

PRACTICA PROGRAMACIÓN SERVICIOS Y PROCESOS - 2º DAM

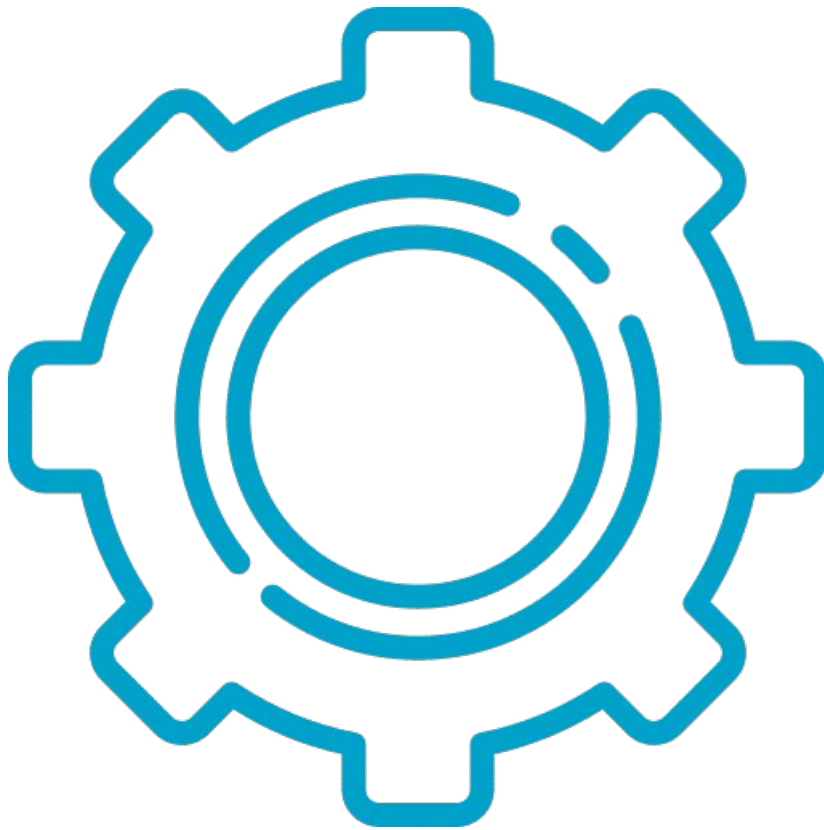
UT1. - Programación multiproceso

Apellidos: Gallego León

Nombre: José Enrique

Nº PC: PC personal

Fecha: 03/11/2023



Ej. 1 – Identificando procesos. Trabajando con procesos en los S.O. - 1,5 ptos

- 1. Identifica tres procesos y tres servicios activos en tu ordenador.
- 2. Indica la información más importante: como nombre, PID y descripción.
- 3. Explica de forma detallada cómo has obtenido esta información, capturas del proceso (capturas de pantalla) y alternativas para obtener la información anterior.

Windows

1.) Para ver los distintos procesos del sistema podemos hacer lo siguiente: “Clic derecho sobre el icono de Windows>Administrador de tareas>Procesos”

Nombre	Estado	7% CPU	80% Memoria	2% Disco	0% Red
Aplicaciones (5)					
> Administrador de tareas		4,2%	51,4 MB	0 MB/s	0 Mbps
> Explorador de Windows		0,2%	55,7 MB	0,1 MB/s	0 Mbps
> LibreOffice (2)		0%	270,7 MB	0 MB/s	0 Mbps
> VirtualBox Manager		0%	7,8 MB	0 MB/s	0 Mbps
> VirtualBox Virtual Machine		0%	71,8 MB	0 MB/s	0 Mbps
Procesos en segundo plano (...)					
Aislamiento de gráficos de dis...		0%	10,2 MB	0 MB/s	0 Mbps

Así podríamos ver los distintos procesos que se están ejecutando en el sistema. Tres de los procesos que podemos identificar serían los siguientes:

“Administrador de tareas”, “Explorador de Windows” y “VirtualBox Manager”.

Con respecto a los servicios:

“AarSvc (Agent Activation Runtime)”, “AJRouter (Servicio de enrutador de AllJoyn)” y “Audiosrv (Audio de Windows)”.

2.) En los servicios podemos identificar “Nombre”, “PID”, “Descripción”, “Estado” y “Grupo”, de los anteriores servicios tenemos la siguiente información.

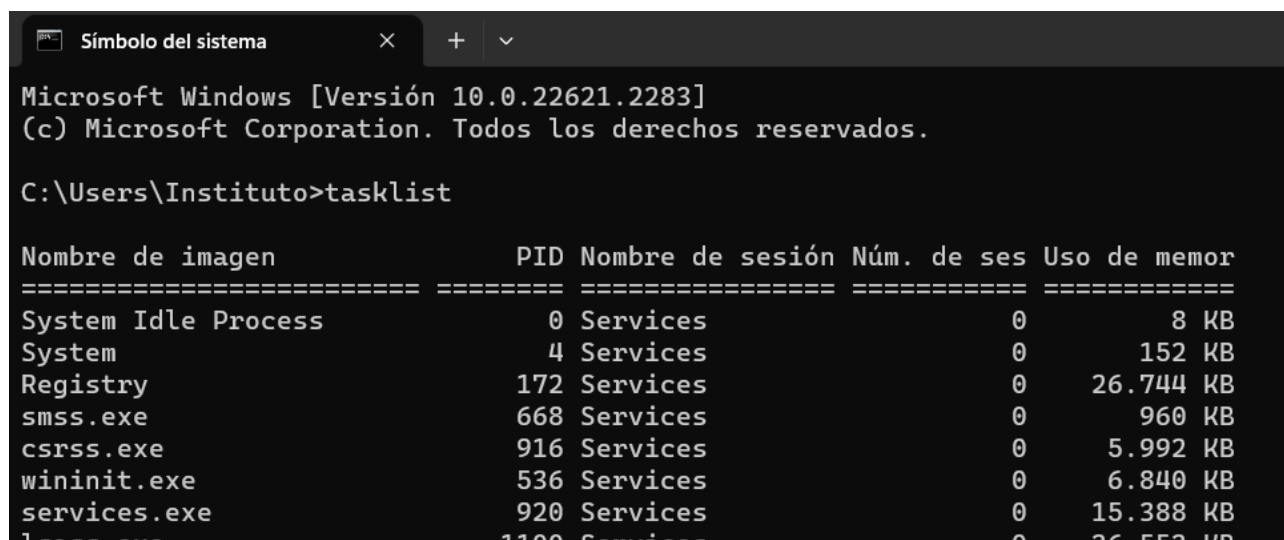
Nombre	PID	Descripción	Estado	Grupo
AarSvc		Agent Activation Runtime	Detenido	AarSvcGroup
AJRouter		Servicio de enrutador de AllJoyn	Detenido	LocalServiceNetworkRestricted
Audiosrv	3392	Audio de Windows	En ejecución	LocalServiceNetworkRestricted

3.) Existen varios métodos para identificar los distintos procesos que se están ejecutando en Windows. Lo podríamos hacer desde la opción del administrador de tareas de tareas, “*Clic derecho sobre el icono de Windows>Administrador de tareas>Procesos*”

Esta sería la manera sencilla de ver los procesos que se están ejecutando en el sistema, ahora veremos como acceder a la información sobre los distintos procesos y servicios desde la consola de Windows.

