

Requisitos previos

Haber realizado lo que se encuentra en [Instalación sistema operativo \(SO\) Ubuntu](#)

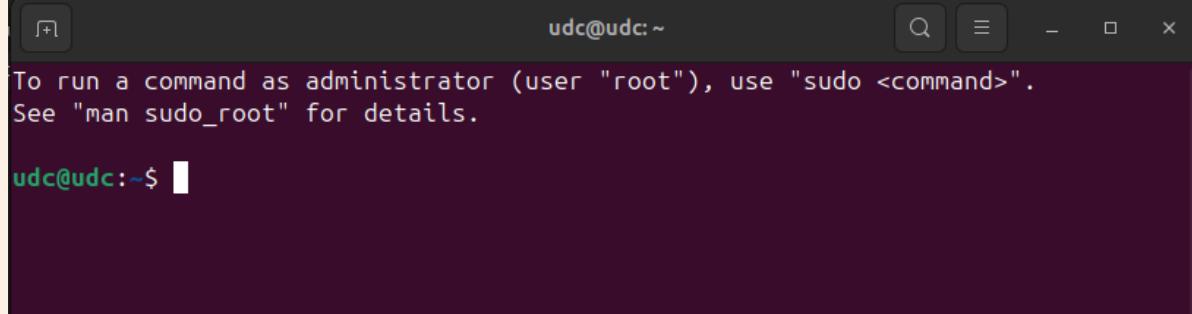
Requerimientos mínimos

- Ubuntu 20.04 o superior
- [CMake](#)
- Compilador compatible con C++11 o superior

⚠ Si es el primer contacto con el sistema operativo

Realizar lo siguiente:

- Abrir una terminal, se puede abrir una terminal con la siguiente terminal `CTRL + ALT + T`.
- Y se abrirá una terminal

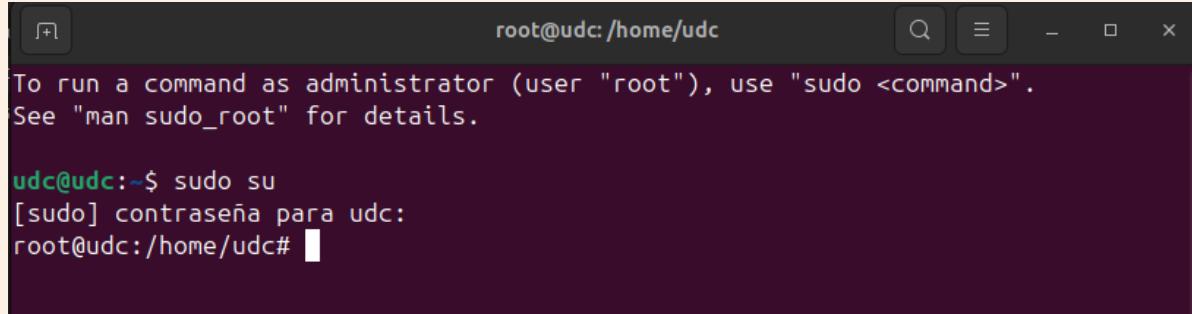


A screenshot of a terminal window titled "udc@udc: ~". The window has standard Linux window controls at the top right. The terminal itself is dark-themed. It displays the following text:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
udc@udc:~\$

- Que pedirá acceder como Super usuario/su

`sudo su`

- Se cambia la visión de la terminal



A screenshot of a terminal window titled "root@udc: /home/udc". The window has standard Linux window controls at the top right. The terminal is dark-themed and shows:
root@udc: /home/udc
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
udc@udc:~\$ sudo su
[sudo] contraseña para udc:
root@udc:/home/udc#

- Ahora vamos a actualizar el sistema operativo, ingresando los siguientes comandos y damos Enter .

```
sudo apt update && sudo apt upgrade -y
```

- Esperamos a que termine el proceso de actualización

```
Configurando nautilus (1:46.4-0ubuntu0.2) ...
Configurando gir1.2-gtk-4.0:amd64 (4.14.5+ds-0ubuntu0.7) ...
Configurando tecla (46.0-1ubuntu0.1) ...
Configurando mutter-common-bin (46.2-1ubuntu0.24.04.13) ...
Configurando libgtk-4-media-gstreamer (4.14.5+ds-0ubuntu0.7) ...
Configurando simple-scan (46.0-0ubuntu2.1) ...
Configurando gnome-shell-extension-desktop-icons-ng (46+really47.0.9-1ubuntu5) ...
...
Configurando gnome-control-center (1:46.7-0ubuntu0.24.04.3) ...
Configurando libmutter-14:amd64 (46.2-1ubuntu0.24.04.13) ...
Configurando gnome-remote-desktop (46.3-0ubuntu1.1) ...
Configurando gir1.2-mutter-14:amd64 (46.2-1ubuntu0.24.04.13) ...
Procesando disparadores para desktop-file-utils (0.27-2build1) ...
Procesando disparadores para initramfs-tools (0.142ubuntu25.5) ...
update-initramfs: Generating /boot/initrd.img-6.14.0-37-generic
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
Procesando disparadores para gnome-menus (3.36.0-1.1ubuntu3) ...
Procesando disparadores para libc-bin (2.39-0ubuntu8.6) ...
Procesando disparadores para syslog (8.2312.0-3ubuntu9.1) ...
Procesando disparadores para man-db (2.12.0-4build2) ...
Procesando disparadores para dbus (1.14.10-4ubuntu4.1) ...
Procesando disparadores para shared-mime-info (2.4-4) ...

[ Progreso: [ 99% ] [#########################################] ]
```

- Con eso ya el sistema operativo se actualizo y se encuentra al día con los paquetes recién actualizados.

1. Instalar dependencias

Pasos previos

- Haber actualizado los paquetes del sistema operativo

🔥 Dependencias

- Ejecutar en la terminal el siguiente comando

```
sudo apt install libglfw3-dev libxinerama-dev libxcursor-dev libxi-dev libyaml-cpp-dev libeigen3-dev git cmake build-essential python3-pip python3-venv libboost-all-dev libspdlog-dev -y
```

Realizar lo siguiente

- Crear una carpeta donde ubicaremos la clonación de cada recurso y contrucción e instalación de dependencias de los mismos

2. Clonación Mujoco y construcción

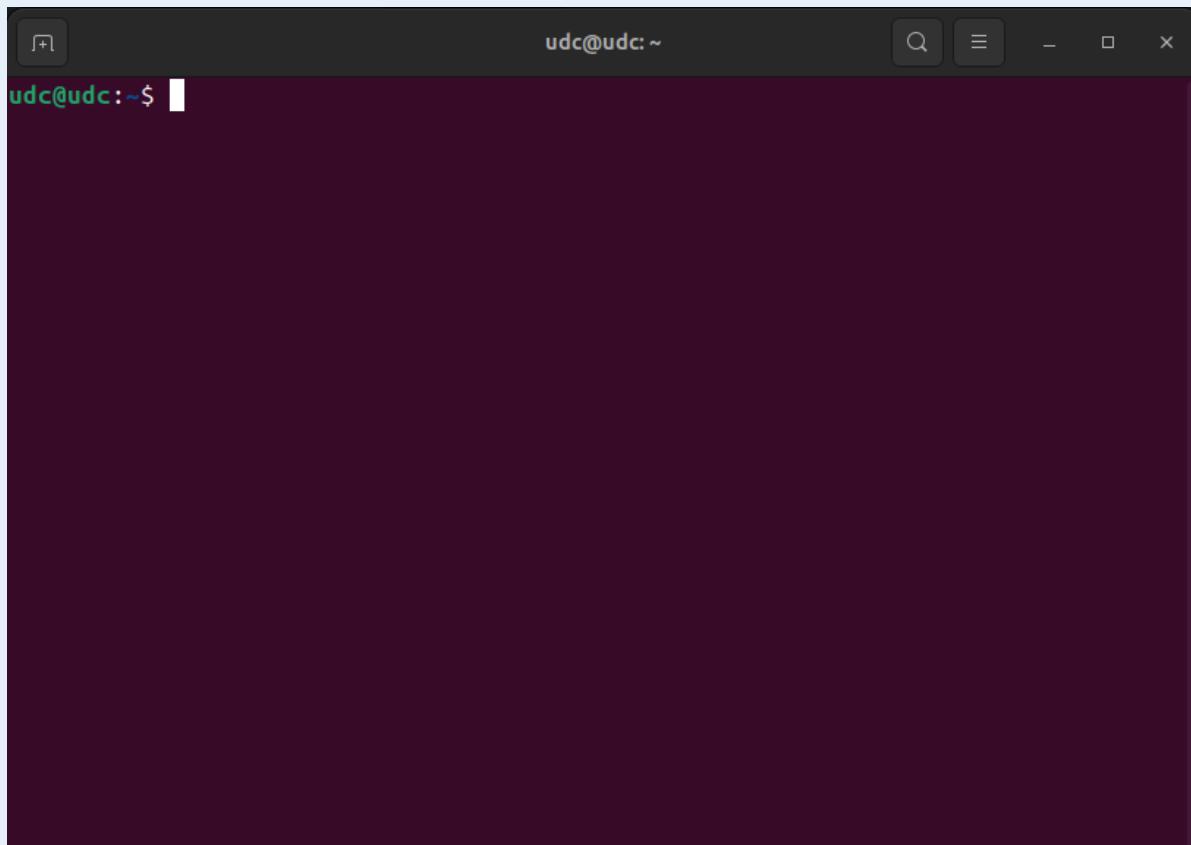
Implementación de los entornos que constituyen el desarrollo para la programación y pruebas, para el robot Unitree-G1 .

Tener en cuenta!!

Debemos tener creada una carpeta donde vamos a realizar clonación y construcción del recurso para poder realizarlo paso a paso.

Pasos a realizar para la carpeta donde ubicaremos nuestros recursos

- Abrimos una terminal



```
udc@udc:~$
```

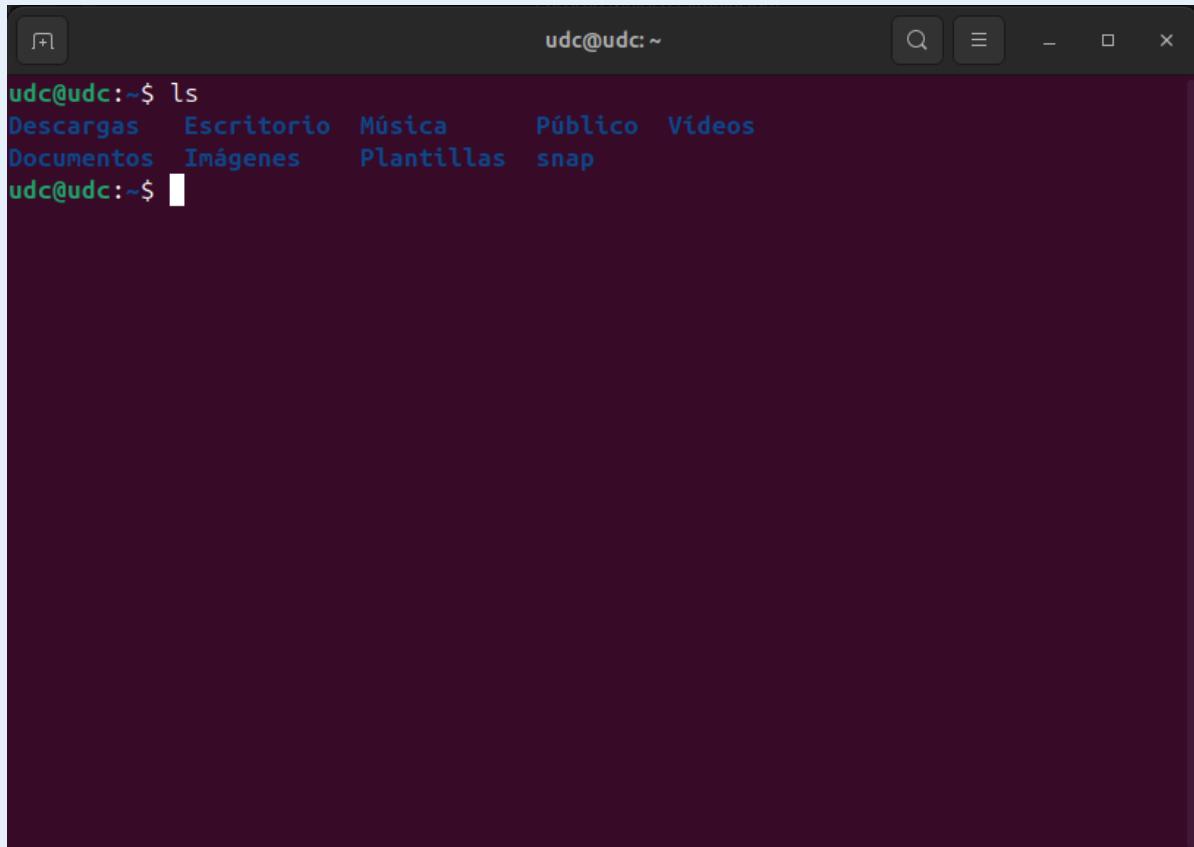
- Ingresamos el siguiente comando para identificar las carpetas existentes en la actual ubicación del terminal

```
ls
```

O

dir

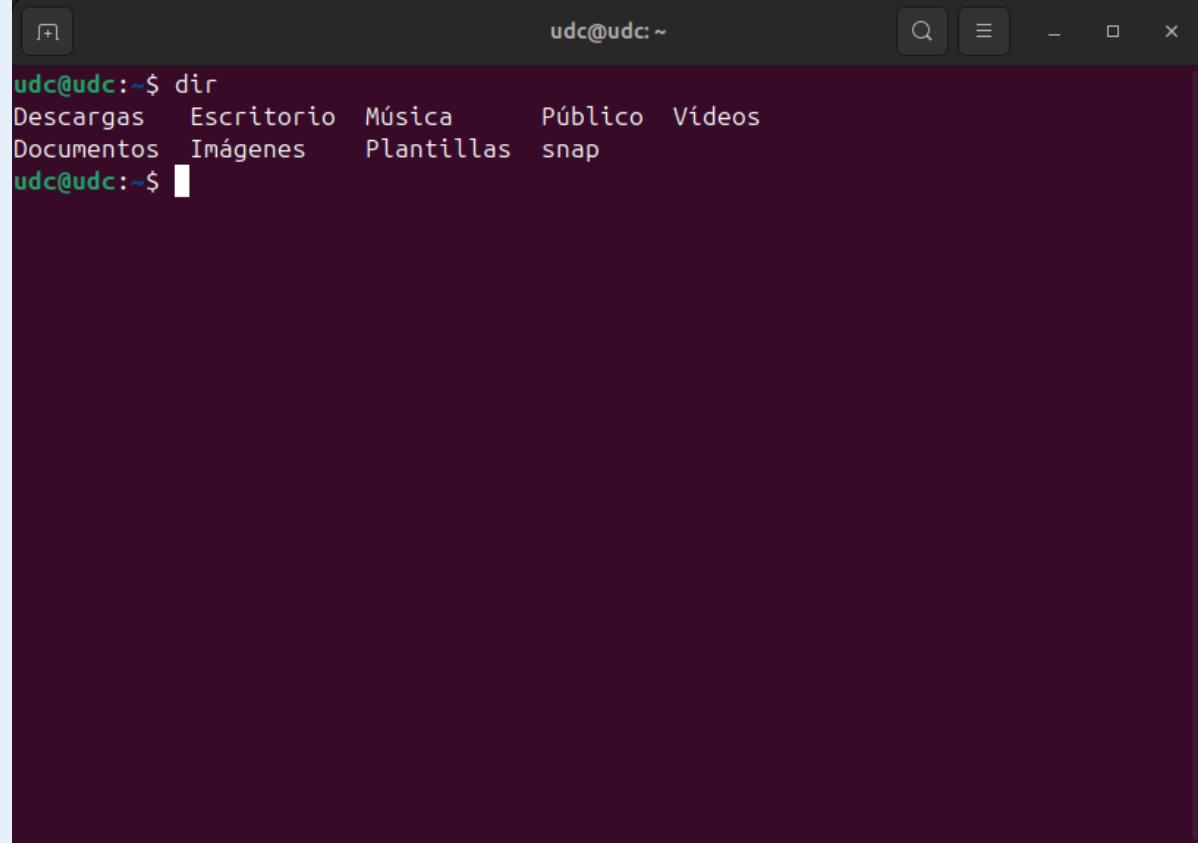
- Al usar uno de los dos comando podremos visualizar de la siguiente manera la salida del comando, demostraré como el comando `ls` nos da una salida:



A screenshot of a terminal window titled "udc@udc: ~". The window shows the command "ls" being run, which lists several directories: "Descargas", "Escritorio", "Música", "Público", "Vídeos", "Documentos", "Imágenes", "Plantillas", and "snap". The terminal has a dark background and light-colored text. The window title bar includes standard icons for minimize, maximize, and close.

```
udc@udc:~$ ls
Descargas Escritorio Música     Públco  Vídeos
Documentos Imágenes Plantillas snap
udc@udc:~$
```

- Ahora mostrare como se ve usando `dir`:



A screenshot of a terminal window titled 'udc@udc: ~'. The window shows the command 'dir' being run, which lists several directories: Descargas, Escritorio, Música, Público, Vídeos, Documentos, Imágenes, Plantillas, and snap. The terminal has a dark background and light-colored text.

- Los anteriores comando dan la misma salida, pero visualmente diferente.

