y de buffer consumida por el kernel.

Estadísticas sobre la memoria del sistema

free *[opciones]*

Estadísticas de *free*

	total	usado	libre	compart.	búffers	almac.
Mem:	1025792	504220	521572	2536	49876	242420
-/+ buffers/cache:		211924	813868			
Intercambio:		1047548	0	1047548		

Utilización de memoria: *free* II

Descripción de estadísticos de **free**:

- Línea 1: indica la información respectiva al total de memoria RAM disponible, RAM en uso (incluyendo buffer y cache), RAM libre, RAM compartida, RAM usada para buffers y RAM usada para cachear información.
- Línea 2: Indica el total de buffers/cache usado y disponible.
- Línea 3: Indica el total de memoria disponible para swap, cantidad en uso de swap y cantidad libre.

Sistemas de ficheros y dispositivos de almacenamiento I

Espacio usado en **sistemas de ficheros**.

df -k

- Informa de su **capacidad**, del espacio **usado** y del **disponible** en todos los sistemas de ficheros **montados actualmente** en **unidades de 1024 bytes**.
- Donde está montado.
- Información *concreta* de un sistema de ficheros (df -k */dev/sda1*)

```
usuario@rufus:~$ df -k
S.ficheros     bloques de 1K    Usados Disponibles Uso% Montado en
/dev/sda5          175510184 110696656    55898076  67% /
udev                   10240         0       10240   0% /dev
tmpfs                 411160      1044      410116   1% /run
```

Figura 4: df -k

Sistemas de ficheros y dispositivos de almacenamiento II

Espacio utilizado por un **directorio** dado y sus **subdirectorios**.

du -sh

- Informa del espacio usado por la jerarquía de ficheros por debajo del directorio o fichero especificado.
- **-s** Imprime solamente el uso de espacio del fichero o directorio dado no de los subdirectorios.
- **-h** Salida en KiB,MiB,GiB.

```
usuario@rufus:~$ du -sh
84G     .
```

Figura 5: du -sh

Sistemas de ficheros y dispositivos de almacenamiento III

Particiones swap, tamaño, cantidad usada, etc

cat /proc/swaps

Salida de *cat /proc/swaps*

Filename Type Size Used Priority
/dev/sda5 partition 1047548 0 -1

Información de usuario I

Muestra los últimos usuarios que han entrado en el sistema, cúando lo han hecho, y si aún siguen en él. Se lee del archivo /var/log/wtmp

last
last -*n* (n = nº de últimas conexiones)

Las últimas conexiones del usuario alumno son:

alumno pts/1 :0.0 Wed Feb 4 11:36 still logged in
root pts/8 :0.0 Mon Jan 19 11:21 - 11:22 (00:00)
reboot system boot 3.13.0-32-generi Fri Jan 16 12:35 - 12:36 (00:00)
wtmp begins Fri Jan 16 12:35:16 2015

Información del Hardware I

Listado de puertos para I/O.

cat */proc/devices*

Este archivo muestra los diversos dispositivos de carácteres y de bloque actualmente configurados (no incluye dispositivos cuyos módulos no están cargados). Una salida de datos de ejemplo de este archivo quedaría de la siguiente manera:

Información del Hardware II

Estadísticas de *cat /proc/devices*

Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttys
5 /dev/tty
5 /dev/console
Block devices:
1 ramdisk
3 ide0
9 md
22 ide1
253 device-mapper
254 mdp

Diagnosis de Red I

Comprobar conexión con un host

ping *direcciónWeb*
ping *direcciónIP*

Estadísticas de *ping google.com*

— google.com ping statistics —
5 packets transmitted, 5 received, 0 % packet loss, time 4006ms
rtt min/avg/max/mdev = 4.197/4.274/4.335/0.070 ms

Diagnosis de Red II

Informe del camino a una máquina remota

traceroute *direccion*

- Paquetes **perdidos** en primeros **elementos** de la **ruta** indican problemas en la **LAN**.
- Al contrario indican **problemas** de conexión del **servidor**.
- Las conexiones **fallidas** se **indican** mediante una línea con tres **asteriscos**.

Diagnosis de Red III

Informe del camino a una máquina remota + Estadísticas

mtr -*direccion*
mtr -rwc 10 *direccion*
mtr –no-dns –report *direccion*

- Se puede considerar un **híbrido** entre el **ping** y el **traceroute**.
- **MTR** es una herramienta **potente** que permite a los **administradores** del sistema **identificar** y aislar **errores** de **red**.
- Existe una **versión** para sistemas **Windows** llamada **WinMTR**.

Diagnosis de Red IV

Los errores que pueden ser diagnosticados son:

- Host **destino** mal **configurado**.
- **Router** del ISP mal **configurado**.
- Límites a **conexiones** ICMP. **Pueden** derivar en **paquetes** perdidos.

Índice

Ejecución del interfaz de comandos

- **Ejecución**: Windows + R ⇒ cmd;
- **Ejecución** como **Administrador**: Buscar cmd; Botón derecho ejecutar como administrador.
- **Ejecutar archivo** como **Administrador**:
  - runas /user:[usuario con privilegio] [ubicación del archivo ejecutable]
  - runas /user:administrador /home/alumno/ejecutable
- **Activar/Desactivar** la cuenta de **administrador**:
  - net user administrador /active:yes
  - net user administrador /active:no

Equivalencia con comandos Unix I

- **arp** → arp: **correspondencia** direcciones **IP-Físicas** (*arp -a*).
- **assoc**: muestra y modifica asociaciones de extensiones y programas.
- **at** → at. Ya en desuso (schtasks.exe).
- **attrib** → chmod: muestra o cambia los atributos de un archivo.
- **cd** → cd: cambio de directorio. En Unix sin argumentos cambia al directorio $HOME.
- **cd** → pwd: sin argumentos.
- **chkdsk** → fsck: chequeo de sistemas de ficheros y reparación en caso de daños.

Equivalencia con comandos Unix II

- **cls** → clear
- **copy** → cp
- **date; time** → date: en Windows aparte de mostrar la hora y fecha permite cambiarlas.
- **del** → rm
- **deltree** → rm -r
- **dir** → ls: "dir"también funciona en algunos Linux.
- **doskey /h; F7 key** → history. doskey permite hacer macros.

Equivalencia con comandos Unix III

- **edit** → vi; emacs, etc. En Unix, la variable $EDITOR puede contener el nombre del editor preferido. (No viene por defecto instalado).
- **exit** → exit; Control-D
- **explorer** → nautilus; etc.
- **fc** → diff
- **find** → grep
- **ftp** → ftp

Equivalencia con comandos Unix IV

- **help** → man
- **hostname** → hostname
- **ipconfig /all** → ifconfig -a
- **mem** → top
- **mkdir; md** → mkdir
- **ln** → mklink
- **more** → more
- **move** → mv

Equivalencia con comandos Unix V

- **nslookup** → nslookup. Sin argumentos entre en el modo interactivo.
- **ping** → ping
- **reboot; shutdown -r** → shutdown -r
- **regedit** → edit /etc/*: En Unix los ficheros bajo los directorios /etc y /usr/local/etc serían los equivalentes a los registros Windows pero aquí pueden ser editados como texto plano.
- **rmdir** → rmdir
- **rmdir /s** → rm -r. Windows por defecto muestra un prompt y/n. En Unix se usa rm -i.

Equivalencia con comandos Unix VI

- **set** → env. Para mostrar una variable individual en Windows set < *variable* > es el equivalente a echo $< *variable* > en Unix.
- **sort** → sort
- **start** → &: ejecución en segundo plano.
- **systeminfo** → uname -a
- **tasklist** → ps.

Equivalencia con comandos Unix VII

- **tracert** → traceroute
- **tree** → find; ls -R
- **type** → cat
- **ver** → uname -a
- **xcopy** → cp -R.

Herramientas Administrativas I

- Windows **contiene** una gran **cantidad** de **herramientas** de sistema.
- Suelen **almacenarse** en la **carpeta** "Herramientas Administrativas" (**Administrative Tools**)
- Son **potentes** y **complejas**: no accesibles para **usuarios**.
- Algunas de ellas **sólo** están **disponibles** en versiones **Profesional** o **Enterprise**.
- Las **estudiadas** pertenecen a Windows 8.1 **Professional**.
- El **comando** *control admintools* permite acceder a estas **herramientas**.

Herramientas Administrativas II

Servicios de Componentes (Component Services)

Configuración y administración de componentes COM y COM+.
Ejecución: comexp.msc, dcomcnfg

Administración de equipos (Computer Management)

Integra varias herramientas: Shared folders, Local and User groups y Disk Management
Ejecución: compmgmtlauncher, compmgmt.msc

Herramientas Administrativas III

Optimizar Unidades (Optimize Drives)

Así se llama el Defragmentador a partir de las versiones 8 y 8.1. Trabaja con SSD y discos duros mecánicos. SSOO modernos incluyen defragmentación automática.
Ejecución: defrag, dfrgui.

Liberador de espacio en disco. (Disk Cleanup)

Elimina ficheros innecesarios, temporales, ficheros desinstalados por windows, etc.
Ejecución: cleanmgr.

Herramientas Administrativas IV

Visor de Eventos. (Event Viewer)

Muestra el log de eventos. Puede ayudar a detectar errores aunque la mayoría de mensajes son irrelevantes.
Ejecución: eventvwr.msc .

Iniciador iSCSI (iSCSI Initiator Configuration Tool)

Uso de protocolo SCSI a través de redes TCP/IP.
Ejecución: iscsicpl.

Herramientas Administrativas V

Directiva de seguridad local. (Local Security Policy)

Establecer políticas de seguridad como pueden ser las relacionadas con la gestión de contraseñas.
Ejecución: secpol.msc.

Administrador de orígenes de datos ODBC. (ODBC Data Sources)

Permite configurar los drivers OBDC para intercomunicación y las fuentes de datos. Se debe diferenciar entre aplicaciones de 32 bits y 64 bits.
Ejecución: odbcad32.

Herramientas Administrativas VI

Monitor de Rendimiento. (Performance Monitor)

Es una herramienta para administradores que permite generar informes de diagnóstico y rendimiento del sistema.
Ejecución: perfmon.msc.

Administrador de Impresion. (Print Management)

Interfaz detallada para visualizar y manejar las impresoras y los trabajos de impresión.
Ejecución: printmanagement.msc

Herramientas Administrativas VII

Monitor de recursos. (Resource Monitor)

Información sobre la utilización de recursos hardware CPU, disco, red, y memoria. También se puede visualizar esta utilización por aplicación.
Ejecución: resmon.

Servicios. (Services)

Servicios instalados en el sistema y gestión de los mismos. Los servicios son programas de bajo nivel ejecutados en segundo plano. Cuidado con la desactivación de servicios.
Ejecución: services.msc.

Herramientas Administrativas VIII

Configuración del sistema. (System Configuration)

Establece las opciones de arranque y de inicio.
Ejecución: msconfig.

Información del sistema. (System Information)

Componentes Hardware instaladas en nuestro computador así como su configuración y el modelo. También muestra alguna información del sistema como son las variables de entorno.
Ejecución: msinfo32.

Herramientas Administrativas IX

Planificador de Tareas (Task Scheduler)

Ejecución, visualización y gestión de procesos planificados.
Ejecución: taskschd.msc

Funciones avanzadas del firewall (Windows Firewall with Advanced Security)

Más avanzado que el firewall ya que permite crear y gestionar reglas avanzadas como que sólo ciertas aplicaciones se conecten a Internet o que un programa servidor sólo atienda peticiones de ciertas IPs.
Ejecución: wf.msc (firewall.cpl)

Herramientas Administrativas X

Herramienta de diagnóstico de la memoria (Windows Memory Diagnostic)

Chequea la RAM para detectar defectos. Escribe datos en diferentes sectores de la RAM y los lee. Si hay diferencias es cuando se detecta un mal funcionamiento de la memoria.
Ejecución: mdsched

Windows PowerShell + Powershell ISE

Es el sucesor del Windows Command Prompt. Es una interfaz de comandos potente ya que permite el desarrollo de scripts. Windows PowerShell ISE proporciona una interfaz gráfica al PowerShell.
Ejecución: powershell; powershell_ise

Bibliografía

- Gómez López, Julio. **Administración de sistemas GNU-Linux**. Starbook, 2010.
- Colobran Huguet, Miquel. **Administración de sistemas operativos en red**. UOC, 2008.
- Gómez López, Julio. **Administración de sistemas operativos Windows y Linux: un enfoque práctico**. RA-MA, 2006.

Tema 1. Administración de Sistemas

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Tema 2. Automatización

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Índice

Objetivos

- Estudiar los **fundamentos teóricos** del reconocimiento de **patrones** mediante la utilización de **Expresiones Regulares**.
- Realizar una **introducción** a la **automatización** de **bajo nivel** mediante la **programación** BASH y POWERSHELL.
- Estudiar los procesos de **automatización** de tareas en **sistemas** mediante la utilización de herramientas **remotas** y de **alto nivel**.

Índice

Introducción

- Ciertos aspectos de la **administración** necesitan del **reconocimiento** de **patrones**.
- Este **reconocimiento** se realiza mediante una serie de **comandos** específicos que **toman** como **argumentos Expresiones Regulares**.
- Antes de estudiar las mismas se van a establecer sus **fundamentos teóricos** realizando las **definiciones** básicas de **lenguajes** y operaciones con **palabras** y con los propios lenguajes.

Definiciones básicas

Alfabeto (Σ): conjunto no vacío finito de símbolos.

$$\Sigma_1 = \{\text{na, pa, la, bra}\}$$

Palabra: secuencia finita de símbolos de un alfabeto.

lapa, nala, branala, ...

Longitud de una palabra ($|x|$): número de símbolos del alfabeto que la forman.

- $|branala| =$

Definiciones básicas

Alfabeto (Σ): conjunto no vacío finito de símbolos.

$$\Sigma_1 = \{\text{na, pa, la, bra}\}$$

Palabra: secuencia finita de símbolos de un alfabeto.

lapa, nala, branala, ...

Longitud de una palabra ($|x|$): número de símbolos del alfabeto que la forman.

- $|branala| =$
- Sobre Σ_1 $|branala| = 3$
- Sobre Alf. Español $|branala| = 7$

Definiciones básicas

Sobre $\Sigma_3 = \{z,l,m,cd,hv\}$

- x=zzlm $\rightarrow |x| =$
- x'=cdhvzz $\rightarrow |x'| =$

- **Palabra vacía** (λ): $|\lambda| = 0$
- **Universo** de un alfabeto ($W(\Sigma)$): Todas las palabras que se pueden formar con símbolos de Σ.

Universo sobre Σ_1

$$W(\Sigma_1) = \{\lambda, na, pa, la, bra, napa, nala, ...\}$$

Definiciones básicas

- **Lenguaje** sobre un alfabeto (L(Σ)): cualquier subconjunto de W(Σ).

Ejemplos de lenguajes sobre Σ_1

$$L_1\ (\Sigma_1) = \{\text{nana, napa, lana}\}$$
$$L_2\ (\Sigma_1) = \{\lambda, \text{pana, palabra, pala}\}$$

Operaciones con palabras

- x e y son palabras, la **concatenación** $x \cdot y$, es: palabra formada por los símbolos de x seguidos de los símbolos de y

En Σ_1 si x = pa e y = labra

$x \cdot y$ = palabra

- La **potencia iésima** de x concatena i veces x.

En Σ_2={0, 1} y x=10

x^5 =1010101010
x^0 = ?

Operaciones con palabras

Sobre $\Sigma_3 = \{z, l, m, cd, hv\}$, x=cdz, y=hvm

xy=

Sobre $\Sigma_3 = \{z, l, m, cd, hv\}$ y w=zlcd

$w^0 =$
$|w^1| =$
$w^2 =$
$w^3 =$

Operaciones con palabras

- **Reflexión**: si $x = A_1A_2...A_n$ entonces la palabra inversa de x es: $x^{-1} = A_n...A_2A_1$
- En Σ_1 si x = pala entonces x^{-1} = lapa

Sobre $\Sigma_3 = \{z, l, m, cd, hv\}$

x= zllm; x^{-1} =
x'= cdlz; x'^{-1} =

Operaciones con lenguajes

- La **unión de dos lenguajes** contiene las palabras que pertenezcan a cualquiera de los dos lenguajes.

- $L_1(\Sigma_1) = \{\text{nana, napa, lana}\}$
- $L_2(\Sigma_1) = \{\lambda, \text{nana, pana, palabra, pala}\}$
- $L_1(\Sigma_1) \cup L_2(\Sigma_1) = \{\lambda, \text{nana, pana, palabra, pala, napa, lana}\}$

Operaciones con lenguajes

- La **intersección** de dos lenguajes contiene las palabras que pertenezcan a los dos lenguajes.

- $L_1(\Sigma_1) = \{\text{nana, napa, lana}\}$
- $L_2(\Sigma_1) = \{\lambda, \text{nana, pana, palabra, pala}\}$
- $L_1(\Sigma_1) \cap L_2(\Sigma_1) = \{\text{nana}\}$

Operaciones con lenguajes

- La **resta** de $L_1(\Sigma_1)$ y $L_2(\Sigma_1)$ contiene las palabras que pertenecen a L_1 y no pertenecen a L_2.

- $L_1(\Sigma_1) = \{\text{nana, napa, lana}\}$
- $L_2(\Sigma_1)) = \{\lambda, \text{nana, pana, palabra, pala}\}$
- $L_1(\Sigma_1)$ - $L_2(\Sigma_1) = \{\text{napa, lana}\}$
- $L_2(\Sigma_1)$ - $L_1(\Sigma_1) = \{\lambda, \text{pana, palabra, pala}\}$

Operaciones con lenguajes

- La **concatenación** contiene las palabras formadas por la concatenación de palabras de L_1 y de L_2.

- $L_1(\Sigma_1) = \{\text{nana, napa, lana}\}$
- $L_2(\Sigma_1) = \{\lambda, \text{nana, pana, palabra, pala}\}$
- $L_1(\Sigma_1) \cdot L_2(\Sigma_1) = \{\text{nana, napa, lana, nananana, nanapana, nanapalabra, ...}\}$
- ¿Por qué aparece "napa" que sólo pertenece a L_1?

Operaciones con lenguajes

$\Sigma_3 = \{z, l, m, cd, hv\}$ con $L_1 = \{zl\}$ y $L_2 = \{cd, m\}$

L= $L_1 \cdot L_2 = ?$

L= $L_2 \cdot L_1 = ?$

Operaciones con lenguajes

- La **potencia iésima** de un lenguaje es la concatenación i veces del lenguaje con él mismo. La potencia cero de un lenguaje genera un lenguaje con un único elemento que es la **palabra vacía**.

- $L_1(\Sigma_2) = \{0, 1\}$
- $L_1(\Sigma_1)^2 = \{00, 01, 10, 11\}$

Operaciones con lenguajes

L = {cd}

- $L^0 = \{\lambda\}$
- $L^1 = ?$
- $L^2 = ?$
- $L^3 = ?$

Operaciones con lenguajes

L = {cdz, l}

- $L^0 = ?$
- $L^1 = ?$
- $L^2 = ?$
- $L^3 = ?$

Operaciones con lenguajes

- La **clausura positiva** de un lenguaje L es la unión de todas sus potencias (salvo la cero).

- $L^+ = \bigcup_{i=1}^{\infty} L^i$
- $\Sigma^+ = W(\Sigma)$ - { λ }

$L_1(\Sigma_2) = \{0, 1\}$

$L_1(\Sigma_2)^+ = \{0, 1, 00, 01, 10, 11, 000, 001, ...\}$

Operaciones con lenguajes

- El **cierre o clausura** de un lenguaje L es la unión de la cadena vacía a la clausura positiva del lenguaje.

- $L^* = \bigcup_{i=0}^{\infty} L^i$
- $\Sigma^* = W(\Sigma)$

$L_1(\Sigma_2) = \{0, 1\}$

$L_1(\Sigma_2)^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$

Operaciones con lenguajes: Ejercicios I

Σ= {a,b,c,d}

Diseñe un lenguaje que contenga todas las cadenas que empiecen en a y acaben en b.
L=

Σ= {0,1,2}

Diseñe un lenguaje que contenga todas las cadenas de longitud 3 que sean múltiplos de 2, salvo la cadena 210 y la 222.
L=

Operaciones con lenguajes: Ejercicios II

Σ= {a,b}

Diseñe un lenguaje formado por todas las palabras que comienzan por “a”.
L=

Σ= {a,b}

Diseñe un lenguaje formado por todas las palabras que contienen la subcadena “bb”.
L=

Operaciones con lenguajes: Ejercicios III

Σ= {0,1,2,..,9}

Diseñe un lenguaje formado por todos los números naturales, evitar los números que contienen ceros a la izquierda.
L=

Σ= {0,1,2,...,9}

Diseñe un lenguaje formado por todos los números enteros, evitar los números que contienen ceros a la izquierda.
L=

Operaciones con lenguajes: Ejercicios IV

$\Sigma = \{0,1,2,\ldots,9\}$

Diseñe un lenguaje formado por el conjunto de los números divisibles por 5.
L=

Σ=?

Definir el lenguaje formado por todas aquellas direcciones de correo electrónico bien formadas de los alumnos de la UCLM.
L=

Expresiones Regulares I

- Las **Expresiones Regulares (ER)** son fórmulas que representan de manera concisa **Lenguajes Regulares**.

Además se **emplean** para describir **filtros** + cadenas **válidas** en:

- **Lenguajes** de **programación**
- **Utilidades** de **programación**
- **Software** en general

ER sobre un alfabeto

- $\emptyset$ es una ER.
- λ es una ER.
- Cada símbolo de Σ es una ER.
- α y β son ER: $\alpha+\beta$ es una ER
- α y β son ER: $\alpha \cdot \beta$ $(\alpha\beta)$ es una ER
- α es una ER: α* es una ER
- α es una ER: (α) es una ER

Lenguaje representado por una ER

Toda ER α define **recursivamente** un **lenguaje** L mediante las siguientes **reglas**:

- $\alpha = \emptyset \rightarrow \mathrm{L}(\alpha) = \emptyset$
- $\alpha = \lambda \rightarrow \mathrm{L}(\alpha) = \{\lambda\}$
- α=a y $a \in \Sigma \rightarrow \mathrm{L}(\alpha)=\{\mathrm{a}\}$
- α y β son ER $\rightarrow \mathrm{L}(\alpha+\beta)=\mathrm{L}(\alpha) \cup \mathrm{L}(\beta)$
- α y β son ER $\rightarrow \mathrm{L}(\alpha \cdot \beta)=\mathrm{L}(\alpha) \cdot \mathrm{L}(\beta)$
- α es ER $\rightarrow \mathrm{L}(\alpha^*)=\mathrm{L}(\alpha)^*$
- α es ER $\rightarrow \mathrm{L}((\alpha))=\mathrm{L}(\alpha)$

Precedencia de operadores I

De **mayor** a **menor** precedencia:

- **Paréntesis**.
- **Clausura**: aplicada a la ER más **pequeña** que quede a su **izquierda**.
- **Concatenación**.
- **Unión** $(+)$.

Precedencia de operadores II

la **expresión** 01*+1, ¿Qué lenguaje **representa**?

- $1^* \rightarrow \{\lambda, 1, 11, 111,...\}$
- $01^* \rightarrow \{0, 01, 011, 0111, ..\}$
- $01^* + 1 \rightarrow \{0, 1, 01, 011, 0111, ...\}$

"La cadena 1 más todas las cadenas que comienzan por cero y están seguidas por cualquier número de unos"

la expresión (0+1)*+1, ¿Qué lenguaje representa?

la expresión 0+(1*+1), ¿Qué lenguaje representa?

Propiedades Algebraicas

- Se puede **obtener** más de una **expresión** regular para **generar** un **mismo** lenguaje.
- α y β son equivalentes si $\mathrm{L}(\alpha) = \mathrm{L}(\beta)$
- Las propiedades **algebraicas** van a permitir la **simplificación** de ER para **obtener** la solución más **sencilla**.

Ejercicios

Sea $\Sigma = \{0,1\}$ genere el lenguaje que contenga todas las cadenas:

- de longitud par.
- de longitud impar.
- que tienen al menos un 1.
- cuya longitud es igual o menor que 4.
- cuya longitud es mayor que 6.

ER básicas I

- Los caracteres son tratados como **literales**.
- . **cualquier** carácter
- [ ] cualquiera de los **caracteres** entre **corchetes**, p.e. [abc] concuerda con a, b o c; [a-z] concuerda con cualquier letra **minúscula**.
- | **alternancia**: [abc] equivale a a|b|c.
- [^] cualquier carácter que no esté entre **corchetes**.

ER básicas II

- Dentro de [ ] los **metacaracteres** pierden su **significado** especial: p.e. [a.]c **concuerda** con ac y .c
- \(\) permite agrupar ER
- ^ **principio** de línea
- {n} n **ocurrencias** de la RE previa
- {n,} **al menos** n ocurrencias de la RE **previa**
- {n, m} **entre** n y m ocurrencias de la RE previa

ER extendidas (egrep) + metacaracteres I

- $+ \rightarrow \alpha^{+} \rightarrow \alpha\alpha*$
- ?: r? $\rightarrow$ r+λ
- [:space:] caracteres en blanco
- [:blank:] espacio y tabulado
- [:alnum:] caracteres alfánuméricos (letras y números)
- [:digit:] dígitos
- [:alpha:] alfabéticos

ER extendidas (egrep) + metacaracteres II

- [:upper:] mayúsculas
- [:lower:] minúsculas
- [:xdigit:] dígitos hexadecimales
- [:punct:] signos de puntuación
- [:cntrl:] caracteres de control

Ejercicios

Caracterice los lenguajes generados por las siguientes ER:

- a..c
- 0[abc]0
- 0[^abc]0
- 0[a-z]0
- ^abc
- abc$
- ab*c
- b[cq]*e
- .*
- abc.*
- 0 \(*abc*\)*0

Conclusiones

- Las **expresiones regulares** se **rigen** por una serie de **normas** y hay una **construcción** para cualquier **patrón** de caracteres.
- **No** existe un **lenguaje estándar** para las expresiones regulares.
- **DIFERENTES DIALECTOS**
- Todos los **dialectos** siguen los **mismos principios**.

Índice

Comandos útiles I

- **cat**: **concatenación** de ficheros.
- **cp**: copia de **ficheros**.
- **date**: fecha y **hora**.
- **grep**: **búsqueda** de cadenas.
- **head**: **primeras** líneas.
- **tail**: ultimas líneas.
- **mv**: **mover**.
- **wc**: **conteo** de lineas, **palabras** y caracteres.
- **sort**: mezcla y **comparación**.

Comandos útiles II

- **ssh**: secure shell permite la **conexión remota** a otras máquinas.
- **scp**: secure copy **permite** copiar **ficheros** a través de la **red**.
- **sftp**: secure **ftp**.
- **du**: disk **usage**.
- **uniq**: muestra líneas **únicas** de un **fichero**.
- **cut**: extrae **campos** de una **línea**.
- **tr**: traducción de **caracteres**.
- **sed**: editor de flujo.

Redirección, concatenación y tuberías

- Los **descriptores** de fichero **asociados** con los flujos **stdin**, stdout y **stderr** son 0, 1 y 2, respectivamente.
- **stdout** a fichero: *ls -l > ls.txt*
- **stderr** a fichero: *cat ls.txt 2> lserror.txt*
- **stdout** a fichero y stderr a fichero:
  cat lsdata.txt completedata.txt 2 > caterror.txt
- stderr y **stdout** a un único fichero:
  cat ls.txt & > alldataerrors.txt
- Los comandos para la **concatenación** y para las **pipes** son >> y | respectivamente.

Creación de Macros

- Se realiza mediante la creación de Alias.
- Debe editarse .bash profile or .bashrc.
- Su sintaxis es: alias name=command.
- Mediante el comando alias se activan todos los alias.
- Mediante el comando unalias se desactivan.

Mi primer script

- Creo el **archivo** con un **editor**.
- Lo guardo con **extensión .sh**.
- Añado **permisos** de **ejecución** del archivo.
- **También** se puede **ejecutar** con: *bash script.sh*.
- Muy **recomendable** con **depuración**: *bash -xv script.sh*.

Listado 1: Hola Mundo

```
1 #!/bin/bash
2 # Hola.sh
3 echo "Hola, mundo!"
4 exit 0
```

Gestión de Argumentos

Listado 2: Significado de argumentos

```
1  #!/bin/bash
2  #1 ./argumentos.sh 1 hola "la casa" 55
3  echo Nombre: $0
4  echo PID: $$
5  echo Argumentos: $#
6  echo Argumentos: $*
7  echo Argumentos en array: $@
8  echo Tercer argumento: $3
9  shift
10 echo Tercer argumento: $3
11 echo El directorio local es $HOME
12 ls -l > prueba.txt
13 rm prueba.txt
14 echo "La orden rm devuelve $?"
15 rm prueba.txt
16 echo "La orden rm devuelve $?"
17 rm prueba.txt 2> /dev/null
18 echo "La orden rm devuelve $?"
```

Asignación de variables

- No hay tipos de datos.

Listado 3: Asignación de variables

```
1  #!/bin/bash
2  #1 ./asigvariable.sh Juan Moreno
3
4  nombre=$1
5  apellido=$2
6  nombreCompleto="$1 $2"
7  echo "Hola D. $nombre $apellido; D. $nombreCompleto."
8  a=55
9  echo "a vale: $a"
10 exit 0
```

Espresiones relacionales lógicas I

- **Devuelven** *true* or *false*.
- Se **evalúan** con test expresion o [expresion].

Los **operadores lógicos** son:

- ! expr: **NOT**.
- expr a -a expr b: **AND**.
- expr a -o expr b: **OR**.

Espresiones relacionales lógicas II

Los **operadores de archivos** son:

- -f archivo: fichero **regular**
- -d archivo: es un **directorio**
- -r archivo: fichero **existe** y tiene permisos de **lectura**.
- -w archivo: fichero **existe** y tiene permisos de **escritura**.
- -x archivo: fichero **existe** y tiene permisos de **ejecución**.
- -s archivo: fichero **existe** y no es **vacío**.

Espresiones relacionales lógicas III

Los **operadores con enteros** son:

- a -eq b: **equal** to.
- a -ne b: **not** equal to.
- a -lt b: **less** than.
- a -le b: **less** than or **equal** to.
- a -gt b: **greater** than.
- a -ge b: **greater** than or **equal** to.

Espresiones relacionales lógicas IV

Los **operadores para cadenas** son:

- a = b true: igual.
- a != b true: distinto.
- -n string: el string no es vacío.
- -z string: el string es vacío.

Espresiones relacionales lógicas V

Listado 4: Comparación entre cadenas y enteros

```
1  #!/bin/bash
2  #1  ./comparacion.sh
3  #2  bash -xv  ./comparacion.sh
4
5  echo "true se corresponde con 0 y false con 1"
6  x="001"
7  [ $x = "001" ]
8  echo "Igualdad entre $x y 001 es $?"
9  [ $x = "1" ]
10 echo "Igualdad entre $x y 1 es $?"
11 [ $x -eq 001 ]
12 echo "Igualdad entre $x y el entero 001 es $?"
13 [ $x -eq 1 ]
14 echo "Igualdad entre $x y el entero 1 es $?"
15 x=1.1
16 [ $x -eq 1.1 ]
17 echo "Igualdad entre $x y el real 1.1 es $?"
18 exit 0
```

Espresiones relacionales lógicas VI

Listado 5: Operadores de ficheros; if y valores devueltos

```
1  #!/bin/bash
2  #1 ./vacio.sh vacio.sh
3  #2 echo $?
4  #3 ./vacio.sh vaci.sh
5  #4 echo $?
6
7  if [ -s $1 ]
8  then
9    echo "El fichero $1 no está vacío"
10   exit 0
11 else
12   echo "El fichero $1 está vacío"
13   exit 1
14 fi
```

Espresiones relacionales lógicas VII

Listado 6: Operadores de ficheros; if ; and

```
1  #!/bin/bash
2  #1 ./vacioEjecucion.sh vacioEjecucion.sh
3  #2 chmod -x vacioEjecucion.sh
4  #3 bash vacioEjecucion.sh vacioEjecucion.sh
5
6  if [ -s $1 -a -x $1 ]
7  then
8    echo "El fichero $1 no está vacío y tiene permisos de ejecucióm"
9    exit 0
10 else
11   echo "El fichero $1 está vacío o no tiene permisos de ejecución"
12   exit 1
13 fi
```

Operadores aritméticos I

- $ expr 8 + 4
- $ expr 5 - 3
- $ expr 18 / 3
- $ expr 25 % 5
- $ expr 10 * 2
- $ echo ‘expr 5 * 3‘

Operadores aritméticos II

Listado 7: suma; multiplicación y for

```
1  #!/bin/bash
2  #1 ./sumador.sh 1 3 4 5 5
3  # ./sumador.sh (no control argumentos)
4
5  suma=0
6  mult=1
7
8  for i in $@
9  do
10   suma=`expr $suma + $i`
11   mult=`expr $mult \* $i`
12 done
13
14 echo -e "La suma es $suma.  \nLa multiplicación es $mult."
15 exit 0
```

bc I

- Ejecutar *bc* o *bc -q*

```
27/68
???
scale=3
27/68
????
quit
```

bc II

Listado 8: Operaciones con reales

```
1  #!/bin/bash
2  # bc.sh
3
4  a=5
5  b=7
6
7  echo `expr $a + $b`
8  echo `expr $a / $b`
9
10 echo "scale=3;$a/$b" | bc
11
12 exit 0
```

funciones

Listado 9: Uso de funciones

```
1  #!/bin/bash
2  # mifuncion.sh
3  # .\mifuncion.sh 2 3 5
4  # .\mifuncion.sh 1 1 1 1 1
5
6  function miprimerafuncion () {
7  x="$1"
8  return `expr $x \* 2`
9  }
10
11 miprimerafuncion $#
12
13 echo "$?"
```

read

Listado 10: Lectura por teclado

```
1  #!/bin/bash
2  #bcread.sh
3
4  echo "Introduzca el valor de a"
5  read a
6  echo "Introduzca el valor de b"
7  read b
8
9  echo `expr $a + $b`
10 echo `expr $a / $b`
11
12 echo "scale=3;$a/$b" | bc
13
14 exit 0
```

dialog

Listado 11: Cuadros de diálogo

```
1  #!/bin/bash
2  # dialog.sh
3
4  if dialog --title "Nacionalidad"
5  --backtitle "Datos para inscripción"
6  --yesno "¿Tiene nacionalidad española?" 0 0
7  then
8
9    echo "si"
10
11 else
12
13   echo "no"
14
15 fi
16
17 exit 0
```

sleep

Listado 12: Pausa en la ejecución

```
1  #!/bin/bash
2  # sleep.sh
3
4  for i in `df`
5  do
6    echo $i
7    if [ $i = "Montado" ]; then
8      sleep 2
9    fi
10 done
11
12 exit 0
```

Cut I

- Permite el procesamiento de la información por líneas y la recuperación de la misma.

Listado 13: Usos básicos de cut

```
1  #!/bin/bash
2  # ./contarLineas.sh tests.txt
3
4  echo "El fichero \"$1\" contiene `wc $1 | cut -c3-5` líneas."
5
6  echo "El fichero \"$1\" contiene `wc $1 | cut -d" " -f3` líneas."
7
8  exit 0
```

Cut II

Listado 14: cut y case

```
1  #!/bin/bash
2  # ./usocut.sh
3
4  case $1 in
5
6  1) echo `echo "abcdefghijklmnopqrstuvwxyz" | cut -c1,3,5`;;
7  2) echo `echo "abcdefghijklmnopqrstuvwxyz" | cut -c1-3,5`;;
8  3) echo `echo "abcdefghijklmnopqrstuvwxyz" | cut -c-10,15`;;
9  4) echo `echo "abcdefghijklmnopqrstuvwxyz" | cut -c5-`;;
10 5) echo `echo "abcdefghijklmnopqrstuvwxyz" | cut -d "d" -f 1`;;
11 6) echo `echo "abcdefghijklmnopqrstuvwxyz" | cut -d "d" -f 2`;;
12 *) echo "No ha introducido una opción válida"
13
14 esac
15
16 exit 0
```

Grep

```
grep -F "$(␣printf␣'hola\ndog'␣)" texto.txt

grep -f patron.txt texto.txt

grep -o a texto.txt

grep -v "^[[:space:]]*$" texto.txt (grep -c "^$" filename)

grep a\{3,4\} texto.txt

egrep a\{3,4\} texto.txt

egrep a+ texto.txt

grep lusarski texto.txt tests.txt

grep -r hola ../../..
```

Sort

```
ls -li

ls -li | sort

sort -k1.3,1.3 texto.txt

sort -k1.2,1.3 tests.txt > tests2.txt

ls -li | sort -t' ' -k9.1,9.1

ls -li | sort -t' ' -k1.8,1.8

ls -li| tr -s ' ' |sort -t ' ' -k10.1,10.1
```

find

```
find /var -name *.log -print0

find /home -name *.pdf  -atime -1

find /var -size +122k

find /var -empty

find /home -nouser -ls
```

tr

```
cat texto.txt | tr a-z A-Z

df | tr -s "␣"

cat texto.txt | tr -sc A-Za-z '\n'

cat texto.txt | tr -s "\n*"

cat tests.txt | tr -c "[a-zA-Z0-9]" "_"

cat texto.txt | tr -d aeiou
```

uniq

```
cat regions.fas | grep -v \> | sort | wc -l

cat regions.fas | grep -v \> | sort | uniq | wc -l

cat regions.fas | grep -v \> | sort | uniq -c

cat regions.fas | grep -v \> | sort | uniq -c | sort -n
| tail -5

cat texto.txt | tr -sc A-Za-z '\012' | sort | uniq -c |
sort -n | tail -n 1
```

sed

- sed -e /s/viejaER/nuevaER”
- sed -e /s/viejaER/nuevaER/g”

```
echo Pantalla | sed -e"s/a$/b/"

echo Pantalla | sed -e"s/a/b/"

echo Pantalla | sed -e"s/a/b/g"

echo fichero.txt | sed -e"s/\.txt/\.txt\.old/"

sed -f cambiosgenesis.sed genesis.txt
```

awk

```
awk '{print $1}' tests.txt

df | awk '{print $2}'

df | awk '{print $5 " , " $1}'

awk '/ God /' genesis.txt

awk '/ God / {print $1}' genesis.txt

awk '/, God / {print $1}' genesis.txt
```

Índice

Objetivos

- Conocer las **similitudes** y **diferencias** con la programación **BASH**.
- Adquirir las **habilidades** necesarias para el manejo de **expresiones**, variables y **estructuras** de **control** en PS.
- **Conocer** el concepto de **cmdlet** y su asociación al concepto de **objeto**.
- Manejar los **cmdlets** básicos de **filtrado**, **recuperación** de información y **manipulación**.
- Implementar **scripts** que **integren** todo lo anterior.

Ejecución de Comandos Linux en PS I

- cat
- dir
- mount
- rm
- cd
- echo
- move
- rmdir
- chdir
- erase
- popd

Ejecución de Comandos Linux en PS II

- sleep
- clear
- h
- ps
- sort
- cls
- history
- pushd
- tee
- copy
- kill
- pwd

Ejecución de Comandos Linux en PS III

- type
- del
- lp
- r
- write
- diff
- ls
- ren

Expresiones I

Listado 15: Creación de expresiones

```
PS> "Hola␣Mundo"

PS> "Hola" + "␣Mundo"

PS> "@" * 65

PS> 1..50

PS> 2 + 3 * 4 / 5

PS> (2+3)*4/5
```

Expresiones II

Listado 16: Asignación

```
PS> var = 2

PS> $var = 2

PS> echo $var

PS> $var

PS> $date = get-date

PS> $vector=1,2,3,4

PS> Svector

PS> "Pos␣0:␣" + $vector[0]
```

Expresiones III

Listado 17: Asignación y manejo básico de vectores

```
PS> $procesos = get-process

PS> $procesos[0]

PS> $procesos[0,2]

PS> $procesos[2,3]

PS> foreach ($i in $procesos) {echo "hola"; $i}
```

CmdLets I

- Windows PowerShell está **basado** Microsoft **.**NET Framework y **acepta** y devuelve **objetos** de .NET Framework. **No** devuelve **texto** como los comandos shell **clásicos**.
- Un **cmdlet** es una combinación de **verbo** y **nombre** separado por un **guión** que comprende un **comando** y un **objeto** sobre el que actúa el comando (**Get-Help**).

CmdLets II

- Los objetos **devueltos** tienen **propiedades** que se muestran como pares **nombre-valor**.
- Se pueden usar **individualmente** o pueden ser unidos para la **realización** de tareas más **complejas**.
- En la instalación **inicial** ya se cuenta con más de **cien cmdlets**.

CmdLets III

Los **cmdlets** según el **verbo** utilizado tienen una **tarea** específica:

- **get**: obtener datos.
- **set**: establecer o modificar datos.
- **format**: aplicar formato a los datos.
- **out**: enviar la salida a un destino.

CmdLets IV

Los **básicos** son los **siguientes**:

- **get-help**: obtener información de ayuda.
- **get-command**: nos muestra por pantalla la lista de comandos.
- **get-service**: obtener información sobre los servicios y su estado.
- **get-process**: obtener información sobre los procesos.
- **get-date**: el shell nos dará información sobre la fecha y hora.
- **get-psdrive**: podemos conocer las unidades del sistema.

CmdLets V

Existen **cuatro** modos de **uso** del cmdlet "out":

- **out-host**: dirige los datos de salida a la pantalla.
- **out-printer**: dirige los datos de salida a la impresora.
- **out-null**: para descartar datos innecesarios producto de una ejecución.
- **out-file**: para almacenar en un fichero los datos de salida.

CmdLets VI

Listado 18: Ejemplos de uso de cmdlets

```
PS> get-command *.exe

PS> get-service | format-list

PS> get-command -name -get-location -syntax

PS> notepad; get-process notepad

PS > stop-process -id ???? -confirm

PS> notepad; get-process notepad

PS > stop-process -id ???? -verbose

PS> notepad; get-process notepad

PS > stop-process -id ????
```

CmdLets VII

Listado 19: Opción whatif

```
PS> notepad; get-process notepad

PS > stop-process -id ???? -whatif

PS> stop-computer -whatif

PS> get-process | stop-process

PS> get-process | stop-process -whatif
```

CmdLets VIII

Listado 20: Creación de Alias

```
PS> get-command calc.exe

PS> set-alias cal c:\windows\system32\calc.exe

PS> Get-Alias

PS> Get-Alias ca*

PS> remove-item alias:cal

PS> new-alias mydir get-childitem

PS> mydir

PS> remove-item alias:mydir
```

Filtrado de Información I

Listado 21: Cmdlets para filtrado de información

```
PS> get-help Sort-Object

PS> get-help sort-object -online

PS > get-help where-object -online

PS> get-help foreach-object -online

PS> get-help Select-Object -online

PS> get-help Group-Object -online
```

Filtrado de Información II

Listado 22: Ordenamiento y Selección

```
PS> Get-Process | Sort-Object pm -desc |
Select-Object  -first 10
```

- El primero **obtiene** todos los **procesos** en **ejecución**.
- El comando **Sort-Object** recibe los **objetos** y los **ordena** según la "propiedad de **memoria física**" (pm) de manera **descendente**.
- En tercer lugar, el comando **Select-Object**, muestra los diez **primeros**.

Filtrado de Información III

Listado 23: Información sobre salida de cmdlets

```
PS> Get-Process | get-member

PS> Get-Childitem | get-member

PS> Get-Process | foreach-object { $_.CPU -gt 1} | get-member

PS> $a=5; $a | get-member

PS> $a= get-member -InputObject 1; $a

PS> $a= get-member -InputObject 1,3,5; $a
```

Filtrado de Información IV

Listado 24: Ordenamiento

```
PS> Get-Process | Sort-Object Handles

PS> Get-Process | Sort-Object VirtualMemorySize

PS> Get-Process | Sort-Object CPU -Descending
```

Filtrado de Información V

Listado 25: Selección de objetos

```
PS> Get-Process | Where-Object { $_.PriorityClass -eq "Normal"}

PS> Get-Process | Where-Object { $_.PriorityClass -ne "Normal"}

PS> Get-Process | Where-Object { $_.CPU -gt 15}

PS> Get-childitem | where-Object {$_.length -gt 1000000}

PS> Get-childitem | where-Object {$_.extension -eq ".dll"}
```

Filtrado de Información VI

Listado 26: Selección de campos

```
PS> Get-Psdrive | foreach-Object { $_.name }

PS> Get-Psdrive | foreach-Object { $_.used }

PS> Get-Psdrive | foreach-Object { $_.free -gt 1}
```

Habilitación de Ejecución de Scripts

- Ejecutar **Powershell** como **Administrador**.

Listado 27: Cambiar la política de ejecución

```
PS> Get-ExecutionPolicy

PS> Set-ExecutionPolicy RemoteSigned

PS> Get-ExecutionPolicy
```

Primeros Scripts I

Listado 28: Argumentos

```
1  #argumentos.ps1
2
3  Write-Host "Soy el script llamado" $MyInvocation.InvocationName `
4    "y tengo"  $args.count  "argumentos"  `
5    "el primero es " $args[0]
6
7  $lectura= Read-Host "Introduce un valor"
8  write-host $lectura "es el valor introducido"
```

Primeros Scripts II

Listado 29: Tratamiento de otras entradas

```
1  #Inputs.ps1
2
3  #Entrada tubería
4  $Entrada="$input"
5  write-host "Ha llegado" $Entrada
6
7  #Entrada Fichero
8  $Entrada=Get-Content Inputs.ps1
9  Write-Host @Entrada -separator "`n"
10
11 #Entrada Salidas Comandos
12
13 $name=(Get-WmiObject Win32_operatingSystem)
14 $version=(Get-WmiObject Win32_operatingSystem).version
15 write-host "Ejecución de" $name.name versión $version `
16     "y directorio actual" $(Get-Location)
```

Primeros Scripts III

Listado 30: If elseif y ficheros como argumentos

```
1  #if.ps1
2
3  if ($args.Length -ne 2){
4
5      write-host "uso correcto: " `
6      $MyInvocation.InvocationName "<argumento> <argumento2>"
7
8  } elseif ((Test-Path $args[0]) -or  (Test-Path $args[1])){
9
10     write-host "Uno o los dos argumentos son ficheros"
11
12     }
13 else
14     {
15     write-host "Ninguno de los dos argumentos es un fichero"
16     }
```

Primeros Scripts IV

Listado 31: Identificadores a argumentos y Expresiones Reg.

```
1  #argumentos2.ps1
2  #.\argumentos2.ps1 -cadena a -numero 1
3  #.\argumentos2.ps1 a 1
4
5  param ($numero, $cadena)
6
7  write-host "1er argumento" $numero "2nd Argumento" $cadena
8
9  if (($numero -match "^[0-9]+$") -and
10        ($cadena -cmatch "^[aeiou]{8}$")){
11
12      write-host "Los argumentos son correctos"
13
14      }
15 else {
16      write-host "Los argumentos deben ser un número
17 y una cadena formada por ocho vocales minúsculas"
18      }
```

Primeros Scripts V

Listado 32: for y foeach

```
1  #for.ps1
2
3  for($i=1;$i-le9;$i=$i+2){
4      write-host $i
5  }
6
7  foreach ($i in 100..125) {
8      write-host $i
9  }
```

Primeros Scripts VI

Listado 33: foreach y directorios

```
1  #for.ps1 (segunda parte)
2
3  foreach ($i in Get-ChildItem){
4
5      if ($i.PSIsContainer){
6
7          write-host $i "es un directorio y su fecha de creación"
8          $i.CreationTime
9          }
10
11     else {
12
13         write-host $i.Name "es un fichero y su tamaño es"
14         $i.length
15         }
16 }
```

Primeros Scripts VII

Listado 34: While

```
1  #while.ps1
2
3  $i=0
4
5  while ($i -le 3){
6
7
8      write-host $i
9
10         $i++
11
12         }
```

Primeros Scripts VIII

Listado 35: While con comandos almacenados en variables y ER

```
1  #while.ps1(segunda parte)
2
3  $file=get-childitem
4
5  $i=0
6
7  while ($i -le $file.Length){
8
9      if ($file[$i].Name -match "^[aeiou]"){
10
11                 #cuidado con mayusculas
12                 write-host $file[$i].Name "empieza por vocal"
13             }
14
15     $i++
16
17     }
```

Primeros Scripts IX

Listado 36: Funciones

```
1  #funciones.ps1
2
3  #declaracion
4  function suma($a, $b){
5     write-host "La suma de $a y $b es" $a+$b
6     write-host "La suma de $a y $b es" ($a+$b)
7     }
8
9  suma 2 3.8
10 suma $args[0] $args[1]
```

Bibliografía

- Gómez López, Julio. **Administración de sistemas GNU-Linux**. Starbook, 2010.
- Colobran Huguet, Miquel. **Administración de sistemas operativos en red**. UOC, 2008.
- Gómez López, Julio. **Administración de sistemas operativos Windows y Linux: un enfoque práctico**. RA-MA, 2006.
- Hjelmas, Erik. **PowerShell tutorial**. `http://www.ansatt.hig.no/erikh/tutorial-powershell/powershell-notes.pdf`. Último acceso: 16 de abril de 2014.

Tema 2. Automatización

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Tema 3. Modelos Estructurales de Sistemas Operativos

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Índice

Introducción I

- La ingeniería de un **sistema** tan **grande** y **complejo** como un SSOO moderno debe garantizar un **funcionamiento apropiado**.
- Además, el sistema debe **modificarse** y **actualizarse** con **facilidad**.
- Una solución es la de **dividir** el sistema en **componentes** más **pequeños** (no **monolítico**).
- Cada **módulo** debe estar perfectamente **definido** (entradas, salidas y funciones).

Objetivos I

- Llegar a **conocer** como ha sido la **evolución** de los Sistemas Operativos.
- Repasar las **técnicas** que han pasado desde los SSOO **experimentales** a SSOO **comerciales**.
- Estudiar cómo los diferentes **componentes** de un SSOO se **interconectan** para componer un **kernel**.
- Establecer las **diferencias** entre la **programación** en modo **kernel** y **usuario**.
- Conocer como se **gestionan** SSOOs con una gran **complejidad** y **tamaño**.

Objetivos II

- **Discriminar** entre el concepto de **módulo** y **capa** dentro del contexto de los SSOO.
- Estudiar la **organización** de un SSOO concreto como es **Android**.
- **Conocer** los diferentes sistemas que conforman las **familias Windows** y las **distribuciones Linux**.
- Conocer **características** a nivel general de las **Máquinas Virtuales**.

Índice

Los **componentes principales** de un SSOO son:

- Gestión de **procesos**.
- Gestión de la **memoria** principal.
- Gestión de **archivos**.
- Gestión del Sistema de **Entrada/Salida**.
- Gestión de **almacenamiento secundario**.
- Sistema de **seguridad** y **protección**.
- Sistema de **interpretación** de **órdenes**.
- El **kernel**.

Gestión de Procesos I

Recursos:

- Tiempo de **CPU**.
- Memoria.
- **Archivos**.
- Dispositivos de **E/S**.
- Puede necesitar también datos de **inicialización**.
- Al finalizar debe **devolver** sus recursos al **SSOO**.

Gestión de Procesos II

Actividades del **SSOO**;

- **Crear** y eliminar procesos.
- Suspender y **reanudar** procesos.
- Proveer mecanismos para la **sincronización** de procesos.
- Proveer mecanismos para la **comunicación** de procesos.
- Proveer mecanismos para manejar **bloqueos mútuos**.

Gestión de la Memoria Principal

- La CPU puede **direccionar** y **acceder** directamente a ella.
- Las **instrucciones** deben estar en la MP para que la CPU pueda **ejecutarlas** (**carga**).

Actividades del **SSOO**:

- Saber qué **partes** de la **memoria** se están **usando**, cuáles están libres y **quién** las está usando.
- Decidir qué **procesos** cargar en la **memoria**.
- **Asignar** y **liberar** espacio de memoria.

Gestión de Archivos

- Un archivo es un conjunto de **información relacionada** generalmente programas y datos.
- Cuando varios **usuarios** tienen **acceso** a los archivos, se debe **controlar quién** y de **qué** modo accede a ellos.

Actividades del **SSOO**:

- **Crear** y **eliminar** archivos.
- Crear y eliminar **directorios**.
- Proveer las **primitivas** para manejo de **archivos** y **directorios**.
- Establecer la **correspondencia** archivo-almacenamiento secundario.
- **Guardar** los archivos en almacenamientos **no volátiles**.

Gestión del sistema de E/S

- Se trata de un conjunto de **dispositivos** muy variados y **complejos** de **programar**.

Actividades del **SSOO**:

- Proporcionar una **interfaz uniforme** para el **acceso** a los dispositivos.
- Proporcionar **manejadores** para los **dispositivos** concretos.
- Tratar **automáticamente** los **errores** más típicos.
- Para los dispositivos de **almacenamiento**, usar **cachés**.
- Para los **discos**, planificar de forma **óptima** las **peticiones**.

Gestión del almacenamiento secundario

- **Almacenamiento no volátil**: casi todos los **programas** (compiladores, ensambladores, rutinas de ordenación, editores y formateadores) se **guardan** en **disco** hasta que se **cargan** en **memoria**.

Actividades del **SSOO**:

- Administración del **espacio libre**.
- Asignación del **almacenamiento**.
- **Planificación** del disco. Existen diversos **algoritmos** para planificar el servicio de las **solicitudes** de entrada y salida del disco.

Sistema de seguridad y protección

- Es preciso **proteger** a un **proceso** de los demás, ya que el sistema de computación admite **múltiples usuarios** y la ejecución **concurrente** de procesos.
- **Protección**: Mecanismo para **controlar** el **acceso** de programas, procesos o usuarios a los **recursos** definidos por un **sistema** de computador.

Sistema de interpretación de órdenes

- **Interfaz** entre **usuario** y **SSOO**. Para que un usuario pueda **dialogar** directamente con el SSOO.
- **Ejemplos**: **JCL** en sistemas por lotes, **COMMAND**.COM en MS-DOS, **powershell** y **cmd** en WINDOWS, **shell** de UNIX.

El SSOO proporciona una **interfaz** de usuario básica **para**:

- **Cargar** programas.
- Abortar **programas**.
- Introducir **datos** a los programas.
- Trabajar con **archivos**.
- Trabajar con **redes**.

El kernel

El **núcleo** es la **parte fundamental** del SSOO:

- Es el responsable de **facilitar** a los distintos programas **acceso** seguro al **hardware** de la computadora.
- Se encarga de **decidir** qué programa podrá hacer **uso** de un **dispositivo** de hardware y durante cuánto tiempo.
- **Proporciona** una **interfaz** limpia y uniforme al hardware subyacente mediante **abstracciones** del **hardware**.

Funcionalidades del núcleo

Sus **funciones básicas** son:

- **Garantizar** la carga y la **ejecución** de los **procesos**.
- **Garantizar** las **entradas**/**salidas** de manera correcta.
- Proponer un **interfaz** entre el **espacio núcleo** y los **programas** del espacio de **usuario**.

Otras funciones no necesariamente proporcionadas por el kernel son:

- Sistemas de **archivos**.
- Funciones de **redes**.
- Servicios.

Programación en modo kernel

Dos tipos de **situaciones** equivalentes en **complejidad**:

- Programar **aplicaciones** de **usuario** de forma tal de que el usuario pueda sacar **provecho** de su sistema **computacional**.
- Programar **funcionalidades** en el **kernel** de forma tal que se pueda **ayudar** a los **desarrolladores** de aplicaciones a entregar mejores **herramientas** a los usuarios.

Criterios de utilización I

- La programación en modo **kernel** debiera utilizarse en **situaciones** donde el **rendimiento** es **crítico** y en aquellas situaciones donde se requiere acceso a **hardware** que **no** es **accesible** a las aplicaciones de **usuario**.
- La programación en modo kernel **no** sirve para **todas** las **situaciones**.

Criterios de utilización II

- En **general** la programación en modo Kernel debe **restringirse** al mínimo **posible**.
- Hay **funcionalidades** que pueden ser mejor **proporcionadas** por aplicaciones de **usuario** y no hay porque involucrar al **kernel**.
- Muchas veces la **mejor solución** involucra un poco de **programación** en modo **kernel** y una aplicación de **usuario** que se encarga del resto.

Diferencias con programación de usuario I

Alcance:

- Una **aplicación** sólo **afecta** a quien la utiliza y sólo de manera **temporal**.
- El **kernel** afecta a **todos** los usuarios y **aplicaciones**.

Consumo de memoria:

- El kernel reside en **memoria física**. Cualquier **byte** usado en modo kernel es un byte **menos** para el resto del **sistema**.
- Las **aplicaciones** que residen en memoria **virtual**, por consiguiente pueden ser intercambiadas (**swap**) o bien **eliminadas** de la memoria.

Diferencias con programación de usuario II

Aislamiento:

- Un **error** en una **aplicación** de usuario por lo general queda **aislado** a sólo esa aplicación.
- Un **error** en el **kernel** afecta a **todo** el **sistema** y por lo general hace el sistema **irrecuperable**.

Ventajas

- Puede otorgar **mejor rendimiento**: aunque si no se **programa correctamente** el desempeño puede **empeorar** de manera **trágica**.
- Acceso a **recursos no disponibles** para **aplicaciones** de usuario: **llamadas** no **disponibles** a través de la **biblioteca** C o recursos de **hardware**.

