Requisistos previos

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

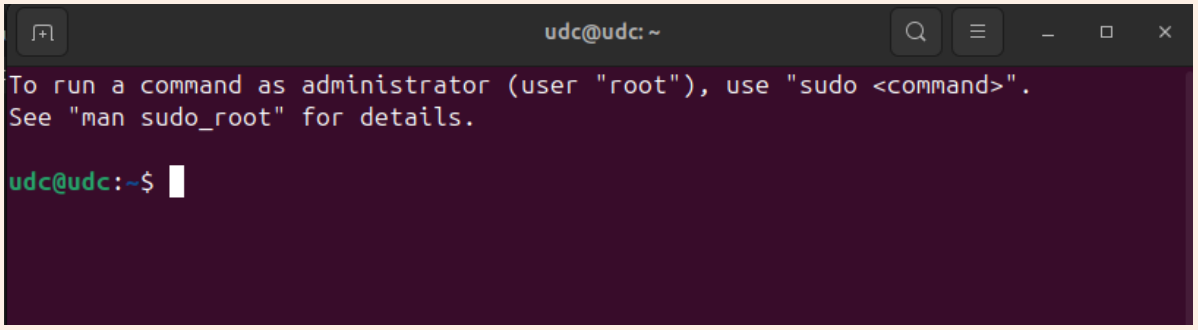
Requerimientos minimos

- 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:

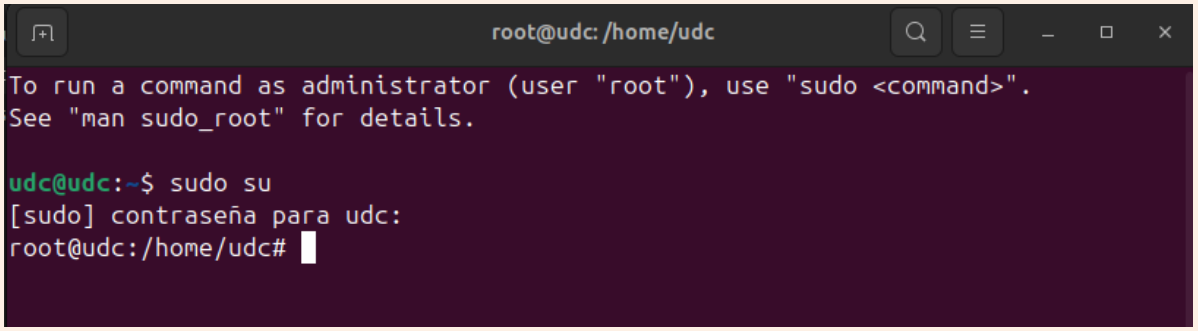
- Abrir una terminal, se puede abrir una terminal con la siguiente terminal `CTRL + ALT + T`.
- Y se abrira una terminal



- Que pedira acceder como `Super usuario/su`

```
sudo su
```

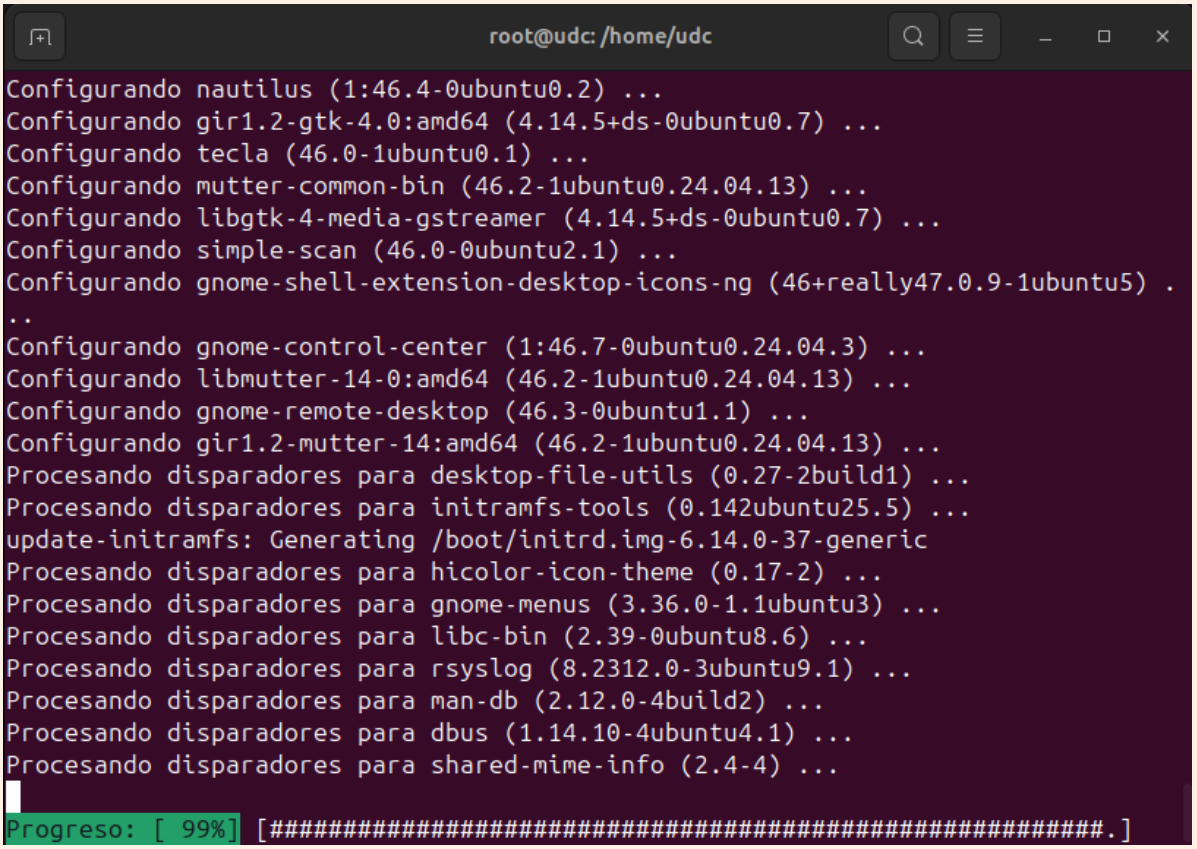
- Se cambia la visión de la terminal



- 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



- 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 contruccion e instalación de dependencias de los mismos

