Kernel: CPU y Memoria

Diseño y Evaluación de Configuraciones

Curso 2012-13

Miguel Telleria de Esteban

`telleriam AT unican.es`

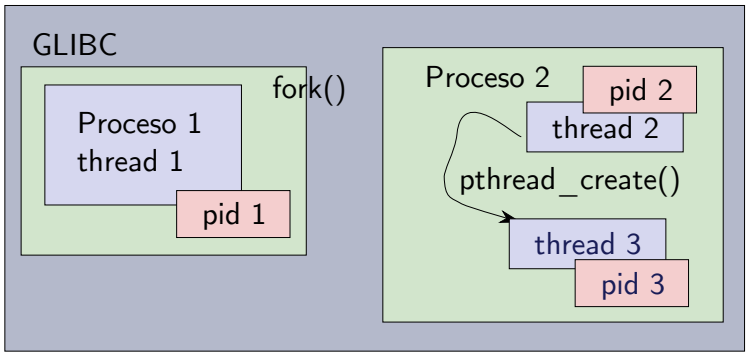
Computadores y Tiempo Real

http://www.ctr.unican.es

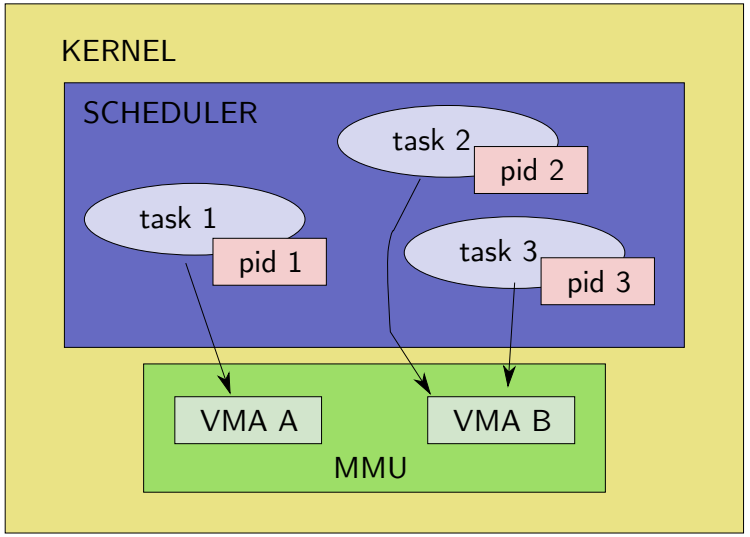
Uso de CPU y planificación

Procesos y Threads

POSIX API: fork(), getpid(), pthread_create()



SYSCALLS: clone, getpid, gettid



- GNU/Linux soporta procesos y threads
  - **Procesos:** Espacio de memoria propio
  - **Threads**: Espacio de memoria compartido
- Para el kernel procesos y threads se planifican indistintamente:
  - PID
  - Política de planificación
  - Prioridad
  - Afinidad
  - Estadísticas
  - Relojes de ejecución

Planificador de Linux

Desde el 2.6.23 (2007) CFS: Completely Fair Scheduler

Jerarquia I Real Time:

- Baja latencia (para alta prioridad)
- Bajas prioridades pueden no ejecutar
- SCHED_RR: Timeslice fija para todos
- SCHED_FIFO: Hasta que no termine no para

SCHED_FIFO | SCHED_RR

Jerarquia II: Timeshare

SCHED_OTHER

- Alto throughput “para todos”
- Todas las tareas tiene garantizada la CPU
- SCHED_OTHER: Round-robin con timeslice inversamente proporcional al nice

TASKs (rbtree)

Multicores: Balanceo de carga periódico segun afinidad

Parámetros de planificación de CPU en Linux

- Policy: Política de planificación
  - SCHED_OTHER o SCHED_NORMAL: Por defecto, round-robin con timeslice variable
  - SCHED_RR: POSIX Real-Time: Round robin con timeslice fija
  - SCHED_FIFO: POSIX Real Time: prioridades fijas

  Las políticas SCHED_FIFO y SCHED_RR tienen prioridad sobre SCHED_OTHER.
- Nice: Avaricia del proceso
  - Sólo se usa en SCHED_OTHER, se ignora para real time
  - Oscila entre: (-20 valor más prioritario y 19), por defecto 0
- rt-priority: Nivel de preempción
  - Se usa en SCHED_RR y SCHED_FIFO, ignorada en SCHED_OTHER
- Máscara de afinidad
  - Define en que CPU(s) un proceso admite ser ejecutado

Nice y nivel de preempcion entremezcladas

- 100 – 139 valor inverso a la prioridad (100 más prioritario)
  - **Donde:** PROCFS**:** /proc/<pid>/sched (prio).
  - **NOTA**: Cuando se cambia a RT este valor permanece igual (pero es ignorado)
- 0 – 99 creciente en prioridad (sólo en SCHED_FIFO, SCHED_RR)
  - **Donde**
    - chrt -p <pid>
    - API's de POSIX (set_sched_param)
- -20 a 19 (SCHED_OTHER) orden inverso a prioridad
  - **Donde:**
    - Htop,
    - chrt -p <pid>
    - Gnome-system-monitor
- 0 – 39 (SCHED_OTHER) orden inverso a la prioridad
  - **Donde**:
    - top

Asignación de prioridad y política desde consola

- Comando **chrt**
  - Vemos en que prioridad corre: Usamos **-p para el pid**
    - chrt -p 16951

```
pid 16951's current scheduling policy: SCHED_OTHER
pid 16951's current scheduling priority: 0
```

  - Pasamos a SCHED_FIFO prioridad alta 80/90 **-p para prioridad**
    - chrt –fifo -p 80 169510
    - chrt -p 16951

```
pid 16951's current scheduling policy: SCHED_FIFO
pid 16951's current scheduling priority: 80
```

