CIFP JUAN DE COLONIA

# Programación de Servicios y Procesos

## UT1: Programación multiproceso

**Alexis López Briongos Dam2t**

**05/10/2023**

# ***Índice***

# 1

## a)

```
alexis@alexis-if05:~$ ps
    PID TTY          TIME CMD
   1913 pts/0    00:00:00 bash
   2508 pts/0    00:00:00 ps
```

- Muestra los procesos del sistema

```
alexis@alexis-if05:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b───2*[{gnome-session-b}]
        │      │                 │                 └─2*[{gdm-wayland-ses}]
        │      │                 └─2*[{gdm-session-wor}]
        │      └─2*[{gdm3}]
        ├─gnome-keyring-d───3*[{gnome-keyring-d}]
        ├─2*[kerneloops]
        ├─networkd-dispat
        ├─packagekitd───2*[{packagekitd}]
        ├─polkitd───2*[{polkitd}]
        ├─power-profiles-───2*[{power-profiles-}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─stkit-daemon───2*[{stkit-daemon}]
```

- Muestra los procesos con una representación jerárquica

## b)

```
  PID USUARIO   PR  NI    VIRT    RES    SHR S  %CPU  %MEM     HORA+ ORDEN
 2907 alexis    20   0 3262336 447808 145212 S   6,1  22,3   0:17.39 firefox
```

```
alexis@alexis-if05:~$ ps 2907
    PID TTY      STAT   TIME COMMAND
   2907 ?        Sl     0:19 /snap/firefox/2987/usr/lib/firefox/firefox
```

```
alexis@alexis-if05:~$ kill 2907
```

- Envíamos una señal al proceso deseado (por PID).

```
alexis@alexis-if05:~$ kill 2907
alexis@alexis-if05:~$ ps 2907
    PID TTY      STAT   TIME COMMAND
alexis@alexis-if05:~$
```

