

FP.057 - (P) Seguridad, procesos y zócalos

AA1. Gestión de procesos

Alumno: JAVIER DE LA VEGA EDER

Consultor: Lean Zabala Iglesias
CFGS en Desarrollo de Aplicaciones Multiplataforma(ICB02)
FP UOC - Fundació Jesuïtes Educació

Índice

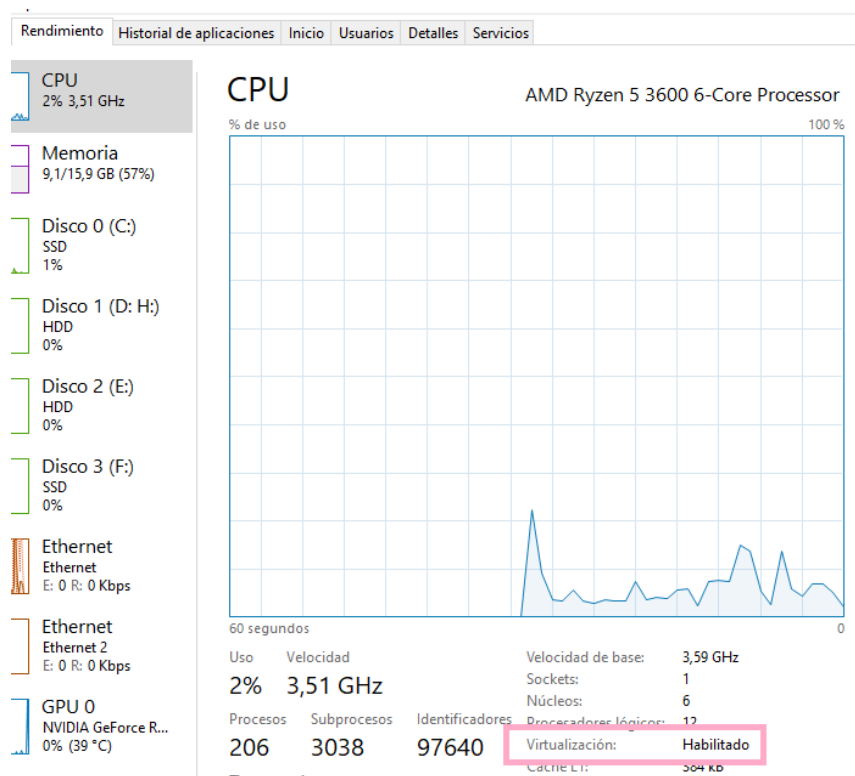
Instalación de la Máquina Virtual	3
Instalación y personalización del SO Debian en VM	5
Instalación de Guest-Additions	7
Instalación Visual Studio Code en Debian	9
Monitorizar los procesos del Sistema Debian (Shell)	10
Identificación de Procesos (PID) en el Sistema Debian	11
¿Cuáles serían las opciones necesarias para mostrar los procesos de un usuario concreto en vista detallada?	12
Comando Kill	12
Ejecutar el programa “python ACT1_EJ1_v2.py”	13
Mensaje de error; corregimos el script de Python	15
Webgrafía	16

Instalación de la Máquina Virtual

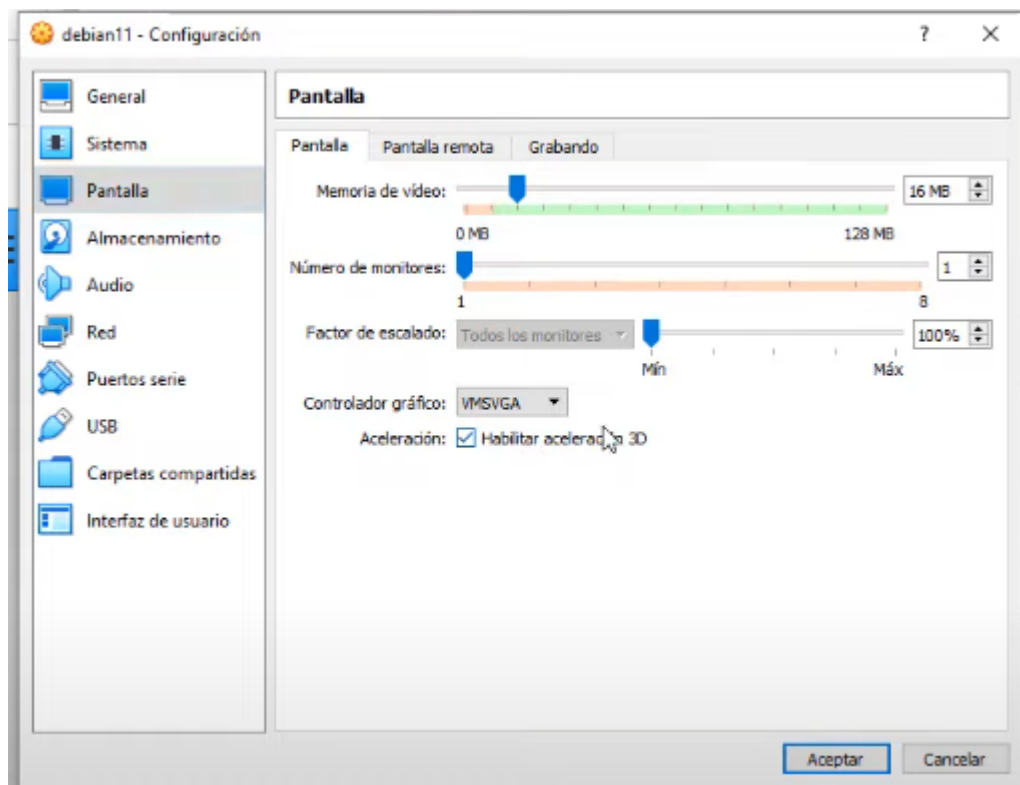
Procedemos a Instalar [Virtual Box](#) y configurar la máquina virtual donde instalaremos Debian 12.



