

**PRUEBA ABIERTA: UD4**

# **GESTIÓN DE PROCESOS**

Francisco Javier Martínez Reguera

# ÍNDICE

1. Introducción .....	Pág. 1
2. Gestión de procesos en Linux .....	Pág. 1
2.1. Visualización de procesos con el comando <b>ps</b> .....	Pág. 1
2.2. Terminación de procesos con el comando <b>kill</b> .....	Pág. 6
2.3. Gestión de prioridades con el comando <b>nice</b> .....	Pág. 8
3. Gestión de procesos en Windows .....	Pág. 8
3.1. Gestión de procesos a través de la interfaz gráfica .....	Pág. 8
3.2. Gestión de procesos a través del terminal .....	Pág. 10
4. Conclusiones .....	Pág. 12
Bibliografía .....	Pág. 13

# 1. INTRODUCCIÓN

Un programa ejecutable es un conjunto de instrucciones y datos almacenados en un fichero. Cuando esto se carga en la memoria y se pone en ejecución, se convierte en un proceso. Conocer el funcionamiento y administración de dichos procesos es fundamental para un buen uso de un sistema informático.

Un proceso está formado por la imagen binaria de un programa, cargada total o parcialmente en la memoria física, y un área de memoria para almacenar datos temporales, conocida como pila. La imagen binaria está formada por las instrucciones y datos del programa. La imagen binaria y la pila son el programa en sí mismo, pero para que el SO pueda controlar el programa hacen falta una serie de estructuras de datos, que fundamentalmente son la tabla de páginas para traducir las direcciones virtuales generadas por el proceso en las direcciones físicas en las que se encuentra almacenado y una estructura de control, conocida como PCB, para que el sistema operativo pueda controlar su ejecución.

Un proceso pasa por varios estados durante su ejecución. Los estados posibles para un proceso son:

- **Nuevo:** el proceso se acaba de crear, pero aún no ha sido admitido en el grupo de procesos ejecutables por el sistema operativo. Habitualmente, nada más que un proceso se crea resulta admitido, pasando al estado de listo. Sin embargo, en situaciones con sobrecarga el SO puede decidir retardar la admisión.
- **Listo:** el proceso está esperando a ser asignado al procesador para su ejecución.
- **En ejecución:** el proceso tiene la CPU y ésta ejecuta sus instrucciones.
- **En espera:** el proceso está esperando a que ocurra algún suceso, como por ejemplo la terminación de una operación de entrada o salida.
- **Terminado:** el proceso ha sido sacado del grupo de procesos ejecutables por el sistema operativo. Después de que un proceso es marcado como terminado se liberarán los recursos utilizados por ese proceso (por ejemplo, la memoria).

A continuación veremos cómo gestionar dichos procesos en distintos sistemas operativos.

## 2. GESTIÓN DE PROCESOS EN LINUX

### 2.1. VISUALIZACIÓN DE PROCESOS CON EL COMANDO **PS**

Comenzaremos con el comando **ps**, que simplemente nos muestra los procesos en ejecución. Crearemos primero un proceso de tipo **sleep**, que ejecutaremos en segundo plano añadiendo un ampersand al final de la línea de comando, y a continuación comprobaremos que está en ejecución.

```
javier@ubuntu: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
javier@ubuntu:~$ sleep 30 &  
[1] 2478  
javier@ubuntu:~$ ps  
  PID TTY          TIME CMD  
 2470 pts/0    00:00:00 bash  
 2478 pts/0    00:00:00 sleep  
 2479 pts/0    00:00:00 ps  
javier@ubuntu:~$
```

Observamos que el proceso se ha iniciado correctamente y que se encuentra en ejecución. Las cuatro columnas resultantes de usar el comando **ps** son:

- **PID**: el número de identificación del proceso.
- **TTY**: el nombre de la consola en la que el usuario ha iniciado sesión.
- **TIEMPO**: la cantidad de tiempo de procesamiento de la CPU que el proceso ha utilizado.
- **CMD**: el nombre del comando que inició el proceso

Por defecto, **ps** muestra sólo los procesos que se ejecutaron desde su propia terminal. Las opciones **-e** y **-A** hacen que se muestren todos los procesos del sistema.

```
javier@ubuntu: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
javier@ubuntu:~$ ps -e  
  PID TTY          TIME CMD  
    1 ?           00:00:15 systemd  
    2 ?           00:00:00 kthreadd  
    4 ?           00:00:00 kworker/0:0H  
    6 ?           00:00:00 mm_percpu_wq  
    7 ?           00:00:07 ksoftirqd/0  
    8 ?           00:00:01 rcu_sched  
    9 ?           00:00:00 rcu_bh  
   10 ?           00:00:00 migration/0  
   11 ?           00:00:00 watchdog/0  
   12 ?           00:00:00 cpuhp/0  
   13 ?           00:00:00 cpuhp/1  
   14 ?           00:00:00 watchdog/1  
   15 ?           00:00:00 migration/1  
   16 ?           00:00:01 ksoftirqd/1  
   18 ?           00:00:00 kworker/1:0H  
   19 ?           00:00:00 kdevtmpfs  
   20 ?           00:00:00 netns  
   21 ?           00:00:00 rcu_tasks_kthre  
   22 ?           00:00:00 kauditd  
   24 ?           00:00:00 khungtaskd  
   25 ?           00:00:00 oom_reaper  
   26 ?           00:00:00 writeback
```