---

2. Clonación Mujoco y construcción

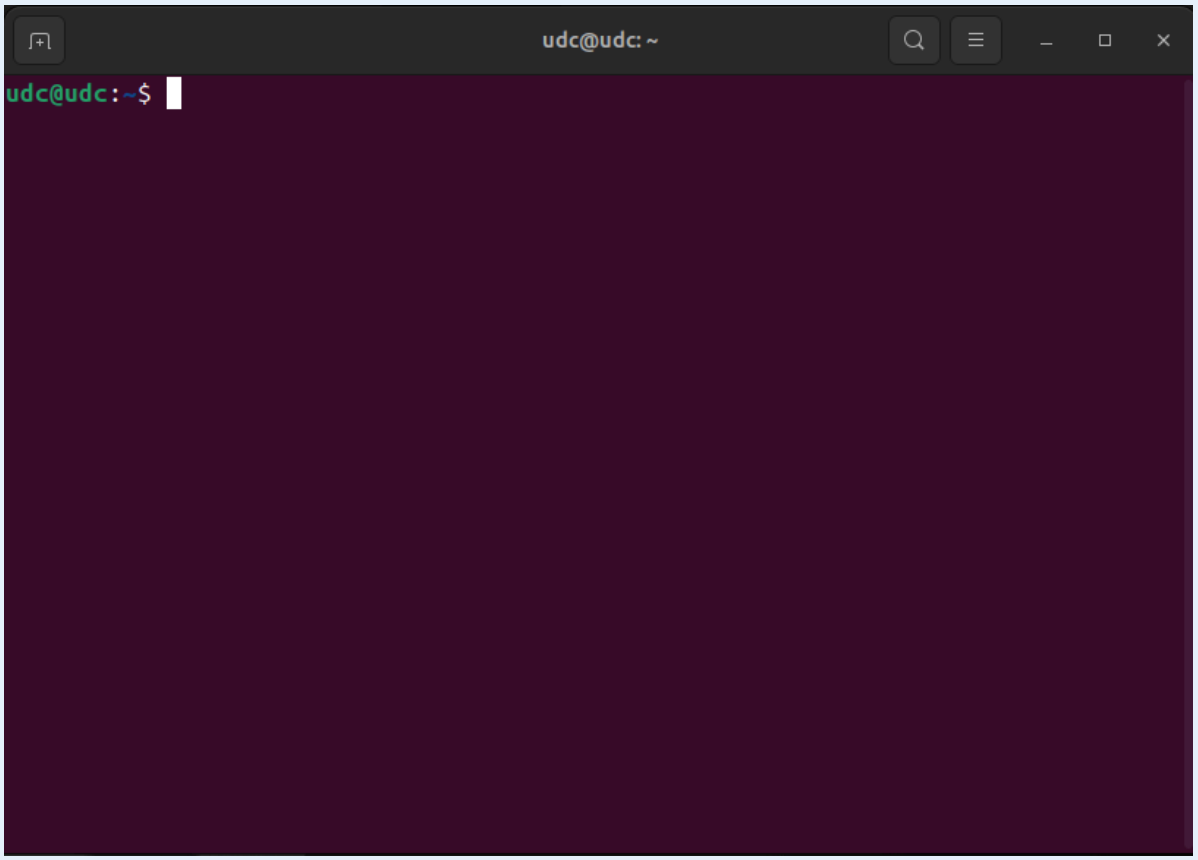
Implementación de los entornos que constituyen el desarrollo para la programación y pruebas, para el robot `Unitree-G1`.

⚠ Tener en cuenta!!

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

Pasos a realizar para la carpeta donde ubicaremos nuestros recursos

- Abrimos una terminal



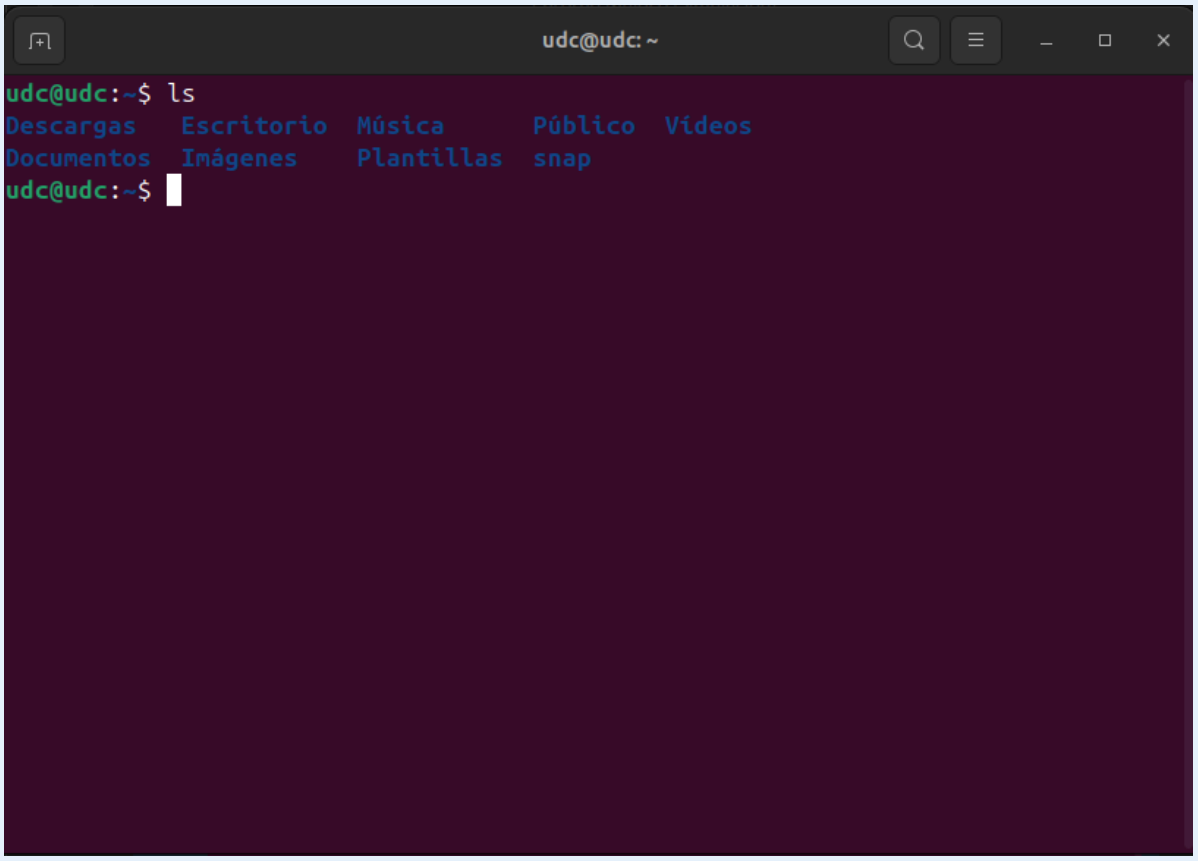
- Ingresamos el siguiente comando para identificar las carpetas existentes en la actual ubicación del terminal

```
ls
```