Para acceder a la consola de Windows podremos hacerlo desde el propio buscador pulsando sobre el icono de Windows, o presionando Tecla Window + R y escribiendo “CMD”

Una vez en el CMD escribiremos “tasklist” y ahí nos saldrán todos los procesos que se estan ejecutando en ese momento en el sistema.



```
Microsoft Windows [Versión 10.0.22621.2283]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Instituto>tasklist

Nombre de imagen                PID Nombre de sesión Núm. de ses Uso de memor
=====
System Idle Process             0 Services              0          8 KB
System                          4 Services              0         152 KB
Registry                       172 Services            0        26.744 KB
smss.exe                       668 Services            0          960 KB
csrss.exe                      916 Services            0         5.992 KB
wininit.exe                    536 Services            0         6.840 KB
services.exe                   920 Services            0        15.388 KB
lsass.exe                     1100 Services            0        26.552 KB
```

Linux

1.) Para ver los procesos que se están ejecutando en Linux tendremos que hacerlo desde la consola de Linux con el comando “top”.

```
jgalleo293@jgalleo293-VirtualBox: ~
top - 18:22:27 up 19 min, 2 users, load average: 0.41, 0.71, 0.75
Tareas: 212 total, 1 ejecutar, 211 hibernar, 0 detener, 0 zombie
%Cpu(s): 0,8 us, 0,3 sy, 0,0 ni, 99,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 2711,6 total, 75,1 free, 1885,2 used, 962,2 buff/cache
MiB Intercambio: 2704,0 total, 2438,6 free, 265,4 used, 826,4 avail Mem

  PID  USUARIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  HORA+  ORDEN
    998  jgalleo+  20   0 4930832 332236 91060 S   6,0  12,0  4:26.70  gnome-shell
   3846  jgalleo+  20   0 3102960 312308 119740 S   1,7  11,2  1:03.83  firefox
   4203  jgalleo+  20   0 2457000 124012 83320 S   1,3   4,5  0:09.39  Privileged Cont
   6111  jgalleo+  20   0 628804 54012 43080 S   1,3   1,9  0:01.97  gnome-terminal-
    24   root    20   0      0      0      0 I   0,3   0,0  0:00.59  kworker/1:0-events
   1110  jgalleo+  20   0 386108 6144 5248 S   0,3   0,2  0:00.09  gvfs-afc-volume
   1730  jgalleo+  20   0 489660 20452 13676 S   0,3   0,7  0:00.60  update-notifier
   6141  jgalleo+  20   0 11996 5760 3584 R   0,3   0,2  0:00.95  top
     1   root    20   0 169876 11860 6740 S   0,0   0,4  0:06.06  systemd
     2   root    20   0      0      0      0 S   0,0   0,0  0:00.00  kthreadd
     3   root     0 -20      0      0      0 I   0,0   0,0  0:00.00  rcu_gp
     4   root     0 -20      0      0      0 I   0,0   0,0  0:00.00  rcu_par_gp
     5   root     0 -20      0      0      0 I   0,0   0,0  0:00.00  slub_flushwq
     6   root     0 -20      0      0      0 I   0,0   0,0  0:00.00  netns
     8   root     0 -20      0      0      0 I   0,0   0,0  0:00.00  kworker/0:0H-events_highpri
     9   root    20   0      0      0      0 I   0,0   0,0  0:01.95  kworker/u8:0-events_power_efficient
    10   root     0 -20      0      0      0 I   0,0   0,0  0:00.00  mm_percpu_wq
    11   root    20   0      0      0      0 I   0,0   0,0  0:00.00  rcu_tasks_kthread
    12   root    20   0      0      0      0 I   0,0   0,0  0:00.00  rcu_tasks_rude_kthread
    13   root    20   0      0      0      0 I   0,0   0,0  0:00.00  rcu_tasks_trace_kthread
    14   root    20   0      0      0      0 S   0,0   0,0  0:00.31  ksoftirqd/0
    15   root    20   0      0      0      0 I   0,0   0,0  0:01.27  rcu_preempt
    16   root    rt   0      0      0      0 S   0,0   0,0  0:00.01  migration/0
    17   root   -51   0      0      0      0 S   0,0   0,0  0:00.00  idle_inject/0
    18   root    20   0      0      0      0 I   0,0   0,0  0:00.25  kworker/0:1-events
    19   root    20   0      0      0      0 S   0,0   0,0  0:00.00  cpuhp/0
```

Tres procesos a identificar:

“Firefox”, “Gnome-shell” y “gnome-terminal”.

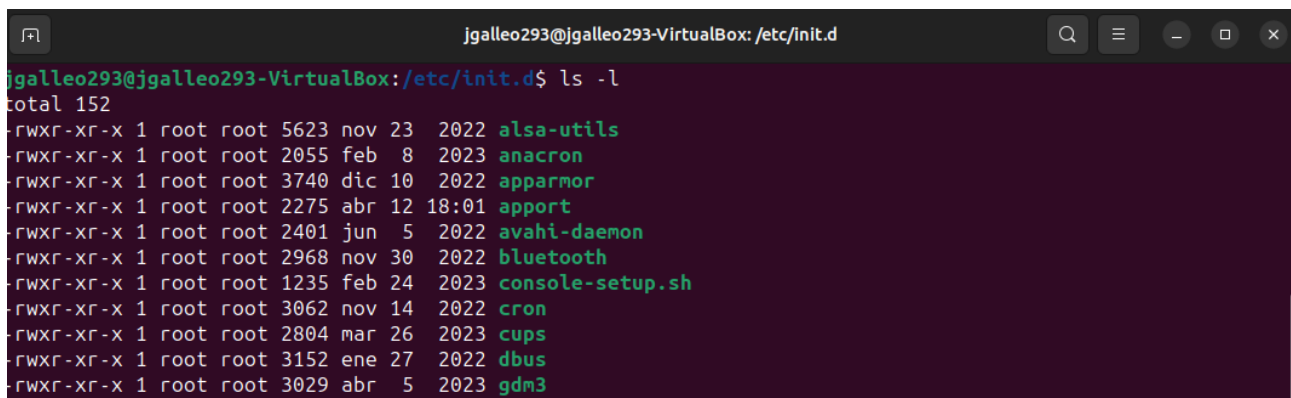
Para ver los servicios tendríamos que poner en la consola el comando “ls” desde el directorio “/etc/init.d”.

```
jgalleo293@jgalleo293-VirtualBox: /etc/init.d
jgalleo293@jgalleo293-VirtualBox:~$ su
Contraseña:
su: Fallo de autenticación
jgalleo293@jgalleo293-VirtualBox:~$ su
Contraseña:
su: Fallo de autenticación
jgalleo293@jgalleo293-VirtualBox:~$ service -- status-all
.: unrecognized service
jgalleo293@jgalleo293-VirtualBox:~$ cd /etc/init.d
jgalleo293@jgalleo293-VirtualBox:~$ cd /etc/init.d
jgalleo293@jgalleo293-VirtualBox:/etc/init.d$ ls
alsa-utils          dbus                openvpn             udev
anacron              gdm3                plymouth            ufw
apparmor             grub-common         plymouth-log        unattended-upgrades
apport              hwclock.sh          procps              uuid
avahi-daemon         irqbalance          rsync               whoopsie
bluetooth            kerneloops          saned               x11-common
console-setup.sh    keyboard-setup.sh  speech-dispatcher
cron                 knod                spice-vdagent
cups                 open-vm-tools       sssd
jgalleo293@jgalleo293-VirtualBox:/etc/init.d$
```

Tres de los servicios que podemos ver son:

“Openvpn”, “dbus” y “Bluetooth”.

