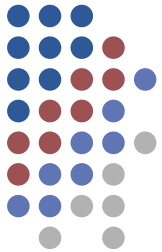
S.I.
Unidad 11

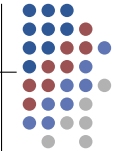
Administración de GNU/Linux

Gestión de procesos



Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Gestión de procesos

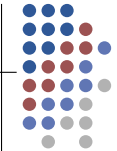


- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Gestión de procesos

- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Introducción



- Un usuario puede ejecutar varios procesos a la vez
  - *Recuerda: GNU/Linux = multitarea*

- En cada terminal, sólo uno de los procesos en ejecución puede interactuar directamente con el usuario
  - Se trata del proceso en **primer plano**
  - ***Foreground***
  - No se puede hacer otra tarea hasta que el proceso termine
  - *Ejemplo: al lanzar el siguiente comando, va a tardar un buen rato, por lo que no podremos hacer otra cosa que esperar a que termine*

```
$ls -R /
```

Introducción

- El resto de procesos se ejecutan en **segundo plano**
  - ***Background***
  - La salida de un proceso en segundo plano **sí** que se muestra por pantalla si no se redirige

- ¿Cómo se puede ejecutar un proceso en segundo plano?
  - Añadimos al final de la instrucción un **&**

Introducción

- *Ejemplo:*
  - Queremos ejecutar desde terminal el programa gráfico ***Firefox***

```
$firefox
```

  - Vemos que se inicia, pero no podemos utilizar el terminal hasta que finalice la ejecución de *firefox*

Introducción

- *Ejemplo:*
  - Para evitar todo esto, debemos ejecutar:

```
$firefox &
```

  - Al escribir esto, vemos que GNU/Linux devuelve el control a la consola al instante, mostrando una línea similar a esta:

```
[1] 17860
```

  - Esto significa que el proceso *firefox* se ha ejecutado con el **PID 17860**
    - El **PID** es el identificador del proceso (***Process ID***)
    - Cada proceso lanzado en el sistema tiene un número identificador

Introducción

- **Estados de un proceso**
  - Ejecutándose (*running*)
    - El proceso se está ejecutando, usando CPU
    - Forma parte de la cola de ejecución del kernel
  - En espera (*sleeping*)
    - El proceso se está ejecutando, pero está esperando a que se produzca un evento
    - *Ejemplo*:

```
$comando1 ; comando2
```

      - `comando1` estará en ejecución
      - `comando2` estará en espera hasta que no finalice `comando1`

Introducción

- **Estados de un proceso**
  - Parado (*Stopped)*
    - Cuando paramos un proceso, este se queda en este estado esperando a ser "reactivado"
    - Es como si estuviera "en pausa"
  - Zombie
    - El proceso ha finalizado irregularmente y está en memoria
    - El superusuario debería comprobar que no quedan procesos de este tipo en memoria, no es deseable

Gestión de procesos

- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Monitorización de procesos

- **ps**
  - Información de los procesos en ejecución **del usuario** que ejecuta el comando
  - Sólo aparecen los comandos del terminal actual
- **ps u**
  - Lo mismo, pero con más información

R = Ejecutándose
S = En espera
T = Parado
Z = Zombie

- **ps aux**
  - Información de todos los procesos **de todos los usuarios** del sistema

¿Cómo filtrar datos?

Monitorización de procesos

- **pgrep**
  - Muestra el **PID** del proceso cuyo nombre es pasado por parámetro

```
$pgrep firefox
```

- Otra opción para poder filtrar un proceso de todo el listado es utilizar el comando `ps` anidado con `grep`
  - *Ejemplo: queremos filtrar el proceso firefox de todo el listado de procesos*