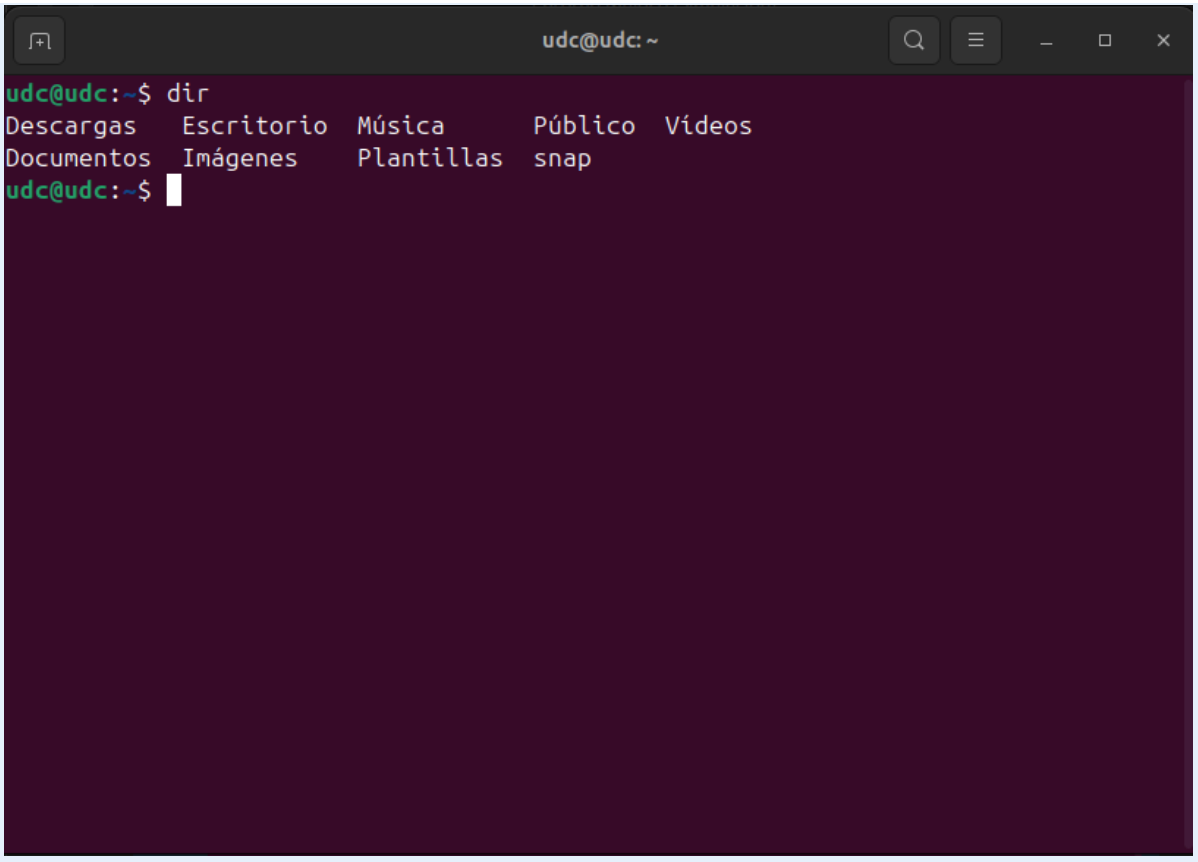
O

```
dir
```

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



- Ahora mostrare como se ve usando `dir`:



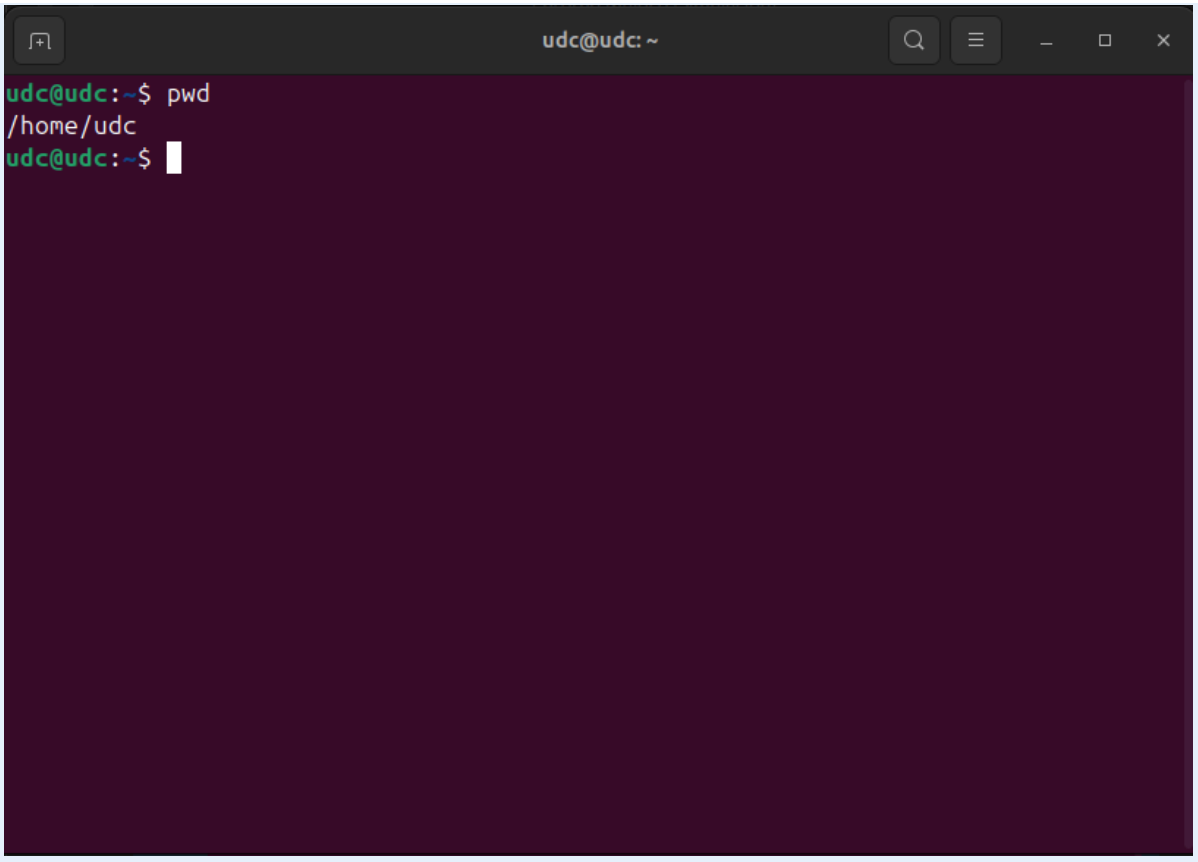
- Los anteriores comando dan la misma salida, pero visualmente diferente.