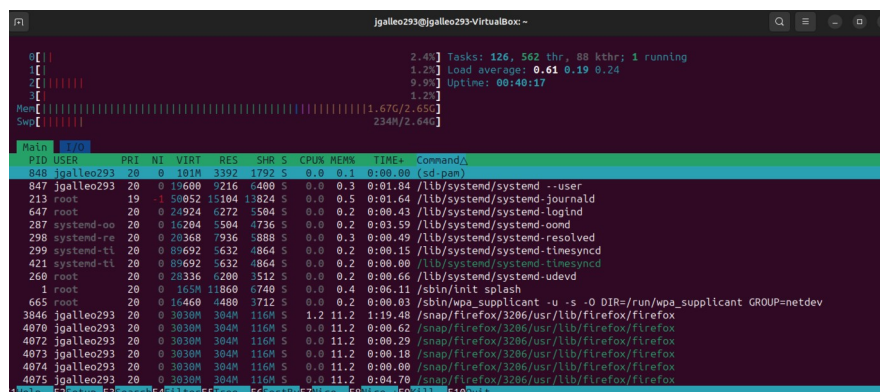
2.) Para ver más información sobre los servicios en Linux debemos usar el mismo comando añadiendo “-l” de la siguiente manera:



```
jgalleo293@jgalleo293-VirtualBox: /etc/init.d$ ls -l
total 152
-rwxr-xr-x 1 root root 5623 nov 23 2022 alsa-utils
-rwxr-xr-x 1 root root 2055 feb  8 2023 anacron
-rwxr-xr-x 1 root root 3740 dic 10 2022 apparmor
-rwxr-xr-x 1 root root 2275 abr 12 18:01 apport
-rwxr-xr-x 1 root root 2401 jun  5 2022 avahi-daemon
-rwxr-xr-x 1 root root 2968 nov 30 2022 bluetooth
-rwxr-xr-x 1 root root 1235 feb 24 2023 console-setup.sh
-rwxr-xr-x 1 root root 3062 nov 14 2022 cron
-rwxr-xr-x 1 root root 2804 mar 26 2023 cups
-rwxr-xr-x 1 root root 3152 ene 27 2022 dbus
-rwxr-xr-x 1 root root 3029 abr  5 2023 gdm3
```

Aquí podemos ver el total de servicios que tiene el sistema, el nombre de cada servicio, donde se encuentran, la fecha de instalación, etc.

3.) Para obtener la información anterior tendríamos que abrir la consola o terminal de Linux, ahí para poder ver todos los procesos podemos usar el comando top, pero si queremos poder gestionar la información que nos muestran de los procesos debemos instalar el comando “htop”. Esto se hace con el siguiente comando: “sudo apt-get install htop”. Una vez hecho esto podremos ejecutar htop



```
jgalleo293@jgalleo293-VirtualBox: ~$ top
top: 2.4% Tasks: 126, 562 thr, 88 kthr; 1 running
top: 1.2% Load average: 0.61 0.19 0.24
top: 9.9% Uptime: 00:40:17
top: 1.2%
top: 1.67G/2.65G
top: 234M/2.64G

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 848 jgalleo293 20    0 101M 3392 1792 S  0.0  0.1  0:00.00 /sd-pan
 847 jgalleo293 20    0 19600 7216 6400 S  0.0  0.3  0:01.84 /lib/systemd/systemd--user
213 root      19   -1 58052 15104 13824 S  0.0  0.5  0:01.64 /lib/systemd/systemd-journald
647 root      20    0 24924 6272 5584 S  0.0  0.2  0:00.43 /lib/systemd/systemd-logind
287 systemd-oo 20    0 16204 5504 4736 S  0.0  0.2  0:03.59 /lib/systemd/systemd-oomd
298 systemd-re 20    0 20368 7936 5888 S  0.0  0.3  0:00.49 /lib/systemd/systemd-resolved
299 systemd-tl 20    0 89692 5632 4864 S  0.0  0.2  0:00.15 /lib/systemd/systemd-timesyncd
421 systemd-tl 20    0 89692 5632 4864 S  0.0  0.2  0:00.00 /lib/systemd/systemd-timesyncd
260 root      20    0 28336 6200 3512 S  0.0  0.2  0:00.66 /lib/systemd/systemd-udev
   1 root      20    0 165M 11860 6740 S  0.0  0.4  0:06.11 /sbin/init splash
665 root      20    0 16460 4480 3712 S  0.0  0.2  0:00.03 /sbin/wpa_supplicant -u -s -O DIR=/run/wpa_supplicant GROUP=netdev
3846 jgalleo293 20    0 3830M 304M 116M S 11.2 11.2 1:19.48 /snap/firefox/3206/usr/lib/firefox/firefox
4070 jgalleo293 20    0 3830M 304M 116M S  0.0 11.2  0:00.62 /snap/firefox/3206/usr/lib/firefox/firefox
4072 jgalleo293 20    0 3830M 304M 116M S  0.0 11.2  0:00.29 /snap/firefox/3206/usr/lib/firefox/firefox
4073 jgalleo293 20    0 3830M 304M 116M S  0.0 11.2  0:00.18 /snap/firefox/3206/usr/lib/firefox/firefox
4074 jgalleo293 20    0 3830M 304M 116M S  0.0 11.2  0:00.00 /snap/firefox/3206/usr/lib/firefox/firefox
4075 jgalleo293 20    0 3830M 304M 116M S  0.0 11.2  0:04.70 /snap/firefox/3206/usr/lib/firefox/firefox
```

Para los servicios desde la terminal o consola tendremos que situarnos en el directorio, “/etc/init.d” y ahí escribimos “ls -l” para que nos muestre información detallada sobre los servicios de Linux.

Ej. 2 – Comunicación entre procesos. Redireccionando la E/S de procesos en el shell del S.O. - 1,5 ptos

- 1. Explica los operadores de redireccionamiento existentes en los sistemas operativos y pon un ejemplo de su posible uso. En la explicación se incluirán capturas de pantalla que demuestren la realización de la tarea.

Los operadores de redireccionamiento son caracteres especiales que se pueden utilizar en conjunto con los comandos que se escriben en la consola, se utilizan para redirigir la entrada o la salida del comando.

Existen varios caracteres especiales tanto en Windows como Linux para el redireccionamiento, aunque los más utilizados son los símbolos “>” y “<”. Estos tienen distintas variaciones, adjunto una tabla con los caracteres especiales de redireccionamiento más usados, veremos también algunos ejemplos con estos operadores de redireccion:

Windows (Operadores de redireccion)

Redirection Operators Cheat Sheet		
Redirection Operator	Explanation	Example
>	The greater-than sign is used to send to a file, or even a printer or other device, whatever information from the command would have been displayed in the Command Prompt window had you not used the operator.	assoc > types.txt
>>	The double greater-than sign works just like the single greater-than sign but the information is appended to the end of the file instead of overwriting it.	ipconfig >> netdata.txt
<	The less-than sign is used to read the input for a command from a file instead of from the keyboard.	sort < data.txt
	The vertical pipe is used to read the output from one command and use it for the input of another.	dir sort