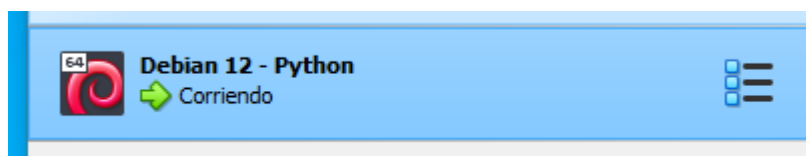
Es importante tener la virtualización habilitada en Windows para que la máquina funcione por lo que, para revisarlo hacemos **Click derecho** en la **barra de tareas** > **Administrador de tareas** < **Pestaña: Rendimiento**. Aquí buscamos la opción **Virtualización** y debe estar **Habilitado**.




Instalamos el SO en la máquina Virtual a través de la opción “**Nueva**” y la configuramos para después instalar la ISO que descargamos de la página de [Debian](https://www.debian.org/).

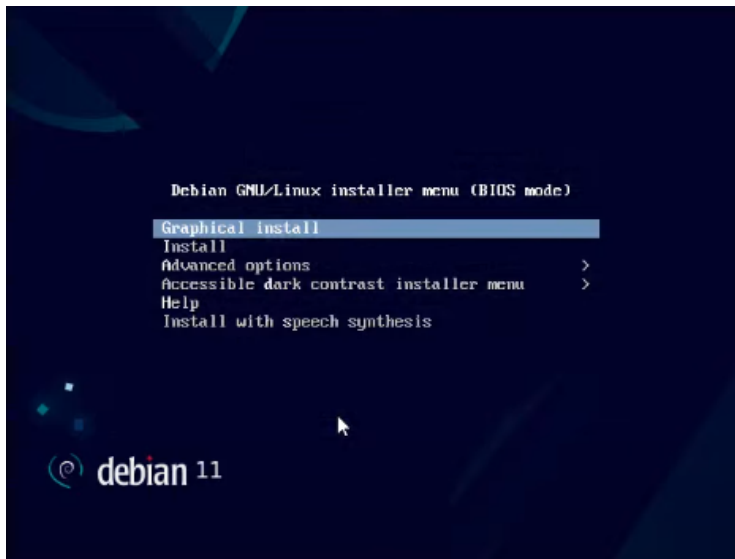


Es recomendable que en la opción de Pantalla, ya que después lo configuraremos para modo transparente, **Habilitar aceleración 3D** y darle la memoria máxima de vídeo. Con esto conseguiremos una mejora notable en el rendimiento de nuestro SO Debian.



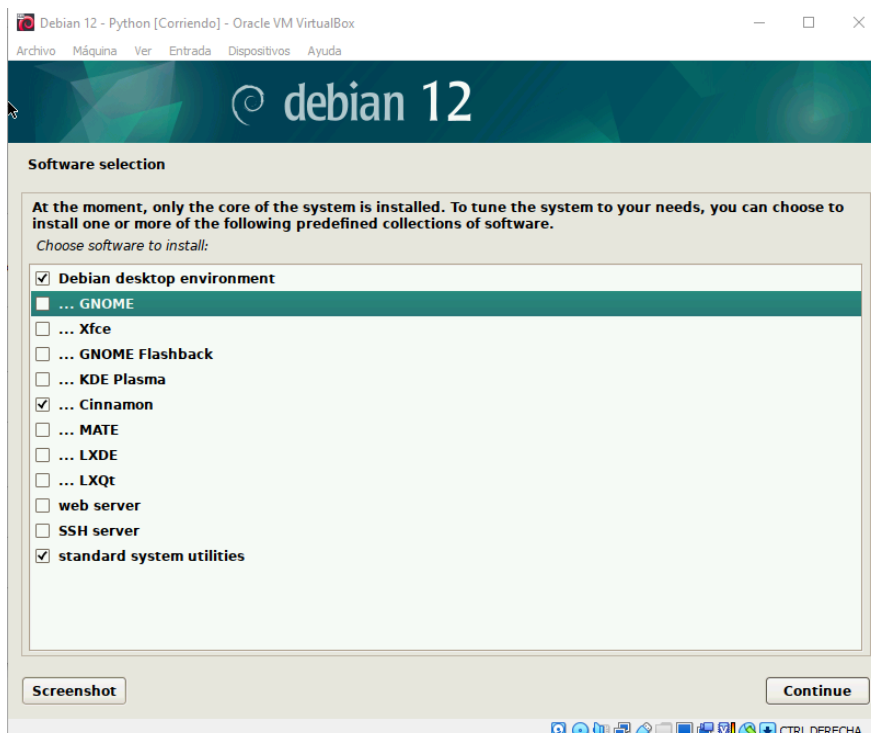
Instalación y personalización del SO Debian en VM