-
- Ahora con el siguiente comando podemos visualizar el Path o Dirección actual, ingresando en la terminal el siguiente comando:

⚠ Importante ya que tendremos que saber ubicarnos entre carpetas y movernos entre ellas

```
pwd
```

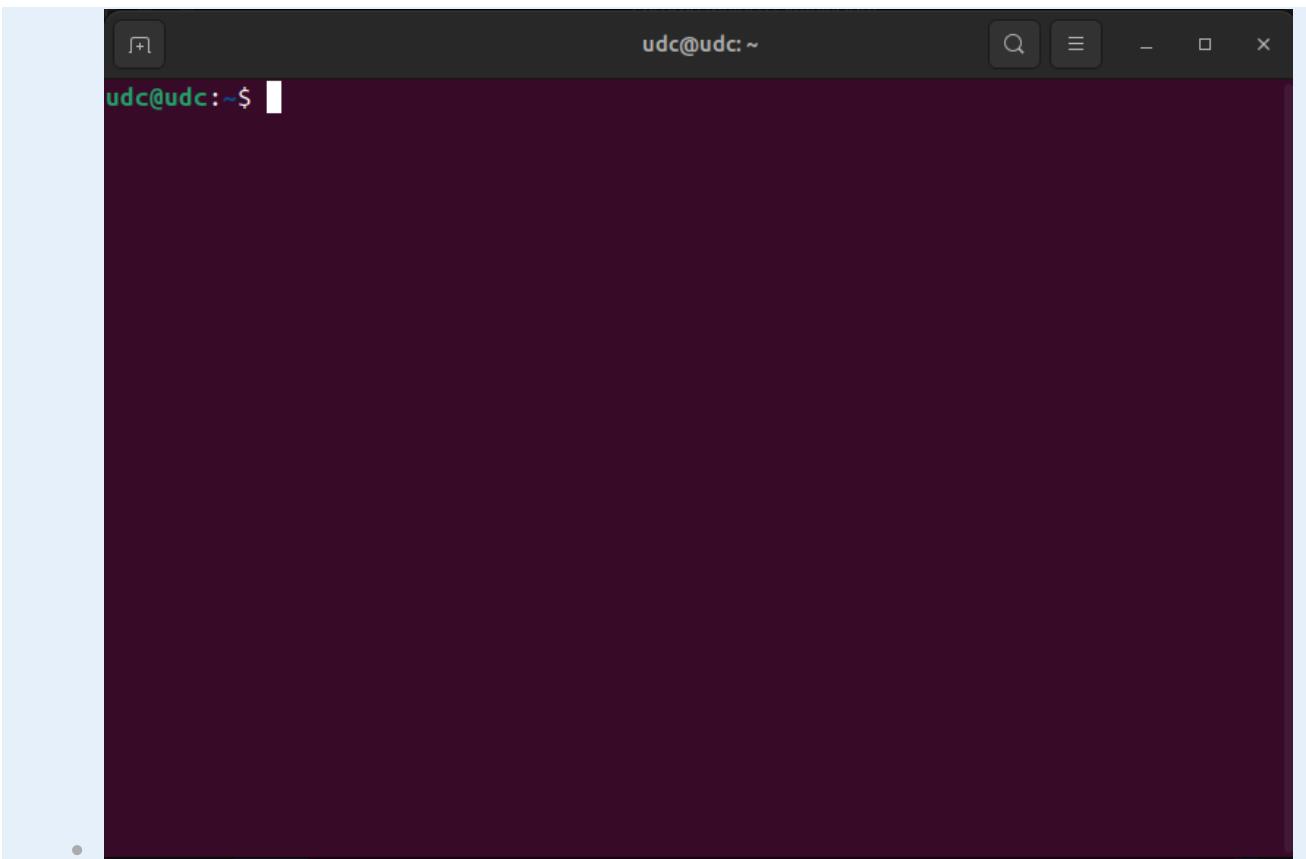
- Nos dara la siguiente salida:

- Eso es conocida como la raíz del usuario, no es lo mismo que la raíz del sistema.

Importante, trabajaremos desde la raíz del usuario, para reconocerlo es cuando en la terminal,

de manera general siempre vemos el siguiente simbolo ~, eso significa que estamos ubicados en la raíz del usuario.

-
- Como siguiente paso vamos a crear una carpeta desde la terminal y ubicarnos en esa carpeta.
 - Abrimos una terminal:



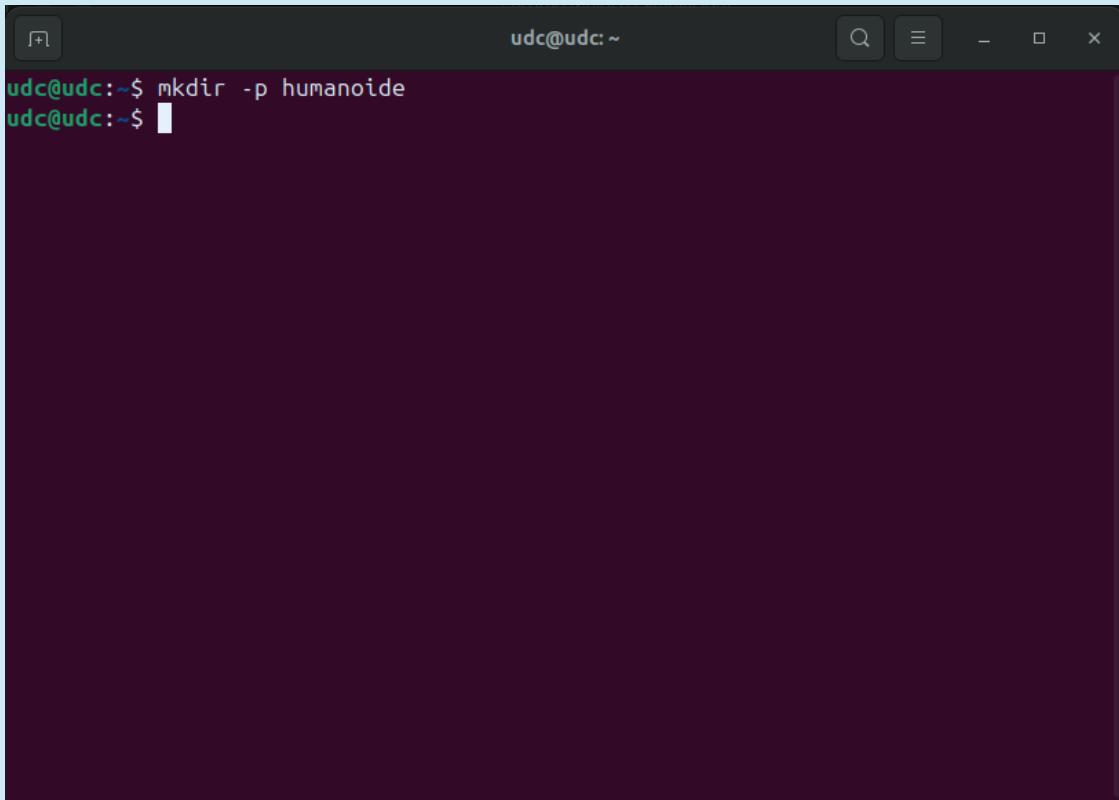
- Para el ejemplo usare el siguiente nombre para la carpeta principal que contendra los recursos `humanoide` .
- Ingresaremos en la terminal el siguiente comando junto al nombre de la carpeta:

```
mkdir -p humanoide
```

🔗 Funcionamiento del comando

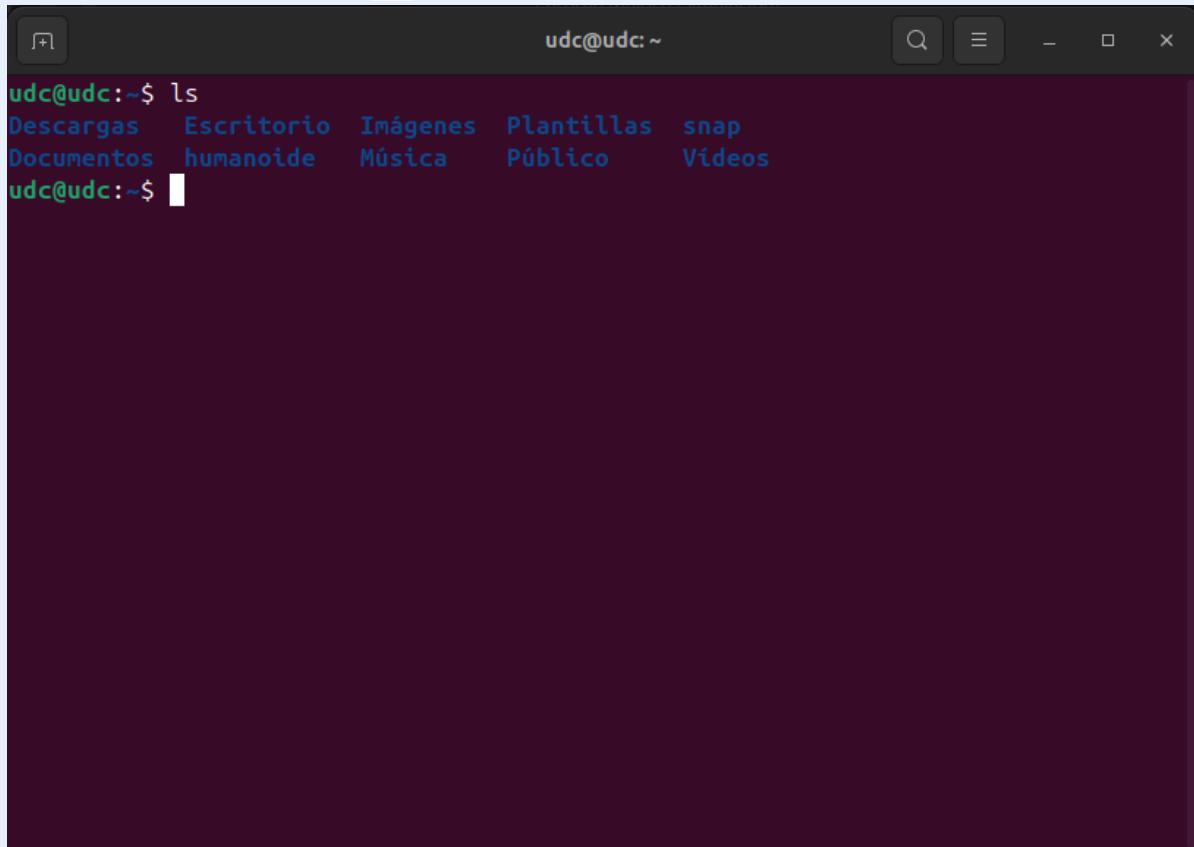
Ese comando ejecutara desde la terminal la creación de una carpeta, usando el comando `mkdir` , el uso de `-p` es una validación que nos lanzara un mensaje diciendo si el directorio ya existe, caso contrario de que sino lanza un mensaje significando que no existia y se creo la carpeta , y al final el nombre de la carpeta `humanoide` .

- Ejemplo:



```
udc@udc:~$ mkdir -p humanoide
udc@udc:~$
```

-
- Ahora para verificar que la carpeta se creo usar uno de los dos comando anteriormente mencionados para identificar las carpeta según donde estes ubicado, para este usare `ls`.



```
udc@udc:~$ ls
Descargas Escritorio Imágenes Plantillas snap
Documentos humanoide Música Público Vídeos
udc@udc:~$
```

- El siguiente paso es movernos a la carpeta creada, en este punto ya debemos saber y haber verificado que si existe la carpeta. Usando el comando `cd` (*Change directory*)

⚡ 🌍 ¿Para qué sirve exactamente?

✓ 1. Entrar a un directorio

```
cd humanoide
```

✓ 2. Regresar al directorio anterior

```
cd ..
```

✓ 3. Ir al directorio raíz / (Raíz del sistema)

```
cd /
```

✓ 4. Ir a tu carpeta personal (Raíz del usuario)

```
cd ~
```

o simplemente:

```
cd
```

✓ 5. Ir a una ruta absoluta

```
cd /home/udc/humanoide
```

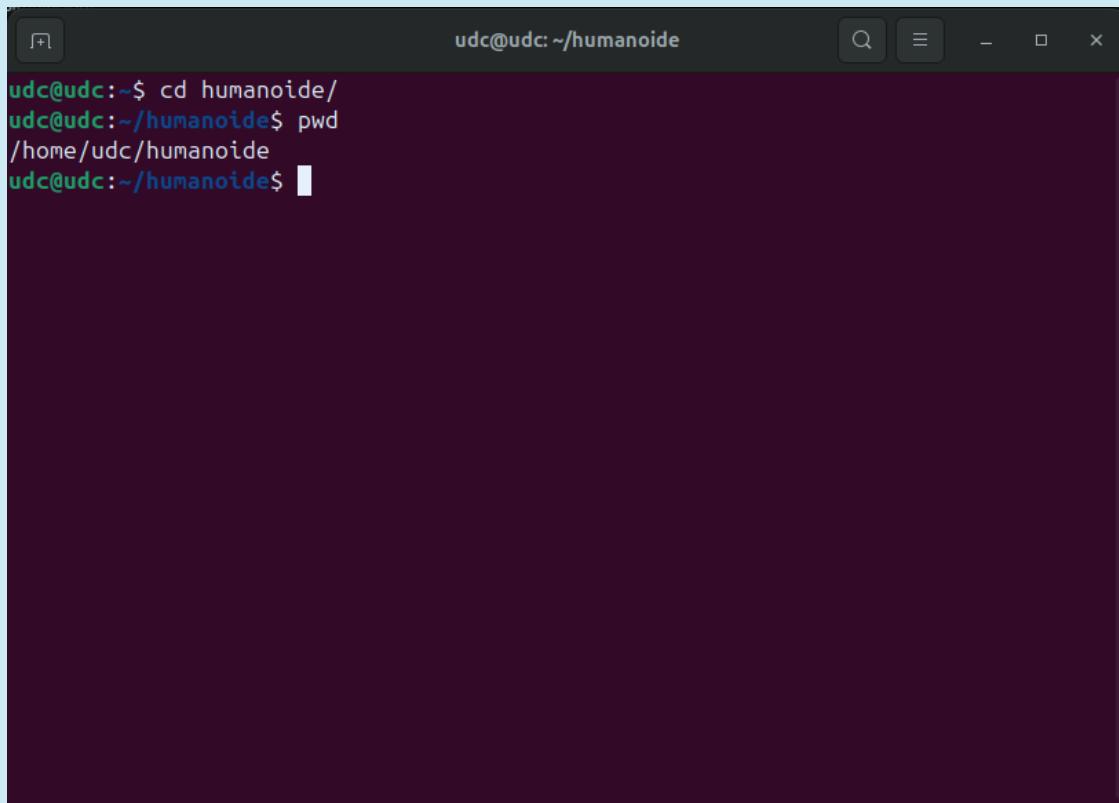
✓ 6. Ir a una ruta relativa

Estando en `/home/udc`, si haces:

```
cd humanoide
```

Te mueve a /home/udc/humanoide .

- Ejemplo:



```
udc@udc:~$ cd humanoide/
udc@udc:~/humanoide$ pwd
/home/udc/humanoide
udc@udc:~/humanoide$
```

- Al ya tener claro el funcionamiento, es importante familiarizarse con ello.

⚠ En este punto ya debemos saber como crear carpetas, movernos y salir de ellas.

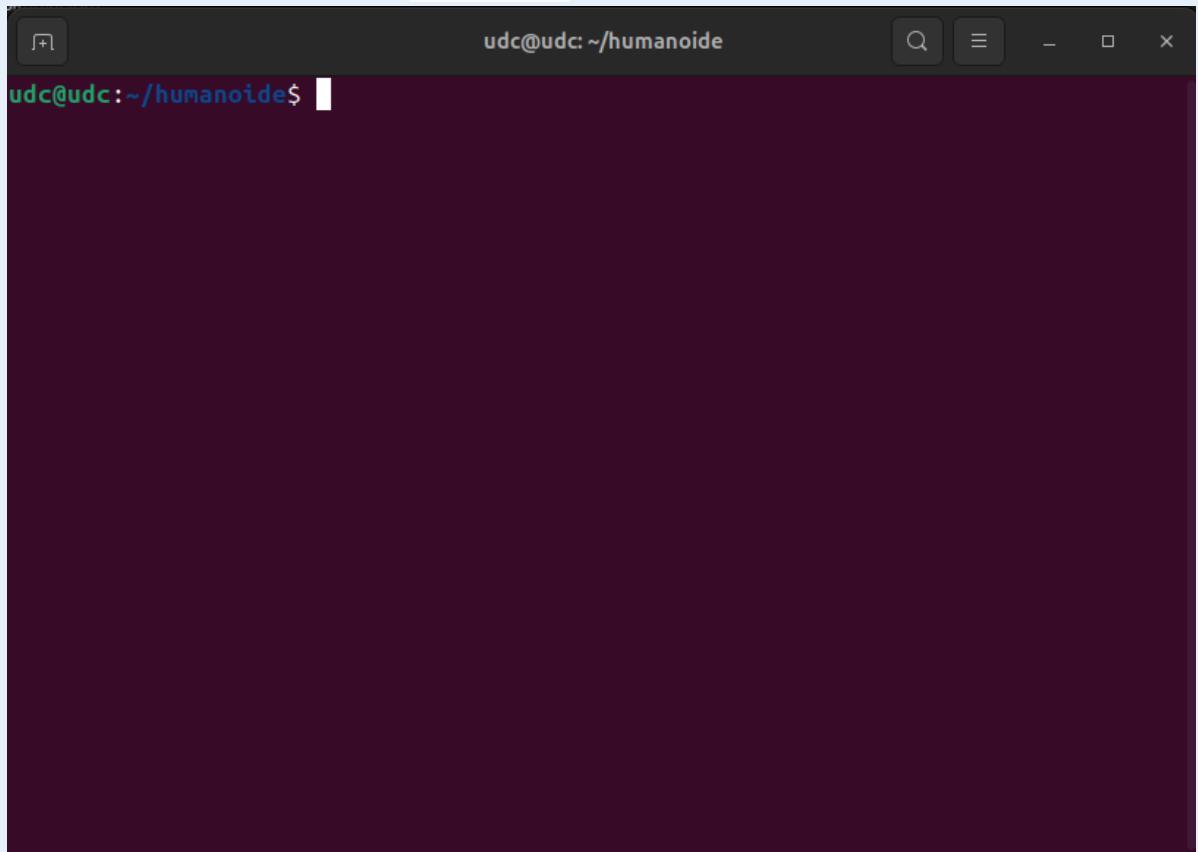
Clonación del recurso (Mujoco)

- Primer recurso a clonar y construir para el entorno a montar.