Se pueden mostrar los procesos que pertenecen a un usuario concreto con la opción **-u** seguida del nombre de usuario. Si no se especifica usuario, se considerará el que realiza la consulta como se puede ver a continuación.

```
javier@ubuntu: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
javier@ubuntu:~$ ps -u  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
javier    1502  0.0  0.2 212292  6048 tty2    Ssl+  16:43   0:00 /usr/lib/gdm3/  
javier    1504  1.6  3.0 509360 62128 tty2    Rl+   16:43   0:58 /usr/lib/xorg/  
javier    1517  0.0  0.8 717744 16352 tty2    Sl+   16:43   0:02 /usr/lib/gnome  
javier    1640  5.0 18.6 3472664 376020 tty2    Sl+   16:43   3:05 /usr/bin/gnome  
javier    1662  0.1  0.5 377900 10444 tty2    Sl    16:43   0:05 ibus-daemon --  
javier    1666  0.0  0.4 297100  8076 tty2    Sl    16:43   0:00 /usr/lib/ibus/  
javier    1668  0.0  1.1 357096 22960 tty2    Sl    16:43   0:00 /usr/lib/ibus/  
javier    1740  0.0  1.2 528096 24284 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1743  0.0  0.5 349524 10212 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1744  0.0  0.2 423548  5964 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1745  0.0  0.2 275936  5912 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1749  0.0  0.5 471556 11704 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1751  0.0  0.5 393092 10232 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1753  0.0  1.1 505492 24196 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1761  0.0  0.4 343456 10080 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1763  0.0  1.1 441584 23260 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1765  0.0  0.4 297008  8608 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1766  0.0  1.1 669652 23976 tty2    Sl+   16:43   0:01 /usr/lib/gnome  
javier    1767  0.0  1.1 356756 22544 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1768  0.0  0.4 375064  9164 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1770  0.0  1.1 482856 23100 tty2    Sl+   16:43   0:00 /usr/lib/gnome  
javier    1773  0.0  1.3 1141904 26520 tty2    Sl+   16:43   0:00 /usr/lib/gnome
```

Para saber a qué usuario pertenece cada proceso de la lista extendida obtenida con la opción **-e** podemos añadir **-f**. Se nos mostrará para cada proceso, además del UID (ID de usuario del propietario del proceso), la siguiente información:

- **PID**: ID del proceso.
- **PPID**: ID del proceso padre.
- **C**: número de hijos que tiene el proceso.
- **TIEMPO**: hora a la que comenzó el proceso.
- **TTY**: nombre de la consola en la que el usuario inició sesión.
- **HORA**: cantidad de tiempo de procesamiento de la CPU que el proceso ha utilizado.
- **CMD**: nombre del comando que inició el proceso.

A continuación se muestra un ejemplo de ello.

```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1         0  0  16:39 ?        00:00:15 /sbin/init auto noprompt
root           2         0  0  16:39 ?        00:00:00 [kthreadd]
root           4         2  0  16:39 ?        00:00:00 [kworker/0:0H]
root           6         2  0  16:39 ?        00:00:00 [mm_percpu_wq]
root           7         2  0  16:39 ?        00:00:07 [ksoftirqd/0]
root           8         2  0  16:39 ?        00:00:01 [rcu_sched]
root           9         2  0  16:39 ?        00:00:00 [rcu_bh]
root          10         2  0  16:39 ?        00:00:00 [migration/0]
root          11         2  0  16:39 ?        00:00:00 [watchdog/0]
root          12         2  0  16:39 ?        00:00:00 [cpuhp/0]
root          13         2  0  16:39 ?        00:00:00 [cpuhp/1]
root          14         2  0  16:39 ?        00:00:00 [watchdog/1]
root          15         2  0  16:39 ?        00:00:00 [migration/1]
root          16         2  0  16:39 ?        00:00:01 [ksoftirqd/1]
root          18         2  0  16:39 ?        00:00:00 [kworker/1:0H]
root          19         2  0  16:39 ?        00:00:00 [kdevtmpfs]
root          20         2  0  16:39 ?        00:00:00 [netns]
root          21         2  0  16:39 ?        00:00:00 [rcu_tasks_kthre]
root          22         2  0  16:39 ?        00:00:00 [kauditd]
root          24         2  0  16:39 ?        00:00:00 [khungtaskd]
root          25         2  0  16:39 ?        00:00:00 [oom_reaper]
root          26         2  0  16:39 ?        00:00:00 [writeback]
```

Cabe mencionar que al usar la opción **-F** podemos obtener aún más columnas de información.

A veces puede resultar de utilidad ver qué procesos iniciaron otros procesos. Utilizamos la opción **-H** para hacerlo, que nos ilustrará esta jerarquía añadiendo sangrados a la lista. Para agregar un poco más de claridad, podemos pedirle al comando **ps** que agregue algunas líneas ASCII para dibujar la jerarquía como un árbol. La opción para hacer esto es **--forest**. La diferencia entre estas opciones se muestra en el ejemplo siguiente.

```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ sleep 30 &
[1] 2741
javier@ubuntu:~$ ps
  PID TTY          TIME CMD
 2733 pts/0    00:00:00 bash
 2741 pts/0    00:00:00 sleep
 2743 pts/0    00:00:00 ps
javier@ubuntu:~$ ps -H
  PID TTY          TIME CMD
 2733 pts/0    00:00:00 bash
 2741 pts/0    00:00:00 sleep
 2744 pts/0    00:00:00 ps
javier@ubuntu:~$ ps --forest
  PID TTY          TIME CMD
 2733 pts/0    00:00:00 bash
 2741 pts/0    00:00:00 \_ sleep
 2745 pts/0    00:00:00 \_ ps
javier@ubuntu:~$
```

Una vez encontrada la ID del proceso que nos interesa, puede usarse el comando **ps** para enumerar los detalles usando la opción **-p** seguida del ID del proceso. Añadiendo **-f** o **-F** podemos ver mayor cantidad de información de nuestra consulta:

```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ ps
  PID TTY          TIME CMD
 2938 pts/1        00:00:00 bash
 2946 pts/1        00:00:00 ps
javier@ubuntu:~$ ps -p 2938
  PID TTY          TIME CMD
 2938 pts/1        00:00:00 bash
javier@ubuntu:~$ ps -fp 2938
UID          PID    PPID  C STIME TTY          TIME CMD
javier        2938    2565  0 17:56 pts/1        00:00:00 bash
javier@ubuntu:~$ ps -Fp 2938
UID          PID    PPID  C  SZ  RSS  PSR STIME TTY          TIME CMD
javier        2938    2565  0  7466  5136   1 17:56 pts/1        00:00:00 bash
javier@ubuntu:~$
```

También podemos buscar procesos mediante la función **grep**, como se muestra a continuación.

```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ ps -e | grep gedit
 3022 pts/1        00:00:00 gedit
javier@ubuntu:~$
```