Desventajas

- Los **recursos** disponibles en modo Kernel son por lo general muy **limitados**: **ausencia** de **printf** o no existe **aritmética** de punto **flotante** (manejo directo registros FPU).
- El **flujo** del tiempo **no** es **lineal**: una aplicación comienza y termina de manera lineal y el uso de **locks** y llamadas al **scheduler** hacen que la **predicción** del flujo del tiempo sea muy **difícil**.
- **No** existe **retroalimentación** ni **interacción** posible por parte del **usuario**: lo máximo que se puede lograr es la **modificación** de algunos parámetros **generales** de **funcionamiento**.

Cuando programar en modo kernel I

Cuando se requiera **acceso** directo al **hardware**:

- Es decir cuando se requiera un **acceso** más **complejo** que lo que ofrecen **iopl** (changes the I/O privilege level of the calling process allowing disable interrupts) e **ioperm** (permiso a un rango de direcciones a las cuales se tiene que acceder).
- Cuando se requiere trabajar **directamente** sobre las **interrupciones**.
- Cuando se requiere un **manejo** muy **fino** de la **memoria** o del **scheduler**.

Cuando programar en modo kernel II

Para implementar **protocolos** de **comunicación**:

- Es **prudente** implementar sólo los **protocolos** de **comunicación** que son **independientes** de un **hardware** específico.
- En algunos casos es necesario **dividir** la **implementación** en dos partes, de forma tal de permitir que el **protocolo** en si sea **tratado** por el **kernel** y la **interacción** con el hardware y el usuario se realice a través de una **aplicación** de **usuario**.

Cuando programar en modo kernel III

Cuando se requiera **implementar** políticas **especiales**:

- Por ejemplo, cuando se requiera **cambiar** el **scheduler** por otro (**RTAI Linux)** (Tiempo real).
- Cuando se **requieren** políticas especiales de **manejo** de **memoria**.
- Para programar **políticas** de **control** de **recursos**. Security-Enhanced Linux (**SELinux**) es un módulo de **seguridad** para el kernel Linux que proporciona el **mecanismo** para soportar **políticas** de seguridad para el control de **acceso**.

Cuando programar en modo kernel IV

Cuando las **circunstancias** lo hagan necesario:

- Es decir, hay casos en los que **no** es **fácil clasificar** el por qué de la **necesidad** de programar en modo **kernel**.
- La **experiencia** indica que se presentan **situaciones** donde es **preferible** programar en modo **kernel**.

Cuando NO programar en modo kernel I

Aplicaciones **simples** realizables a nivel de **usuario**:

- Muchas veces es **posible** solucionar los **problemas** a través de una **aplicación** de **usuario**.
- El agregar **funcionalidades** sin un **fundamento** de peso por lo general trae más **problemas** que **soluciones**.

Cuando NO programar en modo kernel II

Cuando se requiera **interacción** por parte del **usuario**:

- El **kernel** por definición no incorpora **mecanismos** para **interactuar** con el **usuario**.
- Aún cuando es posible **implementar** algunos **mecanismos** de interacción, el **rendimiento** decae en forma casi **exponencial**.
- En estos casos es mejor **separar** el **problema** en dos **partes** e **implementar** la parte **interactiva** en una aplicación de usuario.

Cuando NO programar en modo kernel III

Cuando no haya un **beneficio** claro para todo el **sistema**:

- El **beneficio** debe ser claro para el **sistema** en cuestión, no para todos los **sistemas** existentes ni todos los **casos** posibles.
- Hay que tener cuidado de no implementar **funcionalidad** ya **existente** que podría ser mejor **aprovechada** a través de una **biblioteca** de sistema.

Índice

Evolución de los SO

- La **progresión** de los SSOO ha sido **gradual** a lo largo de los años.
- Pero en la **última década**, han aparecido **nuevos sistemas** operativos y **nuevas versiones** de sistemas operativos ya existentes que han provocado un **cambio** brusco en el **campo** de los SSOO.
- Los nuevos SSOO han de **responder** a las **evoluciones** en el **Hardware**, nuevos tipos de **aplicaciones** y los nuevos retos en **seguridad**.

Evolución de los SO

- No sólo se **requieren** cambios y **mejoras** en las **arquitecturas** existentes sino más bien **nuevas** políticas de **organización** de los SSOO.
- Una amplia **variedad** de **técnicas** y elementos han sido **introducidos** tanto en SSOO **experimentales** como **comerciales**, pero la mayor parte del desarrollo se ha realizado en las siguientes categorías:
  - Arquitecturas basadas en **micronúcleos**.
  - Programación **multihilo**.
  - Multiprocesamiento simétrico (**SMP**).
  - Sistemas operativos **distribuidos**.
  - Diseño **orientado** a **objetos**.

MICRONUCLEOS

Núcleos monolíticos I

- **Software** y **estructuras** de datos se encuentra en un **único** módulo lógico.
- **No** existen **interfaces** entre los componentes del SO.
- **No** existe **ocultación** de **información** entre los distintos procedimientos y estructuras de datos.
- Normalmente, un núcleo monolítico está implementado como un **único proceso** con todos sus **elementos** compartiendo el **mismo** espacio de **direcciones**.

Núcleos monolíticos II

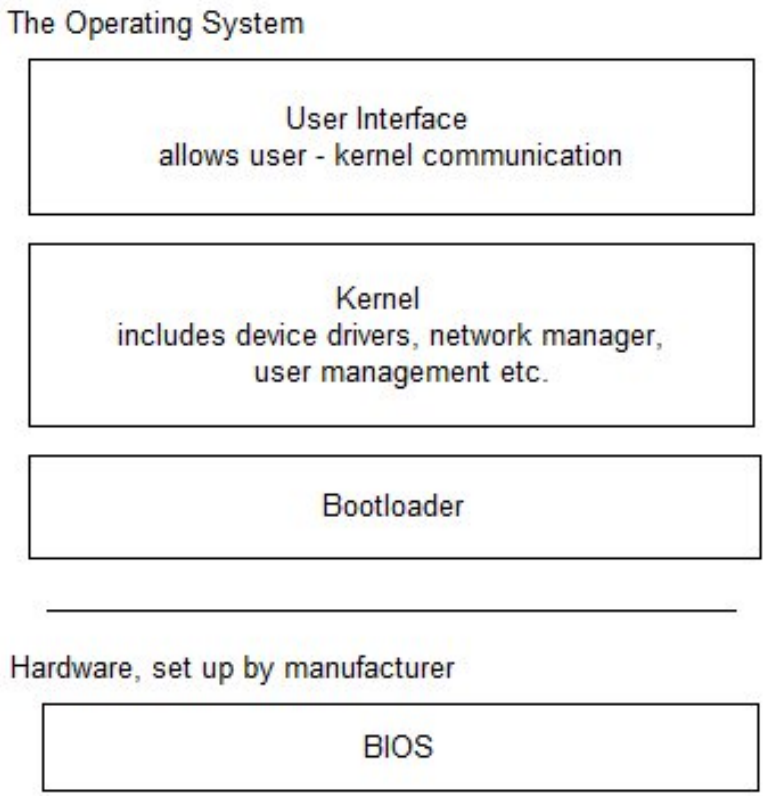


Figura: Estructura de un SSOO monolítico

Núcleos monolíticos III

Ventajas:

- En su **diseño no** hay que tener en cuenta tanto las **interrelaciones** entre **partes**.
- Son muy **eficientes** si están **correctamente implementados**.
- **Estructura** muy **sencilla** para sistemas operativos **pequeños**.

Inconvenientes:

- Difícil **mantenimiento**.
- Difíciles de **comprender**.

Arquitecturas micronúcleo I

- Una arquitectura **micronúcleo** asigna solo un conjunto **reducido** de **funciones** esenciales al núcleo, incluyendo los espacios de direcciones, comunicación entre procesos (IPC) o planificación básica.
- El **resto** de servicios del SSOO son proporcionados por **procesos**, llamados **servidores**, que se ejecutan en modo **usuario** y son tratados como cualquier otra **aplicación**.

Arquitecturas micronúcleo II

- Esta aproximación **separa** el desarrollo del **núcleo** y de los **servidores**.
- Este tipo de diseño **simplifica** la **implementación** y proporciona mayor **flexibilidad**, y es una configuración adecuada para sistemas **distribuidos**. En esencia, el micronúcleo **interactúa** del mismo modo, con un servidor **local** que con un servidor **remoto**.

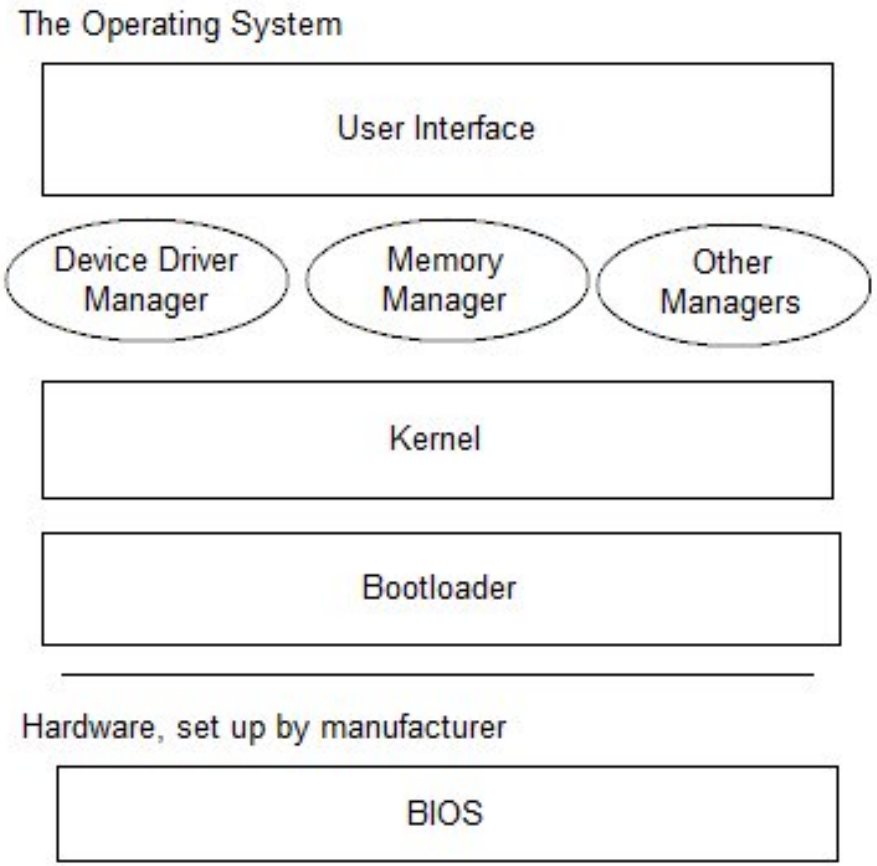


Figura: Estructura de un SSOO basado en micronúcleo

PROGRAMACIÓN MULTIHILO

Hebras

- La programacion **multihilo** es una **técnica** que permite que un proceso, que ejecuta una aplicación, se pueda dividir en **hilos** que pueden correr de manera **concurrente**.
- Es útil en **aplicaciones** que llevan a cabo una serie de **tareas independientes** que no necesitan ser "serializadas". Ej: Servidor BBDD.
- Trabajar con múltiples **hebras** supone **menor sobrecarga** para el procesador que trabajar con múltiples **procesos**.
- Los hilos son también **útiles** como forma de **estructurar procesos** que son parte del **núcleo** del SSOO.

MULTIPROCESAMIENTO SIMÉTRICO (SMP)

SMP

- La **arquitectura** de los computadores ha **cambiado** al disponer de más de un procesador.
- Para conseguir una mayor **eficiencia** y **fiabilidad**, se utiliza una técnica llamada **multiprocesamiento simétrico**, que no sólo se refiere a una **arquitectura** hardware sino también al **comportamiento** del SSOO que permite **explotar** dicha arquitectura.

SMP

Se **define** multiprocesador simétrico al computador que:

- Tiene más de un **procesador**.
- Los procesadores **comparten** memoria, sistema de E/S y están **interconectados** por un bus.
- Todos los procesadores pueden realizar las **mismas operaciones** (*simetría*).

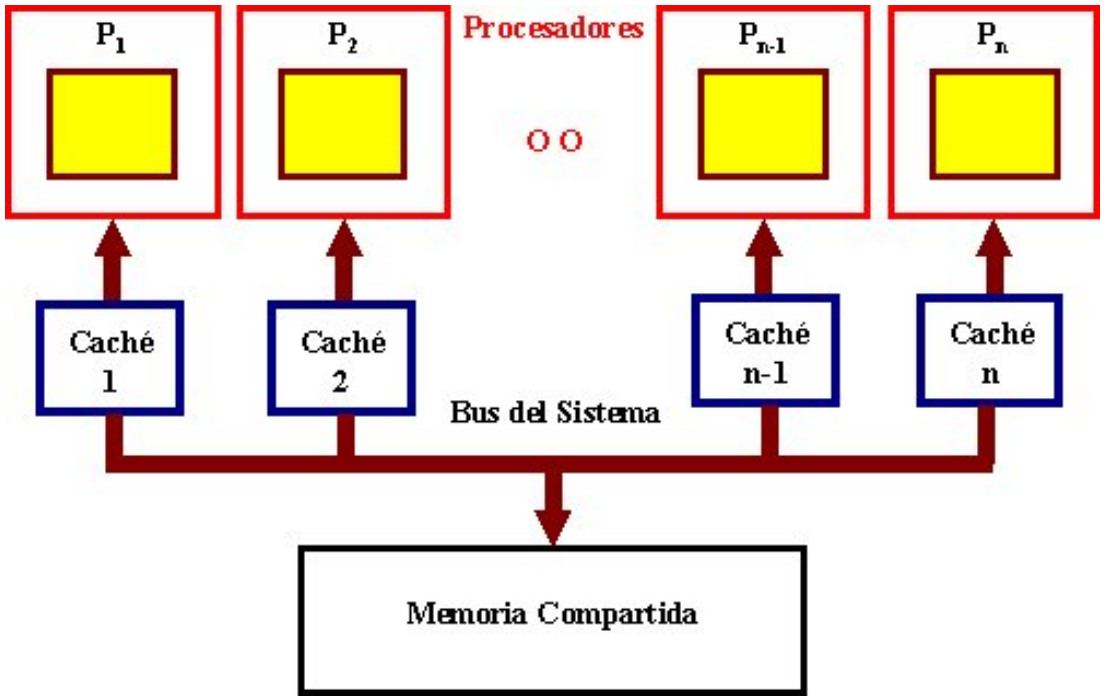


Figura: Estructura de un SMP

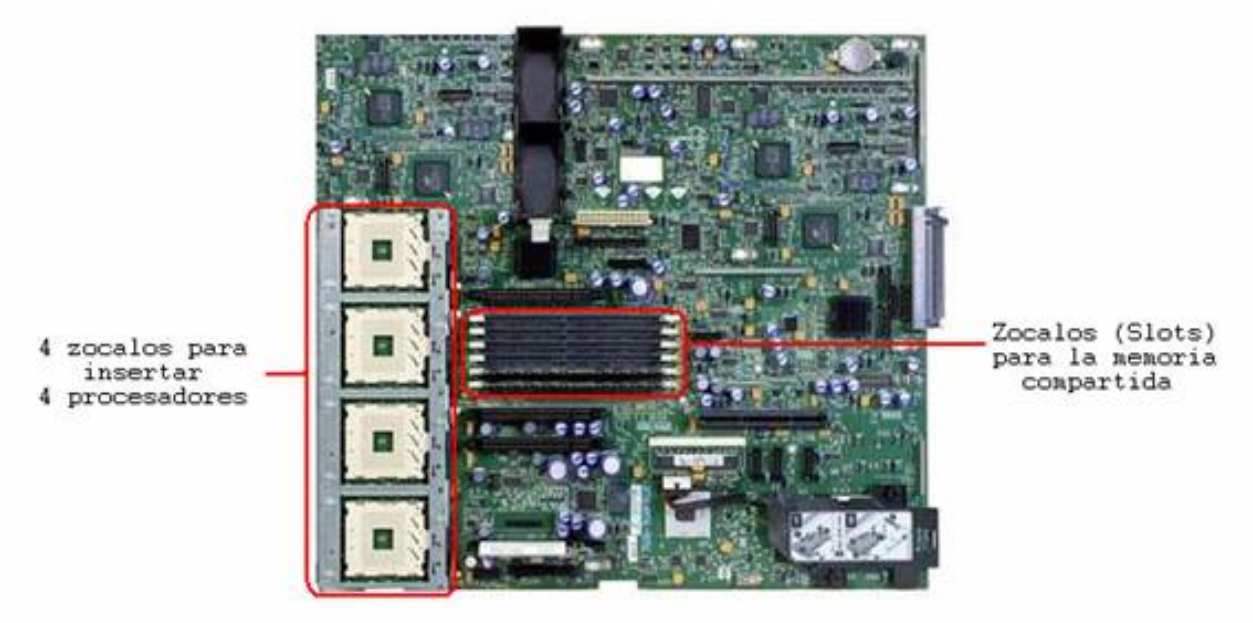


Figura: Circuitería de un SMP (multiple chip)

SMP

El SSOO de un SMP planifica procesos o **hilos** en **todos** los **procesadores**. Las principales **ventajas** respecto a arquitecturas monoprocesador son:

- **Rendimiento**: Se puede **organizar** la **carga** de trabajo para ser ejecutada en paralelo.
- **Disponibilidad**: Si **falla** un **procesador**, debido a la simetría el **sistema** sigue **funcionando**.
- **Crecimiento**: Se puede **mejorar** el rendimiento del sistema **añadiendo** nuevos **procesadores**.
- **Escalabilidad**: El mercado ofrece productos de diferente **precio** y **rendimiento** basándose únicamente en el número de procesadores disponibles.

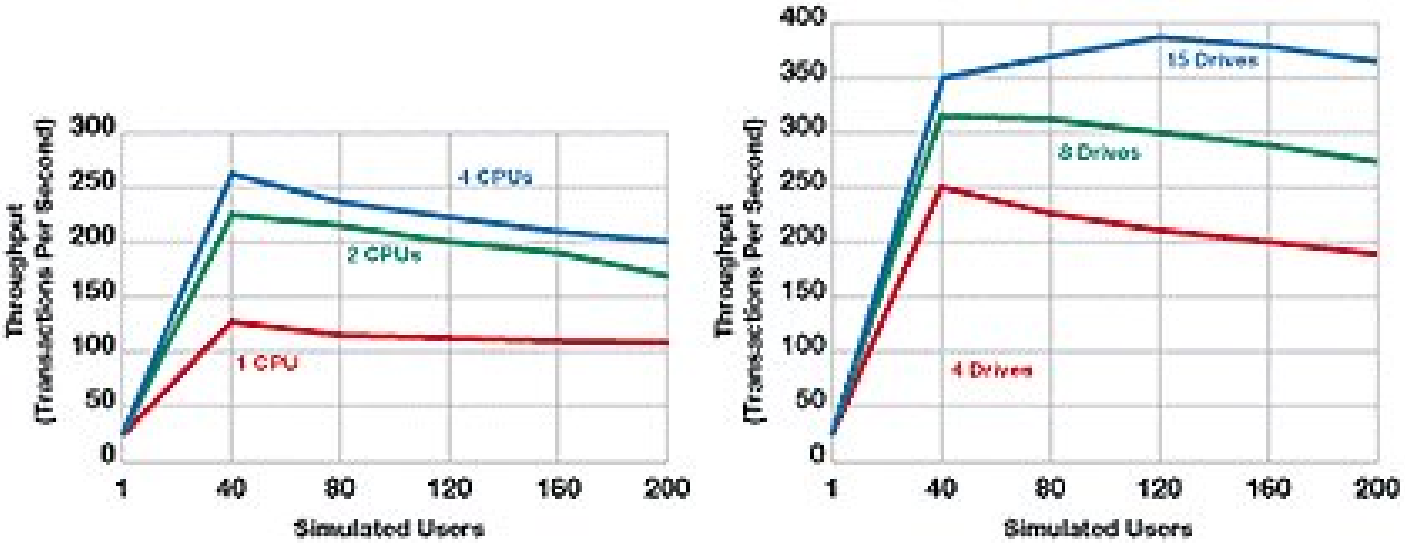


Figura: Rendimiento en un SMP

SSOO DISTRIBUIDOS

SSOO Distribuidos

- El SSOO hace que la existencia de **múltiples procesadores** en un computador sea **transparente** para el usuario.
- Un problema distinto es que esa **transparencia** sea posible cuando los procesadores están **distribuidos** en un clúster de computadores, cada uno con su propia memoria, sistema de E/S, memoria secundaria, etc.
- Con un SSOO distribuido se **trabaja** con espacios de memoria **únicos** tanto de memoria **principal** como de memoria **secundaria**.
- Además, existen otras **facilidades** como los sistemas de **archivos** distribuidos.

DISEÑO ORIENTADO A OBJETOS

SSOO y POO

- Otra **innovación** importante en el **diseño** de SSOO es la utilización de tecnologías **orientadas** a **objetos**.
- Este proceso aporta **disciplina** al proceso de añadir **extensiones** modulares a un **kernel** de tamaño pequeño.
- Una estructura basada en objetos permite a los programadores **personalizar** un **SSOO** sin perturbar la **integridad** del sistema.
- Este tipo de diseño también **facilita** el desarrollo de **herramientas distribuidas** y **sistemas** operativos **distribuidos**.

Índice

- Cada vez se han **requerido** más **funcionalidades** a los Sistemas Operativos.
- Además el **hardware** cada vez es más **rápido** y tiene una mayor **capacidad**.
- Por todo esto, la **complejidad** y **tamaño** de los Sistemas Operativos ha **crecido** de manera considerable.

Líneas en Kernel Linux

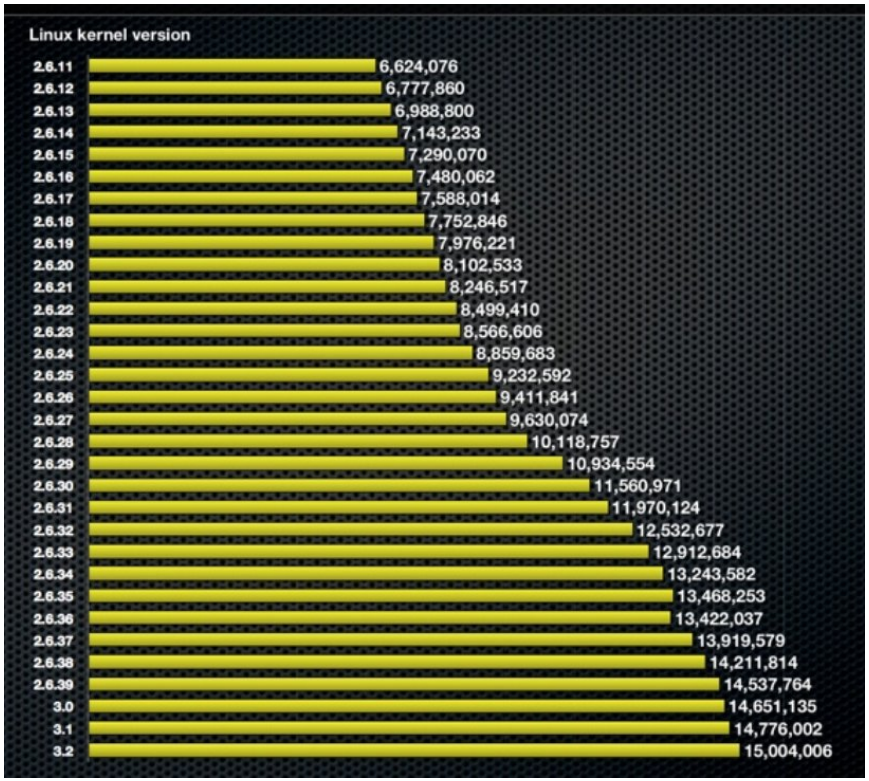


Figura: Evolución del número de líneas de Código

Problemas comunes

El **tamaño** de un SSOO completo y la **complejidad** de su propio funcionamiento, conduce a que existan cuatro **problemas comunes**:

1. La **distribución** de un SSOO es muy **lenta**. Esto ocurre tanto en los nuevos como en las actualizaciones de los antiguos.
2. **Constantemente** aparecen **bugs** que deben ser corregidos con una serie de **actualizaciones**.
3. El **rendimiento** no es el esperado.
4. Es **imposible** el desarrollo de un SSOO complejo que **no** sea **vulnerable** a una conjunto de ataques, incluyendo virus, gusanos y accesos no autorizados.

SISTEMAS MODULARES

Modularidad I

- Para tener capacidad de **manejar** la **complejidad** de un SSOO e intentar **solucionar** en parte los **problemas** planteados con anterioridad, existen grandes cantidades de estudios e **investigaciones** sobre como debe ser la **estructura** de un SSOO.
- Algunas cosas obvias, es que el **sistema** debe ser **modular**.
- La **funcionalidad** se encuentra **dividida** entre **componentes** lógicos independientes con interfaces bien **definidas**.

Modularidad II

- Un SSOO **modular** se **implementa** mediante diversos módulos de **programa** y/o **procesos**.
- **Ventajas**: fáciles de **mantener** y modificar debido a la **encapsulación** funcional y a la **abstracción** de datos.
- **Inconveniente**: potencial **degradación** del **rendimiento**

ESTRUCTURA JERÁRQUICA EN CAPAS.

ARQUITECTURA ANDROID.

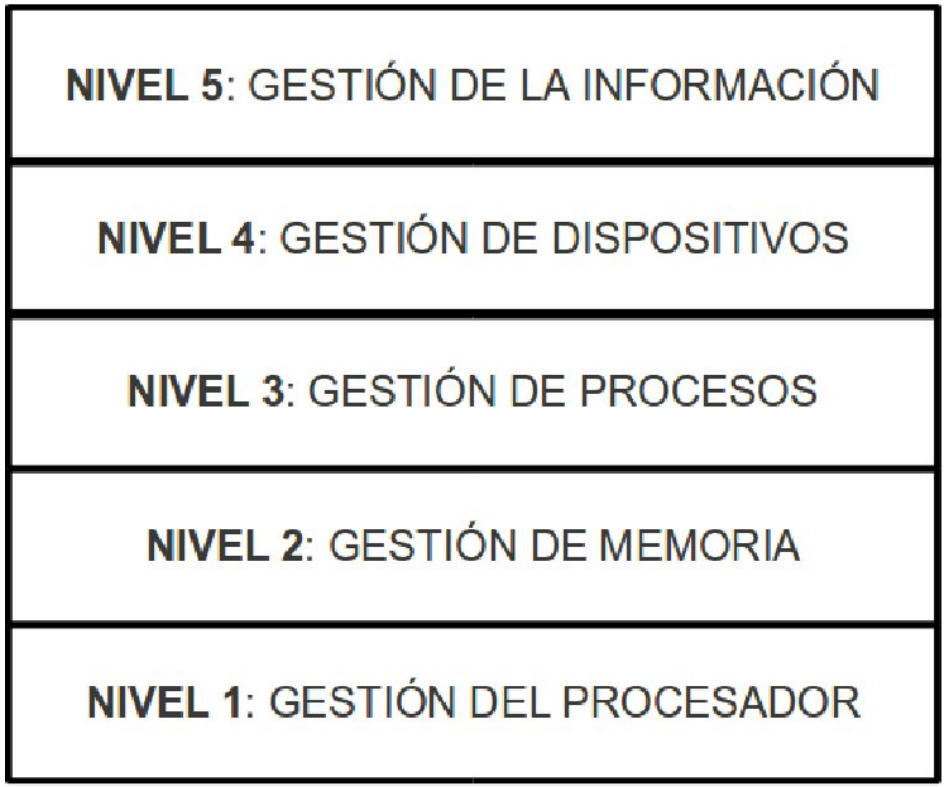
- La **estructuración** en **módulos** anteriormente vista, **no** es **suficiente** para resolver ciertos problemas. Además, se requiere una organización de más alto nivel.
- Esto se consigue con el concepto de **estructura jerárquica** en **capas**.
- El **diseño** por **capas** lleva asociada una **jerarquización** que permite disminuir la complejidad observable de un sistema
- **Concepto** de **capa**: conjunto de **funciones claramente definidas** hacia un objetivo común.

Criterios para establecer la división en capas

- La **división** por capas se realiza **en función** del nivel de **abstracción** y los **tiempos** de ejecución.
- El **sistema** se puede ver como un **conjunto** de **niveles**.
- Cada **nivel** tiene implementado un **subconjunto** de **funciones** del SSOO.
- Un **nivel confía** en el nivel **inferior** la realización de primitivas de más **bajo nivel** y el **ocultamiento** de los detalles acerca de dichas funciones.
- Un **nivel** da un conjunto de **servicios** al nivel **superior**.

Niveles clásicos

Figura: Niveles clásicos en un SSOO



Ventajas del empleo de capas

- Las **estructuras** internas y **algoritmos** de una capa **no** son **visibles** a las demás.
- El **sistema** puede **evolucionar** fácilmente. Lo único que hay que **mantener** son las **interfaces**.
- Pueden existir **realizaciones** alternativas llevadas a cabo por diferentes **grupos** de **trabajo**.
- Algunas **capas** pueden ser **transparentes** si sus servicios no son necesarios.
- Cada **capa** se **codifica** y se **prueba** de modo **independiente**. Esto es muy importante en el desarrollo del software.

Estructura de Android

Figura: Arquitectura de Android

Condicionantes de los dispositivos móviles

- **Poca** duración de la **batería**.
- **Recursos** muy **escasos** (memoria, cpu).
- **Limitaciones** visuales (**pantallas** con poca resolución).
- **Compartición** y priorización de **recursos**.

Figura: Arquitectura de Android detallada

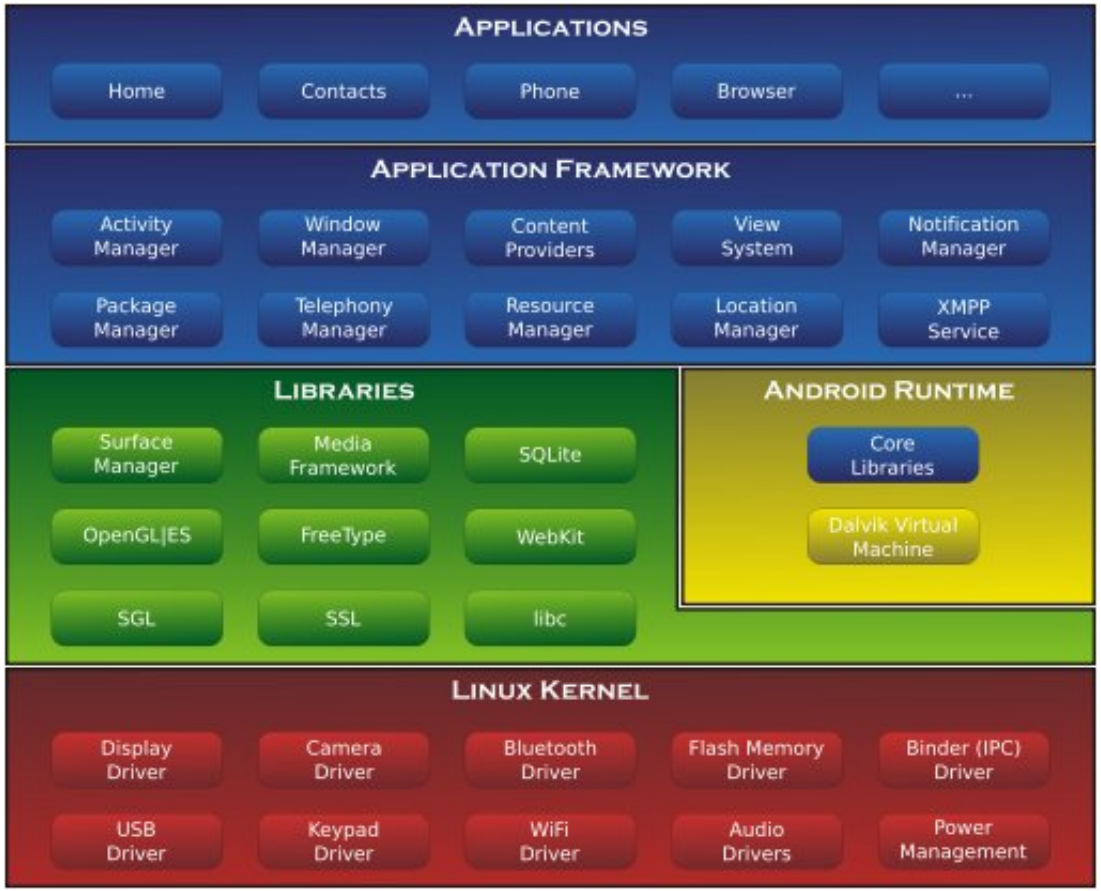
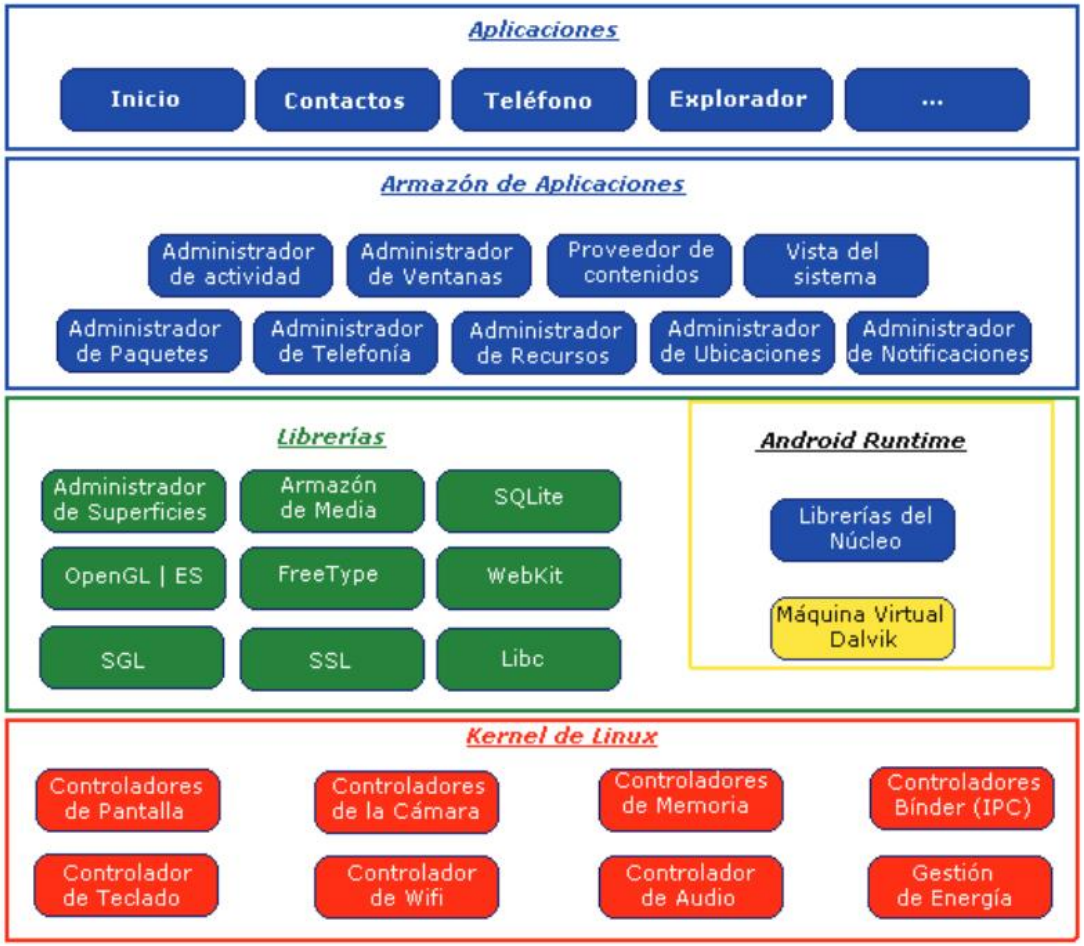


Figura: Arquitectura de Android detallada



Kernel de Android I

- Es un **kernel** de **Linux** (2.6) **adaptado** a las características hardware de los dispositivos **móviles**.
- El **núcleo** actúa como capa de **abstracción** entre el **hardware** y otras capas. Así, un desarrollador no puede **acceder** directamente al **hardware** sino mediante la utilización de **librerías** disponibles en capas superiores.
- Esto tiene la **ventaja** de que **no** hay que conocer **características** detalladas del **hardware**. Por ej. características de la cámara.

Kernel de Android II

- El **kernel** también gestiona la **memoria**, **energía** y otros recursos del teléfono. Obviamente, también los **procesos** y elementos de **comunicación** a nivel de SSOO.

Android aprovecha:

- La **seguridad**.
- Gestión de **memoria**.
- Gestión de **procesos**.
- **Red** y modelo de **drivers**.

Librerías de Android I

- Las **bibliotecas nativas** o **librerías**, están escritas en **C** o **C++** y están compiladas para el hardware de cada teléfono.
- Las **desarrolla** el **fabricante** y las instala en el dispositivo antes de su venta.
- El **objetivo** de las librerías es proporcionar **funcionalidad** a las **aplicaciones** (Application Framework) para tareas que se repiten con **frecuencia** evitando codificarlas de manera repetida.

Librerías de Android II

- **Surface Manager**: gestión del acceso a la pantalla.
- **Media Framework**: reproducción de imágenes, audio y video.
- **SQLite**: pequeña base de datos relacional.
- **WebKit**: navegador (Browser) optimizado.
- **SGL**: gráficos 2D.
- **Open GL — ES**: librerías 3D.
- **FreeType**: renderización de vectores e imágenes (bitmap).

Entorno de ejecución de Android I

- Las aplicaciones se **codifican** en **Java** y son **compiladas** en un formato específico para que la **máquina** virtual las **ejecute**.
- La ventaja de esto es que las **aplicaciones** se **compilan** una **única** vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán **ejecutarse** en **cualquier** dispositivo **Android** que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Entorno de ejecución de Android II

- **Dalvik** es una **variación** de la máquina virtual de **Java**, por lo que no es compatible con el bytecode Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la **extensión .dex** que es específico para **Dalvik**.
- Las **librerías** del **núcleo** son aquellas clases que son usadas **principalmente** por el JDK.

Entorno de ejecución de Android III

- En **Android**, debido a temas de **licencia**, es un subconjunto de **Apache Harmony**.
- Apache Harmony "fue" una **implementación** open source y **gratuita** de **Java** desarrollada por la **Apache Software Foundation**.
- Este proyecto se inició en **2005** y finalizó en **2011**.
- **Dalvik** utiliza un **subconjunto** de Harmony para el núcleo de su librería de clases. Por ejemplo, las clases **J2ME**, **AWT** y **Swing no** son **soportadas** en Android.

Framework de aplicaciones en Android

- Está compuesta por todas las **clases** y **servicios** que **utilizan** directamente las **aplicaciones** para realizar sus funciones. La mayoría de los **componentes** de esta capa son **librerías Java** que acceden a los recursos de las capas anteriores a través de la máquina virtual **Dalvik**.
- Estas **son**: **Activity** Manager, Windows **Manager**, **Content** Provider, Views, **Notification** Manager, Package Manager, **Telephony** Manager, Resource Manager, **Location** Manager, Sensor Manager, Cámara, **Multimedia**.

Aplicaciones en Android I

- En la **última capa** se incluyen todas las **aplicaciones** del dispositivo, tanto las que tienen interfaz de usuario como las que no, las **nativas** (C o C++) y las **administradas** (Java), las **preinstaladas** y las **instaladas** por el usuario.
- En esta capa encontramos también la **aplicación principal** del sistema: Inicio (**Home**) o lanzador (**launcher**). Esta permite ejecutar otras aplicaciones mostrándolas en diferentes escritorios donde se pueden ubicar accesos directos a aplicaciones o incluso **widgets**.

Aplicaciones en Android II

- Cuando **salimos** de una aplicación y volvemos al **launcher** lo que hace Android es **ejecutar** de nuevo dicho programa
- **Android** proporciona un **entorno** sumamente **poderoso** ya que **nada** dentro de Android es **inaccesible**.

FAMILIA DE SSOO WINDOWS

Propósito Doméstico

- **Windows XP**
- **Windows Vista** (2007). Importantes **cambios** técnicos, **inestabilidad**, sobrecarga de **recursos** de hardware y alta **incompatibilidad** con sus predecesores.
- **Windows 7** (2009). Mayor **compatibilidad**, mejoras en el **rendimiento** (velocidad y recursos), **débil** al ataque por **virus** informáticos.
- **Windows 8** (2012). **Nueva** pantalla **inicio**, **Apps**, **fin** del **menú inicio**, IE10, etc.
- **Windows 10** (2015). Regreso del **Menú Inicio**, optimización **táctil**, **Metro** o **Modern** ejecutadas sobre el escritorio, escritorios **virtuales**, **multitarea** mejorada con **snap**.

Familia Windows Server I

- **Windows Server 2003** (2003) reemplazando a la Windows **2000**, importante número de características nuevas, enfoque en la **seguridad**.
- **Windows Server 2003 R2** (2006)
- **Windows Server 2008** (2008). Mejoras en el **control** del **hardware** y en políticas de **seguridad**.
- **Windows Server 2008 R2** (2009). Último con soporte 32 bits.
- **Windows Server 2011** (2011). Basado en Windows Server 2008 R2.

Familia Windows Server II

- **Windows Server 2012** (2012) y **Windows Server 2012 R2** (2013). Elementos **comunes** con Windows 8 a nivel de **escritorio**, diferentes métodos de organización de aplicaciones (nombre, fecha instalación, más usado y por categoría), más posibilidades en la configuración de doble monitor, IE11, etc.
- **Windows Server 2016** (2016). Desarrollo simultáneo con Windows 10. Security y Cloud-ready.
- No olvidar los **Service Packs** de cada versión.
- Disponibles para **estudiantes** de la UCLM en `www.dreamspark.com`

Soporte

- Fecha de disponibilidad general: **05/03/2006**
- Fecha de fin del soporte técnico principal: **13/07/2010**
- Fecha de fin del soporte técnico extendido: **14/07/2015**
- Fecha de fin de soporte de Service Pack: **14/04/2009** (24 meses después del lanzamiento del siguiente service pack o al final del ciclo de vida de soporte técnico del producto).

DISTRIBUCIONES LINUX

DEBIAN + SLACKWARE + REDHAT

- **Linux Mint**: Distribución basada en **Ubuntu** cuya principal **característica** es su **fácil** utilización.
- **Ubuntu**: basada en Debian, centrada en la **facilidad de uso**. Muy popular (mucho soporte comunitario). El entorno de escritorio por defecto es GNOME. (KDE: Kubuntu). Desktop, Server, Cloud, kylin, Core.
- **Redhat Enterprise**; **pago de licencia**, calidad contenidos y soporte. Enfocada a empresas.
- **Fedora**: patrocinada por **RedHat** y **fácil** de **instalar**.
- **Debian**: proceso de instalación un poco **más complejo**. Muy **estable** antes que actualizada.

- **OpenSuse**:**Fácil de instalar** es una versión libre de la distribución comercial SuSE.
- **SuSE Linux Enterprise**: soporte a los usuarios por parte de la empresa que la distribuye, **Novell**. Enfocada a **empresas**.
- **Slackware**: de las **pioneras**. Problemas de **actualización** que la dejaron un poco en el olvido.
- **Gentoo**: esta distribución es una de las **únicas** que incorporaron un concepto totalmente **nuevo** en Linux. Es una sistema inspirado en **BSD-ports**.

LinuxMint o Ubuntu

LinuxMint o Ubuntu. La Interfaz I

Figura: Interfaz de Linux Mint (Cinnamon)

LinuxMint o Ubuntu. La Interfaz II

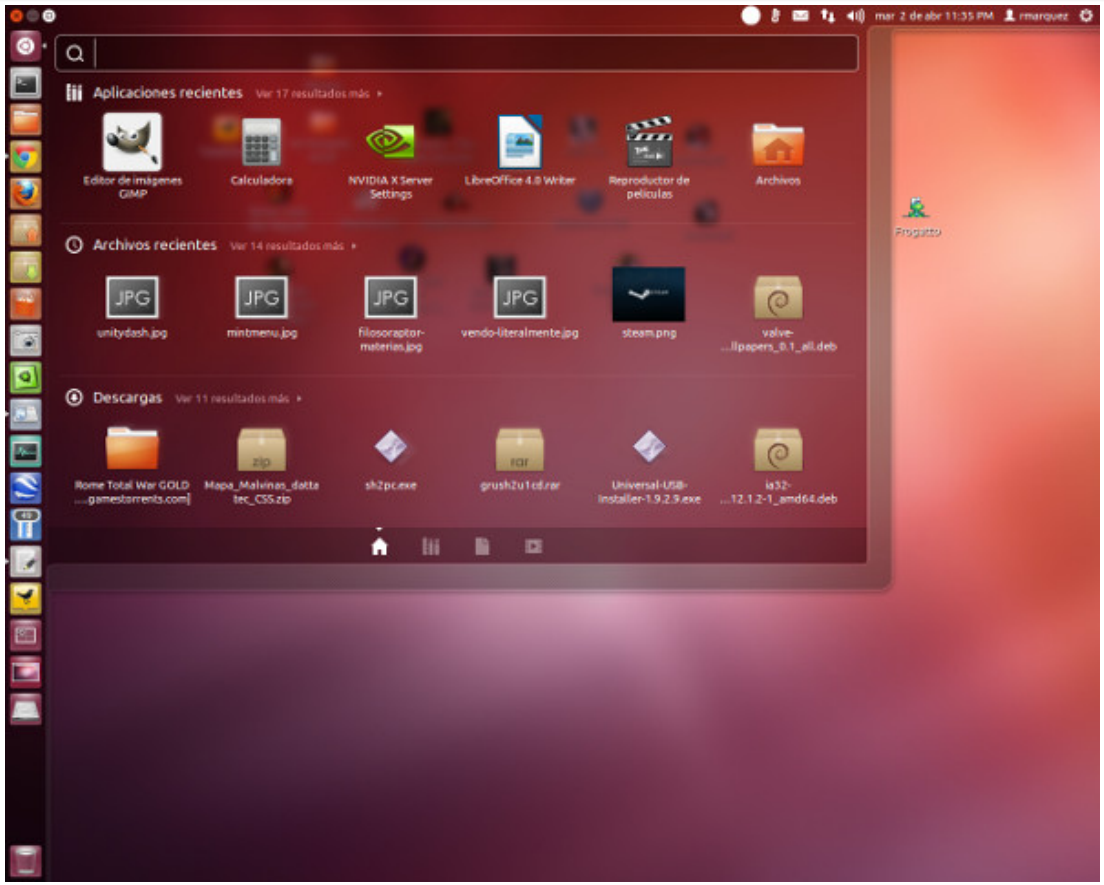


Figura: Interfaz de Ubuntu (Mac OS X)

LinuxMint o Ubuntu I

- Ambas son muy **recomendables** para la **iniciación** en sistemas Linux.
- **Velocidad**: **Ubuntu** muy rápido en equipos de gama **media-alta**. Linux Mint más **rápido** en equipos de gama **baja**.

LinuxMint o Ubuntu II

- **Gestores de aplicaciones**: el **centro de software de Ubuntu** es mucho más completo. El de Linux Mint es mucho más **básico**.
- **Personalización de la interfaz**: En este aspecto Linux **Mint** tiene **ventaja** sobre Ubuntu ya que permite fácilmente cambiar **preferencias** en la interfaz **gráfica**. En **Ubuntu** el usuario tendría que buscar **aplicaciones** de **terceros** para **personalizar** un poco mas la interfaz.

LinuxMint o Ubuntu III

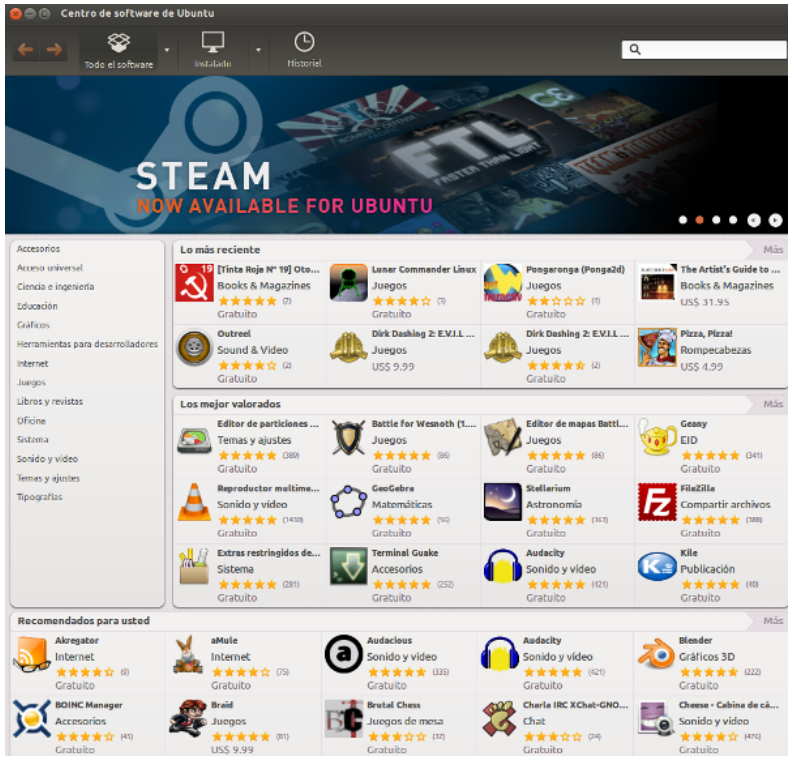


Figura: Centro de Sofware de Ubuntu

LinuxMint o Ubuntu IV

Figura: MintStore

Índice

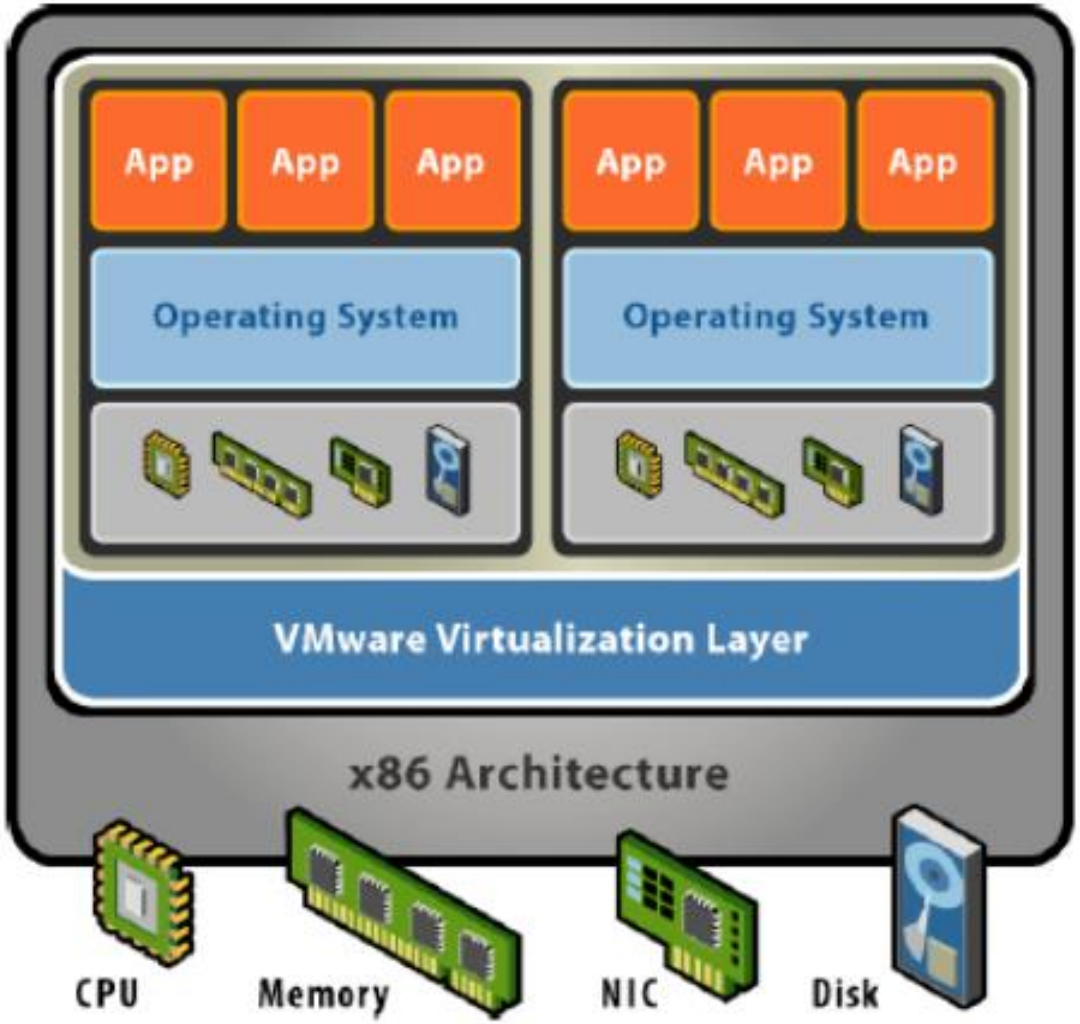
¿Qué es la virtualización? I

- **Técnica** que permite, mediante **software**, convertir una **máquina real** en varias **máquinas independientes** que se ejecutan al mismo tiempo.
- Cada una de estas máquinas independientes recibe el nombre de "**máquina virtual huésped**" (guest), y puede estar **gestionada** por cualquier **sistema operativo** (o casi).
- La **máquina real** sobre la que se ejecutan los huéspedes se denomina "**máquina anfitriona**" (host).

¿Qué es la virtualización? II

- El **software** que hace posible la ejecución **simultánea** de varios **huéspedes** sobre un único **anfitrión** es el "**hipervisor**" (**monitor de máquina virtual (VMM)**).
- Cada huésped "verá" su propia **CPU**, **memoria**, **discos**, etc, independientemente de los **recursos** de que disponga el anfitrión o el **resto** de **huéspedes**.

¿Qué es la virtualización? III



OBJETIVOS Y REQUISITOS

Objetivos de la virtualización I

En centros de datos:

- **Integrar** en una única máquina varios **servidores**. Ahorrando en **adquisición** de equipos, **actualización** de hardware, **costes** de mantenimiento (**refrigeración**, por ejemplo), espacio, etc.
- Se **mantiene** entre **máquinas** el mismo nivel de **aislamiento** que si éstas estuviesen **físicamente** separadas (un **fallo** en una máquina virtual **no afecta** al resto).

Objetivos de la virtualización II

En entornos personales:

- Lograr, con una **única** máquina, **separar** varios **entornos** de **desarrollo** y/o pruebas.
- Aprender a **utilizar** nuevo **software** o nuevas técnicas sin poner en **riesgo** el software **preexistente** en nuestro equipo:
  - Instalar sobre **Windows** una máquina virtual **VMWare** que sea un **Linux CentOS** con Oracle express instalado.
  - Montar y **probar** una **red local** formada por varias **máquinas** virtuales sobre un único **anfitrión**

Objetivos de la virtualización III

- Facilidad de **migración**: cada máquina virtual puede llegar a "**empaquetarse**" en uno o varios **ficheros**; si esos ficheros se **copian** a otro **anfitrión**, la máquina huésped se podrá poner en marcha también en ese **segundo** equipo (con la única **condición** de que el segundo **anfitrión** tenga **instalado** el mismo **hipervisor** que el primero).
- Se consiguen **ejecutar** programas **antiguos** que **no funcionan** bien en **sistemas** operativos **modernos**.

Requisitos I

Capacidad de procesamiento suficiente:

- **Varias** unidades de **procesamiento**, al menos dos **cores** (intel coreo duo, por ejemplo).
- El **hipervisor** se pueden **configurar** para establecer el **número** de **cores** disponibles para cada huésped.

Requisitos II

RAM suficiente:

- En entornos **personales**, a partir de **2 GB**.
- En **servidores**, a partir de **4 GB**.
- En todo caso, **prever** las **necesidades** de **RAM** de cada sistema operativo **huésped** y dotar a la máquina **anfitriona** de la cantidad de RAM adecuada.
- En **SSOO** de **32 bits no** se pueden **exceder** las **4 GB** de RAM.