⚠ Haber realizado el paso anterior de creación y ubicación de la carpeta humanoide

Pasos

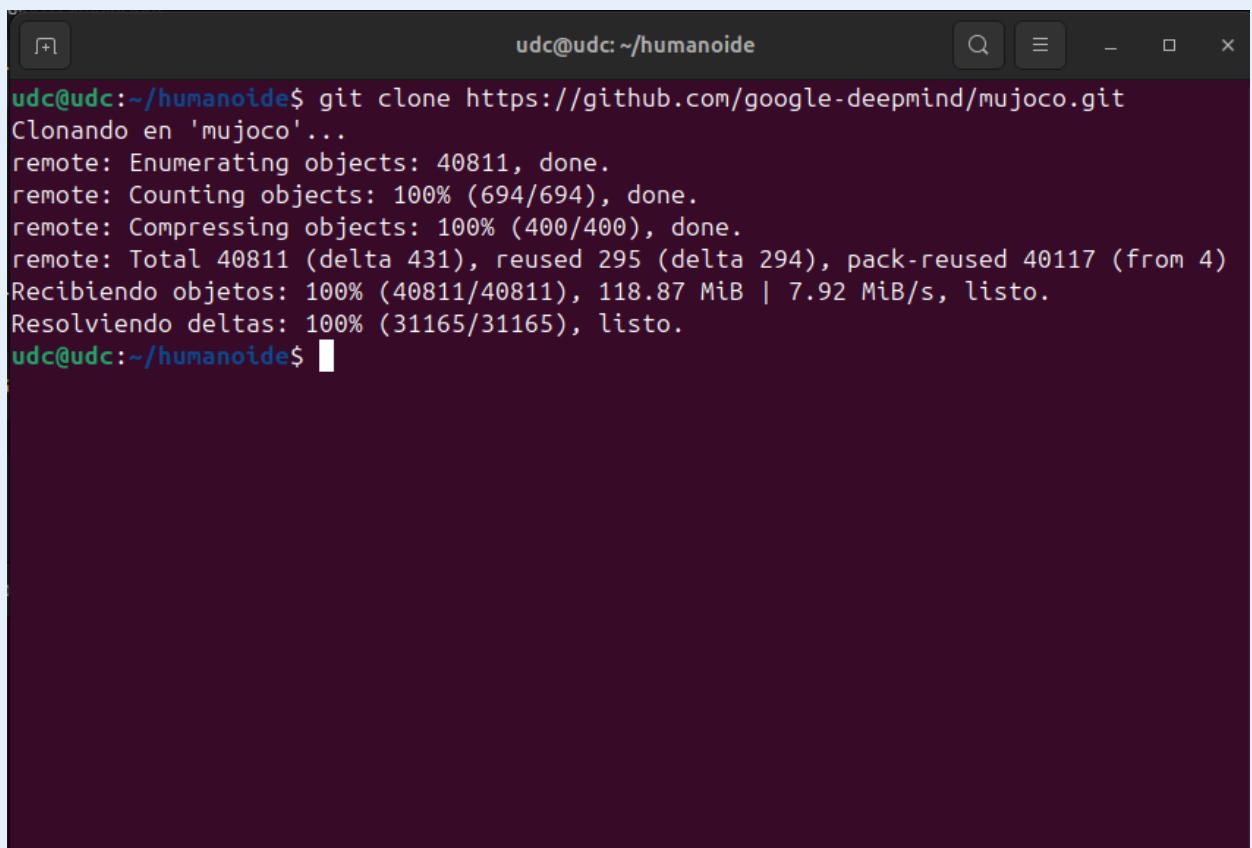
- Estar ubicado en la carpeta humanoide



A screenshot of a terminal window with a dark background. The title bar shows "udc@udc: ~/humanoide". The command prompt "udc@udc: ~/humanoide\$" is visible at the top left, followed by a blank white input field.

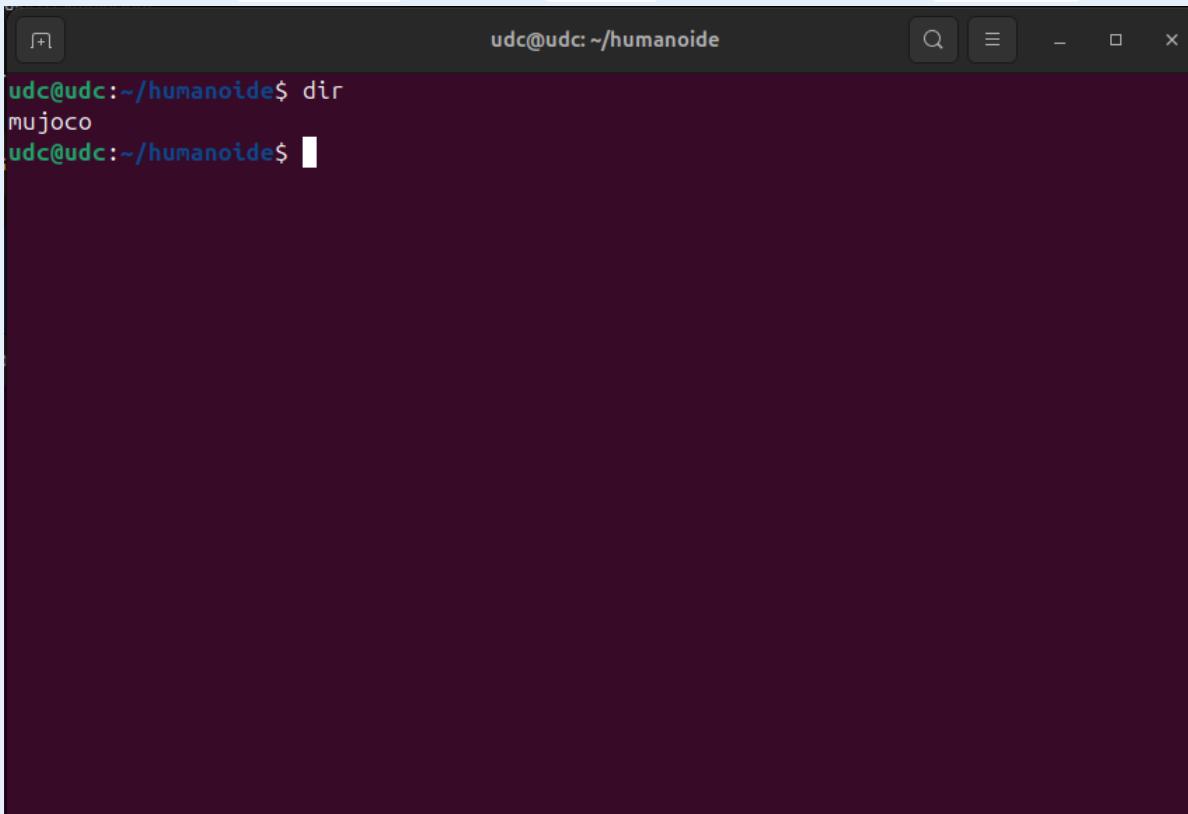
- Ingresar en la terminal el recurso a clonar y damos Enter .

```
git clone https://github.com/google-deepmind/mujoco.git
```



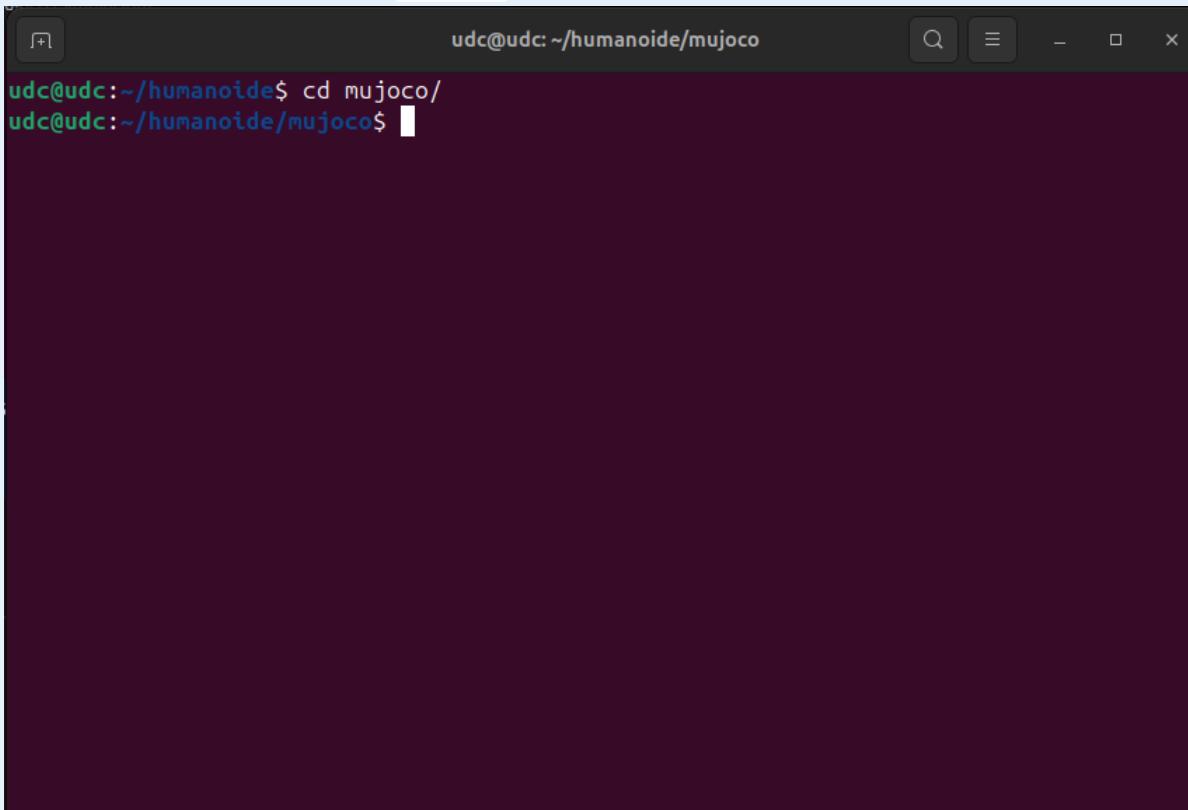
A screenshot of a terminal window showing the output of a "git clone" command. The title bar shows "udc@udc: ~/humanoide". The command "git clone https://github.com/google-deepmind/mujoco.git" is entered at the prompt. The terminal displays the progress of the cloning process, including object enumeration, counting, compressing, and receiving objects from the remote repository. The message "Recibiendo objetos: 100% (40811/40811), 118.87 MiB | 7.92 MiB/s, listo." indicates the completion of the download. The final prompt "Resolviendo deltas: 100% (31165/31165), listo." is shown, followed by the input field.

- Verificamos que al finalizar la clonación habrá una nueva carpeta en el interior de nuestra carpeta `humanoide`, llamada `mujoco`, para ello usamos `ls` o `dir`.



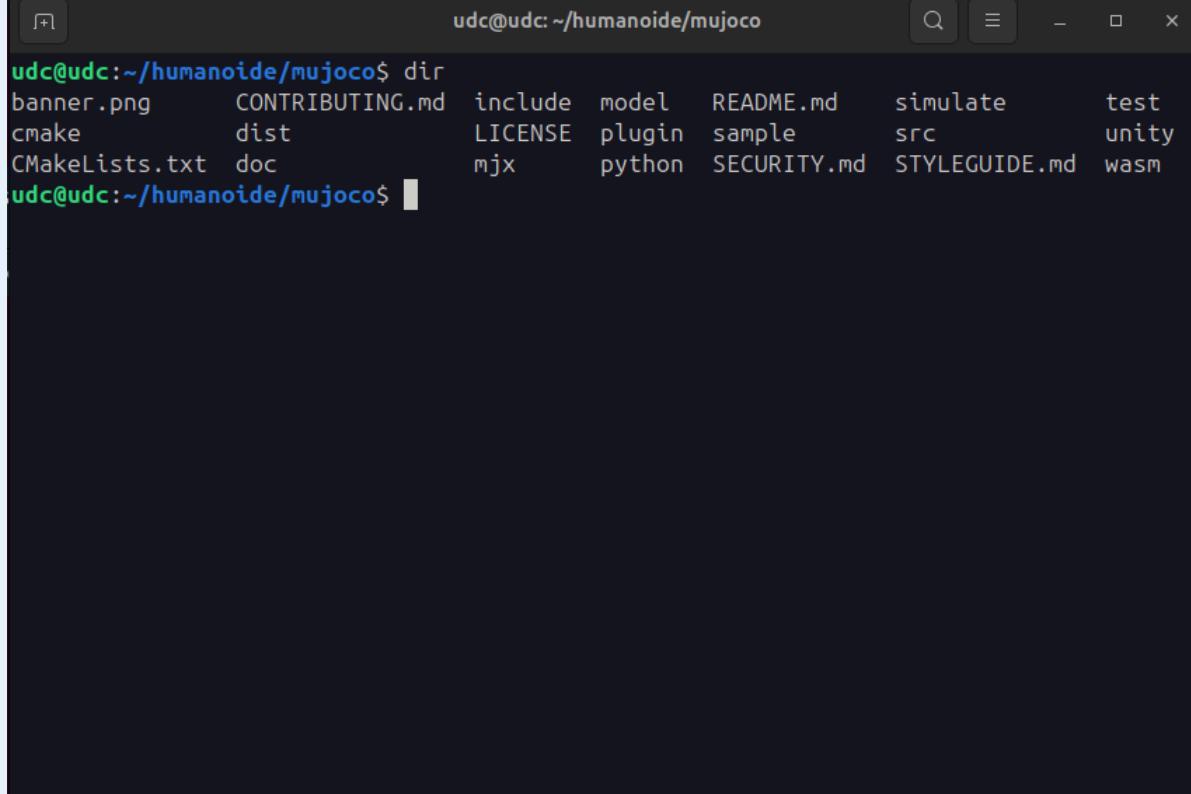
```
udc@udc:~/humanoide$ dir
mujoco
udc@udc:~/humanoide$
```

- Al ya validar que si exista, significa que se clono exitosa mente, y el siguiente paso es ingresar a la carpeta de `mujoco`.



```
udc@udc:~/humanoide$ cd mujoco/
udc@udc:~/humanoide/mujoco$
```

- Podemos visualizar el cotonido de `mujoco` al haber ingresado, usado `dir` o `ls`, nos mostrara archivos, carpetas e imagenes.

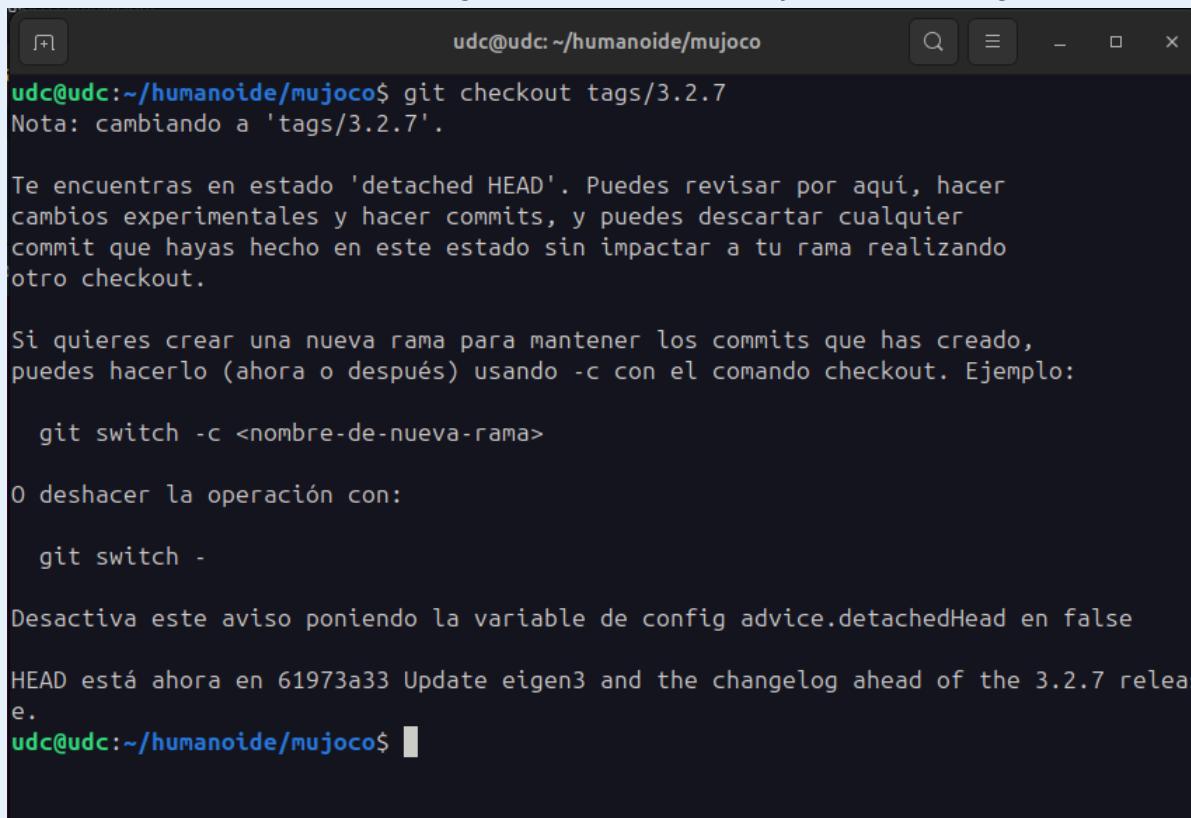


```
udc@udc:~/humanoide/mujoco$ dir
banner.png      CONTRIBUTING.md  include  model  README.md  simulate  test
cmake           dist             LICENSE  plugin  sample   src        unity
CMakeLists.txt  doc             mjx       python  SECURITY.md  STYLEGUIDE.md  wasm
udc@udc:~/humanoide/mujoco$
```