El comando “ipconfig” en Windows, se nos muestra la configuración de la red a la que pertenecemos, con el operador de redirección “>” podemos pasar esa configuración por ejemplo a un documento de texto y guardar la información ahí.

```
Administrador: Símbolo del sistema

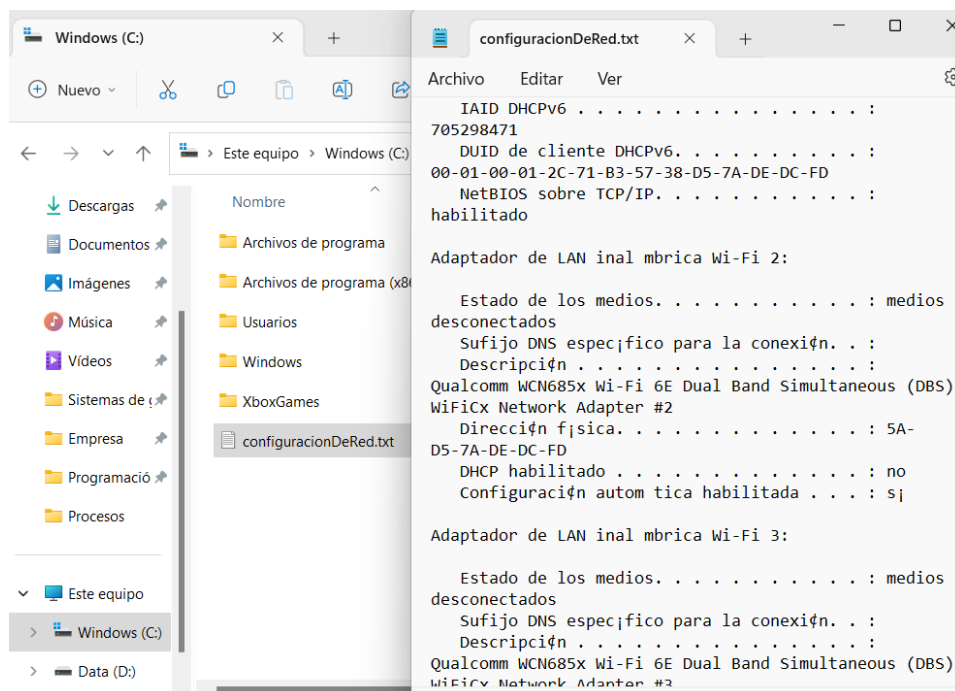
Descripción . . . . . : Qualcomm WCN685x Wi-Fi 6E Dual Band Simultaneous (DBS) WiFiCx Network Ada
pter
Dirección física. . . . . : 
DHCP habilitado . . . . . : sí
Configuración automática habilitada . . . : sí
Vínculo: dirección IPv6 local. . . : 
Dirección IPv4. . . . . : 
Máscara de subred . . . . . : 
Concesión obtenida. . . . . : martes, 10 de octubre de 2023 8:19:21
La concesión expira . . . . . : miércoles, 18 de octubre de 2023 8:07:15
Puerta de enlace predeterminada . . . . . : 
Servidor DHCP . . . . . : 
IAID DHCPv6 . . . . . : 
DUID de cliente DHCPv6. . . . . : 
Servidores DNS. . . . . : 

NetBIOS sobre TCP/IP. . . . . : habilitado

Adaptador de Ethernet Conexión de red Bluetooth:

Estado de los medios. . . . . : medios desconectados
Sufijo DNS específico para la conexión. . : 
Descripción . . . . . : Bluetooth Device (Personal Area Network)
Dirección física. . . . . : 
DHCP habilitado . . . . . : sí
Configuración automática habilitada . . . : sí

C:\Windows\System32>ipconfig/all>C:\configuracionDeRed.txt
```

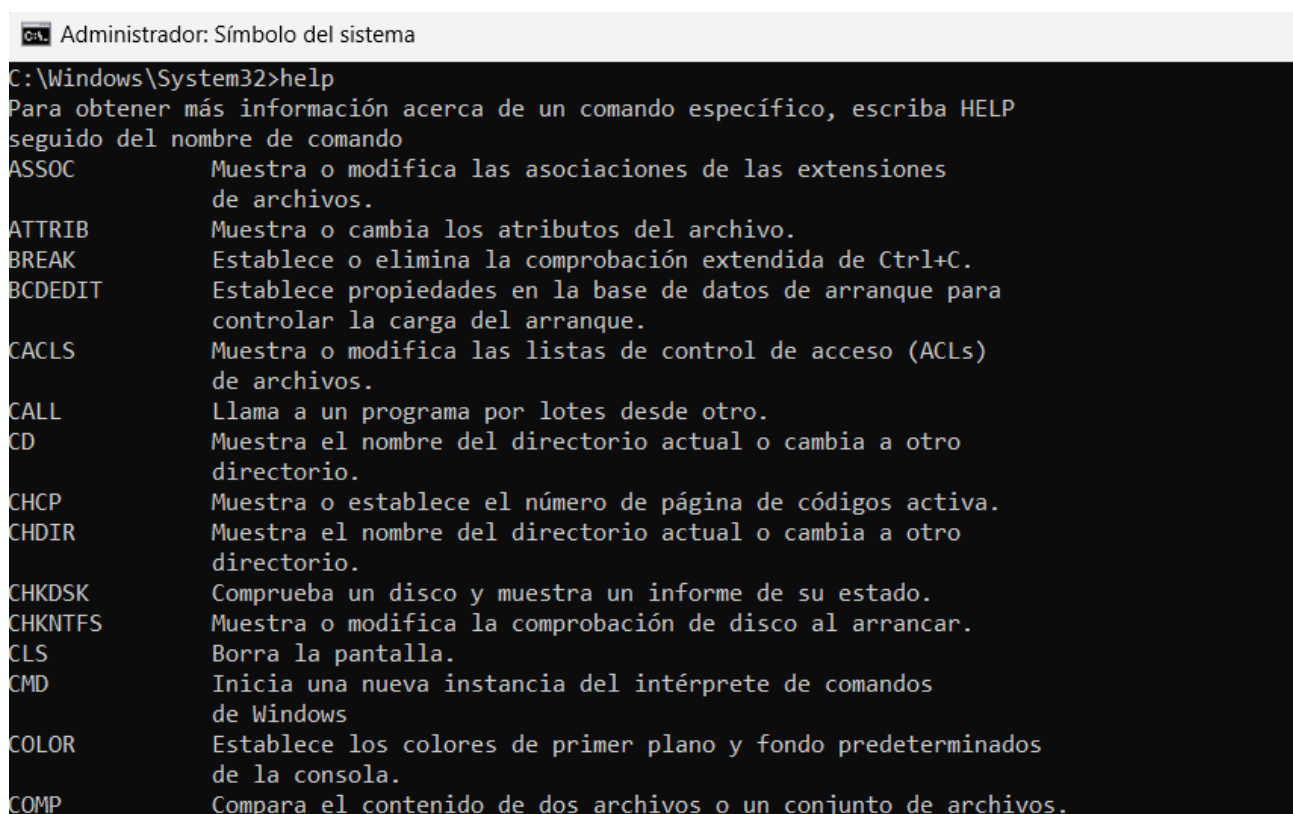


También podríamos utilizar el operador de redirección “|” para copiar al portapapeles la información de ipconfig.