Requisitos III

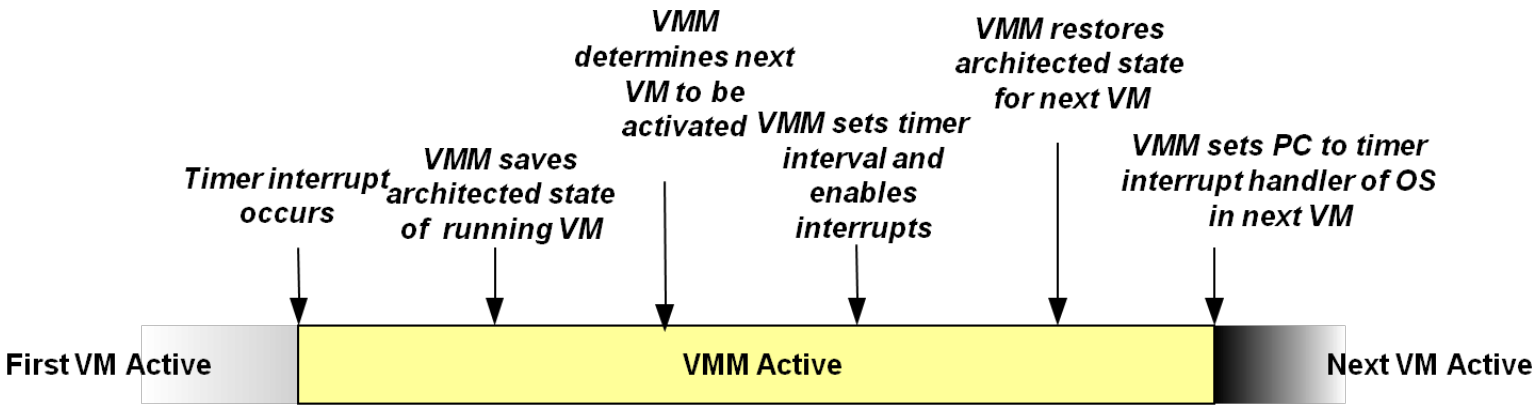
Espacio en disco suficiente:

- Tener en cuenta las **exigencias** de espacio en disco de cada **sistema operativo huésped** y de las **aplicaciones** que en él instalemos.
- Utilizar **discos** duros **rápidos** (SCSI, SATA...).
- Posibles **alternativas** de **uso** del espacio en **disco**:
  - Una **partición** o disco separado para cada **sistema huésped** (XEN y KVM lo permiten). Es lo preferible.
  - Uno o varios **ficheros** para cada sistema **huésped** (lo permiten todos los **hipervisores**).

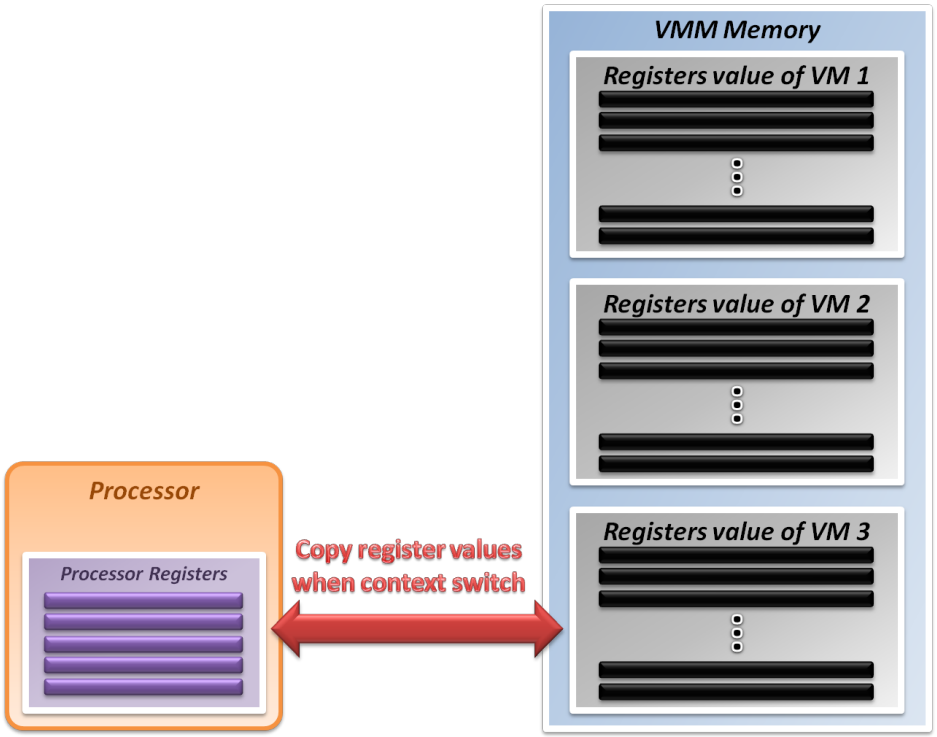
Cambios de contexto entre VMs I

- Ocurre la **interrupción** (reloj) en la máquina **virtual**.
- Hay un cambio de **contexto** a la VMM (**hipervisor**).
- VMM graba el estado de **ejecución** de la **VM**.
- VM **determina** cuál es la próxima VM en **ejecutarse**.
- VMM **determina** el tiempo de **ejecución**.
- VMM restaura el **estado** de la siguiente VM.
- VMM configura el PC para **interrumpirse** en el tiempo **determinado**.
- Se **activa** la **siguiente**.

Cambios de contexto entre VMs II



Cambios de contexto entre VMs III



INSTRUCCIONES

CLASIFICACIÓN DE CPUs

- No se evaluará la materia posterior a este punto.
- Como complemento se recomienda estudiar los requerimientos de virtualización de Popek y Goldberg.

Tipos de instrucciones I

En el campo de la **arquitectura** los diseñadores de **CPUs**, separan las instrucciones en dos **categorías** diferentes:

- Instrucciones **privilegiadas**: estas instrucciones son "**atrapadas**" por el SSOO si se está en modo **usuario** y no lo son si la máquina está en modo **núcleo**. Ej: instrucción para **modificar** el page table base register (**PTBR**).
- **Instrucciones no privilegiadas**: cualquier otra instrucción. Ej: interrupciones **software**, operaciones aritméticas **normales**.
- La **decisión** de qué operaciones son **privilegiadas** o no es del **diseñador** de la **CPU**.

Tipos de instrucciones II

En el campo de la **virtualización**, los diseñadores de **hipervisores** separan las instrucciones en dos **categorías**:

- **Instrucciones sensibles**: son aquellas que interactúan con el **Hardware**, y se dividen en:
  - Instrucciones sensibles de **control**: aquellas que intentan cambiar la **configuración** de los **recursos** en el sistema.
  - Instrucciones sensibles de **comportamiento**: aquellas cuyo **comportamiento** o resultado dependen de la **configuración** de los **recursos** (por ejemplo, el contenido de un registro o el modo del **procesador**).

Tipos de instrucciones III

- **Instrucciones no sensibles**: todas las demás. Ej: operación **aritmética**.
- De forma más genérica, en **virtualización** las **instrucciones sensibles** se pueden clasificar en **llamadas al sistema**, las **interrupciones hardware** y las **excepciones**.

Tipos de instrucciones IV

Arquitectura ARMv6 ISA:

- Branch instructions.
- Data-processing instructions.
- Multiply instructions.
- Parallel addition and subtraction instructions.
- Extend instructions.
- Miscellaneous arithmetic instructions.
- Other miscellaneous instructions.
- Status register access instructions.
- Load and store instructions.
- Load and Store Multiple instructions.
- Exception-generating instructions.
- Coprocessor instructions.

CPU Virtualizable

- Todas las instrucciones **sensibles** son **privilegiadas**.
- Para este tipo de CPUs es muy fácil **implementar** el **hipervisor**. El **hipervisor** trabajará en modo **privilegiado** y el SO guest, trabajará en modo no **privilegiado**.
- Cuando el **guest** quiere ejecutar una instrucción **sensible**, ésta es atrapada por el **hipervisor** que está ya ejecutándose en modo **privilegiado**.
- De esta manera, existe una **seguridad** de que el Guest no podrá realizar **cambios** en la configuración **hardware** de manera directa y que todos los recursos **hardware** importantes están bajo el control del **hipervisor**.

CPU No Virtualizable

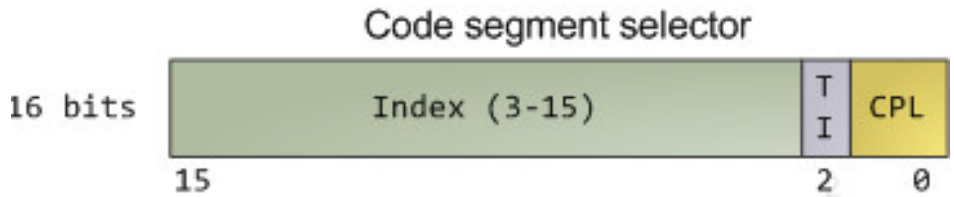
- Se define aquella **instrucción** que es sensible y no **privilegiada** como "**Instrucción Crítica**".
- Para una CPU no **virtualizable**, es complejo de implementar el **hipervisor**. Si está en modo **privilegiado** y el Guest en no privilegiado, si el Guest ejecuta una instrucción crítica, ésta no será atrapada por el hipervisor de manera automática.
- Por tanto, las instrucciones **críticas** pueden ejecutarse en modo **privilegiado** o modo no privilegiado y este trato diferente puede causar **problemas**.
- Como resultado, los diseñadores del **hipervisor**, deben hacer que las instrucciones **críticas** sean atrapadas por el hipervisor, y dejar que el hipervisor emule sus **comportamientos**.

TÉCNICAS

Inexistencia de virtualización I

La arquitectura **x86** admite la **ejecución** de instrucciones en **cuatro niveles** de **privilegio**:

- Estos cuatro niveles vienen **codificados** en los dos bits menos significativos **(Current Privilege Level)** del registro **CS**.



- Normalmente **sólo** se utilizan **dos** de estos **cuatro** niveles: el **nivel 0** (modo privilegiado o **kernel**) y el **nivel 3** (modo **usuario**).

Inexistencia de virtualización II

- Una **instrucción** que se ejecute en modo **privilegiado** puede tener acceso al **hardware** de la máquina (RAM, tabla de páginas, puertos de E/S, etc).
- El **hardware** provoca automáticamente el **cambio** al **modo 0** cuando se produce una **interrupción hardware** o una excepción.
- El **sistema operativo** también puede **cambiar** el nivel de **privilegio**.
- Las **aplicaciones** de usuario se **ejecutan** en el **nivel 3** (modo usuario): sus instrucciones **no** pueden **acceder** al **hardware** de la máquina **directamente** (cualquier acceso que **necesiten** a través de **peticiones** al sistema operativo, **llamadas al sistema**).

Inexistencia de virtualización III

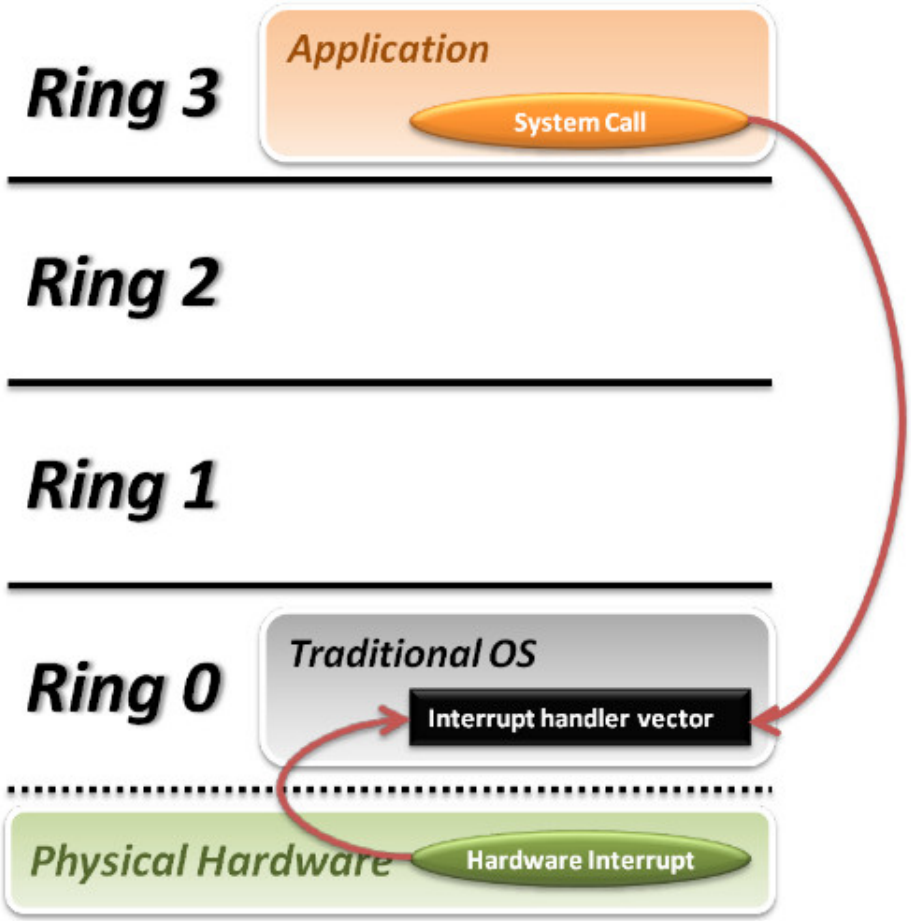
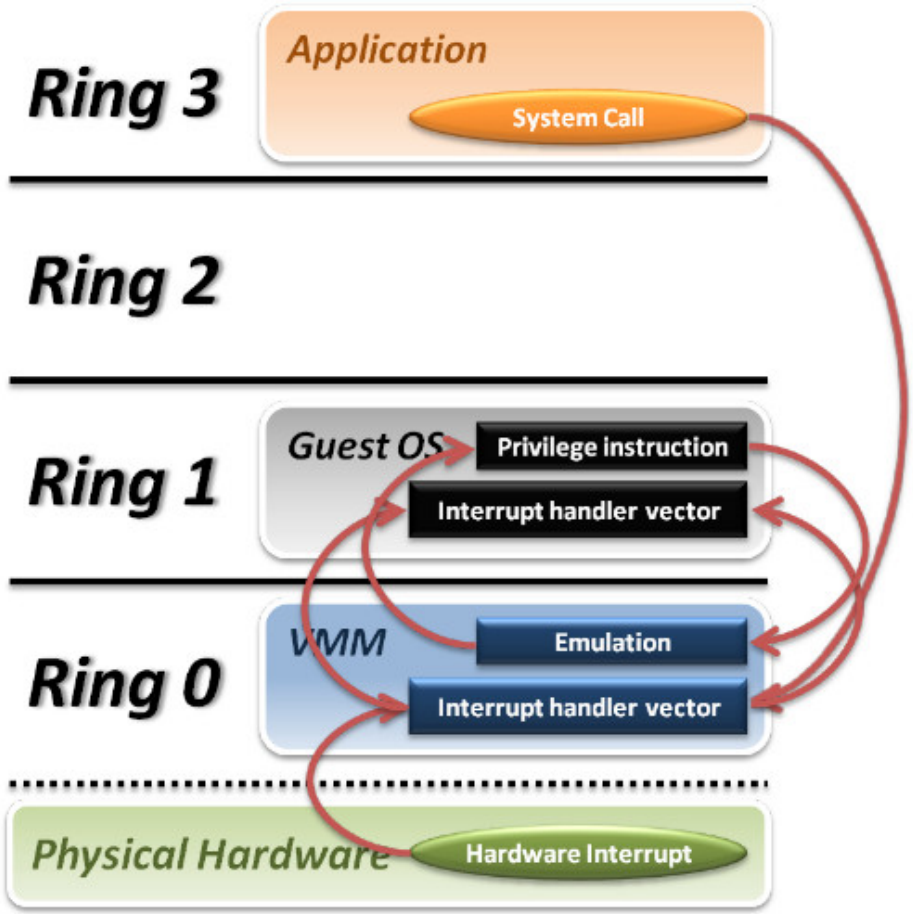


Figura: SO tradicional

Emulación (Trap and Emulate Model) I

- La **emulación** permite ejecutar cualquier **sistema operativo no modificado** que soporte la plataforma emulada.
- La principal desventaja de la **emulación** es el bajo **rendimiento**.
- Ejemplos: los productos de **VMware**, QEmu, Bochs o **Parallels**.

Emulación (Trap and Emulate Model) II



Emulación (Trap and Emulate Model) III

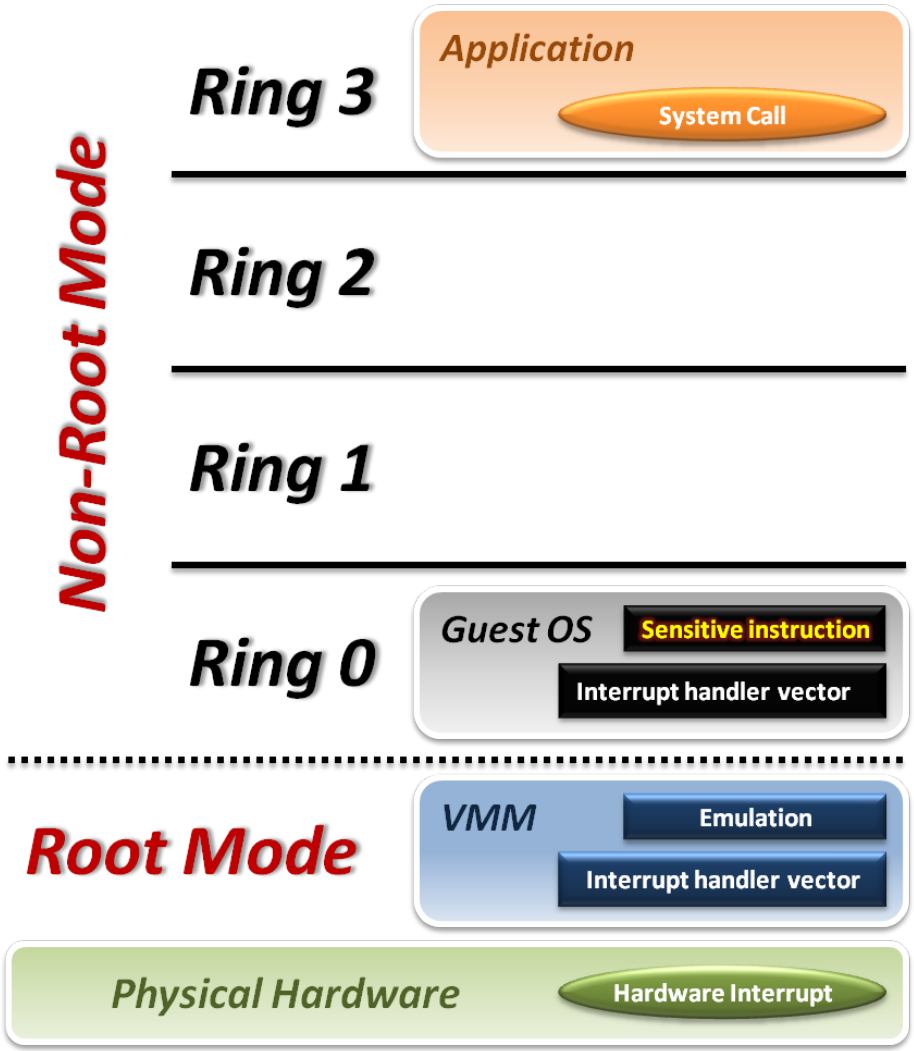
Interacción VMM y huésped:

- **Llamadas al sistema**: CPU se dirige al vector de **interrupciones** de la máquina **virtual** y ésta lo devuelve al vector del SO **huésped**.
- **Interrupciones hardware**: CPU se dirige al vector de **interrupciones** de la máquina **virtual**, de nuevo se salta al vector del **huésped** y se ejecuta una instrucción **privilegiada**.
- **Instrucciones privilegiadas**: En el SO huésped se **captura** para la **emulación** de la instrucción. Después de la emulación, la máquina virtual **retorna** al sistema **huésped**.

Virtualización de hardware no virtualizable I

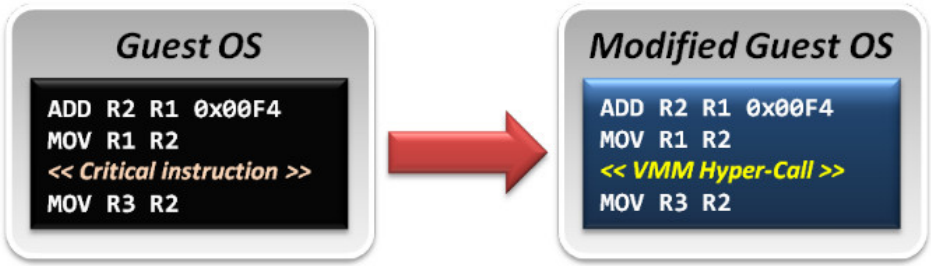
1. **Paravirtualización**: Modificar el Guest para que no ejecute las instrucciones críticas. Se substituyen por llamadas al hipervisor y así se "atrapan".
2. Virtualización **Completa**, **Binary translation**: escribe en el guest aquellas instrucciones críticas como instrucciones no sensibles (usuario).
3. Virtualización **Completa**, **Hardware assistance**: Procesadores **diseñados** para soportar la **virtualización**. El procesador se **comportará** de manera diferente para al **guest** y para el **host**, así diseño del VMM será más **eficiente** y simple.

Virtualización de hardware no virtualizable II



Paravirtualización I

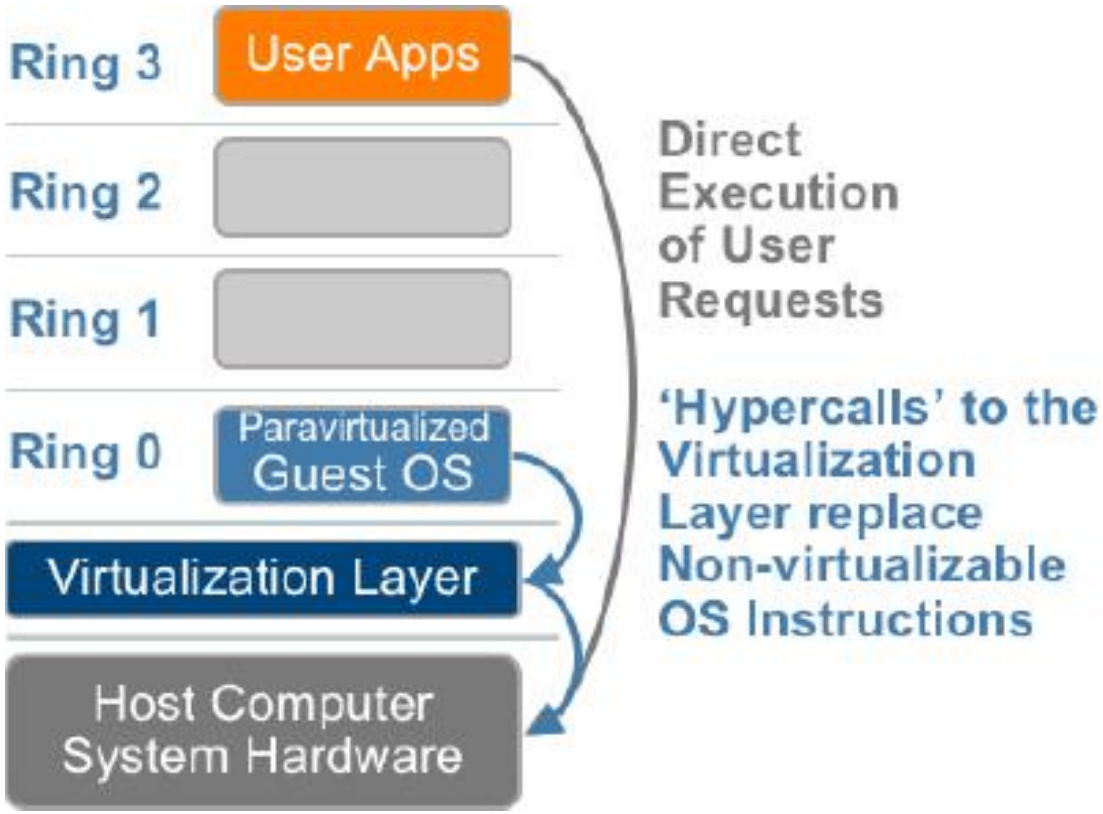
- En esta **técnica**, se modifica el **Guest** para que no llame a **instrucciones** críticas. Ahora si hay que tener **conocimiento** del **hipervisor** y debe colaborar con él. La VMM **proporcionará** una serie de interfaces para realizar las **hiperllamadas**.



Paravirtualización II

- Ejemplo de lugar de **descarga** de sistemas **huéspedes** adaptados a la **paravirtualización** (para Xen, en este caso): `http://stacklet.com/downloads/templates/xen3`
- Puesto que es necesario **modificar** el sistema operativo huésped, la **paravirtualización** suele estar presente solamente en entornos **Linux** (disponibilidad de código **fuente**)

Paravirtualización III



CASOS PARTICULARES

VMWARE

- Usa **virtualización** total y las **tecnologías** Intel-VT y AMD-V.
- Posibles sistemas **antitriones**: Linux y Windows.
- Posibles sistemas **huésped**: Linux y **Windows**, entre otros.
- **Difusión**: Muy **amplia**, sobre todo en entornos **personales**.
- Versiones:
  - **Workstation** 8 (escritorio): permite **ejecutar** y configurar máquinas **virtuales**; es de pago.
  - **Server** 2 (interfaz web): permite ejecutar y configurar **máquinas** virtuales; es **gratuito**, aunque VMware **no** presta **soporte**.
  - **Player** (escritorio): permite **ejecutar** máquinas **virtuales**; es **gratuito**.
- Sitio web: `http://www.vmware.com`

VirtualBox

- Emplea **virtualización** total y **explota** las tecnologías Intel-VT y AMD-V.
- Producto de **Oracle**.
- Sistemas **anfitriones**: **Linux** y **Windows**.
- Sistemas **huésped**: Linux y Windows, entre otros.
- **Difusión**: **Amplia**, centrada en entornos **personales**.
- Versión: 4.1.8 (**libre**).
- Sitio web: `http://www.virtualbox.org`

Bibliografía I

1. Stallings, William. **Operating systems : internals and design principles** – 6th ed–, Pearson education, cop. 2009
   - Capitulo Sistemas Operativos Modernos: Desarrollos. Páginas 77 a 80.
   - Capítulo Estructura del Sistema. Páginas 73 a 77.
2. **Arquitectura de Android**. `http://androideity.com/2011/07/04/arquitectura-de-android/`
   - Sección Arquitectura Android.
3. **Introducción a la arquitectura de un sistema Android**. `http://slashmobility.com/slash/cursos/introduccion-android/FO-2-Int` `pdf`
   - Sección Arquitectura Android.
4. **Introducción a Android** `http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/android.pdf`

Bibliografía II

- Sección Arquitectura Android.

5. **Linux Mint vs Ubuntu** `http://libuntu.wordpress.com/2013/04/03/linux-mint-vs-ubuntu-cual-distro-es-la-mejor-para-los`
   - Sección distribuciones Linux.
6. **Administración avanzada de SSOO. Tema 2 Virtualización**. `http://webs.um.es/einiesta/miwiki/doku.php?id=docencia`
   - Sección Virtualización.
7. **Virtualization techniques** `http://www.cs.nthu.edu.tw/~ychung/syllabus/Virtualization.htm`
   - Sección Virtualización.

Tema 3. Modelos Estructurales de Sistemas Operativos

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Tema 4. Proceso de Arranque y Parada

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Índice

Objetivos

- Analizar las **diferencias** y **similitudes** de los procesos de arranque en los sistemas **Windows** y **Linux**.
- Conocer la **evolución** del proceso de **arranque** los SSOO **Windows** a partir del **Vista**.
- Estudiar los sistemas de **protección** ante el **malware** en Windows 8.
- Estudiar los **cambios** en Linux de **System V** a **upstart**.
- Conocer la **organización** de los **archivos** necesarios para el arranque del **sistema**.

Índice

SISTEMAS WINDOWS.

UEFI BIOS I

- UEFI (**Unified Extensible Firmware Interface**) es una interfaz de firmware estándar para PCs, diseñada para **reemplazar** el **BIOS** (sistema básico de entrada y salida).
- Es un **estándar** creado por más de **140 compañías** tecnológicas que forman parte del consorcio **UEFI**, en el que se incluye **Microsoft**.
- Mayor **seguridad**, ya que ayuda a proteger el proceso previo al inicio (o **prearranque**) frente a ataques de **bootkit**.

UEFI BIOS II

- Tiempos de **inicio** y **reanudación** desde la **hibernación** más rápidos.
- **Compatibilidad** con unidades de más de **2,2 terabytes (TB)**.
- **Compatibilidad** con modernos **controladores** de dispositivos de **firmware** de 64 bits que el sistema puede usar para **direccionar** más de 17,2 mil millones de gigabytes (GB) de **memoria** durante el **inicio**.

UEFI BIOS III

ARRANQUE EN WINDOWS 7.

Cambios incorporados respecto versiones anteriores

- **NTLDR** (responsable de mostrar el menú de arranque y cargar el kernel) es substituido por **Windows Boot Manager** y el **Windows Boot Loader**.
- **boot.ini** (fichero que contenía entradas describiendo las distintas opciones de arranque) es substituido por el fichero de registro **boot configuration data (BCD)**.
- **Ntdetect.com** ha sido introducido dentro del kernel.
- Se deja de dar soporte a **perfiles de hardware** (Un perfil de hardware es un conjunto de instrucciones que indican a Windows qué dispositivos debe iniciar cuando se inicia el equipo, o que configuración debe usar para cada dispositivo.)

Boot Configuration Data

- Introduce **nuevas funcionalidades**: "Startup Repair tool"
- BCD se almacena en un fichero de datos con el **mismo formato** que el **registro** y que se localiza en partición EFI (para equipos que soportan EFI) o en el volumen del sistema.
- En sistemas operativos basados en **BIOS**, el fichero de registro de BCD se encuentra en $\backslash Boot \backslash Bcd$ de la partición activa.
- En sistemas operativos basados en **EFI** en $\backslash EFI \backslash Microsoft \backslash Boot$.

Boot Configuration Data

El fichero de datos BCD puede contener la siguiente información:

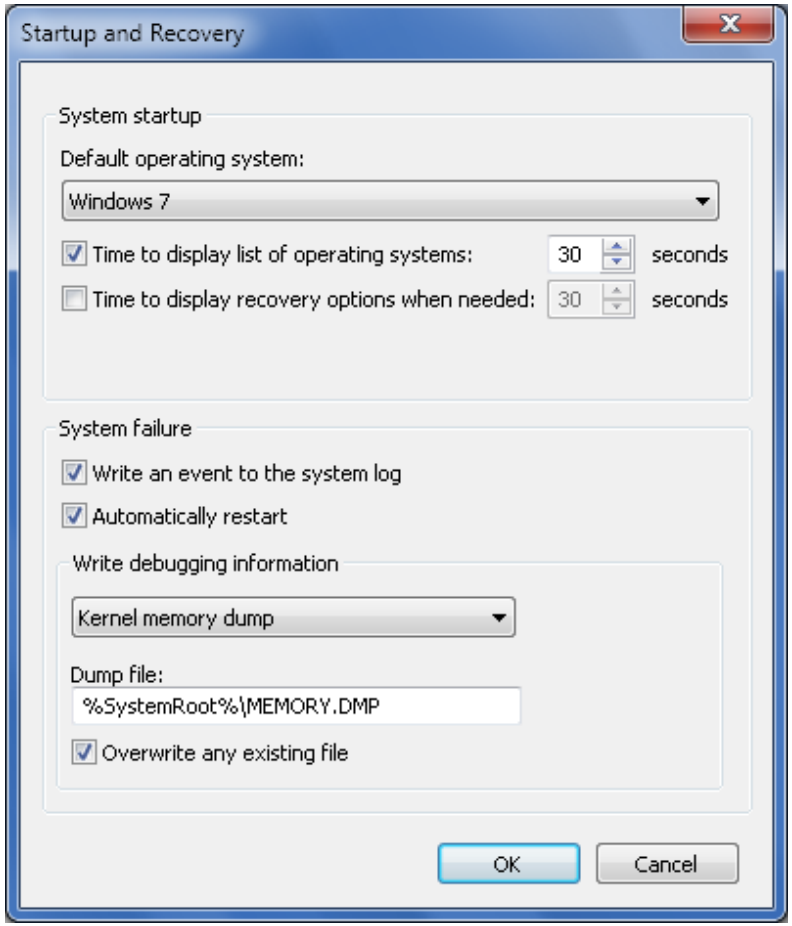
- Descripciones de configuración del **Windows Boot Manager** (*Bootmgr*)
- Entradas para el arranque del **Windows Boot Loader** (*Windows**System32**WinLoad.exe*).
- Entradas para la ejecución del **Windows Resume Application** (*Windows**System32**WinResume.exe*).
- Entradas para la ejecución del **Windows Memory Diagnostic** (*Boot**MemTest.exe*)
- Entradas para el NTLDR

Boot Configuration Data

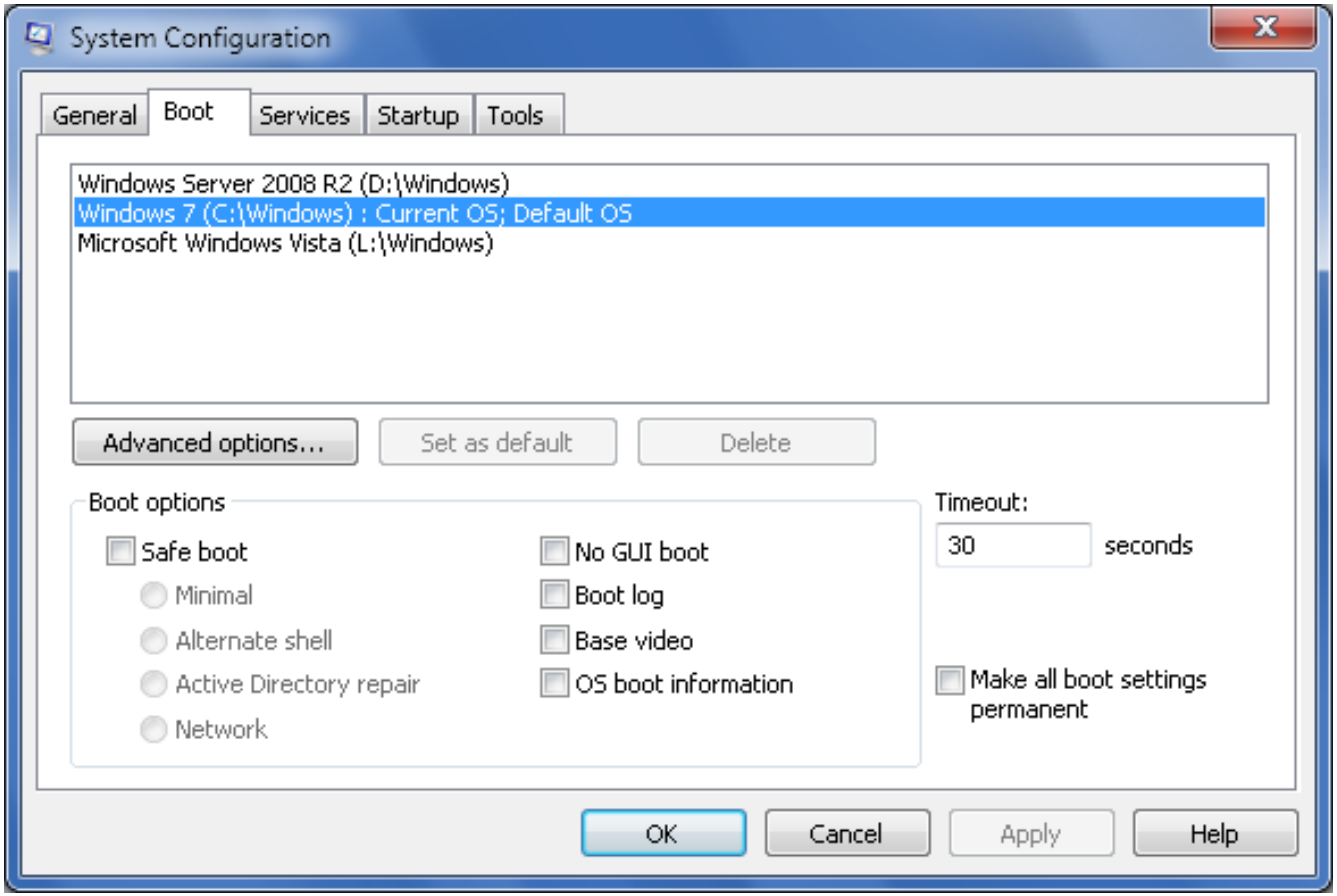
Se puede **modificar el BCD** de las siguientes (y distintas) formas:

- **Inicio y Restauración** (Propiedades avanzadas).
- Utilidad de Configuración del Sistema (**Msconfig.exe**).
- The BCD Windows Management Instrumentation (**WMI**) provider permite su utilización por scripts que modifiquen el BCD. **http://msdn.microsoft.com/en-us/library/aa362675.aspx**
- **BCDEdit.exe**: Por línea de comandos.
- Otras aplicaciones que **no** son de **Microsoft**, por ejemplo, VisualBCD.

Inicio y Restauración



Inicio y Restauración



BCDEdit

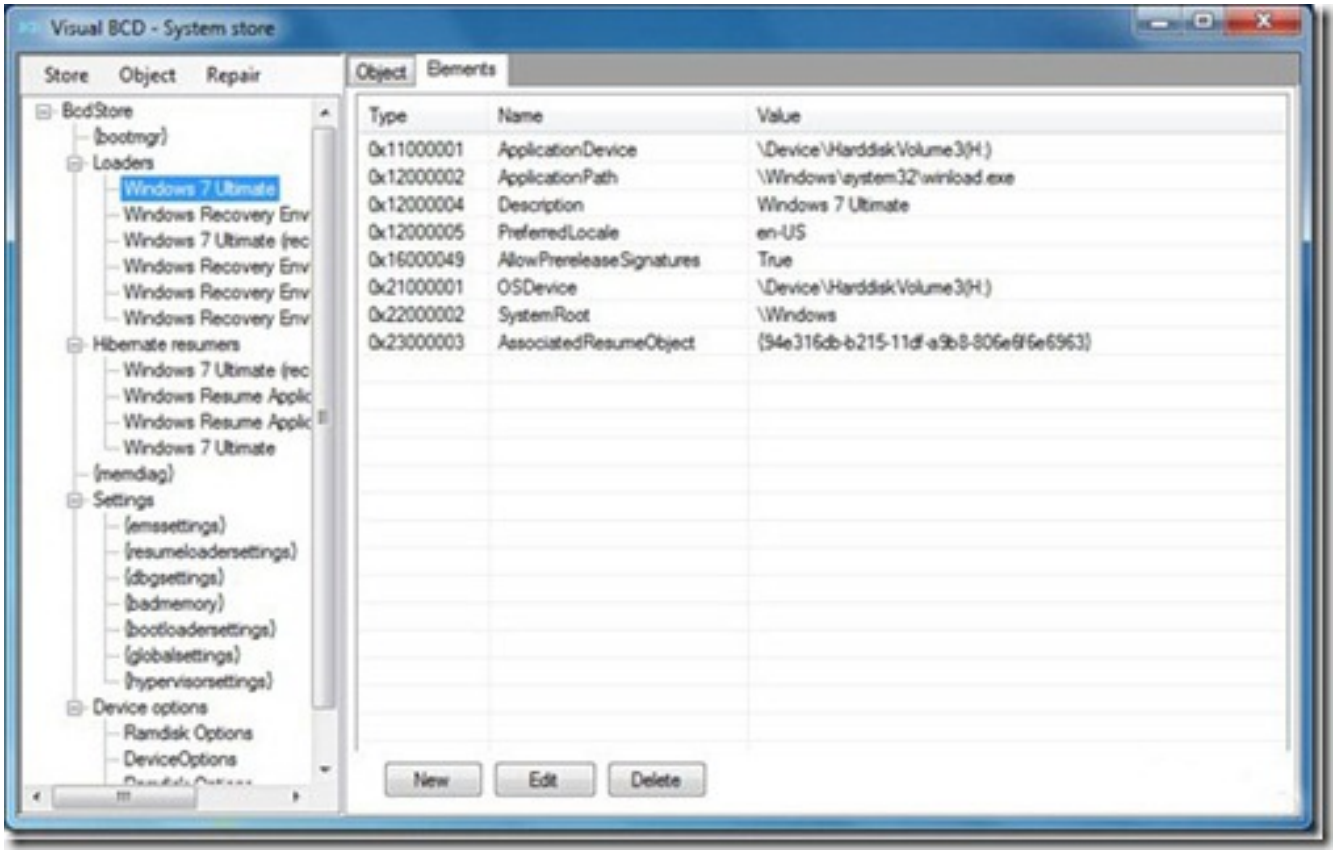
```
C:\>bcdedit /v

Windows Boot Manager
--------------------
identifier              {9dea862c-5cdd-4e70-acc1-f32b344d4795}
device                  partition=\Device\HarddiskVolume1
description             Windows Boot Manager
locale                  en-US
inherit                 {7ea2e1ac-2e61-4728-aaa3-896d9d0a9f0e}
default                 {ec70e4f9-87e6-11e1-a814-baa7f8fb05b7}
resumeobject            {ec70e4f8-87e6-11e1-a814-baa7f8fb05b7}
displayorder            {ec70e4f9-87e6-11e1-a814-baa7f8fb05b7}
                        {ec70e4fc-87e6-11e1-a814-baa7f8fb05b7}
toolsdisplayorder       {b2721d73-1db4-4c62-bf78-c548a880142d}
timeout                 30

Windows Boot Loader
-------------------
identifier              {ec70e4f9-87e6-11e1-a814-baa7f8fb05b7}
device                  partition=C:
path                    \Windows\system32\winload.exe
description             Windows 7
locale                  en-US
inherit                 {6efb52bf-1766-41db-a6b3-0ee5eff72bd7}
recoverysequence        {ec70e4fa-87e6-11e1-a814-baa7f8fb05b7}
recoveryenabled         Yes
osdevice                partition=C:
systemroot              \Windows
resumeobject            {ec70e4f8-87e6-11e1-a814-baa7f8fb05b7}
nx                      OptIn

Windows Boot Loader
-------------------
identifier              {ec70e4fc-87e6-11e1-a814-baa7f8fb05b7}
device                  vhd=[C:]\vhd\2008R2.vhd,locate=custom:12000002
path                    \Windows\system32\winload.exe
description             Windows Server 2008 R2
locale                  en-US
inherit                 {6efb52bf-1766-41db-a6b3-0ee5eff72bd7}
recoverysequence        {ec70e4fa-87e6-11e1-a814-baa7f8fb05b7}
recoveryenabled         Yes
osdevice                vhd=[C:]\vhd\2008R2.vhd,locate=custom:22000002
systemroot              \Windows
resumeobject            {ec70e4f8-87e6-11e1-a814-baa7f8fb05b7}
nx                      OptIn
```

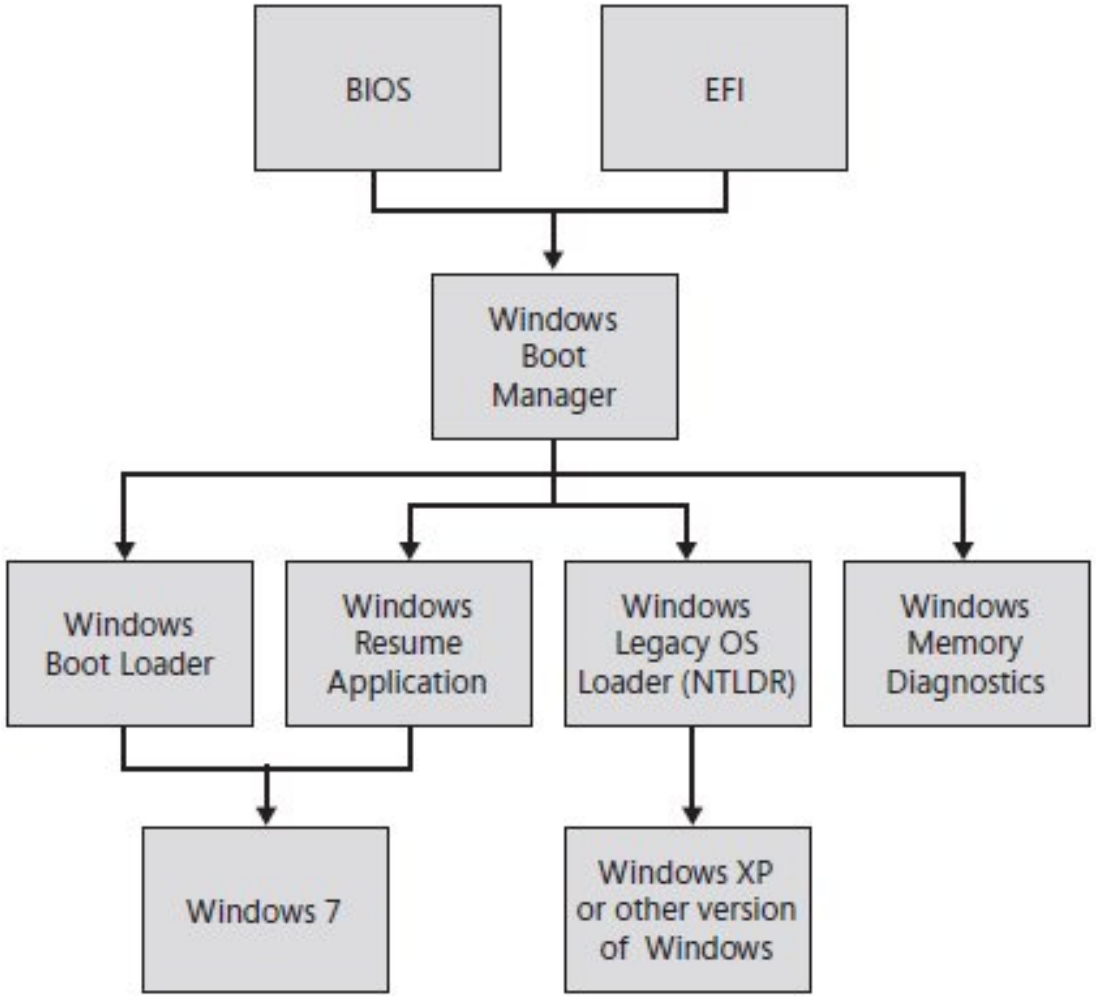
VisualBCD


Visual BCD - System store
Store
Object
Repair
BcdStore
{bootmgr}
Loaders
Windows 7 Ultimate
Windows Recovery Env
Windows 7 Ultimate (rec
Windows Recovery Env
Windows Recovery Env
Windows Recovery Env
Hibernate resumers
Windows 7 Ultimate (rec
Windows Resume Applic
Windows Resume Applic
Windows 7 Ultimate
{memdiag}
Settings
{emssettings}
{resumeloadersettings}
{dbgsettings}
{badmemory}
{bootloadersettings}
{globalsettings}
{hypervisorsettings}
Device options
Ramdisk Options
DeviceOptions
Object
Elements
Type
Name
Value
0x11000001
ApplicationDevice
\Device\Harddisk\Volume3(H:)
0x12000002
ApplicationPath
\Windows\system32\winload.exe
0x12000004
Description
Windows 7 Ultimate
0x12000005
PreferredLocale
en-US
0x16000049
AllowPrereleaseSignatures
True
0x21000001
OSDevice
\Device\Harddisk\Volume3(H:)
0x22000002
SystemRoot
\Windows
0x23000003
AssociatedResumeObject
{94e316db-b215-11df-a9b8-806e6f6e6963}
New
Edit
Delete

Arranque en Windows 7

1. Power-on self test (POST) phase.
2. Initial startup phase.
3. Windows Boot Manager phase.
4. Windows Boot Loader phase.
5. Kernel loading phase.
6. Logon phase.

Arranque en Windows 7



POST: Power-on Self Test

- **PROM Checksum**: La información sobre **dispositivos** en la **ROM** es íntegra.
- **Segment Map Address**: Se comprueba que la **memoria** es **funcional** al 100 %.
- **Page Map Address**: Divide la memoria en **segmentos de paginación** y se realizan las correspondientes comprobaciones.
- **Lectura Memoria C-MOS**. Esta **información** es utilizada en el resto del **proceso de arranque**.

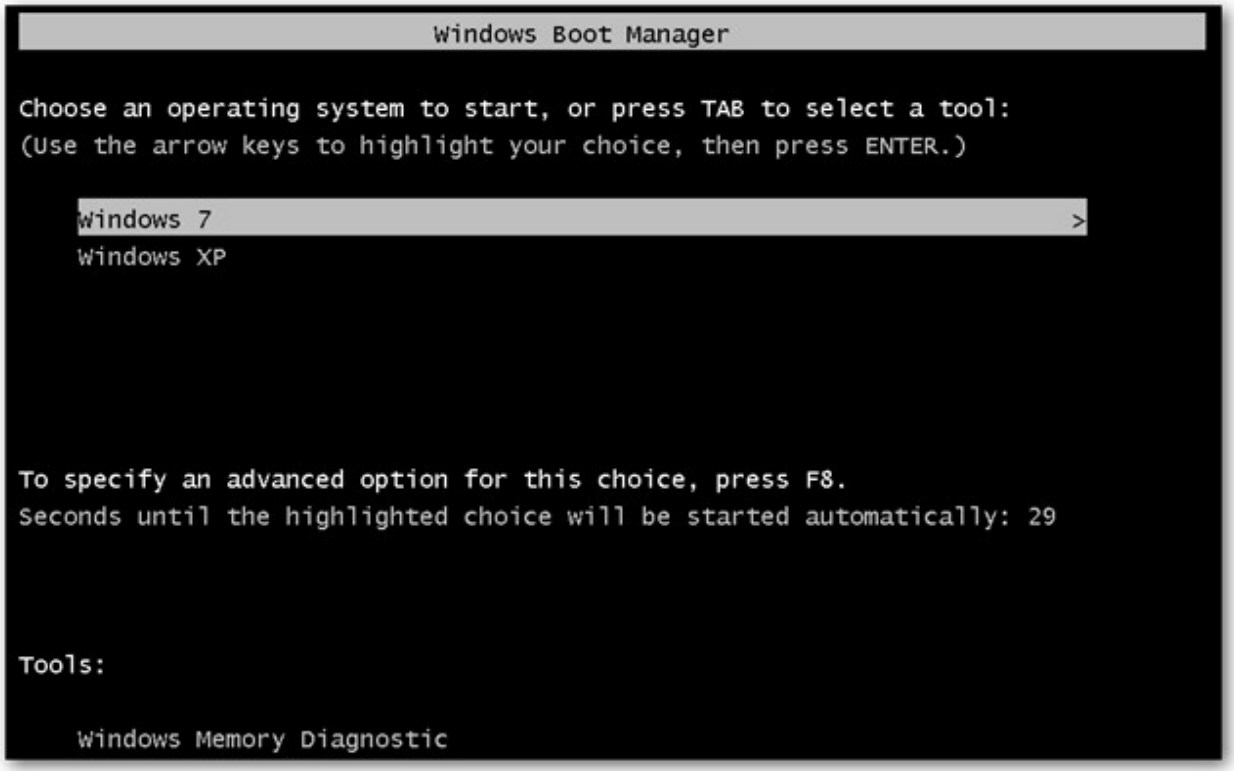
Initial startup phase

- Después del **POST** los computadores deben encontrar el **Windows Boot Manager.**
- Este proceso difiere si es un computador **EFI o BIOS**.
- El sistema **EFI** tiene su propio manejador de **arranque**. Al instalar Windows 7, se añade una entrada a este manejador de arranque que apunta al Windows Boot Manager: *\Efi\Microsoft\Boot\Bootmgfw.efi*
- Windows configura el EFI boot manager para que se muestre sólo **2 segundos** y así minimizar el tiempo de arranque.

Windows Boot Manager Phase

- En computadores que tienen un **único SSOO** no se muestra.
- De todas formas espera unos segundos por si el usuario pulsa F8 para acceder a **opciones avanzadas de arranque**.
- Si no se pulsa ninguna tecla se ejecuta el **Windows Boot Loader**.

Windows Boot Manager Phase



Windows Boot Loader Phase

1. Carga el **núcleo** Ntoskrnl.exe pero no lo ejecuta.
2. Carga la **Hardware Abstraction Layer** (HAL).
3. Carga el **registro del sistema**: (*System32\Config\System*)
4. Lee la clave *HKEY_LOCAL_MACHINE\SYSTEM\Services* para los **drivers de los dispositivos**. Carga todos los drivers configurados para el arranque en memoria. Los drivers no se inician hasta la ejecución del núcleo.
5. Permite la **paginación de memoria**.
6. Pasa el **control al núcleo** que inicia la siguiente fase.

Windows Boot Loader Phase

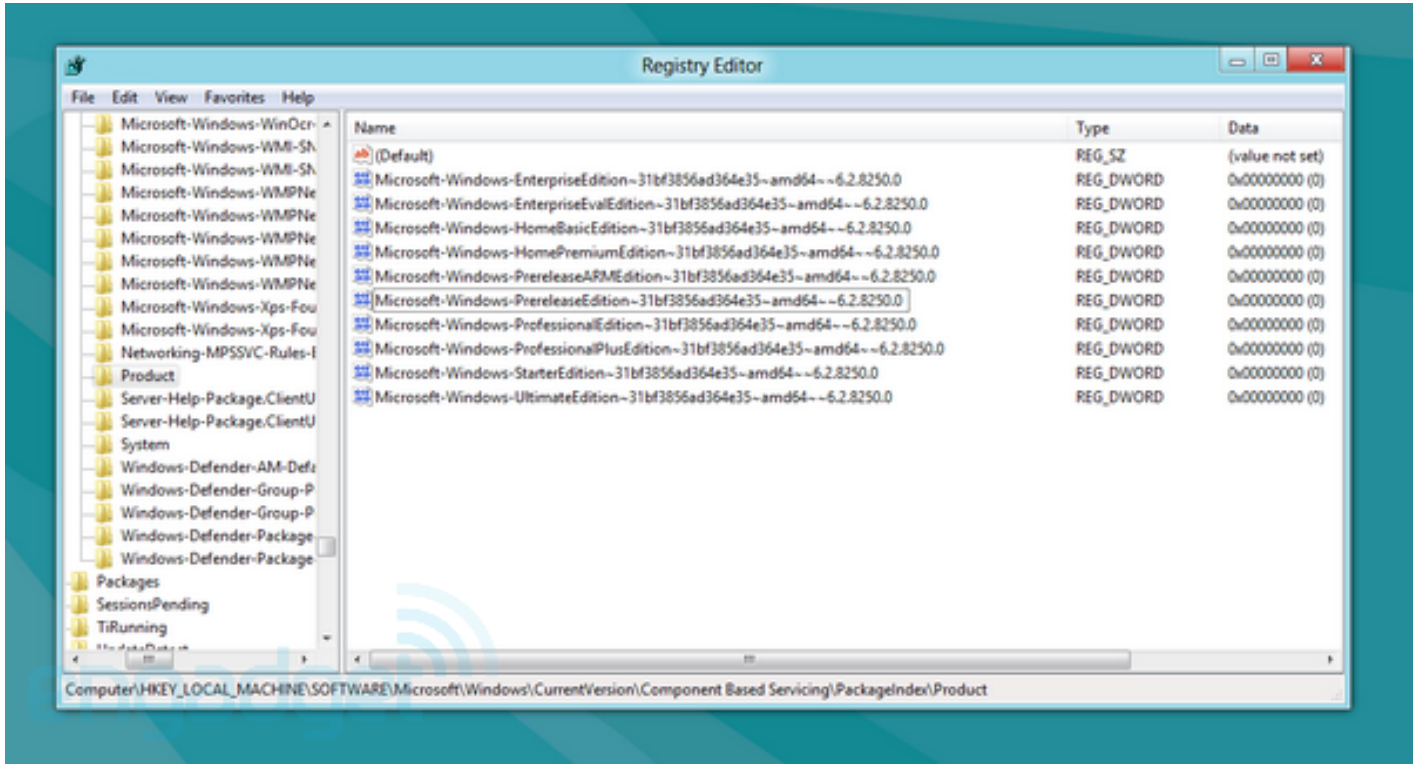


Figura: Editor del registro de Windows 8

Kernel Loading Phase

- Una vez cargados en memoria el **kernel y la HAL**, éstos de manera conjunta inicializan un grupo de software llamado **Windows Executive**.
- Windows Executive procesa la información de configuración en: (*HKLM\SYSTEM\CurrentControlSet*)
  iniciando los **drivers** y los **servicios**.

Kernel Loading Phase: Session Manager

Una vez iniciados los servicios y drivers, el núcleo ejecuta el **Subsistema Administrador de Sesiones** (Smss.exe).

- Es un proceso que se ejecuta hasta que el **computador se apaga**.
- Crea e inicializa las **variables de entorno**.
- Permite el cambio de modo **texto** a modo **gráfico**.
- Inicia el **Logon Manager** (*%SystemRoot%\System32\Winlogon.exe*).
- Crea páginas de **memoria virtual**.

Logon Phase

- **Winlogon.exe** es un servicio de sistema que permite “log on“ and ”log off“.
- Inicia la **Local Security Authority (LSA)** (Lsass.exe).
- Se pasa la información de manera segura a LSA que realiza la **autentificación**.

ARRANQUE EN WINDOWS 8.

Windows app store y Malware

- Se incorporan **características** para ayudar en la protección del **malware**.
- Todas las **aplicaciones** antes de estar en la **Windows Store apps** deben **satisfacer** una serie de **requerimientos**.
- Este proceso de **certificación** comprueba varios **aspectos**, incluyendo la seguridad para **prevenir** que se introduzca **malware** en la Store.
- Pero incluso, si entra software **malicioso**, Windows 8 incluye **características** de **seguridad** que pueden mitigar el **impacto**.
- Por ejemplo, estas **aplicaciones** se ejecutan de manera **aislada** (**sandbox**) y no tiene **privilegios** para acceder a los datos de los **usuarios** o cambiar **configuraciones** del sistema.