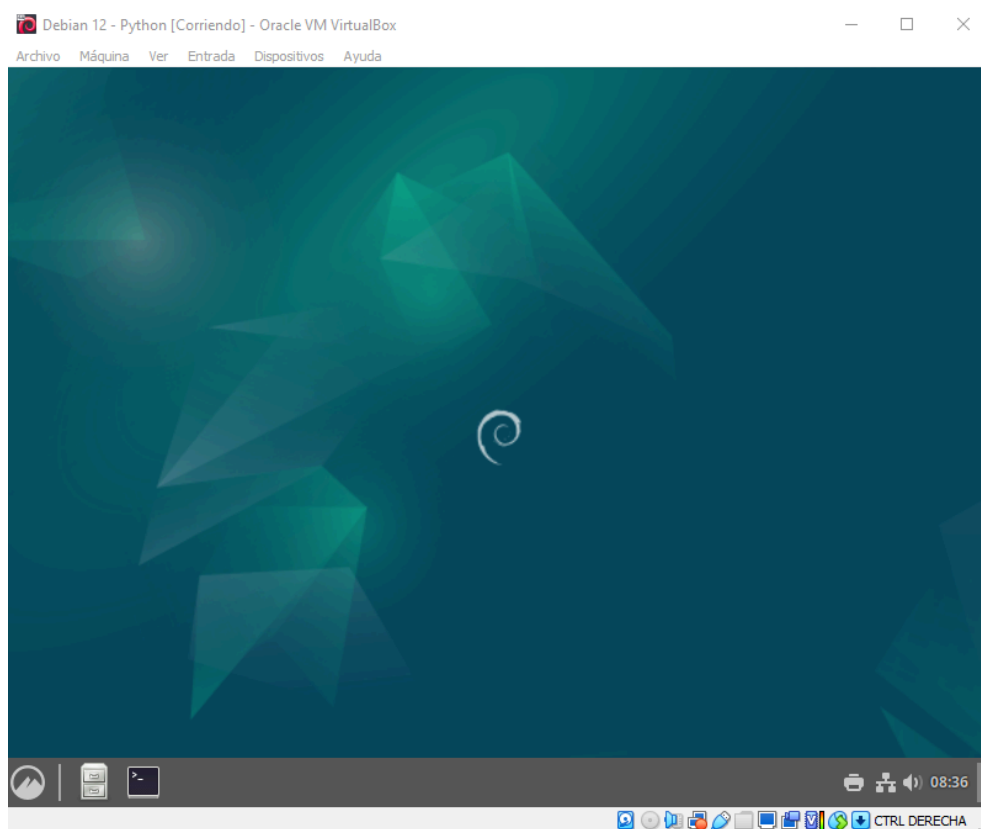
Ejecutamos y realizamos la instalación de Debian.  `debian-12.2.0-amd64-netinst`



Seguimos el proceso de instalación escogiendo ubicación, nombre de usuario y contraseñas tanto para el Super Usuario, como el de acceso a la máquina. Utilizaremos todo el Disco Duro previamente asignado para esta instalación.

Durante la instalación, llegados al siguiente punto debemos elegir un entorno que más se acomode a nuestras necesidades y se ajuste a nuestras preferencias. **En mi caso he decidido instalar Cinnamon** porque utiliza elementos clásicos con los cuales estoy más familiarizado y además ofrece mayor capacidad de personalización frente a Gnome. Gnome es un entorno más sencillo y simplificado, además de optimización de recursos para equipos limitados.





Con Debian ya instalado, abrimos Terminal:

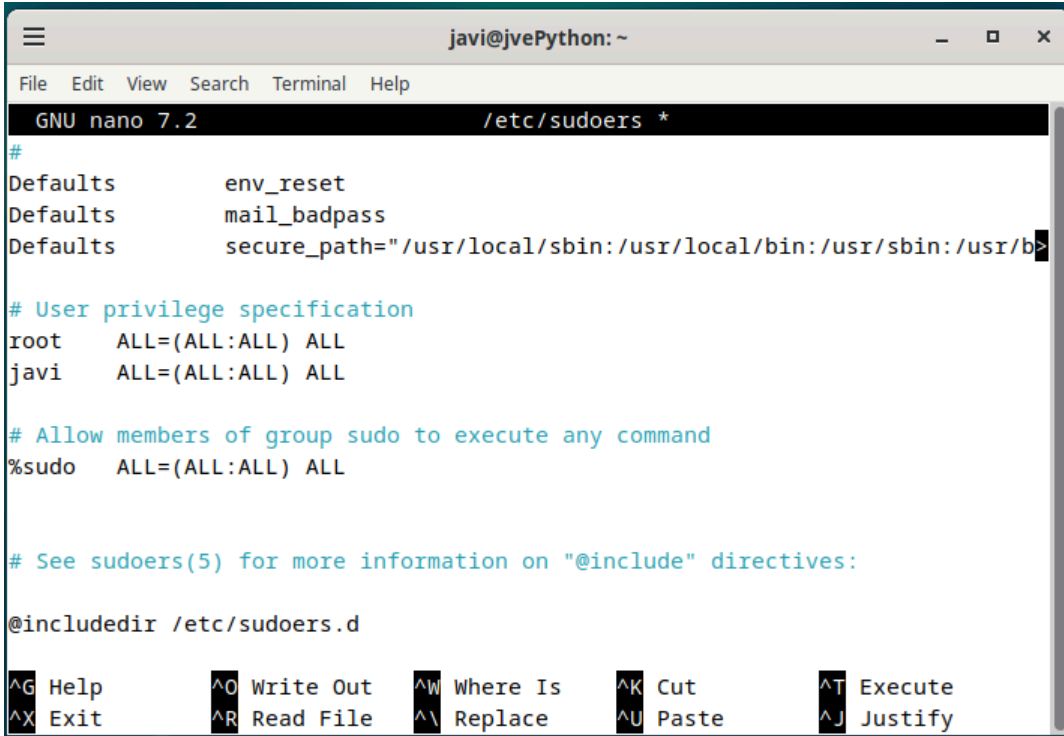
Escribimos “su” y seguida la contraseña de acceso como Superusuario.

```
javi@jvePython: ~
File Edit View Search Terminal Help
javi@jvePython:~$ su
Password:
root@jvePython: /home/javi#
```