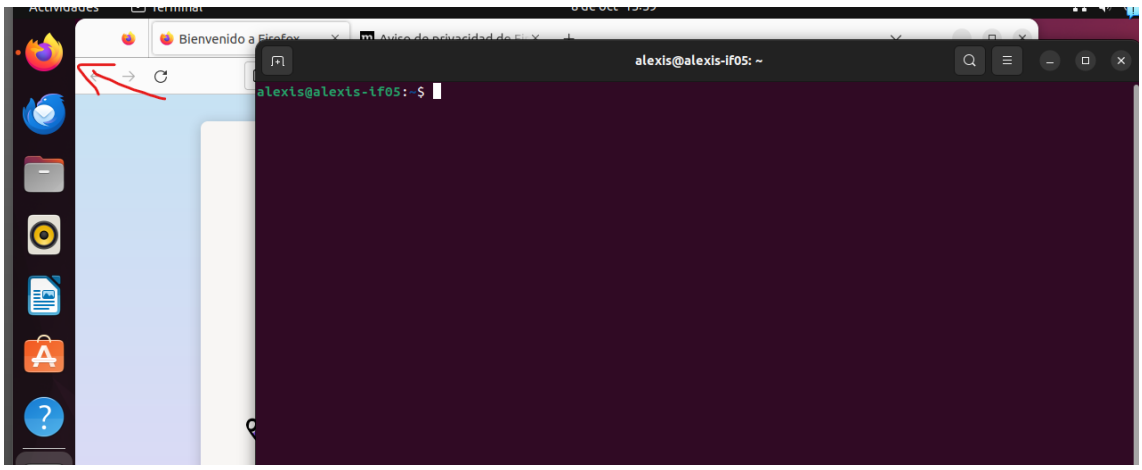
- Proceso eliminado

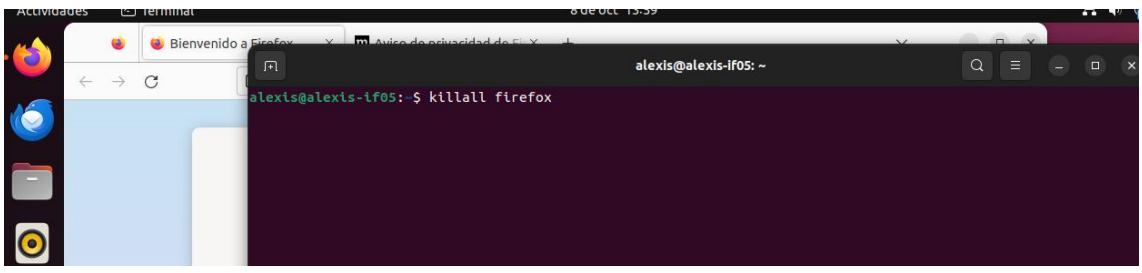
## c)

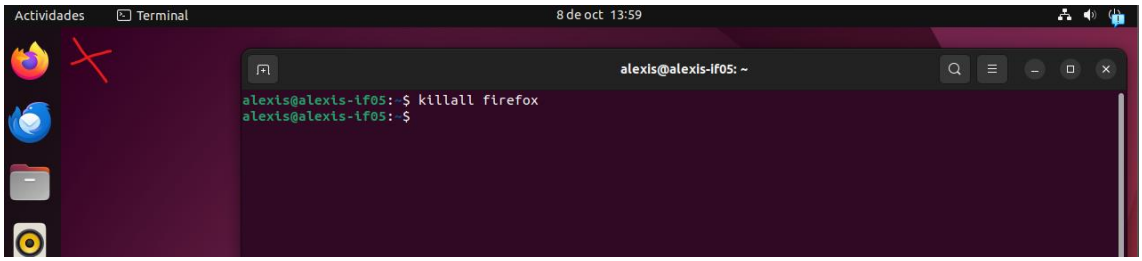
```
alexis@alexis-if05:~$ killall
Modo de empleo: killall [OPCIÓN]... [--] NOMBRE...
      killall -l, --list
      killall -V, --version

  -e,--exact          requiere coincidencia exacta para nombres muy largos
  -I,--ignore-case    MAYÚS/minux indistinguibles para coincidencia de nombre
                      del proceso
  -g,--process-group  mata grupo de procesos de vez de proceso
  -y,--younger-than   mata procesos más recientes que HORA
  -o,--older-than     mata procesos más antiguos que HORA
  -i,--interactive    pide confirmación antes de matar
  -l,--list           lista todos los nombres de señales conocidas
  -q,--quiet          no escribe quejas
  -r,--regexp         interpreta NOMBRE como una expreg extendida
  -s,--signal SEÑAL   envía esta señal en vez de SIGTERM
  -u,--user USUARIO   mata solo proceso(s) ejecutándose como USUARIO
  -v,--verbose        informa si la señal se ha enviado correctamente
  -V,--version        muestra información sobre la versión
  -w,--wait           espera a que los procesos mueran
  -n,--ns PID         coincidencia con procesos que pertenecen al mismo
                      espacio de nombres que PID
  -Z,--context EXPREG mata solo proceso(s) que tienen contexto
                      (debe preceder a otros argumentos)
```

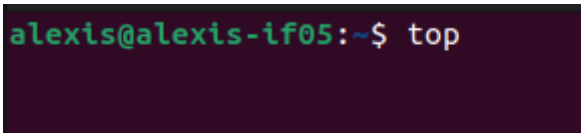
- Enviamos una señal a un proceso deseado (por nombre).