- 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
```

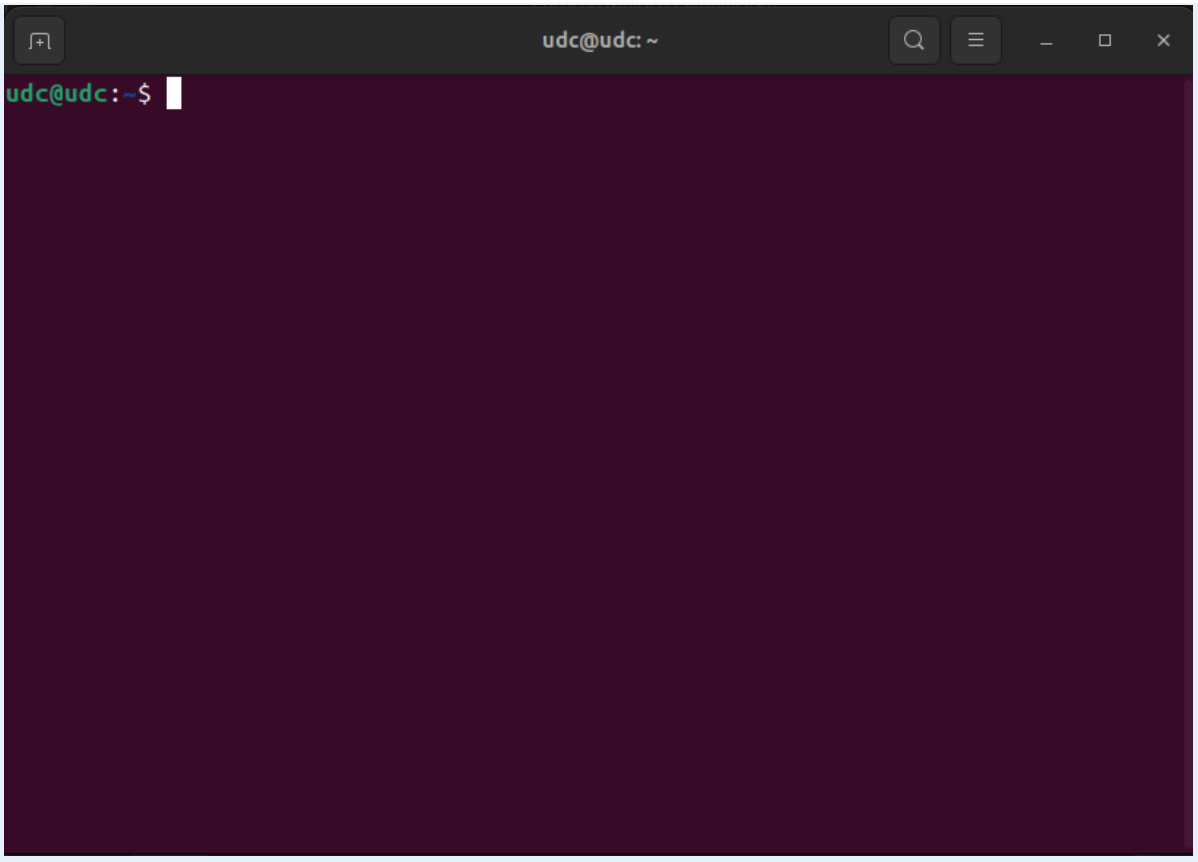
- Nos dara la siguiente salida:



- 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:



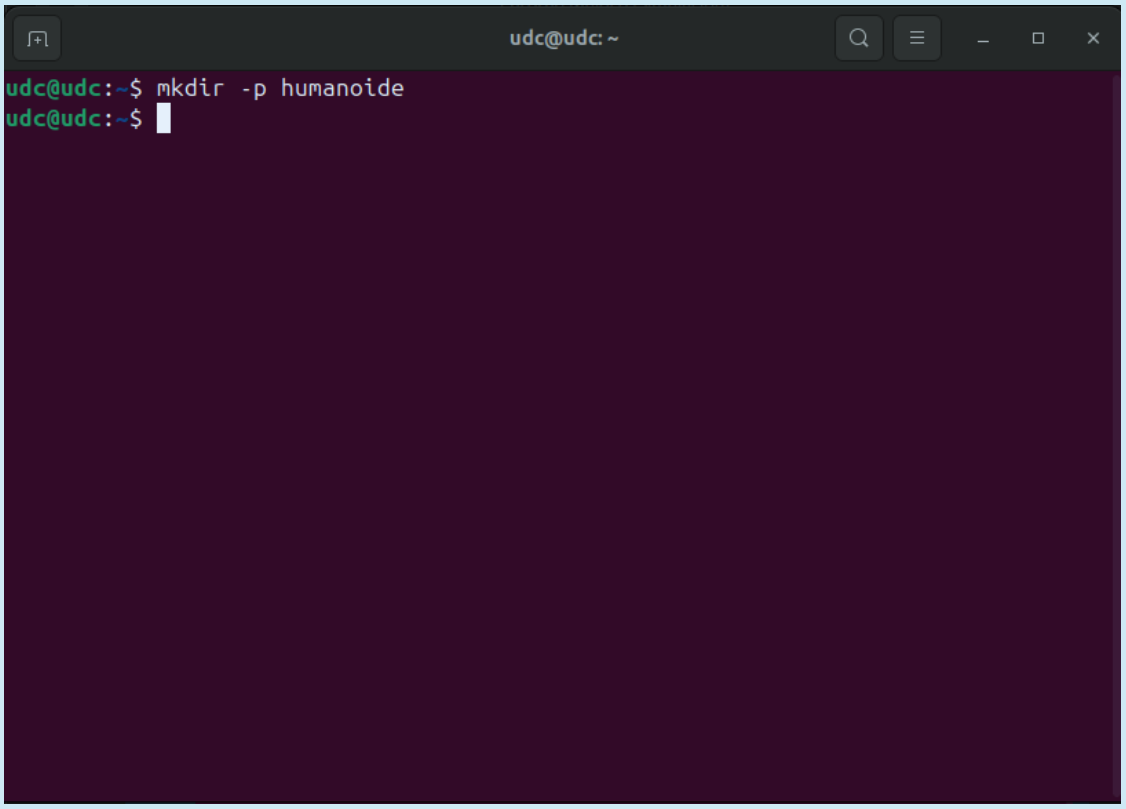
- Para el ejemplo usare el siguiente nombre para la carpeta principal que contendra los recursos `humanoide`.
- Ingresaremos en la terminal el siguiente comando junto al nombre de la carpeta:

```
mkdir -p humanoide
```

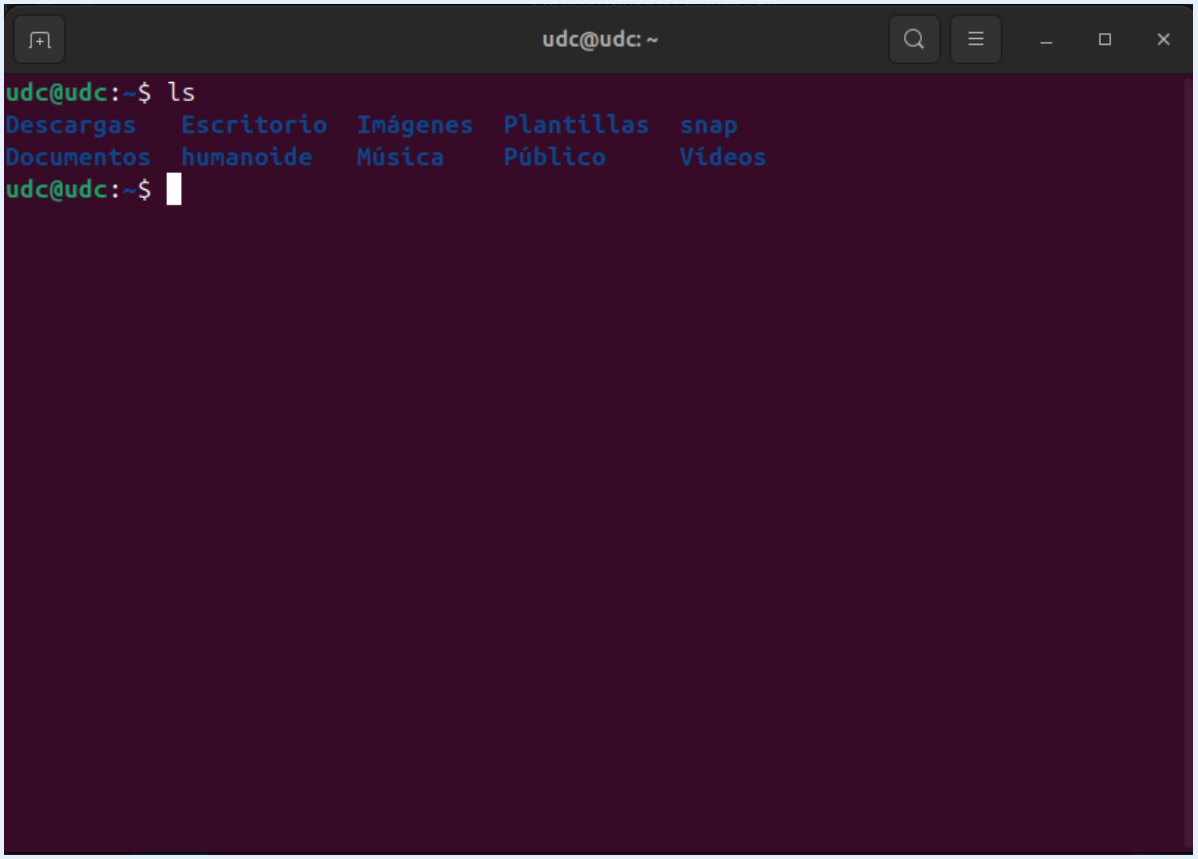
Funcionamiento del comando

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 significa que no existia y se creo la carpeta , y al final el nombre de la carpeta `humanoide`.

- Ejemplo:



- 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`.



- 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
```