- Ahora haremos un cambio drástico, vamos a cambiar de versión de `mujoco` a una en particular `3.2.7`, lo realizamos usando el siguiente comando:

```
git checkout tags/3.2.7
```

- Continuamos en la terminal e ingresamos el comando y veremos lo siguiente:



```
udc@udc:~/humanoide/mujoco$ git checkout tags/3.2.7
Nota: cambiando a 'tags/3.2.7'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aquí, hacer cambios experimentales y hacer commits, y puedes descartar cualquier commit que hayas hecho en este estado sin impactar a tu rama realizando otro checkout.

Si quieras crear una nueva rama para mantener los commits que has creado, puedes hacerlo (ahora o después) usando -c con el comando checkout. Ejemplo:

git switch -c <nombre-de-nueva-rama>

O deshacer la operación con:

git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD está ahora en 61973a33 Update eigen3 and the changelog ahead of the 3.2.7 release.
udc@udc:~/humanoide/mujoco$
```

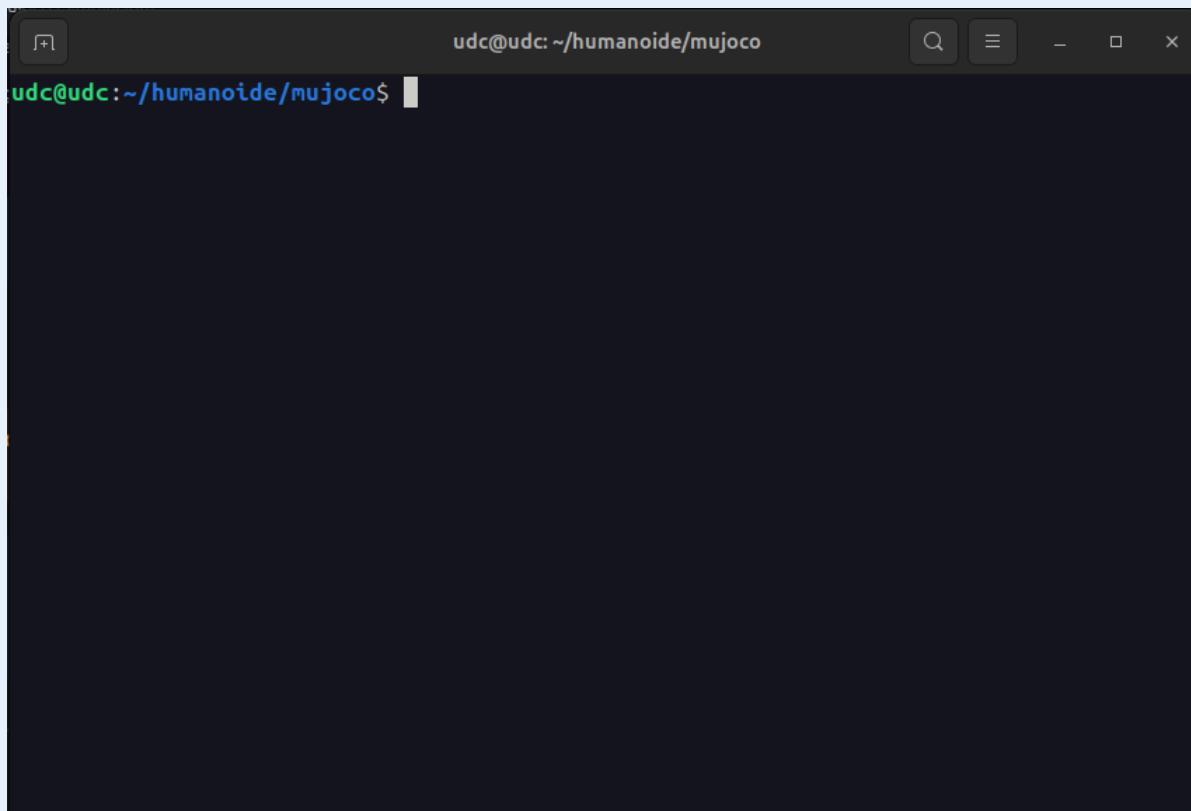
Significa que fue exitoso el cambio...

Construcción de Mujoco

⚠ Esta es la continuación para el recurso clonado importante estar en la carpeta humanoide/mujoco

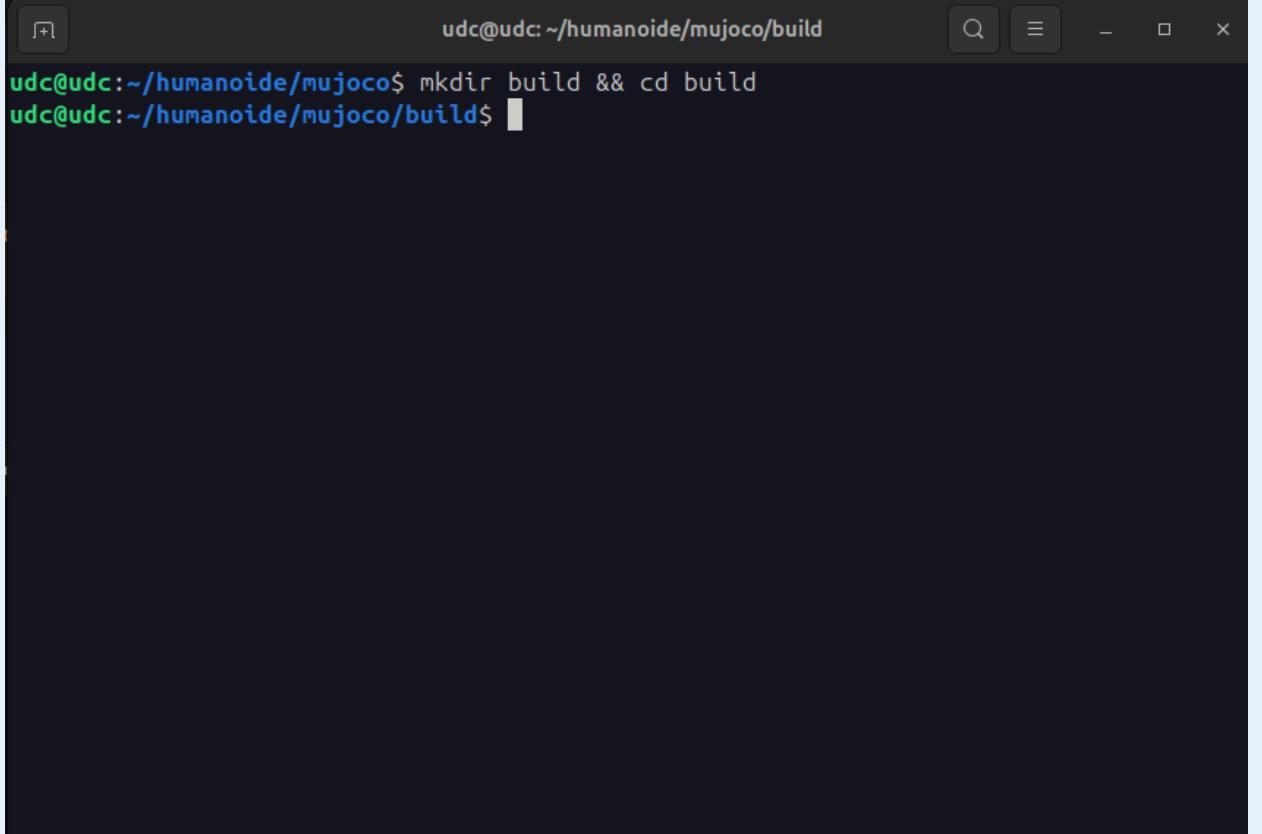
Pasos

- Desde la ubicación de:



- Vamos a crear una carpeta llamada `build` y movernos a ella desde una sola orden:

```
mkdir build && cd build
```

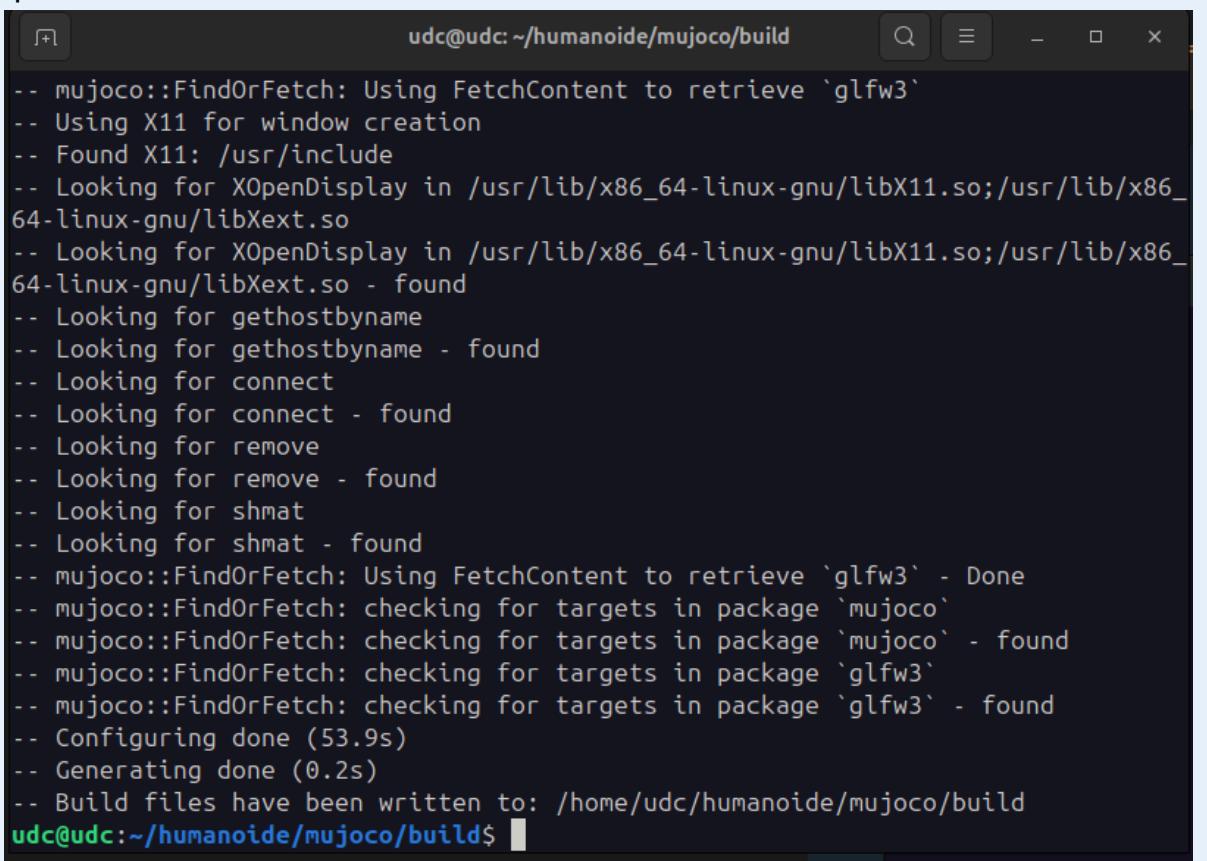


```
udec@udc:~/humanoide/mujoco/build$ mkdir build && cd build  
udec@udc:~/humanoide/mujoco/build$
```

- Y para iniciar la construcción usaremos `cmake`, el proximo paso es, en la terminal ingresar el siguiente comando:

```
cmake ..
```

- Esperamos unos segundos y en la parte final debemos ver el siguiente mensaje, que nos confirma la construcción exitosa



```
-- mujoco::FindOrFetch: Using FetchContent to retrieve `glfw3`  
-- Using X11 for window creation  
-- Found X11: /usr/include  
-- Looking for XOpenDisplay in /usr/lib/x86_64-linux-gnu/libX11.so;/usr/lib/x86_64-linux-gnu/libXext.so  
-- Looking for XOpenDisplay in /usr/lib/x86_64-linux-gnu/libX11.so;/usr/lib/x86_64-linux-gnu/libXext.so - found  
-- Looking for gethostbyname  
-- Looking for gethostbyname - found  
-- Looking for connect  
-- Looking for connect - found  
-- Looking for remove  
-- Looking for remove - found  
-- Looking for shmat  
-- Looking for shmat - found  
-- mujoco::FindOrFetch: Using FetchContent to retrieve `glfw3` - Done  
-- mujoco::FindOrFetch: checking for targets in package `mujoco`  
-- mujoco::FindOrFetch: checking for targets in package `mujoco` - found  
-- mujoco::FindOrFetch: checking for targets in package `glfw3`  
-- mujoco::FindOrFetch: checking for targets in package `glfw3` - found  
-- Configuring done (53.9s)  
-- Generating done (0.2s)  
-- Build files have been written to: /home/udc/humanoide/mujoco/build  
udec@udc:~/humanoide/mujoco/build$
```

- Continuando en la carpeta `build` ingresaremos el siguiente comando que nos finalizara contrucción

`make`

- Nuevamente esperamos unos minutos y podremos visualizar lo siguiente:

```
/elasticity_test.cc.o
[ 98%] Linking CXX executable ../../bin/elasticity_test
[ 98%] Built target elasticity_test
[ 98%] Building CXX object test/plugin/actuator/CMakeFiles/pid_test.dir/pid_test_
.cc.o
[ 98%] Linking CXX executable ../../bin/pid_test
[ 98%] Built target pid_test
[ 98%] Building CXX object python/mujoco/util/CMakeFiles/array_traits_test.dir/a
rray_traits_test.cc.o
[ 98%] Linking CXX executable ../../bin/array_traits_test
[ 98%] Built target array_traits_test
[ 98%] Building CXX object python/mujoco/util/CMakeFiles/func_traits_test.dir/fu
nc_traits_test.cc.o
[100%] Linking CXX executable ../../bin/func_traits_test
[100%] Built target func_traits_test
[100%] Building CXX object python/mujoco/util/CMakeFiles/func_wrap_test.dir/fun
c_wrap_test.cc.o
[100%] Linking CXX executable ../../bin/func_wrap_test
[100%] Built target func_wrap_test
[100%] Building CXX object python/mujoco/util/CMakeFiles/tuple_tools_test.dir/tu
ple_tools_test.cc.o
[100%] Linking CXX executable ../../bin/tuple_tools_test
[100%] Built target tuple_tools_test
udc@udc:~/humanoide/mujoco/build$
```

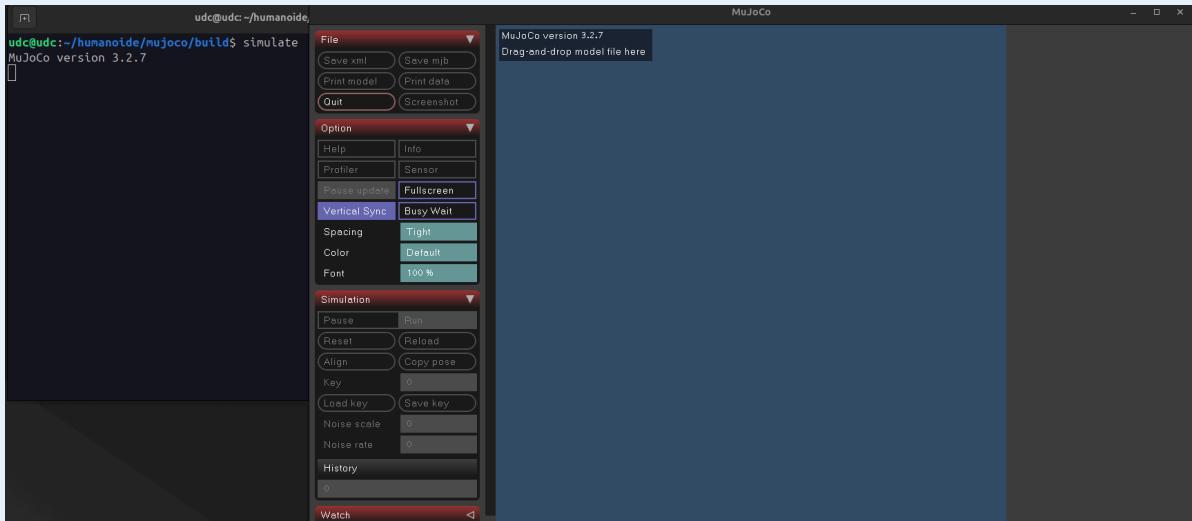
- Ahora colocamos otro

`sudo make install`

- Ya se construyo con éxito, para comprobar que todo se realizo debidamente lanzaremos el siguiente comando `simulate`, debemos aun estar en la carpeta `build`

`simulate`

- Visualizaremos lo siguiente



3 - Clonación y construcción de Unitre_sdk2

Este recurso debe estar en la carpeta `humanoide` donde clonamos el recurso de `mujoco`

```
udc@udc:~/humanoide$
```

Pasos

Clonar el recurso de `unitree_sdk2`, con `ls` debemos estar visualizando en la carpeta `humanoide` dos carpetas `mujoco` y `unitree_sdk2`.

```
git clone https://github.com/unitreerobotics/unitree_sdk2.git
```

A screenshot of a terminal window titled "udc@udc: ~/humanoide". The window shows the command "git clone https://github.com/unitreerobotics/unitree_sdk2.git" being run. The output of the command is displayed, showing the progress of cloning the repository, including object enumeration, counting, compressing, and receiving objects. Once completed, the directory listing shows two files: "mujoco" and "unitree_sdk2". The terminal prompt "udc@udc:~/humanoide\$" is visible at the bottom.

```
git clone https://github.com/unitreerobotics/unitree_sdk2.git
Clonando en 'unitree_sdk2'...
remote: Enumerating objects: 2622, done.
remote: Counting objects: 100% (579/579), done.
remote: Compressing objects: 100% (279/279), done.
remote: Total 2622 (delta 355), reused 300 (delta 300), pack-reused 2043 (from 2)
Recibiendo objetos: 100% (2622/2622), 90.96 MiB | 10.13 MiB/s, listo.
Resolviendo deltas: 100% (1170/1170), listo.
ls
mujoco  unitree_sdk2
udc@udc:~/humanoide$
```