Aislamiento de procesos

- Es un **mecanismo** para ejecutar **programas** con seguridad y de manera **separada**. A menudo se **utiliza** para ejecutar **código** nuevo, o **software** de dudosa confianza **proveniente** de terceros.
- Ese **aislamiento** permite **controlar** de cerca los **recursos** proporcionados a los programas "**cliente**" a **ejecutarse**, tales como espacio **temporal** en discos y **memoria**.
- Habitualmente se **restringen** las **capacidades** de acceso a **redes**, la habilidad de **inspeccionar** la máquina **anfitrión** y dispositivos de entrada entre **otros**.
- En este sentido, el **aislamiento** de procesos es un **ejemplo** específico de **virtualización** (**applets**, celdas (**límites** en recursos impuestos por el **núcleo**), máquinas virtuales y máquinas **nativas**).

Niveles de protección

- Windows 8 tiene disponibles diferentes **niveles** de **protección** para aplicaciones de **escritorio** y **datos**.
- Windows **Defender** (antiguo Microsoft **AntiSpyware**) utiliza firmas (entradas tabla software **malicioso**) para **detectar** y "poner en cuarentena" aplicaciones **conocidas** como **maliciosas**.
- The **SmartScreen Filter** previene al **usuario** antes de que este ejecute **aplicaciones** no **confiables**.
- **Antes** de que una **aplicación** pueda cambiar **configuraciones** del sistema, el usuario tendrá que **autorizar** la utilización de **privilegios administrativos** para la misma.

Seguridad en el arranque: Trusted Boot

- Las **protecciones** anteriores contra el **malware** son válidas sólo cuando **Windows 8** está **arrancado**.
- ¿Qué **ocurre** para el **malware** moderno y más **específicamente** para **bootkits** (infección MBR) que son capaces de **iniciarse** antes del **arranque**?
- Al ejecutar **Windows 8** en un PC **certificado W8** o cualquier PC que soporte **UEFI**, **Trusted Boot** protege al PC desde el **arranque** hasta el inicio del software **antimalware** del sistema.
- Aunque **infectara** al **sistema** sí que al menos sería **detectado** el malware y no estaría **oculto**.

RootKits

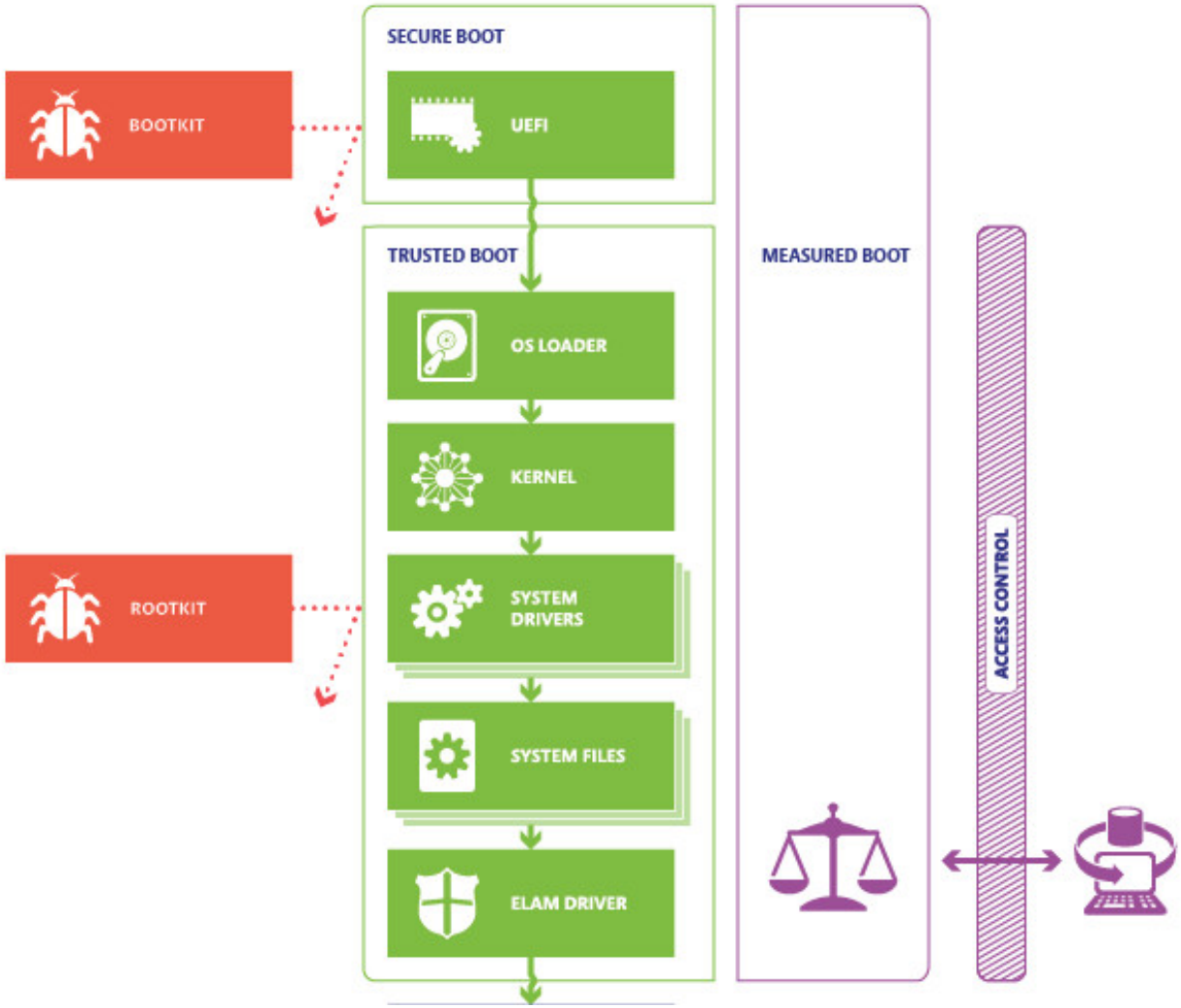
- Los **rootkits** son una clase **peligrosa** y **sofisticada** de **malware** que se ejecuta en **modo núcleo**, utilizando así los mismos **privilegios** que el SSOO.
- Al tener sus **privilegios** e **iniciarse** antes que el **SSOO**, ellos pueden **ocultarse** a otras **aplicaciones**.
- A menudo, los **rootkits** son parte de **programas completos** de malware que pueden solicitar **logins**, recordar **passwords**, transferir archivos de **usuario** e incluso capturar datos **encriptados**.

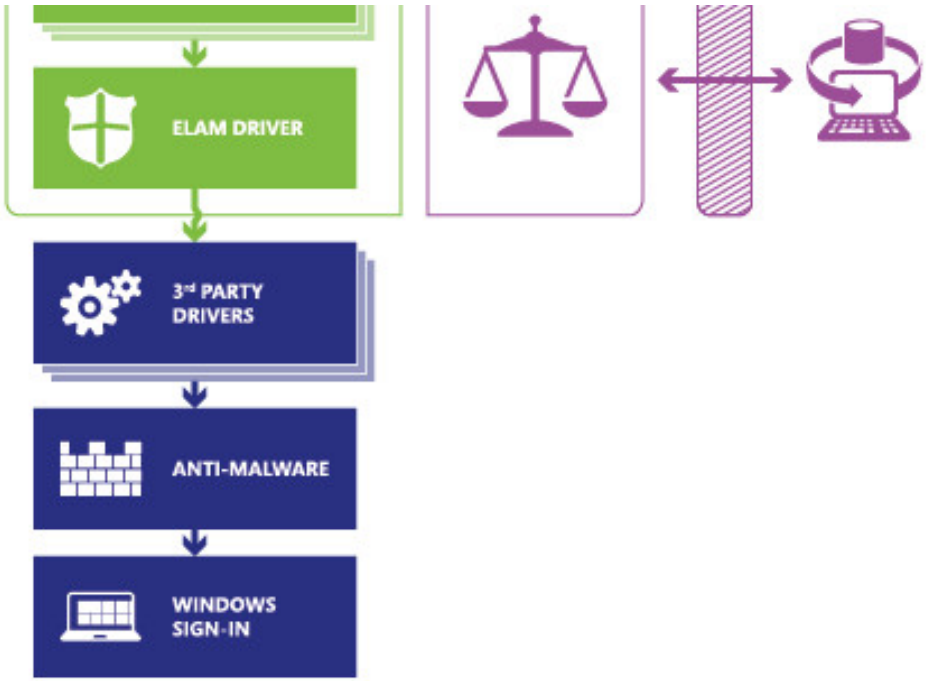
Tipos de rootkits

- **Firmware rootkits**: sobreescriben el **firmware** del sistema **básico** de I/O del PC o de otro **Hardware**. Es por esto, que puede **iniciarse** antes que el propio **SSOO**.
- **Bootkits**: substituyen el **bootloader** del sistema **operativo**, así el PC carga este tipo de **malware** antes que el **SSOO**.
- **Kernel rootkits**: substituyen una parte del **kernel** del **SSOO** y se iniciaría **automáticamente** una vez iniciado el **SSOO**.
- **Driver rootkits**: se hacen **pasar** por **drivers** confiables **utilizados** por el sistema para comunicarse con el **hardware** del PC.

Contramedidas de prevención rootkits y bootkits

- **Secure Boot**: los PCs con firmware **UEFI firmware** y **Trusted Platform Module** (TPM) pueden **configurarse** para cargar sólo **bootladers** confiables.
- **Trusted Boot**: Windows comprueba la **integridad** de cada **componente** del proceso de arranque antes de ser **cargado**.
- **Early Launch Anti-Malware (ELAM)**: Se encarga de **comprobar** todos los **drivers** antes de su carga, así **evita** la carga de **drivers** no aprobados.
- **Measured Boot**: El **firmware** del PC registra (logs) el **proceso** de arranque. Este es **enviado** por el SSOO a un **servidor** de **verificación** que puede **determinar** la "salud" del PC.


SECURE BOOT
BOOTKIT
UEFI
TRUSTED BOOT
MEASURED BOOT
OS LOADER
KERNEL
ROOTKIT
SYSTEM DRIVERS
SYSTEM FILES
ELAM DRIVER
ACCESS CONTROL



- **Secure Boot** y **Measured Boot** sólo pueden funcionar en PCs con **UEFI** 2.3.1 y un **chip TPM**. Afortunadamente, todos los **PCs certificados W8** satisfacen estos requerimientos, así como muchos PCs **diseñados** para versiones **anteriores** de Windows.

Secure Boot

- Los **PCs** sin este sistema simplemente **ejecutan** el **bootloader** en el HDD.
- Los PCs equipados con **UEFI** verifican primero que el **firmware** esté firmado **digitalmente**. Si está **activado** el Secure Boot, el firmware **comprueba** que el **bootloader** está **firmado** digitalmente y así comprobar que no ha sido **modificado**.
- Si esto es así, el **firmware** inicia el **bootloader** sólo si uno de las siguientes **condiciones** es verdadera:
  1. El bootloader fue **firmado** usando un **certificado** de **confianza**. En el caso de PCs certificados para W8, se comprueba el **certificado** de **Microsoft**.
  2. El usuario ha **aprobado** "manualmente" la **firma** digital. Esto **permitiría** al usuario a **arrancar SSOO** que no son de Microsoft.

Secure Boot

Todos los **PCs** basados en **X-86** y **certificados** para W8 deben **satisfacer** varios **requerimientos** ligados a Secure Boot:

- Deben tener por defecto **activado** el Secure Boot.
- Deben "confiar" en el **certificado** de **Microsoft** (y por tanto en cualquier bootloader firmado por Microsoft).
- Deben permitir **configurar** Secure Boot para "**confiar**" en otros **bootloaders**.
- Deben permitir al usuario **deshabilitar** de manera completa **Secure Boot**.

Los requerimientos anteriores permiten **protección** frente a **boot-kits** y permiten además, ejecutar cualquier **SSOO**.

Secure Boot: arranque de otros sistemas

Para arrancar otros sistemas existen tres opciones:

1. Utilizar un **SSOO** con un **bootloader certificado**. Como los PCs **certificados** para W8 deben "**confiar**" en el certificado **Microsoft**, ésta ofrece un **servicio** para analizar y firmar el **bootloader** de cualquier SSOO y por tanto satisfacer el **requerimiento** anterior. De hecho, para **Linux** existe ya un certificado obtenible en `http://sysdev.microsoft.com`.
2. Configurar **UEFI** para "confiar" en un **bootloader** concreto. Esto se realiza **añadiendo** una firma a la **BBDD** de la **UEFI**.
3. **Apagando** Secure Boot. Esto obviamente **inhabilita** la **protección** frente a **bootkits**.

Trusted Boot

- Una vez **finalizado** Secure Boot, **entra** en escena, **Trusted Boot**.
- El **bootloader** verifica la **firma digital** del kernel de **Windows 8** antes de su carga.
- **Complementariamente**, el **núcleo** verifica cada **componente** del proceso de **inicio** del SSOO, incluyendo los **drivers** para el arranque, **ficheros** de inicio y el **ELAM**.
- Si algún **fichero** ha sido **modificado**, no sería cargado.
- A menudo, el mismo W8 puede **reparar** de manera **automática** el componente "**corrupto**", recuperando así la **integridad** completa del **sistema** y permitiendo que el PC se inicie con **normalidad**.

ELAM: Early Launch AntiMalware

- **Secure Boot** protege al **bootloader** y Trusted Boot al **núcleo**. La siguiente oportunidad para el **malware** está en la infección de **drivers externos** a Microsoft.
- **Tradicionalmente** las aplicaciones de **antimalware** se iniciaban **tras** la carga de los **drivers**.
- **ELAM** realiza su función **antes**, pero esta debe ser **simple** para no **ralentizar** el arranque.
- ELAM se limita a **comprobar** que cada **driver** a arrancar está en la lista de drivers **certificados**. Si no es así, **no** lo **carga**.

Measured Boot

- Si un **PC** de nuestra organización llega a ser **infectado** por un **rootkit**, sería necesario **conocerlo**.
- Las aplicaciones **antimalware** pueden **informar** de **infecciones** al Dpto. **Técnico**, pero no funcionan con rootkits que **ocultan** su presencia.
- Se podría decir que **nunca** se puede **asegurar** que un PC está **libre** de **malware**.
- Por tanto, los PCs **infectados** pueden continuar **conectados** a la **red** de la empresa, capturando **passwords**, ficheros, etc.

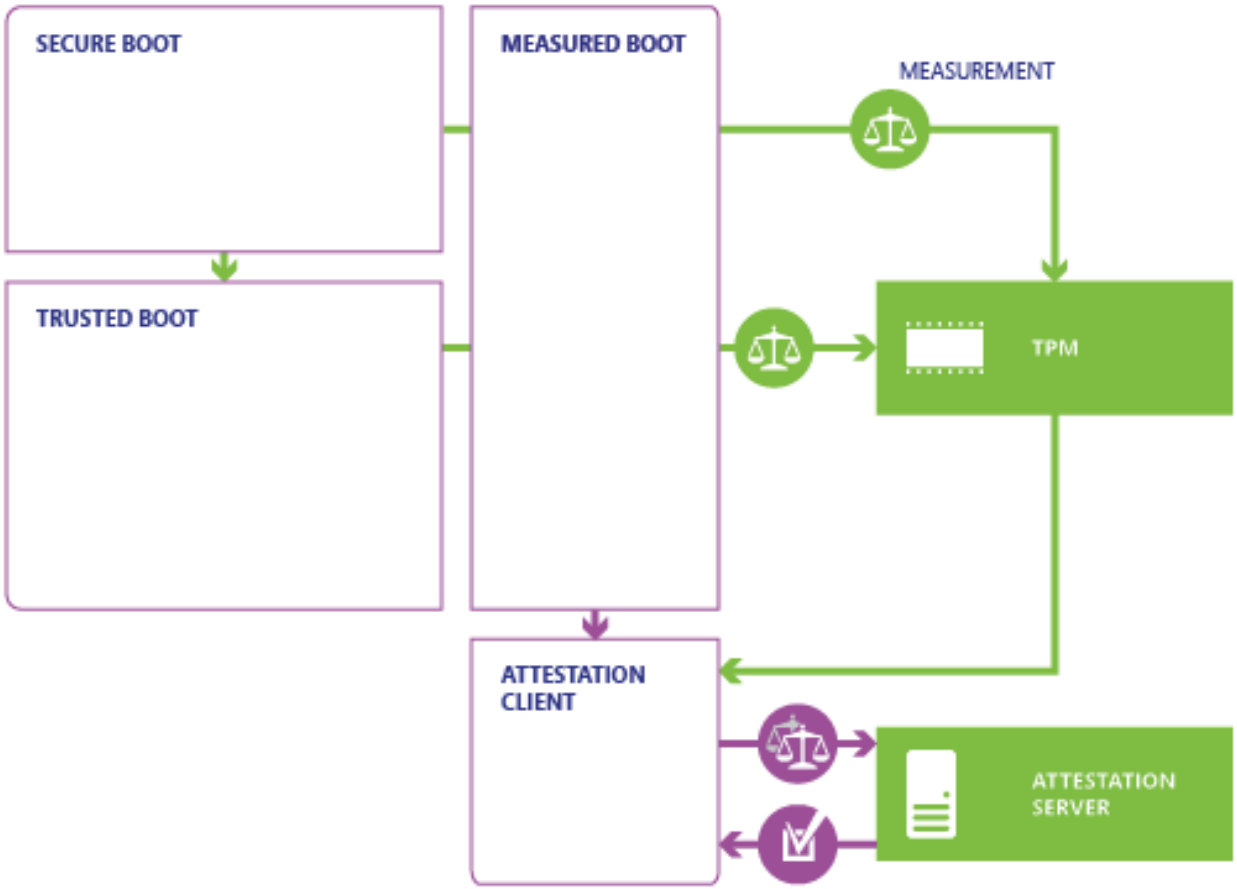
Measured Boot

Proporciona un **servidor** en la **red** que verifica la **integridad** de todo el proceso de **arranque** de Windows mediante el siguiente proceso:

- El firmware UEFI almacena en el **TPM** de manera **encriptada**, el firmware, el bootloader, los arranques de los **drivers** y todo aquellos que será cargado **antes** de la app **antimalware**.
- Al finalizar el **proceso** de **arranque**, Windows inicia el proceso de **certificación remoto** del cliente. El servidor de **certificaciones** envía al cliente una **llave única**.
- Esta llave la usa el **TPM** para firmar **digitalmente** el registro (log) **guardado** por la **UEFI**.
- El cliente **envía** el registro al **servidor**.

Dependiendo de la **configuración**, el servidor ahora determina si se **garantiza** al cliente acceso **limitado** o **completo** a la red.

Measured Boot



Measured Boot

- Windows 8 incluye los **programas** necesarios para dar **soporte** a **Measured Boot**, pero se necesitarán otras herramientas "**externas**" para montar el **servidor** externo de **certificación** y el **cliente** local.
- Existen herramientas como la **TPM Platform Crypto-Provider Toolkit** de Microsoft Research o de Microsoft Enterprise **Security MVP** la Dan Griffin's Measured Boot Tool.
- **Measured Boot** utiliza las **funcionalidades** de **UEFI**, TPM y Windows 8 para **proporcionar** una forma de asegurar de manera **confidencial** la confiabilidad de un PC **cliente** a través de la **red**.

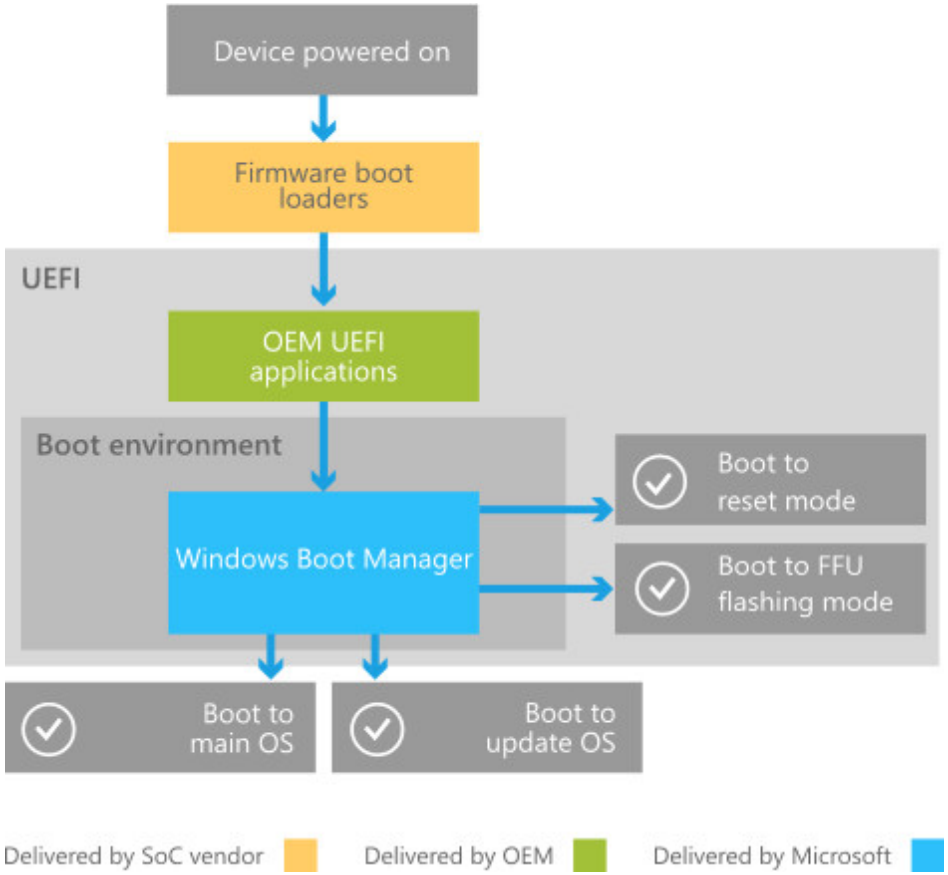
Conclusiones Arranque Windows 8

- **Secure** Boot, **Trusted** Boot y **Measured** Boot crean una **arquitectura** resistente a **bootkits** y **rootkits**.
- En Windows 8, estas **características** tiene el potencial de **eliminar malware** a nivel de **kernel**.
- Esta es la **solución** más **pionera** antimalware que **Windows** haya realizado hasta ahora.
- Con Windows 8 se puede de manera **efectiva** verificar la **integridad** del **SO**.

ARRANQUE

WINDOWS 10.

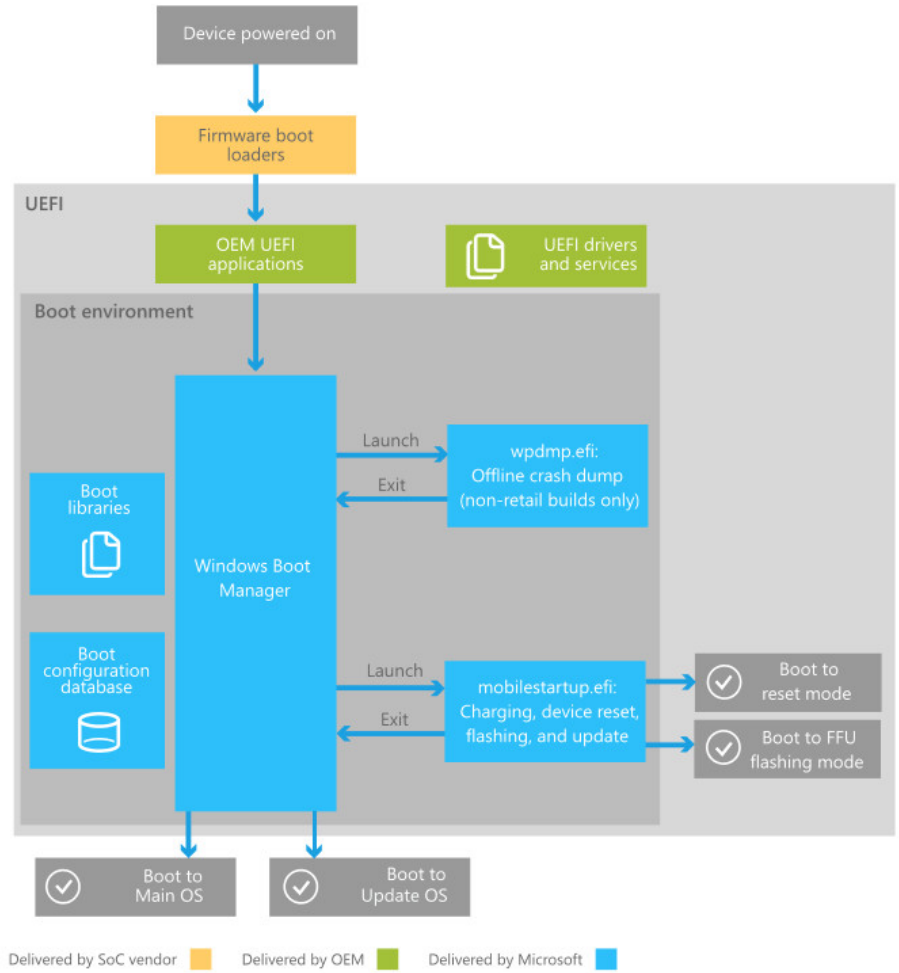
Proceso de arranque



Modos de arranque

- El **dispositivo** se asegura de que hay suficiente **potencia** para el arranque.
- El **arranque** se puede realizar para modificar o restaurar un sistema **operativo** sobre el computador o puede ser un arranque sobre el sistema operativo **principal**.
- El **update** OS es un SSOO mínimo proporcionado por **Microsoft** y se usa específicamente para instalar **actualizaciones**.
- FFU **flashing** mode se refiere a una aplicación **UEFI** que "flashes" una imagen de un SO a un **dispositivo** de **almacenamiento**.

Windows 10 Boot Manager



Componentes I

- El **Boot Manager** inicializa las **boot libraries** y lee la **boot configuration database** de donde obtiene que aplicaciones ejecutar en el arranque y en que orden. Las lanza de manera **secuencial** y cada **aplicación** devuelve el control al Boot Manager cuando **finaliza**.
- Las **Boot libraries** son librerías de funciones que extienden alguna **funcionalidad** de la **UEFI** y a las que sólo tienen acceso las **aplicaciones** lanzadas por el **Boot Manager**.

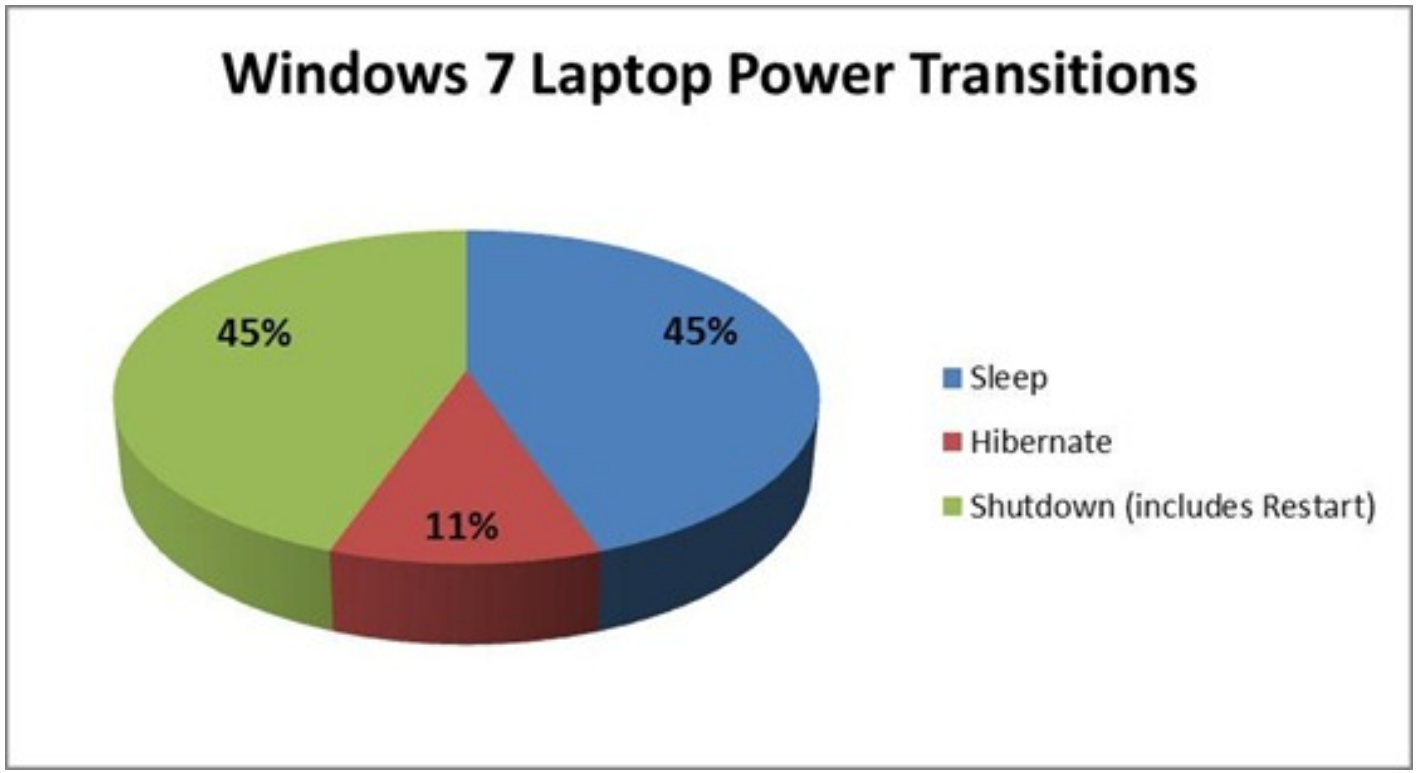
Componentes II

- En **acabados no controlados** de una sesión anterior, se puede recuperar parte de la memoria física con **wpdmp.efi**.
- Siempre se acaba ejecutando **mobilestartup.efi** que lanza algunos servicios de arranque, entre ellos de **carga de batería**, y decide si **arrancar** en flashing, reset o arranque **normal**.

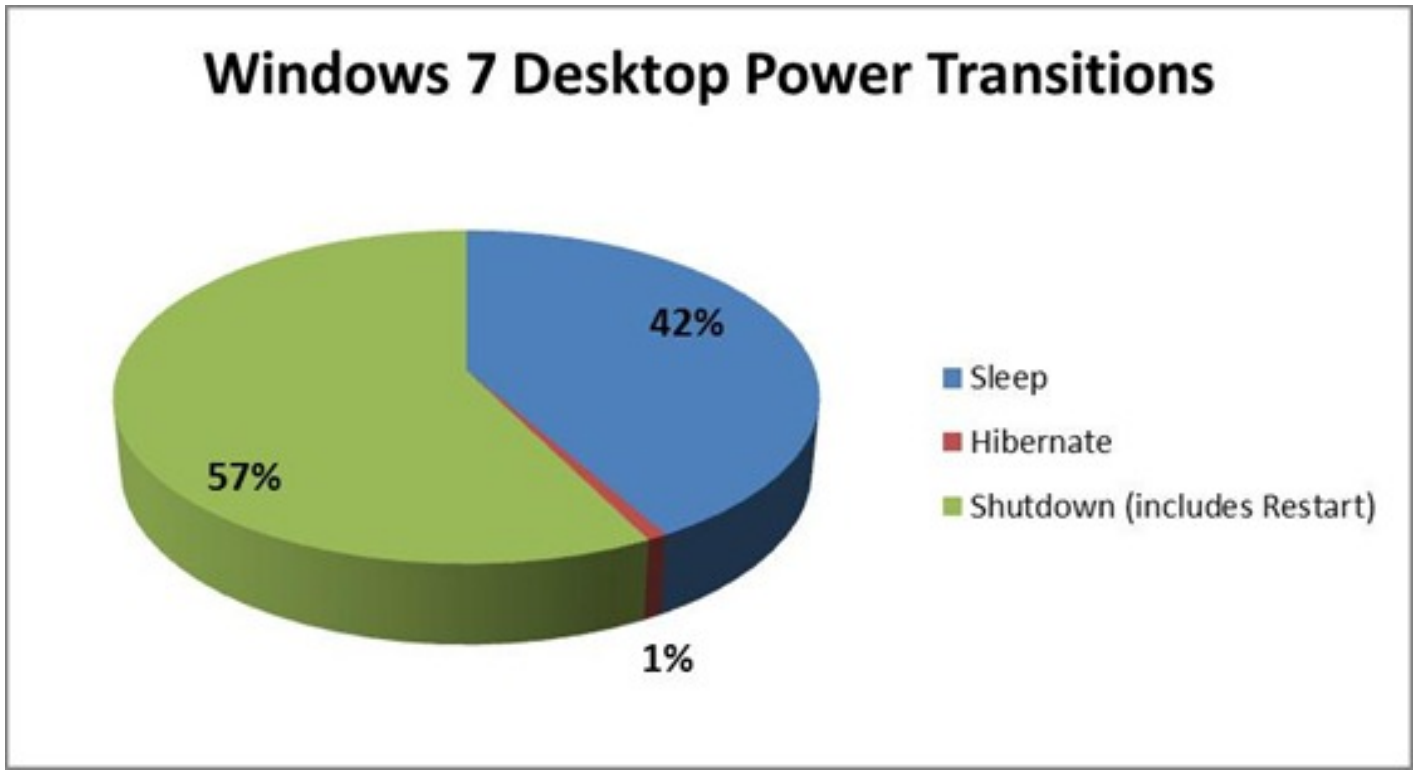
SUSPENSIÓN, HIBERNACIÓN Y APAGADO

WINDOWS 8.

Criterios de diseño: usuarios de Windows 7



Criterios de diseño: usuarios de Windows 7



Preferencias de los usuarios

- Muchos **usuarios** prefieren el **apagado** completo para que no se utilice ninguna **energía** y también así preservar la vida de la batería.
- Además al **arrancar** de nuevo se **elimina** todo lo que había en memoria de la sesión anterior (**fresh start**).
- La **hibernación** también permite que no se **consuma** energía ya que la **RAM** se vuelca a disco.
- La **suspensión** es la mejor opción para **apagados** y **encendidos** rápidos, pero consume **energía** para guardar los **contenidos** de la RAM. En **telefonía** móvil se trabaja con esta opción y en muy pocas **ocasiones** se reinicia el **dispositivo**.

Objetivos para Windows 8

- **Consumo** efectivo **cero**.
- **Sesión** nueva tras el **arranque**.
- **Tiempos** muy **rápidos** entre el instante en el que se **presiona** el botón de **arranque** y se puede utilizar el **PC**.
- Todo lo **anterior** sin requerir de nuevo **hardware**.

Proceso de apagado en Windows 7 I

1. El **usuario** inicia el **apagado** a través del menú inicio o del **botón** de apagado. Una **aplicación** puede iniciar también el apagado llamando a alguna API como **ExitWindowsEx**() o **InitiateShutdown**().
2. **Windows** envía **mensajes** a las **aplicaciones** en ejecución, pudiendo éstas salvar **datos** y **configuraciones**. Las aplicaciones pueden también **solicitar** un pequeño **tiempo** extra para **finalizar** alguna tarea en **proceso**.
3. Windows **cierra** las **sesiones** de **usuario** para cada usuario conectado.

Proceso de apagado en Windows 7 II

4. Windows envía **mensajes** a los servicios **notificándoles** que el proceso de **apagado** se ha iniciado, y por tanto **apaga** dichos **servicios**. Se apagan servicios en **paralelo** y en **serie** cuando hay **dependencias** entre éstos. Aquellos **servicios** que no responden se **apagan** de manera **forzada**.
5. Windows envía un **mensaje** a los **dispositivos** para que se **apaguen**.

Proceso de apagado en Windows 7 III

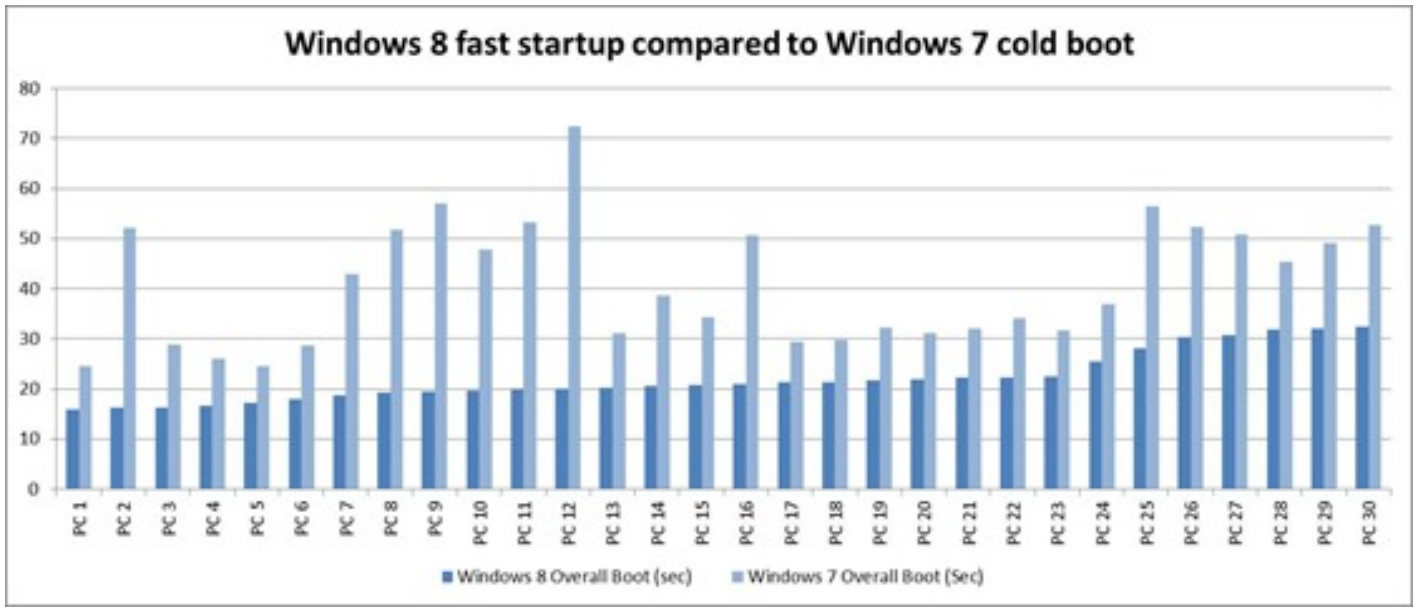
6. Windows cierra la **sesión** del sistema o del **kernel** (conocida como "**sesión 0**").
7. Windows **salva** cualquier **dato** temporal al disco duro del **sistema** para asegurar que se salva **correctamente**.
8. Windows envía una **señal** vía el interfaz **ACPI** (Advanced Configuration and Power Interface) al sistema para **apagar** el **PC**.

Proceso de apagado en Windows 8 I

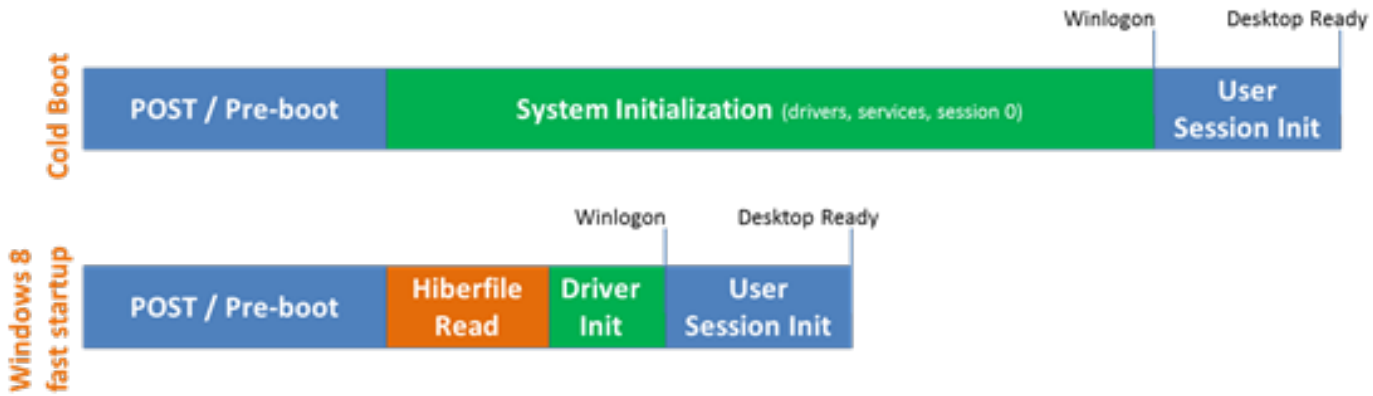
- Se podría decir que en un **apagado** "tradicional", se cierran todas las **sesiones** de todos los **usuarios** y en la "**sesión** del **kernel**" se cierran todos los **servicios** y **dispositivos** y preparando así al **sistema** para un **apagado** completo.
- Aquí se **encuentra** la **diferencia clave** para Windows 8: como en Windows 7, se **cierran** todas las **sesiones** de **usuario**, pero en vez de cerrar la sesión del **kernel**, ésta se **hiberna**.
- Si se **compara** con una hibernación **completa**, ésta incluye gran cantidad de **páginas** de **memoria**, que son las **usadas** por las **aplicaciones**, la hibernación de la "**sesión 0**" es mucho más **rápida**.

Proceso de apagado en Windows 8 II

- En la **hibernación** se guarda el **estado** del sistema y el contenido de la **memoria** al fichero **hiberfil.sys** y éste se lee de **nuevo** al **reiniciar** el equipo.
- La **ganancia** de **velocidad** con esta técnica **estaría** entre el 30 % y el 70 %.



Proceso de apagado en Windows 8 III



- Otro **factor** que **agiliza** este tipo de arranque es que ahora se permite al reiniciar **utilizar** todos los **cores** en un sistema multicore, en **paralelo**, dividiendo así el trabajo **necesario** para **leer** el **hiberfile** y **descomprimir** sus contenidos.

Hiberfile I

- Es un **fichero** muy **grande** que **normalmente** se dimensiona al 75 % de la **RAM**.
- El **fichero** es una **reserva** para **almacenar** los datos para la **hibernación**.
- **Normalmente** se almacena **información** equivalente a un un $10 - 15\,\%$ de la **RAM** física, pero esto varía **dependiendo** de los **drivers**, servicios y otros factores.

Hiberfile II

- El **sistema** trata de manera **ligeramente** diferente al **hiberfile** con respecto a otros **ficheros**. Por ejemplo, el **Volume Snapshot Service** (backup) lo ignora.
- Si se **deshabilita** la **hibernación** existen comandos que permiten reclamar este **espacio** para el sistema. Por ejemplo: "**powercfg /hibernate off**"
- El comando "**powercfg /hibernate /size**" permite especificar un valor entre **0** y **100** que indica el **porcentaje** de la **RAM** física que se reservará en el **hiperfile**.

Índice

ARRANQUE GENÉRICO.

Monousuario y Multiusuario

Arranque automático:

- **No** requiere de la **intervención** del administrador.

Arranque Manual:

- Se ejecutan sólo **scripts** básicos.
- Después se transfiere el **control** al **administrador**.
- **Monousuario** (sólo el administrador tiene **acceso** al sistema).
- Algunos **fallos** en dispositivos conducen al arranque **manual**.

Pasos del proceso de arranque

- **Carga** e inicialización del **kernel**.
- **Detección** y **configuración** de dispositivos.
- Ejecución de **procesos base**.
- **Monousuario**: Entrada del Administrador.
- Ejecución de archivos de comandos de arranque (**scripts rc**).
- Operación en **multiusuario**.

Carga e inicialización del núcleo

Núcleo (*unix*, *vmunix*, *vmlinuz*):

- Programa que se ejecuta en modo **supervisor** del **procesador**.
- Inicia las **tablas** de **interrupciones** y **excepciones**.
- Inicia la **gestión** de **memoria** paginada.
- Inicia el **scheduler**.
- Calcula la **memoria necesaria** para él.

Detección y configuración de dispositivos

El **núcleo** tiene **asignados** una serie de **dispositivos**.

- Intenta determinar si un **dispositivo** está **presente**.

Si está presente:

- Intenta determinar **parámetros** avanzados de **funcionamiento**.
- Inicia los **controladores** (device drivers).

Una vez finalizado se monta el **directorio raíz** en algún dispositivo. (**panic**).

ARRANQUE DETALLADO.

Arranque de un PC I

- El **arranque** tiene dos fases: arranque **hardware** y arranque **software**

Bajo el control del iniciador ROM	Test del hardware Carga en memoria del cargador del SO
Bajo el control del cargador (*boot*) del SO	Carga en memoria componentes del SO
Inicialización bajo el control de la parte residente del SO	Test del sistema de ficheros Creación de estructuras de datos internas Completa la carga del SO residente Creación de procesos *login*

Se entra en la fase normal de funcionamiento del SO

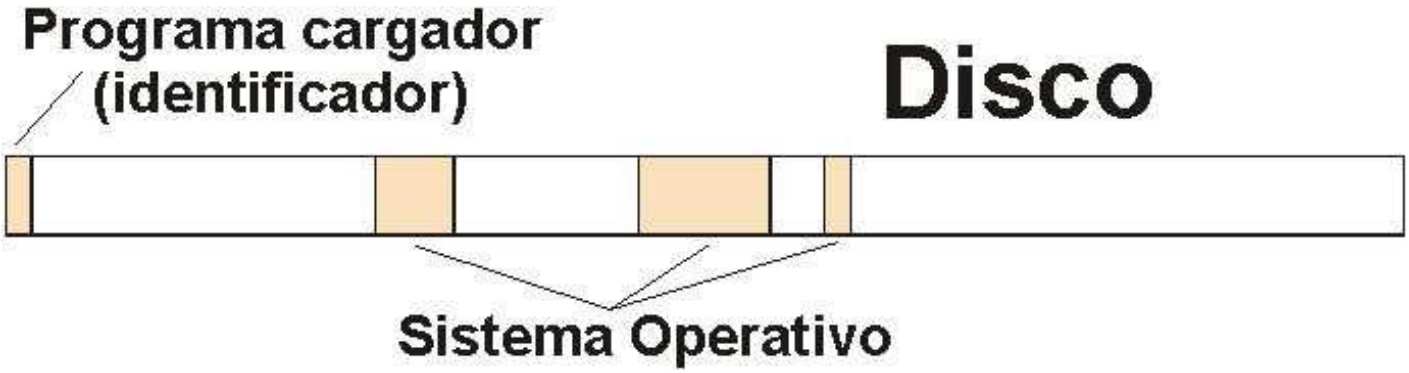
Arranque de un PC II

- **Iniciador ROM**: programa de arranque disponible en la ROM.
- Al arrancar el ordenador, se envía una **señal eléctrica** y se cargan un conjunto de valores **predefinidos** en los registros.
- Por ejemplo, en el contador del programa se carga la dirección de inicio del iniciador **ROM**.
- El iniciador ROM realiza **tres funciones**:
  1. **Comprueba** el sistema, detectando sus **características** y comprobando su **funcionamiento**.
  2. Lee y **almacena** en memoria el programa cargador del S.O.
  3. Pasa el control al **cargador** del S.O., saltando a la **dirección** de memoria donde lo ha **almacenado**.

Arranque de un PC III

- El programa **cargador** (master boot program o boot program) está en los primeros **sectores** del disco y con un tamaño **prefijado**.
- Estos sectores se conocen como **Master Boot Record** (o Volume Boot Record).
- Es el **encargado** de cargar el **núcleo** (o kernel) del S.O. y pasarle el control.
- El iniciador de la **ROM** y el S.O. tienen un **acuerdo** sobre el programa **cargador** (ubicación, dirección de arranque y tamaño), de esta manera el **iniciador** puede **soportar** varios S.Os

Arranque de un PC IV



- El **núcleo del S.O.** continúa el proceso de **arranque**: realiza una comprobación del **hardware** del sistema, se prepara a sí mismo para ejecutar el sistema inicializando sus tablas internas, creando las **estructuras de datos** necesarias, etc.
- A **continuación** crea el proceso **Init** y le pasa el **control**

Arranque de un PC V

- Para que el **núcleo** no tenga un tamaño muy **grande**, la mayoría de las opciones se **compilan** como **módulos**, que se cargarán cuando se **necesiten**.
- Por ello, en el **arranque** el núcleo necesitará cargar **algunos** módulos para poder **iniciar** el sistema, p.e. el módulo **ext3** para acceder al **SF**.
- El fichero *initrd_version.img* cargará los **módulos** que el núcleo necesita para poder **arrancar**: el núcleo primero carga el **initrd** y le pasa el control, el initrd carga los **módulos** necesarios y le devuelve el control al núcleo y entonces el núcleo **continuará** el proceso de arranque.

Arranque de un PC VI

- El proceso Init termina el **proceso** de **arranque**, dejando el sistema en modo **multiusuario** preparado para que los usuarios **trabajen** en él.
- El proceso **Init** usa una serie de ficheros **scripts** que le indican las acciones a realizar.

Arranque de un PC VII

Las **tareas** que realiza el proceso **Init** son:

- **Chequea** los sistemas de **ficheros**.
- **Monta** los sistemas de ficheros **permanentes**.
- Activa las áreas de **swapping** o intercambio.
- Activa los **demonios** y la **red** (NFS, NIS, etc.).
- **Limpia** los sistemas de **ficheros** (borra los directorios temporales).
- Habilita el **login** a los **usuarios** del sistema.

GESTOR DE ARRANQUE.

GRUB I

- GRUB entiende **sistemas de ficheros** y formatos ejecutables del núcleo, así que permite arrancar un sistema operativo cualquiera sin necesidad de saber la **posición física del núcleo en el disco**.
- Ofrece una interfaz **semigráfica** (menús y recuadros de texto) muy intuitiva al arrancar,
- En cualquier **momento** se pueden editar los **parámetros** de las **opciones** del menú de arranque, con un editor simple pero efectivo.
- Cada vez que se **actualiza el kernel** o la configuración de arranque no hay que ejecutarlo.

GRUB II

- GRUB se **instala** en el master boot record **(M.B.R.)** y hace de las funciones de master boot program **(M.B.P.).**
- Fichero de **configuración**: /boot/grub/grub.cfg (en algunos sistemas es /boot/grub/menu.lst).
- **Ejecutable** (para instalación): /usr/sbin/grub-install.
- Soporta el modo **Direccionamiento Lógico** de **Bloques** (LBA).
- **Lee** los sistemas de **ficheros** Ext2, Ext3 y Ext4.
- GRUB se puede **instalar** el sector de **arranque** de la partición de Linux, en este caso sólo se **lanzará** si es esa la partición **activa**.
- Lee el fichero de **configuración** en cada **arranque** (las **modificaciones** se toman de manera **automática**).

GRUB III

- Dispone de tres **interfaces** con distinto grado de **funcionalidad**:
  - De **menú** $\rightarrow$ **seleccionar** S.O.
  - Del **editor** de menú de entrada: permite **modificar** líneas de **órdenes** antes de arrancar el **sistema operativo**, p.e. para pasar **parámetros** al **núcleo** o corregir **errores** del fichero. (tecla e), para pasar **parámetros** al **núcleo** (tecla a).
  - **De línea de órdenes**: ejecutar órdenes **interactivamente** (tecla c).

GRUB IV

- Terminología de **GRUB**, numerando los **dispositivos** según los reconozca la **BIOS**:

Nombres de dispositivos

$$(< tipo\_de\_dispositivo >< numero\_dispositivo\_bios >, < numero\_particion >)$$
(hd0,0), /dev/sda1

Nombres de ficheros

$$(< tipo\_de\_dispositivo >< numero\_dispositivo\_bios >, < numero\_particion >)/path$$
(hd0,0), /boot/grub/grub.conf

GRUB V

- (hd0,0) first primary partition of the first hard disk
- (hd0,1) second primary partition
- (hd0,2) third primary partition
- (hd0,3) fourth primary partition (usually an extended partition)
- (hd0,4) first logical partition
- (hd0,5) second logical partition
- (hd0) /dev/hda
- (hd0,0)/dev/hda1
- (hd1,4)/dev/hdb5

GRUB VI

```
default=0                              #Opción por defecto
timeout=10                             #Tiempo de espera
password --md5 $1$4hKKr1$LvSjN89PmeeHXBljr13yq0
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Fedora Core Linux (2.6.37)       #Etiqueta
     root (hd0,0)       #Parti. a montar por GRUB, donde está el núcleo
     kernel /boot/vmlinuz-2.6.37 ro root=LABEL=/    #Núcleo y parámetros
     initrd /boot/initrd-2.6.37.img                 #Fichero initrd
title Windows 2008                     #Etiqueta
     rootnoverify (hd0,2)              #Partición a usar
     chainloader +1                    #Pasarle el control


# Observa esta diferencia-----------------------------------------------
# root=LABEL=/  Indica al núcleo cuál es el SF raíz del SO
# root (hd0,0)  Dónde encuentra GRUB los ficheros del núcleo
```

RUNLEVELS Y UPSTART.

Runlevels en Debian I

- Los sistemas operativos consideran que el **sistema** puede **estar** en distintos niveles de **ejecución** (o arranque), y no sólo en **multiusuario** y **monousuario**.
- En Linux estos **niveles** de **ejecución** son:
  - 0: el sistema está **apagado**
  - 1, s ó S: modo **monousuario**
  - 2: **multiusuario** sin funciones de red, el mismo que el 3 pero sin las **utilidades** de red. (En ocasiones está sin usar y puede ser redefinido).
  - 3: multiusuario **completo**, con terminales en modo **texto**.
  - 4: sin usar, puede ser **redefinido** por el **administrador**.
  - 5: **multiusuario** con **pantalla** de inicio de **sesión** basada en X.
  - 6: el **sistema** se está **reiniciando**.

Runlevels en Debian II

- /sbin/telinit: permite **cambiar** de nivel de **ejecución**:
  - telinit 1: a modo **monousuario**.
  - telinit 6: **reiniciar** el sistema.
  - telinit 3: **cambiar** al nivel 3.
- /sbin/runlevel: saber en qué **nivel** está el **sistema**.
- Hay un nivel por **defecto** en el que **arranca** el **sistema**, que se **establecía** en el fichero /etc/**inittab** con la línea: **id:5:initdefault:**.
- En **Upstart**, el nivel por defecto se **define** en /etc/init/rc-sysinit.conf (env).

Runlevels en Debian III

- A partir de Ubuntu 6.10, los **niveles** de **ejecución** son:
  - 0: **Halt**-
  - 1: **Single**-user mode.
  - 2: **Graphical** multi-user with **networking**
  - 3-5: **Unused** but configured the same as **runlevel** 2.
  - 6: **Reboot**.
- Al **arrancar**, mediante el **GRUB**, al **núcleo** se le puede pasar como **parámetro** un número **indicando** el nivel en el que queremos arrancar. En este caso se obviará el nivel por **defecto**.

Targets a partir de Ubuntu 16.04

- El término **runlevel** se substituye por el de target.
- El comando **systmctl** permite **cambiar** de target (isolate) y establecer el target por **defecto** (set-default):

Runlevel: Target
0: poweroff.target
1: rescue.target
2, 3, 4: multi-user.target
5 graphical.target
6 reboot.target

Arranque mediante Upstart I

- Proceso de **arranque/parada** del sistema **basado** en **eventos**, que son ejecutados por el proceso **Init**.
- De forma **asíncrona** realiza las **siguientes** tareas:
  - **Dirige** el inicio de las **tareas** y **demonios**.
  - **Controla**, si es **necesario**, los **demonios** mientras el sistema está **encendido**.
  - **Detiene** los **demonios** durante el proceso de **apagado**.
- En el **directorio** /etc/init (antes en el /etc/event.d/) hay una serie de **eventos** (ficheros) que **Init** ejecuta según el **orden** y las dependencias **establecidas** en los **mismos**.
- Estos **eventos** indican qué tarea **ejecutar**, cuándo y cómo, **mediante** su propio **lenguaje**.