La opción **-C** permite buscar un proceso usando el nombre del comando que inició el proceso. En el siguiente ejemplo buscamos si hay algún proceso corriendo el popular editor de texto de Linux.

```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ ps -e | grep gedit
 3022 pts/1    00:00:00 gedit
javier@ubuntu:~$ ps -C gedit
  PID TTY          TIME CMD
 3022 pts/1    00:00:00 gedit
javier@ubuntu:~$
```

Con la opción **-o** se puede seleccionar qué columnas se desean incluir en la salida de **ps** añadiendo el especificador de la columna, que puede encontrarse en el manual del comando **ps**. Se puede ordenar además la salida utilizando la opción **--sort**. En la siguiente imagen ordenamos cada PID por el tiempo de CPU y mostramos únicamente los 10 primeros resultados (más el encabezado de la tabla).

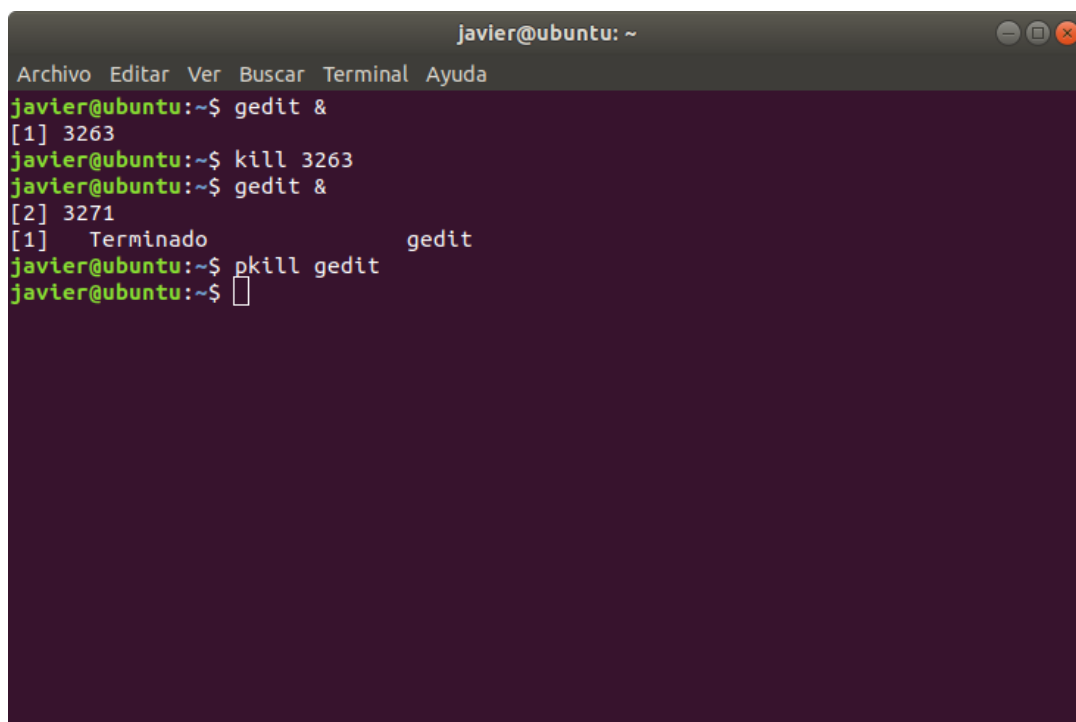
```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ ps -e -o pid,pcpu --sort -pcpu | head -11
  PID %CPU
 1640  5.7
  3205  2.7
 1834  1.9
 1504  1.7
 1082  1.6
 1841  1.6
 1250  0.7
 2565  0.7
 1006  0.4
 2117  0.4
javier@ubuntu:~$
```

## 2.2. TERMINACIÓN DE PROCESOS CON EL COMANDO KILL

Hasta aquí hemos aprendido a identificar procesos, pudiendo extraer distinta información de ellos como es el PID. Conocido éste, podemos eliminar cualquier proceso utilizando el comando **kill** seguido del

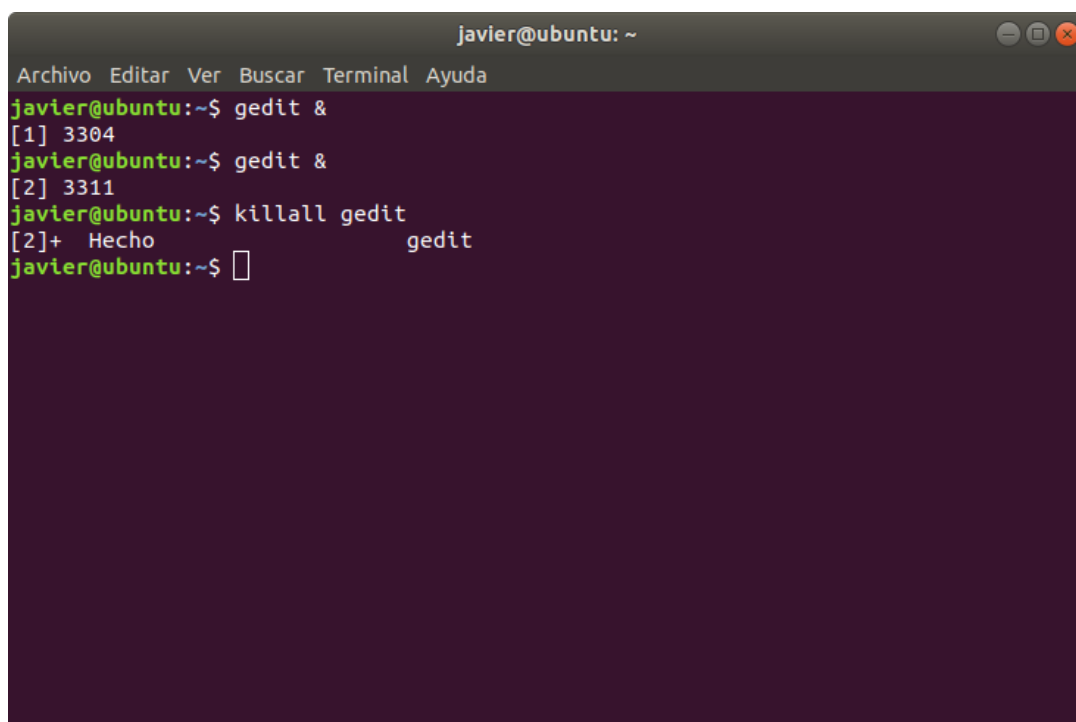


PID o el comando **kill** seguido de su nombre. A continuación se muestra un ejemplo aplicando ambos métodos.



```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ gedit &
[1] 3263
javier@ubuntu:~$ kill 3263
javier@ubuntu:~$ gedit &
[2] 3271
[1] Terminado gedit
javier@ubuntu:~$ pkill gedit
javier@ubuntu:~$
```