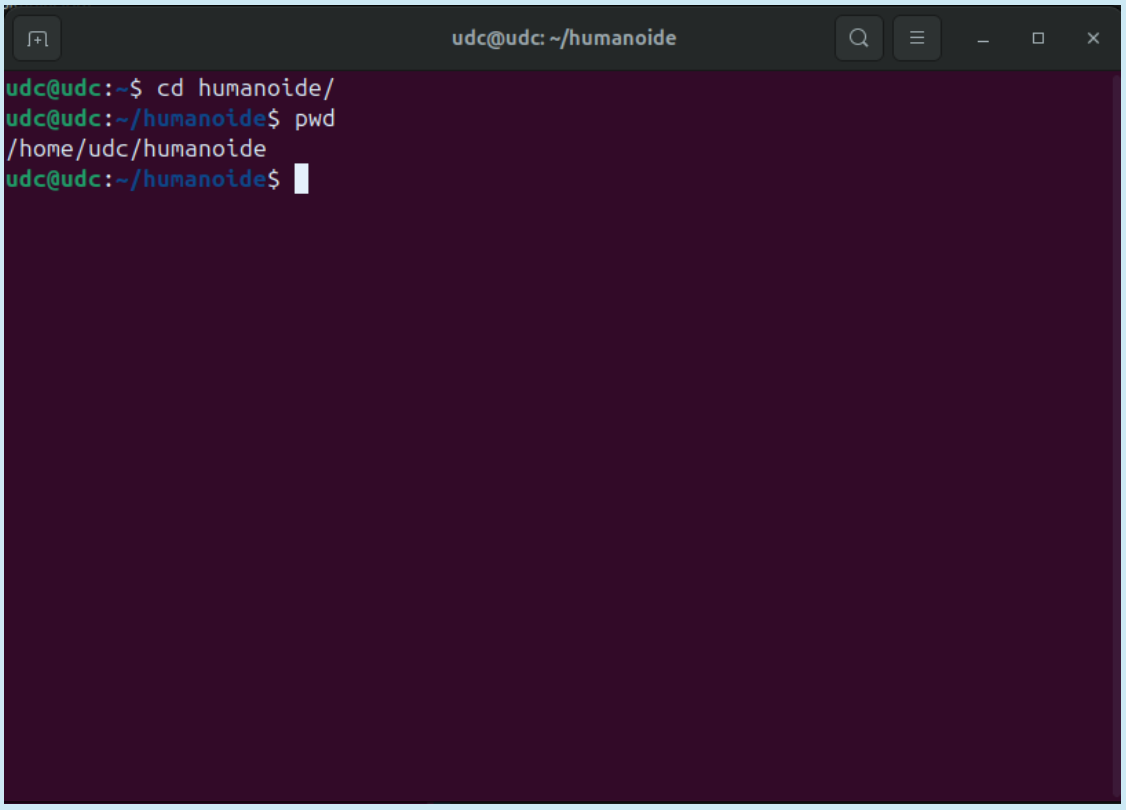
✔️ 6. Ir a una ruta relativa

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

```
cd humanoide
```

Te mueve a `/home/udc/humanoide`.

- Ejemplo:



- 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.**

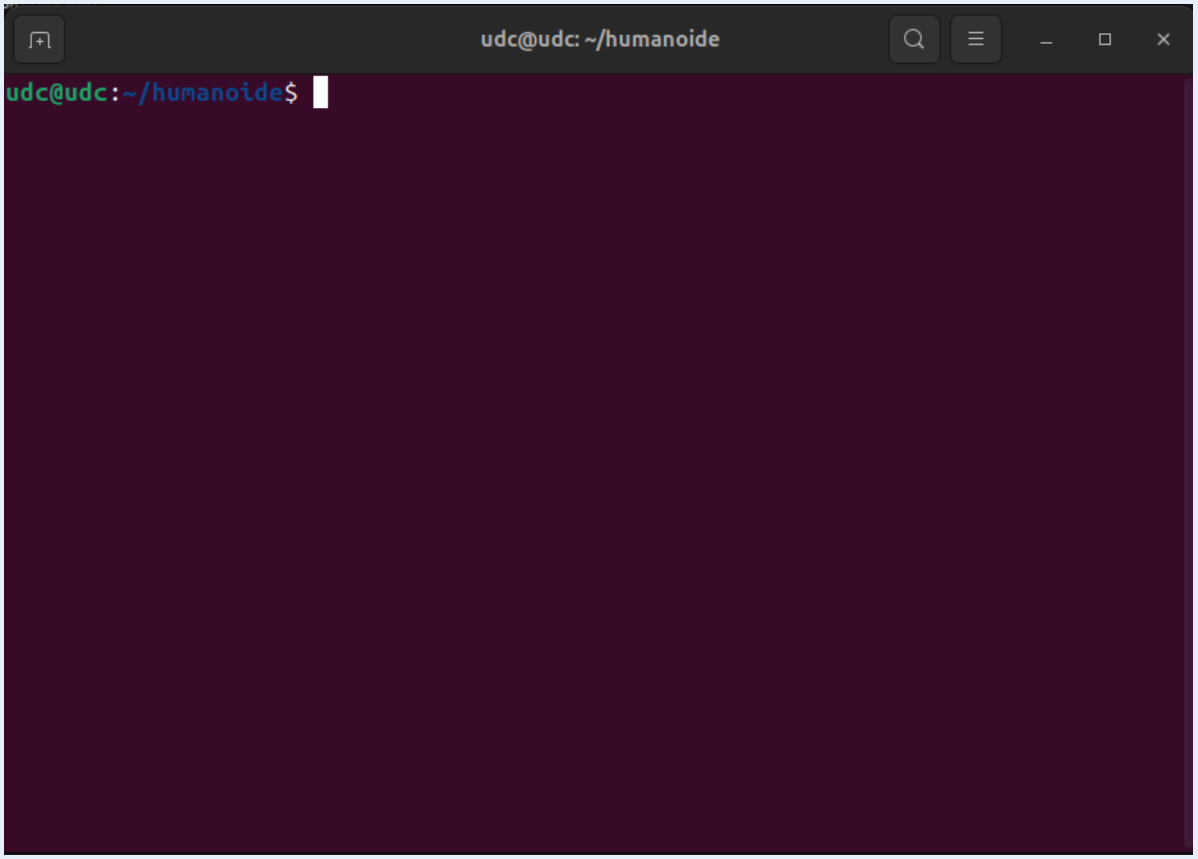
Clonación del recurso (Mujoco)