Arranque mediante Upstart II

- La orden **initctl** permite indicar al **proceso** Init que realice determinadas **acciones**:
  - **start** evento.
  - **stop** evento.
  - **status** evento.
- Los **ficheros de eventos** están en modo **texto** siguiendo la siguiente **nomenclatura**:

Arranque mediante Upstart III

`exec <orden><argumentos>` ⇒ ejecutar la orden con esos argumentos:

```
exec /etc/rc.d/rc 0
exec /sbin/mingetty tty2
```

`script...end script` ⇒ ejecutar el guión shell indicado:

```
script
        set $(runlevel || true)
        if [ "$2" != "0" ] && [ "$2" != "6" ]; then
            set $(runlevel --set 0 || true)
        fi
        if [ "$1" != "unknown" ]; then
            PREVLEVEL=$1
            RUNLEVEL=$2
            export PREVLEVEL RUNLEVEL
        fi
        exec /etc/rc.d/rc 0         # Combina exec con script
end script
```

Arranque mediante Upstart IV

`start on <event>` ⇒ describe bajo qué condiciones se lanzará ese evento

```
start on startup                start on stopped rc2
start on runlevel 5             start on started prefdm
```

`stop on <event>` ⇒ describe bajo qué condiciones se parará ese evento

```
stop on runlevel [35]           stop on started prefdm
```

`respawn` ⇒ volver a lanzar ese proceso o demonio cuando muera

`console` ⇒ hacia dónde redirigir la salida del evento

`pre-start` ⇒ ejecutar la orden/guión shell antes de lanzar ese proceso

```
pre-start exec rm -f /var/run/crond
pre-start script
    if [ "$RUNLEVEL" == "S"]
    then
       RUNLEVEL=1
    fi
end-script
```

Arranque mediante Upstart V

`post-start` ⇒ ejecutar la orden/guión shell después de lanzar ese proceso

```
post-start exec touch/var/run/crond
post-start script
    if [ "$RUNLEVEL" == "1"]
    then
       RUNLEVEL=S
    fi
end-script
```

`pre-stop` ⇒ ejecutar la orden/guión shell antes de parar ese proceso

```
pre-stop exec ...
pre-stop script  /  end-script
```

`post-stop` ⇒ ejecutar la orden/guión shell después de parar ese proceso

```
post-stop exec ...
post-stop script  /  end-script
```

Arranque mediante Upstart VI

```
start on startup
stop on runlevel
console output
script
        /etc/rc.d/rc.sysinit
end script
post-stop script
  runlevel=$(/bin/grep initdefault /etc/inittab | cut -f 2 -d ":")
  [ -z "$runlevel" ] && runlevel="3"
  exec telinit $runlevel
end script
-------------------------------------------------
start on runlevel 5
stop on runlevel [!5]
console output
script
        exec /etc/rc.d/rc 5
end script
```

Ficheros de Inicialización I

- Varios **ficheros** script **generales**, llamados **rc*** que están en /etc, son **ejecutados** al arrancar por el **Init** mediante un evento.
- Varios **ficheros** scripts **específicos** de cada nivel de **arranque**, que están en el **directorio** /etc/rcn.d siendo n el nivel de arranque, son **ejecutados** al arrancar o cambiar de **nivel**.
  - Son **ficheros** scripts que su nombre **empieza** por K o S, **seguido** de un número de 2 dígitos y un nombre **descriptivo**:

Ejemplo nombre ficheros de inicialización

K35smb K15httpd S40atd S50xinetd S60cups S99local

  - El script /etc/rc (o /etc/rc.d/rc) es el **encargado** de ejecutar estos ficheros **scripts**, según el nivel de **arranque**.

Ficheros de Inicialización II

- Los ejecuta en orden **alfabético**, primero los K después los S, los dos dígitos **establecen** el orden entre todos los **K** y todos los **S**.
- Los ficheros K sirven para **detener** demonios o **matar** procesos.
- Los ficheros S sirven para **lanzar** demonios o **ejecutar** funciones de inicialización.
- De esta manera, para cada nivel de **inicialización**, se especifica qué **demonios** tienen que estar activos y qué demonios no **tienen** que estar **activos**.

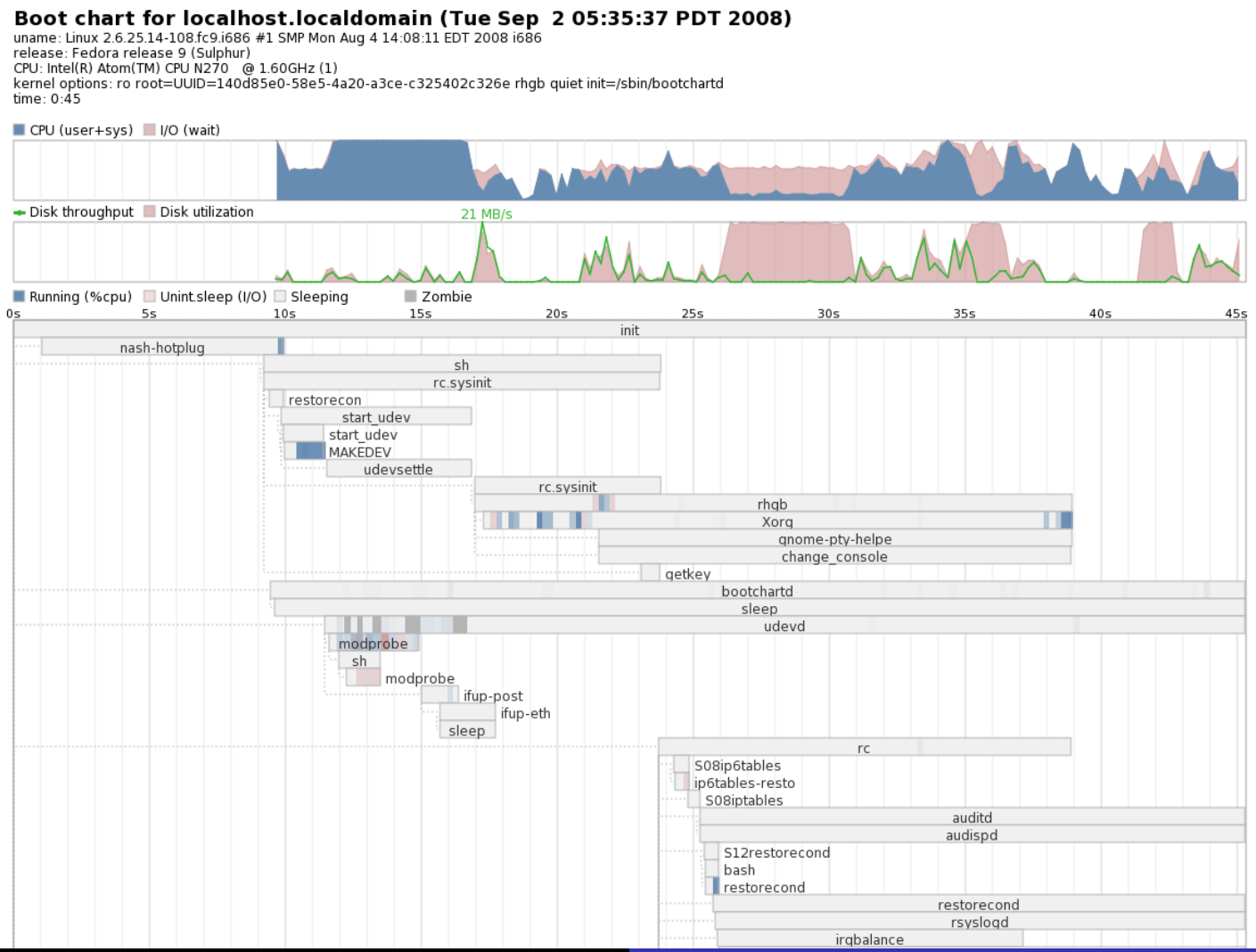
- Estos **ficheros** son **enlaces simbólicos** al fichero con el mismo nombre **descriptivo** que está en el directorio **/etc/init.d** (o /etc/rc.d/init.d).

Ficheros de Inicialización III

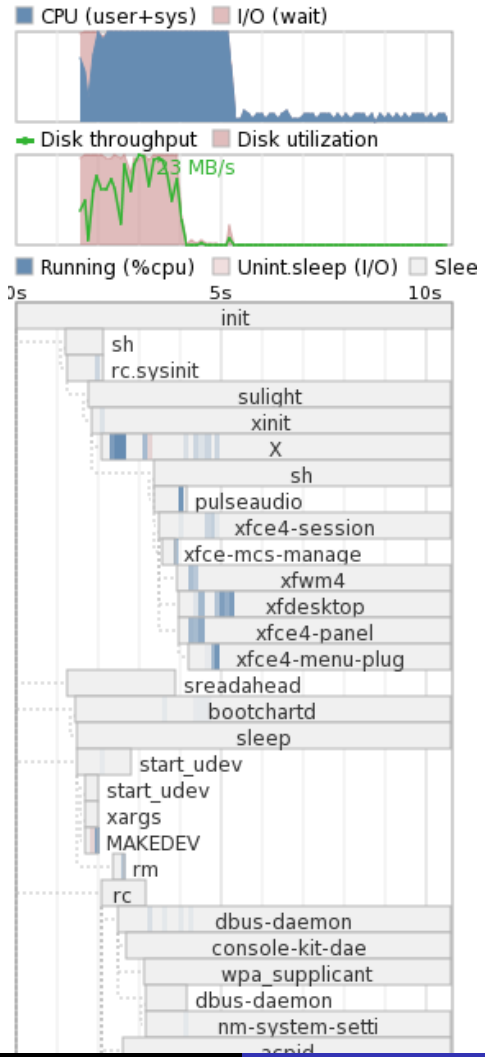
- Los ficheros **scripts** reciben varios **parámetros**: start, stop, restart, etc. (esto permite **lanzar** o **relanzar** demonios sin **reiniciar** el **sistema**) rc ejecuta los K con el parámetro stop y los S con start.
- Existen **herramientas** que permiten **manipular** el orden del **arranque** de los trabajos, antes **chkconfig**, ahora update-rc.d.
- También se pueden lanzar **demonios** manualmente:

```
/etc/rc.d/init.d/cups restart
```

Intentos de optimización en el arranque



Intentos de optimización en el arranque



PARADA DEL SISTEMA.

Parada del Sistema I

- En **ocasiones** es necesario **apagar** o **reiniciar** el sistema: mantenimiento, **diagnóstico**, hardware nuevo, etc.
- Las **acciones** que se realizan en el proceso de **parada** son:
  1. Se **notifica** a los **usuarios**.
  2. A los **procesos** en ejecución: se les envía la **señal** de terminación (**TERM**).
  3. Se **paran** los **demonios**.
  4. A los **usuarios** que quedan **conectados** se les echa del **sistema**.
  5. A los **procesos** que queden en **ejecución** se les envía la señal de fin (**KILL**).
  6. **Actualizaciones** de disco **pendientes** (integridad del SF) con **sync**.

Parada del Sistema II

7. **Dependiendo** del tipo de **shutdown**: se cambia a modo **monousuario**, se apaga el **ordenador** o se reinicia el **sistema**.

- **shutdown** [opciones] **tiempo** [mensaje]
  - Sin opciones: modo **monousuario** (telinit 1).
  - -r: **reiniciar** (telinit 6)
  - -h: parar (telinit 0)
  - -c: **cancelar**
  - -k: **simular** un **shutdown**, pero realmente no se lleva a cabo
  - Tiempo: +**minutos**, now, h:m

Caídas del sistema y problemas de arranque I

Causas de caídas del sistema:

- **Fallos** hardware.
- Errores de hardware **irrecuperables**.
- Fallos de luz (cortes o altibajos).
- Otros problemas **ambientales**.
- **Problemas** de entrada/**salida**.
- Problemas de algún **sistema** de **ficheros**.

Caídas del sistema y problemas de arranque II

Problemas de arranque

- **Hardware** mal instalado.
- **No** se puede leer el sistema de **ficheros** de los discos de **trabajo**.
- Hay en el disco áreas **dañadas** que no pertenecen al sistema de ficheros (p.e. tabla de **particiones**).
- Hardware incompatible.
- **Errores** en la **configuración** del sistema.

Caídas del sistema y problemas de arranque III

- La orden **dmesg** permite visualizar los **mensajes** producidos durante el **arranque**.
- En el **arranque** al núcleo se le pueden pasar **otros** parámetros:
  - **root=particion**: indicar que monte como partición raíz una distinta.
  - **init=ejecutable**: que en vez del proceso Init lance otro proceso.
  - **single**: arrancar en modo monousuario.
  - Un **número** indicando el **nivel** de arranque.

Bibliografía I

1. Windows 7 : Working with Boot Configuration Data. http://programming4.us/desktop/2490.aspx
2. Securing the Windows 8 Boot Process. http://technet.microsoft.com/en-us/windows/dn168167.aspx
3. Suspensión y apagado en W8. http://blogs.msdn.com/b/b8/archive/2011/09/08/delivering-fast-boot-times-in-wi aspx
4. Modificar suspensión y apagado en Windows 8. http://www.xatakawindows.com/bienvenidoawindows8/como-modificar-la

Bibliografía II

5. Arranque en Windows 10 https://msdn.microsoft.com/en-us/library/windows/hardware/dn756626%28v=vs.85%29.aspx
6. Booteo en cinco segundos. http://www.taringa.net/posts/linux/1608709/Logran-bootear-linux-en-5-segundos.html
7. Herramientas alternativas para configuración del arranque en Linux. http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m7/index.html

Tema 4. Proceso de Arranque y Parada

Sistemas Operativos II

Escuela Superior de Informática de Ciudad Real

Universidad de Castilla-La Mancha

Tema 5. Gestión de la memoria virtual.

Sistemas Operativos II

Universidad de Castilla-La Mancha

Índice

Objetivos

- **Entender** qué es la Memoria **Virtual** y por qué se **utiliza**.
- **Comprender** qué es la **propiedad** de **localidad** y su relación con el **comportamiento** de un programa en **ejecución**.
- **Conocer** los distintos **algoritmos** de **sustitución**.
- **Conocer** la **teoría** del **conjunto** de **trabajo** y el problema de la **hiperpaginación**.

Índice

Jerarquía de Memoria I

- **Dos principios sobre memoria**: Menor **cantidad**, acceso más rápido; Mayor cantidad, menor **coste** por byte.
- Los elementos frecuentemente accedidos se ponen en memoria rápida, **cara y pequeña**; el resto, en memoria lenta, **grande y barata**.

Jerarquía de Memoria II

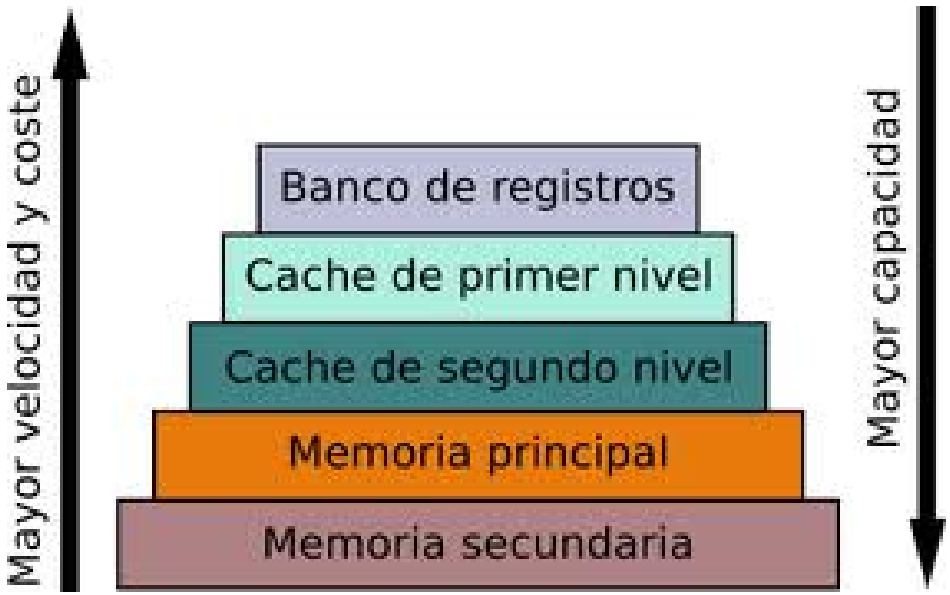


Figura : Jerarquía de Memoria

Jerarquía de Memoria III

Dedicated L1
- Locality keeps most critical data in the L1 cache
- Lowest latency
- 2 loads per cycle

Dedicated L2
- Sized to accommodate the majority of working sets today
- Dedicated to eliminate conflicts common in shared caches
  - Better for Virtualization

Shared L3 – NEW
- Victim-cache architecture maximizes efficiency of cache hierarchy
- Fills from L3 leave likely shared lines in the L3
- Sharing-aware replacement policy
- Ready for expansion at the right time for customers

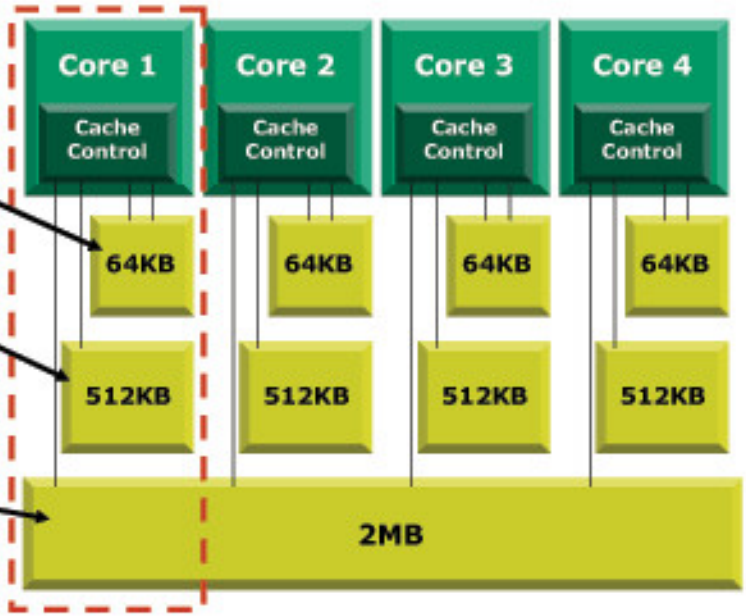


Figura : Niveles de Caché

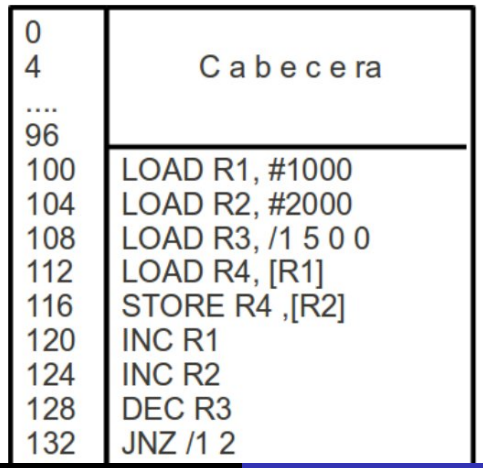
Objetivos a tener en cuenta

- **Conocer** las distintas **organizaciones** de la **memoria**.
- **Gestión**: dado un **esquema** de organización qué estrategias de **asignación**, **substitución** y **búsqueda** se deben seguir para obtener el rendimiento óptimo.
- Establecer **mecanismos** de **protección** del SSOO con los procesos de usuario y de los **procesos** de usuario entre ellos.

Espacio de direcciones lógico y físico

- **Espacio de direcciones lógico**: conjunto de direcciones **lógicas** o **virtuales** generadas por un programa (CPU).
- **Espacio de direcciones físico**: conjunto de direcciones **físicas correspondientes** a las direcciones **lógicas** en un instante dado (Memoria).

Figura : Fichero Ejecutable



Carga absoluta y reubicación

- **Carga absoluta**: **Asignar** direcciones **físicas** al **programa** en tiempo de compilación.
- **Reubicación**: **Capacidad** de **cargar** y **ejecutar** un programa en un lugar **arbitrario** de la memoria.
- La **reubicación** puede ser **estática**, cuando la decisión de dónde ubicar el programa se hace en **tiempo de carga**, o **dinámica**, cuando la traducción de direcciones lógicas a físicas se hace en **tiempo de ejecución**.

Índice

- La **asignación** puede ser **contigua** o **no contigua**.
- **Contigua**: el almacenamiento de un programa se hace en un único bloque de **posiciones continuas** de memoria. Las particiones pueden ser **fijas** (fragmentación interna) o **variables** (fragmentación externa). **Técnicas de asignación de espacio** para particiones variables son: primer ajuste, mejor ajuste y peor ajuste.
- **No contigua**: El programa se divide en bloques que pueden ser colocados en **zonas no continuas** de memoria principal. Se usan técnicas como la **paginación**, la **segmentación** y la **segmentación paginada**.

Monoprogramación y multiprogramación

- **Monoprogramados**: memoria principal dividida en dos partes: **SSOO residente** y **proceso de usuario**. La asignación contigua en sistemas monousuario implementa **protección** a través del **registro** de **relocalización** (o límite).
- **Multiprogramados**: existen **varios** procesos en **memoria** y el tamaño máximo de cada proceso es el tamaño de la **memoria** física **disponible**.

Multiprogramación

- El esquema más sencillo, es **dividir** la **memoria** en regiones o **particiones** cada una de las cuales puede ser ocupada por **un proceso**.
- Existen **dos formas** de dividir la memoria en **particiones**: particiones **fijas** o estáticas y particiones variables o **dinámicas**.
- La **protección** se obtiene a través de los **registros base** y **límite** (los valores para un proceso se guardan en su **PCB**).

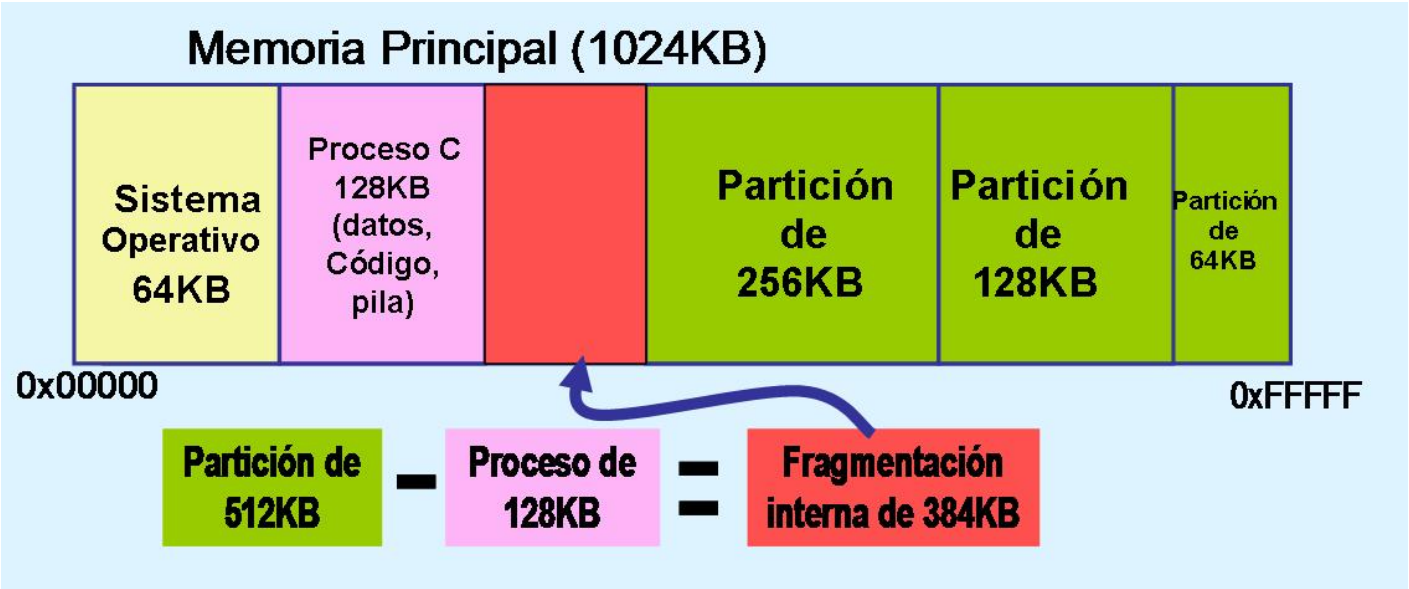
Particiones Fijas

- El **número** y **tamaño** de cada **partición** viene establecido por el sistema y es **constante**.
- Las **particiones** pueden tener **igual** tamaño o pueden ser de **distinto tamaño**.
- Estrategias de **asignación**: **proceso** - **partición** se pueden gestionar mediante una **cola** única o **varias** colas (particiones de distinto tamaño).

Problema de las Particiones Fijas

- **Uso** de la **memoria** principal **ineficiente**. Cualquier **proceso**, sin importar lo pequeño que sea, **ocupará** una partición **completa**
- **Fragmentación**: **incapacidad** del sistema operativo para **asignar** posiciones de memoria principal **no utilizadas**.
- Dos tipos: **interna** y **externa**.
- Problema de fragmentación en **particiones fijas**: fragmentación **interna**.

Fragmentación Interna



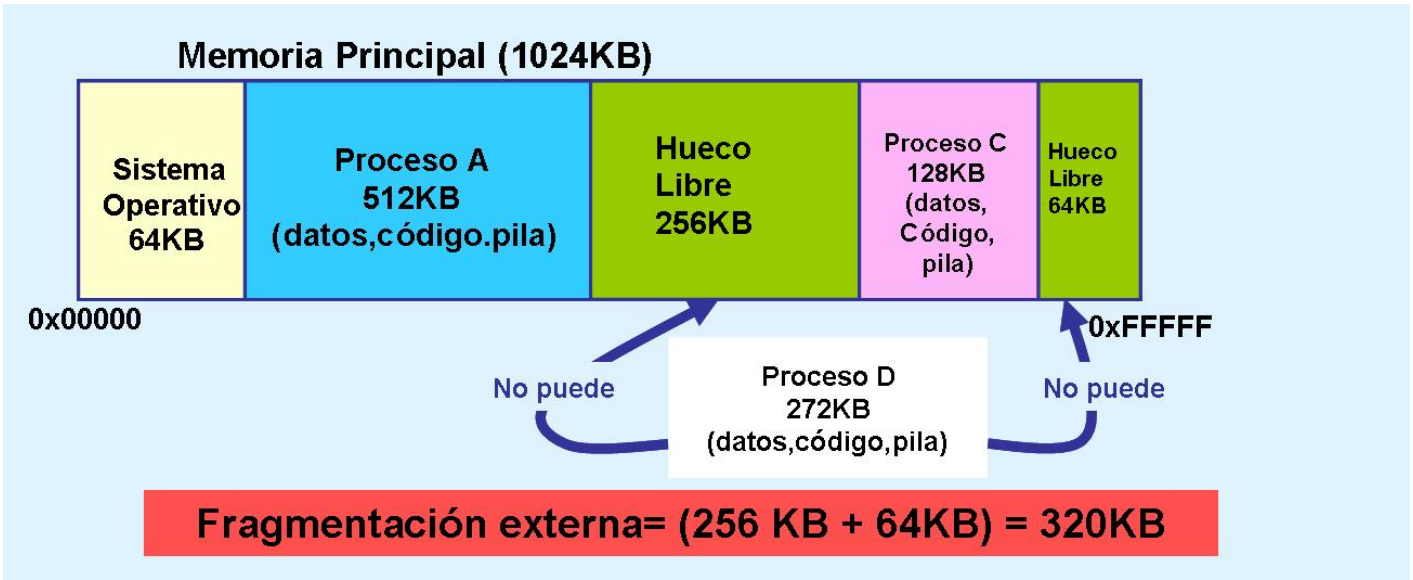
Particiones Variables

- Las particiones son **variables** en **número** y **tamaño**.
- Cuando llega un **proceso** a memoria se le **asigna** la memoria que necesita: se crea una **partición**.
- El **SSOO** mantiene información sobre las zonas de memoria asignadas (**tabla de particiones**) y las libres (**huecos**).
- Es necesario tener una **estrategia de asignación de espacio** y **gestión de espacio libre**.

Problema uso de Particiones Variables

- **Fragmentación externa**: **existe** el **espacio** necesario para satisfacer una petición pero **no** es **contiguo**.
- **Compactación**: técnica utilizada para reducir la fragmentación externa. Consiste en arrastrar los contenidos de memoria a un lugar para **reunir** toda la **memoria libre** en un bloque.
- **Problemas**: requiere **reubicación** dinámica, **consume recursos** del sistema y el **sistema** se **detiene** mientras se realiza.

Fragmentación Externa



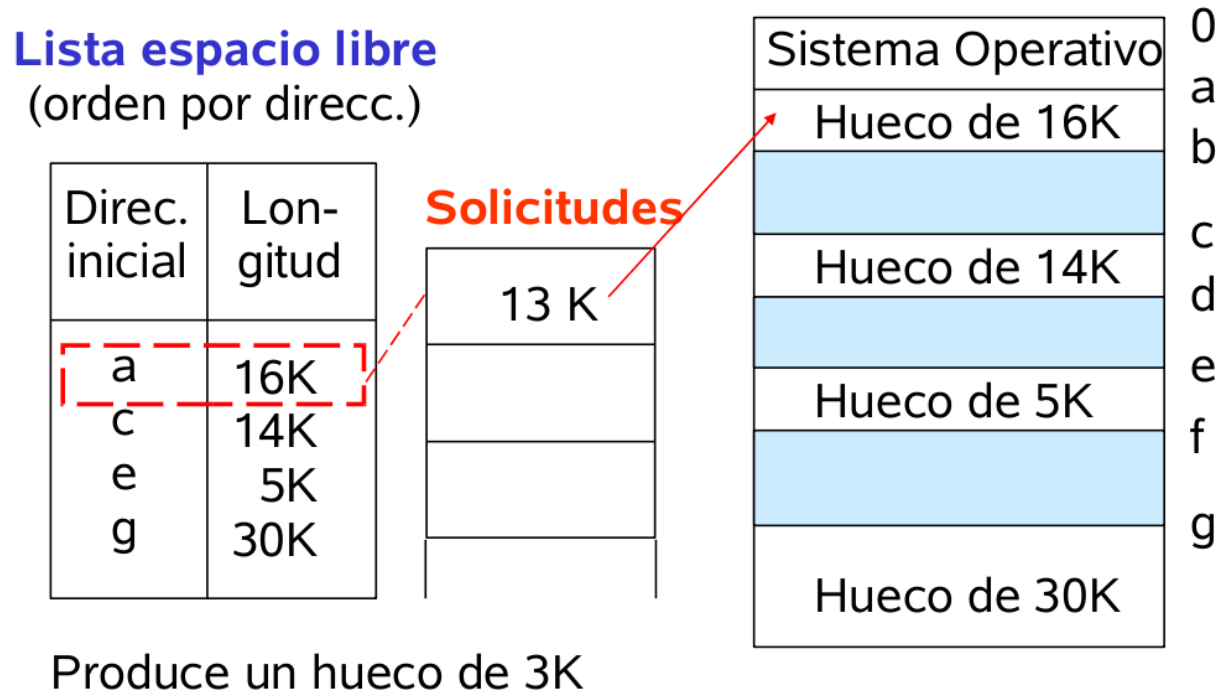
Estrategias de asignación de espacio

- Tratan de responder a la cuestión de cómo **satisfacer** una **petición** de **tamaño n** desde una lista de huecos.

Las estrategias son:

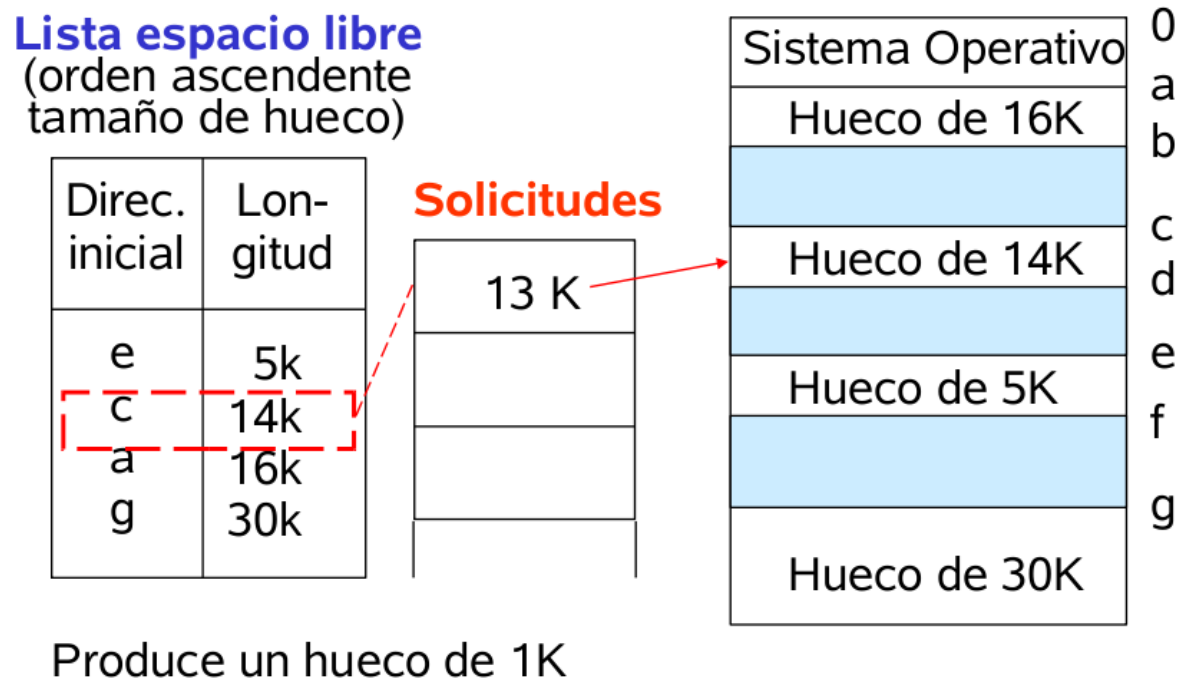
- **Primer ajuste**: asigna el **primer hueco** lo suficientemente grande para satisfacer la petición
- **Mejor ajuste**: asigna el **hueco más pequeño** que mejor se ajuste al espacio necesitado. Se debe buscar en la **lista entera**, si no está ordenada por tamaños. Produce el hueco sobrante menor.
- **Peor ajuste:** asigna el **hueco mayor**. Debemos buscar en **toda la lista**. Produce el hueco sobrante mayor

Primer Ajuste

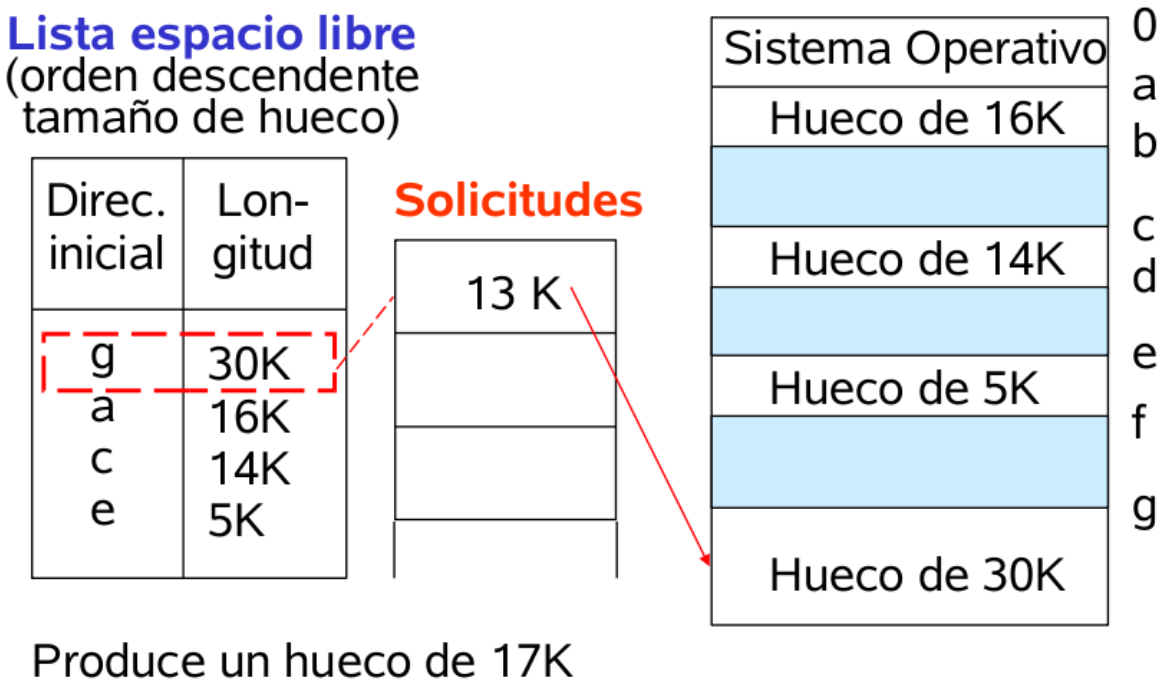


Produce un hueco de 3K

Mejor Ajuste



Peor Ajuste



Produce un hueco de 17K

Gestión de espacio libre

- **Mapas de bits: divide** la memoria principal en **unidades** de **asignación** y se utiliza un bit por cada una de ellas que indica si está libre (0) u ocupada (1)
- **Lista enlazada**: **P**roceso o **H**ueco, dirección de **inicio** y **tamaño**.
- Con respecto a la **protección de memoria** de cada proceso se necesita saber la posición **inicio** y **final** de cada **partición**.

Gestión de espacio libre

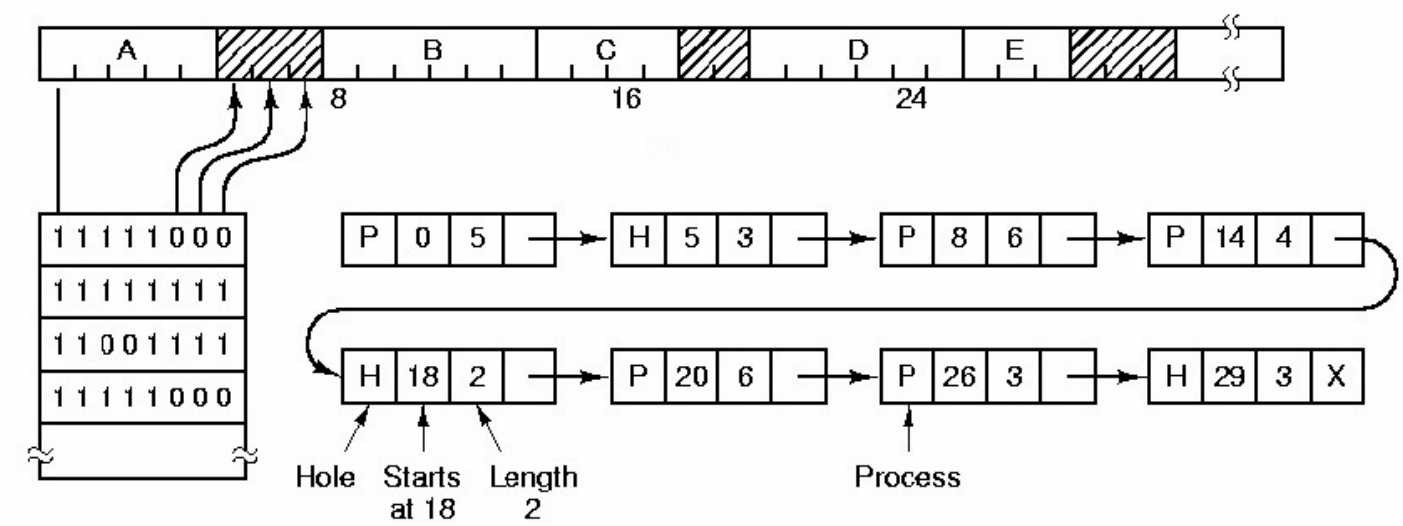


Figura : Mapa de Bits y Lista Enlazada

Índice

- **Overlay**: Mecanismo que permite ejecutar **programas más grandes** que la **memoria principal**. Esto se consigue teniendo en memoria las instrucciones y datos que se **necesitan** en un **instante concreto**. La dificultad viene de que el SSOO no da soporte y toda la **responsabilidad** de establecer la estructura de overlay es del **programador**.
- **Swapping**: Intercambio de procesos entre memoria y memoria masiva. El **disco** debe **albergar** las **imágenes** de memoria de los procesos. Un factor crítico es el **tiempo de transferencia**. El intercambio puede ser **dinámico** o **estático**.

El **intercambiador** tiene las siguientes **responsabilidades**:

1. Seleccionar procesos para **retirarlos** de MP.
2. Seleccionar procesos para **incorporarlos** a MP.
3. **Gestionar** y **asignar** el espacio de intercambio.

Localización del espacio de intercambio

- **Intercambio Dinámico**: un **único archivo** de intercambio global del sistema cubre las **necesidades** de intercambio de **todos** los **procesos** (Problema: elección de su tamaño)
- **Intercambio Estático**: existe un **archivo** de intercambio **dedicado** por cada proceso intercambiable del sistema. No impone límites al número de procesos intercambiados. **Problema**: necesita **más espacio** en **disco**, los **accesos** son más **lentos** y los **direccionamientos** son más **complicados**.

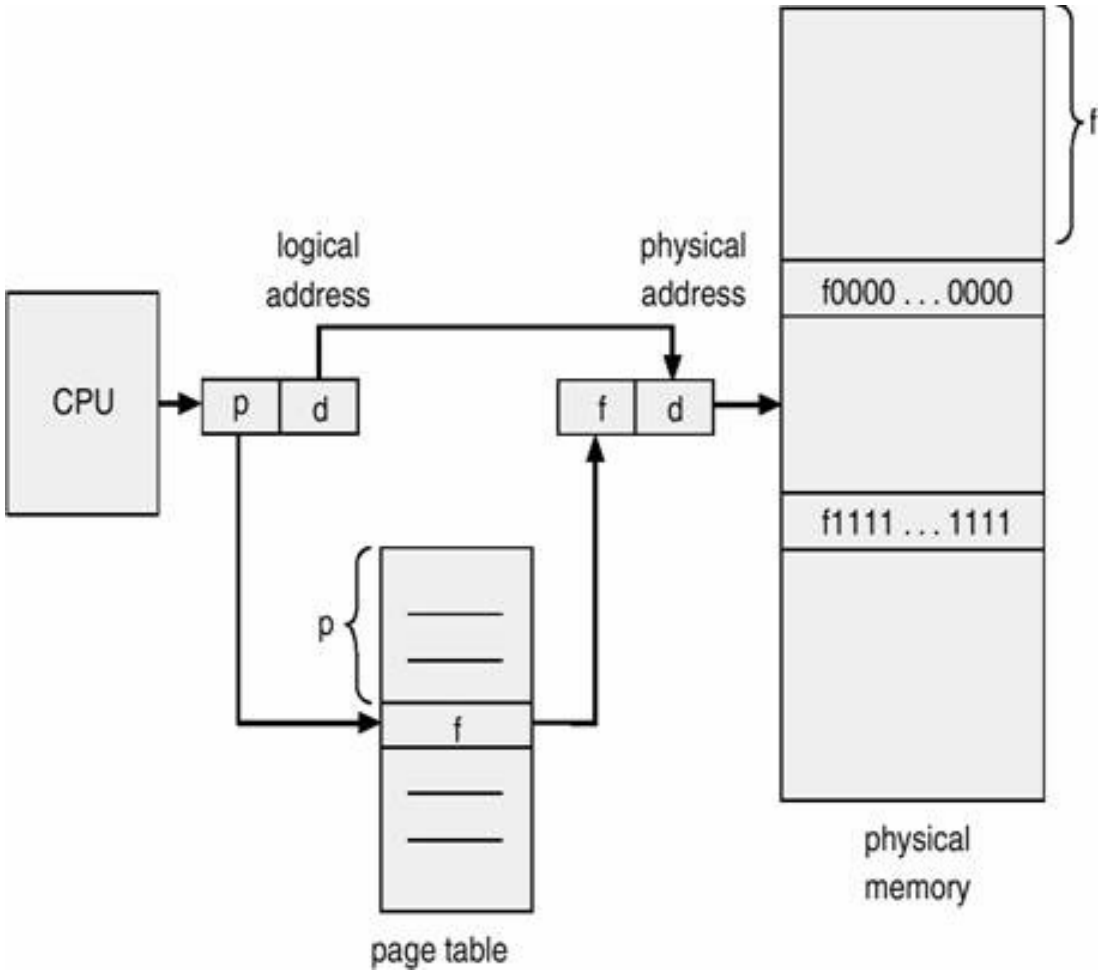
Índice

- La **paginación** es un **esquema de gestión de memoria** que permite que el **espacio de direcciones físicas** de un proceso **no sea contiguo**.
- La memoria física se **divide** en bloques de **tamaño fijo**, denominados **marcos de página**. El tamaño es **potencia de dos**, de 512 bytes a 16 MB (normalmente entre 4KB y 8KB).
- El **espacio lógico** de un proceso se divide en **bloques del mismo tamaño**, denominados **páginas**.
- Los **marcos** de páginas **contendrán páginas** de los procesos

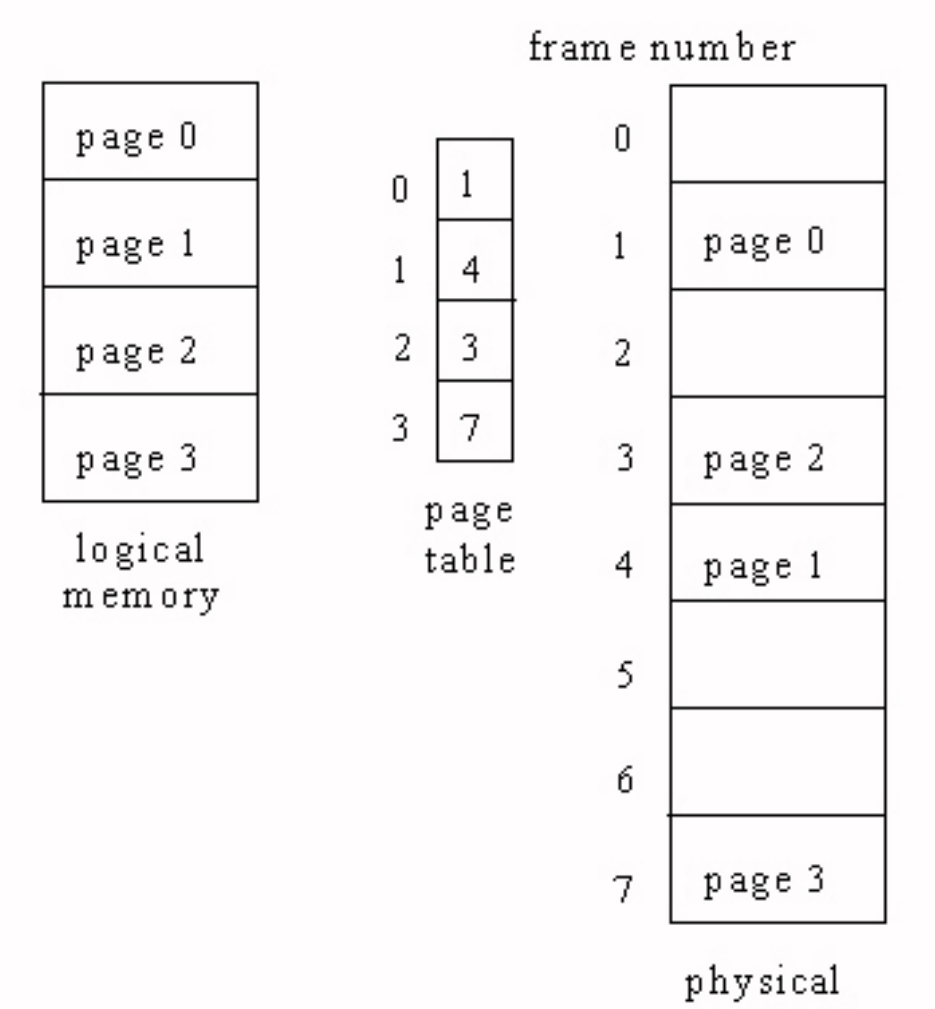
Direcciones lógicas y físicas

- Las **direcciones lógicas**, que son las que genera la **CPU** se dividen en **número de página (p)** y **desplazamiento dentro de la página (d)**: **[p, d]**
- Las **direcciones físicas** se dividen en **número de marco** (**m**, dirección base del marco donde está almacenada la página) y **desplazamiento** (d): **[m,d]**

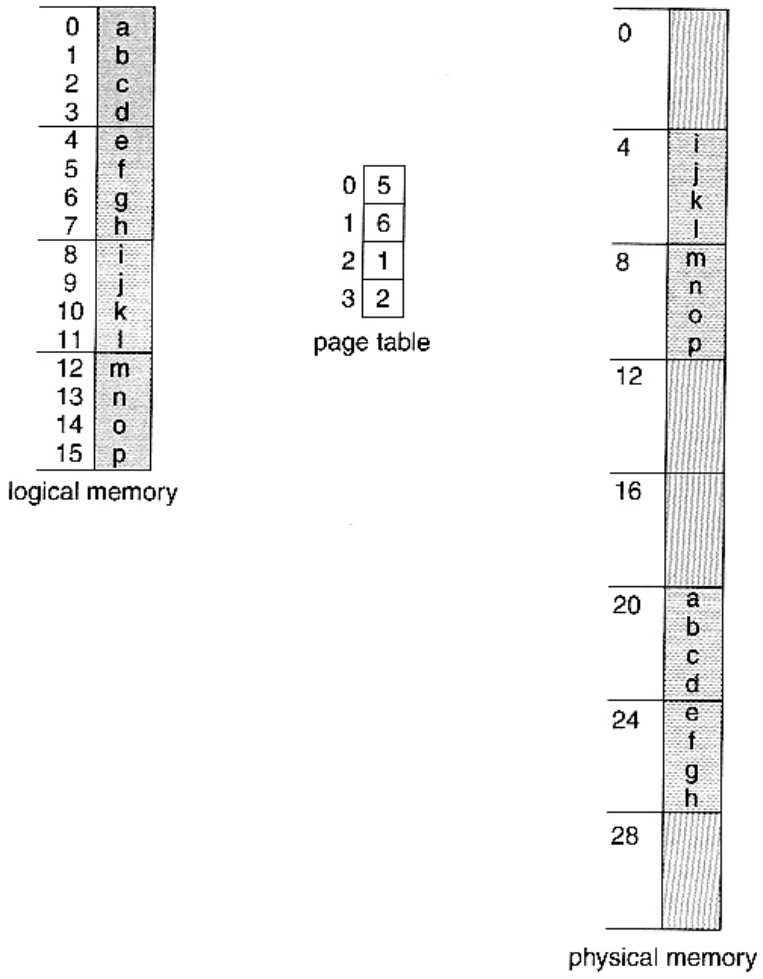
Hardware de Paginación



Modelo de paginación de la mem. lógica y física.



Ejemplo de paginación. (Pag. 4 bytes, Mem. 32 bytes).



Tablas de marcos.

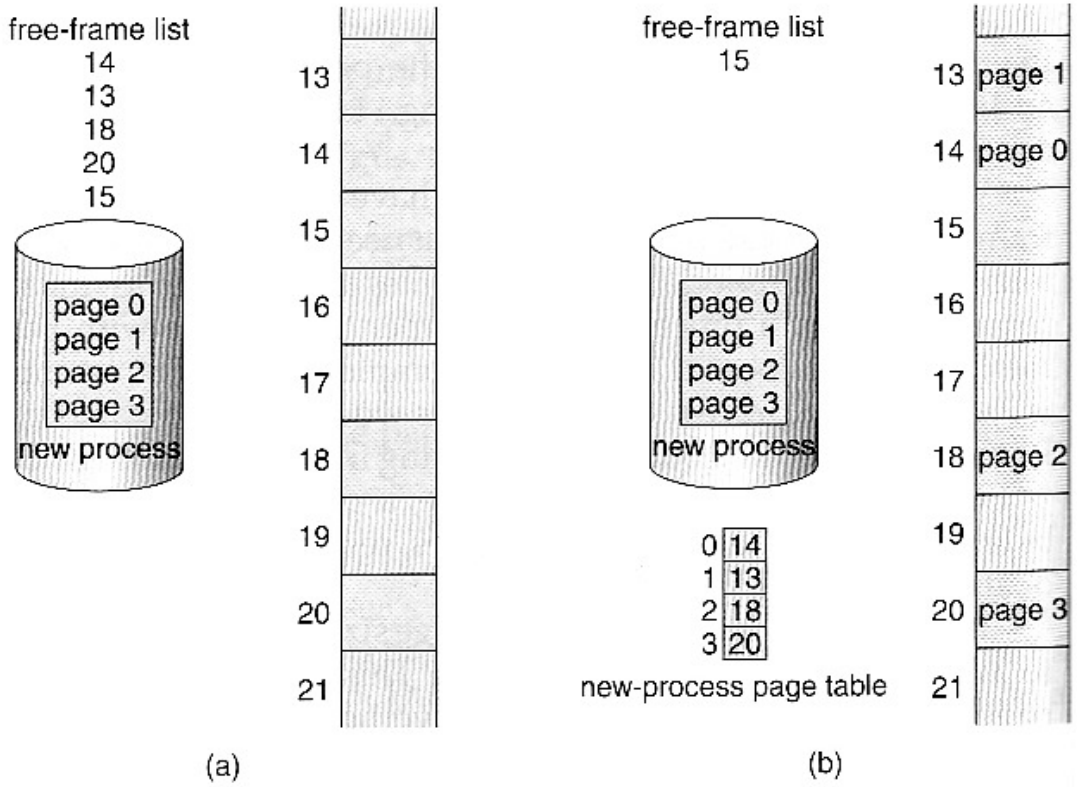


Figura : (a): antes de la asignación. (b): después de la asignación

Índice

Índice

- El tamaño del **programa**, los **datos** y la **pila** puede exceder la cantidad de memoria física disponible para él.
- Se usa un almacenamiento a dos niveles:
  1. **Memoria Principal**: partes del proceso necesarias en un momento dado
  2. **Memoria Secundaria**: espacio de direcciones completo del proceso
- Hay que establecer **mecanismos** para **conocer** qué se **encuentra** en **memoria** principal y una **política** de **movimientos** entre Mem. principal y Mem. secundaria.

En muchos casos **no** es **necesario** tener el programa **completo** en memoria para poder **ejecutarlo**:

- Código para **tratamiento** de **errores**.
- A las **matrices**, **listas** y **tablas** se les suele asignar **más** memoria de la que realmente **necesitan**.
- Ciertas opciones y **características** de un programa pueden llegar a usarse en muy **raras** ocasiones.

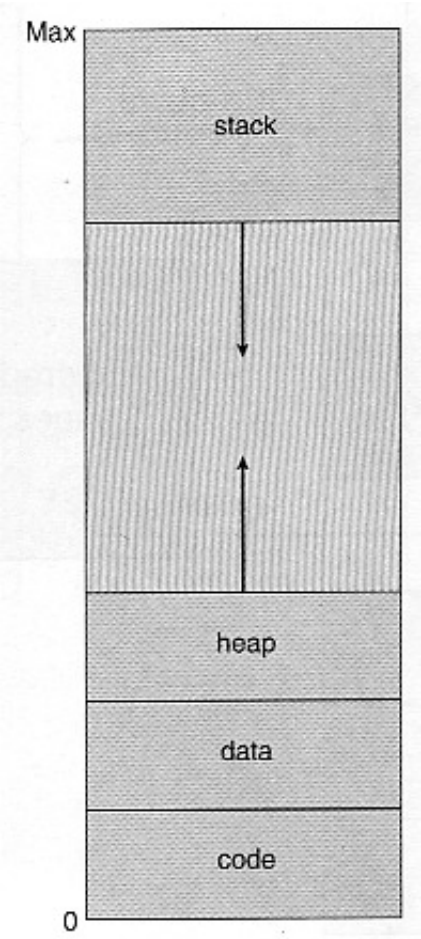
La **posibilidad** que existe de **ejecutar** programas que están **parcialmente** en memoria aporta que:

- **No** existe **limitación** en el tamaño de los programas.
- Se **mejora** la tasa de **utilización** del **procesador** al poder tener cargados más programas en memoria (parcialmente).
- La memoria virtual **facilita** la tarea de **programación** enormemente.

Espacio de direcciones virtual

- Hace referencia a la forma **lógica** de **almacenar** un **proceso** en memoria.
- La **dirección** lógica de **inicio** es la 0.
- El almacenamiento es **contiguo**.
- Se contempla la asignación **dinámica** de memoria (**cúmulo** y **pila**). Espacios de direcciones **dispersos**.

Espacio de direcciones virtual



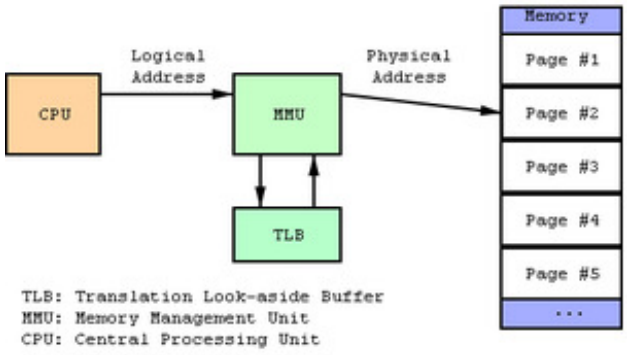
Unidad de Gestión de Memoria (MMU) I

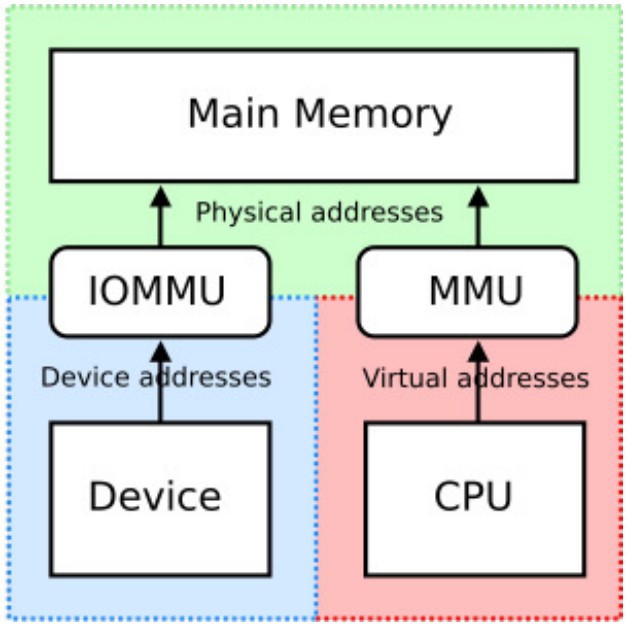
- La MMU (**Memory Management Unit**) es un dispositivo **hardware** que **traduce** direcciones virtuales a direcciones físicas.
- Este dispositivo está **gestionado** por el **SSOO**.
- En el esquema MMU más simple, el valor del **registro base** se **añade** a cada dirección generada por el proceso de usuario al mismo tiempo que es enviado a memoria.