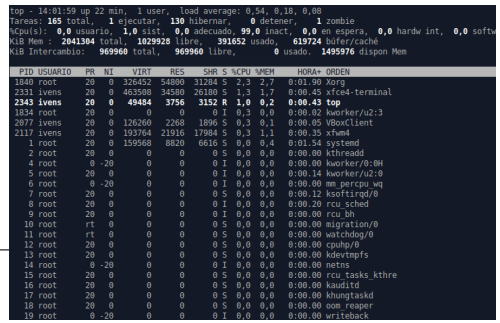
```
$ps aux | grep firefox
```

Monitorización de procesos

- **pstree**
  - Muestra un árbol de procesos
  - Nos permite ver quién es el proceso padre de cada proceso

Monitorización de procesos

- **top**
  - Nos permite ver:
    - Información del sistema (memoria libre, carga del sistema, uptime,...)
    - La lista de procesos y sus estados
  - Ordenación
    - Por ocupación de CPU → Mayúsculas + P
    - Por ocupación de memoria → Mayúsculas + M
  - Salir
    - q

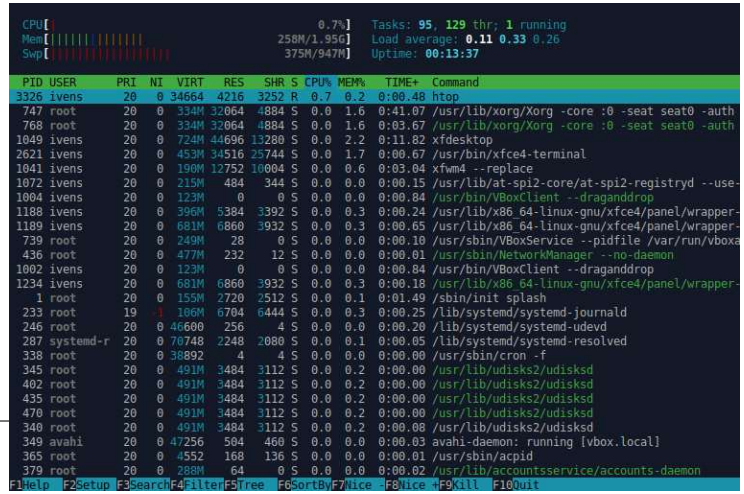


Monitorización de procesos

- **htop**
  - Se trata de una versión modernizada de top, aunque no suele venir instalada por defecto en las distribuciones

```
#apt install htop
```

Nos va a facilitar la gestión de procesos que veremos más adelante en este bloque, tales como mandar señales a procesos (tecla F9)



Gestión de procesos

- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Señales a procesos

- Una señal es una manera que tiene un proceso de comunicarse con otro
- Las más importantes son:
  - **SIGSTOP (nº19)**
    - Detiene un proceso
  - **SIGCONT (nº18)**
    - Continua un proceso previamente parado
  - **SIGTERM (nº15)**
    - Termina un proceso de forma ordenada
  - **SIGKILL (nº9)**
    - Mata un proceso

Señales a procesos

- Sólo podremos enviar señales a nuestros procesos
  - **root** podrá enviar señales a cualquier proceso
- Para enviar una de estas señales a un proceso, utilizaremos la orden **kill**
- Podemos utilizarla de dos formas:

```
$kill nombre_señal proceso
```

  - *Ejemplo:*

```
$kill SIGTERM 1560
```

```
$kill -número_señal proceso
```

  - *Ejemplo:*

```
$kill -15 1560
```


Señales a procesos

Ejemplo:

- Ejecutamos el navegador **firefox** en segundo plano:

```
$firefox &
```

- Comprobamos cuál es su PID

```
$pgrep firefox
```

- Enviamos una señal de parada a su proceso

A elegir

```
$kill SIGTERM 16507
$kill -15 16507
```

...y vemos que el proceso *firefox* tiene, por ejemplo, el PID = 16507

Repite el ejemplo, pero ahora enviando una señal para matar el proceso

Gestión de procesos

- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Prioridades de procesos

- Los valores de prioridad que GNU/Linux asigna a un proceso van:
  - Desde el **-20** (máxima prioridad)…
  - …Hasta el **19** (mínima prioridad)