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

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

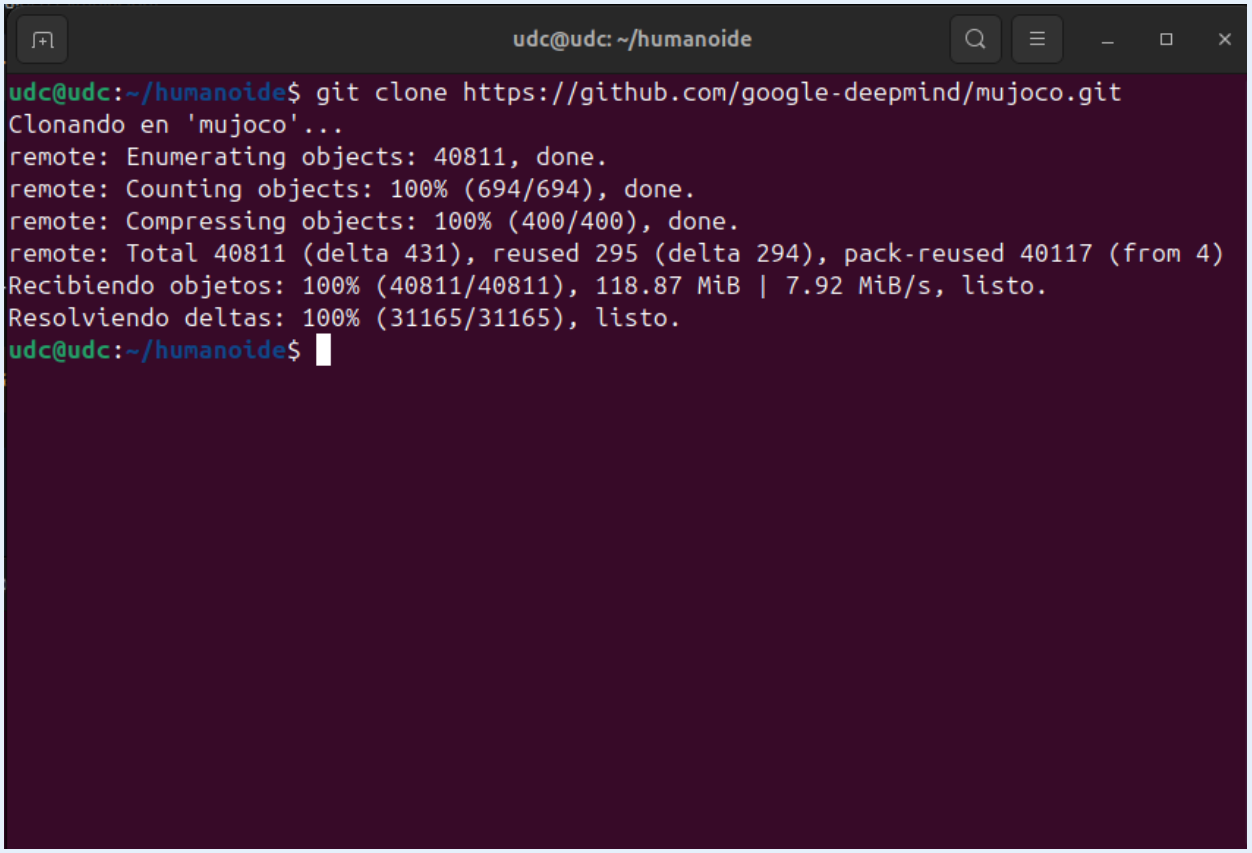
Pasos

- Estar ubicado en la carpeta `humanoide`

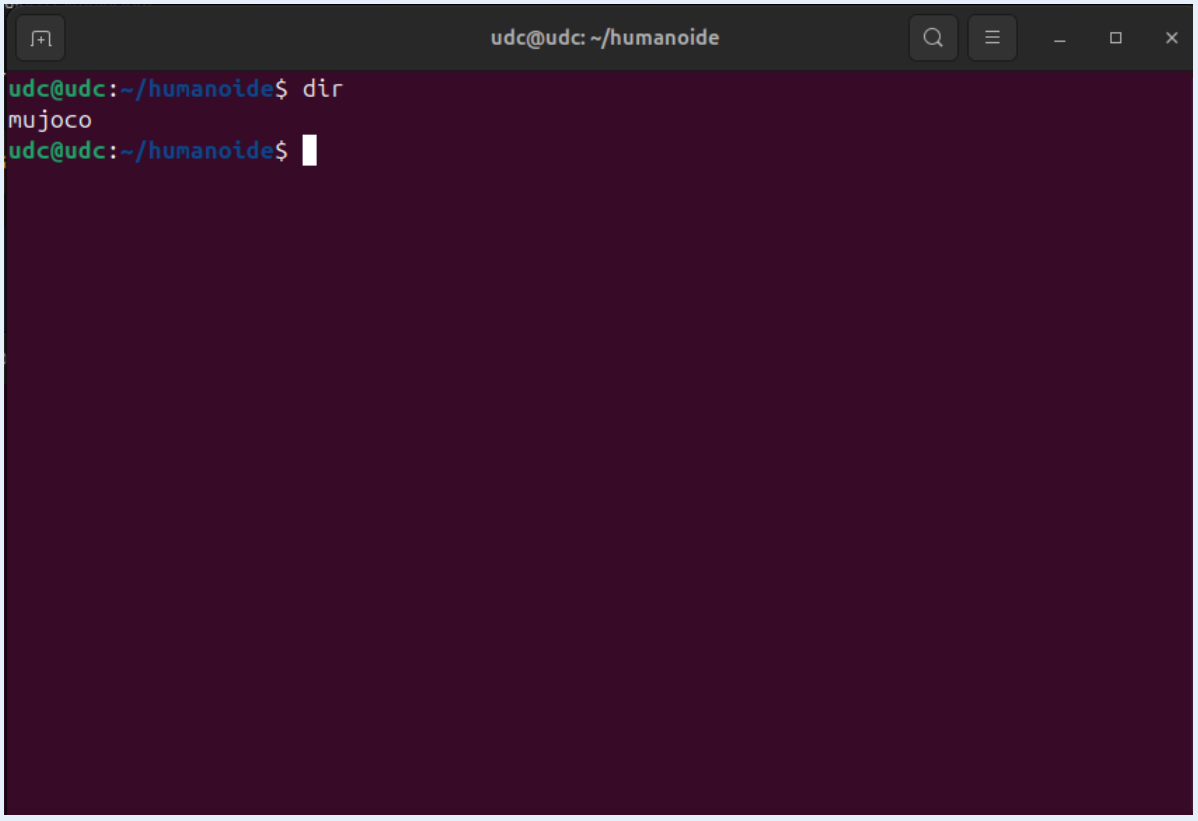


- Ingresar en la terminal el recurso a clonar y damos `Enter`.