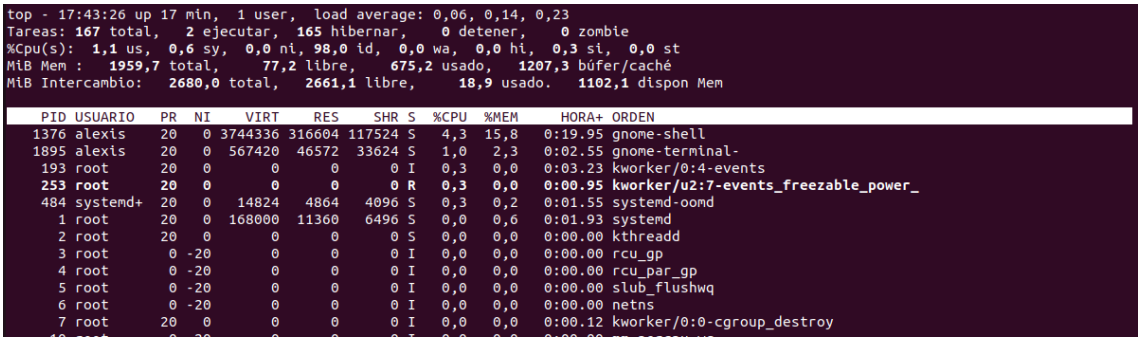


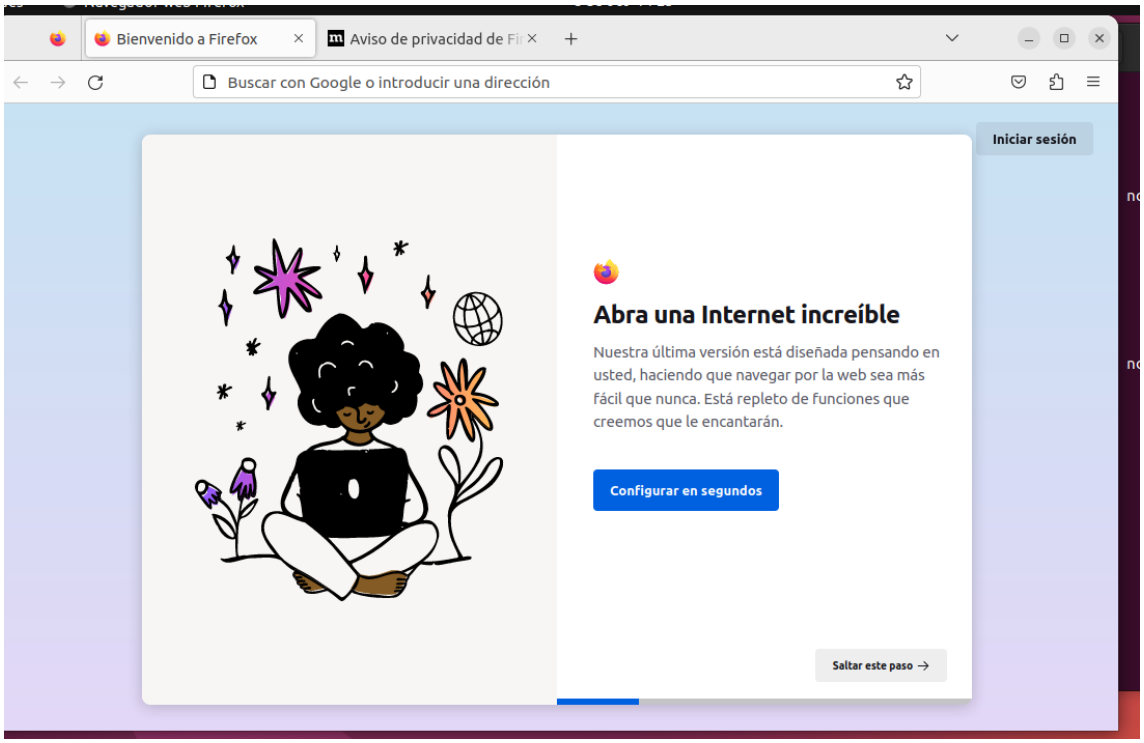



## d)




## e)




- Abrimos Firefox de nuevo para establecerle como proceso prioritario.

```
alexis@alexis-if05:~$ pgrep -o firefox
5721
```

- Utilizamos este comando para conocer el PID del proceso Firefox.

```
alexis@alexis-if05:~$ sudo renice -n -10 -p 5721
5721 (process ID) prioridad anterior 0, nueva prioridad -10
```

- Ahora hemos establecido el proceso de Firefox con mayor prioridad sobre otros procesos.