Para eliminar múltiples procesos con el mismo nombre podemos usar el comando **killall** seguido del nombre.

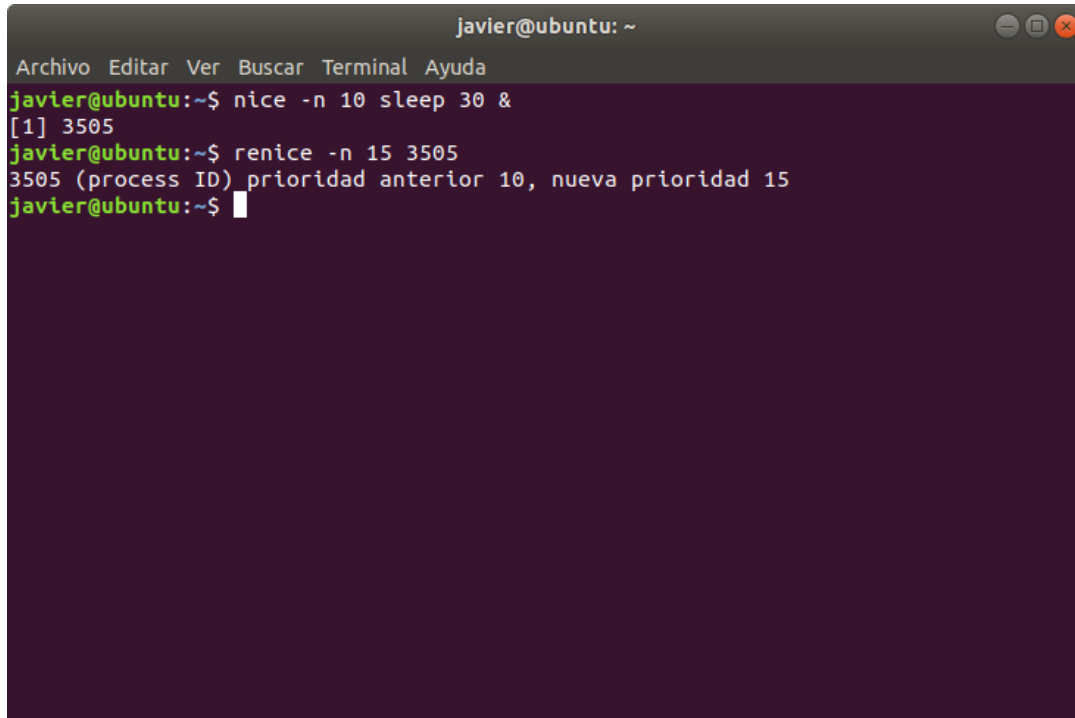


```
javier@ubuntu: ~
Archivo Editar Ver Buscar Terminal Ayuda
javier@ubuntu:~$ gedit &
[1] 3304
javier@ubuntu:~$ gedit &
[2] 3311
javier@ubuntu:~$ killall gedit
[2]+ Hecho gedit
javier@ubuntu:~$
```

Podemos añadir el parámetro **-w**, el cual espera a que los procesos terminen haciendo dicha comprobación cada segundo. También es interesante conocer el parámetro **-u**, que finaliza solo los procesos ejecutados como usuario. Si ejecutamos **killall -u <Nombre de usuario>** se terminarán todos los procesos ejecutados por dicho usuario.

## 2.3. GESTIÓN DE PRORIDADES CON EL COMANDO NICE

El comando **nice** en Linux nos permite modificar la prioridad de un proceso frente al resto. Solo los usuarios root pueden modificar la prioridad de todos los procesos. Los demás usuarios podrán modificar la prioridad de los procesos de los que son propietarios. La sintaxis del comando **nice** es la siguiente: **nice -n <PRIORIDAD> <COMANDO>**. Para modificar la prioridad de un proceso existente se empleará **renice -n <NUEVA PRIORIDAD> <PID>**. A continuación se muestra un ejemplo en el que se crea un proceso de tipo **sleep** con una prioridad dada que posteriormente se modifica.

A screenshot of a terminal window titled 'javier@ubuntu: ~'. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows the following commands and output:

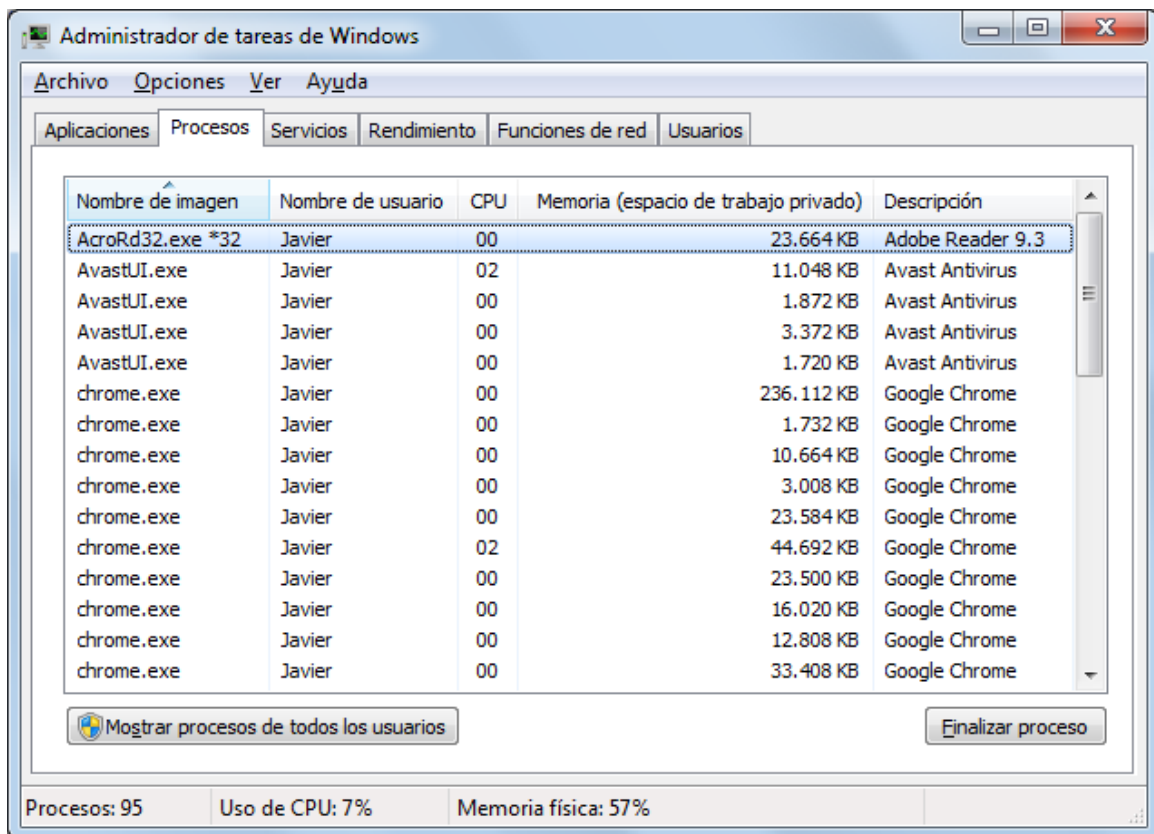
```
javier@ubuntu:~$ nice -n 10 sleep 30 &
[1] 3505
javier@ubuntu:~$ renice -n 15 3505
3505 (process ID) prioridad anterior 10, nueva prioridad 15
javier@ubuntu:~$
```

El rango de valores **nice** abarca desde -20 (más favorable al proceso) hasta 19 (menos favorable al proceso).

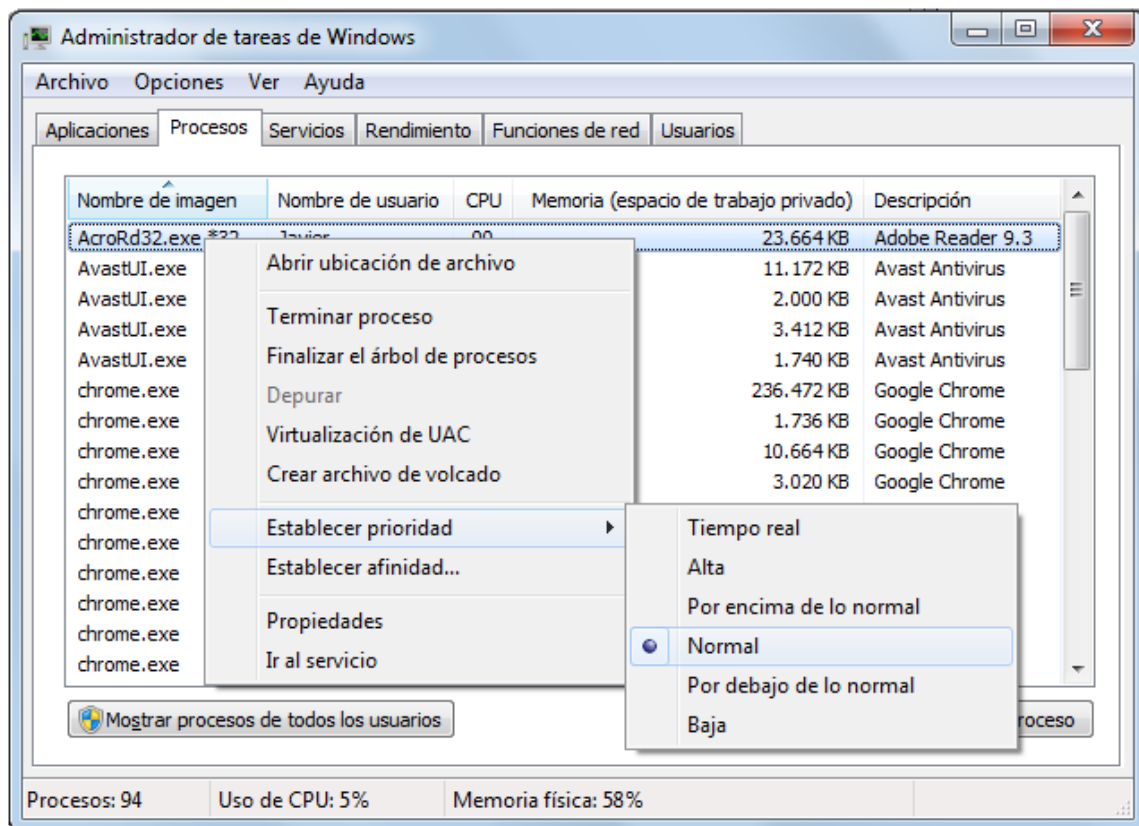
## 3. GESTIÓN DE PROCESOS EN WINDOWS

### 3.1. GESTIÓN DE PROCESOS A TRAVÉS DE LA INTERFAZ GRÁFICA

Para administrar los procesos en Windows podremos optar por usar la interfaz del sistema operativo o el terminal **CMD**. Si optamos por usar la interfaz gráfica, tendremos que dirigirnos al Administrador de tareas, que podremos encontrar fácilmente utilizando el buscador de Windows o usando el atajo [Ctrl] + [Alt] + [Supr]. Una vez ahí, seleccionaremos la pestaña Procesos.