- Ingresamos a la carpeta `unitree_sdk2`

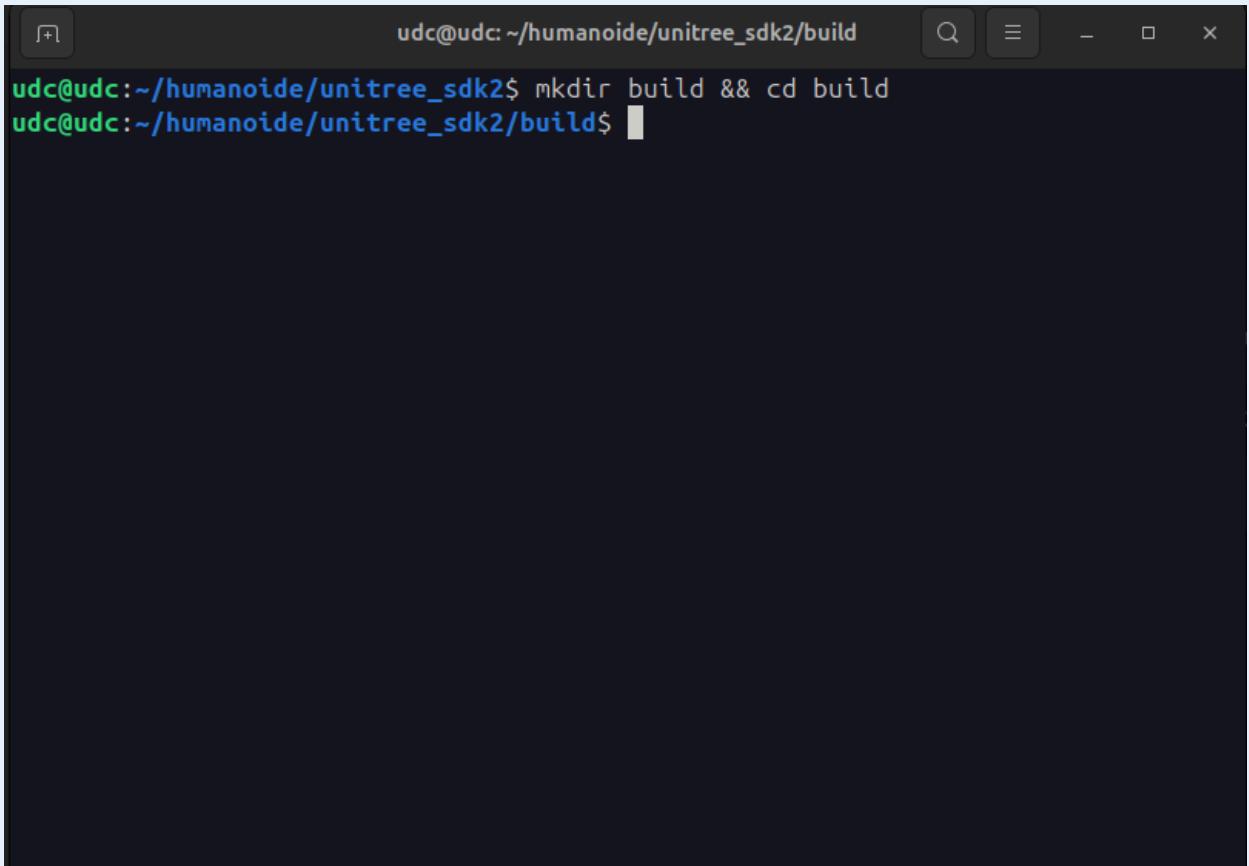
```
cd unitree_sdk2
```

A screenshot of a terminal window titled "udc@udc: ~/humanoide/unitree_sdk2". The window shows the command "cd unitree_sdk2/" being run. The output shows the user's path changing to the "unitree_sdk2" directory. The terminal prompt "udc@udc:~/unitree_sdk2\$" is visible at the bottom.

```
cd unitree_sdk2/
udc@udc:~/unitree_sdk2$
```

- Vamos a crear una carpeta llamada `build` y movernos a ella desde una solar orden:

```
mkdir build && cd build
```



A screenshot of a terminal window titled "udc@udc: ~/humanoide/unitree_sdk2/build". The window shows the command "mkdir build && cd build" being typed and then completed. The terminal has a dark background with light-colored text. The window has standard Linux-style window controls at the top.

- Y para iniciar la construcción usaremos `cmake`, el proximo paso es, en la terminal ingresar el siguiente comando:

```
cmake ..
```

Después ingresamos el siguiente comando:

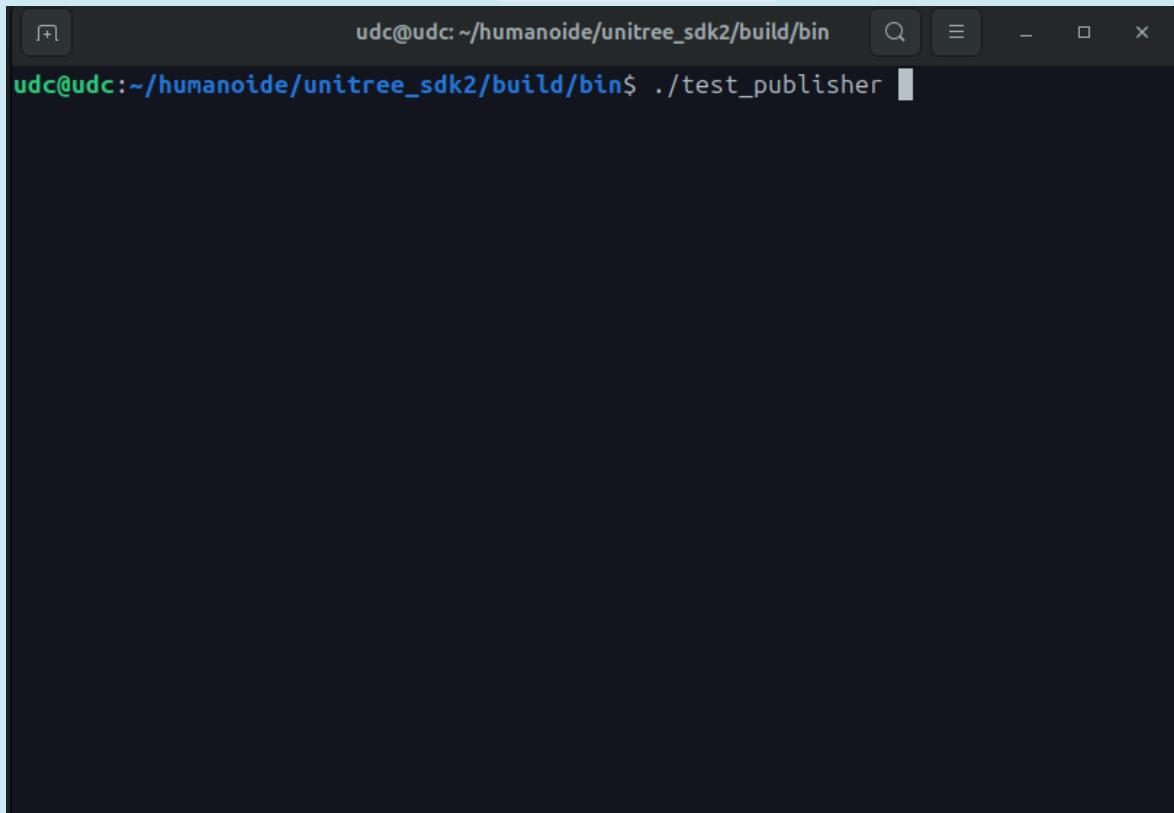
```
sudo make install
```

Para probar si esta correctamente instalado necesitamos dos terminales, en la siguiente ubicación `~/humanoide/unitree_sdk2/build/bin`.

```
cd ~/humanoide/unitree_sdk2/build/bin
```

 Terminal #1

Ejecutaremos un script de prueba `./test_publisher`



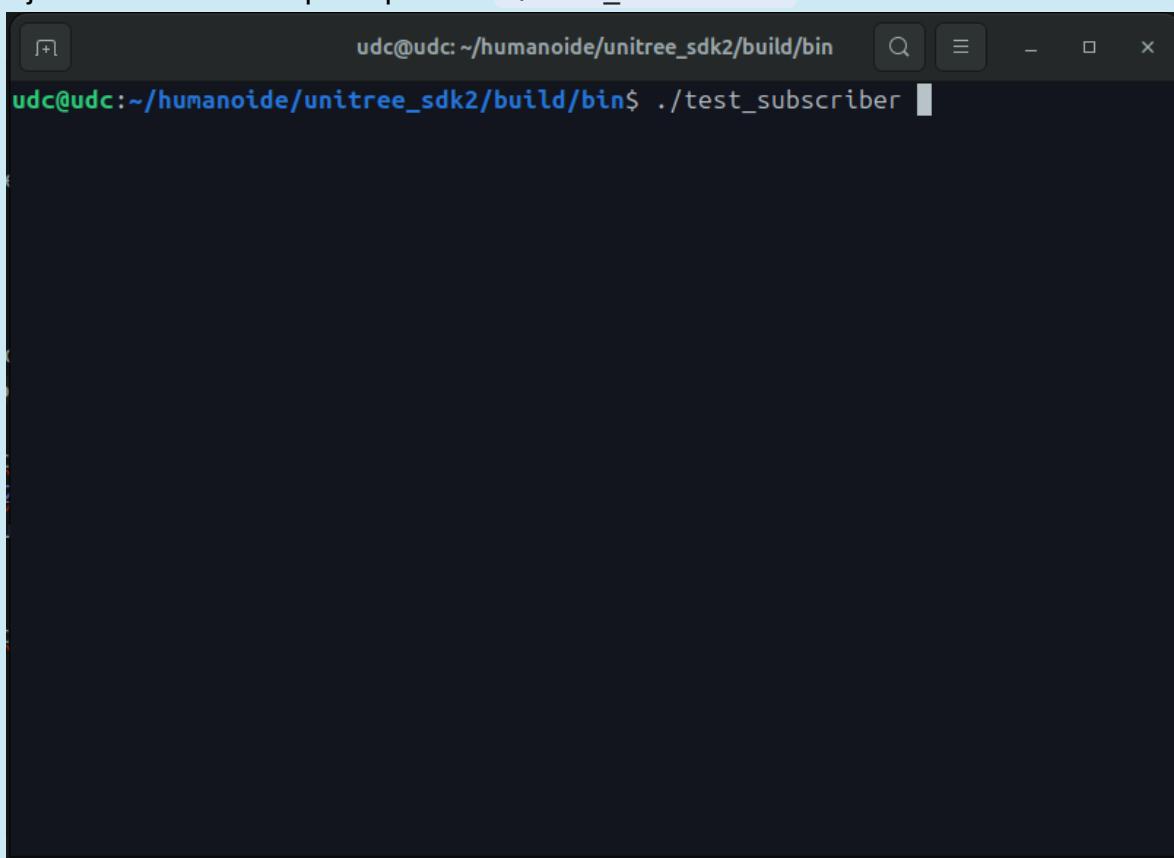
```
udc@udc:~/humanoide/unitree_sdk2/build/bin$ ./test_publisher
```

A screenshot of a terminal window titled "Terminal #1". The window has a dark background and light-colored text. At the top, it shows the path "udc@udc:~/humanoide/unitree_sdk2/build/bin" and the command prompt "\$". Below the title bar, there are several small icons. The main area of the terminal is empty, showing only the command line itself.

- Estara esperando una conexion que la realizaremos desde la segunda terminal

⚡ Terminal #2

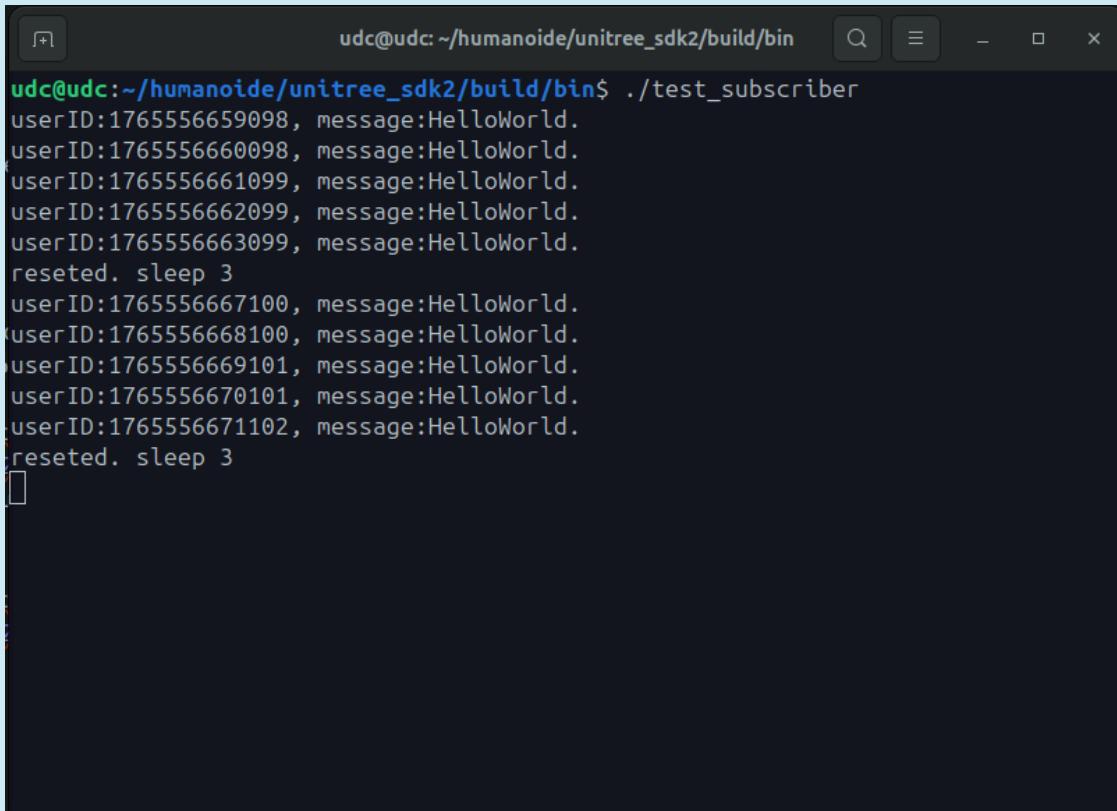
Ejecutaremos un script de prueba `./test_subscriber`



```
udc@udc:~/humanoide/unitree_sdk2/build/bin$ ./test_subscriber
```

A screenshot of a terminal window titled "Terminal #2". The window has a dark background and light-colored text. At the top, it shows the path "udc@udc:~/humanoide/unitree_sdk2/build/bin" and the command prompt "\$". Below the title bar, there are several small icons. The main area of the terminal is empty, showing only the command line itself.

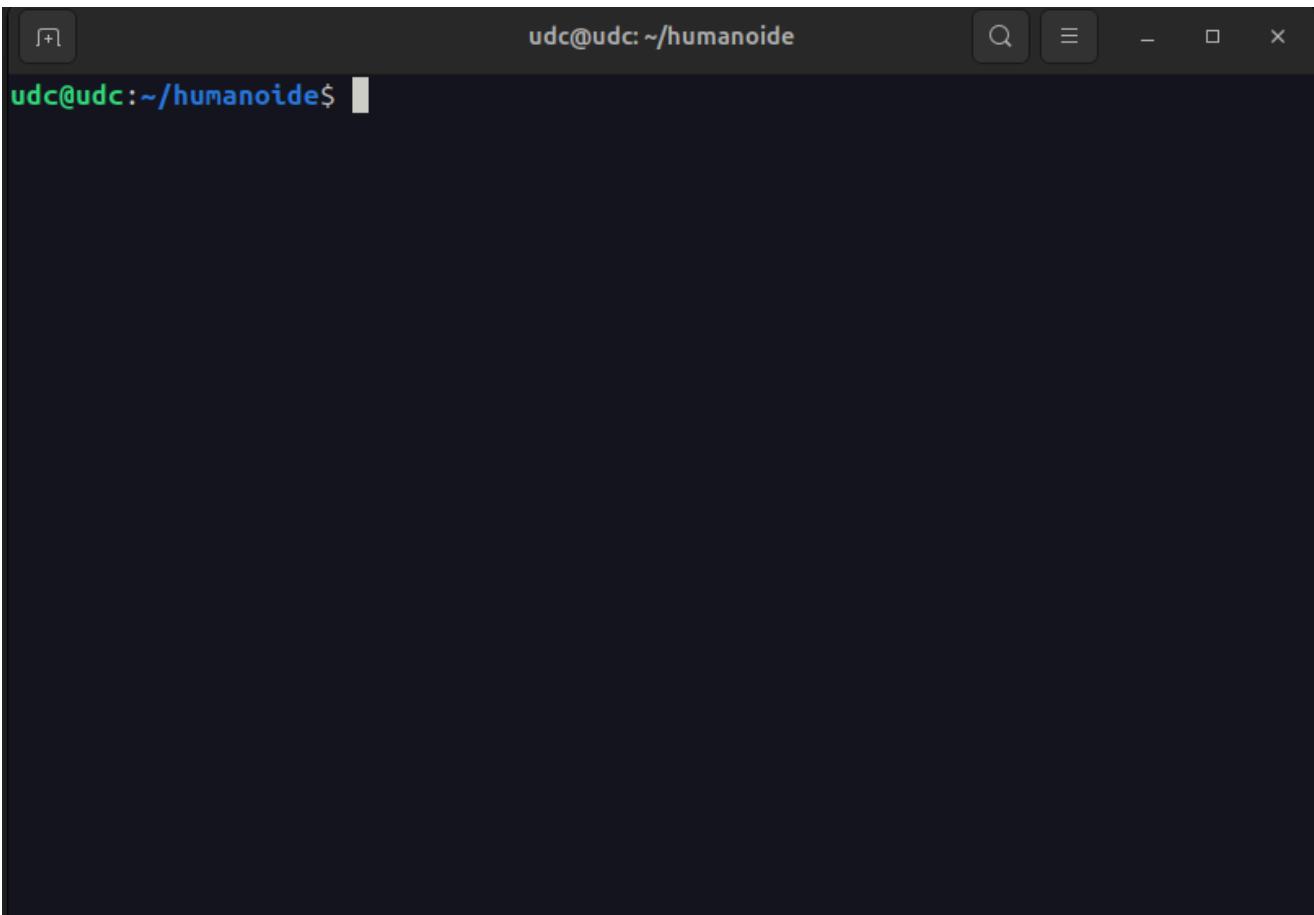
- Resultado de la espera



```
udc@udc:~/humanoide/unitree_sdk2/build/bin$ ./test_subscriber
userID:1765556659098, message>HelloWorld.
userID:1765556660098, message>HelloWorld.
userID:1765556661099, message>HelloWorld.
userID:1765556662099, message>HelloWorld.
userID:1765556663099, message>HelloWorld.
reseted. sleep 3
userID:1765556667100, message>HelloWorld.
userID:1765556668100, message>HelloWorld.
userID:1765556669101, message>HelloWorld.
userID:1765556670101, message>HelloWorld.
userID:1765556671102, message>HelloWorld.
reseted. sleep 3
```