Prioridades de procesos

- Mediante el comando **nice** podremos establecer la prioridad con la que se ejecuta un proceso

```
$nice [-n valor_nice] [comando]
```

```
$nice -n 7 find / -name "*.mp3"
```

```
$nice -n -15 tar -cvzf backup.tgz ~/datos/
```

nice ≈ "gentil"

"Un proceso muy gentil deja paso al resto de procesos para que utilicen la CPU"

Prioridades de procesos

- Si hemos ejecutado un comando y queremos cambiar su prioridad, podemos hacerlo usando el comando **renice**

```
$renice [nuevo_valor_nice] [PID]
```

```
$renice 19 6535
```

```
$renice -20 1407
```

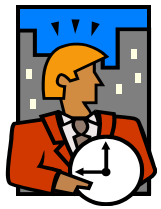
- *La aplicación htop también permite modificar los valores nice de forma "gráfica" utilizando las teclas F7 y F8*

Gestión de procesos

- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Planificación de procesos

- Existe un proceso que se lanza automáticamente al iniciar el sistema llamado **cron**
- El usuario puede programar la ejecución de tareas y **cron** se encargará de lanzarlas cuando corresponda



- Las tareas se programan en el fichero `/etc/crontab`

Planificación de procesos

- Contenido del fichero `/etc/crontab`
  - Una tarea programada por línea
  - Campos separados por espacios / tabulaciones
    - Minuto [0 - 59]
    - Hora [0 - 23]
    - Día del mes [1 - 31]
    - Mes [1 - 12]
    - Día de la semana [0 - 6] (Domingo = 0)
    - Usuario que lo lanza
    - Comando

Un asterisco en un campo indicaría cualquier valor

Planificación de procesos

- *Ejemplos:*

```
#min  hor  ddm  mes  dds  user  cmd
15    8    *    *    *    paco  /usr/bin/backup
45    19   15   *    *    juan  /home/juan/uno
25    21   7    3    *    root  /usr/bin/otro.sh
```

El usuario paco lanzará todos los días, a las 8:15h ,el programa backup que está en /usr/bin

El usuario juan lanzará todos los días 15 del mes, a las 19:45h el comando uno, localizado en su directorio personal

root lanzará todos los 7 de marzo, a las 21:25h, el comando otro.sh, localizado en /usr/bin

Planificación de procesos

- *Ejemplos:*

```
#min  hor  ddm  mes  dds  user  cmd
00    11   *    *    6    paco  /home/paco/mas
15    8,9  *    *    *    rosa  /home/rosa/bck
45    19   1-10 *    *    pepe  /usr/bin/otro.sh
```

El usuario paco lanzará todos los sábados (día 6), a las 11:00h, el programa mas que está en su directorio personal

El usuario rosa lanzará todos los días, a las 8:45h y 9:45h, el comando bck, localizado en su directorio personal

El usuario pepe lanzará los primeros 10 días de cada mes, a las 19:45h, el comando otro.sh, localizado en /usr/bin

Planificación de procesos

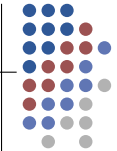
- *Ejemplos:*

```
#min   hor    ddm    mes     dds    user   cmd
25     21     7      3,6,9   *      root   /usr/bin/otro2.sh
00     11     *      *       1-5    luis   /usr/bin/listar
```

root lanzará todos los 7 de marzo, 7 de junio y 7 de septiembre, a las 21:25h, el programa otro2.sh que está en /usr/bin

El usuario luis lanzará de lunes a viernes, a las 11:00h, el comando listar, localizado en /usr/bin

Gestión de procesos



- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Trabajos

- En GNU/Linux, cuando arranca un proceso, se asigna (además del PID) un número de trabajo
- Recordemos cuando hemos ejecutado un comando en segundo plano:

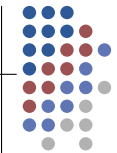
```
$firefox &
```

- Vimos que el sistema mostraba una línea similar a esta:

```
[1] 17860
```

  - *Significa que GNU/Linux ha creado el trabajo **1** para el proceso 17860*

Trabajos



- Un trabajo puede contener varios procesos
  - Si no fuera así, no tendría sentido que existieran los números de trabajo (con el PID valdría)
- ¿Cómo crear un trabajo de varios procesos?

```
$( comando1; comando2; comando3 ) &
[1] 1517
```

Se ha creado el trabajo **nº1**

El PID del **último** proceso es el 1517

  - Se ejecutará en 2º plano el proceso *comando1*
  - Cuando éste termine, se ejecutará en 2º plano *comando2*
  - Cuando éste termine, se ejecutará en 2º plano *comando3*
  - Cuando el último termine, el trabajo habrá concluido

Trabajos

- Al terminar un trabajo, el número de trabajo queda libre para ser utilizado
  - Esto no ocurre con los PID: el contador sigue incrementándose
- Para ver los trabajos activos en el sistema, utilizaremos el comando ***jobs***

```
$jobs
```


Trabajos

Ejemplo:

```
$jobs
[1]  Ejecutando   comando1; comando2 &
[2]- Hecho        comando3; comando4
[3]+ Ejecutando   comando5; comando6 &
```

- El signo "+" → Último trabajo que se ha ejecutado
- El signo "-" → El anterior trabajo
- Sin signo → Otros trabajos
- En este ejemplo, el trabajo 2 ya ha terminado
  - Se muestra para informar al usuario
  - La próxima vez que se ejecute *jobs* ya no aparecerá en la lista

Trabajos

- En Bash, los trabajos se especifican con el carácter %

Ejemplo:

- Matar el trabajo número 3

```
$kill -9 %3
```


Trabajos

- Podemos pasar a primer plano un trabajo o proceso que estuviera en segundo plano
  - Comando ***fg*** (foreground)

```
$fg %2
```

Esto pasaría a un primer plano el trabajo número 2

```
$fg 1563
```

Esto pasaría a un primer plano el proceso con PID = 1563

Trabajos

- Podemos pasar a segundo plano un proceso o trabajo que estuviera en primer plano
  - Comando ***bg*** (background)

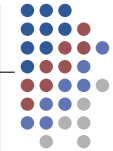
```
$bg %2
```

Esto pasaría a un segundo plano el trabajo número 2

```
$bg 1765
```

Esto pasaría a un segundo plano el proceso con PID = 1765

Trabajos

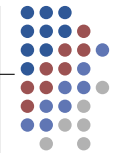


- Pero para esto último… ¿cómo es posible si, mientras se está ejecutando en primer plano, el usuario no puede continuar trabajando (no puede teclear nada)?
- Solución:
  - Disponemos de varias consolas
  - Podemos parar el proceso previamente
    - **`Ctrl+Z`**
    - **`$kill SIGSTOP proceso/trabajo`**

Gestión de procesos

- Índice
  - Introducción
  - Monitorización de procesos
  - Señales a procesos
  - Prioridades de procesos
  - Planificación de procesos
  - Trabajos
  - Directorio /proc

Directorio /proc



- GNU/Linux contiene un pseudo-sistema de ficheros montado en el directorio `/proc`
- Es un directorio virtual
  - Los ficheros que contiene no están almacenados en ningún disco duro
  - Son una forma curiosa que tiene GNU/Linux de obtener información del kernel en tiempo real
    - `ps`, `top`,… cogen la información de este directorio

Directorio /proc

- Para leer la información de estos ficheros virtuales, usaremos un simple ***cat***

- Algunos de ellos son:
  - **cpuinfo** Información de la/s CPU del sistema
  - **meminfo** Información de la memoria del sistema
  - **partitions** Particiones montadas en el sistema
  - **swaps** Particiones destinadas a swapping
  - **version** Información del kernel
  - …

```
$cat /proc/cpuinfo
```


Directorio /proc

- Para cada proceso que está en ejecución, encontraremos un subdirectorio dentro de */proc* con la siguiente aparciencia:

/proc/*numero_proceso*

- Dentro de él encontraremos información relativa a dicho proceso