Para poder moverte por la consola de Windows existen distintos comandos que te muestran información muy interesante sobre el sistema o también te permiten crear distintos archivos y/o ficheros en el sistema. Veamos algunos de ellos:

Comando “*help*” muestra todo los tipos de comandos que hay en la consola



Comando “tree” sirve para mostrar los directorios existentes en el sistema

```
C:\Windows\System32>tree
Listado de rutas de carpetas para el volumen Windows
El número de serie del volumen es FC64-328A
C:
├── 0409
├── 1028
├── 1029
├── 1031
├── 1033
├── 1036
├── 1040
├── 1041
├── 1042
├── 1045
├── 1046
├── 1049
├── 1055
├── 2052
├── 3082
├── A-Volute
│   └── AGSConfigurator
│       ├── NahimicAP04
│       └── NahimicV3
├── AdvancedInstallers
└── AMD
```

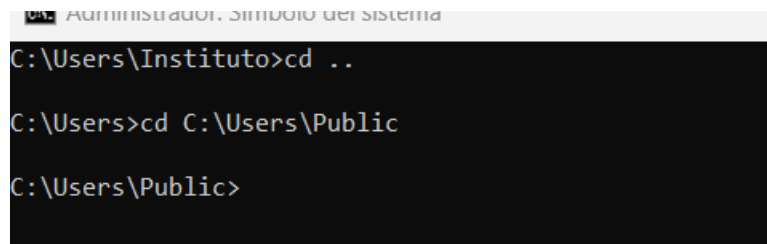
Comando “dir” muestra una lista de directorios y subdirectorios en un directorio elegido

```
C:\Users\Instituto>dir
El volumen de la unidad C es Windows
El número de serie del volumen es: FC64-328A

Directorio de C:\Users\Instituto

02/10/2023  17:52    <DIR>          .
19/09/2023  13:25    <DIR>          ..
02/10/2023  09:10    <DIR>          .android
26/09/2023  12:07           70 .gitconfig
19/09/2023  10:25    <DIR>          .jdk
19/09/2023  14:03    <DIR>          .templateengine
10/10/2023  08:16    <DIR>          .VirtualBox
18/08/2023  16:21    <DIR>          Contacts
11/10/2023  12:12    <DIR>          Desktop
03/10/2023  09:15    <DIR>          Documents
11/10/2023  10:36    <DIR>          Downloads
18/08/2023  16:21    <DIR>          Favorites
26/09/2023  10:02    <DIR>          intellij-chatgpt
18/08/2023  16:21    <DIR>          Links
18/08/2023  16:21    <DIR>          Music
04/10/2023  12:41    <DIR>          OneDrive
18/08/2023  16:21    <DIR>          PCManager
27/09/2023  18:09    <DIR>          Pictures
18/08/2023  16:21    <DIR>          Saved Games
```

Comando “cd” sirve para moverse entre carpetas del sistema con la consola



```

Administrador de símbolos del sistema
C:\Users\Instituto>cd ..

C:\Users>cd C:\Users\Public

C:\Users\Public>
  
```

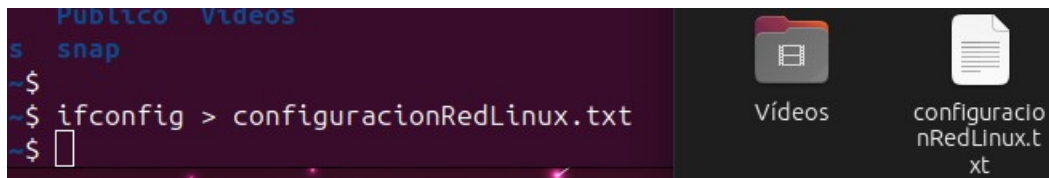
Linux (Operadores de redirección)

Símbolo	Descripción
>	Redirecciona stdout hacia un archivo. Lo crea si no existe, si existe lo sobrescribe. ls -l > lista.txt (La salida del comando se envía a un archivo en vez de la terminal.)
>>	Redirecciona stdout hacia un archivo. Lo crea si no existe, si existe concatena la salida al final de este. ps -ef >> procesos.txt (Concatena al archivo procesos.txt la salida del comando.)
<	Redirecciona stdin desde un archivo. El contenido de un archivo es la entrada o input del comando. mail user < texto.txt (El cuerpo del correo a enviar proviene desde un archivo, en vez del teclado.)
2> 2>>	Redirecciona stderr hacia un archivo. Crea (>) o concatena (>>) la salida de errores a un archivo. (ver ejemplos)
1>&2	Redirecciona stdout hacia donde stderr apunte. (ver ejemplos)
2>&1	Redirecciona stderr hacia donde stdout apunte. (ver ejemplos)
OTROS REDIRECCIONAMIENTOS QUE NO UTILIZAN FDs	
<<	Conocido como HERE-DOCUMENT o HereDoc (ver ejemplos)
<<<	Conocido como HERE-STRING (ver ejemplos)
	El símbolo (pipe) es un tipo de redireccionamiento ya que la salida (stdout) de un comando es la entrada (stdin) de otro. ls /etc grep services (La salida del comando a la izquierda de se convierte en la entrada del comando a la derecha.)
tee	El comando tee redirecciona la salida (stdout) a ambos, un archivo y a la terminal. ps -ef tee procesos.txt (La salida de ps se muestra en la terminal y al mismo tiempo se redirecciona al archivo <i>procesos.txt</i> . Con la opción -a (tee -a) concatena al archivo.)

Estos operadores de redirección tienen una funcionalidad similar a los operadores de redirección de Windows, vamos a explicar algunos de ellos.

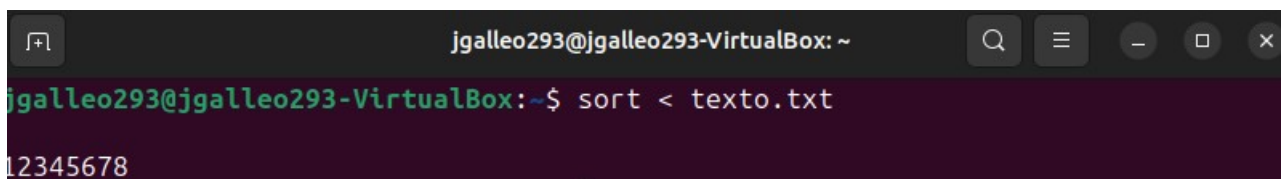
Comenzaremos con un ejemplo similar al realizado en el apartado de Windows, con el comando `ifconfig` mostraremos en la consola la información de la red a la que estamos conectada (Si la terminal no nos permite ejecutar el comando, debemos instalarlo con `sudo apt install net-tools`).

Para guardar esta configuración utilizaremos el operador de redirección `>` y lo podremos guardar en un `.txt`.



```
Publico Videos
s snap
~$
~$ ifconfig > configuracionRedLinux.txt
~$
```

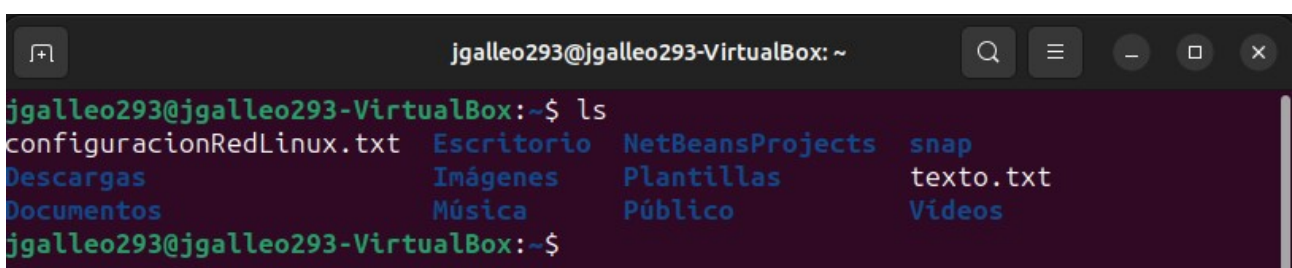
Con el operador de redirección `<` es la entrada de otro comando, por ejemplo, podemos usar el comando `sort` para imprimir por pantalla la entrada de un archivo de texto, y con el comando de redirección `<` usaremos de entrada `texto.txt` que tiene escrito `12345678`.



```
jgalleo293@jgalleo293-VirtualBox: ~
jgalleo293@jgalleo293-VirtualBox:~$ sort < texto.txt
12345678
```