Como vemos, en la parte inferior derecha encontramos un botón que nos permitirá finalizar el proceso. Para ver más opciones podemos hacer click derecho con el ratón sobre el proceso de interés.



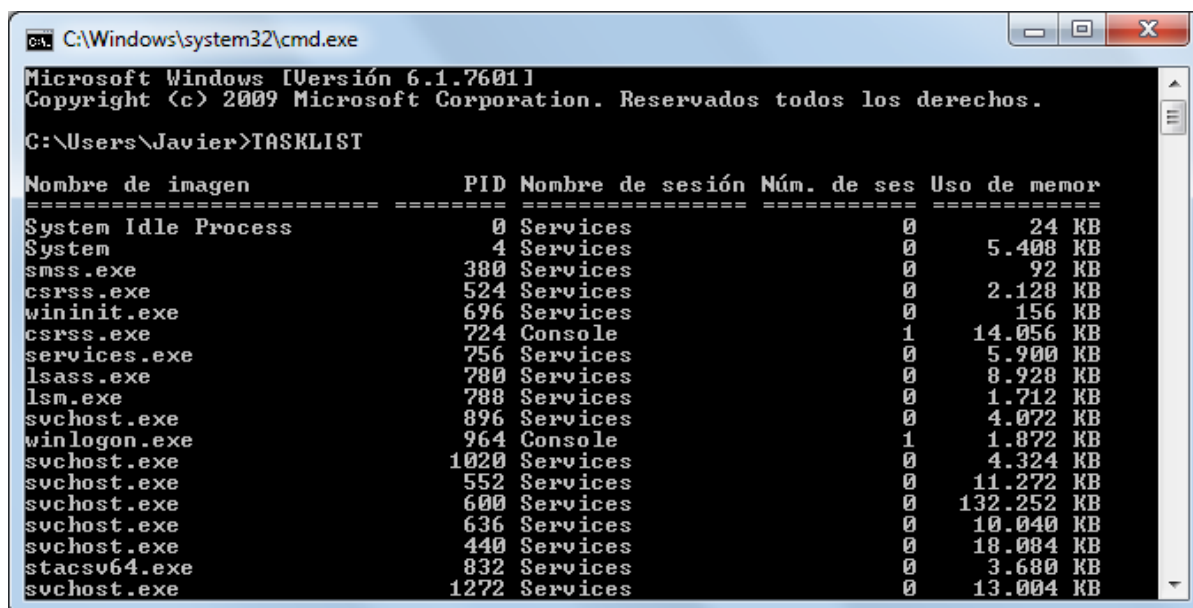
Como vemos, de esta manera podemos elegir terminar el proceso, finalizar el árbol de procesos o establecer su prioridad entre otras opciones.

## 3.2. GESTIÓN DE PROCESOS A TRAVÉS DEL TERMINAL

En cuanto a la administración de procesos a través del terminal, los comandos **TASKLIST** y **TASKKILL** son fundamentales, ya que nos permiten obtener información, crear listas detalladas y detener aplicaciones, tareas y procesos aun cuando están bloqueados y no responden. Además, usarlos es algo sencillo, incluso si no se tiene experiencia en el uso de la línea de comandos.

En primer lugar abriremos el **CMD**. Para ello pulsaremos el símbolo de Windows y la tecla R al mismo tiempo. Se abrirá una pequeña ventana con el título de Ejecutar. Dentro del rectángulo de texto escribiremos **CMD** y pulsaremos sobre el botón de aceptar. Ya tendremos nuestro terminal operativo.

Escribiendo **TASKLIST** en el terminal nos aparecerá una lista de procesos donde se indicará su nombre, PID y uso de memoria entre otros:

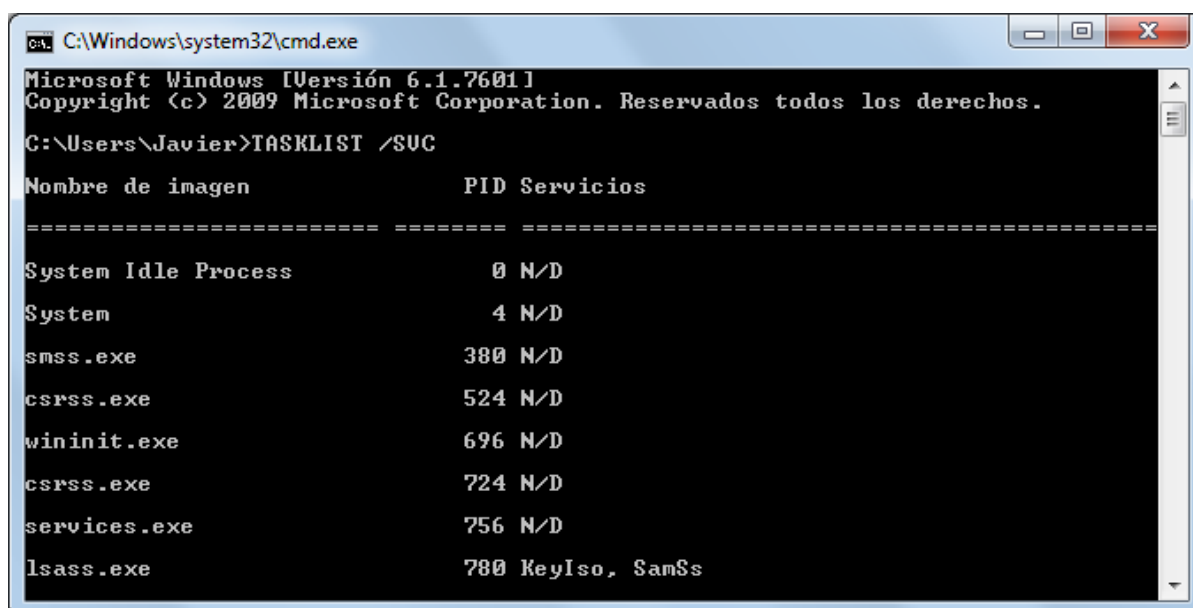


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Javier>TASKLIST

Nombre de imagen          PID Nombre de sesión Núm. de ses Uso de memor
=====
System Idle Process       0 Services          0         24 KB
System                    4 Services          0        5.408 KB
smss.exe                 380 Services          0         92 KB
csrss.exe                524 Services          0        2.128 KB
wininit.exe              696 Services          0         156 KB
csrss.exe                724 Console          1       14.056 KB
services.exe             756 Services          0        5.900 KB
lsass.exe                780 Services          0        8.928 KB
lsm.exe                  788 Services          0        1.712 KB
svchost.exe              896 Services          0        4.072 KB
winlogon.exe             964 Console          1        1.872 KB
svchost.exe             1020 Services          0        4.324 KB
svchost.exe              552 Services          0       11.272 KB
svchost.exe              600 Services          0      132.252 KB
svchost.exe              636 Services          0       10.040 KB
svchost.exe              440 Services          0       18.084 KB
stacsv64.exe             832 Services          0        3.680 KB
svchost.exe             1272 Services          0       13.004 KB
```

El parámetro **/V** nos muestra información detallada de cada tarea ejecutándose, mientras que **/SVC** muestra información adicional de los servicios hospedados en cada proceso, como podemos ver en el siguiente ejemplo.

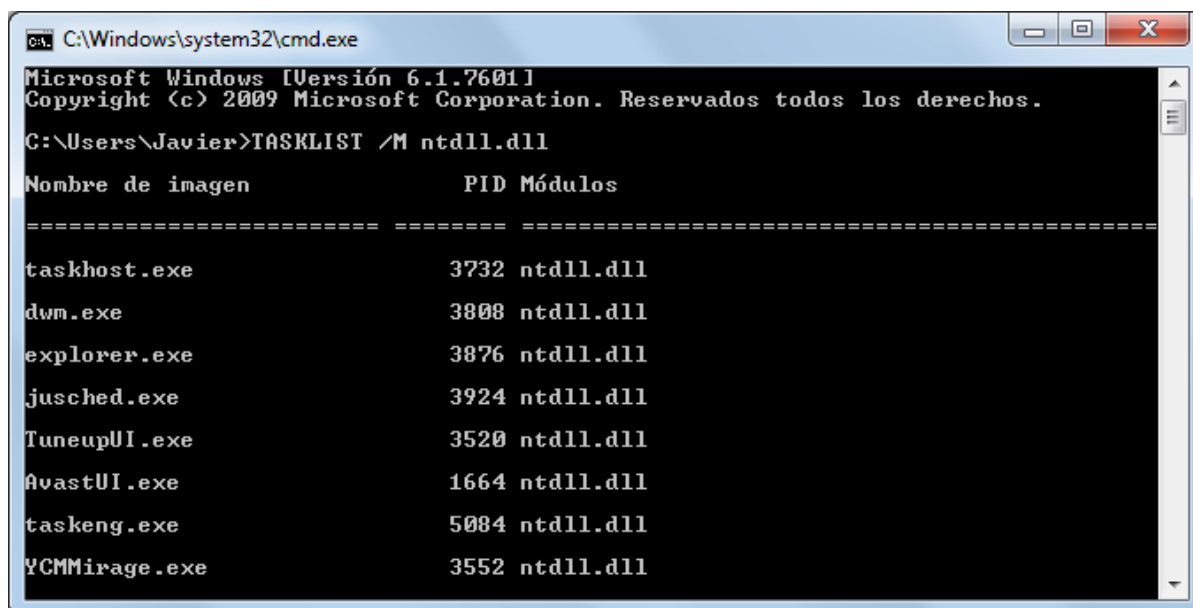


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Javier>TASKLIST /SVC

Nombre de imagen          PID Servicios
=====
System Idle Process       0 N/D
System                    4 N/D
smss.exe                 380 N/D
csrss.exe                524 N/D
wininit.exe              696 N/D
csrss.exe                724 N/D
services.exe             756 N/D
lsass.exe                780 KeyIso, SamSs
```

Otro parámetro interesante es /M, que seguido de un módulo .dll o. exe muestra todas las tareas que usan dicho módulo:

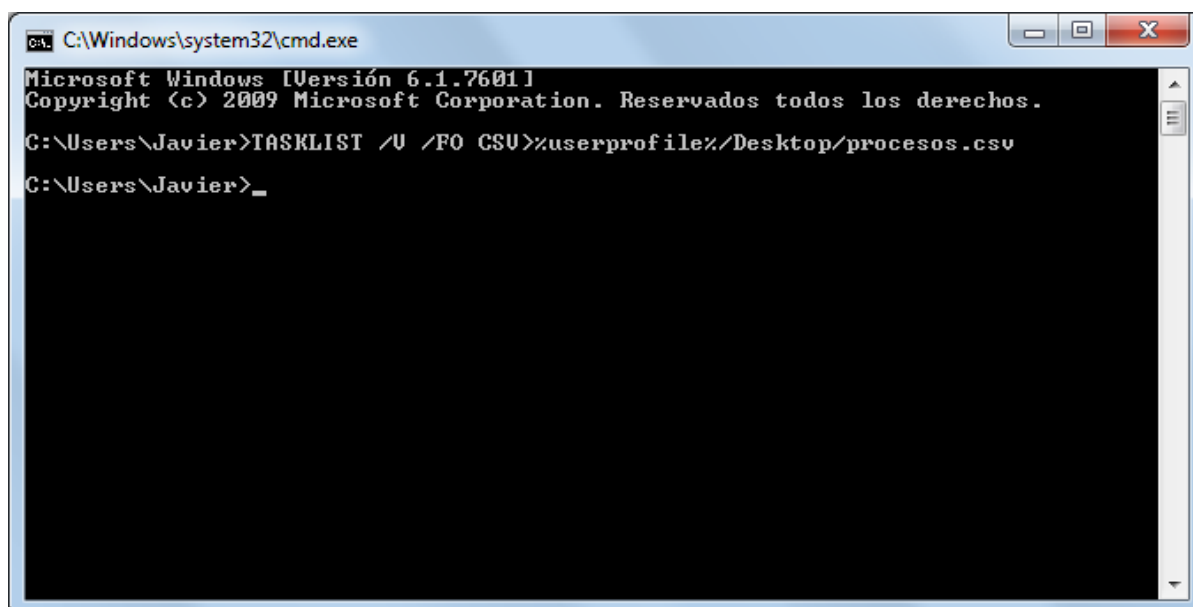


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Javier>TASKLIST /M ntdll.dll