Unidad de Gestión de Memoria (MMU) II

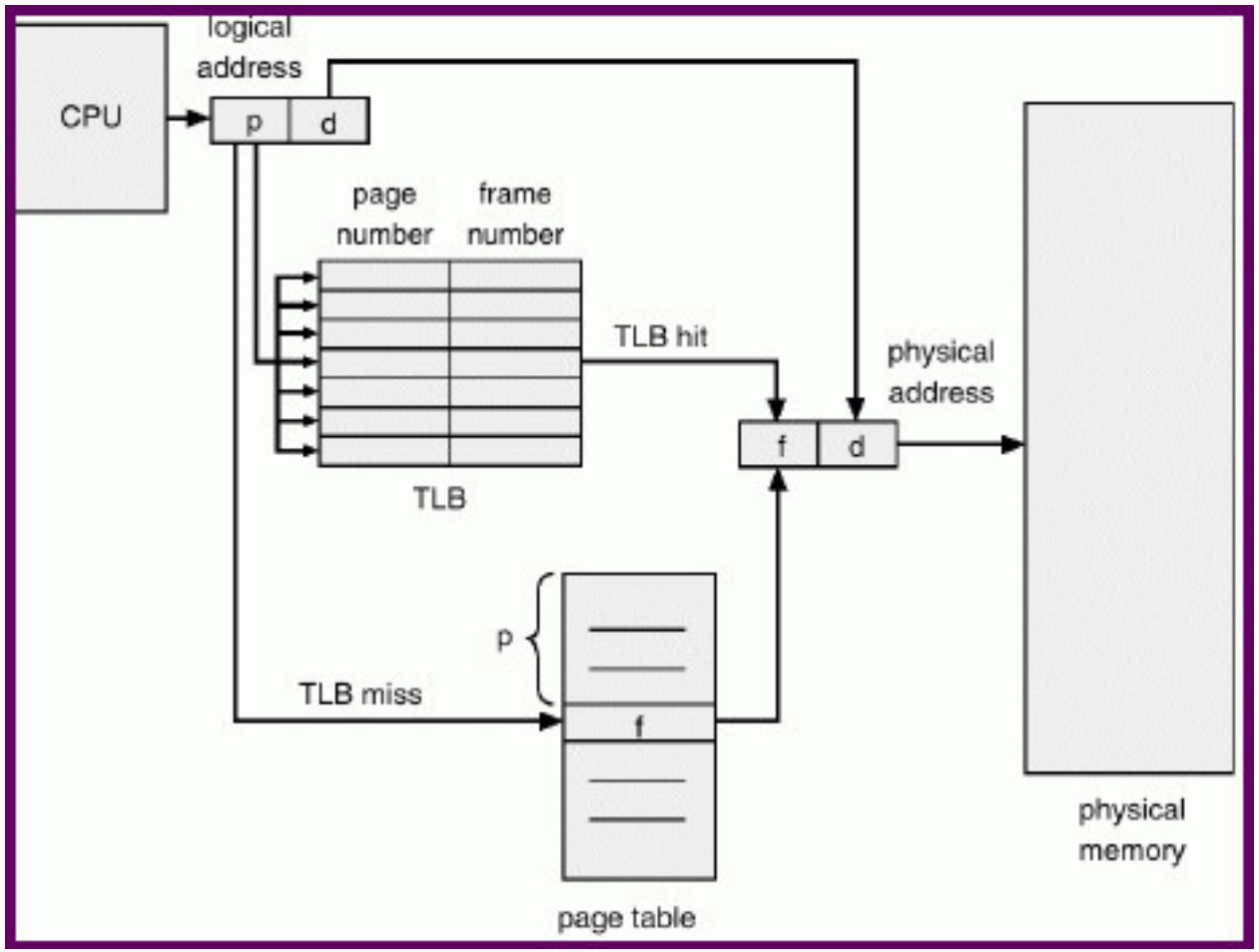
- El programa de **usuario** trata **sólo** con direcciones **lógicas**; éste nunca ve direcciones reales.
- Además de la traducción, el MMU deberá **detectar** si la dirección aludida se **encuentra** o no en Mem. **principal**. Si no está generará una interrupción (si se encuentra en Mem. secundaria).

Unidad de Gestión de Memoria (MMU) III





Unidad de Gestión de Memoria (MMU) IV



Índice

Carga de un programa en memoria

- Un **programa** se podría cargar **completo** en memoria.
- **No óptimo** en ciertas situaciones: **usuario** selecciona una **opción**.
- La **paginación por demanda**: sólo carga páginas cuando sean **necesarias**.
- Se usa **comúnmente** en los sistemas de memoria virtual.
- La paginación por demanda pura contempla **no cargar** nunca una **página** en **memoria** hasta que sea requerida.

Paginación por demanda frente a anticipada

- En **paginación por demanda**, no se debe transferir ninguna página del almacenamiento secundario al principal hasta que un proceso en ejecución haga **explícitamente referencia a ella**.
- En la **paginación anticipada**, el sistema operativo intenta **predecir las páginas que necesitará un proceso** y entonces carga dichas páginas cuando hay espacio disponible.

Paginación por demanda frente a anticipada

- Los resultados de la teoría de la computabilidad, específicamente el **problema de la detención**, indican que **no** se puede **predecir con precisión** la trayectoria de ejecución que seguirá un programa (carga páginas equivocadas).
- La paginación por demanda **garantiza** que las **únicas páginas** que se transfieren a memoria son aquéllas que **requieren** los **procesos**.
- Los **procesos** de **estimación** en paginación anticipada pueden ser muy **costosos** temporalmente.

Paginador (perezoso) vs Intercambiador

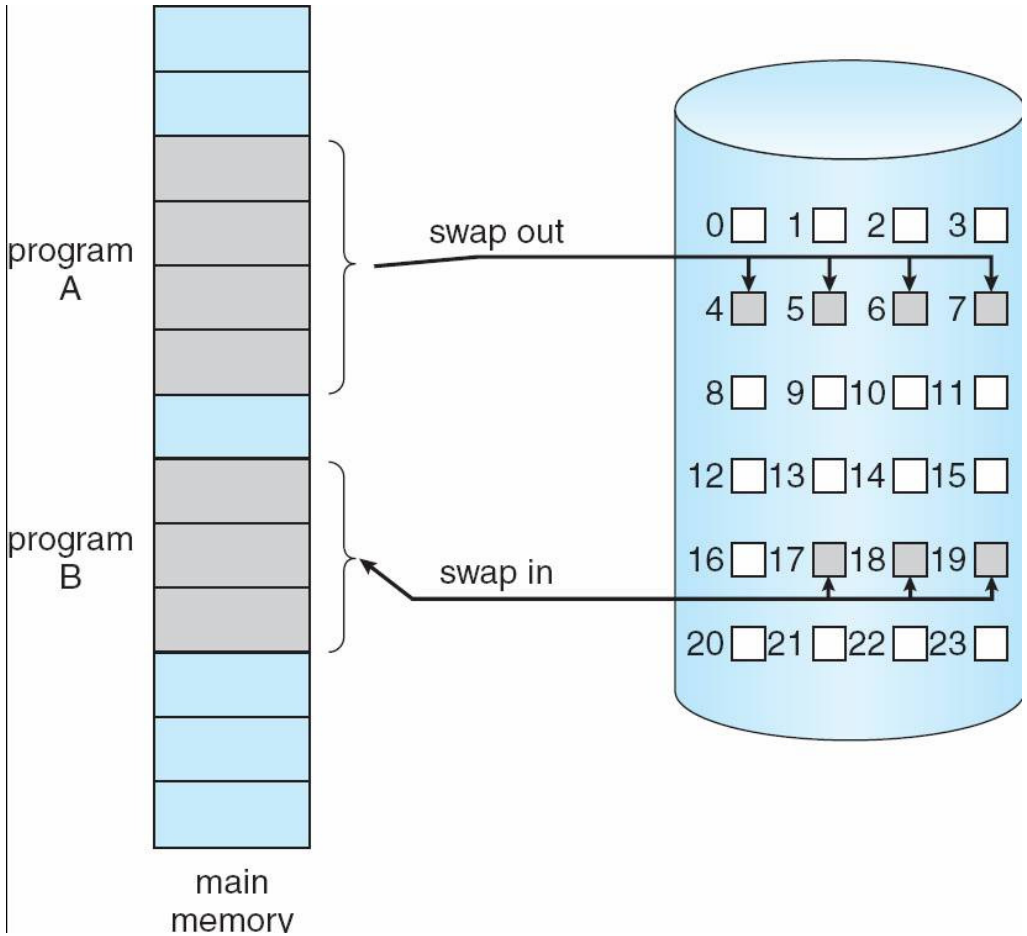
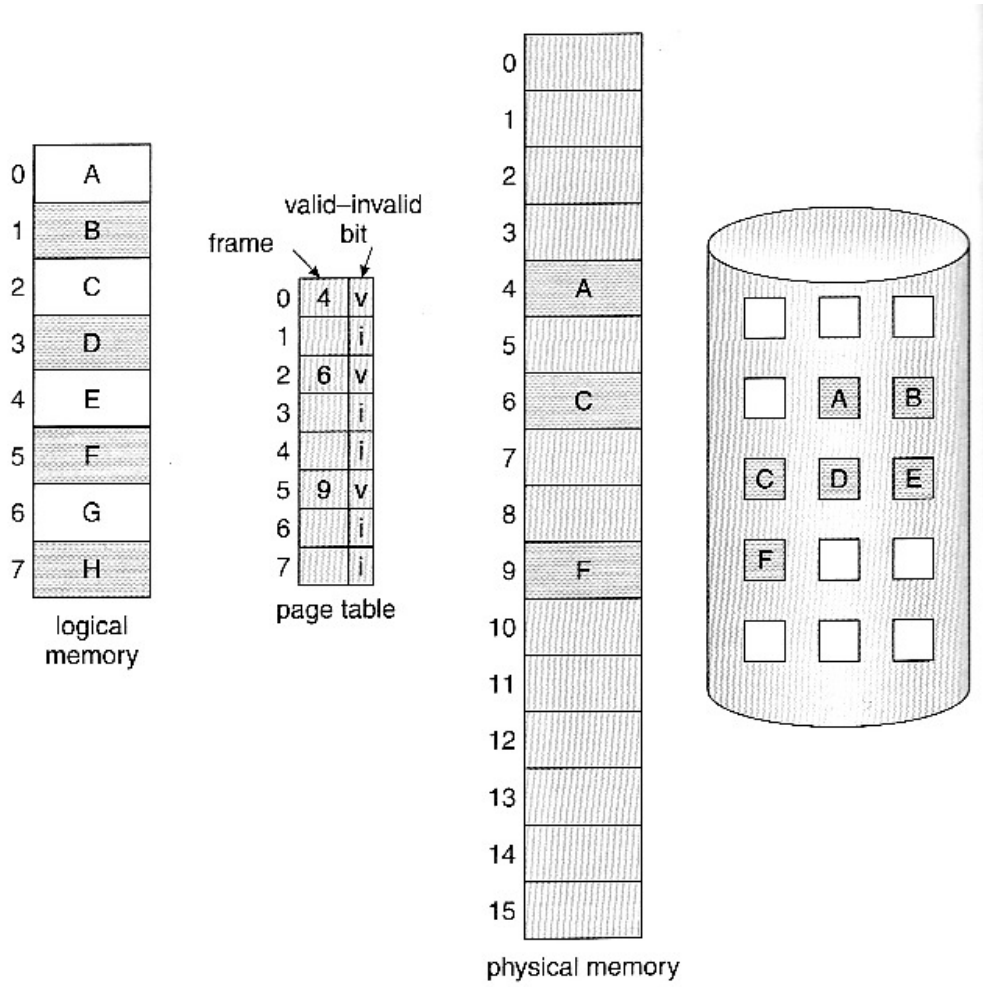


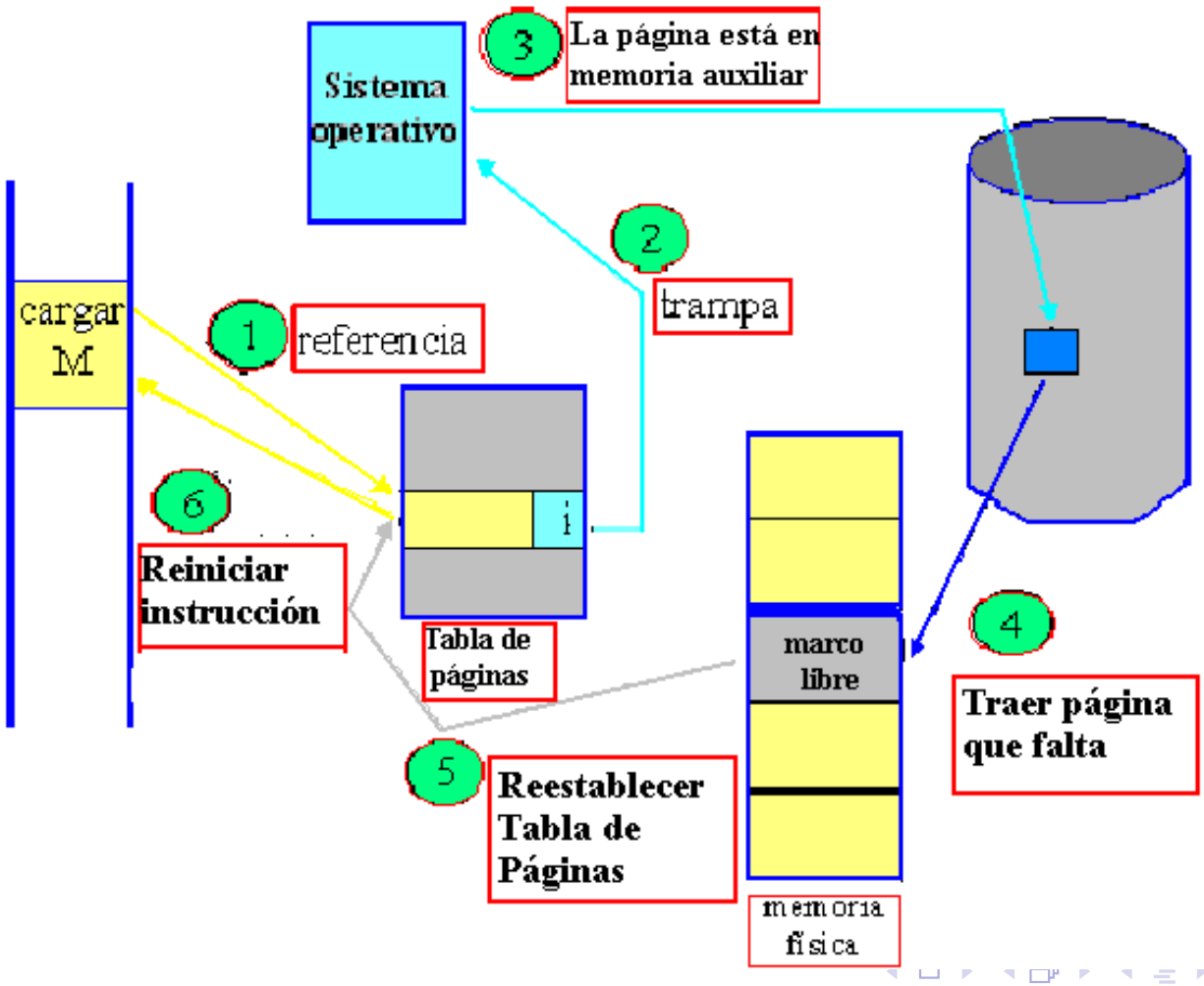
Tabla de Páginas (bit de validez)



Fallo de página

- Que haya una **página** en la tabla de páginas marcada como **inválida no** tiene **consecuencias** si el proceso **no intenta** nunca **acceder** a dicha página.
- El **acceso a páginas** residentes en **memoria** ejecuta el proceso con **normalidad**.
- El **acceso** a una **página** marcada como **inválida** provoca una **interrupción de fallo de página**.

Fallo de página



Fallo de página

- El acceso a **nuevas** páginas de manera continua **degradaría** el rendimiento del sistema.
- Pero los programas tienden a la llamada "**localidad de referencia**".
- Esto hace que las **prestaciones** que pueden obtenerse sean **razonables**.
- Hardware: **tabla** de **páginas** modificada + memoria **secundaria** (espacio de **intercambio**).

Reinicio de una instrucción

- El **estado** del proceso **interrumpido** está **guardado**: registro, código de condición y contador de instrucciones.
- Se debe **reiniciar** el proceso en el **mismo lugar** y en el mismo **estado**.
- Si el **fallo** es por una **instrucción** se lee y **decodifica** de nuevo dicha instrucción.
- Si el **fallo** es por un **operando**: se lee de nuevo la **instrucción** y se vuelve a leer el **operando**.

Reinicio de una instrucción

- **C = ADD(A ,B)**
  1. Extraer y decodificar instrucción.
  2. Extraer **A**.
  3. Extraer **B**.
  4. Sumar **A** y **B**.
  5. Almacenar la suma en **C**. (Situación más **desfavorable**).

Rendimiento

- Este tipo de paginación puede **afectar** de manera significativa el **rendimiento** de un sistema informático.
- Como medida se usa el **tiempo de acceso efectivo**.
- El **tiempo de acceso a memoria** (*ma*), está entre 10 y 200ns.
- Si **no** hay **fallos** de página: *tiempo acceso efectivo* $\leftarrow$ *ma*.

Rendimiento: tiempo de fallo de página I

- Si p es la **probabilidad** de que exista un **fallo** de **página**, $p \in [0,1]$ y además p es próximo a cero.
- $\textit{tiempo acceso efectivo} \leftarrow (1-p) \cdot ma + p \cdot \textit{tiempo fallo pagina}$.
- Quedaría **calcular** el **tiempo** de fallo de página.

Rendimiento: tiempo de fallo de página II

1. **Acceso** a la **tabla** de **páginas** donde se observa que el **bit** de **validez** indica que no está en memoria.
2. **Guardar** los **registros** del **usuario** y el estado del proceso.
3. Determinar que la **interrupción** fue un fallo de página (vector de interrupciones).
4. Verificar que la **referencia** a la página fue **válida**.
5. Encontrar un **marco libre** y planificar una operación de **disco** para leer la página deseada y colocarla en el marco recién asignado.
6. Durante la espera, **asignar** la **CPU** a algún otro usuario.
7. **Interrupción** del **disco** (E/S terminada).

Rendimiento: tiempo de fallo de página III

8. **Guardar** los **registros** y el estado de proceso del otro usuario (si se ejecutó el paso 6).
9. Determinar que la **interrupción** provino del **disco**.
10. **Corregir** la **tabla** interna que se guarda junto con el **proceso** y la tabla de páginas, de modo que indiquen que la **página** ya está en **memoria**.
11. **Esperar** que la **CPU** se asigne otra vez a este proceso.
12. **Restaurar** los **registros** de usuario, el estado del proceso y la nueva tabla de páginas, y **reanudar** la **instrucción** interrumpida.

Rendimiento: tiempo de fallo de página IV

- Son **tres** los **componentes** principales del tiempo de servicio del fallo de página:
  1. **Servir** la **interrupción** de fallo de página ([1, 100] microsegundos).
  2. **Leer** la **página** *(8ms = 3 (latencia m.) + 5 (búsqueda) + 0.05 (transf.))*
  3. **Reiniciar** el **proceso** ([1, 100] microsegundos).
- Se puede suponer un tiempo de *8ms* (¿Un sólo proceso?)

Rendimiento: tiempo de fallo de página V

- *tiempo acceso efectivo* $\leftarrow (1-p)\cdot 200 + p\cdot 8000000 = 200 + 7999800\cdot p$.
- El **tiempo de acceso efectivo** es directamente **proporcional** a la **tasa de fallos de página**. Suponiendo un **fallo** de página **cada 1000** accesos tenemos:
- *tiempo acceso efectivo* $\leftarrow$ $200 + 7999800\cdot 0{,}001 \leftarrow 200 + 7999,8 = 8{,}2$ microsegundos
- Ese sería el **tiempo de acceso** si hay un fallo cada 1000 accesos.
- La computadora se **ralentiza** en un **factor** de **40** (4100 %).

Rendimiento: tiempo de fallo de página VI

- Si se quiere sólo que la **degradación** sea como **máximo** de un **10 %**.
- Sólo se permitiría **un fallo** de página **cada 399.990** accesos a memoria.

$$220 > 200 + 7{,}999{,}800 \cdot p,$$

$$20 > 7{,}999{,}800 \cdot p,$$

$$p < 0{,}0000025.$$

Espacio de Intercambio

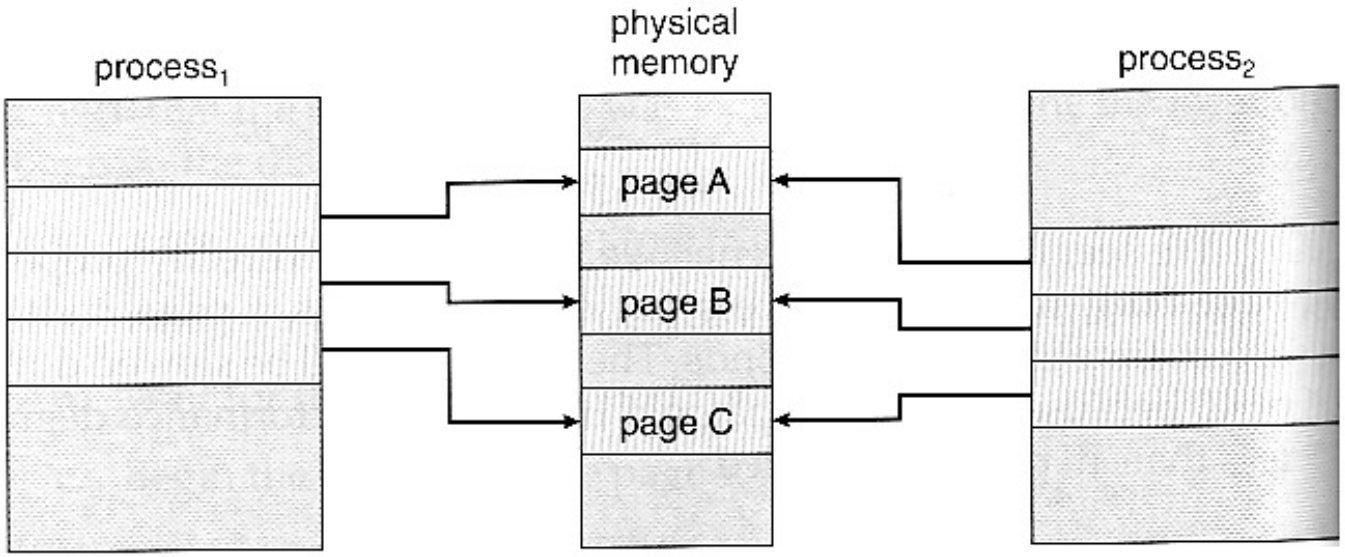
- Las operaciones de **E/S** dirigidas al **espacio de intercambio** son, en general, más **rápidas** que las operaciones sobre el **sistema de archivos**.
- Esto es debido a que se usan **tamaños** de **bloque** mayores y **no** se utilizan mecanismos de **búsqueda ni** de asignación **indirecta**.
- Se puede **obtener mayor** tasa de **transferencia** si se dispone de la **imagen** completa de un **archivo**, nada más iniciar el proceso y luego usar paginación por demanda.
- Otra **opción** es acceder la **primera** vez al sistemas de **archivos**, pero una vez se substituye una página, almacenar ésta en el **espacio** de **intercambio**. Así **sólo** se **leen** del sistema de archivos las páginas **necesarias**.

Índice

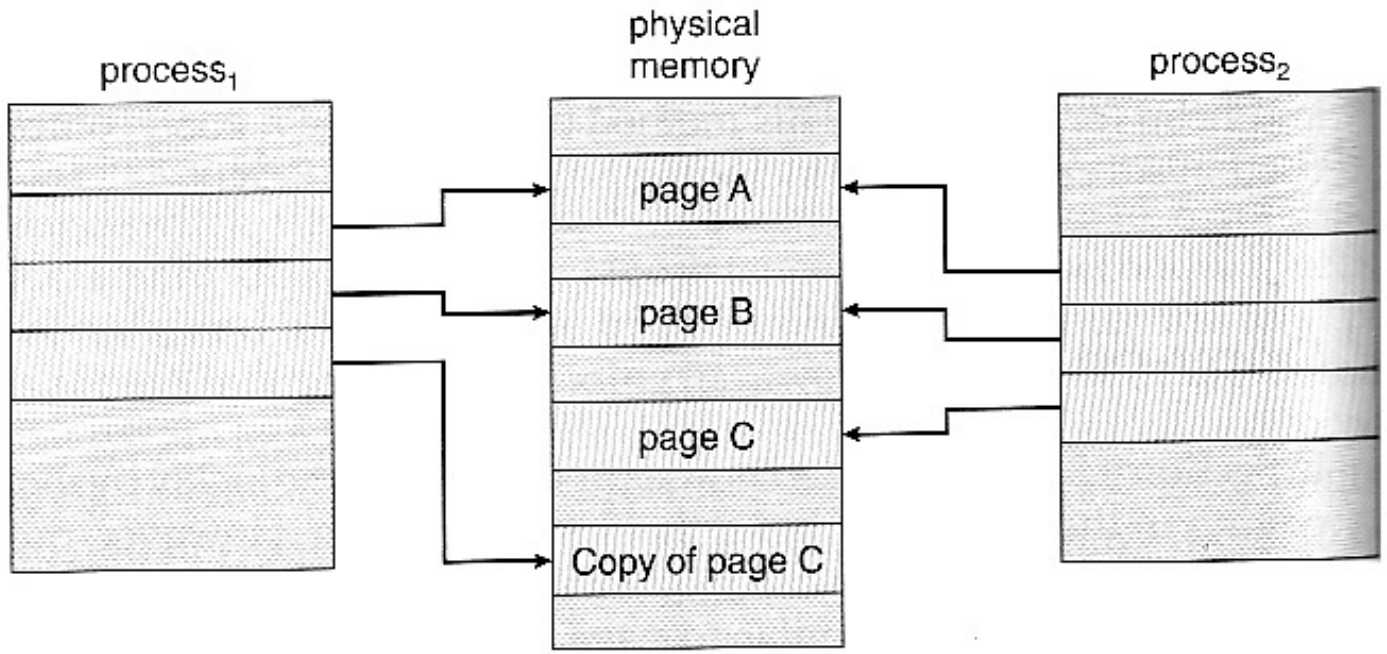
Creación de procesos

- La orden *fork()* **duplica** las **páginas** que pertenecen al **padre**.
- Muchos procesos **hijos** invocan la llamada *exec()* de manera **inmediata** a su creación. Esto haría **innecesario** esa **copia**.
- Como **alternativa** se utiliza una técnica llamada **copia durante la escritura**. Permite que **padres** e **hijos** inicialmente **compartan** las mismas páginas.
- Esto suele **afectar** a las páginas de **datos** y **no** a las de código **fuente**.

Antes de que proc. 1 modifique pág. C



Después de que proc. 1 modifique pág. C



Conjunto compartido de marcos libres + vfork

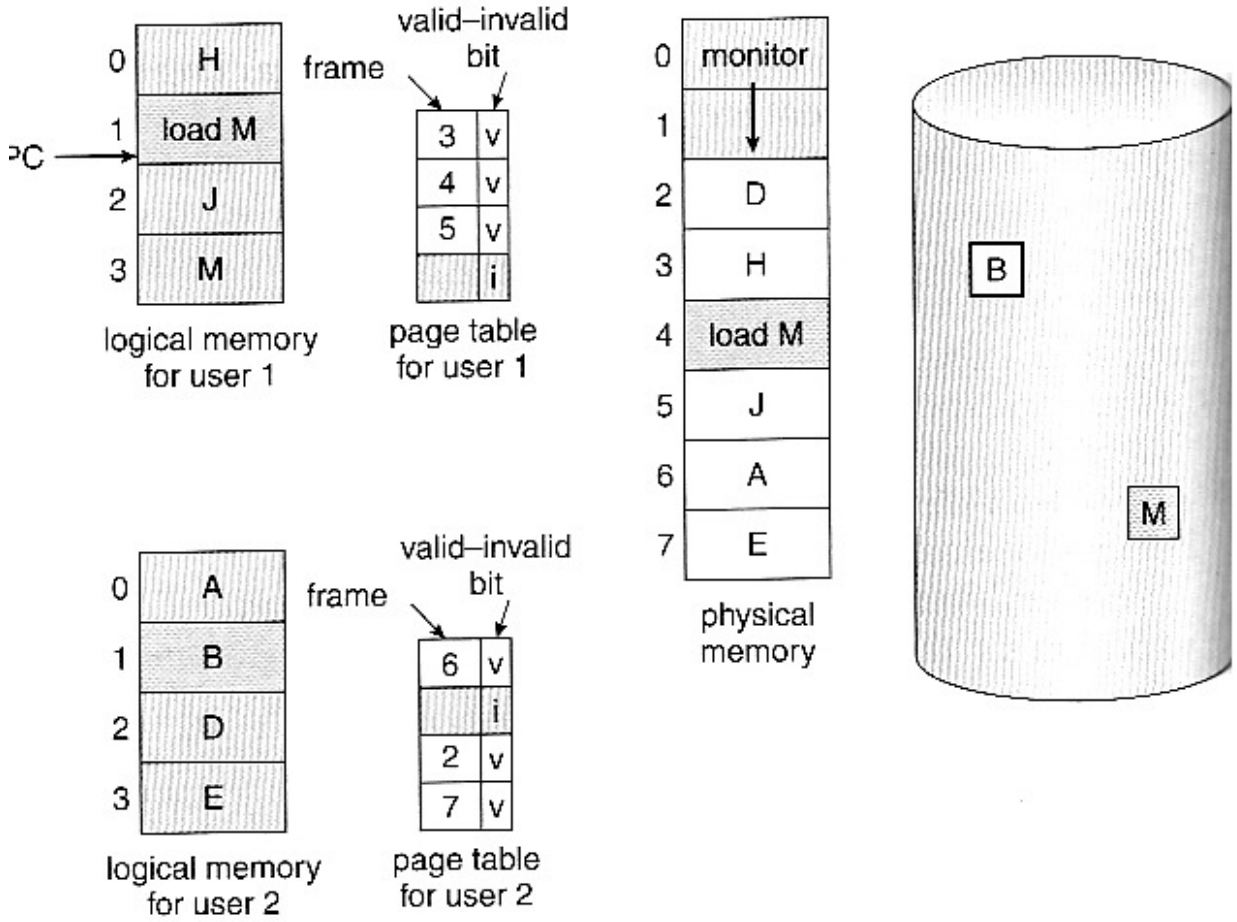
- Cabe preguntarse **donde** se **ubican** las páginas **duplicadas**.
- Algunos **SSOO** proporcionan el llamado **conjunto compartido de marcos libres**, para copia durante la escritura o bien cuando el proceso (pila o cúmulo) debe expandirse.
- Estas **páginas** se **asignan** mediante una técnica llamada **relleno de ceros bajo demanda**.
- *vfork()* **suspende** al proceso **padre** y el **hijo** utiliza sus páginas de memoria. **No** usa **copia durante la escritura**. ¿Qué ocurre cuando se reanuda el padre?

Índice

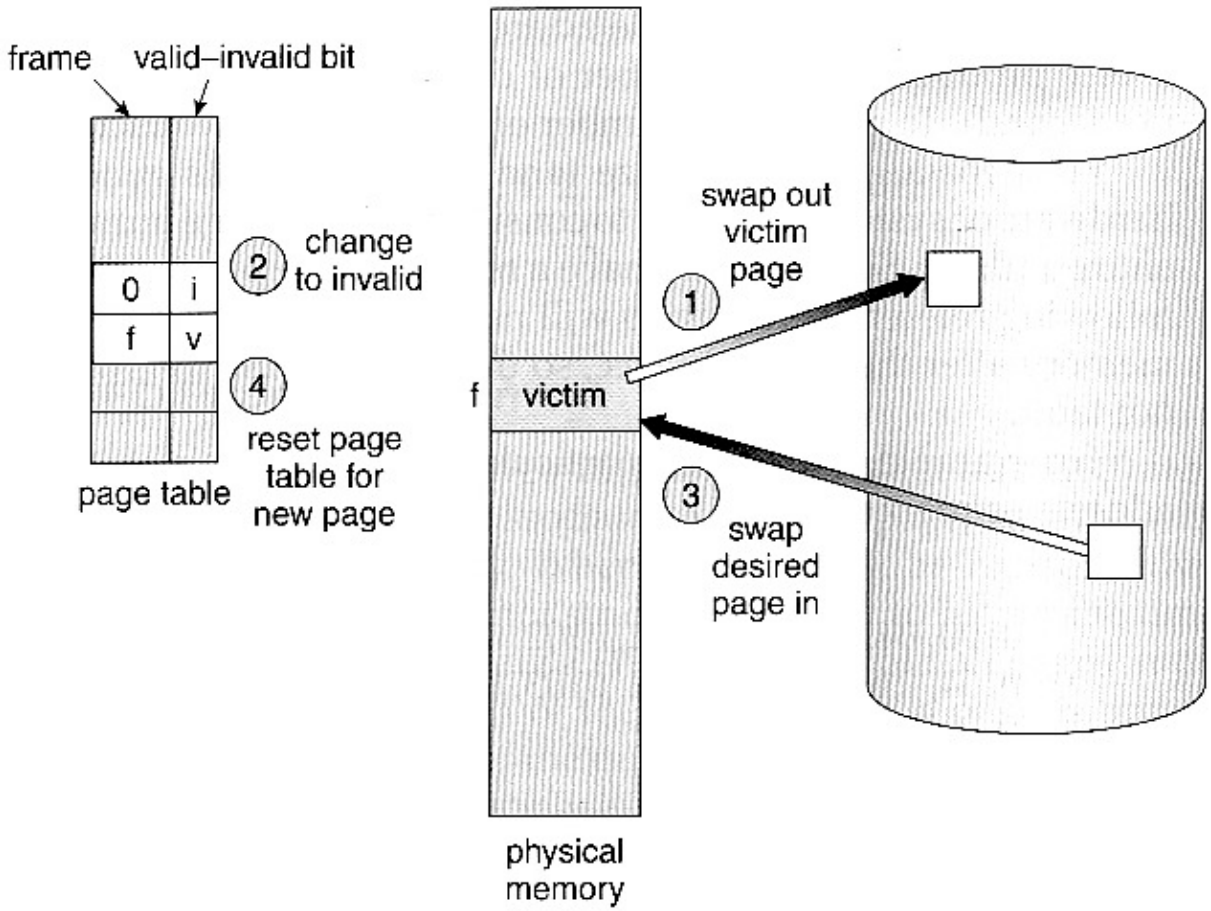
Sobreasignación de Memoria

- Disponemos de **procesos** que **necesitan 10** páginas, pero realmente sólo **necesitan** acceder a **5** de sus páginas.
- Esta situación permite **incrementar** el **grado** de **multiprogramación**: *Memoria = 40 marcos* $\leftarrow$ *8 procesos*.
- Este **incremento** del grado de **multiprogramación** conduce a lo que se conoce como **sobreasignación de memoria**. *6 procesos* $\leftarrow$ *30 marcos*. ¿Si todos necesitan cargar sus 10 páginas? *6 procesos* $\leftarrow$ *60 marcos* (40 disponibles). ¿Qué hacer?
- Además, la **memoria no** almacena **únicamente** direcciones de **procesos**, p.e., búferes de E/S,

Sobreasignación de memoria



Sustitución de páginas



Páginas Sucias

- Si **no** hay ningún **marco libre**, se necesitan **dos transferencias** a disco (**descarga** y **carga**).
- Esta situación **duplica el tiempo de servicio de fallo de página** e incrementa en la misma proporción el **tiempo efectivo de acceso**.
- Para **reducir** esta **carga**, se usa el **bit de modificación** o **bit sucio**.
- El **hardware** activa este **bit** cada vez que se **escribe** en una **palabra** de la página.
- Si **no** ha sido **modificada** la página, el proceso de **descarga no** es **necesario**. Tampoco lo es si la página es de **solo lectura**.

Asignación de marcos

- Hay que **decidir** cuantos **marcos** asignar a cada **proceso**.
- **Asignación por igual**: se asignan el **mismo** número de **marcos** a todos los **procesos**. Si hay *m* marcos, y *n* procesos. A cada proceso se se asignan *m/n marcos*.
- **Asignación proporcional por tamaño**: asigna según tamaño del proceso

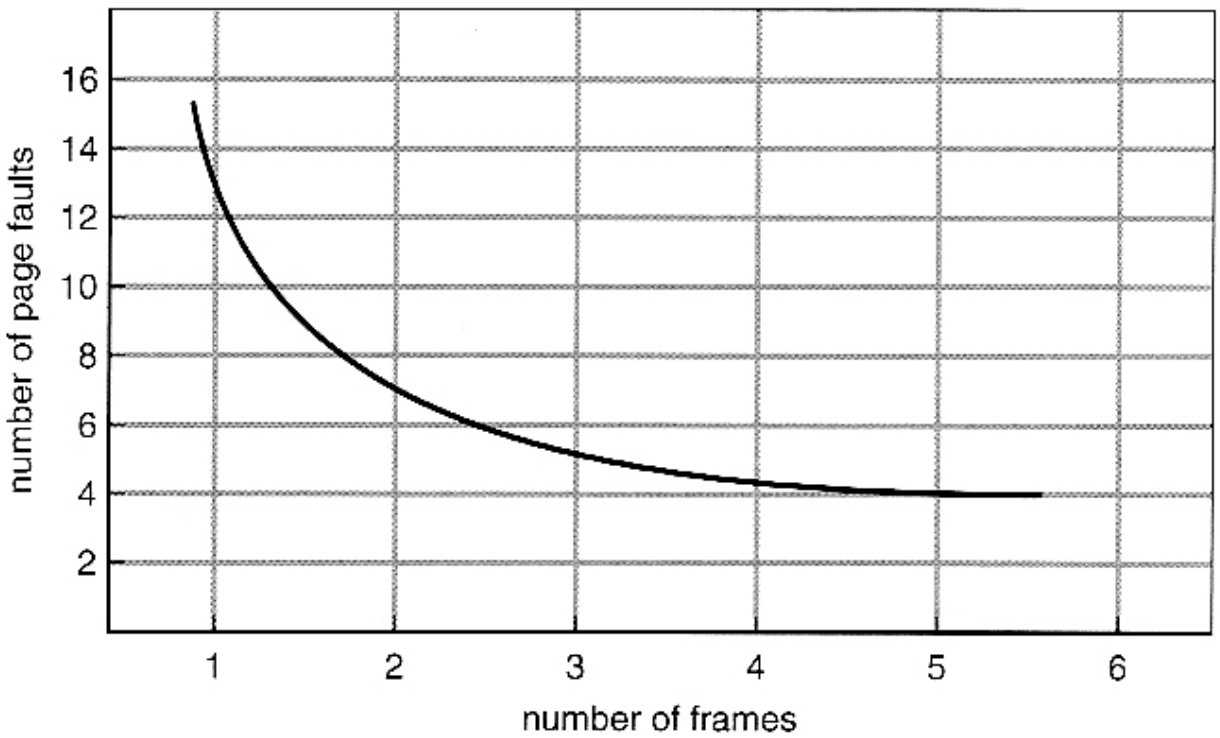
Algoritmos de sustitución de páginas

- Seleccionan que **marcos** hay que **sustituir**.
- Son **críticos**, ya que cualquier **pequeña mejora** en los métodos de paginación bajo demanda proporcionan un **gran beneficio** en términos de rendimiento del sistema.
- Existen **multitud** de **algoritmos** de este tipo. Cada **SSOO** puede tener el suyo **propio**.
- Criterio de selección: **tasa de fallos de página más baja**.

Evaluación de algoritmos de sustitución

- La **cadena de referencia** contiene una secuencia de **páginas** a **cargar** en memoria.
- Se pueden generar cadenas de referencia usando **generadores de números aleatorios**.
- También obteniendo la **traza de un sistema**. En 1 segundo, se requieren alrededor de un **millón** de **direcciones**.
- Para **conocer** el número de **fallos** de página tenemos que conocer el **número de marcos de memoria**.
- Cadena=(7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1). Número de marcos: 3.

Gráfico fallos página vs. número de marcos

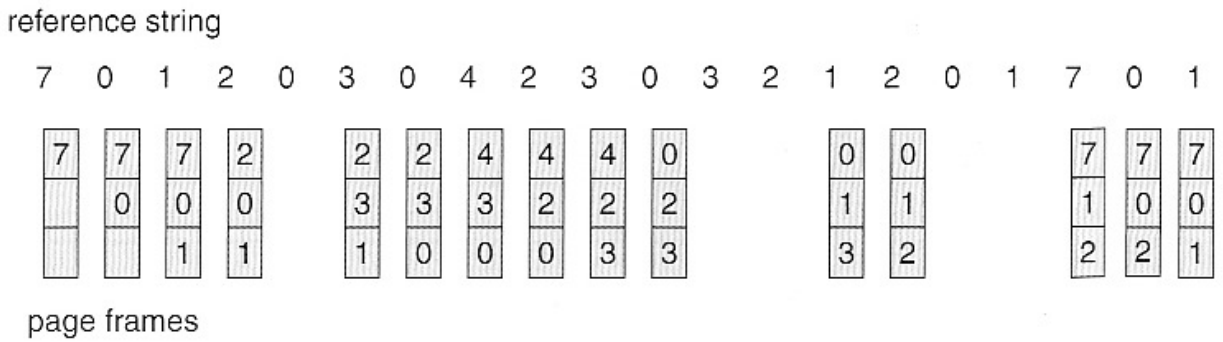


Sustitución de páginas FIFO

- Es el algoritmo **más simple**.
- La página **víctima** es la **más antigua**.
- No hace falta anotar explícitamente el instante de carga. Con **cola FIFO** simulamos su **funcionamiento**.

Sustitución de páginas FIFO

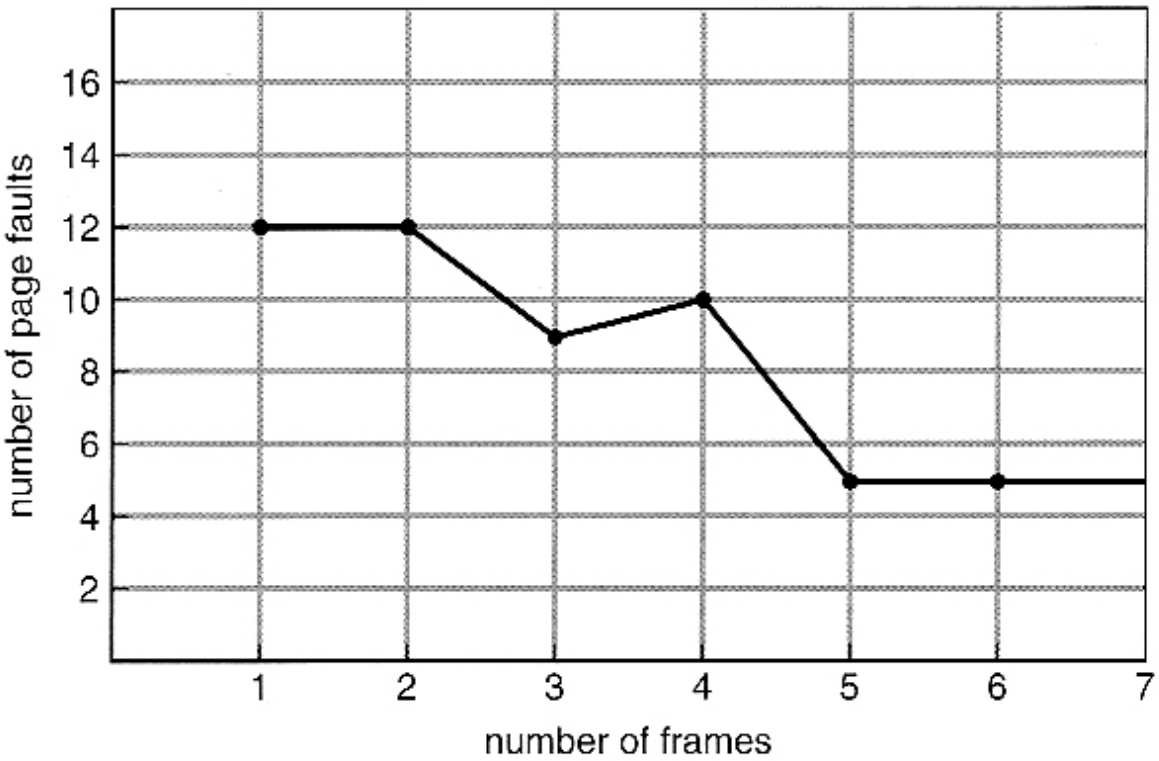
- Es el algoritmo **más simple**.
- La página **víctima** es la **más antigua**.
- No hace falta anotar explícitamente el instante de carga. Con **cola FIFO** simulamos su **funcionamiento**.



Anomalía de Belady

- Cadena=(1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5)
- Contar **fallos** de página para **3 marcos** y para **4 marcos**.

Anomalía de Belady

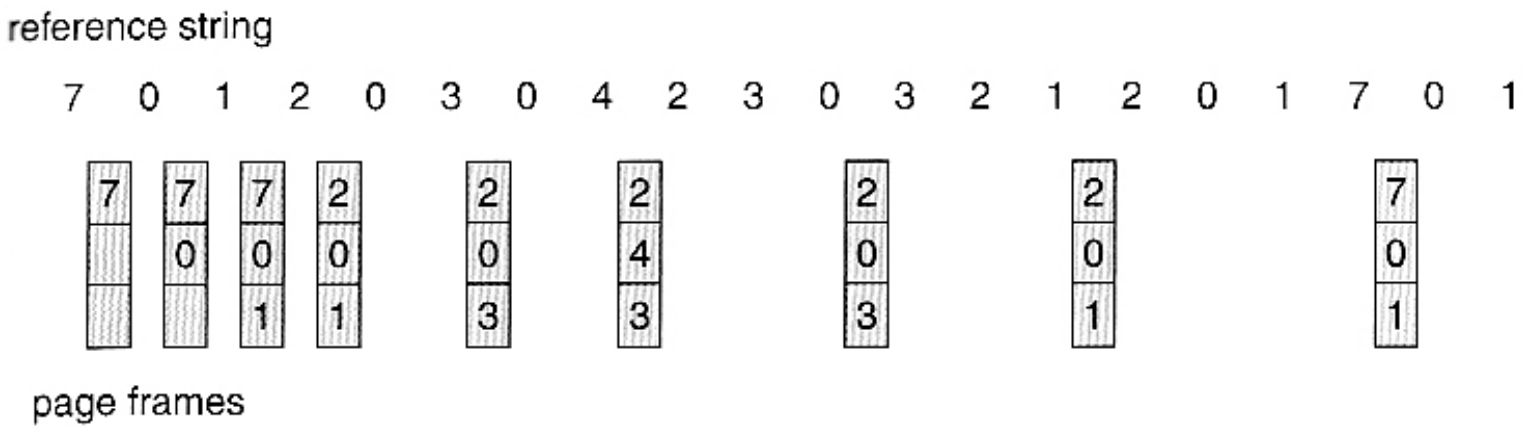


Sustitución óptima de páginas

- Se **redefine** el concepto de **algoritmo óptimo** como aquel que **minimiza fallos** de página y **no** está sujeto a la anomalía de **Belady**.
- **Sustituir** la página que **no vaya a ser utilizada** durante el período de tiempo más largo.
- **Problema**: conocimiento **a priori** de la cadena de referencia. Se usa como elemento de **comparación**.

Sustitución óptima de páginas

- Se **redefine** el concepto de **algoritmo óptimo** como aquel que **minimiza fallos** de página y **no** está sujeto a la anomalía de **Belady**.
- **Sustituir** la página que **no vaya a ser utilizada** durante el período de tiempo más largo.
- **Problema**: conocimiento **a priori** de la cadena de referencia. Se usa como elemento de **comparación**.

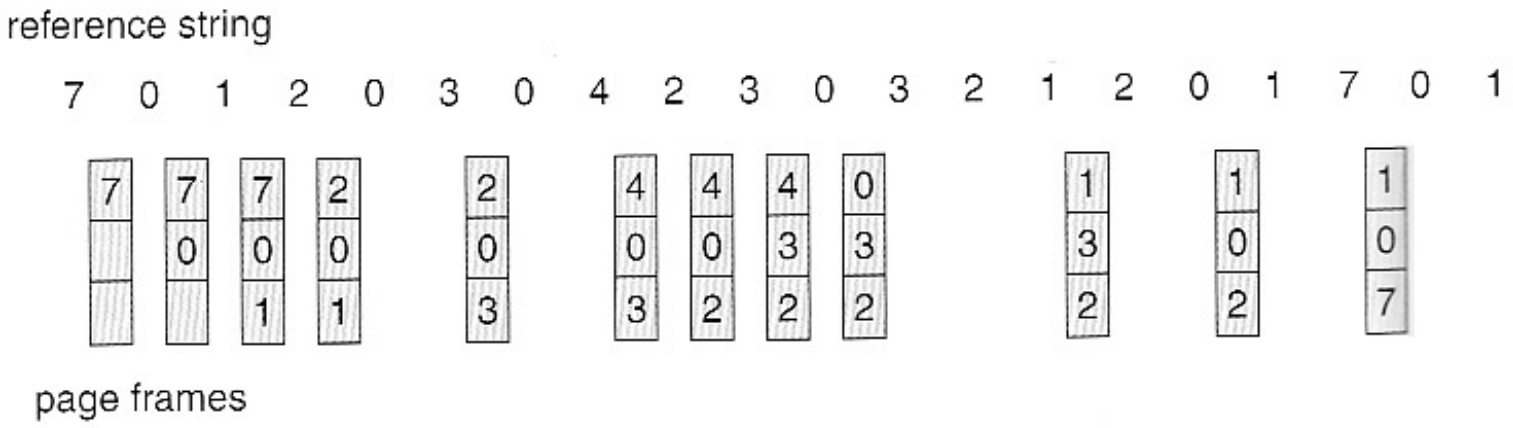


Sustitución de páginas LRU

- Surge como **aproximación** al algoritmo **óptimo**. Se utiliza el **pasado reciente** en vez del **futuro próximo**.
- **Sustituir** la página que **no haya sido utilizada durante el período más largo de tiempo**.
- **least recently used**: menos recientemente utilizada.

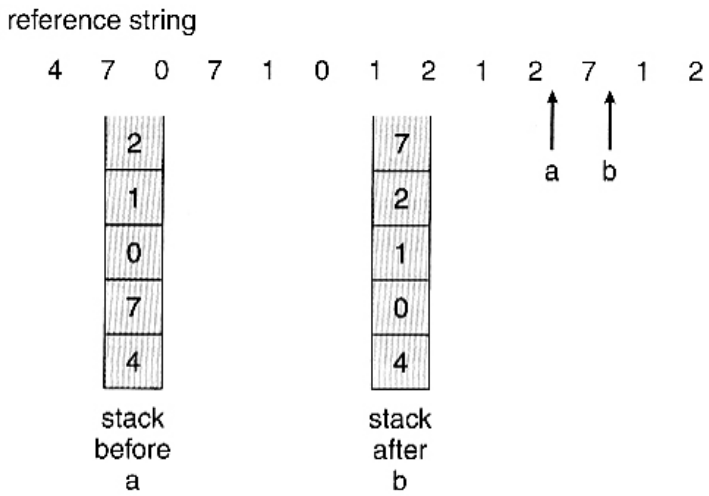
Sustitución de páginas LRU

- Surge como **aproximación** al algoritmo **óptimo**. Se utiliza el **pasado reciente** en vez del **futuro próximo**.
- **Sustituir** la página que **no haya sido utilizada durante el período más largo de tiempo**.
- **least recently used**: menos recientemente utilizada.



Implementación de LRU

- Algoritmo bastante **utilizado** porque se considera que da **buenos resultados**.
- El problema es que puede **requerir** bastante **asistencia hardware**.
- Existen **dos implementaciones**: por contador y por pila. Siempre por Hardware (no uso INTS).



Sustitución de páginas mediante aproximación LRU

- Algunos sistemas informáticos **no** pueden proporcionar la **asistencia** hardware requerida por LRU.
- Pero **si** proporcionan una **ayuda** mediante el llamado **bit de referencia**.
- Este bit se **activa** cada vez que se **referencia** la **página** y se encuentran en cada entrada a la tabla de páginas.
- **Inicialmente** está a **0** y el hardware lo pone a **1** cuando se **referencia** la página.
- Se puede saber las páginas **utilizadas** pero **no** el **orden**.

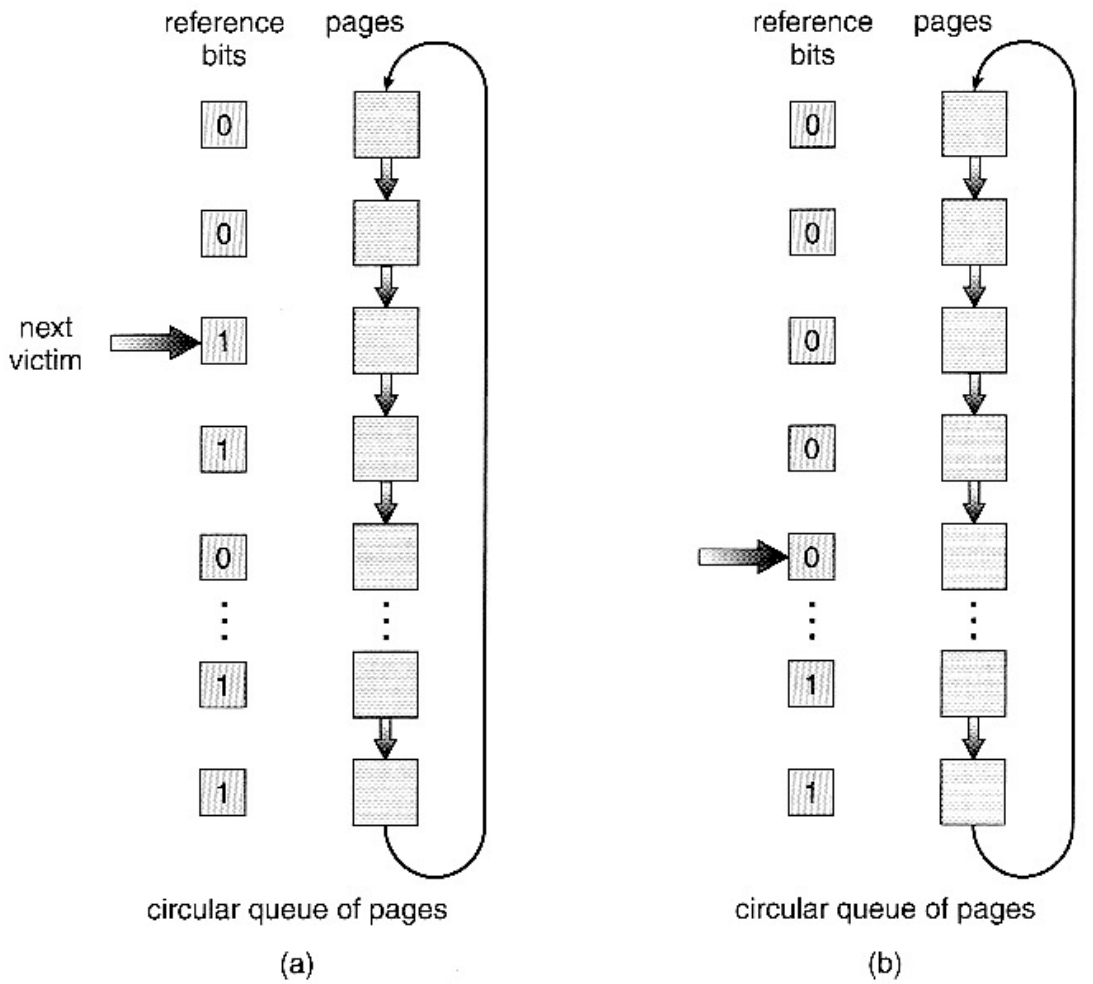
Algoritmo de los bits de referencia adicionales

- En vez de **1 bit** de referencia por página, se usa **1 byte**.
- Ese **byte** representa el **histórico** de utilización de una página en los últimos **8 periodos temporales**.
- El bit de referencia se transfiere al de **mayor peso**.
- **Ejemplos**: 00000000, 11111111, 10010010 y 01100011
- La página con número **más bajo** (E.S.S) será la menos recientemente utilizada (**víctima**).

Algoritmo de segunda oportunidad

- Algoritmo de sustitución **FIFO** que inspecciona el **bit de referencia**.
- Si el bit es **0**, la página se **sustituye**. Si es **1**, el bit se pone a **0** pero no se sustituye (**segunda oportunidad**).
- A continuación se estudia la página **próxima** en la **FIFO**.
- Se conoce también como "**algoritmo del reloj**" y se implementa mediante una **cola circular**.
- Si **todos** los **bits** de referencia son **1**, se comporta como un **FIFO**.

Algoritmo de segunda oportunidad



Algoritmo mejorado de segunda oportunidad

- Se utiliza una **pareja de bits** (bit de referencia, bit de modificación).
- Como **enteros**: (0,0), (0,1), (1, 0), (1, 1).
- Da como **segundo criterio** de selección de víctimas, la **no modificación** de las páginas.
- Esto permite **reducir** las operaciones de **Entrada/Salida**.

