# Gestión de procesos y servicios

Gestión de usuarios y procesos

# Comando top

- Si deseamos obtener una actualización continua de los procesos actuales usaremos este comando.
- Permite visualizar el estado del sistema, con información que se actualiza en tiempo real.

```
top - 12:26:15 up  2:49,  1 user,  load average: 0,62, 0,61, 0,61
Tasks: 260 total,   1 running, 259 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6,2 us,  1,8 sy,  0,0 ni, 91,9 id,  0,1 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :  15736,5 total,   9806,7 free,   2753,2 used,   3176,5 buff/cache
MiB Swap:  35064,0 total,  35064,0 free,      0,0 used.  12154,9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  10253 mikel     20   0 9047292 379500 118564 S  13,5   2,4  11:54.70 chrome
   2047 mikel     20   0 3980820 276736 104696 S   7,9   1,7   3:05.62 gnome-s+
   3516 mikel     20   0  657344 147496  85088 S   5,3   0,9   5:44.68 chrome
   1862 mikel     20   0  848760  67144  41804 S   2,3   0,4   1:57.55 Xorg
   4922 mikel     20   0 3207948 138000 102764 S   2,0   0,9   2:12.79 teams
   3481 mikel     20   0  794728 308916 176832 S   1,7   1,9   6:58.19 chrome
  10433 mikel     20   0 4827868 295980 116836 S   1,3   1,8   1:59.08 chrome
   3697 mikel     20   0   16,7g 405104 113840 S   1,0   2,5   9:21.73 chrome
  11068 mikel     20   0   21496   4264   3452 R   1,0   0,0   0:09.22 top
   1195 root      20   0  862808  82400  46264 S   0,7   0,5   0:42.22 dockerd
   2619 mikel     20   0  320872   8036   6912 S   0,7   0,0   0:01.16 gsd-hou+
   3520 mikel     20   0  379748 102352  68328 S   0,7   0,6   1:37.64 chrome
```

# Salida del comando top

- 1ª línea: hora actual, tiempo del sistema encendido, número de usuarios y carga media en intervalos de 1, 5 y 15 minutos.
- 2ª línea: número total de tareas, en ejecución, durmiendo (waiting), detenidas (terminated) y zombies.
- 3ª línea: porcentaje de tiempo de CPU usado de procesos de usuario (us), procesos del kernel (sy), ocioso (id), en espera a operaciones de entrada/salida en disco (wa)…
- 4ª línea: tamaño en MB de memoria física total, libre, usada y del buffer.
- 5ª línea: tamaño en MB de la memoria virtual total, libre, usada y disponible.

# Campos de cada proceso (top)

- **PID**: id del proceso
- **USER**: usuario propietario del proceso
- **PR**: prioridad del proceso
- **NI**: Nice value, usado para determinar la prioridad del proceso
- **VIRT**: El tamaño total de memoria virtual usado por la tarea (en Kb)
- **RES**: El tamaño de la memoria física usada por la tare (en Kb)
- **SHR**: El tamaño de la memoria compartida usada por el proceso
- **S**: Estado del proceso (STAT)
- **TIME**: tiempo de CPU utilizado por el proceso.

# Ejecución en primer y segundo plano

- Cuando un usuario ejecuta un comando o script desde el shell, este crea un subshell en el cual se ejecuta realmente el comando o script.
- El usuario ha de esperar a que la ejecución del comando termine para recuperar el control del shell y poder ejecutar otros comandos. En el momento que termina vuelve a aparecer el prompt.
- Esta manera de ejecutar comandos se conoce como ejecución en **primer plano** o en foreground.
- Es posible que, en ocasiones, no se quiera esperar a que la ejecución termine para recuperar el control del shell.

# Ejecución en primer y segundo plano

- Por ejemplo, es frecuente que cuando se inicia una tarea de procesamiento que tarde horas en terminar se prefiera que su ejecución no bloquee el shell durante todas esas horas.
- Cuando la ejecución de una tarea no bloquea el shell se dice que se está ejecutando en **segundo plano** o en background.
- Para hacer que una tarea se ejecute en segundo plano basta con añadir al final de la orden el símbolo "&"
- Cuando ejecutamos una tarea en segundo plano el prompt se devuelve inmediatamente, pudiendo ejecutar otras tareas.

# Ejercicio

- Ejecuta el comando "yes" en primer plano.
- Ejecuta el comando "yes" en segundo plano.
- Para hacer que una tarea se ejecute en segundo plano basta con añadir al final de la orden el símbolo "&".
- Abre otra terminal, localiza el proceso en ejecución.
- Para el proceso ejecutando el comando kill –9 <PID>
- Vuelve a lanzar el proceso en segundo plano, pero esta vez trata de redireccionar la salida a /dev/null para que no se llene la pantalla de la palabra "yes"
- Vuelve a buscar el proceso y mátalo, esta vez podrás hacerlo desde la misma terminal

# Prioridad de las órdenes

- El algoritmo de planificación de Linux determina el orden de ejecución de las distintas tareas en la cola de "listos".
- Este algoritmo emplea una mezcla de algoritmos como Round Robin, FIFO, cola de prioridades, etc...
- La prioridad final (PRI) de cada proceso se calcula mediante un algoritmo que tiene en cuenta varios factores.
- Entre estos factores, existe un parámetro llamado prioridad relativa (nice) que oscila entre los valores –20 (máxima prioridad) y 19 (mínima prioridad).
- Por defecto el valor de nice es 0.
- Para ver las prioridades de los procesos podemos ejecutar ps -l

# Prioridad de las órdenes

- Es posible alterar la prioridad con la que se va a ejecutar una orden añadiendo el comando nice antes de la orden.

**nice –n  <prioridad> <orden>**

- También es posible alterar la prioridad de un proceso que ya está en ejecución gracias al comando renice

**renice <prioridad> -p <PID>**

# Envío de señales

- Los procesos reciben señales para ser controlados desde el propio sistema operativo y desde el exterior.
- Es posible enviar señales a los procesos desde una terminal mediante el comando **kill**.

**kill -<señal> <PID>**

- Existen un conjunto de señales predefinidas que permiten llevar a cabo diferentes acciones.
- Algunos de los propósitos más típicos del envío de señales a procesos son: detener procesos que consuman muchos recursos, interrumpir un proceso que esté bloqueando otras tareas, etc...

# Señales más usadas

- **2 o SIGINT**: interrumpe un proceso, es la señal que ocurre cuando se pulsa CTRL+C desde la shell para parar un proceso. Esta señal puede ser manejada por el propio proceso que se va a detener.
- **9 o SIGKILL**: mata al proceso de forma fulminante.
- **15 o SIGTERM**: mata al proceso de forma "amable", es decir, permite al proceso que termine su ejecución de forma ordenada, liberando recursos y evitando que acabe en un estado inconsistente. Esta señal puede ser ignorada en algunos casos por lo que SIGKILL resulta más efectiva.
- **19 o SIGSTOP**: se pausa la ejecución de un proceso.
- **18 o SIGCONT**: continua la ejecución de un proceso pausado.

# Gestión de servicios

- Existen múltiples procesos que se ejecutan en segundo plano que dan algún tipo de servicio al sistema. Estos procesos se denominan servicios.
- El comando **systemctl** nos permite realizar la gestión de estos servicios gracias a sus diferentes opciones.
- Visualizar servicios instalados: **systemctl list-units -t service –all**
- Parar un servicio: **systemctl stop <servicio>**
- Iniciar un servicio:  **systemctl start <servicio>**
- Reiniciar un servicio: **systemctl restart <servicio>**
- Ver estado de un servicio: **systemctl status <servicio>**