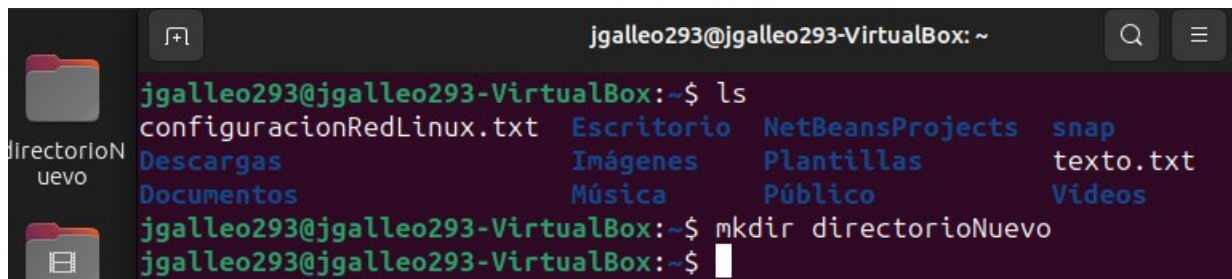
Dentro de los comando de linux, al igual que en Windows podemos trabajar desde la consola, algunos de ellos son los siguientes:

Comando `ls` vere los ficheros de un directorio.



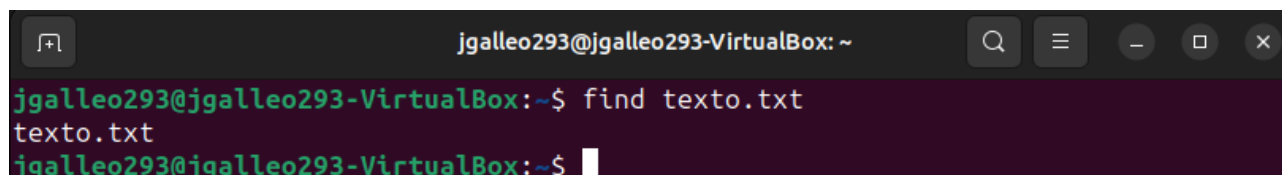
```
jgalleo293@jgalleo293-VirtualBox: ~
jgalleo293@jgalleo293-VirtualBox:~$ ls
configuracionRedLinux.txt  Escritorio  NetBeansProjects  snap
Descargas                 Imágenes   Plantillas        texto.txt
Documentos                Música     Público           Videos
jgalleo293@jgalleo293-VirtualBox:~$
```

Comando “*mkdir*” seguido de un nombre creará una carpeta o directorio con el nombre indicado.

A terminal window titled 'jgalleo293@jgalleo293-VirtualBox: ~' with a search icon and a menu icon in the top right. The terminal shows the following commands and output:

```
jgalleo293@jgalleo293-VirtualBox:~$ ls
configuracionRedLinux.txt  Escritorio  NetBeansProjects  snap
Descargas                  Imágenes   Plantillas        texto.txt
Documentos                 Música     Público           Videos
jgalleo293@jgalleo293-VirtualBox:~$ mkdir directorioNuevo
jgalleo293@jgalleo293-VirtualBox:~$
```

Comando “*find / -name file*” con este comando podemos buscar ficheros y directorios a partir de la raíz del sistema.

A terminal window titled 'jgalleo293@jgalleo293-VirtualBox: ~' with search, menu, and window control icons in the top right. The terminal shows the following commands and output:

```
jgalleo293@jgalleo293-VirtualBox:~$ find texto.txt
texto.txt
jgalleo293@jgalleo293-VirtualBox:~$
```

Ej3. - Arrancando y parando procesos en java (1,5 ptos)

Java, el lenguaje que estamos trabajando, es uno de los más útiles y potentes. Este mismo lo podremos utilizar para crear programas que controlen nuestro equipo, como por ejemplo que abra programas con una serie de comandos, esto lo haremos en Windows y Linux.

Windows

Crearemos un programa Java en nuestro entorno de desarrollo habitual (NetBeans, Eclipse, IntelliJ...), creamos un proyecto/paquete nuevo que se llame “*Ej3*” y escribiremos un código como el que se ve en la imagen.

```

Ej3.java x
2      import java.time.Duration;
3      import java.util.Optional;
4
5      public class Ej3 {
6      public static void main(String[] args) {
7      try {
8          // Iniciar los procesos
9          Process proceso1 = Runtime.getRuntime().exec("notepad.exe");
10         Process proceso2 = Runtime.getRuntime().exec("calc.exe");
11         Process proceso3 = Runtime.getRuntime().exec("mspaint.exe");
12
13         // Mostrar los PID de los procesos
14         System.out.println("PID proceso 1: " + proceso1.toHandle().pid());
15         System.out.println("PID proceso 2: " + proceso2.toHandle().pid());
16         System.out.println("PID proceso 3: " + proceso3.toHandle().pid());
17
18         // Esperar a que los procesos finalicen
19         proceso1.waitFor();
20         proceso2.waitFor();
21         proceso3.waitFor();
22
23         // Calcular y mostrar el tiempo transcurrido para cada proceso
24         Optional<Duration> duracionProceso1 = proceso1.toHandle().info().totalCpuDuration();
25         Optional<Duration> duracionProceso2 = proceso2.toHandle().info().totalCpuDuration();
26         Optional<Duration> duracionProceso3 = proceso3.toHandle().info().totalCpuDuration();
27
28         System.out.println("Tiempo transcurrido proceso 1: " + duracionProceso1.orElse(Duration.ZERO));
29         System.out.println("Tiempo transcurrido proceso 2: " + duracionProceso2.orElse(Duration.ZERO));
30         System.out.println("Tiempo transcurrido proceso 3: " + duracionProceso3.orElse(Duration.ZERO));

```

Su estructura es sencilla, utilizamos “*Process*” para crear un proceso y le indicamos que proceso queremos que cree e inicialice. Después escribimos el nombre del proceso junto a “*.waitFor()*” para que espere a que se ejecuten el resto de programas y no se ejecuten todos a la vez. Este programa ejecuta bloc de notas, calculadora y paint. Al mismo tiempo, se muestra por consola el PID de cada programa ejecutado y el tiempo que han tardado en ejecutarse, si lo ejecutamos nos mostrará los siguientes datos:

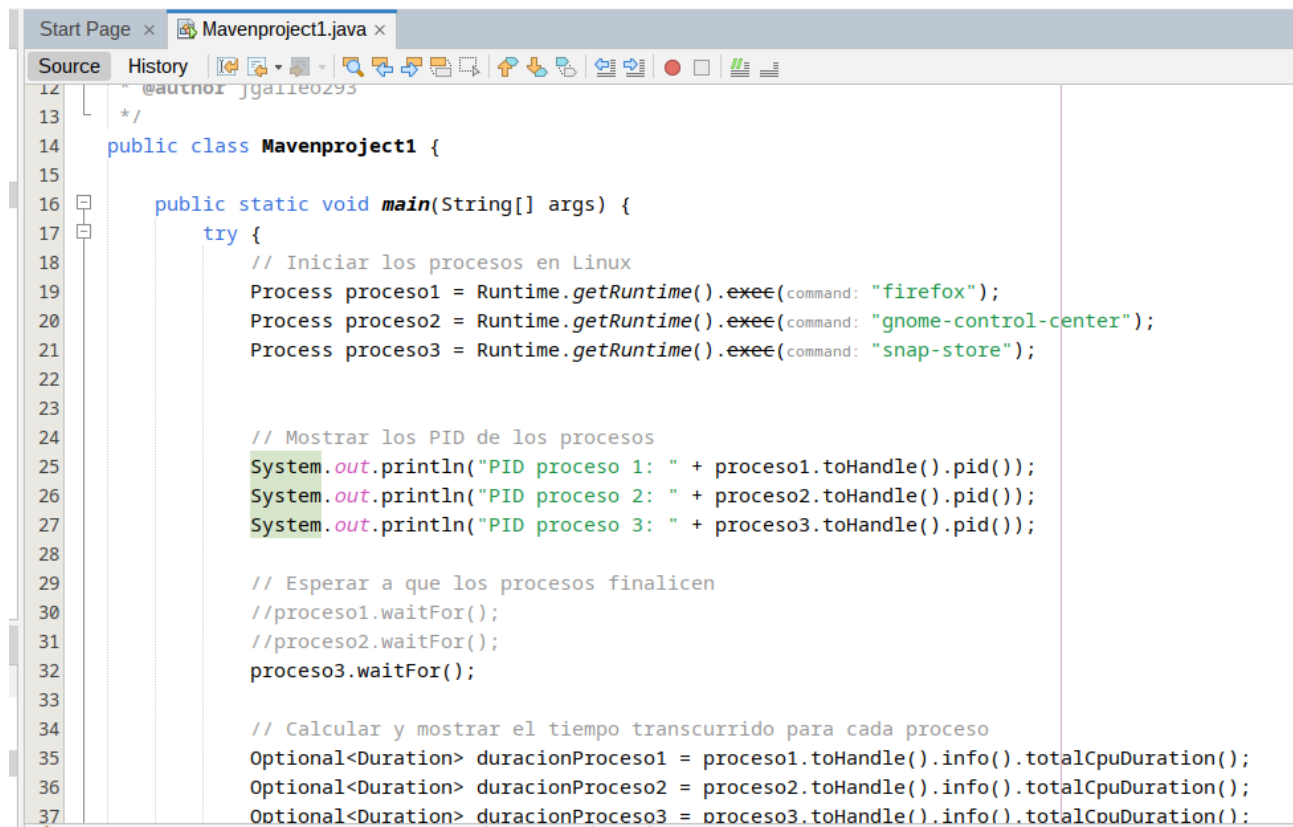
```

PID proceso 1: 2484
PID proceso 2: 6612
PID proceso 3: 8928
Tiempo transcurrido proceso 1: 31 ms
Tiempo transcurrido proceso 2: 62 ms
Tiempo transcurrido proceso 3: 359 ms

```

Linux

Realizaremos el mismo procedimiento en Linux, crearemos un proyecto de Java en nuestro entorno habitual, creamos un proyecto/paquete nuevo que se llame “Ej3” y escribiremos un código como el que se ve en la imagen.



```
12  @author JGalle0293
13  */
14  public class Mavenproject1 {
15
16      public static void main(String[] args) {
17          try {
18              // Iniciar los procesos en Linux
19              Process proceso1 = Runtime.getRuntime().exec(command: "firefox");
20              Process proceso2 = Runtime.getRuntime().exec(command: "gnome-control-center");
21              Process proceso3 = Runtime.getRuntime().exec(command: "snap-store");
22
23
24              // Mostrar los PID de los procesos
25              System.out.println("PID proceso 1: " + proceso1.toHandle().pid());
26              System.out.println("PID proceso 2: " + proceso2.toHandle().pid());
27              System.out.println("PID proceso 3: " + proceso3.toHandle().pid());
28
29              // Esperar a que los procesos finalicen
30              //proceso1.waitFor();
31              //proceso2.waitFor();
32              proceso3.waitFor();
33
34              // Calcular y mostrar el tiempo transcurrido para cada proceso
35              Optional<Duration> duracionProceso1 = proceso1.toHandle().info().totalCpuDuration();
36              Optional<Duration> duracionProceso2 = proceso2.toHandle().info().totalCpuDuration();
37              Optional<Duration> duracionProceso3 = proceso3.toHandle().info().totalCpuDuration();
```

Una vez terminado el código y probado, lo compilaremos en “*jar*” para que se pueda ejecutar en un solo click, este programa abre en linux la configuración, la tienda y el navegador firefox.

```

PID proceso 1: 8505
PID proceso 2: 8514
PID proceso 3: 8522
Tiempo transcurrido proceso 1: 5620 ms
Tiempo transcurrido proceso 2: 5640 ms
Tiempo transcurrido proceso 3: 0 ms
```

Esos son los resultados de la ejecución de los programas que hemos ejecutado en Linux.

Ej. 4 Windows. - Programación Java – Comunicación con comandos S.O – 1 pto

Vamos a realizar algo similar al ejercicio anterior (Ejercicio 3), pero esta vez vamos a visualizar los servicios que están corriendo o se están ejecutando en nuestro sistema operativo.

Para ello deberemos crear una aplicación java .jar, la cual podremos ejecutar, para hacer esto, desde nuestro entorno habitual, crearemos un nuevo proyecto y el código será el siguiente:

```
public class Main {
    public static void main(String[] args) {
        try {
            // Ejecuta el comando "tasklist" y obtiene el resultado
            Process process = Runtime.getRuntime().exec("command: tasklist");
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));

            String line;
            while ((line = reader.readLine()) != null) {
                // Filtra y muestra solo los servicios
                if (line.toLowerCase().contains(".exe") || line.toLowerCase().contains("svchost.exe")) {
                    System.out.println(line);
                }
            }

            // Cierra el lector
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Nos mostrará por consola los servicios que se están ejecutando y algo de información, desde la línea donde declaramos el proceso que queremos que realice el programa, le diremos los que queremos que se ejecute y después leeremos esa información y la mostraremos por la pantalla.

wininit.exe	1048	Services	0	3.512 KB
services.exe	1124	Services	0	6.816 KB
LsaIso.exe	1152	Services	0	1.984 KB
lsass.exe	1200	Services	0	19.624 KB
svchost.exe	1356	Services	0	29.332 KB

Ej. 4 Linux. - Programación Java – Comunicación con comandos S.O – 1 pto

Este ejercicio es similar al ejercicio 4 realizado en el SO Windows, ahora lo trasladaremos a Linux, por lo que tendremos que realizar algunos cambios en el código. Entramos en el entorno creamos un proyecto y escribiremos el código de la imagen:

```
public class Ej4 {
    public static void main(String[] args) {
        try {
            // Ejecuta el comando "ls -al /tmp" y obtiene el resultado
            Process process = Runtime.getRuntime().exec(command: "ls -al /tmp");
            BufferedReader reader = new BufferedReader(new InputStreamReader(in: process.getInputStream()));

            String line;
            while ((line = reader.readLine()) != null) {
                // Filtra y muestra solo los archivos que pertenecen al usuario actual
                if (line.contains(" " + System.getProperty(key: "user.name") + " ")) {
                    System.out.println(x: line);
                }
            }

            // Cierra el lector
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Es un proceso similar a lo realizado en Windows, como resultado tras ejecutarlo nos mostraría lo siguiente:

```
--- exec:3.1.0:exec (default-cli) @ mavenproject1 ---
drwxr-xr-x  2 jgalleo293 jgalleo293  4096 oct 24 17:03 hsperrdata_jgalleo293
-rwxrw-r--  1 jgalleo293 jgalleo293 15800 oct 24 17:03 jansi-2.4.0-2959d0b666ccf685-libjansi.so
-rw-rw-r--  1 jgalleo293 jgalleo293    0 oct 24 17:03 jansi-2.4.0-2959d0b666ccf685-libjansi.so.lck
drwx----- 2 jgalleo293 jgalleo293  4096 oct 24 16:44 .org.chromium.Chromium.7L3PMv
drwx----- 2 jgalleo293 jgalleo293  4096 oct 24 16:44 .org.chromium.Chromium.ECRMPZ
```

Aquí vemos los diferentes servicios que se están ejecutando en el sistema Linux.