Índice

Sistemas monousuario

- Ejemplo **128 kb**, con marcos de **1kb** y el monitor necesita **35kb**.
- Quedarían **93 marcos** para el proceso de usuario.
- Las 93 primeras páginas provocarían **93 fallos de página**.
- Luego se aplicaría algún **algoritmo** de **sustitución**.
- Una vez el proceso termina los **93 marcos** quedan **libres**.

Número mínimo de marcos

- La **asignación** de un número mínimo de **marcos**, viene relacionada con el **rendimiento**.
- Mientras se **minimiza** el número de **marcos** asignado a un proceso se **incrementa** la tasa de **fallos** de página.
- Al menos debemos tener para un proceso **todas** las **páginas** a las que se hace **referencia** en una **instrucción**.
- Por tanto, el **número mínimo de marcos** depende del **juego** de **instrucciones**. Se debe **limitar** el **nivel** de **indirección** (número de accesos necesarios para obtener el valor del dato). Por ejemplo, nivel de indirección máximo de 16, requiere 17 marcos.

Algoritmos de asignación. Equitativa.

- La **forma** más **directa** es realizar una **asignación equitativa**. m marcos y n procesos $\rightarrow$ m/n marcos por proceso
- **93** marcos y **5** procesos $\rightarrow$ **18** marcos.
- Los **tres sobrantes** pueden componer el conjunto de marcos libres.

Algoritmos de asignación. Proporcional. I

- Se asocia **memoria** libre a cada proceso en **función** de su **tamaño**. Sea s_i el tamaño del proceso p_i.
- Sea m el número de **marcos libres** y a_i el número de **marcos** que se **asignarán** al proceso p_i.
- a_i se ajustaría al entero de **necesidad mínima** de cada proceso.

$$S \leftarrow \sum s_i \tag{1}$$

$$a_i \leftarrow \frac{s_i}{S} \cdot m \tag{2}$$

Algoritmos de asignación. Proporcional. II

$$\frac{10}{137} \cdot 62 \approx 4 \tag{3}$$

$$\frac{127}{137} \cdot 62 \approx 57 \tag{4}$$

- En los dos algoritmos de asignación estudiados **no** se tiene en **cuenta** la **prioridad**.

$$\frac{10}{137} \cdot 40 + \frac{2}{3} \cdot 22 \leftarrow 2,9 + 14,6 \approx 18 \tag{5}$$

$$\frac{127}{137} \cdot 40 + \frac{1}{3} \cdot 22 \leftarrow 37 + 7,3 \approx 44 \tag{6}$$

Índice

Definición

- ¿Qué ocurre cuando el **número** de marcos de un proceso cae por **debajo** del **mínimo** requerido por la **arquitectura** de la máquina?
- ¿Qué ocurre si un **proceso** no dispone de **suficientes marcos** para dar soporte a las **páginas** que usa **activamente**?
- **Sobrepaginación**: Un proceso invierte **más tiempo** implementando los mecanismos de **paginación** que en la propia **ejecución** del proceso.

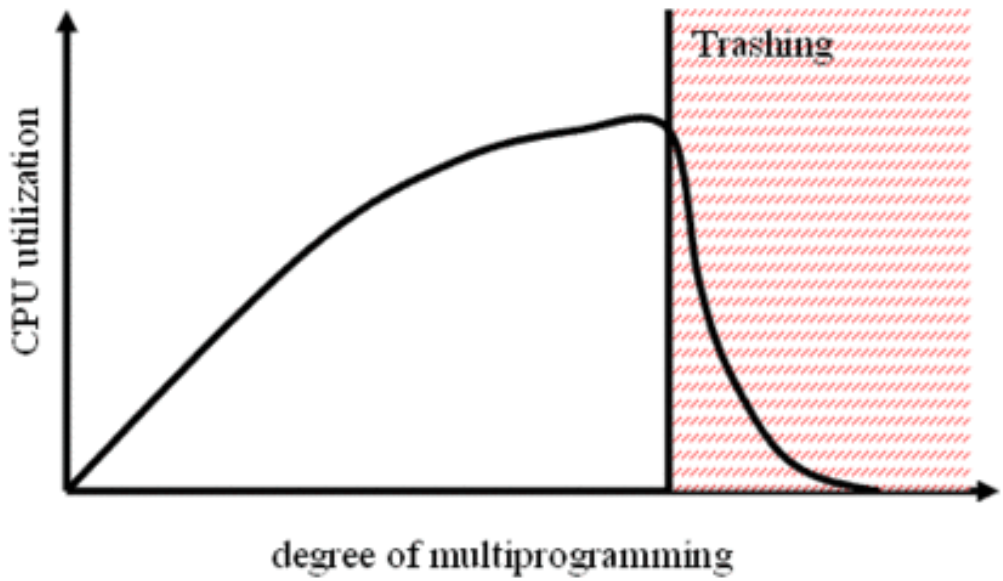
Causas I

- La **sobrepaginación** provoca graves **problemas** de **rendimiento**.
- Si el SSOO considera que el **uso** de la **CPU** es **bajo**, **incrementa** el grado de **multiprogramación**.
- En una política de **sustitución global**, se quitan marcos a otros procesos. Algunos procesos, pueden necesitar más marcos y generan **fallos de página**.

Causas II

- Los **procesos** que se han quedado **sin** esos **marcos**, también **empiezan** a generar **fallos** de página.
- La **cola** de **procesos** preparados puede quedarse **vacía** (todos en paginación). La **tasa** de uso de **CPU** baja, y se incrementa el grado de multiprogramación.

Causas III



Limitar Efectos

- En una situación de sobrepaginación, se debe **decrementar** el grado de **multiprogramación**.
- Usar algoritmos de **sustitución local**.
- Aún así, basta con que haya **un proceso en sobrepaginación** para que **afecte** los *tma* de todos los **demás**.

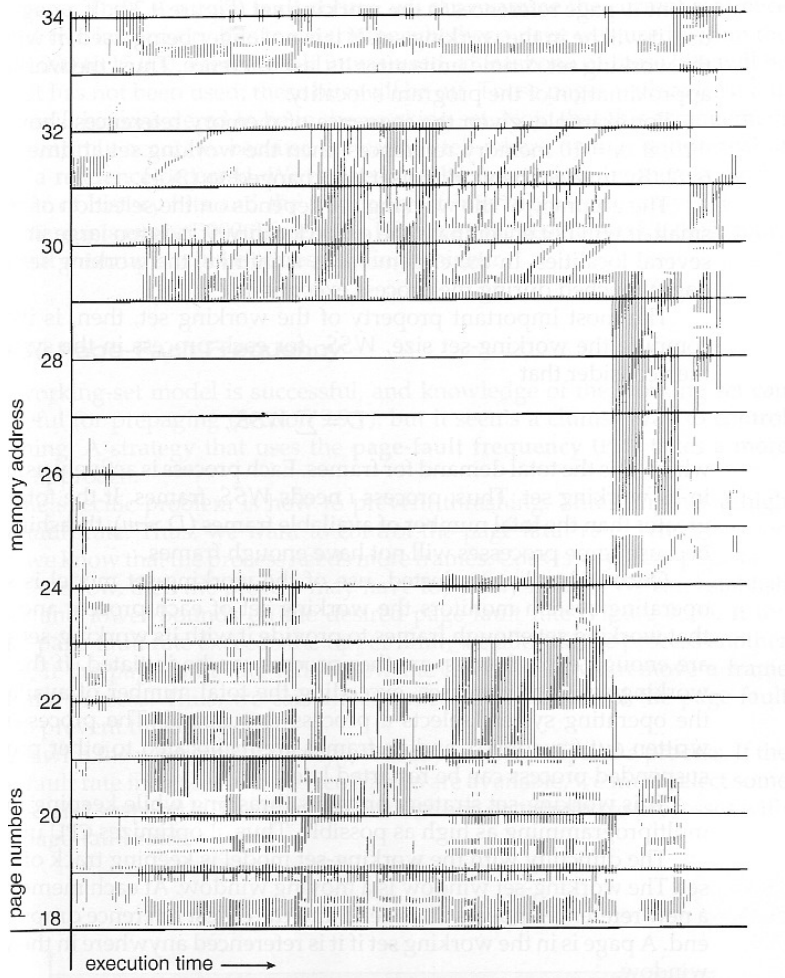
Prevención I

- Para **prevenir** la **sobrepaginación** se proporcionarán a los procesos, tantos **marcos** como **necesiten**. ¿Cómo sabemos la necesidad?
- Se define **conjunto** de **trabajo** de un proceso como el conjunto de **páginas** a las que ha **accedido** el proceso en las últimas *n* **referencias**.
- Estrategia basada en el **conjunto de trabajo** define el **modelo de localidad** de ejecución del proceso.
- A medida que un **proceso** se **ejecuta**, se va **desplazando** de una **localidad** a otra.

Prevención II

- **Localidad**: conjunto de páginas que se utiliza activamente de forma combinada. **Programa**: varias localidades.
- P. ej. cuando se invoca una **función**, ésta define una **nueva localidad** (instrucciones, variables locales y variables globales).
- **No** se generan **nuevos fallos** de página hasta que no se cambia de localidad.

Principio de Localidad

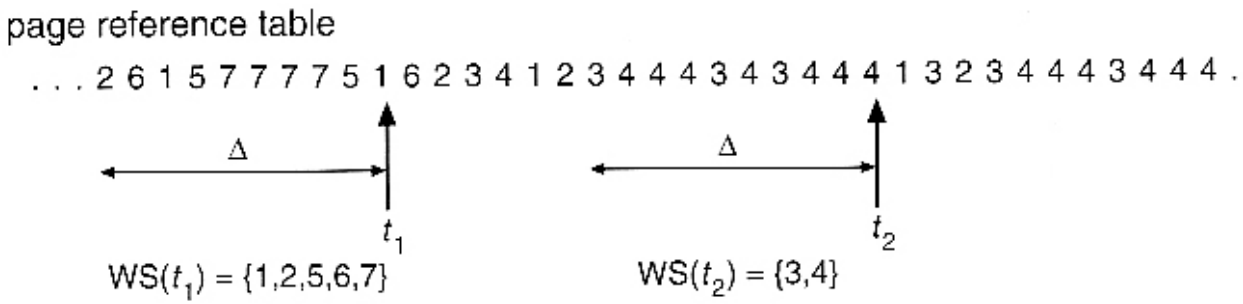


Modelo del Conjunto de Trabajo I

- Se basa en la **suposición** de **localidad**. Utiliza un parámetro Δ para definir el **tamaño de la ventana** del conjunto de trabajo.
- **Examina** las **últimas** referencias. Las Δ referencias últimas constituyen el conjunto de trabajo. Páginas **usadas activamente** se encuentra en dicho conjunto.
- El **conjunto** de trabajo es una **aproximación** a la **localidad** de un proceso.

Modelo del Conjunto de Trabajo II

- Si Δ es pequeña **no abarca** la localidad completa, si es **grande**, puede que se **solapen** varias localidades.
- La **selección** de Δ es **crítica** para cada proceso.



Tamaño del Conjunto de Trabajo

$$D \leftarrow \sum WSS_i \tag{7}$$

- D es la **demanda** total de **marcos**.
- m es el número de **marcos** disponibles.
- Hay sobrepaginación si $D > m$

Funcionamiento basado en el Conjunto de Trabajo

- El SSOO **monitoriza** el conjunto de trabajo de cada **proceso**.
- **Asigna** a cada proceso **marcos** como para satisfacer **requisitos** de tamaño del Conjunto de Trabajo. Si quedan marcos **libres**, se puede **iniciar** otro **proceso**.
- Si se incrementan tamaños tal que $D > m$, se **selecciona** un **proceso** y se **suspende** (liberando memoria).

Frecuencia de fallos de página

- **Alternativa** al modelo anterior.
- **PFF**: Page Fault Frequency.
- **Detectar** sobrepaginación $\rightarrow$ **alta** tasa de **fallos**; necesita más marcos.
- Si **PFF** baja quizás tiene **demasiados** marcos asignados.
- Se **establecen** límite **inferior** y **superior** para **PFF**. Si se supera se asignan marcos (si libres, sino suspensión) y en el caso contrario se liberan marcos.

Índice

Definición y funcionamiento I

- El **espacio** de memoria de **intercambio** o **Swap**, es lo que se conoce como **memoria virtual**.
- La **diferencia** entre la memoria **real** y la **virtual** es que esta última utiliza espacio en el **disco duro** en lugar de un módulo de memoria.
- Cuando la **memoria** real se **agota**, el sistema **copia** parte del **contenido** de esta **directamente** en este espacio de memoria de **intercambio** a fin de poder realizar **otras** tareas.

Definición y funcionamiento II

- Utilizar memoria **virtual** tiene **como** ventaja el proporcionar la memoria **adicional** necesaria cuando la memoria real se ha **agotado** y se tiene que realizar un **proceso**.
- El **inconveniente** radica en que, como consecuencia de **utilizar** espacio en el **disco** duro, la utilización de ésta es mucho más **lenta**.
- Uno puede **percatarse** de esto cuando el disco duro empieza a **trabajar repentinamente** hasta por varios minutos **después** de abrir varias **aplicaciones**.

Definición y funcionamiento III

- ¿Cuanto **espacio** para memoria de intercambio se debe **asignar** al **sistema**?
  1. Menos de **1GB** RAM: **doble** de la **cantidad** total de memoria **RAM**.
  2. Más de **1GB** RAM: misma **cantidad** del total de **memoria** RAM, más **2GB**.

Circunstancias de modificación I

Ampliar el espacio de memoria **virtual** puede ser **práctico** en los siguientes **casos**:

- **Sistemas** en donde adquirir **memoria adicional** es imposible, y se es **consciente** que la memoria de intercambio es **muchísimo** más **lenta** que la memoria **RAM**.
- En **equipos** con trabajo **intensivo** que consumen mucha **memoria** (diseño gráfico, por ejemplo).
- **Servidores** de alto **desempeño** en donde se desea contar con un amplio **margen** de espacio de intercambio para **satisfacer** las demandas de **servicios**.

Circunstancias de modificación II

- Sistemas que **actualizaron** desde una versión de **núcleo** (por ej. 2.2), a una versión de **núcleo** más **reciente** (por ej. 2.4 o 2.6).
- Sistemas donde **aumentó** la cantidad de memoria **RAM** y se **encuentran** con la **problemática** de cubrir la **cuota mínima** de espacio de memoria de **intercambio**.

Opciones de ampliación

- **Cambiar** el **tamaño** de la partición es el método más **efectivo**. Sin embargo, ésto **representa** un **riesgo**, debido que podría ocurrir un **error** durante el proceso de **repartición** que podría **desencadenar** en **pérdida** de datos en un **disco duro**.
- Crear un **archivo** para memoria de **intercambio**: es otro **método** más **sencillo** y sin **riesgo** alguno, consiste en utilizar un **archivo** de intercambio de forma **similar** a como se hace en otros sistemas **operativos**.

Crear una partición de intercambio adicional

- Una vez **cambiada** la **tabla** de **particiones** del disco duro y creada una **nueva** partición de **memoria** de intercambio:

mkswap

```
mkswap -c [dispositivo]
mkswap -c /dev/sda8
```

- la opción -c indica se **verifiquen** sectores del disco duro **buscando** bloques **dañados** a fin de marcar estos y **evitar** utilizarlos.

salida mkswap

```
Setting up swapspace version 1, size=1048576 bytes no label,
UUID=d2fea5ab-c677-8047-789a-e54ae19c506b
```

Activar una partición de intercambio adicional

swapon

swapon [dispositivo]
swapon /dev/sda8

free

total used free shared buffers cached
Mem: 321364 312576 8788 0 940 63428
-/+ buffers/cache: 248208 73156
Swap: 1426416 0 1426416

Utilización automática del espacio de intercambio

Edición /etc/fstab

vim /etc/fstab

Agregar línea

[partición] swap swap defaults 0 0
/dev/sda8 swap swap defaults 0 0

Utilización de un archivo como memoria de intercambio I

- Este **método no** requiere hacer **cambios** en la tabla de **particiones** del disco **duro**.
- Es **idóneo** para **usuarios** poco experimentados, para quienes desean **evitar** tomar **riesgos** al **cambiar** la tabla de **particiones** el disco duro o bien para **quienes** requieren de **memoria** de intercambio **ocasional** o de manera **circunstancial**.
- El archivo de **memoria** de intercambio puede ser **colocado** en cualquier **directorio** del disco **duro**.

Utilización de un archivo como memoria de intercambio II

- Al comando **dd**, se le **especifica** que se escribirán **ceros** (**if=/dev/zero**) para crear el archivo /swap (**of=/swap**), en bloques de **1024** bytes (bs=1024) hasta completar una **cantidad** en bytes determinada (count=[cantidad multiplicada por el valor de bs]).
- En el siguiente **ejemplo** se realiza lo anterior hasta completar **524288000** bytes (1024 por bloque), que equivalen a **512MB**:

dd

```
dd if=/dev/zero of=/swap bs=1024 count=512000
```

Utilización de un archivo como memoria de intercambio III

- Posteriormente se le da **formato**.

mkswap

```
mkswap /swap
Setting up swapspace version 1, size = 511996 KiB no label,
UUID=fed2aba5-77c6-4780-9a78-4ae5e19c506b
```

- Finalmente se **activa** la "partición".

swapon

```
swapon /swap
```

Utilización de un archivo como memoria de intercambio IV

free

total used **free** shared buffers cached
Mem: 321364 312576 8788 0 940 63428
-/+ buffers/cache: 248208 73156
Swap: **3145724** 0 **3145724**

- Para que se **cree** en el **arranque** del sistema:

Edición /etc/fstab

vim /etc/fstab
/swap swap swap defaults 0 0

Optimización del sistema I

- El **núcleo** de GNU/Linux permite **cambiar** con que **frecuencia** las **aplicaciones** y programas son **movidas** de la memoria **física** hacia la memoria de **intercambio**.
- El valor **predeterminado** es **60**, como puede **observarse** al mirar el contenido de **/proc/sys/vm/swappiness**.
- Pueden establecerse **valores** entre **0 y 100**, donde el valor más **bajo** establece que se **utilice menos** la memoria de intercambio, lo cual significa que se **reclamará** en su lugar el **caché** de la **memoria**.

Optimización del sistema II

- El valor **predeterminado** de **60**, fue establecido teniendo en mente a quienes **desarrollan** el **núcleo** de Linux, con la finalidad de permitir realizar pruebas y diagnósticos.
- Para la **mayoría** de los **casos**, conviene cambiar este **valor** por uno más **bajo** a fin de que el sistema utilice **menos** la memoria de **intercambio** y utilice **más** la memoria **cache**.
- Un valor **apropiado** y que **funcionará** para la mayoría de los sistemas en **producción** es **10**.

Cambio valor swappiness

```
echo 10 > /proc/sys/vm/swappiness
```

Optimización del sistema III

Cambio valor swappiness

```
sysctl -w vm.swappiness=10
```

- Este **cambio** permanece **activo** hasta que se **reinicie** el sistema.
- Se puede hacer **permanente**:

Edición /etc/sysctl.conf

```
vim /etc/sysctl.conf
vm.swappiness = 10
```

Liberar la memoria swap en uso I

- Una vez se **activa** nuestra memoria **swap** nuestro sistema se **acostumbra** a **saturar** y **ralentizar**.
- Una **solución** para este **problema** es **reiniciar** el **equipo**, pero existen **soluciones** para **liberar** la memoria Swap y **no** tener que **apagar** el ordenador.
- Se **traspasará** el **contenido** que tenemos **almacenado** en nuestra memoria **swap** hacia la memoria **RAM**.

Liberar la memoria swap en uso II

- Primero hay que **asegurarse** de que la memoria **RAM** tiene capacidad **libre** para poder **albergar** el **contenido** que está **guardado** en la **swap**.
- Para hacer esto **utilzamos** free. Vemos si el tamaño **libre** de **RAM** es mayor que el de la **Swap**.

Traspaso de Swap a RAM

```
sudo swapoff -a ; sudo swapon -a
```

- **Comprobamos** con **free** que se ha realizado **correctamente**.

Bibliografía I

- **Fundamentos de sistemas operativos**. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. Editorial: McGraw-Hill.
  - **Capítulo 8**. Conceptos Generales de Gestión de Memoria.
  - **Capítulo 9**. Organización de la memoria virtual.
- **Gestión de espacio de memoria de intercambio (swap) en Linux**. `http://www.alcancelibre.org/staticpages/index.php/como-swap-linux`
- **Optimizar el uso de la memoria Swap** `http://geekland.hol.es/optimizar-el-uso-de-la-memoria-swap/`

Tema 5. Gestión de la memoria virtual.

Sistemas Operativos II

Universidad de Castilla-La Mancha

Tema 7. Hebras de Ejecución.

Sistemas Operativos II

Universidad de Castilla-La Mancha

Índice

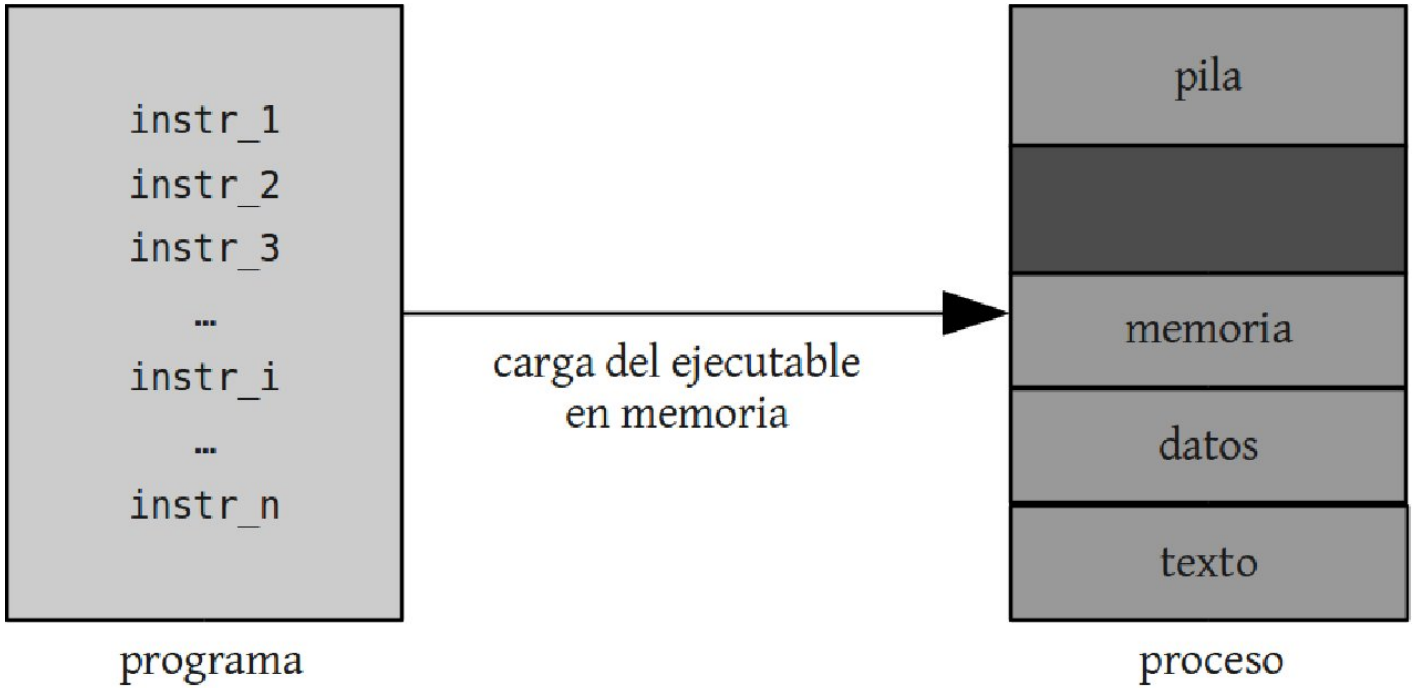
Objetivos

- Introducir la **noción** de **hebra** como unidad fundamental de utilización de la CPU que **constituyen** la base de los sistemas **multitarea**.
- Estudiar las **características** de la **programación** multihebra.
- Estudiar **APIs** para hebras como puede ser **POSIX**.

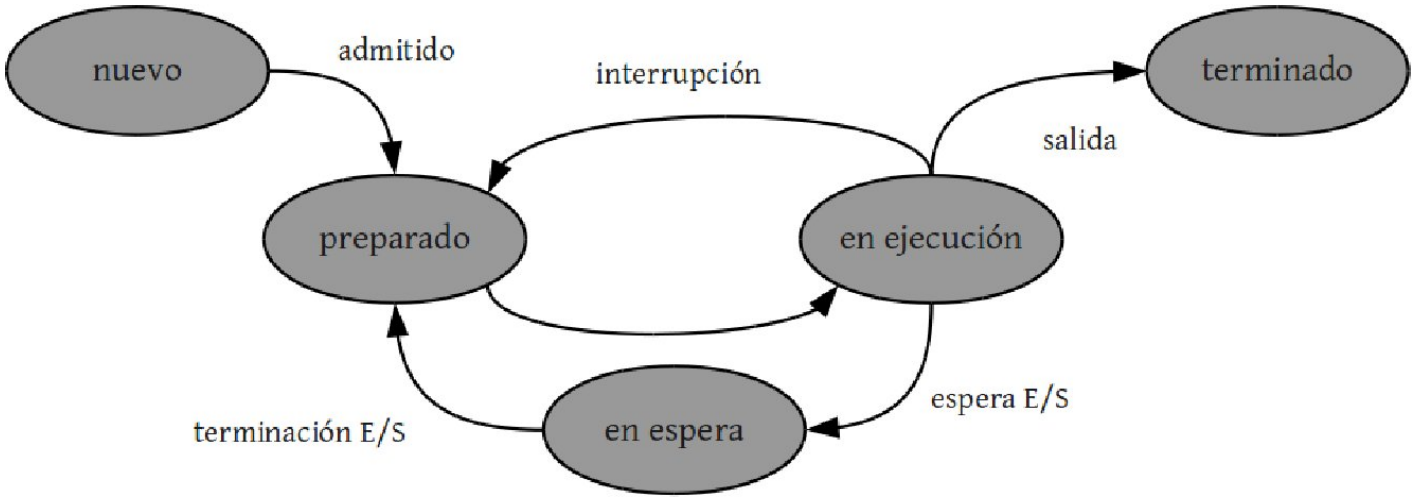
Índice

Proceso en memoria

- Un **proceso** es un **programa** en **ejecución**.

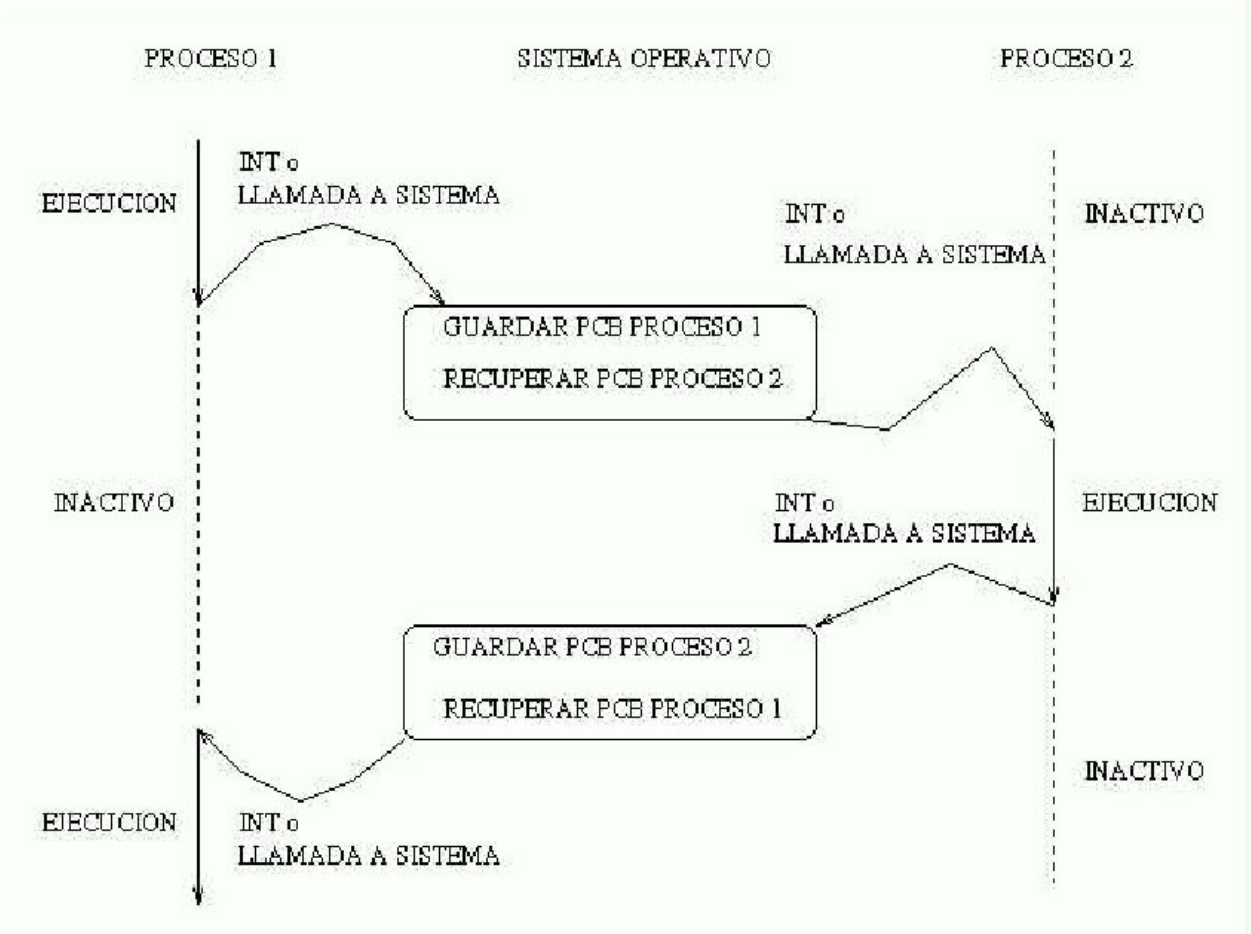


Estados de un proceso

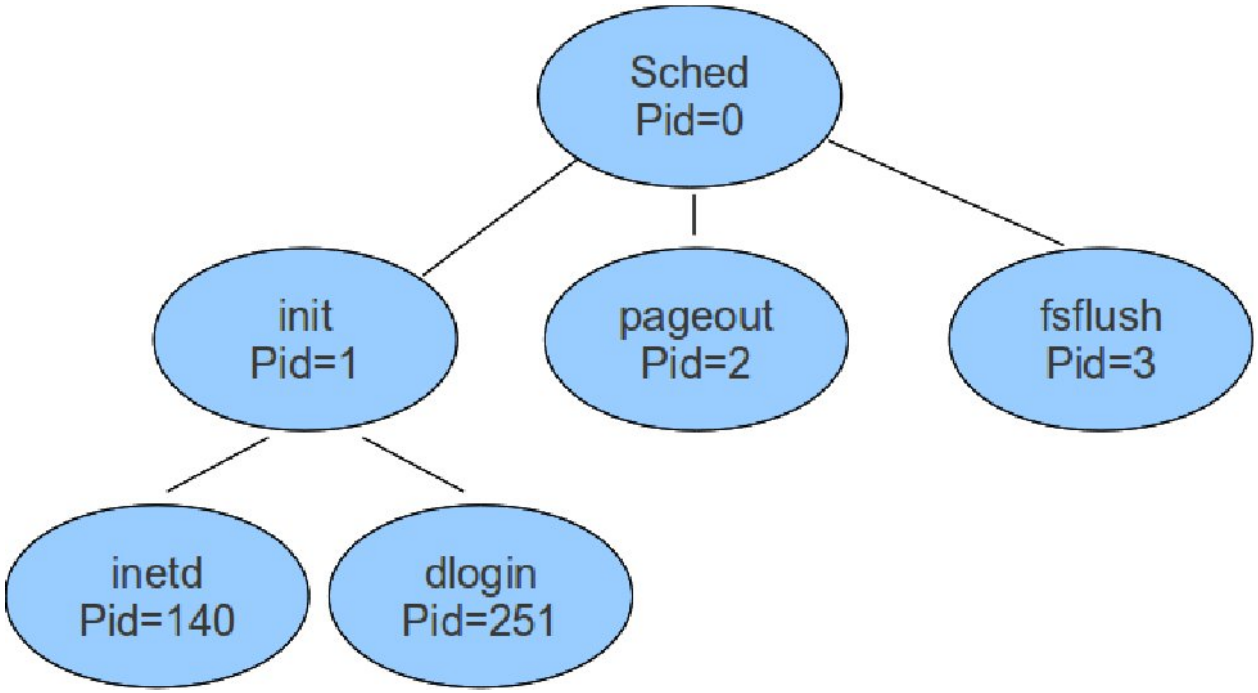


Bloque de control de proceso (PCB)

Cambio de contexto



Árbol de procesos



Índice

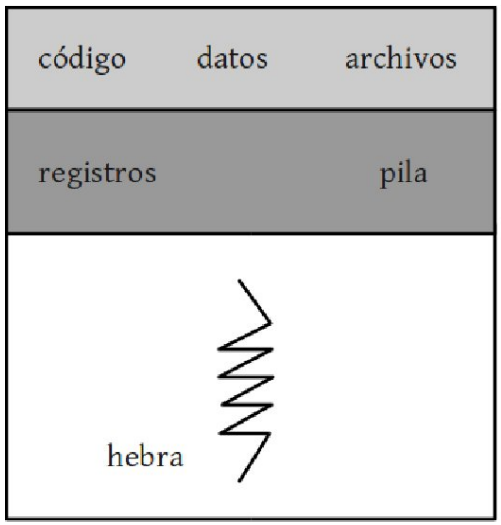
Definición de Hebra

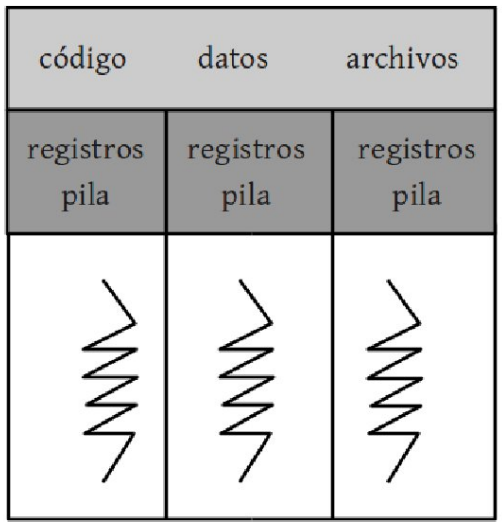
- Una **hebra** es un flujo de control perteneciente a un **proceso** (a veces se habla de tareas con hebras).
- Se les suele denominar también **procesos ligeros**, threads, hilos, etc.
- La **sobrecarga** debida a su creación y comunicación es **menor** que en los procesos **pesados**.
- Cada **hebra** pertenece a un **proceso** pesado (proceso).
- Todas las **hebras** comparten su espacio de **direccionamiento**.
- Cada **hebra** dispone de su propia política de **planificación**, pila y contador de **programa**.

HEBRA VS. PROCESO.

Procesos y Hebras I

- **Procesos**: Procesos **pesados**.
- **Hebras**: Procesos **ligeros**.





Procesos y Hebras II

Todas las **hebras** de un proceso **comparten**:

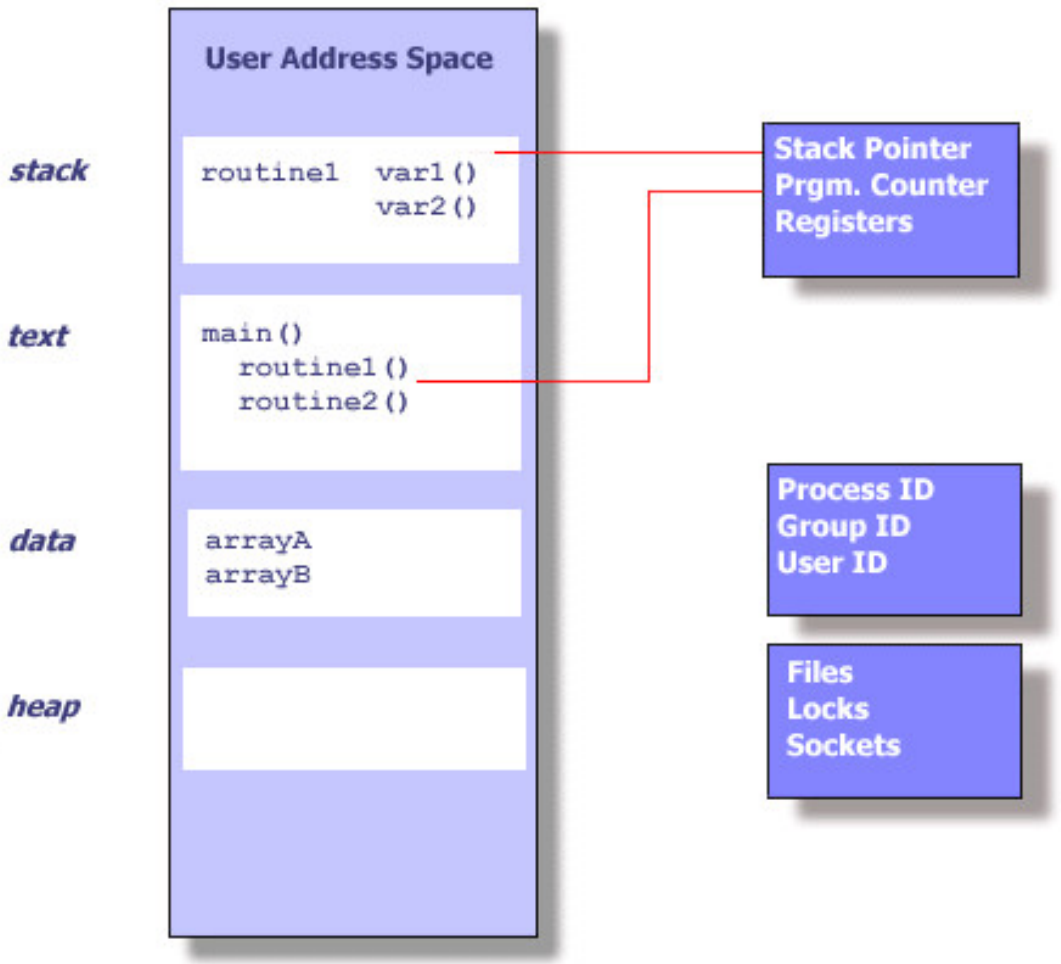
- **Memoria** (código y **variables** globales). Las hebras dentro de un mismo proceso se pueden **comunicar** mediante **variables** globales (¡con cuidado!).
- Descriptores de **archivos** y **sockets** abiertos.
- Manejadores de **señales** (signal **handlers** y signal **dispositions**).
- Entorno de **trabajo** (**directorio** actual, **identificador** de usuario, etc.).

Procesos y Hebras III

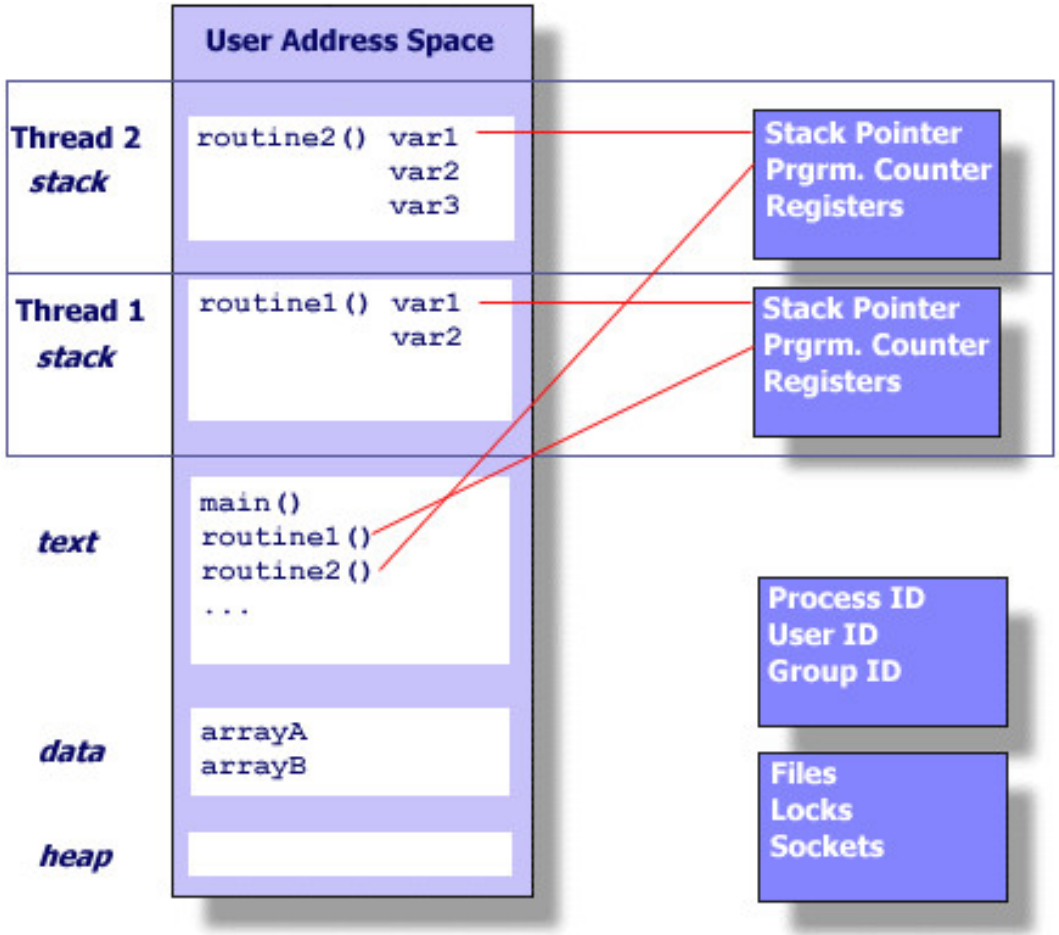
Cada **hebra** tiene su propio:

- **Identificador** de Hebra (Thread ID)
- **Stack**, Registros, **Contador** de Programa. Se guardan en el **Thread Control Block**.

Procesos y Hebras IV



Procesos y Hebras V



VENTAJAS E INCONVENIENTES.

Ventajas I

- Solo existe una **copia** de las variables **compartidas**.
- El **coste** de **comunicación** es mucho **menor** que entre procesos pesados.
- Explotación del **paralelismo**.
- Explotación de **concurrencia**.

Ventajas II

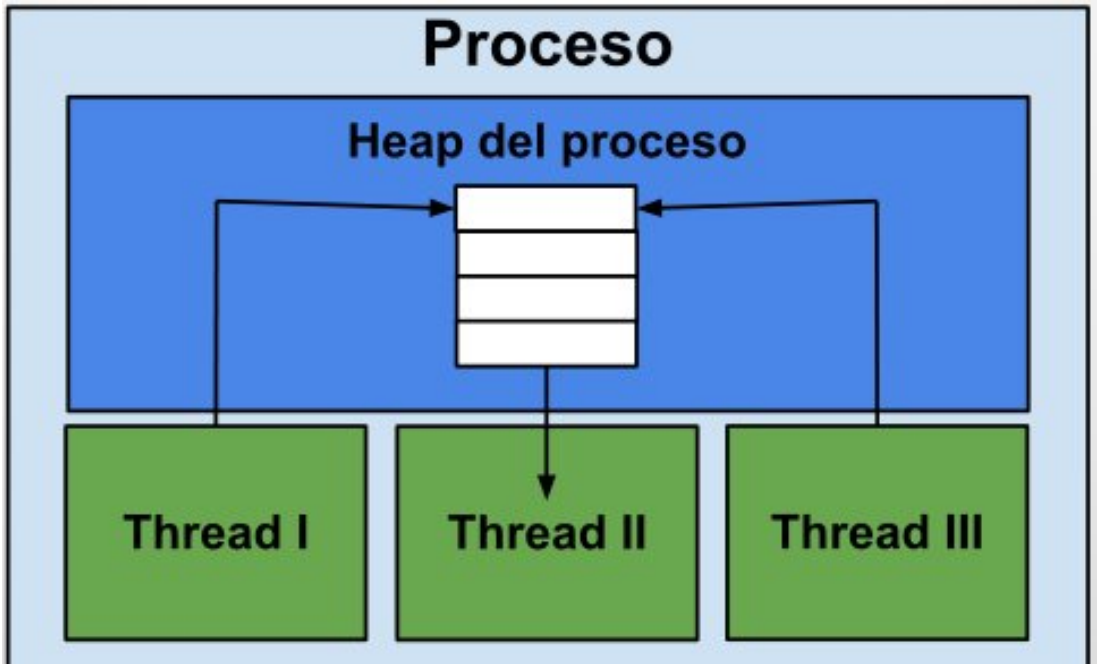


Figura : Acceso a variables compartidas

Ventajas III

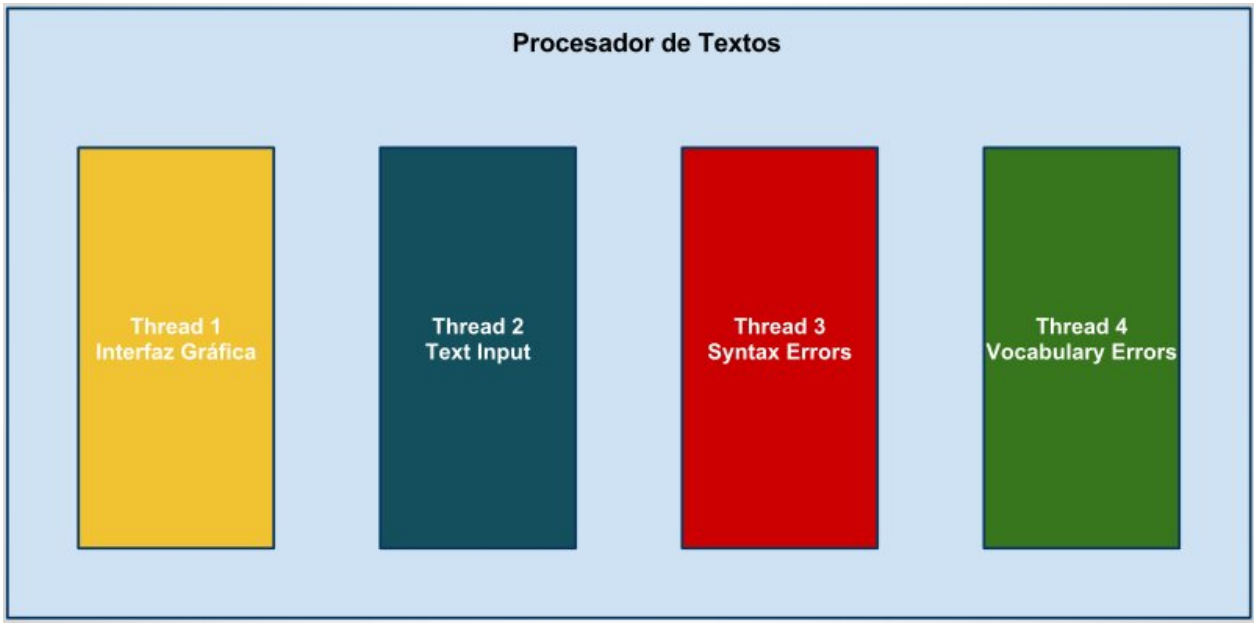


Figura : Concurrencia

Ventajas IV

Platform	fork()			pthread_create()		
	real	user	sys	real	user	sys
Intel 2.6 GHz Xeon E5-2670 (16 cores/node)	8.1	0.1	2.9	0.9	0.2	0.3
Intel 2.8 GHz Xeon 5660 (12 cores/node)	4.4	0.4	4.3	0.7	0.2	0.5
AMD 2.3 GHz Opteron (16 cores/node)	12.5	1.0	12.5	1.2	0.2	1.3
AMD 2.4 GHz Opteron (8 cores/node)	17.6	2.2	15.7	1.4	0.3	1.3
IBM 4.0 GHz POWER6 (8 cpus/node)	9.5	0.6	8.8	1.6	0.1	0.4
IBM 1.9 GHz POWER5 p5-575 (8 cpus/node)	64.2	30.7	27.6	1.7	0.6	1.1
IBM 1.5 GHz POWER4 (8 cpus/node)	104.5	48.6	47.2	2.1	1.0	1.5
INTEL 2.4 GHz Xeon (2 cpus/node)	54.9	1.5	20.8	1.6	0.7	0.9
INTEL 1.4 GHz Itanium2 (4 cpus/node)	54.5	1.1	22.2	2.0	1.2	0.6

Inconvenientes

- **Overhead** (tiempo desperdiciado por cambio de contexto) por **creación** de threads (stack, thread pools).
- **Sincronización**: más **bloqueos** al haber más threads.
- **Colisiones** en el acceso a **memoria**.
- Más **difícil** la **depuración**: debuggers, **trazadores**, puntos de **ruptura**, **reproducción** de la ejecución, etc.

DISEÑO DE PROGRAMAS QUE UTILIZAN HEBRAS.

Diseño de Aplicaciones I

- En las máquinas **multiprocesador** actuales las **hebras** están especialmente **diseñados** para su **utilización** en programación **paralela**.
- Pero no sólo en estos **sistemas**, sino pueden ser **utilizables** en cualquier aplicación de **programación** paralela en **general**.
- De manera general, para sacar **partido** de las **hebras**, éste debe **organizarse** en tareas **independientes** que puedan ser ejecutadas **concurrentemente**.

Diseño de Aplicaciones II

Existen un conjunto de **consideraciones** que hay que tener en **cuenta** a la hora de **diseñar** programas **paralelos**:

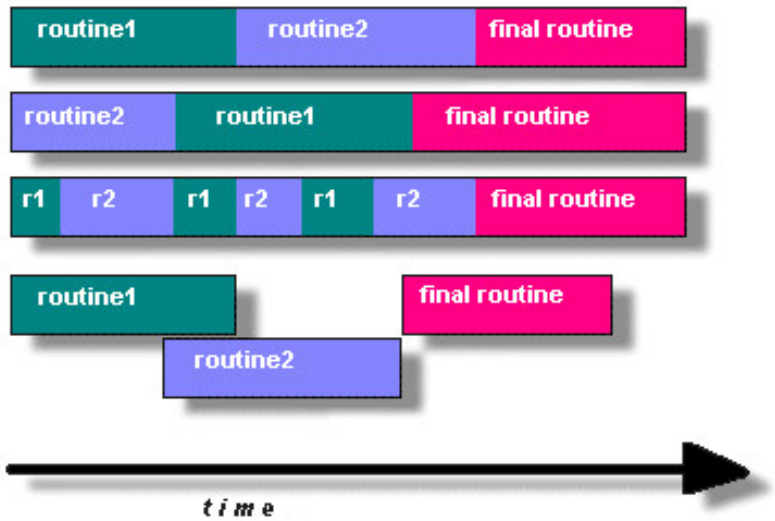
- ¿Qué modelo de **programación** paralela usar?
- El **particionamiento** del problema.
- El **balanceo** de la carga.
- Las **comunicaciones**.

Diseño de Aplicaciones III

- Las **dependencias** entre los datos.
- La **sincronización** y las **condiciones** de carrera.
- Aspectos **relativos** a la **memoria**.
- Aspectos **asociados** a la **E/S**.
- **Complejidad** del programa.
- **Esfuerzo**/Coste/**Tiempo** del **programador**.

Diseño de Aplicaciones IV

- Por **ejemplo**: La rutina1 y rutina2 son **intercambiables**, **intercalables** y **solapables**. Por tanto, son **candidatas** a **implementarse** mediante **hebras** (threading).



Diseño de Aplicaciones V

Los **problemas** que tienen las siguientes **características**, pueden considerarse **adecuados** para **hebras**:

- **Tareas** que pueden ser **ejecutadas**, o datos que pueden ser **manejados** por múltiples programas de manera **simultánea**.
- Existen **esperas** para **llamadas** de E/S de larga **duración**.
- Utilización **exhaustiva** de la **CPU** en algunas partes del **código** pero no en otras.
- Deben contemplarse **respuestas** a eventos **asíncronos**.
- Algunas **tareas** son más **importantes** que otras.

Modelos comunes de programas

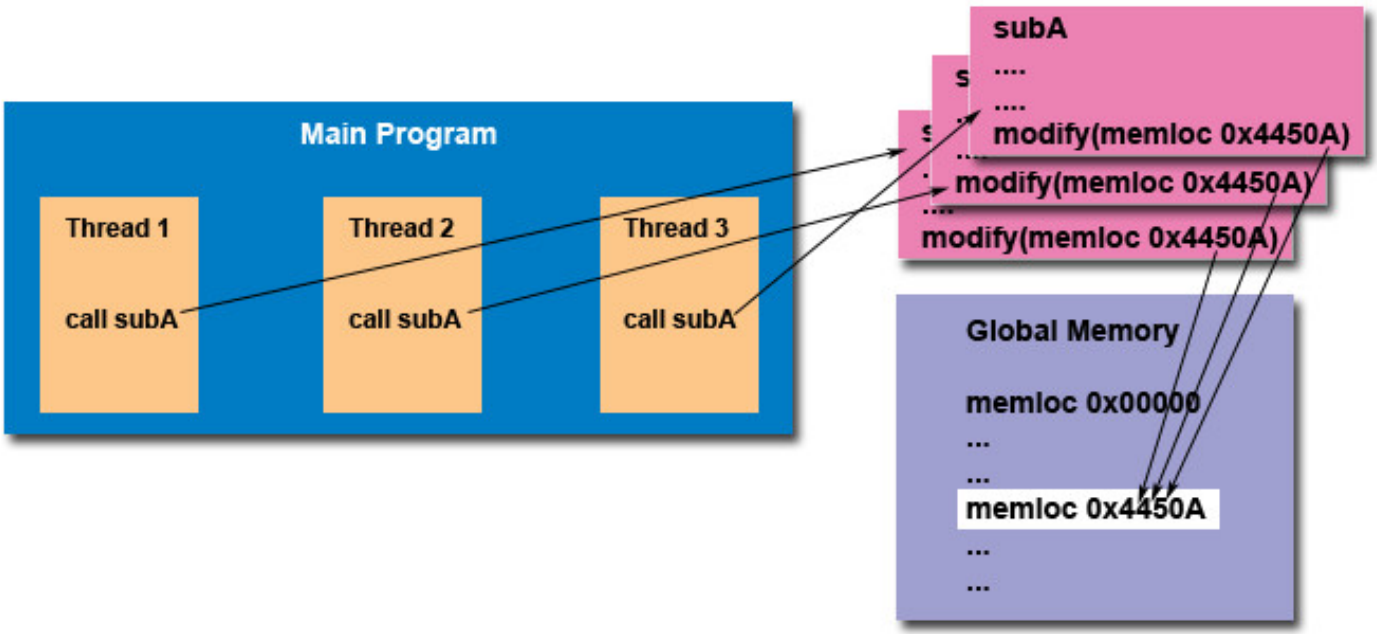
Modelos predefinidos para **programas** que utilizan **hebras** son:

- **Manager/worker**: una hebra, el **Manager** asigna trabajos a otras **hebras**. El **Manager** maneja todas las **entradas** y divide el trabajo para las otras **hebras**.
- **Pipeline**: una tarea es **dividida** en conjuntos de **suboperaciones** serie, pero ejecutadas de manera **concurrente**, por una hebra **diferente**. Ej: cadena de ensamblaje de **automóviles**.
- **Peer**: **similar** el modelo **Manager/Worker** pero la hebra principal además de crear el resto de hebras también "**participa**" en el trabajo.

Seguridad I

- Una **función** (o método) **thread-safe** puede ser invocada por **múltiples** hilos de ejecución sin **preocuparnos** de que los datos a los que accede dicha **función** (o método) sean **corrompidos** por alguno de los hilos ya que se **asegura** la atomicidad de la **operación**, es decir, la función se **ejecuta** de forma **serializada** sin **interrupciones**, por lo general, adquiriendo un **mútex**.
- **Recomendación**: se debe ser **cuidadoso** si una aplicación utiliza **librerías** u otros objetos que no garantizan explícitamente **thread-safeness**. Ante la duda, se debe asumir que no son **thread-safe** y **serializar** las **llamadas** a esta rutina.