  - Vuelta a SCHED_OTHER con nice a 0 (valor por defecto)
    - chrt –other -p 0 16951

Afinidad

- La afinidad se representa mediante una máscara binaria
  - CPU 0: 1 CPU 1: 2 CPU 3: 4 CPU 4: 8 ...
  - Valor 3: CPU's 0 y 1
- Asignación y obtención desde consola (parecido al chrt)
  - Se obtiene con taskset **-p <pid>**
    - Taskset -p 16951

```
pid 16951's current affinity mask: f
```

  - Se asigna con taskset -p <mascara> <pid>
    - taskset -p 1 16951
    - Taskset -p 16951

Estados de un thread en el tiempo

- En un instante dado un thread puede estar:
  - **R** Running
  - **S** Sleep: Suspendido en un wait() o sleep(). Se le puede interrumpir
  - **D** Uninterruptible sleep (tipicamente disk IO). Sólo se le interrumpe con kill -9
  - **T** Traceado en debug
  - **Z** Zombie, difunto:
    - A la espera de que el padre les espere (o muera y sean esperados por init)
- Además puede tener las siguientes añadidos (ej ps):
  - < Valor nice negativo, (avaricioso)
  - N Valor nice positivo, (generoso)
  - L Mantiene páginas en memoria bloqueadas
  - l Tiene pthreads asociados
  - s Lider de sesión: Proceso que controla el terminal
  - + Pertenece al foreground-process-group

CPU ocupada sin ningún thread de usuario

- Existen situaciones en el que el kernel tiene la CPU ocupada sin ejecutar ningún código o servicio de usuario.
  - Atendiendo a una interrupción hardware (ej. timer)
    - Top half (irq ISR)
    - Bottom half (workqueue, tasklet, softirq)
  - Atendiendo a una interrupción software (ej page-fault + swap)
  - Ejecutando tareas de mantenimiento:
    - Actualizando páginas sucias de la page-cache al disco
    - Balanceando la carga de los multicores
  - Respondiendo a necesidades de emergencia
    - Liberando memoria de caches o de procesos
    - Usando la swap del disco
  - Realizando cambios de contexto
  - No realizando **nada (idle)**

Funcionamiento de la memoria

Paginación en Linux

- Hardware MMU (Memory management unit) provee:
  - Manejo en bloques de **páginas** (típicamente 4k en i386).
  - Mapeados de direcciones (i386: tablas PGPT, PMD, PTE)
  - Detección de **fallos de página** (acceso a páginas no mapeadas)
  - Permisos de lectura / escritura / ejecución
  - TLB: Cache de traducción de páginas
- El S.O. (linux) ha de programar todo ello:
  - Maneja un **bitmap de todas las páginas de memoria**.
  - Actualiza las tablas en los cambios de contexto: Espacios de memoria
  - Utiliza los fallos de página para
    - Detectar accesos prohibidos
    - Implementar paginación bajo demanda
    - Implementar cache memoria ↔ disco
    - Implementar espacio swap

Distribución de la memoria física en Linux

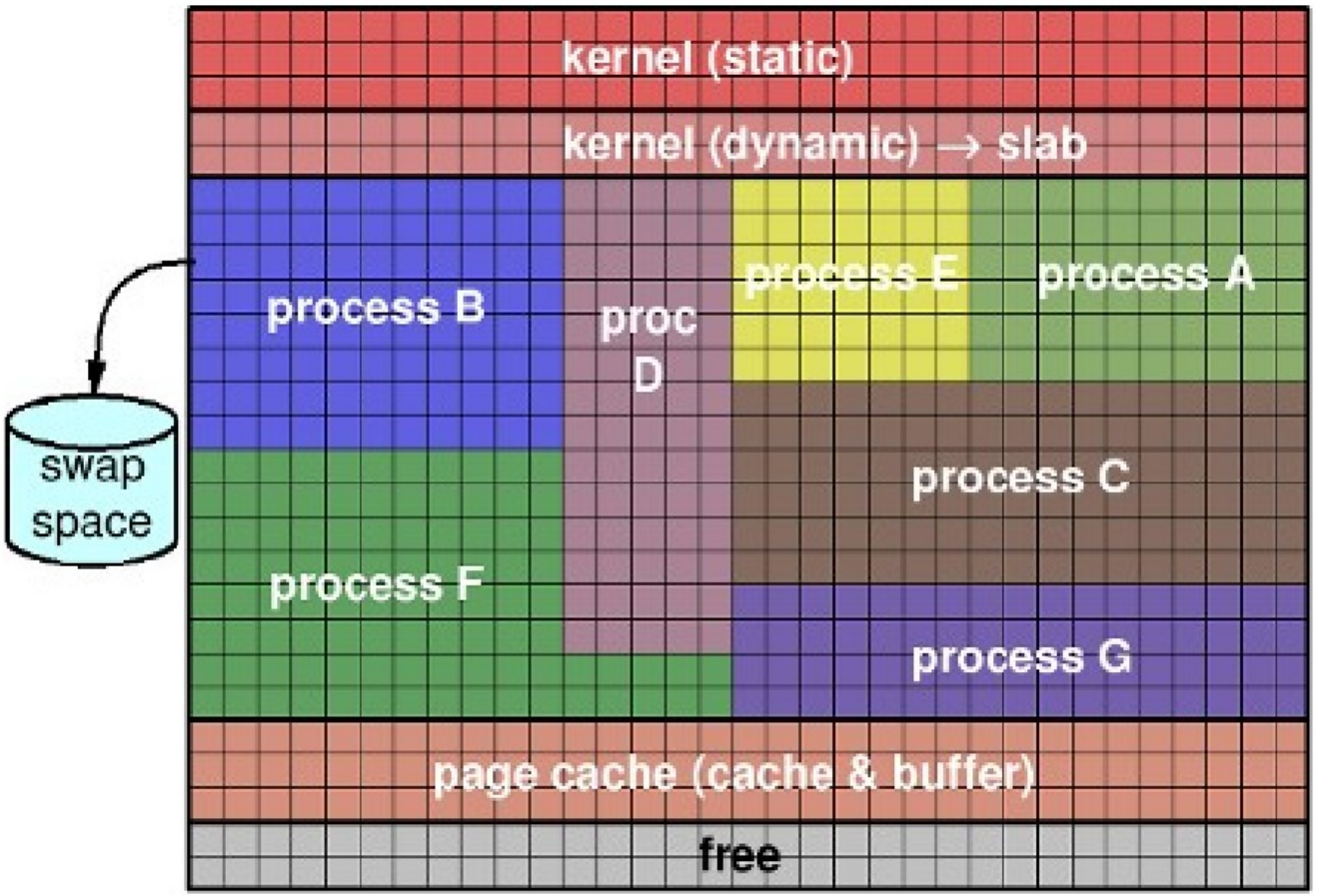
- Páginas para el kernel mismo:
  - El fichero vmlinuz se carga **permanentemente** en memoria
  - Los módulos se cargan bajo demanda (y se quedan permanentes mientras no se quite el módulo)
  - Losas (**slab)** estructuras pre-allocated (inodos, descriptores de procesos, página...)
    - Cada losa ocupa menos que una página, por lo que el kernel maneja su reuso).
    - Parte de estos objetos se cachean (inode cache, dentry cache)
- Páginas para los procesos (de usuario y de kernel)
  - Código ejecutable (**text)** (compartido con otros procesos)
  - Datos estáticos (**data)**
  - Stack y thread-level-storage **(bss)**
  - Heap
- Páginas de caché de disco (page_cache)
  - Caché: Contenido real de disco
  - Buffers: Metadata de los inodos (timestamps, permisos...)
- Pool de páginas libres
  - Para satisfacer rápidamente peticiones urgentes