4 - Clonación e instalación de `unitree_sdk2_python`

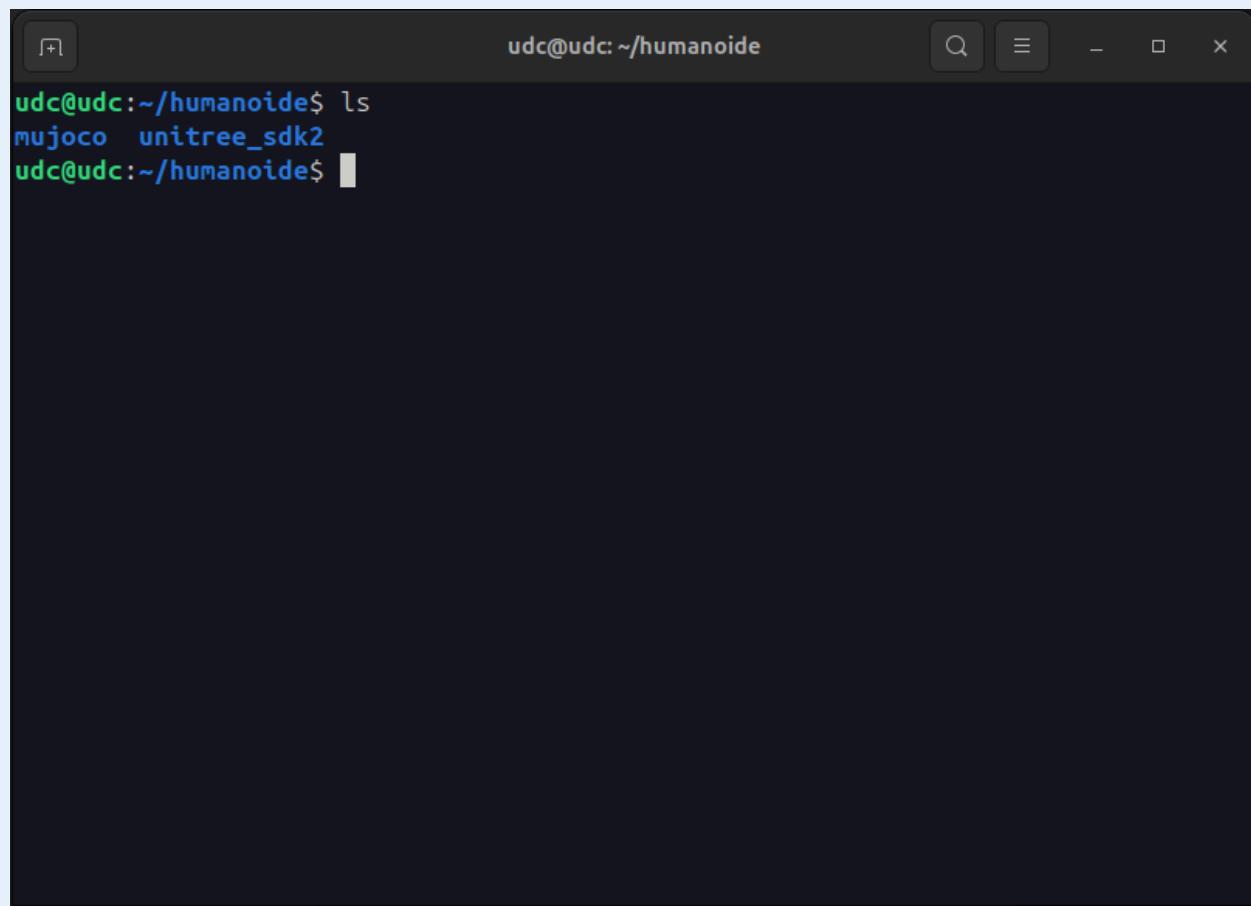
Este recurso debe estar en la carpeta `humanoide` donde clonamos el recurso de `mujoco` - `unitree_sdk2`



A screenshot of a terminal window with a dark theme. The title bar shows the user is at `udc@udc: ~/humanoide`. The command prompt is `udc@udc:~/humanoide$`. The main area of the terminal is entirely black, indicating no output or a blank workspace.

Pasos

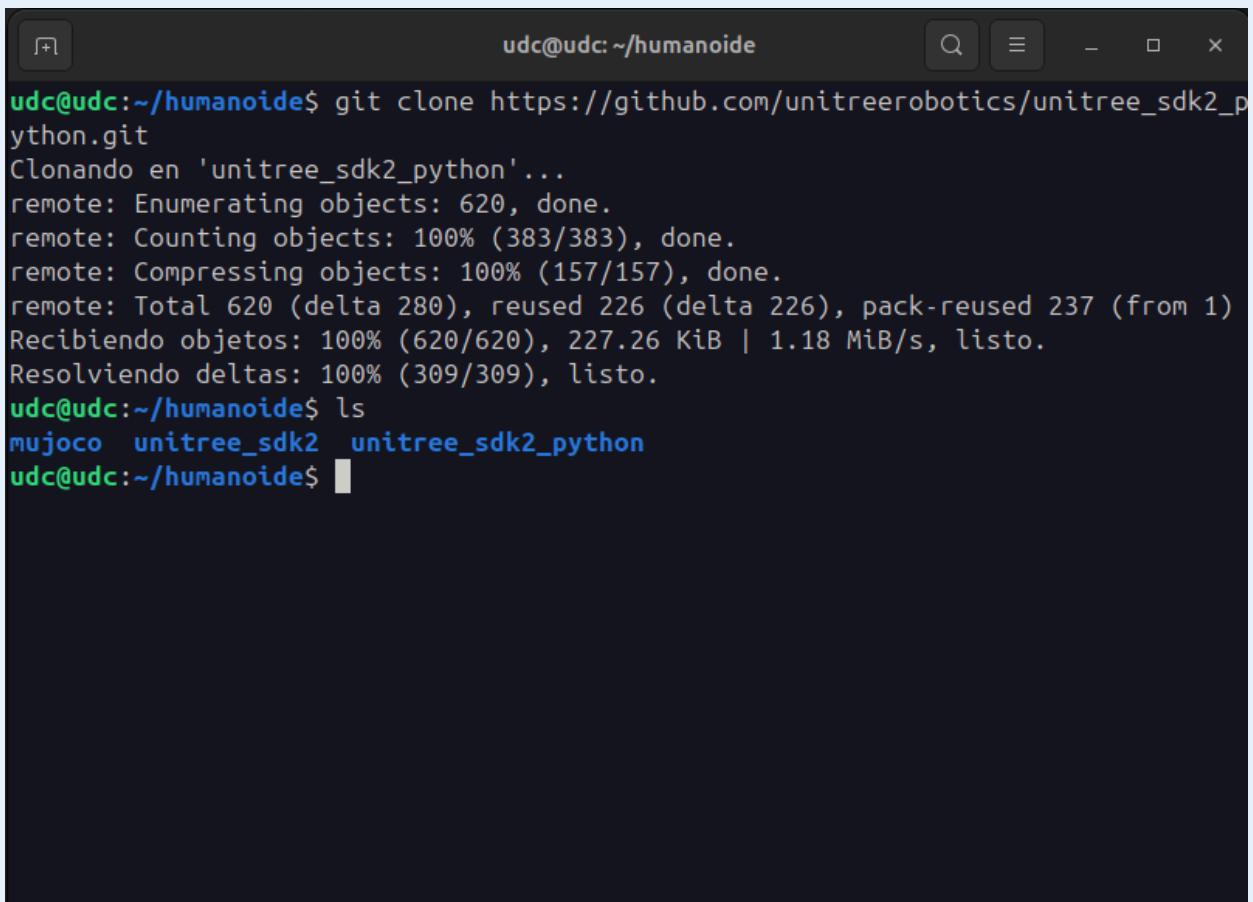
Clonar el recurso de `unitree_sdk2_python`, con `ls` debemos estar visualizando en la carpeta `humanoide` dos carpetas `mujoco` y `unitree_sdk2`.



A screenshot of a terminal window with a dark theme. The title bar shows the user is at `udc@udc: ~/humanoide`. The command prompt is `udc@udc:~/humanoide$`. The user has run the `ls` command, which has returned two entries: `mujoco` and `unitree_sdk2`.

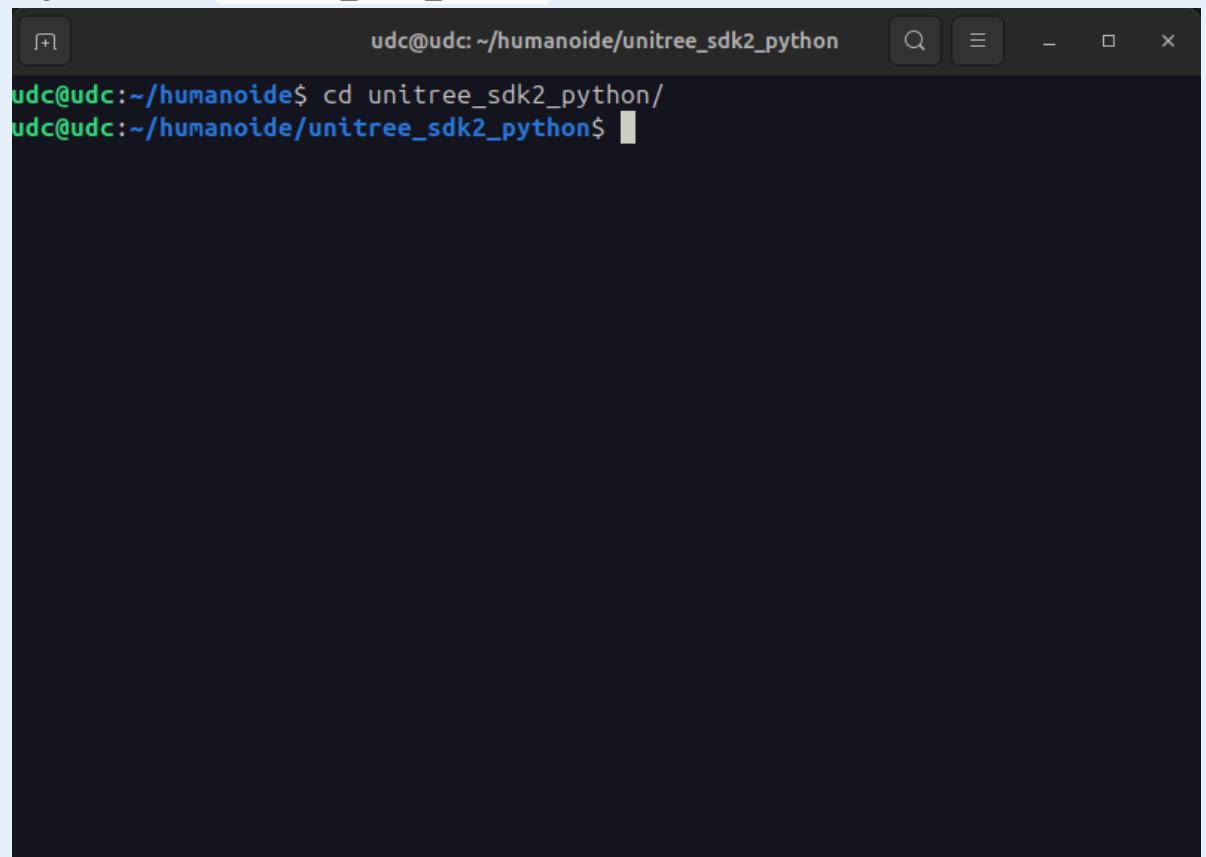
```
udc@udc:~/humanoide$ ls
mujoco  unitree_sdk2
udc@udc:~/humanoide$
```

```
git clone https://github.com/unitreerobotics/unitree_sdk2_python.git
```



```
git clone https://github.com/unitreerobotics/unitree_sdk2_python.git
Clonando en 'unitree_sdk2_python'...
remote: Enumerating objects: 620, done.
remote: Counting objects: 100% (383/383), done.
remote: Compressing objects: 100% (157/157), done.
remote: Total 620 (delta 280), reused 226 (delta 226), pack-reused 237 (from 1)
Recibiendo objetos: 100% (620/620), 227.26 KiB | 1.18 MiB/s, listo.
Resolviendo deltas: 100% (309/309), listo.
ls
mujoco  unitree_sdk2  unitree_sdk2_python
```

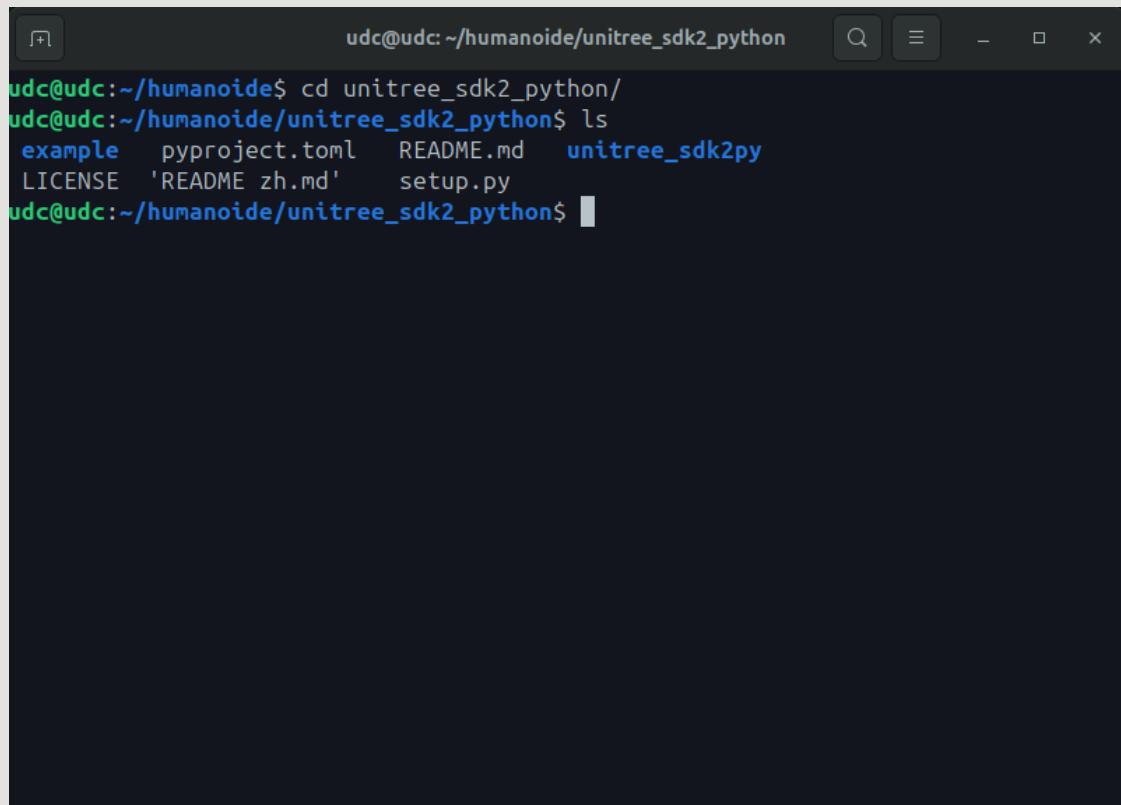
- Ingresamos a `unitree_sdk2_python`



```
cd unitree_sdk2_python
```

⚠️ Tener en cuenta

Ese recurso trabaja con python, utiliza archivo de configuración distinto a los anteriores, en este caso podemos saber cuál es, un `pyproject.toml`