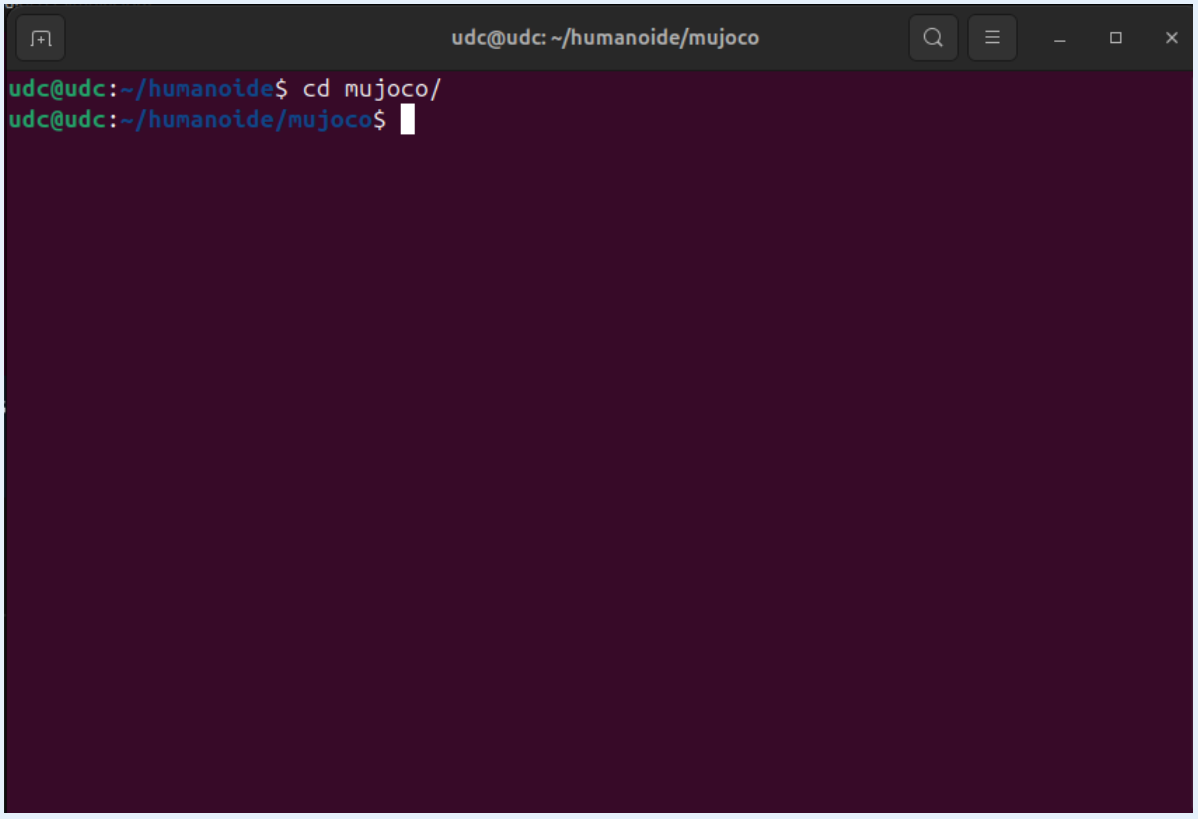
```
git clone https://github.com/google-deepmind/mujoco.git
```



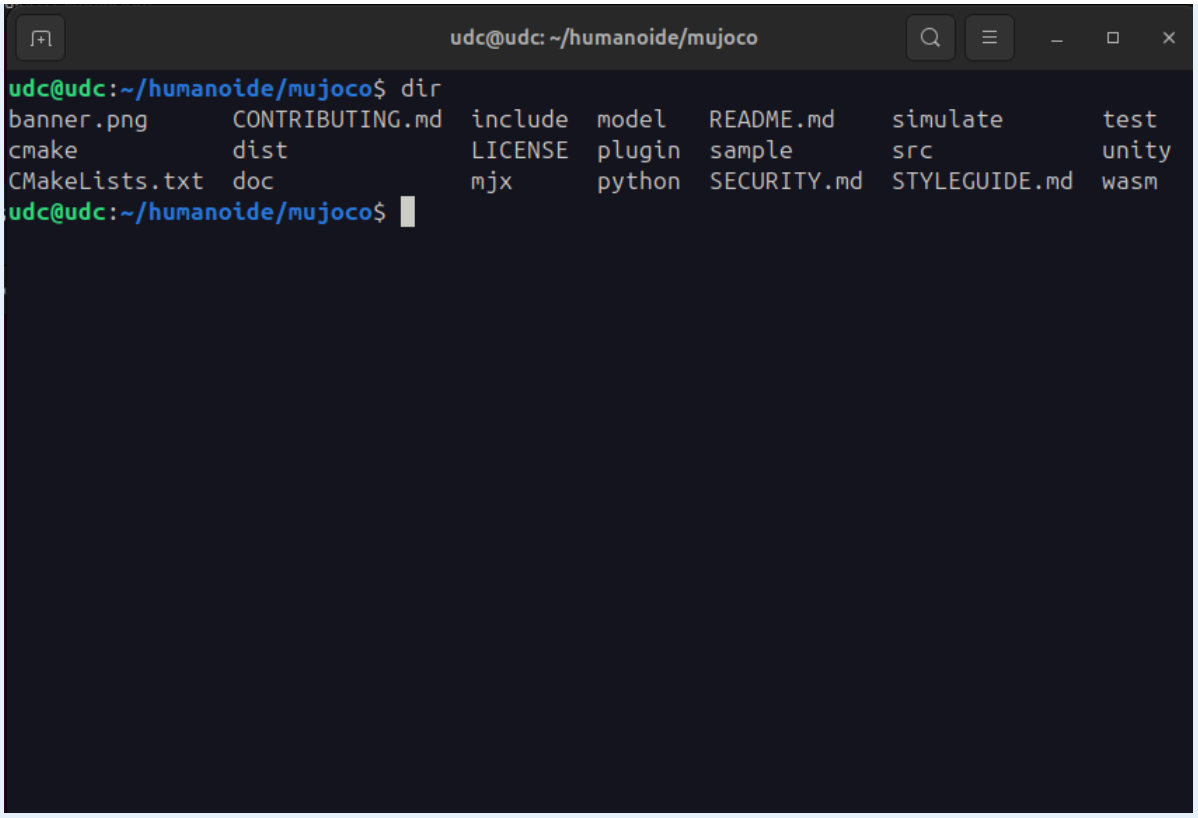
- Verificamos que al finalizar la clonación habrá una nueva carpeta en el interior de nuestra carpeta `humanoide`, llamada `mujoco`, para ello usamos `ls o dir`.