Nombre de imagen                PID Módulos
=====
taskhost.exe                    3732 ntdll.dll
dwm.exe                         3808 ntdll.dll
explorer.exe                    3876 ntdll.dll
jusched.exe                     3924 ntdll.dll
TuneupUI.exe                    3520 ntdll.dll
AvastUI.exe                     1664 ntdll.dll
taskeng.exe                     5084 ntdll.dll
YCMMirage.exe                   3552 ntdll.dll
```

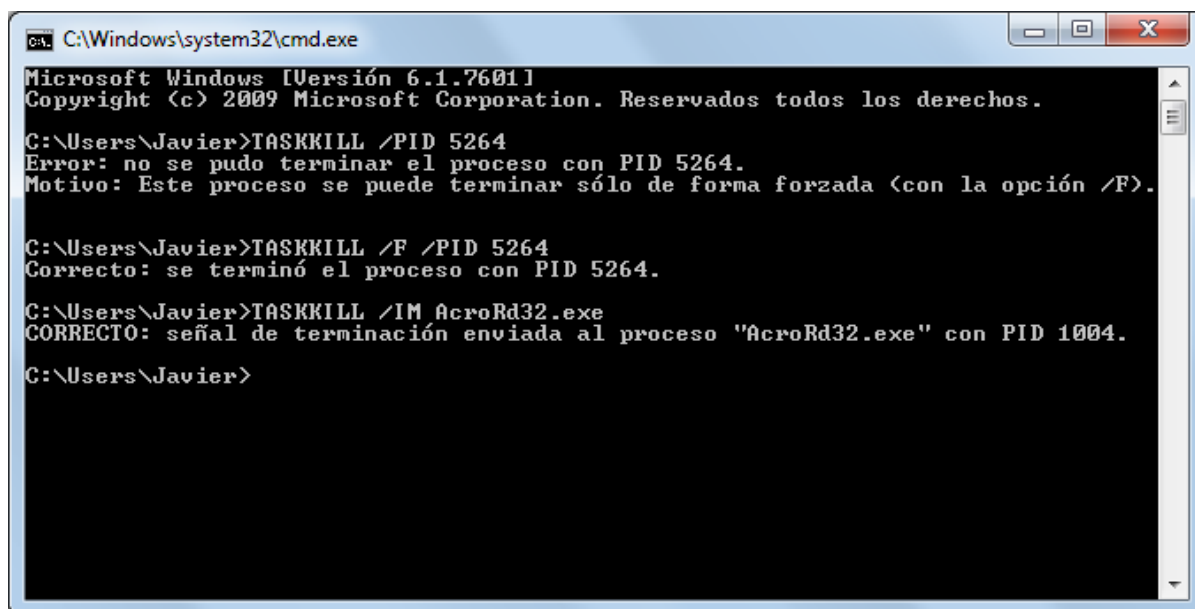
Finalmente cabe mencionar el parámetro /FO, que exporta al directorio señalado la lista de los procesos en ejecución con formato TABLE, LIST o CSV. Por ejemplo, para crear en el escritorio una lista detallada de los procesos en ejecución en un archivo CSV usaremos el comando mostrado en la siguiente figura.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Javier>TASKLIST /U /FO CSV %userprofile%\Desktop/procesos.csv
C:\Users\Javier>_
```

Hablaremos finalmente de TASKKILL, comando que complementa al anterior permitiendo detener tareas o procesos usando el PID (TASKKILL /PID <ID DEL PROCESO>) o el nombre (TASKKILL /IM <NOMBRE DE TAREA>). En ocasiones se nos pedirá incluir el parámetro /F para poder realizar el cierre de forma forzada, como se muestra en el ejemplo siguiente.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Javier>TASKKILL /PID 5264
Error: no se pudo terminar el proceso con PID 5264.
Motivo: Este proceso se puede terminar sólo de forma forzada (con la opción /F).

C:\Users\Javier>TASKKILL /F /PID 5264
Correcto: se terminó el proceso con PID 5264.

C:\Users\Javier>TASKKILL /IM AcroRd32.exe
CORRECTO: señal de terminación enviada al proceso "AcroRd32.exe" con PID 1004.

C:\Users\Javier>
```

Para cerrar todos los procesos secundarios iniciados por una tarea podemos usar el parámetro `/T`. Así, el comando `TASKKILL /F /IM cmd.exe /T` cerraría la consola de **CMD** y todos los procesos secundarios iniciados por ella.

## 4. CONCLUSIONES

En este documento se ha presentado el concepto de proceso, se ha visto su importancia y se mostrado cómo trabajar con ellos en diferentes sistemas, tanto mediante el uso de la interfaz como del terminal. A pesar de haber visto esto de forma superficial, se ha comprobado que mientras que el uso de la interfaz hace de esta gestión una tarea sencilla, un buen conocimiento de los comandos ofrece más posibilidades sin sacrificar con ello la comodidad o rapidez del proceso. Sin duda, el uso de comandos en terminales resulta un campo infinitamente útil y en cuyo estudio merece la pena invertir tiempo.

## BIBLIOGRAFÍA

- [1] *Sistemas Informáticos*, Apuntes de Cesur.
- [2] <http://www.atc.uniovi.es/telematica/2ac/> (Última visita: 15/01/2021)
- [3] <https://eltallerdelbit.com/comandos-procesos-linux/> (Última visita: 15/01/2021)
- [4] <https://elmanualdelmundo.blogspot.com/2019/11/como-usar-el-comando-ps-para-monitorear.html>  
(Última visita: 15/01/2021)
- [5] <https://rm-rf.es/asignar-prioridad-de-cpu-a-procesos-en-linux-con-nice/> (Última visita:  
15/01/2021)
- [6] <https://nksistemas.com/comandos-tasklist-y-taskkill/> (Última visita: 15/01/2021)