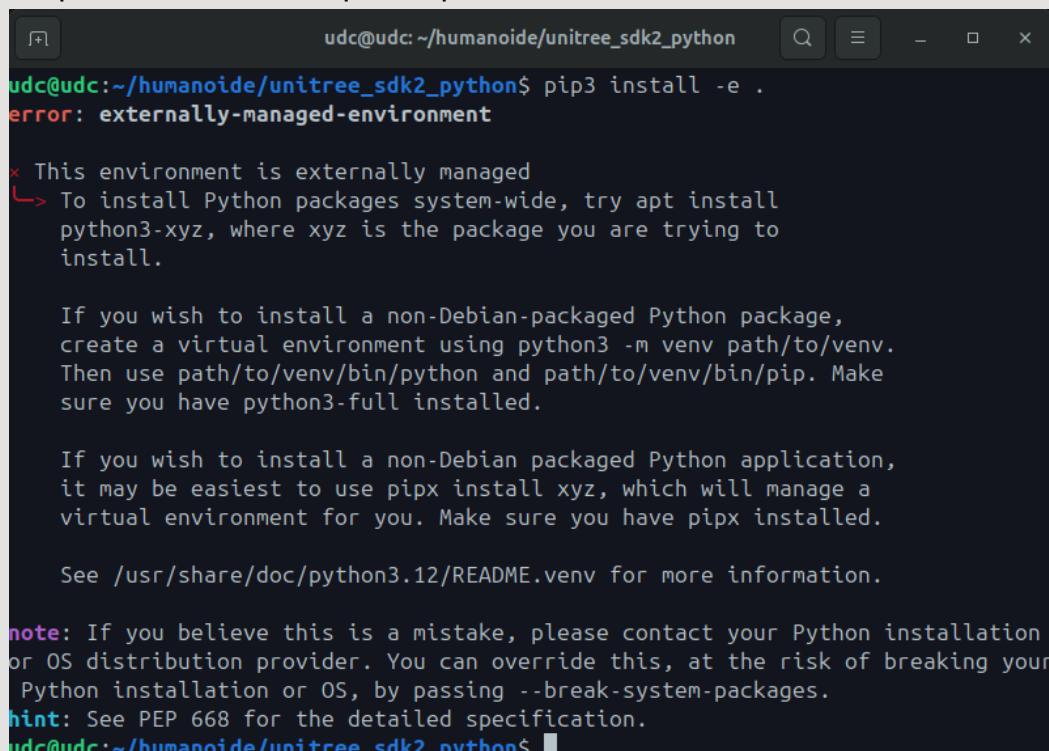


```
udec@udc:~/humanoide/unitree_sdk2_python$ cd unitree_sdk2_python/
udec@udc:~/humanoide/unitree_sdk2_python$ ls
example  pyproject.toml  README.md  unitree_sdk2py
LICENSE  'README zh.md'  setup.py
udec@udc:~/humanoide/unitree_sdk2_python$
```

- Para construir el entorno usaremos el siguiente comando

```
pip3 install -e .
```

- De primer momento se puede presentar un error:



```
udec@udc:~/humanoide/unitree_sdk2_python$ pip3 install -e .
error: externally-managed-environment

  × This environment is externally managed
  ↳ To install Python packages system-wide, try apt install
     python3-xyz, where xyz is the package you are trying to
     install.

  If you wish to install a non-Debian-packaged Python package,
  create a virtual environment using python3 -m venv path/to/venv.
  Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make
  sure you have python3-full installed.

  If you wish to install a non-Debian packaged Python application,
  it may be easiest to use pipx install xyz, which will manage a
  virtual environment for you. Make sure you have pipx installed.

  See /usr/share/doc/python3.12/README.venv for more information.

  note: If you believe this is a mistake, please contact your Python installation
        or OS distribution provider. You can override this, at the risk of breaking your
        Python installation or OS, by passing --break-system-packages.
  hint: See PEP 668 for the detailed specification.
udec@udc:~/humanoide/unitree_sdk2_python$
```

- Para solucionarlo requerimos crear un `environment`, usando un gestor de entornos virtuales para `python`, para ello usar `venv`.

- Nos mantenemos en la siguiente ubicación `~/humanoide/`

```
udc@udc:~/humanoide$
```

- Desde ese punto creamos el entorno, usando el siguiente comando en terminal:

```
python3 -m venv venv
```

- Después activamos el entorno virtual (*OJO, siempre toca activarlo en cada instancia de terminal si queremos trabajar con el código de la carpeta*)
- Activamos en entorno virtual:

```
source venv/bin/activate
```

- Veremos el siguiente cambio:

```
udc@udc:~/humanoide$ source venv/bin/activate  
(venv) udc@udc:~/humanoide$
```

- Vamo a la carpeta de `unitree_sdk2_python` y vamos a usar el comando que nos permite configurar lo necesario para el entorno:

```
udc@udc:~/humanoide/unitree_sdk2_python$  
(venv) udc@udc:~/humanoide$ cd unitree_sdk2_python/  
(venv) udc@udc:~/humanoide/unitree_sdk2_python$
```

```
pip3 install -e .
```

⚠️ Tener en cuenta

Si se dan errores de ejecución o de que la orden no se encuentra, tratar de ejecutar por separado cada paquete que se encuentra al principio del documento.

- Si vuelve el error, nos mostrara lo siguiente

```

  udc@udc: ~/humanoide/unitree_sdk2_python
  Preparing editable metadata (pyproject.toml) ... done
  Collecting cyclonedds==0.10.2 (from unitree_sdk2py==1.0.1)
    Downloading cyclonedds-0.10.2.tar.gz (156 kB)
      156.8/156.8 kB 9.1 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... error
  error: subprocess-exited-with-error

    × Getting requirements to build wheel did not run successfully.
    | exit code: 1
    | [1 lines of output]
    | Could not locate cyclonedds. Try to set CYCLONEDDS_HOME or CMAKE_PREFIX_PA
    TH
    | [end of output]

    note: This error originates from a subprocess, and is likely not a problem wit
    h pip.
  error: subprocess-exited-with-error

    × Getting requirements to build wheel did not run successfully.
    | exit code: 1
    | See above for output.

  note: This error originates from a subprocess, and is likely not a problem with

```

- Mostrando que necesitamos otro recurso, para ello abrimos una nueva terminal, colocaremos lo siguiente y ejecutamos:

```

cd ~
git clone https://github.com/eclipse-cyclonedds/cyclonedds -b
releases/0.10.x
cd cyclonedds && mkdir build install && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=../install
cmake --build . --target install

```

- Regresamo a la terminal donde estamos ubicados en `unitree_sdk2_python` junto al entorno virtual activado, añadimos lo siguiente en la terminal

```

export CYCLONEDDS_HOME="$HOME/cyclonedds/install"
pip3 install -e .

```

```
udc@udc: ~/humanoide/unitree_sdk2_python
udc@udc: ~/humanoide/unitree_sdk2_python
(venv) udc@udc:~/humanoide/unitree_sdk2_python$ export CYCLONEDDS_HOME="$HOME/cyclonedds/install"
pip3 install -e .
Obtaining file:///home/udc/humanoid/unitree_sdk2_python
  Installing build dependencies ... done
  Checking if build backend supports build_editable ... done
  Getting requirements to build editable ... done
    Preparing editable metadata (pyproject.toml) ... done
Requirement already satisfied: cyclonedds==0.10.2 in ./venv/lib/python3.12/site-packages (from unitree_sdk2py==1.0.1) (0.10.2)
Requirement already satisfied: numpy in ./venv/lib/python3.12/site-packages (from unitree_sdk2py==1.0.1) (2.2.6)
Requirement already satisfied: opencv-python in ./venv/lib/python3.12/site-packages (from unitree_sdk2py==1.0.1) (4.12.0.88)
Requirement already satisfied: rich-click in ./venv/lib/python3.12/site-packages (from cyclonedds==0.10.2->unitree_sdk2py==1.0.1) (1.9.4)
Requirement already satisfied: click>=8 in ./venv/lib/python3.12/site-packages (from rich-click->cyclonedds==0.10.2->unitree_sdk2py==1.0.1) (8.3.1)
Requirement already satisfied: rich>=12 in ./venv/lib/python3.12/site-packages (from rich-click->cyclonedds==0.10.2->unitree_sdk2py==1.0.1) (14.2.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in ./venv/lib/python3.12/site-packages (from rich>=12->rich-click->cyclonedds==0.10.2->unitree_sdk2py==1.0.1) (4.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in ./venv/lib/python3.12/site-packages (from rich>=12->rich-click->cyclonedds==0.10.2->unitree_sdk2py==1.0.1) (2.19.2)
Requirement already satisfied: mdurl~0.1 in ./venv/lib/python3.12/site-packages (from markdown-it-py>=2.2.0->rich>=12->rich-click->cyclonedds==0.10.2->unitree_sdk2py==1.0.1) (0.1.2)
Building wheels for collected packages: unitree_sdk2py
  Building editable for unitree_sdk2py (pyproject.toml) ... done
  Created wheel for unitree_sdk2py: filename=unitree_sdk2py-1.0.1-editable-py3-none-any.whl size=570 sha256=ee881b4450e4cbcc74e251e27f3452d0a18a5ca6678b9957c65f3886fcfd7c1d1
  Stored in directory: /tmp/pip-ephem-wheel-cache-tut13bfv/wheels/d0/6a/c5/46cdfa1d816e598db35e4a210e0db0a5cf928a1692fcac29
Successfully built unitree_sdk2py
Installing collected packages: unitree_sdk2py
  Attempting uninstall: unitree_sdk2py
    Found existing installation: unitree_sdk2py 1.0.1
    Uninstalling unitree_sdk2py-1.0.1:
      Successfully uninstalled unitree_sdk2py-1.0.1
Successfully installed unitree_sdk2py-1.0.1
(venv) udc@udc:~/humanoide/unitree_sdk2_python$
```

- Se instalo lo necesario correctamente.

5 - Clonación e instalación de unitree_mujoco

Regresamos a la carpeta `humanoide` y volvemos a la clonacion e instalación del ultimo recurso.

Pasos

- Clonar el recurso

```
git clone https://github.com/unitreerobotics/unitree_mujoco.git
```

- Ingresamos al recurso

```
udec@udc:~/humanoide/unitree_mujoco$ git clone https://github.com/unitreerobotics/unitree_mujoco.git
Clonando en 'unitree_mujoco'...
remote: Enumerating objects: 729, done.
remote: Counting objects: 100% (226/226), done.
remote: Compressing objects: 100% (108/108), done.
remote: Total 729 (delta 160), reused 118 (delta 118), pack-reused 503 (from 2)
Recibiendo objetos: 100% (729/729), 62.57 MiB | 8.69 MiB/s, listo.
Resolviendo deltas: 100% (264/264), listo.
udec@udc:~/humanoide/unitree_mujoco$
```

- Nos movemos a la carpeta `simulate`

```
udec@udc:~/humanoide/unitree_mujoco$ cd simulate
udec@udc:~/humanoide/unitree_mujoco/simulate$
```

- Vamos a crear una carpeta llamada `build` y movernos a ella desde una sola orden:

```
mkdir build && cd build
```

```
udec@udc:~/humanoide/unitree_mujoco$ cd simulate
udec@udc:~/humanoide/unitree_mujoco/simulate$ mkdir build && cd build
udec@udc:~/humanoide/unitree_mujoco/simulate/build$
```

- Y para iniciar la construcción usaremos `cmake`, el proximo paso es, en la terminal ingresar el siguiente comando:

```
cmake ..
```

```
udc@udc:~/humanoide/unitree_mujoco/simulate/build$ cmake ..  
-- The C compiler identification is GNU 13.3.0  
-- The CXX compiler identification is GNU 13.3.0  
-- Detecting C compiler ABI info  
-- Detecting C compiler ABI info - done  
-- Check for working C compiler: /usr/bin/cc - skipped  
-- Detecting C compile features  
-- Detecting C compile features - done  
-- Detecting CXX compiler ABI info  
-- Detecting CXX compiler ABI info - done  
-- Check for working CXX compiler: /usr/bin/c++ - skipped  
-- Detecting CXX compile features  
-- Detecting CXX compile features - done  
Release mode  
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD  
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success  
-- Found Threads: TRUE  
-- Found Boost: /usr/lib/x86_64-linux-gnu/cmake/Boost-1.83.0/BoostConfig.cmake (found version "1.83.0") found components: program_options  
-- Configuring done (0.3s)  
-- Generating done (0.0s)  
-- Build files have been written to: /home/udc/humanoide/unitree_mujoco/simulate/build  
udc@udc:~/humanoide/unitree_mujoco/simulate/build$
```

Después ingresamos el siguiente comando:

```
make
```

- En el siguiente paso, visualizaremos un error el cuál la solución la explicare en los siguientes pasos.

```
udc@udc:~/humanoide/unitree_mujoco/simulate/build$ make  
[ 14%] Building CXX object CMakeFiles/unitree_mujoco.dir/src/main.cc.o  
/home/udc/humanoide/unitree_mujoco/simulate/src/main.cc:17:10: fatal error: glfw_adapter.h: No existe el archivo o el directorio  
17 | #include "glfw_adapter.h"  
| ^~~~~~  
compilation terminated.  
make[2]: *** [CMakeFiles/unitree_mujoco.dir/build.make:104: CMakeFiles/unitree_mujoco.dir/src/main.cc.o] Error 1  
make[1]: *** [CMakeFiles/Makefile2:85: CMakeFiles/unitree_mujoco.dir/all] Error 2  
make: *** [Makefile:91: all] Error 2  
udc@udc:~/humanoide/unitree_mujoco/simulate/build$
```

- Para la solución debemos colocar la ruta absoluta de los archivos u carpetas, *¿Cuál es la ruta absoluta?*

```
# Colocarlo en la sección de include_directories  
/home/udc/humanoide/mujoco/simulate  
/home/udc/humanoide/mujoco/include
```

```
# Colocarlo en la sección de link_libraries  
/home/udc/humanoide/mujoco/build/lib/libsimulate.a
```

```
# Comentamos la ultima linea del archivo CMakeLists.txt con el "#" al inicio de la linea.
```

```
add_executable(jstest src/joystick/jstest.cc  
src/joystick/joystick.cc)
```

```
#add_executable(jstest src/joystick/jstest.cc src/joystick/joystick.cc)
```

- Ahora en la mitad del código, vamos a actualizarlo a las rutas absolutas
Sin modificación

```
file(GLOB SIM_SRC  
src/joystick/joystick.cc  
  
# exclude mujoco/simulate/main.cc  
mujoco/simulate/glfw_*.cc  
mujoco/simulate/platform_*.cc  
mujoco/simulate/simulate.cc  
src/lodepng/lodepng.cpp  
)
```

Modificado

```
file(GLOB SIM_SRC  
src/joystick/joystick.cc  
  
# exclude mujoco/simulate/main.cc  
~/humanoide/mujoco/simulate/glfw_*.cc  
~/humanoide/mujoco/simulate/platform_*.cc  
~/humanoide/mujoco/simulate/simulate.cc  
src/lodepng/lodepng.cpp  
)
```

-
- Abrimos el archivo `CMakeList.txt` con un editor de texto (*VSCode, Vim, Nano*), que está ubicado en `/home/udc/humanoide/unitree_mujoco/simulate`

```
nano /home/udc/humanoide/unitree_mujoco/simulate/CMakeLists.txt
```

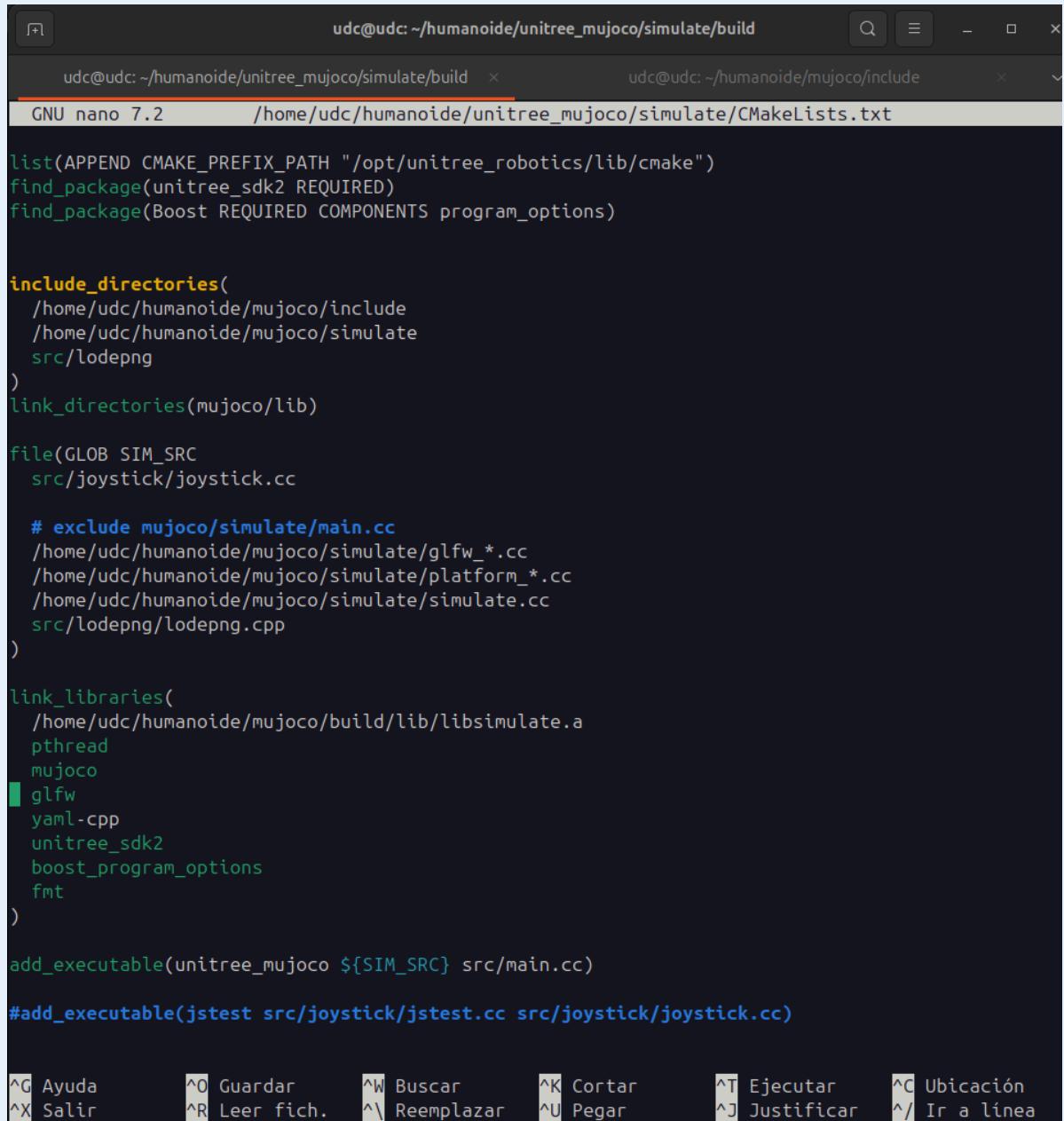
O

```
nano ~/humanoide/unitree_mujoco/simulate/CMakeLists.txt
```