Memoria asociada a un proceso

- El kernel define zonas de memoria VMA para los procesos
  - Puede estar mapeada a un fichero o ser anónima
  - Tiene asociados permisos de ejecución, lectura y/o escritura
  - Puede estar compartida o no con otros procesos

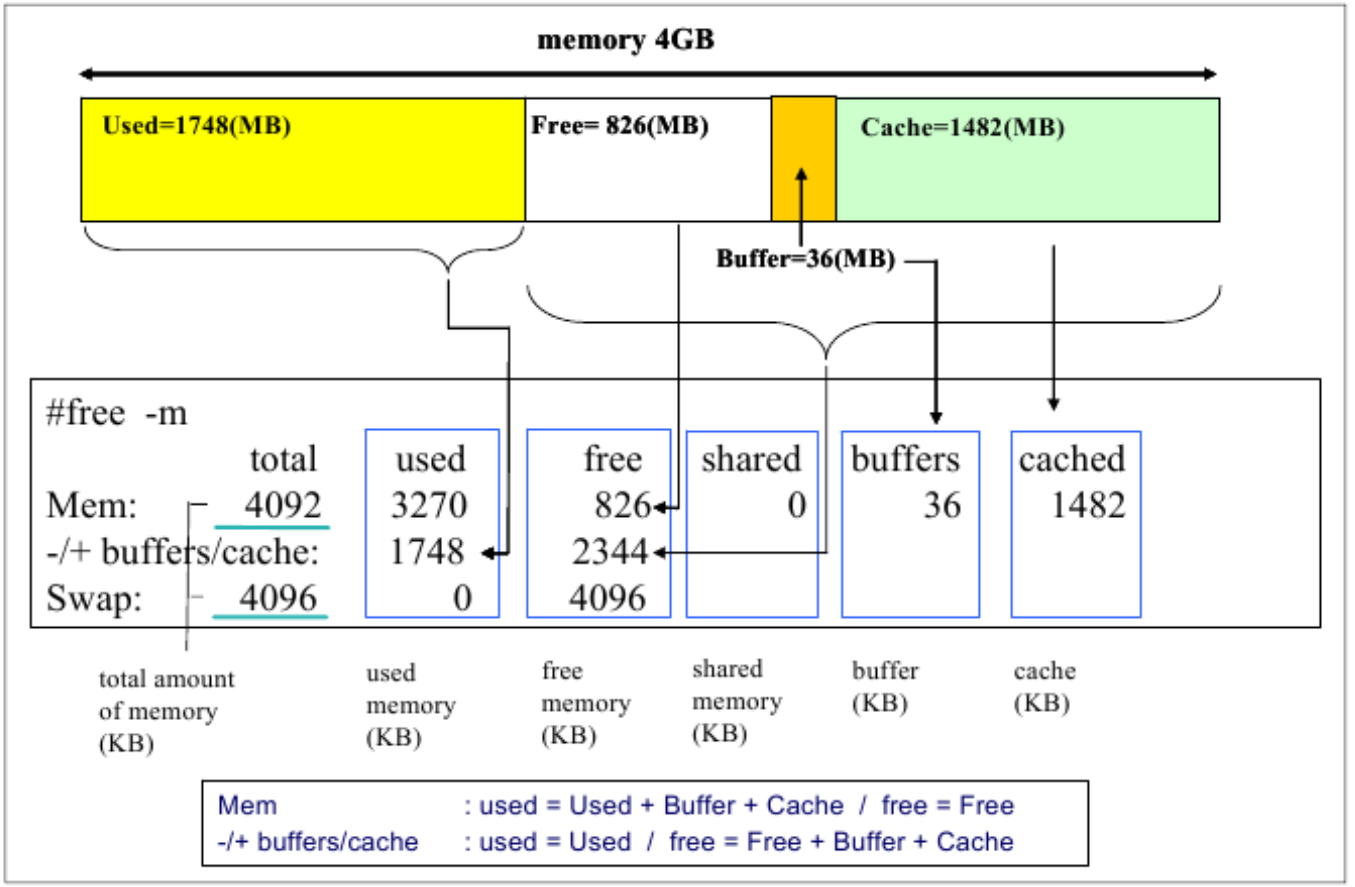
```
pmap 3070

08048000   3152K r-x--  /usr/bin/claws-mail
0835c000    160K rw---  /usr/bin/claws-mail
08384000     80K rw---    [ anon ]
0a372000  42372K rw---    [ anon ]
aad3e000   8620K r----  /usr/share/icons/hicolor/icon-theme.cache
ab5a9000  48536K r----  /usr/share/icons/gnome/icon-theme.cache
ae50f000      4K -----    [ anon ]
afd13000   8192K rw---    [ anon ]
b0513000    112K r--s-  /usr/share/mime/mime.cache
b0d9a000  48536K r----  /usr/share/icons/gnome/icon-theme.cache
b3d00000   2048K rw---    [ anon ]
b469b000   8192K rw---    [ anon ]
b4e9b000    484K r-x--  /usr/lib/claws-mail/plugins/vcalendar.so
b4f14000     40K rw---  /usr/lib/claws-mail/plugins/vcalendar.so
```