## f)

```
alexis@alexis-if05:~$ time firefox
Gtk-Message: 15:35:31.777: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try to not load i

real    0m0,459s
user    0m0,188s
sys     0m0,089s
```

- Ejecutamos el comando time junto a Firefox para saber el tiempo que tarda de ejecución del proceso.
- Aquí está el significado de cada una de las líneas:
    - **real**: Este es el tiempo real total transcurrido desde que se inició el comando hasta que se completó. Representa el tiempo en el mundo real, incluyendo cualquier tiempo de espera o bloqueo del proceso.
    - **user**: Este es el tiempo de CPU que se utilizó en el espacio de usuario (tiempo de CPU utilizado por el proceso en sí).
    - **sys**: Este es el tiempo de CPU utilizado en el espacio del kernel (tiempo de CPU utilizado por el sistema operativo para ejecutar el proceso).

## g) y h)

```
alexis@alexis-if05:~$ firefox &
[1] 6777
```

- Ejecutamos este comando para que Firefox se abra en segundo plano asignándole una posición.

```
alexis@alexis-if05:~$ jobs
[1]+  Ejecutando              firefox &
```

- Con el comando Jobs podemos visualizar todos los procesos que hemos abierto o están en segundo plano.

```
alexis@alexis-if05:~$ fg %1
firefox
```

- Con el comando fg pasamos el proceso a primer plano.

```
alexis@alexis-if05:~$ bg %1
[1]+ firefox &
```

- Con el comando bg pasamos el proceso a segundo plano.

# 2

## a)

```
alexis@alexis-if05:~$ nano alexis_script.sh
```

```
  GNU nano 6.2                          alexis_script.sh *
while true
do
echo "Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle."
sleep 1
done
```

```
alexis@alexis-if05:~$ chmod +x alexis_script.sh
```

- Le cambio los permisos al script para poderlo ejecutar.

```
                                                    alexis@alexis-i
alexis@alexis-if05:~$ sudo ./alexis_script.sh
```

```
alexis@alexis-if05:~$ sudo ./alexis_script.sh
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
```

## b)

```
alexis@alexis-if05:~$ ./alexis_script.sh
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
^Z
[1]+  Detenido                ./alexis_script.sh
alexis@alexis-if05:~$ ./alexis_script.sh
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
^Z
[2]+  Detenido                ./alexis_script.sh
alexis@alexis-if05:~$ ./alexis_script.sh
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
^Z
[3]+  Detenido                ./alexis_script.sh
alexis@alexis-if05:~$ jobs
[1]   Detenido                ./alexis_script.sh
[2]-  Detenido                ./alexis_script.sh
[3]+  Detenido                ./alexis_script.sh
```

- Ejecutamos el script y lo detenemos con la combinación de teclas CTRL+Z, así tres veces.

## c)

```
alexis@alexis-if05:~$ fg %1
./alexis_script.sh
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
Esto es un bucle infinito. Pulse CTRl-C para finalizar el bucle.
^Calexis@alexis-if05:~$ jobs
[2]-  Detenido                ./alexis_script.sh
[3]+  Detenido                ./alexis_script.sh
```

- Con el comando fg %[número del proceso en Jobs] reanudamos el proceso y lo finalizamos con la combinación de teclas CTRL+Z

## d)

```
alexis@alexis-if05:~$ jobs
[2]-  Detenido                ./alexis_script.sh
[3]+  Detenido                ./alexis_script.sh
alexis@alexis-if05:~$ kill %3

[3]+  Detenido                ./alexis_script.sh
alexis@alexis-if05:~$ jobs
[2]-  Detenido                ./alexis_script.sh
[3]+  Terminado               ./alexis_script.sh
```

- Con el comando Jobs comprobamos los procesos que tenemos activos en segundo plano. Con el comando kill %3 finalizamos el comando con el índice 3. Volvemos a ejecutar el comando Jobs para verificar que el proceso ha terminado.