- O si está en la terminal en la siguiente dirección `~/humanoide/mujoco/simulate/`, podemos hacer lo siguiente:

```
nano CMakelists.txt
```

- Modificamos el archivo como se ve en la imagen



```

udc@udc: ~/humanoide/unitree_mujoco/simulate/build
udc@udc: ~/humanoide/unitree_mujoco/simulate/build
GNU nano 7.2      /home/udc/humanoide/unitree_mujoco/simulate/CMakeLists.txt

list(APPEND CMAKE_PREFIX_PATH "/opt/unitree_robotics/lib/cmake")
find_package(unitree_sdk2 REQUIRED)
find_package(Boost REQUIRED COMPONENTS program_options)

include_directories(
    /home/udc/humanoide/mujoco/include
    /home/udc/humanoide/mujoco/simulate
    ${src}/lodepng
)
link_directories(mujoco/lib)

file(GLOB SIM_SRC
    ${src}/joystick/joystick.cc

    # exclude mujoco/simulate/main.cc
    /home/udc/humanoide/mujoco/simulate/glfw_*.cc
    /home/udc/humanoide/mujoco/simulate/platform_*.cc
    /home/udc/humanoide/mujoco/simulate/simulate.cc
    ${src}/lodepng/lodepng.cpp
)

link_libraries(
    /home/udc/humanoide/mujoco/build/lib/libsimulate.a
    pthread
    mujoco
    glfw
    yaml-cpp
    unitree_sdk2
    boost_program_options
    fmt
)
add_executable(unitree_mujoco ${SIM_SRC} ${src}/main.cc)

#add_executable(jstest ${src}/joystick/jstest.cc ${src}/joystick/joystick.cc)

^G Ayuda          ^O Guardar          ^W Buscar          ^K Cortar          ^T Ejecutar          ^C Ubicación
^X Salir          ^R Leer fich.        ^\ Reemplazar       ^U Pegar           ^J Justificar       ^/ Ir a línea

```

Guardamos con la tecla **CTRL + 0** y salimos con **CTRL + X**

- Volvemos a ejecutar:

make

```
udec@udec: ~/humanoide/unitree_mujoco/simulate/build$ make
[ 12%] Building CXX object CMakeFiles/unitree_mujoco.dir/home/udec/humanoide/mujoco/simulate/glfw_adapter.cc.o
[ 25%] Building CXX object CMakeFiles/unitree_mujoco.dir/home/udec/humanoide/mujoco/simulate/glfw_dispatch.cc.o
[ 37%] Building CXX object CMakeFiles/unitree_mujoco.dir/home/udec/humanoide/mujoco/simulate/platform_ui_adapter.cc.o
[ 50%] Building CXX object CMakeFiles/unitree_mujoco.dir/home/udec/humanoide/mujoco/simulate/simulate.cc.o
[ 62%] Building CXX object CMakeFiles/unitree_mujoco.dir/src/joystick/joystick.cc.o
[ 75%] Building CXX object CMakeFiles/unitree_mujoco.dir/src/lodepng/lodepng.cpp.o
[ 87%] Building CXX object CMakeFiles/unitree_mujoco.dir/src/main.cc.o
[100%] Linking CXX executable unitree_mujoco
[100%] Built target unitree_mujoco
udec@udec:~/humanoide/unitree_mujoco/simulate/build$
```

6 - Funcionalidad del entorno

Para esta funcionalidad debes contar con dos instancias de terminal

🔗 Terminal #1

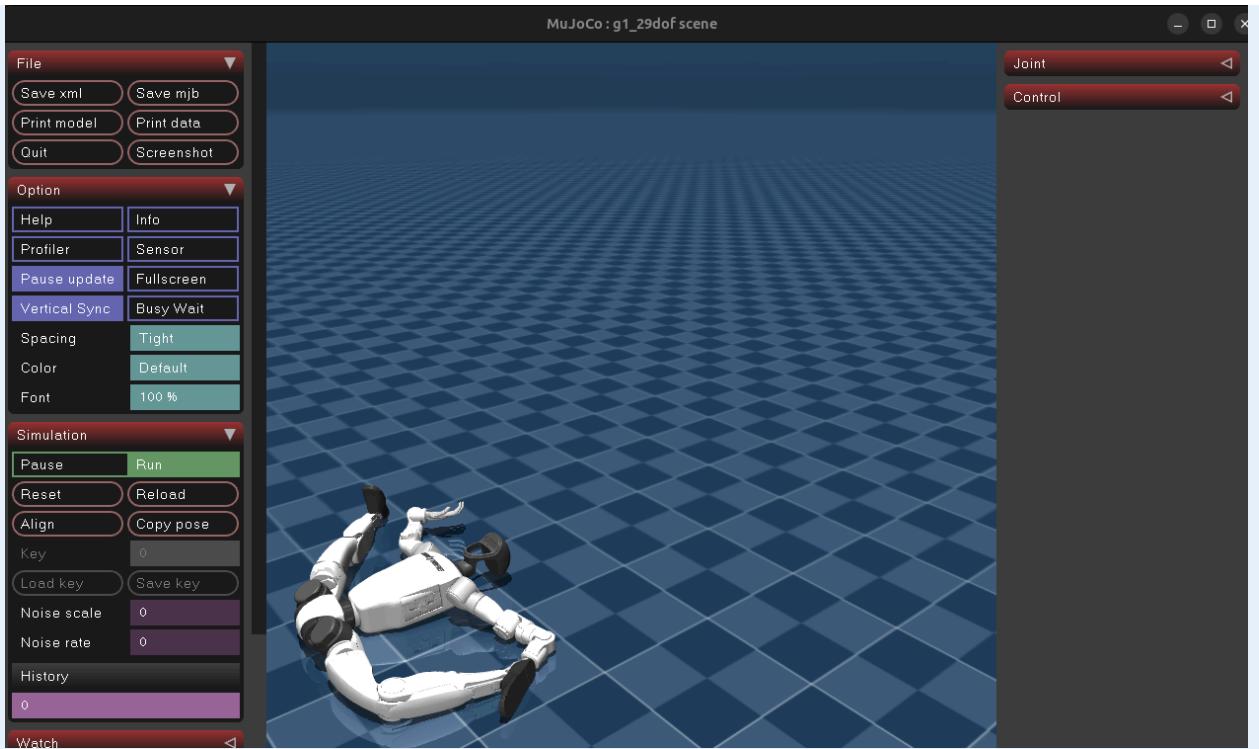
La primera terminal se debe ubicar en la siguiente ruta

```
~/humanoide/unitree_mujoco/simulate/build
```

Ya que vamos a trabajar con el `unitree-g1` vamos a lanzar una escena de simulación con el robot necesario, en la terminal vamos a ingresar el siguiente comando

```
./unitree_mujoco -r g1 -s scene_29dof.xml
```

Visualizaremos el robot, puedes cancelar la visualización desde mujoco buscando la opción `Quit`, o desde la terminal con `CTRL + C`.



⚠ Importante podemos tambien simular en la vida real un arnes para el robot...

- Para ello necesitamos modificar el siguiente archivo, lo abriremos con nano en este ejemplo para modificar rapidamente.

```
nano ~/humanoide/unitree_mujoco/simulate/config.yaml
```

```
GNU nano 7.2          /home/udc/humanoide/unitree_mujoco/simulate/config.yaml
robot: "go2" # Robot name, "go2", "b2", "b2w", "h1", "go2w", "g1"
robot_scene: "scene.xml" # Robot scene, /unitree_robots/[robot]/scene.xml

domain_id: 1 # Domain id
interface: "lo" # Interface

use_joystick: 0 # Simulate Unitree WirelessController using a gamepad
joystick_type: "xbox" # support "xbox" and "switch" gamepad layout
joystick_device: "/dev/input/js0" # Device path
joystick_bits: 16 # Some game controllers may only have 8-bit accuracy

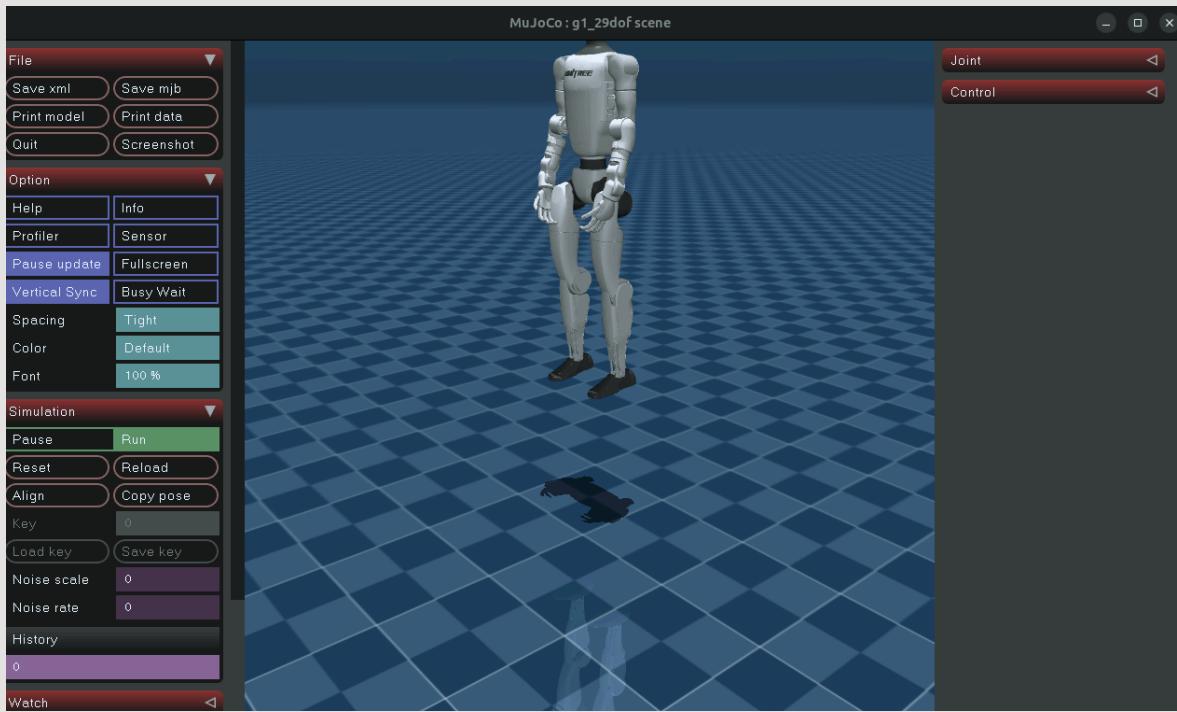
print_scene_information: 1 # Print link, joint and sensors information of robot

enable_elastic_band: 0 # Virtual spring band, used for lifting h1
```

Vamos a modificar una linea, especificamente un valor, en la ultima linea visualizamos un `0`, lo que significa que no simula una banda elastica, vamos a modificarlo y pasarlo a `1`, que nos permitira que el robot al lanzar la escena u simulacion aparezca colgado.

```
enable_elastic_band: 1 # Virtual spring band, used for lifting h1
```

Volvemos a lanzar la escena u simulación



Podemos manipular la altura de la banda para que el robot se eleve con la banda elastica o descender.

- Para subir, presionar las teclas 7 o
- Para bajar, presionar las teclas 8 o
- Para soltarlo de la banda, presionar 9 o Delete/Backspace .

Importante que este la primera instancia de terminal mostrando la simulación, ya que la segunda instancia de terminal envía las instrucciones

Terminal #2

La segunda instanciacion de terminal se debe ubicar en la siguiente ubicación
~/humanoide/unitre_mujoco/simulate_python/test

Tener en cuenta

Tener la primera instancia de terminal como se describio anteriormente, debe estar activo.

```
udc@udc:~/humanoide/unitree_mujoco/simulate_python/test$
```

Ahora vamos a lanzar las instrucciones desde la segunda terminal, para lanzar las instrucciones necesitamos activar el entorno creado con venv

- Activamos el entorno que habíamos creado, nos movemos a la carpeta humanoide

```
udc@udc:~/humanoide$
```

- Activamos el entorno

```
source venv/bin/activate
```

- Visualizamos que el entorno se activo

```
udc@udc:~/humanoide$ source venv/bin/activate
(venv) udc@udc:~/humanoide$
```

- Y nos ubicamos nuevamente en la carpeta
~/humanoide/unitree_mujoco/simulate_python/test

```
cd ~/humanoide/unitree_mujoco/simulate_python/test
```

```
udc@udc:~/humanoide/unitree_mujoco/simulate_python/test$ cd ~/humanoide/unitree_mujoco/simulate_python/test
(venv) udc@udc:~/humanoide/unitree_mujoco/simulate_python/test$
```

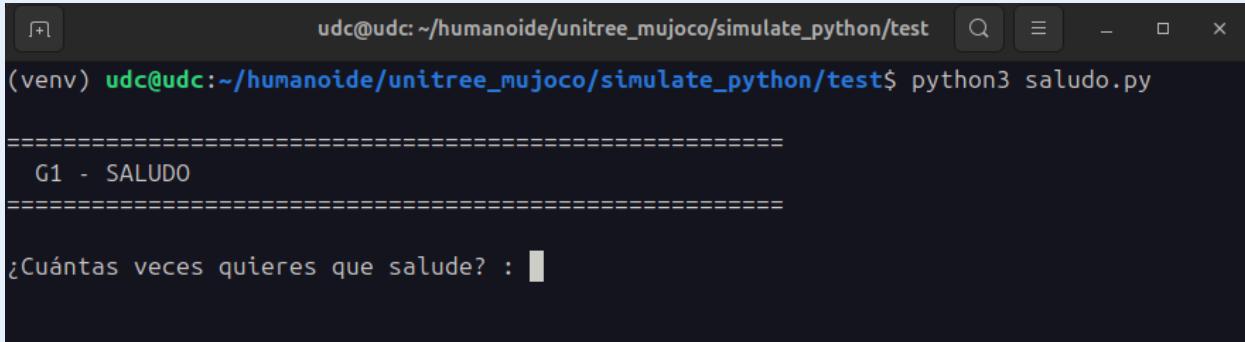
⚠ Creamos un archivo, para el ejemplo `saludo.py` dentro de la carpeta `test`, y le añadimos el código que se encuentra al final de la documentación

- Visualizo el contenido de la carpeta

```
1: udc@udc: ~/humanoide/unitree_mujoco/simulate_python/test ~
udc@udc:~/humanoide/unitree_mujoco/simulate_python/test$ ls
gamepad_test.py  saludo.py  test_unitree_sdk2.py
udc@udc:~/humanoide/unitree_mujoco/simulate_python/test$
```

- Y ejecutamo de la siguiente manera

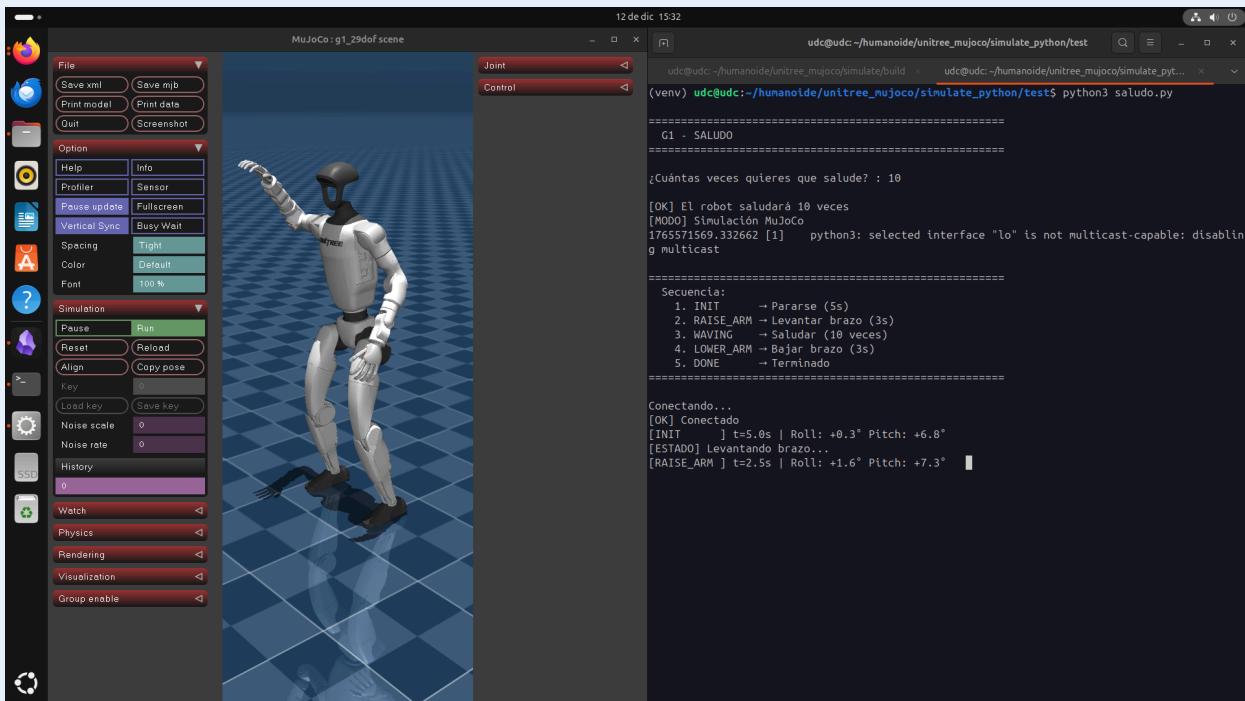
```
python3 saludo.py
```



```
udec@udc: ~/humanoide/unitree_mujoco/simulate_python/test
(venv) udecd@udc:~/humanoide/unitree_mujoco/simulate_python/test$ python3 saludo.py
=====
G1 - SALUDO
=====
¿Cuántas veces quieras que salude? : |
```

💡 Uso

Para el uso correcto, debe verse en ejecución de esta manera:



Enlace para obtención de códigos

En el siguiente enlace se compartirá donde están los códigos de prueba, para que se puedan basar.

- [Código saludo.py](#)
- [Repositorio](#)