Linux intenta usar la máxima memoria



¡¡ Memoria sin usar es memoria desperdiciada !!

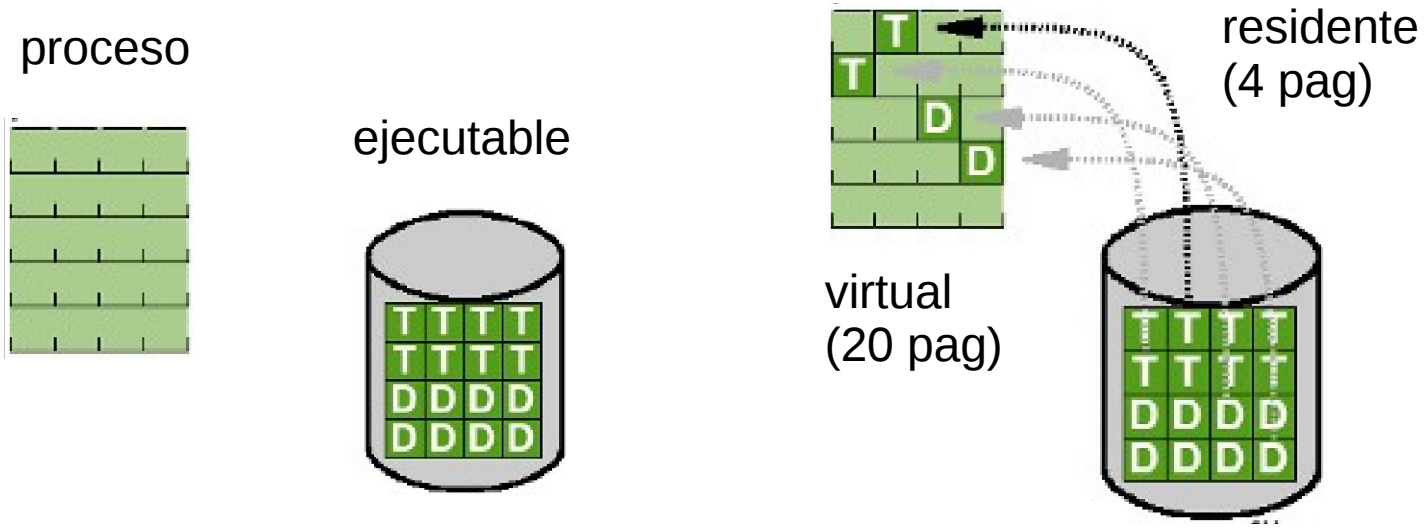
¿Cual es entonces la “memoria disponible”?



- Rule of thumb: Libre + page-cache

Memoria de proceso: virtual vs residente

- Linux utiliza la estrategia de paginación **bajo demanda**
  - malloc, cargar un ejecutable, fork... **no ocupan la memoria inmediatamente**



  - La memoria se va "rellenando" según se va **referenciando**
    - Cada referencia no mapeada provoca un **page-fault** (excepción software)
    - El kernel trata la excepción rellenando la página del proceso (o actualizando el mapeado)

Memoria swap

- La swap es un mecanismo para **proveer a los procesos** más memoria de la disponible físicamente.
- Es memoria residente de **procesos** mapeada en el disco
- Se usa en 2 contextos:
  - **Caso 1**: Cuando no podemos dar servicio a la demanda de memoria
  - **Caso 2**: Cuando un proceso se queda inactivo durante mucho tiempo (ej init)
- Existe un recurso más fuerte: El **Out-Of-Memory killer**

Memoria que no crea problemas de performance

- De más a menos favorable
  - Pool libre
  - Memoria residente recién liberada (que pasa al pool)
  - Page cache ya guardada en el disco y poco usada
  - Slabs poco usados
  - inodes, dentries de la cache con LRU bajo (que se traduce en slabs liberados)

- Estas gestiones de memoria ser realizan automáticamente por los threads del kernel.

Memoria que afecta a la performance

- De más a menos favorable
  - Page cache usada recientemente (pero aún limpia en disco)
  - inodes y dentries con LRU alto
  - Page cache sucia (hay que guardarla en disco primero)
  - Memoria residente de procesos limpia en swap
  - Memoria residente de procesos sucia en swap

- Estas acciones sólo se ponen en marcha cuando un proceso pide un **exceso de memoria residente**.