Escribimos la siguiente línea:

```
root@jvePython: /home/javi# nano /etc/sudoers
```

Rellenamos info para tener los permisos con nuestro propio usuario en el fichero abierto con nano y, una vez lo tengamos como en la siguiente pantalla, pulsamos CTRL+O y ENTER; y CTRL+X para salir.



```

javi@jvePython: ~
File Edit View Search Terminal Help
GNU nano 7.2 /etc/sudoers *
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b>
# User privilege specification
root    ALL=(ALL:ALL) ALL
javi    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify

```

Hacemos update y upgrade para actualizar toda la información de Debian.

```

javi@jvePython:~$ sudo apt-get update
[sudo] password for javi:
Hit:1 http://security.debian.org/debian-security bookworm-security InRelease
Hit:2 http://deb.debian.org/debian bookworm InRelease
Get:3 http://deb.debian.org/debian bookworm-updates InRelease [52.1 kB]
Fetched 52.1 kB in 0s (173 kB/s)
Reading package lists... Done
javi@jvePython:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

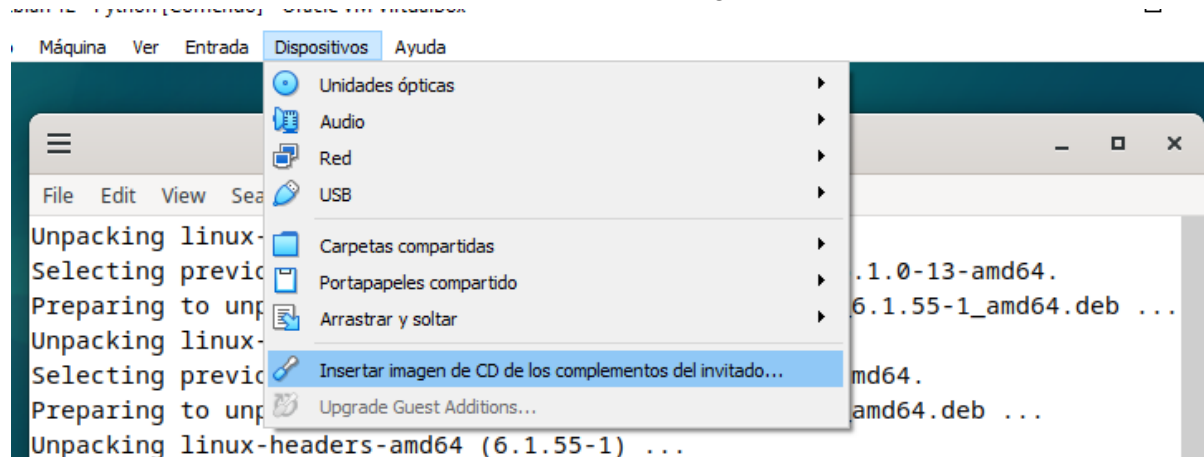
```

Instalación de Guest-Additions

Primero instalamos el conjunto de herramientas y paquetes necesarios para compilar e instalar controladores del kernel y software adicional.

sudo apt install dkms linux-headers-\$(uname -r) build-essential

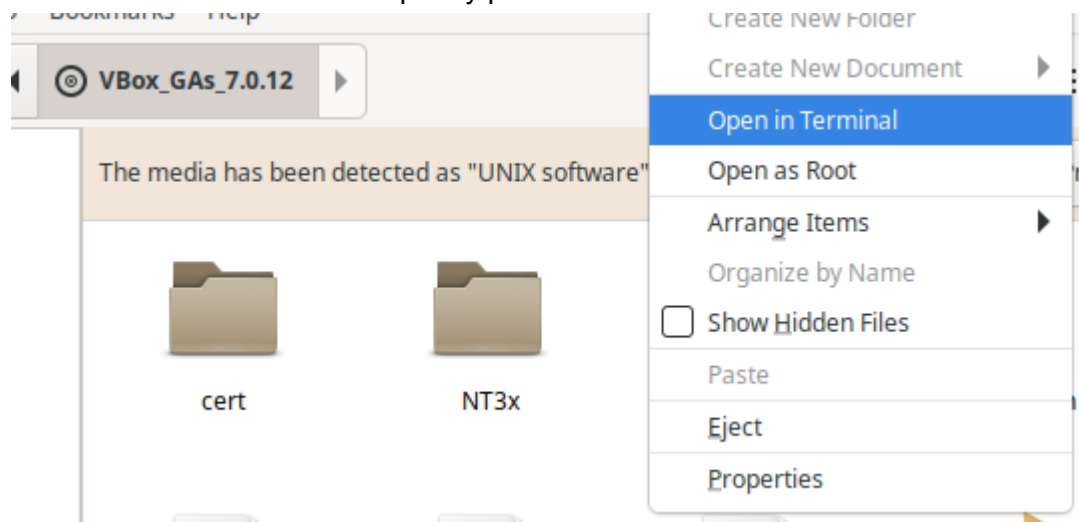
Una vez instalado vamos a Dispositivos > Insertar imagen de CD(...)