Ej. 5. - Java – Comunicación entre procesos – 2 pts

Vamos a realizar una serie de comunicaciones entre los diferentes procesos del sistema, apoyándonos en los programas auxiliares que hemos creado anteriormente.

```
import java.util.Scanner;

public class Columna {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String separator = "[ \\t]+"; // Utilizamos una expresión regular para separar por uno o más espacios o tabuladores

        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            String[] columns = line.split(regex: separator);

            if (args.length > 0 && Integer.parseInt(args[0]) <= columns.length) {
                int selectedColumn = Integer.parseInt(args[0]) - 1;
                System.out.println(columns[selectedColumn]);
            } else {
                System.out.println();
            }
        }

        scanner.close();
    }
}
```

```
import java.util.Scanner;

public class Columna {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String separator = "[ \\t]+"; // Utilizamos una expresión regular para separar por uno o más espacios o tabuladores

        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            String[] columns = line.split(regex: separator);

            if (args.length > 0 && Integer.parseInt(args[0]) <= columns.length) {
                int selectedColumn = Integer.parseInt(args[0]) - 1;
                System.out.println(columns[selectedColumn]);
            } else {
                System.out.println();
            }
        }

        scanner.close();
    }
}
```

```
import java.util.Scanner;

public class NumeroMayor {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int max = Integer.MIN_VALUE;

        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();

            if (line.isEmpty() || !line.matches(regex: "\\d+")) {
                continue;
            }

            int number = Integer.parseInt(s: line);
            if (number > max) {
                max = number;
            }
        }

        System.out.println(x: max);

        scanner.close();
    }
}
```

Estos programas auxiliares los podremos utilizar de forma complementaria con las diferentes ejecuciones que realicemos en la cmd o terminal.

Windows

Tendríamos que indicarle al programa que abra la consola de comandos y en ella escriba la dirección del directorio que queremos analizar, en este caso nos pide que analicemos el directorio windows.

```
public static void main(String[] args) throws IOException {
    try {
        Process process = Runtime.getRuntime().exec(command: "cmd /c dir");
        BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String linea;

        int sumaTamaños = 0;
        int mayorTamaño = 0;

        while ((linea = reader.readLine()) != null) {
            // Ver los datos de la columna que queremos
            String[] columns = linea.split(regex: "\\s+");
            if (columns.length >= 3) {
                try {
                    int size = Integer.parseInt(columns[2]);
                    sumaTamaños += size;
                    if (size > mayorTamaño) {
                        mayorTamaño = size;
                    }
                } catch (NumberFormatException e) {}
            }
        }

        reader.close();
    }
}
```

Este programa lo que hace es lo mencionado anteriormente, analizar el directorio que le hemos indicado y nos analiza el tamaño de los ficheros y los archivos.

Linux

Utilizaremos el mismo programa en linux, simplemente cambiaremos algunas líneas del código como, donde se indica que se tiene que abrir cmd, que abriremos la terminal de linux y la línea donde indicamos el directorio, que la cambiaremos por el comando oportuno en linux.

```
public class Ej5L {
    public static void main(String[] args) throws IOException {
        try {
            Process process = Runtime.getRuntime().exec( command: "ls -al");
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            String linea;

            int sumaTamaños = 0;
            int mayorTamaño = 0;

            while ((linea = reader.readLine()) != null) {
                // Ver los datos de la columna que queremos
                String[] columns = linea.split( regex: "\\s+");
                if (columns.length >= 3) {
                    try {
                        int size = Integer.parseInt(columns[2]);
                        sumaTamaños += size;
                        if (size > mayorTamaño) {
                            mayorTamaño = size;
                        }
                    } catch (NumberFormatException e) { }
                }
            }
        }
    }
}
```

Ej. 7. Programa libre – 0,75 pto

Finalmente, con los conocimientos que hemos aprendido, crearemos un proceso desde cero. En este proceso repasaremos donde explicaremos como se arrancan, comunican y paran procesos desde una aplicación java.

Será un programa sencillo en el que iniciaremos un proceso del sistema, leeremos si se ha ejecutado correctamente e imprimiremos por pantalla si se ha ejecutado correctamente. Y cerraremos el proceso.

Será el mismo programa para linux y windows, cambiando las líneas de código para que concuerden con el sistema.

Windows

```
public class Ej7W {  
    public static void main(String[] args) {  
        // Arrancar un proceso  
        Process proceso = null;  
        try {  
            proceso = Runtime.getRuntime().exec("mspaint.exe");  
            System.out.println("Proceso arrancado.");  
        } catch (IOException e) {  
            System.out.println("Error al arrancar el proceso: " + e.getMessage());  
        }  
  
        // Comunicarse con el proceso  
        if (proceso != null) {  
            // Leer la salida del proceso  
            try {  
                java.io.InputStream inputStream = proceso.getInputStream();  
                java.io.BufferedReader reader = new java.io.BufferedReader(new java.io.InputStreamReader(inputStream));  
                String linea;  
                while ((linea = reader.readLine()) != null) {  
                    System.out.println("Salida del proceso: " + linea);  
                }  
            } catch (IOException e) {  
                System.out.println("Error al leer la salida del proceso: " + e.getMessage());  
            }  
        }  
  
        // Detener el proceso  
        proceso.destroy();  
        System.out.println("Proceso detenido.");  
    }  
}
```

Linux

```
public class Ej7L {  
    public static void main(String[] args) {  
        // Arrancar un proceso  
        Process proceso = null;  
        try {  
            proceso = Runtime.getRuntime().exec("gnome-control-center");  
            System.out.println("Proceso arrancado.");  
        } catch (IOException e) {  
            System.out.println("Error al arrancar el proceso: " + e.getMessage());  
        }  
  
        // Comunicarse con el proceso  
        if (proceso != null) {  
            // Leer la salida del proceso  
            try {  
                java.io.InputStream inputStream = proceso.getInputStream();  
                java.io.BufferedReader reader = new java.io.BufferedReader(new java.io.InputStreamReader(inputStream));  
                String linea;  
                while ((linea = reader.readLine()) != null) {  
                    System.out.println("Salida del proceso: " + linea);  
                }  
            } catch (IOException e) {  
                System.out.println("Error al leer la salida del proceso: " + e.getMessage());  
            }  
  
            // Detener el proceso  
            proceso.destroy();  
            System.out.println("Proceso detenido.");  
        }  
    }  
}
```

Webgrafía

-Stackoverflow. “A Java exception has occurred” opening .jar solution:
<https://stackoverflow.com/questions/25162311/a-java-exception-has-occurred-when-opening-jar>

-Javahispano. “Ejecutar una aplicación con varios .jar”: <http://javahispano.squarespace.com/java-se/post/2086926;jsessionid=4E64C6846BA44927631073CB7D50D2BB.v5-web005>