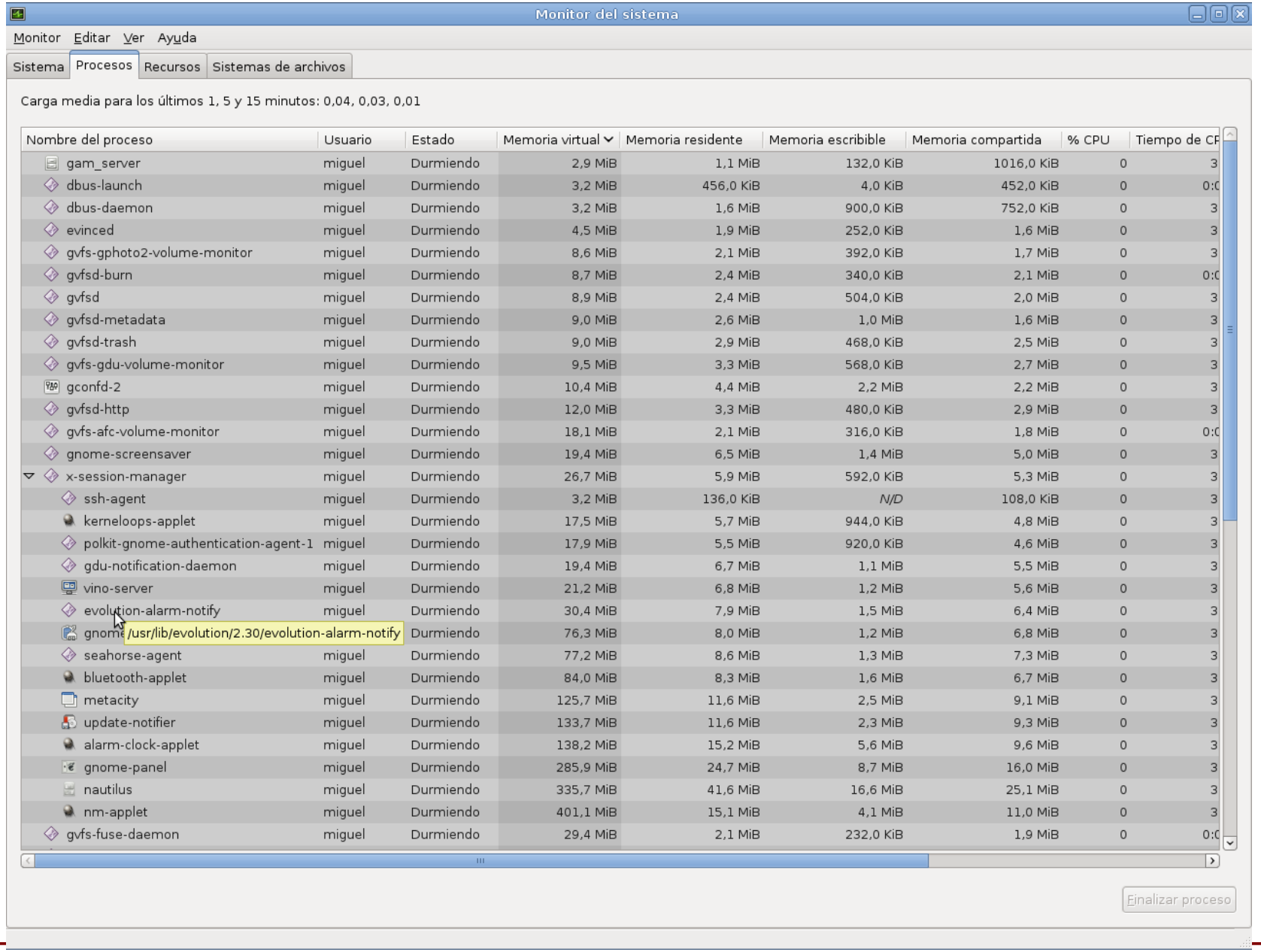
Ejemplo a seguir

- White Paper de los creadores de atop

Herramientas de monitorización

Gnome-System-Monitor

- Herramienta gráfica de monitorización basada en GTK
- Bueno:
  - Intuitíva
  - Se ve en tiempo real la jerarquía de los procesos
  - Se pueden hacer acciones sobre los procesos con un sólo click
  - Muestra información de ficheros abiertos, zonas de memoria
- Negativo:
  - No exporta ningún dato a texto o a fichero
  - No muestra los pthreads
  - No se puede congelar el resultado
  - No se pueden reorganizar las columnas