Después vamos al Gestor de Archivos y en dispositivos ya nos sale el ejecutable VBox_GAs(...).



Damos click derecho en la carpeta y pulsamos sobre Abrir Terminal:



Accedemos como Superusuario:

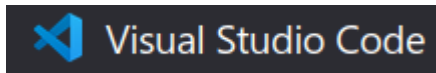
```
javi@jvePython: /media/cdrom0$ su
Password:
root@jvePython: /media/cdrom0#
```

Agregamos el comando para instalar el VBox Linux Additions:

```
root@jvePython: /media/cdrom0# sudo sh ./VBoxLinuxAdditions.run
verifying archive integrity... 100% MD5 checksums are OK. All good.
```

Y con esto ya quedaría instalado para poder aplicar las opciones de transparencia solicitadas.

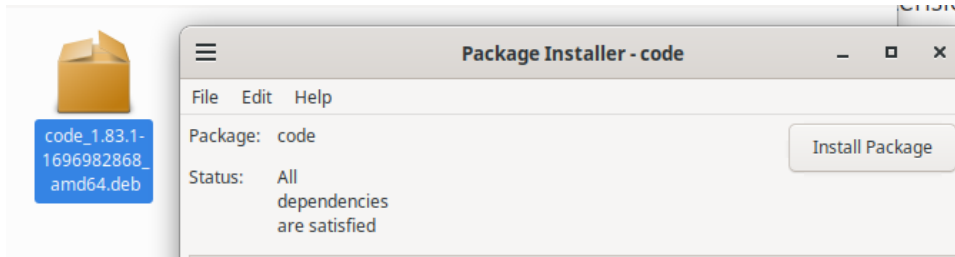
Instalación Visual Studio Code en Debian



Descargamos el instalador .deb de [Visual Studio](#) y lo instalamos.



Podemos instalar de manera automática el VSC haciendo doble click e instalando paquete:

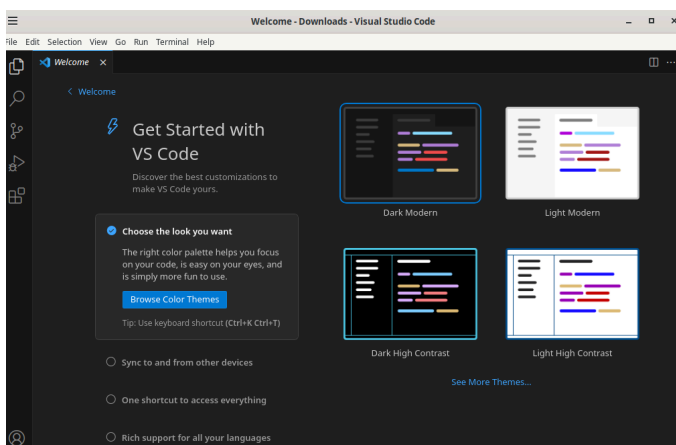
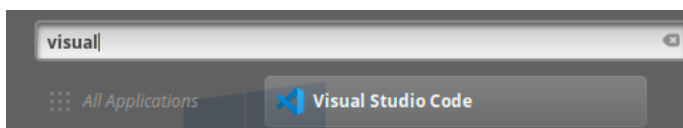


Si por lo que fuera, no podemos, se puede hacer vía línea de comandos abriendo terminal:

```
javi@jvePython:~/Downloads$ sudo dpkg -i code_1.83.1-1696982868_amd64.deb
[sudo] password for javi:
Selecting previously unselected package code.
(Reading database ... 179927 files and directories currently installed.)
Preparing to unpack code_1.83.1-1696982868_amd64.deb ...
Unpacking code (1.83.1-1696982868) ...
```

Podemos ahora ejecutarlo con “code .” o yendo a directamente al acceso directo:

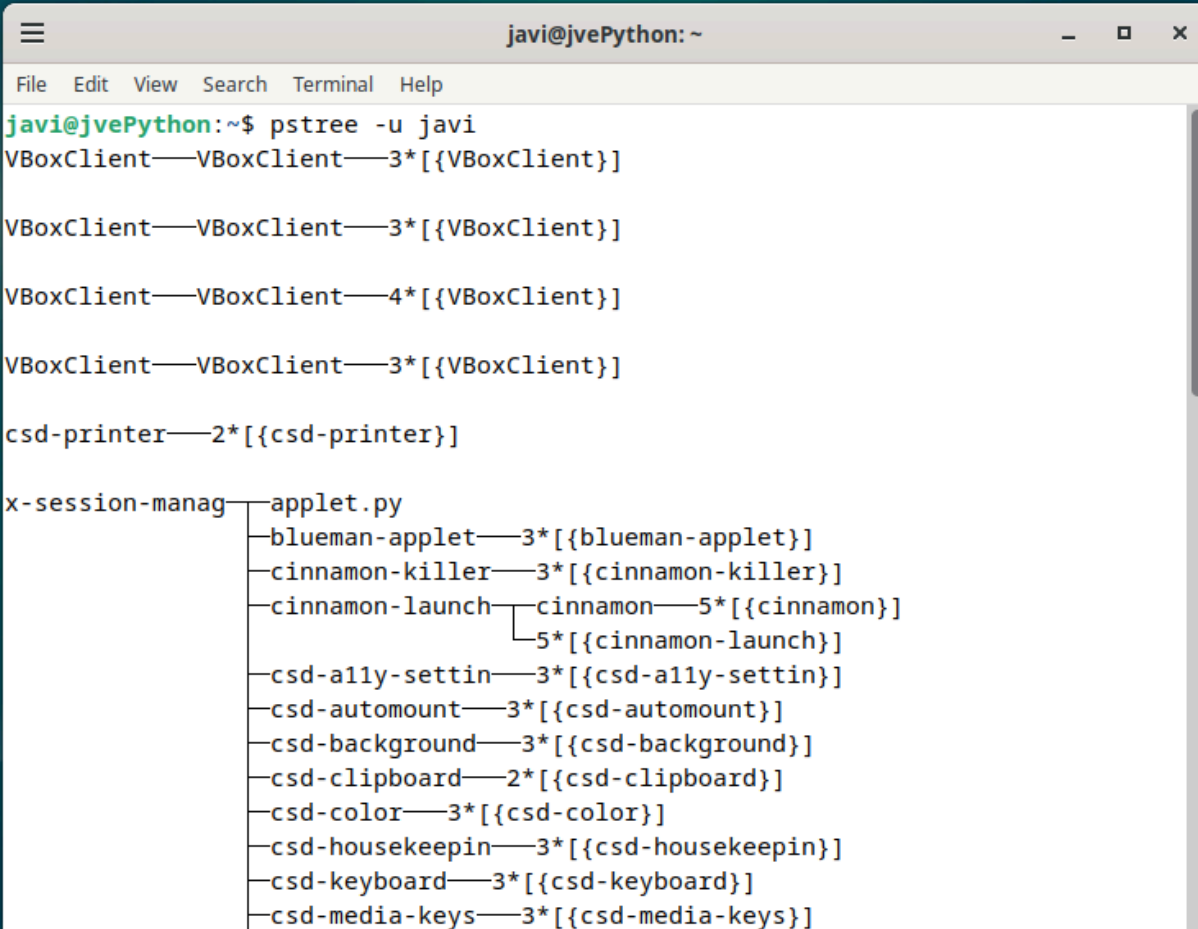
```
javi@jvePython:~/Downloads$ code .
```



Monitorizar los procesos del Sistema Debian (Shell)

Para supervisar los procesos en Debian, podemos aprovechar el comando **ps** desde la línea de comandos. Utilizando la opción **-u "usuario"**, podemos enfocarnos en los procesos de un usuario específico. Esto proporciona una visión estructurada de los procesos y sus relaciones en el sistema, facilitando la identificación de procesos padre e hijo y permitiendo un mejor control de la actividad del sistema.

Para monitorizar los procesos del sistema utilizamos el comando 'pstree' desde terminal:
pstree -u "usuario"



```
javi@jvePython: ~  
File Edit View Search Terminal Help  
javi@jvePython:~$ pstree -u javi  
VBoxClient—VBoxClient—3*[{VBoxClient}]  
  
VBoxClient—VBoxClient—3*[{VBoxClient}]  
  
VBoxClient—VBoxClient—4*[{VBoxClient}]  
  
VBoxClient—VBoxClient—3*[{VBoxClient}]  
  
csd-printer—2*[{csd-printer}]  
  
x-session-manag—applet.py  
                  —blueman-applet—3*[{blueman-applet}]  
                  —cinnamon-killer—3*[{cinnamon-killer}]  
                  —cinnamon-launch—cinnamon—5*[{cinnamon}]  
                  —cinnamon-launch—5*[{cinnamon-launch}]  
                  —csd-a11y-settin—3*[{csd-a11y-settin}]  
                  —csd-automount—3*[{csd-automount}]  
                  —csd-background—3*[{csd-background}]  
                  —csd-clipboard—2*[{csd-clipboard}]  
                  —csd-color—3*[{csd-color}]  
                  —csd-housekeepin—3*[{csd-housekeepin}]  
                  —csd-keyboard—3*[{csd-keyboard}]  
                  —csd-media-keys—3*[{csd-media-keys}]
```

Identificación de Procesos (PID) en el Sistema Debian

Para verificar la relación entre procesos padre e hijo en un sistema Debian, utilizamos la orden **ps** con opciones específicas. Ejecutamos el comando **ps -u TU_USUARIO -o pid,ppid,command**, donde **TU_USUARIO** es el nombre del usuario cuyos procesos queremos analizar. Esto nos proporciona una lista detallada de procesos, con información como el ID del proceso (PID), el ID del proceso padre (PPID) y el comando que inició el proceso. Esta información es crucial para comprender cómo se relacionan los procesos en el sistema y cómo se crean relaciones de padre e hijo entre ellos.

Comando: "ps -u TU_USUARIO -o pid,ppid,command"

```
javi@jvePython:~$ ps -u javi -o pid,ppid,command
PID    PPID  COMMAND
849      1  /lib/systemd/systemd --user
850     849  (sd-pam)
865     849  /usr/bin/pipewire
867     849  /usr/bin/wireplumber
868     849  /usr/bin/pipewire-pulse
869     849  /usr/bin/dbus-daemon --session --address=systemd: --nofork --nop
870     849  /usr/bin/gnome-keyring-daemon --foreground --components=pkcs11,s
877     844  x-session-manager
943      1  /usr/bin/VBoxClient --clipboard
945     943  /usr/bin/VBoxClient --clipboard
958      1  /usr/bin/VBoxClient --seamless
960     958  /usr/bin/VBoxClient --seamless
964      1  /usr/bin/VBoxClient --draganddrop
966     964  /usr/bin/VBoxClient --draganddrop
973      1  /usr/bin/VBoxClient --vmsvga-session
975     973  /usr/bin/VBoxClient --vmsvga-session
976     877  /usr/bin/ssh-agent x-session-manager
984     849  /usr/libexec/gvfsd
989     849  /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f
990     849  /usr/libexec/at-spi-bus-launcher
```

Para verificar si un proceso es el padre de otro, busca un proceso con un "PID" que coincide con el "PPID" de otro proceso. Esto indica que el proceso con "PID" es el padre del proceso con "PPID".