Seguridad II

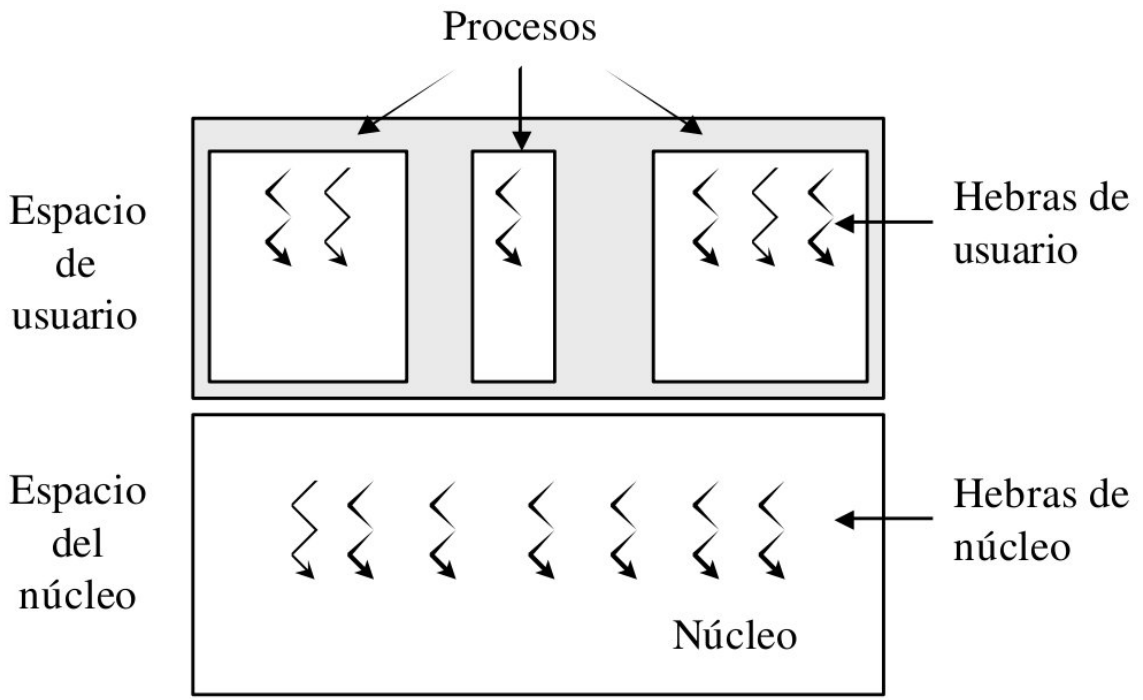


HEBRAS DE USUARIO Y DEL NUCLEO.

Tipos de hebras

Hay dos tipos de hebras:

- Las hebras que se ejecutan en el **espacio** de **usuario**.
- Las hebras que se ejecutan en el **espacio** del **núcleo**



Implementación de hebras

Existen varias formas de hacer **corresponder** las hebras de **usuario** con las hebras de **núcleo**:

- **Muchas** a Una.
- Una a **Una**.
- Muchas a **Muchas**.

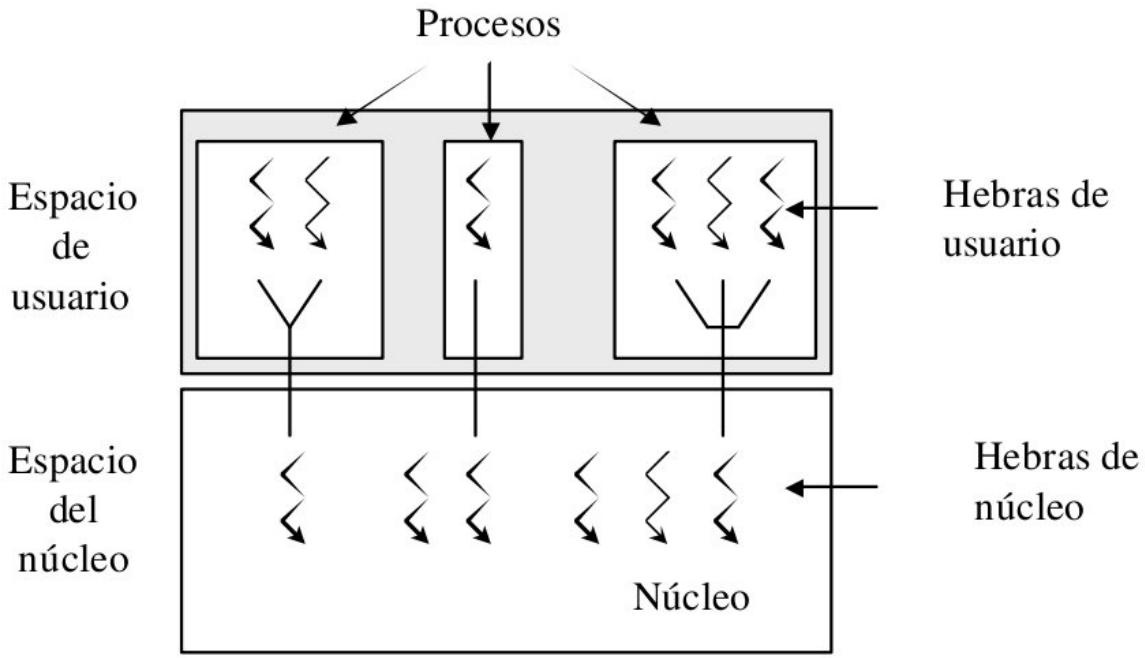
Cada una estas **formas** da lugar, respectivamente, a un **esquema** de implementación **diferente**.

- Implementación a nivel de **usuario**.
- Implementación a nivel de **núcleo**.
- Implementación **híbrida**.

Implementación a nivel de usuario.

Características:

- Correspondencia **Muchas** a **Una**.
- **Todas** las hebras de usuario de un proceso se vinculan a una **única** hebra del **núcleo**.



Implementación a nivel de usuario.

Ventajas:

- Gestión de hebras **eficiente**.
- Esquema empleado en **sistemas** operativos **no multihebra**.

Inconvenientes:

- Si una hebra se **bloquea**, se bloquea **todo** el **proceso**.
- **No** permite **multiprocesamiento**.

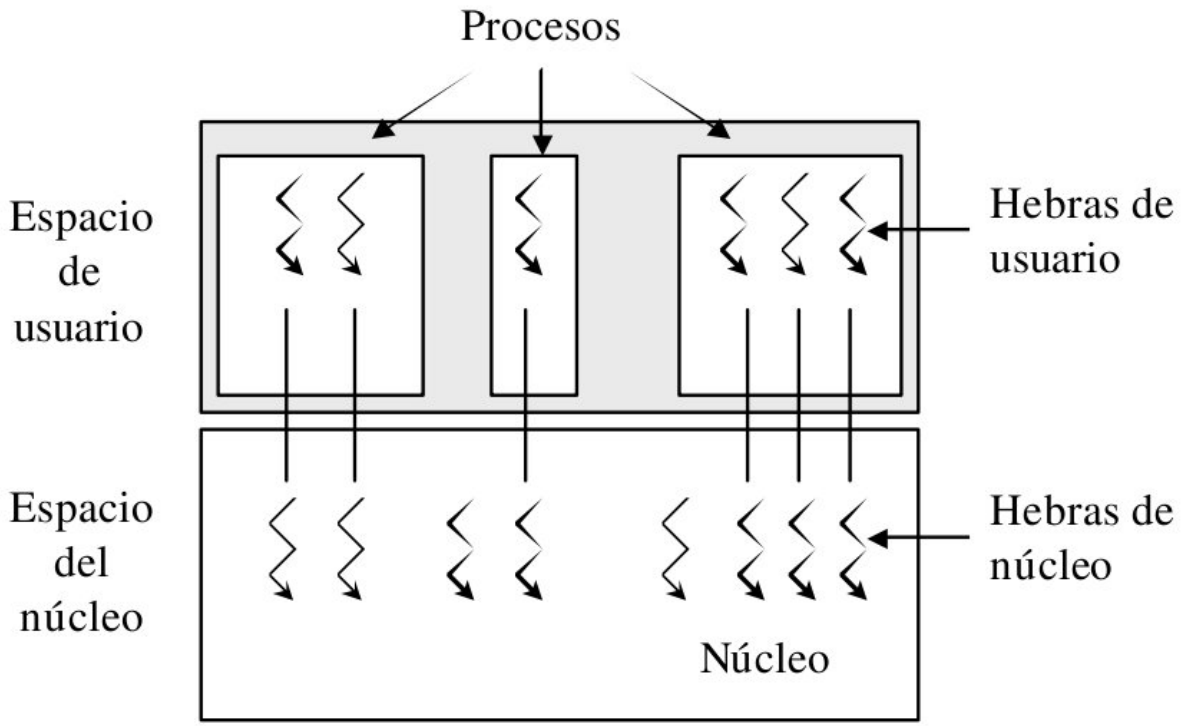
Ejemplos:

- La **biblioteca** de procesos de peso ligero (LWP o light weighted processes) del sistema operativo **SunOS**.
- Los **sistemas** operativos **actuales** no admiten este **esquema**.

Implementación a nivel de núcleo.

Características:

- Correspondencia **Una** a **Una**.
- Cada **hebra** de usuario se vincula a una **única** hebra del **núcleo**.



Implementación a nivel de núcleo.

Ventajas:

- Permite **obtener** todos los **beneficios** de aportan las hebras.

Inconvenientes:

- **Gestión** de hebras más **costosa** que en el caso de la implementación a nivel de usuario.
- **Número** máximo de **hebras** teóricamente **inferior** al de una implementación a nivel de **usuario**.

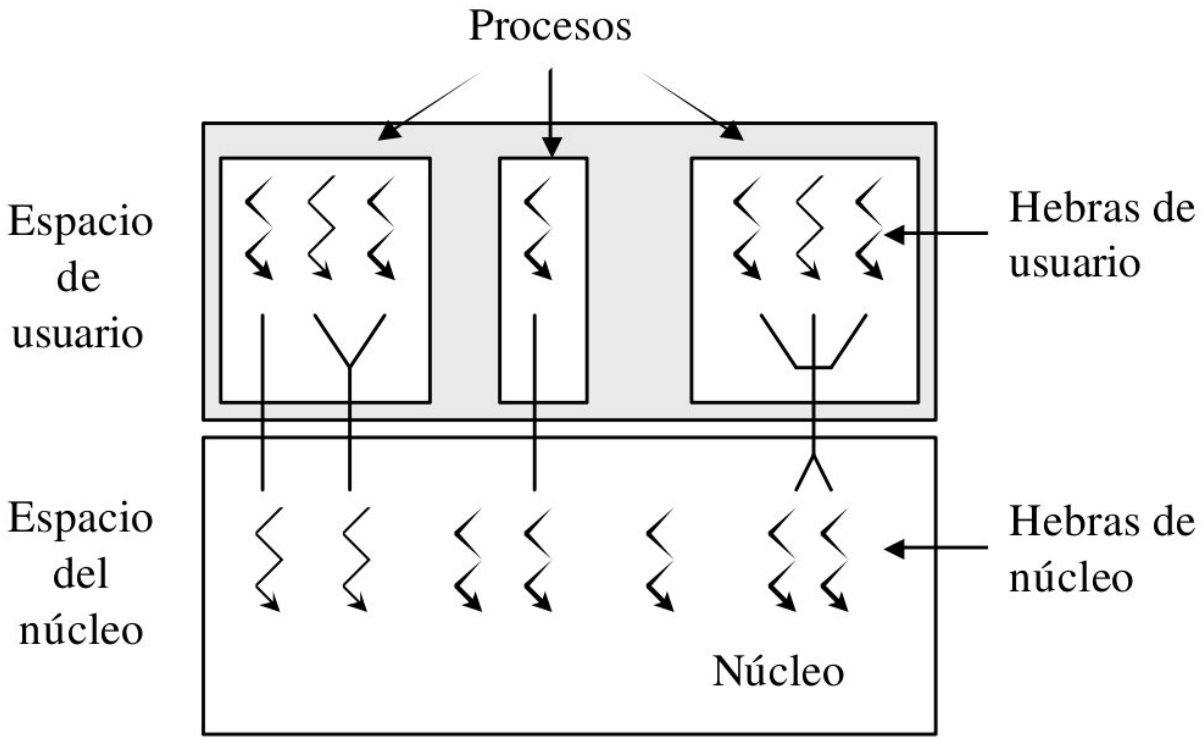
Ejemplos:

- **Windows** NT/2000.
- **Linux**.

Implementación híbrida.

Características:

- Correspondencia **Muchas** a **Muchas**.
- Existe **varias** formas de **vincular** las hebras de **usuario** con las hebras del **núcleo**.



Implementación híbrida.

Ventajas:

- **Combina** las **ventajas** de la implementación a nivel de **usuario** (**bajo** coste de **manipulación**) con las de la implementación a nivel de núcleo (no limita las **posibilidades** que permiten las **hebras**).

Inconvenientes:

- **Modelo** de programación **complejo** cuando la **vinculación** entre hebras de usuario y hebras de núcleo es responsabilidad del **programador**.

Ejemplos:

- SOLARIS.
- Tru64 (antiguo Digital UNIX).

Bibliotecas de Hebras.

Características:

- Los **lenguajes** más usados para programar sobre sistemas operativos (C/C++) **no** incorporan **hebras**.
- Pero puede hacer **uso** de las hebras del sistema operativo a través de una **biblioteca** de **funciones**
- En la actualidad existen dos **bibliotecas** de hebras de uso **mayoritario**:
  - Hebras *Win32*, usadas en los sistemas operativos de 32 bits de Microsoft.
  - Hebras **POSIX** (Pthreads), usadas en sistemas UNIX.

Bibliotecas de Hebras.

Comparación entre hebras Win32 y Pthreads:

- Windows NT/2000 fue diseñado **desde** el principio para que fuese **multihebra**.
- En UNIX se ha **producido** una **adaptación** de procesos tradicionales a procesos multihebra.

Consecuencia:

- En Windows NT/2000 las hebras están **integradas** de forma **natural** dentro del sistema
- En UNIX existen **conflictos** cuando se usan **características** que están pensadas para **procesos** tradicionales (ejemplos: bloqueo de ficheros, señales, etc.)

Índice

Ciclo de vida de una hebra

Los **estados** por los que puede pasar una hebra son:

- **Preparada**.
- En **ejecución**.
- **Bloqueada**: a la espera de un recurso.
- **Terminada**: Finalizada o cancelada.

Agrupamiento de rutinas I

Las **subrutinas** de Pthreads API pueden **agruparse** de manera **informal** en cuatro grupos:

- **Manipulación de hebras**: **rutinas** que trabajan **directamente** sobre las hebras, creándolas, cancelándolas, etc. También se incluyen aquellas que **modifican** o **consultan** los **atributos**.
- **Mutexs**: rutinas para **sincronización** que permiten la **creación**, destrucción, **bloqueo** y desbloqueo de un mútex (**mutual exclusion**). También se pueden modificar **atributos**.

Agrupamiento de rutinas II

- **Variables de condición**: rutinas que comunican **hebras** que comparten un **mútex**. Creación, destrucción, **wait** y **signal** son funciones incluidas en este grupo.
- **Sincronización**: rutinas que **permiten** la **lectura**/**escritura** de bloqueos y barreras.

Agrupamiento de rutinas III

Routine Prefix	Functional Group
pthread_	Threads themselves and miscellaneous subroutines
pthread_attr_	Thread attributes objects
pthread_mutex_	Mutexes
pthread_mutexattr_	Mutex attributes objects.
pthread_cond_	Condition variables
pthread_condattr_	Condition attributes objects
pthread_key_	Thread-specific data keys
pthread_rwlock_	Read/write locks
pthread_barrier_	Synchronization barriers

CREACION, DESTRUCCION

Y CANCELACION.

Listado 1: Creación de hebras

```
1 int pthread_create (
2 pthread_t *thread, //identificador de la hebra
3 pthread_attr_t *attr, //atributos
4 void *(*start)(void *),//función que ejecutará la hebra
5 void *arg);//argumentos para la hebra
```

- Devuelve **cero** si se ha creado **correctamente** y un valor **positivo** en caso **contrario**.

Listado 2: Destrucción de hebras

```
1 void pthread_exit(void *retval);
```

- **retval** es un puntero a un código de **retorno**
- También **finaliza** simplemente **retornando**.

Listado 3: Creación y destrucción hebras (creacion1.c)

```
1  #include <pthread.h>
2  #include <stdio.h>
3
4  static void *func (void*arg){
5     printf("%s\n", (char*)arg);
6     sleep(3);
7     pthread_exit(NULL);
8  } /* fin de la hebra */
9
10 main()
11 {
12    pthread_t thid1, thid2;
13    pthread_create(&thid1, NULL, func , "Hebra 1");
14    pthread_create(&thid2, NULL, func , "Hebra 2");
15    sleep(5);
16    printf("Salida de la hebra Main\n");
17 }
```

Listado 4: creación y destrucción hebras (creacion2.c)

```
1  #include <pthread.h>
2  #include <stdio.h>
3
4  static void *func (void*arg){
5     printf("%s\n", (char*)arg);
6     sleep(3)
7     return;
8  } /* fin de la hebra */
9
10 main()
11 {
12    pthread_t thid1, thid2;
13    pthread_create(&thid1, NULL, func , "Hebra 1");
14    pthread_create(&thid2, NULL, func , "Hebra 2");
15    sleep(5);
16    printf("Salida de la hebra Main\n");
17 }
```

Listado 5: Finalización de hebra main anticipada (creacion3.c)

```
1  #include <pthread.h>
2  #include <stdio.h>
3
4  static void *func (void*arg){
5     printf("%s\n", (char*)arg);
6     sleep(3);
7     pthread_exit(NULL);
8  } /* fin de la hebra */
9
10 main()
11 {
12    pthread_t thid1, thid2;
13    pthread_create(&thid1, NULL, func , "Hebra 1");
14    pthread_create(&thid2, NULL, func , "Hebra 2");
15    //sleep(5);
16    printf("Salida de la hebra Main\n");
17 }
```

Listado 6: Compartiendo la salida estándar (creacion4.c)

```
1  #include <pthread.h>
2  #include <stdio.h>
3
4  static void *func (void*arg){
5     sleep(3);
6     printf("%s\n", (char*)arg);
7     pthread_exit(NULL);
8  } /* fin de la hebra */
9
10 main()
11 {
12    pthread_t thid1, thid2;
13    pthread_create(&thid1, NULL, func , "Hebra 1");
14    pthread_create(&thid2, NULL, func , "Hebra 2");
15    pause();
16    printf("Salida de la hebra Main\n");
17 }
```

Listado 7: Vector de hebras y control de errores pthread_create (vectorhebras1.c)

```
1  #include <pthread.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  #define NUM_THREADS     10
6
7  void *PrintHello(void *threadid)
8  {
9     long tid;
10    tid = (long)threadid;
11    printf("Hola Mundo! Soy yo, la hebra #%ld!\n", tid);
12    pthread_exit(NULL);
13 }
```

Listado 8: Vector de hebras y control de errores pthread_create (vectorhebras1.c)

```
1  int main (int argc, char *argv[])
2  {
3    pthread_t threads[NUM_THREADS];
4    int rc;
5    long t;
6    for(t=0; t<NUM_THREADS; t++){
7       printf("En main: creando la hebra %ld\n", t);
8       rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
9        if (rc){
10          printf("ERROR; código de error %d\n", rc);
11          exit(0);
12       }
13   }
14      pause();
15     pthread_exit(NULL);
16 }
```

Listado 9: Número máximo de hebras por proceso

```
1  #include <pthread.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  #define NUM_THREADS     10
6
7  void *PrintHello(void *threadid)
8  {
9     long tid;
10    tid = (long)threadid;
11    printf("Hola Mundo! Soy yo, la hebra #%ld!\n", tid);
12    pthread_exit(NULL);
13 }
```

Control de devolución de recursos I

Las hebras pueden **operar** en dos **modos** diferentes para controlar la **devolución** de **recursos**:

- **Detached**: opera de modo **autónoma** y cuando termina **devuelve** sus **recursos** (identificador, pila, etc.)
- **Joinable**: en su terminación **mantiene** sus **recursos** hasta que otra hebra invoca a **pthread_join**() sobre el **identificador** de la hebra. También **cancelables** vía ese **identificador**. Se puede emular con puntos de sincronización (creándolas detached).

Control de devolución de recursos II

pthread_join: Esperar por la terminación de un hilo.

Listado 10: Terminación (join1.c)

```
1 #include <pthread.h>
2
3 int pthread_join(pthread_t tid, void **status);
```

- Esta función **suspende** la **ejecución** del hilo que la invoca hasta que el hilo **identificado** por el valor tid **finaliza**.
- Si status no es NULL, el valor devuelto por el hilo (**pthread_exit**) se almacena en la dirección indicada por **status**.
- El valor **devuelto** es o bien el argumento de la función **pthread_exit** o el valor **PTHREAD_CANCELED** si el hilo tid está cancelado.

Control de devolución de recursos III

- El hilo por el que se **espera** su **terminación** debe estar en estado **sincronizable** (joinable).
- La **espera** por la terminación de un **hilo** para el cual ya hay **otro** hilo **esperando**, genera un **error**.

Cancelación I

Listado 11: solicitar cancelación

```
1 int pthread_cancel(pthread_t tid);
```

Listado 12: configuración de estado

```
1 int pthread_setcancelstate(int state, int *oldstate);
```

Listado 13: configuración de tipo

```
1 int pthread_setcanceltype(int type, int *oldtype);
```

Listado 14: creación de un punto de cancelación

```
1 void pthread_testcancel(void);
```

Cancelación II

- La **cancelación** es el mecanismo por el cual un **hilo** puede **solicitar** la **terminación** de la ejecución de **otro**.
- Dependiendo de la **configuración** del hilo al que se solicita su cancelación, puede **aceptar peticiones** de cancelación (**PTHREAD_CANCEL_ENABLE**, estado por defecto) o rechazarlas (**PTHREAD_CANCEL_DISABLE**).
- En caso de **aceptar** peticiones de cancelación, un hilo puede **completar** la **cancelación** de dos formas diferentes: de forma asíncrona (**PTHREAD_CANCEL_ASYNCHRONOUS**), o de forma diferida (**PTHREAD_CANCEL_DEFERRED**, valor por defecto) hasta que se alcance un punto de cancelación.

Cancelación III

- Un punto de cancelación (**cancelation point**) es un punto en el **flujo de control** de un hilo en el que se comprueba si hay **solicitudes** de **cancelación** pendientes
- Cuando un hilo **acepta** una petición de cancelación, el hilo **actúa** como si se hubiese realizado la siguiente invocación **pthread_exit (PTHREAD_CANCELED)**.

Paso de Argumentos

- La **limitación** de pthread_create() es que permite **únicamente** pasar un **argumento** a cada **hebra**.
- Cuando se requiere pasar **más** de un **argumento**, esta limitación se supera creando una **estructura** que contiene todos los **argumentos**.
- Entonces, se pasa un **puntero** a la estructura como **argumento** de **pthread_create()**.
- Todos los **argumentos** se deben pasar por **referencia** y con el **cast** $(void*)$.

IDENTIFICACION.

Identificadores de hebra

Listado 15: Identificación

```
1 pthread_t pthread_self(void);
```

- Cada hebra tiene un **único identificador** (thread ID).
- Los **identificadores** de hebra son del tipo **pthread_t** que generalmente es un valor de tipo **entero sin signo**.
- En los procesos de **depuración** es habitual **imprimir** dicho identificador.

Listado 16: Igualdad

```
1 int pthread_equal(pthread_t t1, pthread_t t2);
```

- Devuelve **cero** si son distintos.

Listado 17: Identificación de la hebra (identificacion1.c)

```
1  #include <pthread.h>
2  #include <stdio.h>
3
4  static void *func (void*arg){
5  printf("%s e id %u\n", (char*)arg, (unsigned)pthread_self());
6  pthread_exit(NULL);
7  } /* fin de la hebra */
8
9  main()
10 {
11    pthread_t thid1, thid2;
12    pthread_create(&thid1, NULL, func , "Hebra 1");
13    pthread_create(&thid2, NULL, func , "Hebra 2");
14    pause();
15    printf("Salida de la hebra Main\n");
16 }
```

ATRIBUTOS.

Creación y Destrucción

Listado 18: Creación

```
1 int pthread_attr_init(pthread_attr_t *attr);
```

Listado 19: Destrucción

```
1 int pthread_attr_destroy(pthread_attr_t *attr);
```

- **pthread_attr_init** inicializa el **objeto** de atributos de un hilo **attr** y establece los valores por defecto.
- Posteriormente, este **objeto**, con los atributos por defecto de un hilo, se puede utilizar para **crear** múltiples **hilos**.
- **pthread_attr_destroy**, **destruye** el objeto de atributos de un hilo, **attr**, y éste no puede volver a utilizarse hasta que no se vuelva a **inicializar**.

Establecimiento y consulta I

- **pthread_attr_get/setxxxxx**: Establecimiento/Consulta **atributos particulares** de un **objeto** con los **atributos** de un hilo.

Listado 20: setters y getters atributos

```
1 #include <pthread.h>
2 int pthread_attr_getstacksize(const pthread_attr_t *restrict attr,
3 size_t *restrict stacksize);
4
5 int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);
```

Establecimiento y consulta II

- **pthread_attr_get/setxxxxx**: Establecimiento/Consulta **atributos particulares** de un **objeto** con los **atributos** de un hilo.

Listado 21: setters y getters atributos

```
1  #include <pthread.h>
2  int pthread_attr_setdetachstate (pthread_attr_t *attr,
3  int detachstate);
4  int pthread_attr_getdetachstate (const pthread_attr_t *attr,
5  int *detachstate);
6  int pthread_attr_setschedpolicy (pthread_attr_t *attr,
7  int policy);
8  int pthread_attr_getdetachstate (const pthread_attr_t *attr,
9  int *policy);
10 int pthread_attr_setscope(pthread_attr_t *attr,
11 int contentionscope);
12 int pthread_attr_getscope(const pthread_attr_t *attr,
13 int *contentionscope);
```

Atributos Definibles I

- **Tamaño** de la pila. Cada hilo tiene una **copia privada** de sus **parámetros** iniciales y de las variables **locales** de la función que ejecuta almacenados en su pila particular.
- !**CUIDADO**: Problemas de **portabilidad¡**. Sólo si el sistema define el símbolo **POSIX_THREAD_ATTR_STACKSIZE** (getconf -a).

Atributos Definibles II

- **Dirección** de la pila. Se especifica una región de memoria para la pila de una hebra. Si el sistema define el símbolo **POSIX_THREAD_ATTR_STACKADDR**.
- La región como mínimo tiene que tener el tamaño de **PTHREAD_STACK_MIN**.
- Un factor **crítico** es conocer hacia **donde** crece la pila en **memoria**, si hacia **arriba** o hacia **abajo**. Cuidado con definir la **misma** zona para dos **threads** distintos.

Atributos Definibles III

Control de **devolución** de recursos:

- **detachstate**: controla si otra hebra podrá esperar por la terminación de esta hebra (pthread_join).
  - **PTHREAD_CREATE_JOINABLE** (valor por defecto)
  - **PTHREAD_CREATE_DETACHED**

Control de **políticas** del **planificador**:

- **schedpolicy**: controla cómo se **planificará** el **hilo**.
  - **SCHED_OTHER** (valor por defecto, planificación normal + no tiempo real)
  - **SCHED_RR** (Round Robin + tiempo real + privilegios root)
  - **SCHER_FIFO** (First In First Out + tiempo real + privilegios root)

Atributos Definibles IV

Scope: controla a que **nivel** es **reconocido** el **hilo**

- **PTHREAD_SCOPE_SYSTEM** (valor por defecto, el hilo es reconocido por el núcleo)
- PTHREAD_SCOPE_PROCESS (no soportado en la implementación LinuxThreads de hilos POSIX)

Política de planificación y ámbito de los procesos I

- La **teoría** de **planificación** proporciona diversas soluciones para **conseguir** un **comportamiento** temporal **predecible** junto a un alto nivel de utilización de la CPU.
- Una de las soluciones más simples y mejor conocidas es la **planificación expulsora** con **prioridades** fija. En esta política cada **thread** tiene una **prioridad** asignada, y el **planificador** elige para su ejecución aquel thread que tiene **mayor** prioridad, de entre los que están listos para ejecutar.
- POSIX RT especifica la planificación expulsora con prioridades fijas para los threads y también para los procesos, con **dos variaciones** que pueden ser seleccionadas a nivel de cada thread, como dos **políticas** de planificación **diferentes**.

Política de planificación y ámbito de los procesos II

En tiempo real se gestionan prioridades distintas de cero con dos políticas de planificación: **SCHED_FIFO** y **SCHED_RR**.

- **SCHED_FIFO**: política de planificación **expulsora** con prioridades que usa orden **FIFO** para determinar el orden en el que se **ejecutan** los threads de la misma **prioridad**. La hebra se **ejecuta** hasta que o bien es **bloqueada** por una solicitud de **E/S** o está **preparada** para **ejecución** otra de mayor prioridad. Se pondrá al **final** de la **cola** de prioridad.

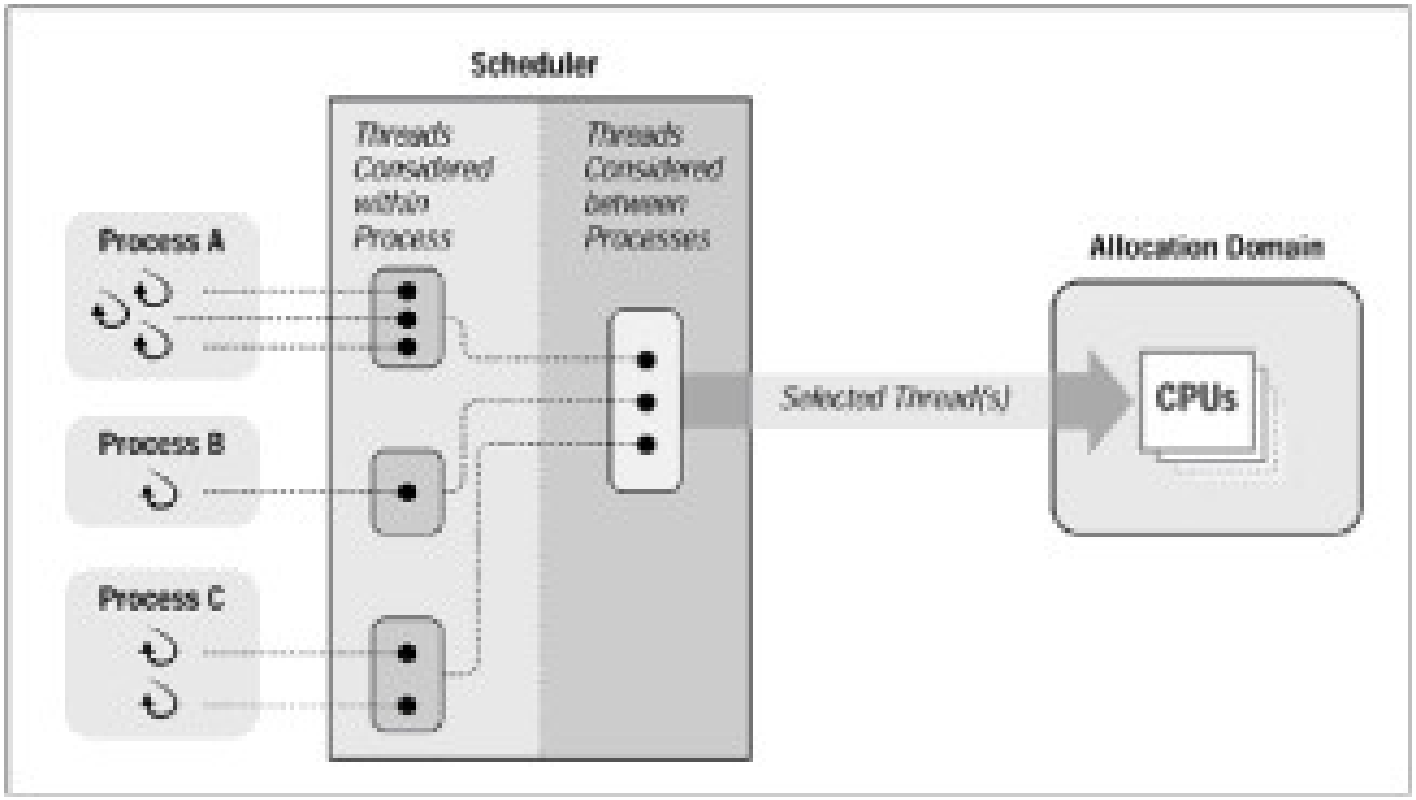
Política de planificación y ámbito de los procesos III

- **SCHED_RR**: política de **planificación** expulsora con **prioridades** que usa un esquema **cíclico** para planificar **threads** de la misma prioridad. Se ejecuta durante un **quantum** de tiempo.
- **SCHED_OTHER** se definió para permitir la **compatibilidad** con implementaciones **preexistentes** y que se utiliza para prioridades **cero** (no requieren RT).

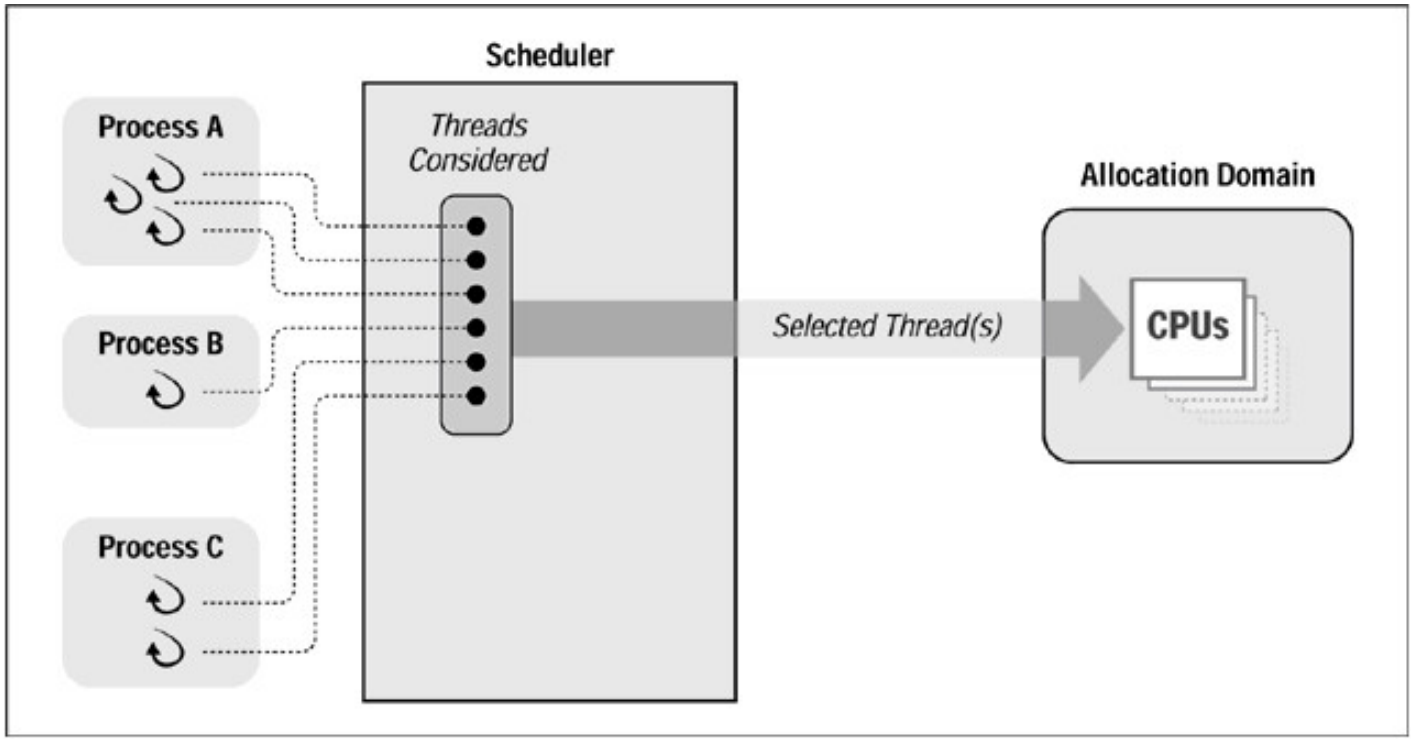
Política de planificación y ámbito de los procesos IV

- Existen las mismas **políticas** que para los **procesos**.
- Los **atributos** de **planificación** se pueden especificar al **crear** el **hilo** en el objeto de atributos.
- Se puede seleccionar entre **dos ámbitos** de planificación:
  - Ámbito de **proceso**: un planificador de segundo nivel planifica los hilos de cada proceso **PTHREAD_SCOPE_PROCESS**.
  - Ámbito de **sistema**: los hilos se planifican como los procesos pesados **PTHREAD_SCOPE_SYSTEM**.

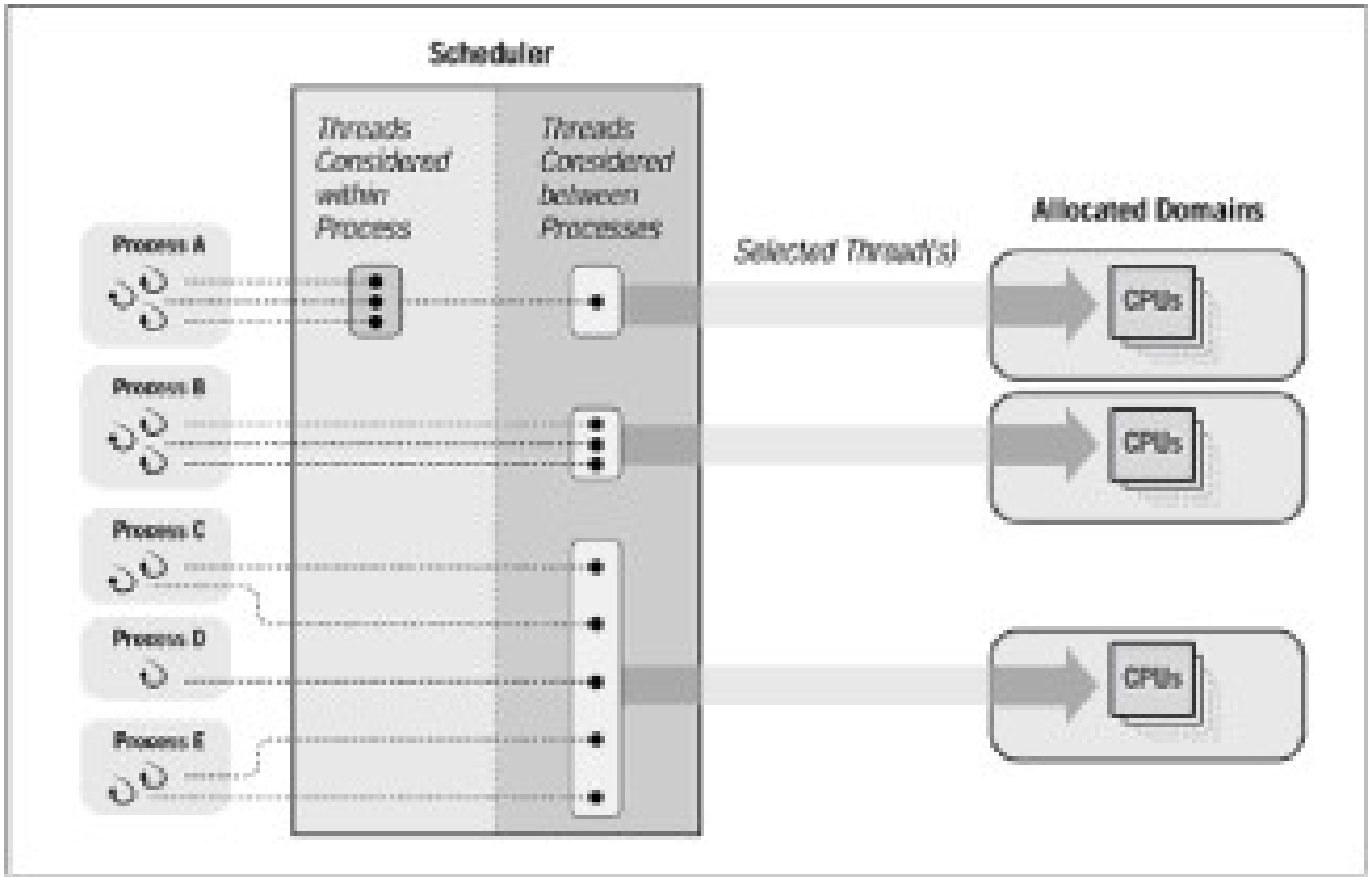
Política de planificación y ámbito de los procesos V



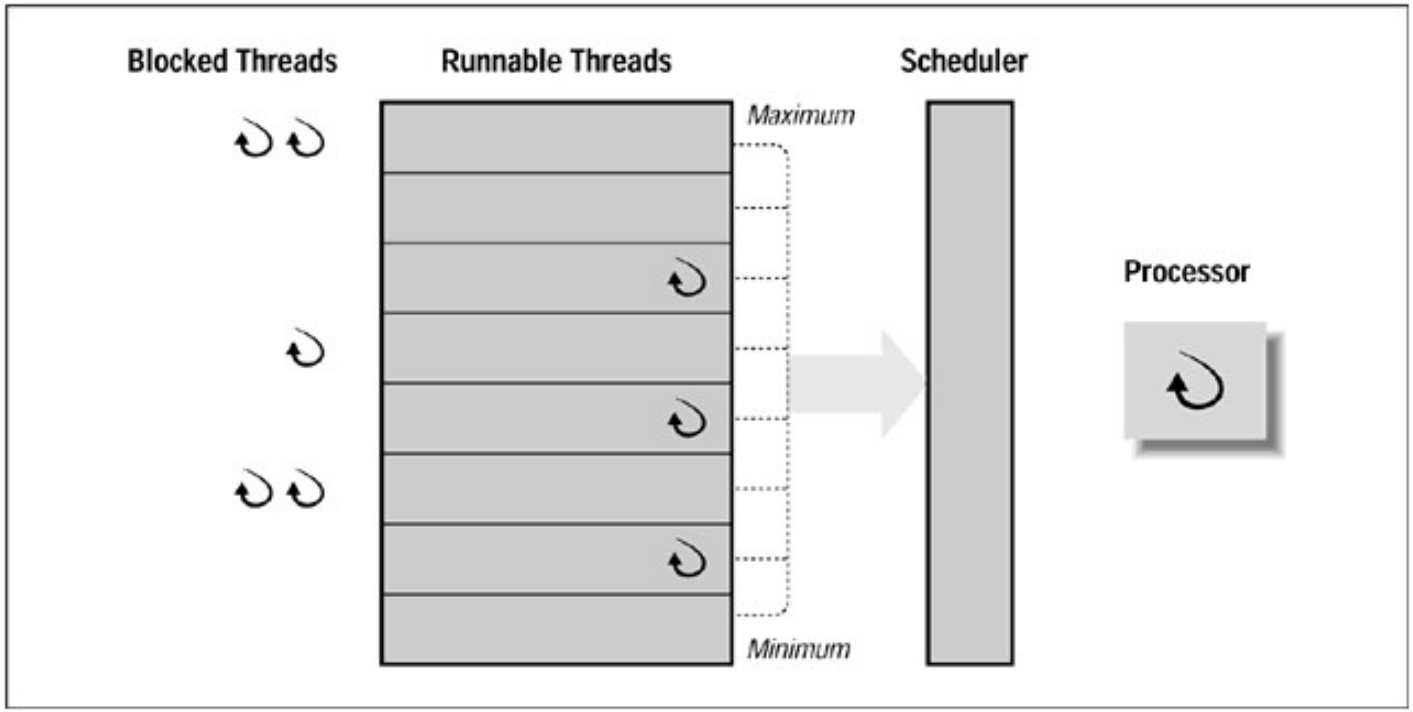
Política de planificación y ámbito de los procesos VI



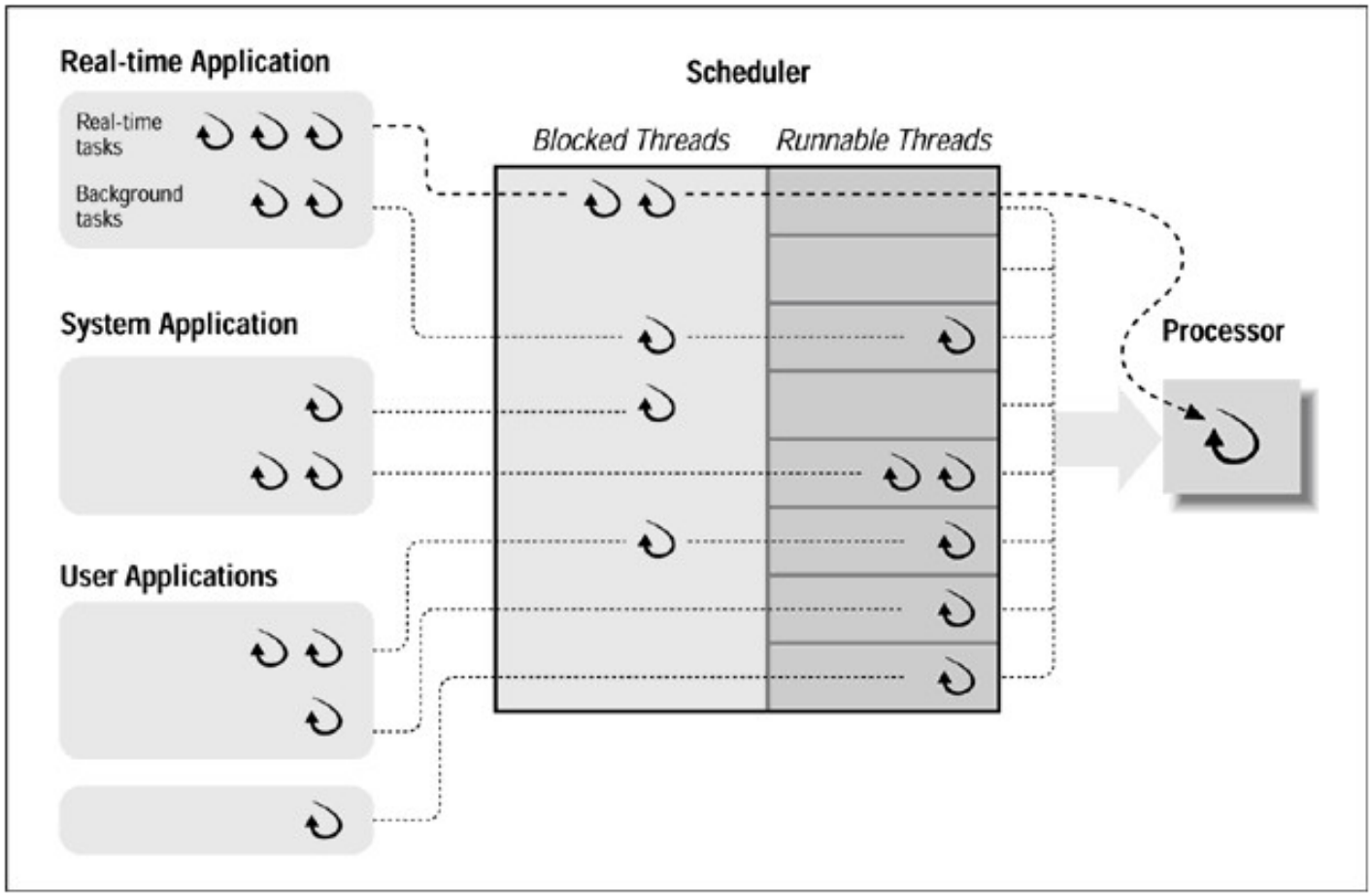
Política de planificación y ámbito de los procesos VII



Política de planificación y ámbito de los procesos VIII



Política de planificación y ámbito de los procesos IX



Gestión de prioridades

Listado 22: Establecer prioridad

```
1  #define HIGHPRIORITY 10
2  pthread_attr_t attr;
3  pthread_t thread;
4  struct sched_param param;
5
6  void *func (void *arg){ // Prototipo ejemplo de func
7  pthread_exit(NULL);
8  }
9  int main (void){
10 if (pthread_attr_init(&attr))
11   {perror("No puedo inicializar atributos"); exit(-1);}
12 if (pthread_create(&thread, &attr, func, NULL))
13   { perror("No puedo crear el thread"); exit(-1);}
14 if(pthread_attr_setschedpolicy (&attr, SCHED_FIFO))
15   perror("No puedo cambiar el planificador");
16 if (pthread_attr_getschedparam(&attr, &param))
17   perror ("No puedo obtener los parámetros");
18 else{
19   param.sched_priority = HIGHPRIORITY;
20   if (pthread_attr_setschedparam(&attr, &param))
21   perror("No puedo cambiar la prioridad");
22 }
23 }
```

SINCRONIZACION.

- Se puede **sincronizar** asegurando el acceso en **exclusión mutua** a variables.
- Se pueden usar **mutexs**, equivalentes a **semáforos binarios**.
- También se pueden usar **variables condición**, que comunican información sobre el estado de datos compartidos.
- ¿Por qué no se usan **semáforos** o sistemas **paso de mensajes**?

Mutex

- Un **mutex** es un mecanismo de **sincronización** indicado para procesos **ligeros**.
- Es un **semáforo binario** con dos operaciones **atómicas**:
  - **lock(m)**: Intenta bloquear el mutex; si el mutex ya está bloqueado el **proceso** se **suspende**.
  - **unlock(m)**: **Desbloquea** el **mutex**; si existen procesos bloqueados en el **mutex** se **desbloquea** a uno.

Manejo de Mutex

Listado 23: Inicialización

```
1 int pthread_mutex_init(pthread_mutex_t *mutex,
2 const pthread_mutexattr_t *attr);
```

Listado 24: Destrucción

```
1 int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

Listado 25: Bloqueo

```
1 int pthread_mutex_lock(pthread_mutex_t *mutex);
```

- **Bloquea un mutex**. No se puede bloquear un **mutex** cuando el **thread** ya lo tiene **bloqueado**. Esto devuelve un **código** de **error**, o se bloquea el thread.

Manejo de Mutex

Listado 26: Bloqueo

```
1 int pthread_mutex_trylock(pthread_mutex_t *mutex);
```

- **Bloquea un mutex si no está bloqueado**. Si el mutex está bloqueado, el thread que llama no se bloquea y la función regresa el código de error: **EBUSY**.

Listado 27: Desbloqueo

```
1 int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

- **Para desbloquear mutex**. **No** se puede **desbloquear** un mutex **desbloqueado**, ni un mutex bloqueado por otro thread, pues los mutex pertenecen a los threads que los bloquean.

Inicialización del Mutex

- Se requiere una **variable global** de tipo **pthread_mutex_t**.
- Usando la macro **PTHREAD_MUTEX_INITIALIZER** se puede iniciar el valor de la variable **Mutex** en tiempo de **compilación**.

Listado 28: Inicialización en compilación

```
1 pthread_mutex_t counter_mtx= PTHREAD_MUTEX_INITIALIZER;
```

Inicialización del Mutex

- Mediante la función **pthread_mutex_init()**, la inicialización se realiza en tiempo de **ejecución**.

Listado 29: Inicialización en ejecución

```
1 pthread_mutex_t counter_mtx;
2 //asignación valor de counter_mtx
3 pthread_mutex_init(& counter_mtx, NULL);
```

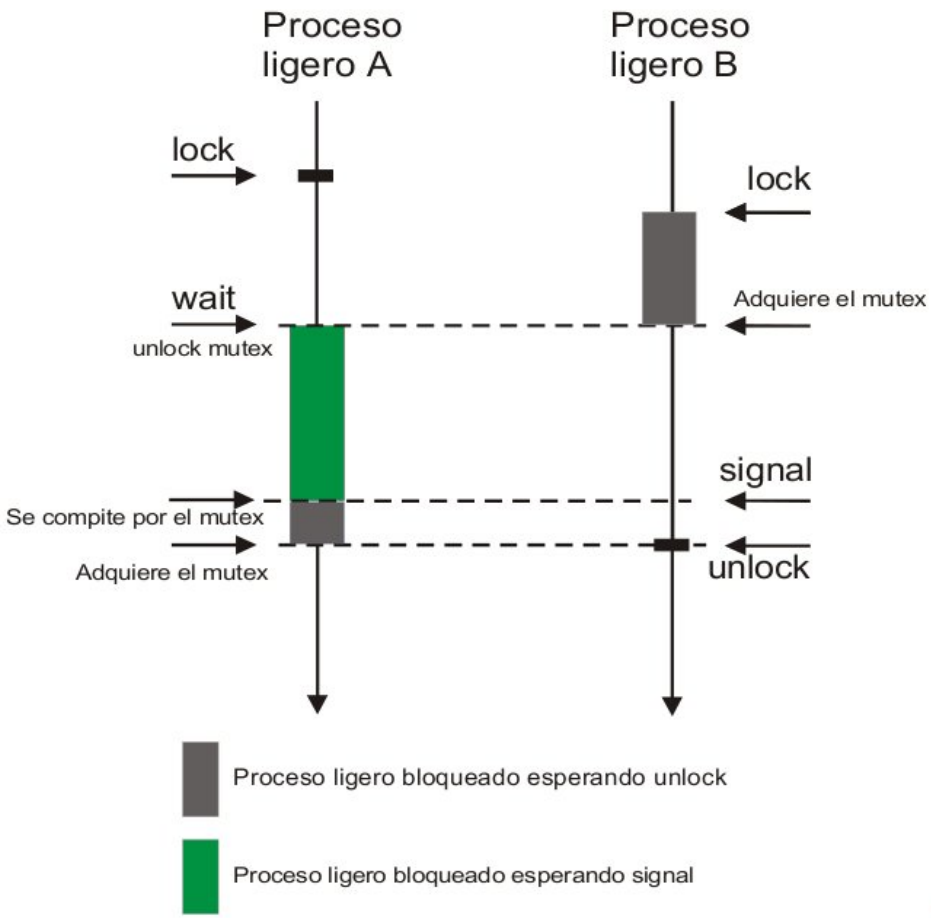
Variables de condición

- Es un **objeto** de **sincronización** que permite **bloquear** a un hilo hasta que otro decide **reactivarlo**. Las **operaciones** que realiza son:
  1. **Esperar una condición**: un hilo se **suspende** hasta que otro **señaliza** la condición. En este punto se comprueba la **condición** y el **proceso** se repite si la condición es falsa.
  2. **Señalizar una condición**: se avisa a uno o más hilos **suspendidos**.
  3. **Broadcast**: se **reactivan** todos los hilos **suspendidos** en la condición.

Uso de variables condición y mutex

- Las variables de sincronización están siempre **asociadas** a un **mutex**.
- Conveniente ejecutarlas **entre lock** y **unlock**.
- Dos operaciones **atómicas**:
  - **pthread_cond_wait()**: **bloquea** la hebra que la ejecuta y le **expulsa** del **mutex**.
  - **pthread_cond_signal()**: **desbloquea** a **una** o **varias** hebras que se encuentran **suspendidas** en la variable de condición. La **hebra** que se **despierta compite** de nuevo por el **mutex**.

Uso de variables condición y mutex



Inicialización de las variables de condición

- Se requiere una **variable global** de tipo **pthread_cond_t**.
- Mediante la macro **PTHREAD_COND_INITIALIZER** se puede iniciar el valor de la variable condición en tiempo de **compilación**.

Listado 30: Inicialización

```
1 static pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

- Mediante la función **pthread_cond_init()** en tiempo de **ejecución**:

Listado 31: Inicialización

```
1 pthread_cond_t cond;
2 pthread_cond_init(& cond, NULL);
```

Espera sobre las variables de condición

Listado 32: wait

```
1 int pthread_cond_wait(pthread_cond_t *cond,
2 pthread_mutex_t *mutex);
```

- Atómicamente ejecuta **pthread_mutex_unlock** sobre mutex y **suspende** al hilo invocante en la variable **cond**.
- Al despertarse, realizará **automáticamente** un **pthread_mutex_lock** sobre mutex.

Espera sobre las variables de condición

Listado 33: timedwait

```
1 int pthread_cond_timedwait(pthread_cond_t *cond,
2 pthread_mutex_t *mutex, const struct timespec *abstime);
```

- Actúa como **pthread_cond_wait**, salvo que la hebra se suspende como mucho hasta que se alcanza el instante de tiempo **abstime**.
- Si se alcanza **abstime** antes de que otra hebra ejecute un **signal/broadcast** sobre la variable de condición, la hebra dormida se **despierta** y devuelve el **error ETIMEDOUT**.

Avisos sobre las variables de condición

Listado 34: signal

```
1 int pthread_cond_signal(pthread_cond_t *cond);
```

- Selecciona la **hebra** más **prioritaria** suspendida en cond y la **despierta**.

Listado 35: broadcast

```
1 int pthread_cond_broadcast(pthread_cond_t *cond);
```

- **Despierta** todas las **hebras** que pueda haber **suspendidas** en **cond**.
- Ambas órdenes **no** tienen **efecto** si **no** hay hebras **suspendidas** en cond.

Conclusiones

- El uso de hebras **facilita** y **potencia** enormemente los procesos de **programación**, pero son **peligrosas**.
- Se requiere prestar **mucha atención**, hasta en los mas **pequeños detalles**, o se puede fácilmente terminar con un **código incorrecto** (que no funciona siempre y/o se cuelga).
- Se debe tener **especial cuidado** con las **librerías**.
- Si una función utiliza **variables estáticas (o memoria global)**, **no** es **segura** su **invocación** desde hebras.
- La hebras POSIX (**pthreads**) incorporan funcionalidad de **Exclusión Mutua** y **Variables Condición**.

Índice

1. **Programming with POSIX Threads**. David R. Butenhof. Addison Wesley.
2. **POSIX Threads Programming**. Blaise Barney, Lawrence Livermore National Laboratory. `https://computing.llnl.gov/tutorials/pthreads/`
3. Chapter 4 - Managing Pthreads. **Pthreads Programming**. Bradford Nichols, Dick Buttlar and Jacqueline Proulx Farrell `http://maxim.int.ru/bookshelf/PthreadsProgram/htm/r_37.html`

Tema 7. Hebras de Ejecución.

Sistemas Operativos II

Universidad de Castilla-La Mancha