Monitor del sistema
Monitor Editar Ver Ayuda
Sistema Procesos Recursos Sistemas de archivos
Carga media para los últimos 1, 5 y 15 minutos: 0,04, 0,03, 0,01
Nombre del proceso Usuario Estado Memoria virtual Memoria residente Memoria escribible Memoria compartida % CPU Tiempo de CP
gam_server miguel Durmiendo 2,9 MiB 1,1 MiB 132,0 KiB 1016,0 KiB 0 3
dbus-launch miguel Durmiendo 3,2 MiB 456,0 KiB 4,0 KiB 452,0 KiB 0 0:0
dbus-daemon miguel Durmiendo 3,2 MiB 1,6 MiB 900,0 KiB 752,0 KiB 0 3
evinced miguel Durmiendo 4,5 MiB 1,9 MiB 252,0 KiB 1,6 MiB 0 3
gvfs-gphoto2-volume-monitor miguel Durmiendo 8,6 MiB 2,1 MiB 392,0 KiB 1,7 MiB 0 3
gvfsd-burn miguel Durmiendo 8,7 MiB 2,4 MiB 340,0 KiB 2,1 MiB 0 0:0
gvfsd miguel Durmiendo 8,9 MiB 2,4 MiB 504,0 KiB 2,0 MiB 0 3
gvfsd-metadata miguel Durmiendo 9,0 MiB 2,6 MiB 1,0 MiB 1,6 MiB 0 3
gvfsd-trash miguel Durmiendo 9,0 MiB 2,9 MiB 468,0 KiB 2,5 MiB 0 3
gvfs-gdu-volume-monitor miguel Durmiendo 9,5 MiB 3,3 MiB 568,0 KiB 2,7 MiB 0 3
gconfd-2 miguel Durmiendo 10,4 MiB 4,4 MiB 2,2 MiB 2,2 MiB 0 3
gvfsd-http miguel Durmiendo 12,0 MiB 3,3 MiB 480,0 KiB 2,9 MiB 0 3
gvfs-afc-volume-monitor miguel Durmiendo 18,1 MiB 2,1 MiB 316,0 KiB 1,8 MiB 0 0:0
gnome-screensaver miguel Durmiendo 19,4 MiB 6,5 MiB 1,4 MiB 5,0 MiB 0 3
x-session-manager miguel Durmiendo 26,7 MiB 5,9 MiB 592,0 KiB 5,3 MiB 0 3
ssh-agent miguel Durmiendo 3,2 MiB 136,0 KiB N/D 108,0 KiB 0 3
kerneloops-applet miguel Durmiendo 17,5 MiB 5,7 MiB 944,0 KiB 4,8 MiB 0 3
polkit-gnome-authentication-agent-1 miguel Durmiendo 17,9 MiB 5,5 MiB 920,0 KiB 4,6 MiB 0 3
gdu-notification-daemon miguel Durmiendo 19,4 MiB 6,7 MiB 1,1 MiB 5,5 MiB 0 3
vino-server miguel Durmiendo 21,2 MiB 6,8 MiB 1,2 MiB 5,6 MiB 0 3
evolution-alarm-notify miguel Durmiendo 30,4 MiB 7,9 MiB 1,5 MiB 6,4 MiB 0 3
/usr/lib/evolution/2.30/evolution-alarm-notify
seahorse-agent miguel Durmiendo 77,2 MiB 8,6 MiB 1,3 MiB 7,3 MiB 0 3
bluetooth-applet miguel Durmiendo 84,0 MiB 8,3 MiB 1,6 MiB 6,7 MiB 0 3
metacity miguel Durmiendo 125,7 MiB 11,6 MiB 2,5 MiB 9,1 MiB 0 3
update-notifier miguel Durmiendo 133,7 MiB 11,6 MiB 2,3 MiB 9,3 MiB 0 3
alarm-clock-applet miguel Durmiendo 138,2 MiB 15,2 MiB 5,6 MiB 9,6 MiB 0 3
gnome-panel miguel Durmiendo 285,9 MiB 24,7 MiB 8,7 MiB 16,0 MiB 0 3
nautilus miguel Durmiendo 335,7 MiB 41,6 MiB 16,6 MiB 25,1 MiB 0 3
nm-applet miguel Durmiendo 401,1 MiB 15,1 MiB 4,1 MiB 11,0 MiB 0 3
gvfs-fuse-daemon miguel Durmiendo 29,4 MiB 2,1 MiB 232,0 KiB 1,9 MiB 0 0:0
Finalizar proceso

Trucos de gnome-system-monitor

- Parando el ratón sobre una fila muestra la línea de comandos
- Se pueden ordenar los procesos por consumo:
  - Consumo de CPU: %CPU
  - Consumo de memoria (privada): Memoria
- Se puede observar la jerarquía de los procesos
- Se pueden ordenar los procesos por usuario

top

- Muestra información sobre memoria y uso de CPU

```
Tasks: 172 total,   3 running, 169 sleeping,   0 stopped,   0 zombie
Cpu(s):  5.3%us,  3.4%sy,  0.0%ni, 91.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2065240k total,  1724464k used,   340776k free,   265180k buffers
Swap:  3903752k total,    39888k used,  3863864k free,   788740k cached
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7570 miguel    20   0  644m 150m  32m R    8  7.5 355609:00 firefox-bin
 9292 miguel    20   0  135m  14m  10m S    3  0.7 355573:07 gnome-terminal
14531 miguel    20   0  424m 176m  80m S    1  8.7 355584:15 soffice.bin
16574 miguel    20   0 97476  21m  17m S    1  1.0   0:39.24 gnome-system-mo
16627 root      20   0  2464 1208  896 R    1  0.1   0:00.08 top
 1519 avahi     20   0  3744 2208 1204 S    0  0.1 355573:23 avahi-daemon
 2884 miguel    20   0  138m  15m 9840 S    0  0.8 355573:31 alarm-clock-app
 2890 miguel    20   0 18556 2124 1808 S    0  0.1   0:09.82 gvfs-afc-volume
12524 miguel    20   0  138m  21m  13m S    0  1.0 355573:18 emacs23
15983 miguel    20   0  138m  20m  13m S    0  1.0   0:09.05 emacs23
    1 root      20   0  2032  620  592 S    0  0.0 355572:54 init
    2 root      20   0     0    0    0 S    0  0.0 355572:53 kthreadd
    3 root      RT   0     0    0    0 S    0  0.0 355572:53 migration/0
    4 root      20   0     0    0    0 S    0  0.0   0:00.46 ksoftirqd/0
    5 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT   0     0    0    0 S    0  0.0 355572:53 migration/1
```

Top: Opciones de arranque y uso interactivo

- Opciones interesantes de arranque
  - **-d ss.cc** Periodo de refresco en centésimas de segundo
  - -p <pid1>,<pid2> Restringir a los pids en cuestion
  - -H Mostrar los pthreads que dependen de cada pid
  - -b Modo batch
- Teclas en tiempo de ejecución interactivo
  - h Ayuda
  - f Permite añadir o quitar campos
  - O Cambiar el criterio de ordenamiento
  - o Elegir el orden de disposición de los campos mostrados
  - W Guardar la configuración para futuras llamadas o modo batch

Top: Valores mostrados por thread

PID	Identificador de proceso (o de thread)
PRI	prioridad de un proceso
P	última CPU usada
NI	prioridad "nice" de un proceso
WCHAN	a qué función espera un proceso
STAT	estado del proceso **S**leep, **R**unable, **Z**ombie, **D**:sleep no interrumpible
%CPU	utilización de la CPU
TIME	tiempo de CPU total
TIME+	tiempo de CPU total, en centésimas de segundo