En el contexto de la salida de la orden ps -la, los datos indican lo siguiente:

El **PID** (identificador del proceso) es **849**.

El **PPID** (identificador del **proceso padre**) es **1**.

El **comando** (Command) es **"/lib/system/systemd --user"**.

Esto sugiere que el proceso con **PID 849** es un **proceso secundario (hijo)** del proceso con **PID 1** y que el comando **"/lib/system/systemd --user"** es el comando que se está ejecutando en ese proceso.

¿Cuáles serían las opciones necesarias para mostrar los procesos de un usuario concreto en vista detallada?

Para mostrar los procesos de un usuario concreto en vista detallada en Debian, debes utilizar la siguiente combinación de opciones con el comando ps.

En nuestro caso sería: ps -u javi -l

Donde javi es el nombre de usuario.

Las opciones utilizadas son las siguientes:

- u: Especifica el usuario cuyos procesos deseas listar.
- l: Muestra una salida detallada que incluye información adicional sobre los procesos, como el estado, el tiempo de CPU utilizado y otros detalles.

```
javi@jvePython:~$ ps -u javi -l
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	1000	849	1	0	80	0	-	4804	do_epo	?	00:00:00	systemd
5	S	1000	850	849	0	80	0	-	25853	-	?	00:00:00	(sd-pam)
0	S	1000	865	849	0	69	-11	-	13145	do_epo	?	00:00:00	pipewire
0	S	1000	867	849	0	69	-11	-	102843	do_sys	?	00:00:00	wireplumbe
0	S	1000	868	849	0	69	-11	-	6886	do_epo	?	00:00:00	pipewire-p
0	S	1000	869	849	0	80	0	-	2498	do_epo	?	00:00:00	dbus-daemo
0	S	1000	870	849	0	80	0	-	59972	do_sys	?	00:00:00	gnome-keyr
4	S	1000	877	844	0	80	0	-	93906	do_sys	?	00:00:00	x-session-
1	S	1000	943	1	0	80	0	-	4794	do_wai	?	00:00:00	VBoxClient
1	S	1000	945	943	0	80	0	-	54341	futex_	?	00:00:00	VBoxClient
1	S	1000	958	1	0	80	0	-	4794	do_wai	?	00:00:00	VBoxClient
1	S	1000	960	958	0	80	0	-	54367	futex_	?	00:00:04	VBoxClient
1	S	1000	964	1	0	80	0	-	4794	do_wai	?	00:00:00	VBoxClient
1	S	1000	966	964	0	80	0	-	54496	futex_	?	00:00:19	VBoxClient
1	S	1000	973	1	0	80	0	-	4794	do_wai	?	00:00:00	VBoxClient
1	S	1000	975	973	0	80	0	-	54359	futex_	?	00:00:01	VBoxClient
1	S	1000	976	877	0	80	0	-	1916	-	?	00:00:00	ssh-agent
0	S	1000	984	849	0	80	0	-	59376	do_sys	?	00:00:00	gvfsd
0	S	1000	989	849	0	80	0	-	95091	futex_	?	00:00:00	gvfsd-fuse
0	S	1000	990	849	0	80	0	-	77843	do_sys	?	00:00:00	at-spi-bus

Comando Kill

El comando **kill -9** seguido del número de proceso asignado se utiliza para forzar la terminación inmediata de un proceso en Linux. Cuando se ejecuta esta instrucción, el proceso especificado se detiene abruptamente sin darle la oportunidad de realizar ninguna limpieza o cierre adecuado.

Para localizar el terminal utilizamos el comando: ps aux | grep terminal

```
javi@jvePython:~$ ps aux | grep terminal
```

javi	3320	0.1	1.7	559740	52412	?	Ssl	21:27	0:02	/usr/libexec/g
nome-terminal-server										

Y aplicamos el kill -9:

```
javi@jvePython:~$ kill -9 3320
```

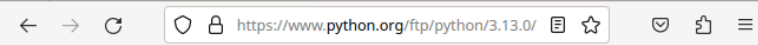
Esto cerrará la ventana de terminal sin mensaje de confirmación.

Ejecutar el programa “python ACT1_EJ1_v2.py”

Comprobamos qué versión de Python tenemos instalados con: **python3 -V**

```
javi@jvePython:~$ python3 -V
Python 3.11.2
```

Estando como superusuario nuevamente: “su”



Index of /ftp/python/3.13.0/

../	13-Oct-2023 11:08	-
amd64a1/	13-Oct-2023 11:08	-
arm64a1/	13-Oct-2023 11:08	-
win32a1/	13-Oct-2023 11:08	-
Python-3.13.0a1.tar.xz	13-Oct-2023 09:24	20137588
Python-3.13.0a1.tar.xz.asc	13-Oct-2023 09:24	963
Python-3.13.0a1.tar.xz.crt	13-Oct-2023 09:24	1071
Python-3.13.0a1.tar.xz.sig	13-Oct-2023 09:24	141
Python-3.13.0a1.tar.xz.sigstore	13-Oct-2023 09:24	4918
Python-3.13.0a1.tgz	13-Oct-2023 09:24	26485133
Python-3.13.0a1.tgz.asc	13-Oct-2023 09:24	963
Python-3.13.0a1.tgz.crt	13-Oct-2023 09:24	1067
Python-3.13.0a1.tgz.sig	13-Oct-2023 09:24	141

Para actualizar Python vamos a la [web oficial](https://www.python.org/) y elegimos la última versión.

Copiamos el enlace y en terminal, vamos a la carpeta opt con: **cd /opt/**
y seguido lo instalamos con **wget -c**:

```
javi@jvePython:~$ cd /opt/
javi@jvePython:/opt$ wget -c https://www.python.org/ftp/python/3.13.0/Python-3.13.0a1.tgz
```

Para descomprimir el archivo recién descargado usamos el comando **tar xzvf**:

```
root@jvePython:/opt# tar xzvf Python-3.13.0a1.tgz
Python-3.13.0a1/
Python-3.13.0a1/.cirrus-DISABLED.yml
Python-3.13.0a1/.coveragerc
```

Para modificar los permisos y otorgarle usuario root, del grupo root:

```
root@jvePython:/opt# chown -R root:root Python-3.13.0a1
root@jvePython:/opt# ls -la
total 25884
drwxr-xr-x  4 root root    4096 Oct 24 23:42 .
drwxr-xr-x 19 root root    4096 Oct 24 08:31 ..
drwxr-xr-x 17 root root    4096 Oct 13 10:52 Python-3.13.0a1
-rw-r--r--  1 root root 26485133 Oct 13 11:24 Python-3.13.0a1.tgz
drwxr-xr-x  8 root root    4096 Oct 24 17:33 VBoxGuestAdditions-7.0.12
root@jvePython:/opt#
```

Para compilar Python necesitaremos instalar varias herramientas y librerías, como **Build-essential** entre otros, que detallo a continuación:

```
root@jvePython:/opt# apt install build-essential zlib1g-dev libgdm-dev
libssl-dev libsqlite3-dev libreadline-dev libffi-dev curl libbz2-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9).
build-essential set to manually installed.
```

Entramos a la carpeta con **CD** y ejecutamos la configuración para mejorar la compatibilidad de hardware cuando compilemos Python, y habilitar el parámetro de compartir.

```
root@jvePython:/opt/Python-3.13.0a1# ./configure --enable-optimizations
--enable-shared
```

Ahora usamos el comando **make** para compilar toda la versión de código Python 3.13.0. El 2 indica la cantidad de cores que tenemos en esta máquina; es opcional poner la cantidad de cores que quieras emplear.

```
root@jvePython:/opt/Python-3.13.0a1# make -j 2
```

Una vez hecho esto, ejecutamos **make altinstall** para que se instalen los atributos, permisos para poder trabajar posteriormente con Python - se hace de manera automática.

```
make[1]: Leaving directory '/opt/Python-3.13.0a1'
root@jvePython:/opt/Python-3.13.0a1# make altinstall
```

Para finalizar, usamos el comando **ldconfig** para que se carguen los módulos y no tengamos problemas al trabajar con esta versión de Python.

```
root@jvePython:/opt/Python-3.13.0a1# sudo /sbin/ldconfig /opt/Python-3.13.0a1/
```

Confirmamos la versión usando **python3 -V**:

```
root@jvePython:/# python3 -V
Python 3.11.2
```

Ejecutamos el archivo "python ACT1_EJ1_v2.py".

Mensaje de error; corregimos el script de Python y ejecutamos

La primera vez, al intentar ejecutarlo me da error porque la versión actual de Python contempla el uso de paréntesis en parte de la sintaxis dada.

```
root@jvePython: /home/javi/Documents/Zocalos# python3 python\ ACT1_EJ1_v2.py
File "/home/javi/Documents/Zocalos/python ACT1_EJ1_v2.py", line 11
    print"texto:", str
    ^^^^^^^^^^^^^^^^^^^^^
```

El texto corregido debería ser así:

```
import os, sys

r, w = os.pipe()
processid = os.fork()

if processid:
    os.close(w)
    r = os.fdopen(r)
    print("Padre leyendo")
    str = r.read()
    print("texto:", str)
    sys.exit(0)
else:
    os.close(r)
    w = os.fdopen(w, 'w')
    print("Hijo escribiendo")
    w.write("Hello World")
    w.close()
    print("Hijo Cierra")
    sys.exit(0)
```

Y al ejecutarlo saldría el siguiente mensaje:

```
root@jvePython: /home/javi/Documents/Zocalos# python3 python\ ACT1_EJ1_v2.py
Padre leyendo
Hijo escribiendo
texto: Hello World
root@jvePython: /home/javi/Documents/Zocalos# Hijo Cierra
```

- El proceso padre muestra "Padre leyendo", lo que indica que está leyendo desde la tubería.
- El proceso hijo muestra "Hijo escribiendo", lo que indica que está escribiendo en la tubería.
- El proceso padre muestra "texto: Hello World", lo que indica que ha leído con éxito el texto escrito por el proceso hijo.
- El proceso hijo muestra "Hijo Cierra" antes de finalizar, lo que indica que ha cerrado la tubería después de escribir en ella.

Webgrafia

Debian OS - <https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/>

Python España - <https://es.python.org/aprende-python/>

Virtual Box - <https://www.virtualbox.org/>

Visual Studio Code - <https://code.visualstudio.com/>

Tutorial de instalación vía [Facultad Autodidacta](#)