%MEM	utilización de la memoria física
VIRT	memoria virtual total (kb)
RES	memoria física total (kb)
SHR	memoria compartida (kb)
SWAP	espacio de SWAP usado (kb)
CODE	espacio que usa el código del proceso (kb)
DATA	espacio que usan los datos (incluido el stack) del proceso (kb)
nFLT	número de páginas que deben ser leídas de disco (page faults)
nDRT	número de páginas que deben ser escritas en disco (dirty pages)

Personalización de top

- Conviene personalizar la apariencia de top y guardar las opciones pulsando -W
  - La configuración se guarda en **~/.toprc**
  - Se mantiene en modo batch. Es la única forma de asegurar que ciertos campos aparezcan en dicho modo.
- Mis preferencias personales:
  - Activar cmdlines completas: **c**
  - Seleccionar los campos:
    - P (processor): **f j**
    - RES (memoria residente): **f Q**
    - eliminar la prioridad (**f H**) quedándome con el nice.
  - Colocar el campo processor antes del nice: **o J J**
  - Activar sumario de CPU's separadas: **1** (el número 1)

Top -b Modo batch

- Usando la opción -b genera un output parseable y periódico.
  - Muestra siempre los mismos campos (usando el fichero de opciones)
  - Ignora cualquier input de teclado o stdin.
- Invocación clásica

```
nice <prioridad-nice> top -b -H -d <periodo ss.cc> -p <lista_de_pids>
```

- Parámetros:
  - Swtiches: **-H** muestra pthreads
  - **Periodo**: Elegir en función de la carga a medir y de la métrica.
    - %CPU y tiempos: Periodos largos ya que los tiempo de ejecución son acumulativos.
    - Memoria: Periodos cortos ya que las muestras son instantáneas
  - **prioridad nice**: Cuanto más prioridad (valor más pequeño) más se forzará la periodicidad de muestreo frente a sobrecarga de CPU

Salida de top modo batch

```
top - 10:46:12 up 17:15,  6 users,  load average: 0.92, 0.53, 0.67
Tasks:  13 total,   1 running,  12 sleeping,   0 stopped,   0 zombie
Cpu0  :  3.7%us,  1.3%sy,  0.0%ni, 94.2%id,  0.8%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  3.1%us,  1.1%sy,  0.0%ni, 95.7%id,  0.1%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2064860k total,  1877996k used,   186864k free,   312388k buffers
Swap:  3903756k total,        0k used,  3903756k free,   924988k cached

  PID USER      P  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6375 miguel    0   0  725m  26m  13m R  100  1.3  34:28.08 java
 6307 miguel    0   0  725m  26m  13m S    0  1.3   0:00.00 java
 6312 miguel    1   0  725m  26m  13m S    0  1.3   0:04.43 java
 6313 miguel    1   0  725m  26m  13m S    0  1.3   0:00.00 java
 6314 miguel    0   0  725m  26m  13m S    0  1.3   0:00.00 java
 6315 miguel    1   0  725m  26m  13m S    0  1.3   0:00.11 java
 6316 miguel    0   0  725m  26m  13m S    0  1.3   0:00.00 java
```

- Consejos para parsear
  - Cada bloque de muestreo comienza en una linea con la palabra “top”
  - Seguido del top está el timestamp del momento de muestreo
  - Después de la siguiente línea vacía se encuentran las cabeceras de os campos
  - Cada valor de campo de cada thread está separado por espacios salvo el COMMAND al final.

htop

- Versión “mejorada” de top basada en ncurses
- Ventajas sobre top
  - Es más interactivo y resumido
  - Permite seleccionar thread's para cambiarles la prioridad, afinidad...
  - Permite mostrar la jeráquica de procesos y thread
  - Se puede llamar directamente a strace y ltrace para tracear llamadas del sistema y de librerías
- Desventaja: No se puede exportar directamente el texto
  - Si se redirige la salida a fichero aparecen códigos de escape ANSI
    - Existen formas de filtrarlo (ej: paquete PERL)
  - Se puede copiar-pegar seleccionando con el ratón **mientras se pulsa SHIFT**

Personalización de htop

- Pulsando **S** (mayúscula) se accede a la pantalla de personalización.
  - La personalización se mantiene entre ejecuciones en **./~htopr**
- Mi personalización personal
  - Métricas: Añadir el hostname a la parte derecha del sumario
  - Display options:
    - Tree view ON
    - Hide kernel threads OFF
    - Display threads in different color ON
    - Show custom thread names ON
    - Highlight program basename ON
    - Detailed CPU time ON
  - Columns: Añadir PROCESSOR, TPGID

ps: Queries sobre procesos

```
ps [opciones] [PID]
```

- Comando para hacer queries sobre procesos
  - **-A** Todos los procesos
  - **-F** Incluír todos los detalles
  - -L Incluir los pthreads

- Todos threads del sistema

```
ps -ALF
```

- Grep is your friend!!

```
ps -ALF | grep `whoami`
```

¿Cual usar?

	Ventajas	Inconvenientes	Aplicación
top	Modo batch Salida parseable	No muestra todos los threads No tiene modo jeráquico No muestra la afinidad	Muestreo periódico de pids conocidos
htop	Muestra todos los threads Buen modo interactivo Fácil de usar Interfaz con strace, lsof, ltrace	No es fácil de parsear	Vigilancia interactiva Toma de comandos (prioridad, afinidad)
ps	Muestra todos los threads Muestra toda la información	Opciones de comando complicadas No tiene muestreo periódico	Queries rápidas de estado
gnome sys monit	Gráfico en tiempo real Re-ordenamiento con un click Muestra los mapas de memoria	No muestra los pthreads No exporta los datos Poco flexible en visualización	Gráficos de consumo en tiempo real

/usr/bin/time

- Esta herramienta mide tiempos de ejecución asociados a una aplicación distinguiendo tiempo de sistema y de usuario
  - /usr/bin/time -v <comando> <argumentos>
    - -v : Muestra el significado de cada campo
- **Nota**: No confundir con el comando **time** de la shell **bash**,
  - Ejecutar siempre especificando el path completo

- Información extra obtenida
  - Fallos de página de acceso a memoria
  - Número de “swaps”
  - Cambios de contexto

Herramientas de información global

- vmstat
- sar
- iostat
- mpstat

- Todas ellas informan de parámetros globales a la plataforma
  - Más orientadas a la configuración del sistema

Referencias

Libros sobre performance en Linux

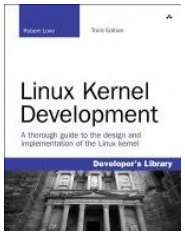
- IBM Redbook: Eduardo CLIENDO, Takechica KUNIMASA, *Linux Performance and Tuning Guidelines,* 2007 Descargable libremente
  - Explica el funcionamiento del kernel en aspectos como memoria e I/O
  - Detalla herramientas de línea de comando y benchmarks estándares
  - Da consejos de configuración del kernel

- Mark WILDING Dan BEHMAN, *Self service Linux: Mastering the Art of Problem Determination* Prentice Hall 2006. Bruce Perens Open Source series. ISBN 978-0-13-147751-3 Descargable libremente

  - Más orientado a desarrollo y a traceado de bajo nivel
  - Detalla el formato ELF y conceptos de debugueo
  - Al final lista diferentes herramientas de monitorización y da un script para captura de datos.

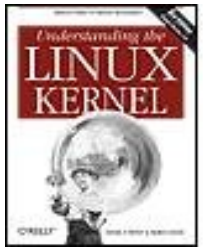
Libros Genéricos sobre el Kernel

- Robert LOVE, *Linux Kernel Development*, 3rd Edition Addison-Wesley 2010, ISBN 978-0-672-32946-8



  - Abarcable y sencillo de leer.
  - Punto de vista: Desarrollador que quiere **añadir código propio** al kernel y utilizar algunos servicios sin interferir en el funcionamiento normal.
  - Aunque es una edición reciente, hay muchas novedades del kernel (ftrace, hrtimers, Xen) que no se han añadido.
  - Un buen review del libro en LWN.net https://lwn.net/Articles/419855/

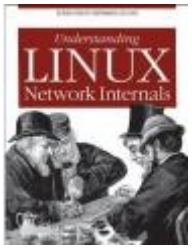
- Daniel POVET, Marco CESATI, *Understanding the Linux Kernel*, 3rd Edition O'Reilly 2005 ISBN 978-0-596-00565-2. Disponible en google books. UTLK3



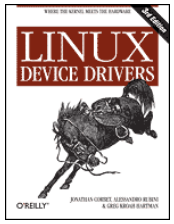
  - Muy completo pero denso de leer y digerir
  - Punto de vista: Investigador o estudiante que quiere **entender como funciona el kernel por dentro** para diagnosticar problemas o copiar las técnicas en otros ámbitos.
  - Aunque tiene ya 6 años, la filosofía del kernel sigue siendo la misma, si bien los procedimientos de funcionamiento que detalla el libro pueden haber evolucionado

Libros puntuales sobre el kernel

- Mel GORMAN, *Understanding Linux Virtual Memory Manager* Prentice Hall 2004. Bruce Perens Open Source series. ISBN 978-0-13-145348-7 Descargable libremente
  - Detalla como el kernel gestiona la memoria. Como siempre las implementaciones evolucionan pero el espíritu se mantiene.



- Christian BENVENUTI, *Understanding Linux Network Internals*, O'Reilly 2006, ISBN 978-0-596-00255-8. Disponible en Google books
  - Único libro escrito sobre el forrogoso tema del tratamiento de redes y protocolos en Linux. Se concentra en las capas de red y bridges (IP, ruteo, ARP...) pero omite aspectos importantes como TCP o Traffic Control.
  - Critica en LWN.net: http://lwn.net/Articles/168894/



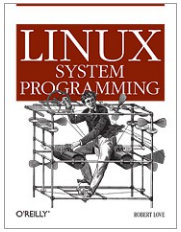
- Jonathan CORBET, Alessandro RUBINI, Greg Kroah-Hartman, *Linux Device Drivers* 3rd Edition, O'Reilly 2005, ISBN-978-0-596-00590-3 Descargable libremente
  - Libro pensado directamente para escribir código en el kernel, sin especificar el funcionamiento de algunos mecanismos. En este aspecto se muy bien con UTLK 3rd.

Libros sobre programación de sistema

- Michael KERRISK, *The Linux Programming Interface* No Starch Press 2010 ISBN 978-1-59327-220-3. Sitio web de soporte

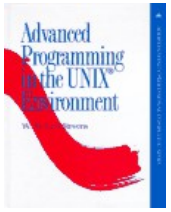
  - Cubre un **amplio** espectro de programación en sistema de una manera **precisa** y fácil de leer.
  - Cubre el caso de Linux y diferencia otros Unixes y BSD.
  - Critica en LWN.net: http://lwn.net/Articles/423417/

- Robert LOVE, *Linux System Programming*, O'Reilly 2007, ISBN 978-0-59600-958-8. Sitio web de soporte



  - Más sencillo y ligero que el LTPI pero menos riguroso y completo
  - Critica en LWN.net: http://lwn.net/Articles/168894/

- W. Richard STEVENS, *Advanced programming in the Unix Environment* Addison-Wesley 1992, ISBN 0-201-56317-7. Sitio web de soporte



  - Clásico, fácil de leer y se toma aún como referencia
  - Anterior al crecimiento de Linux, pero aún válido para la mayoría de los usos