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



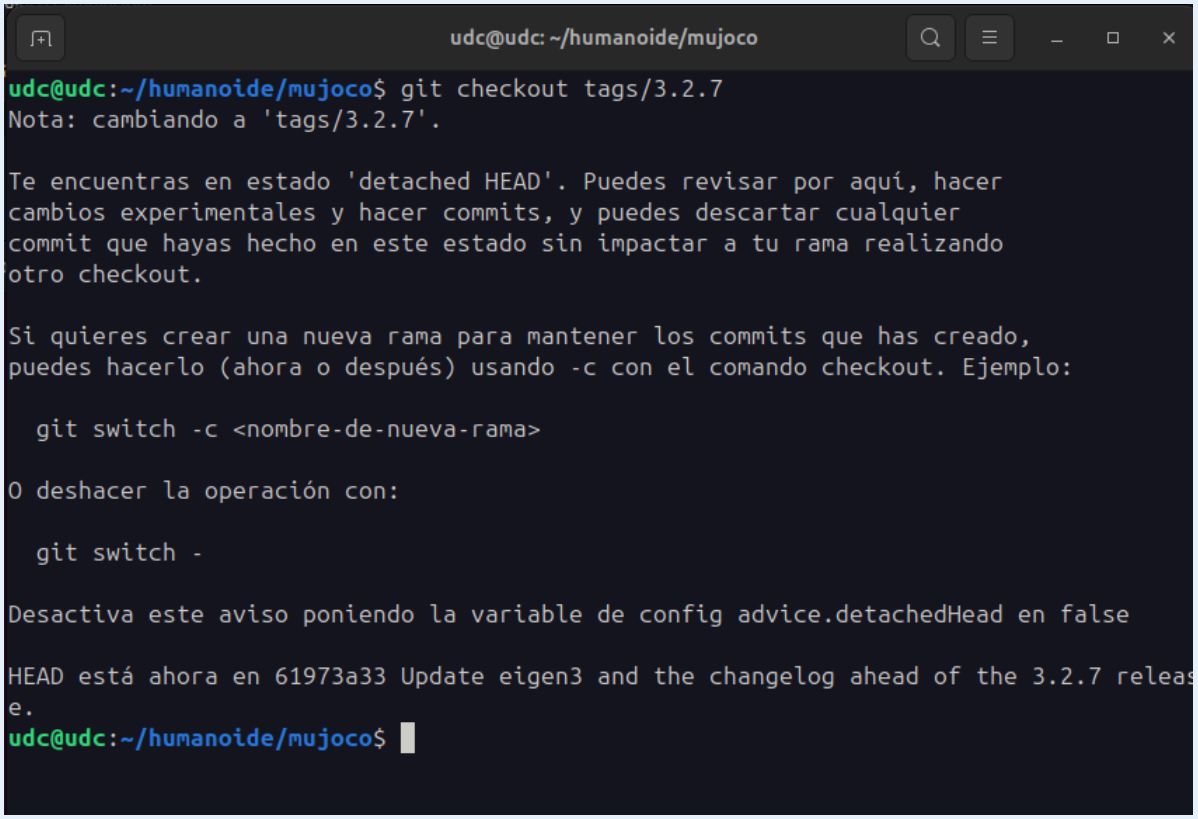
- Podemos visualizar el cotenido de `mujoco` al haber ingresado, usado `dir o ls`, nos mostrara archivos, carpetas e imagenes.



- Ahora haremos un cambio drastico, 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:



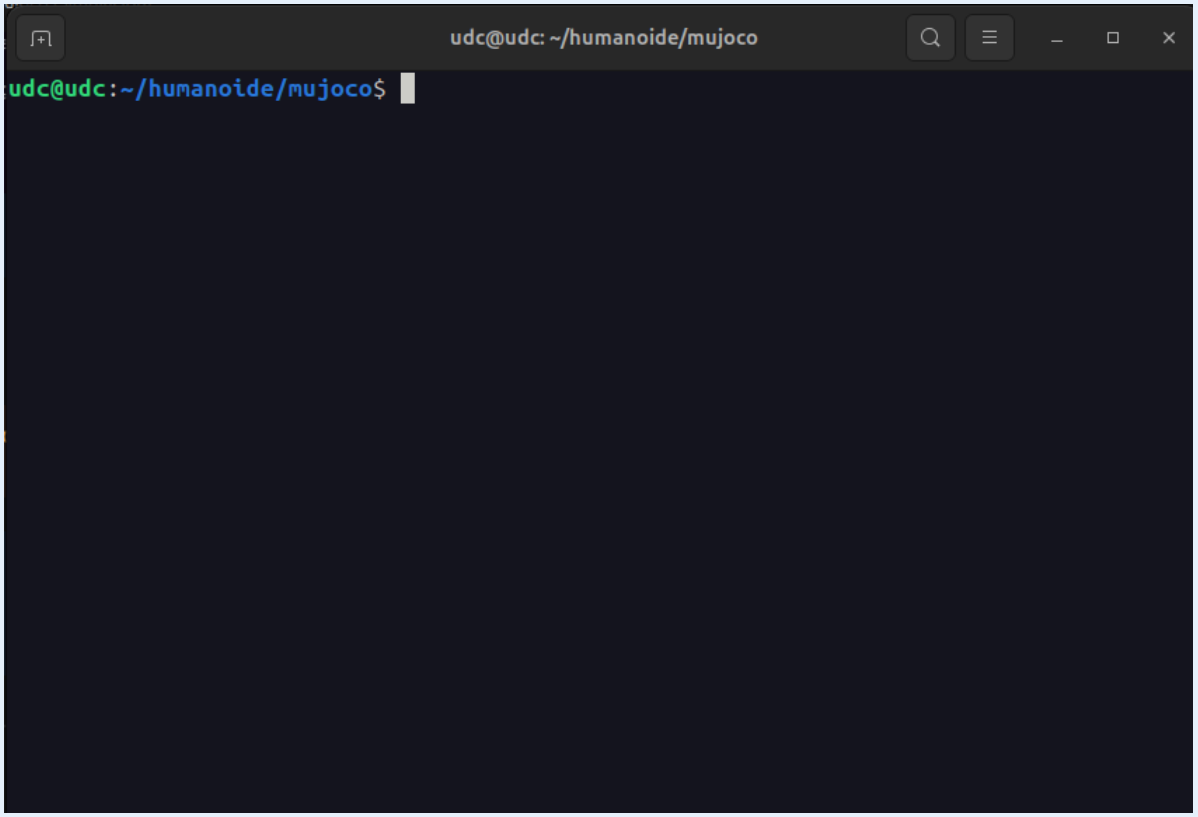
Significa que fue exitoso el cambio...

Construcción de Mujoco

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

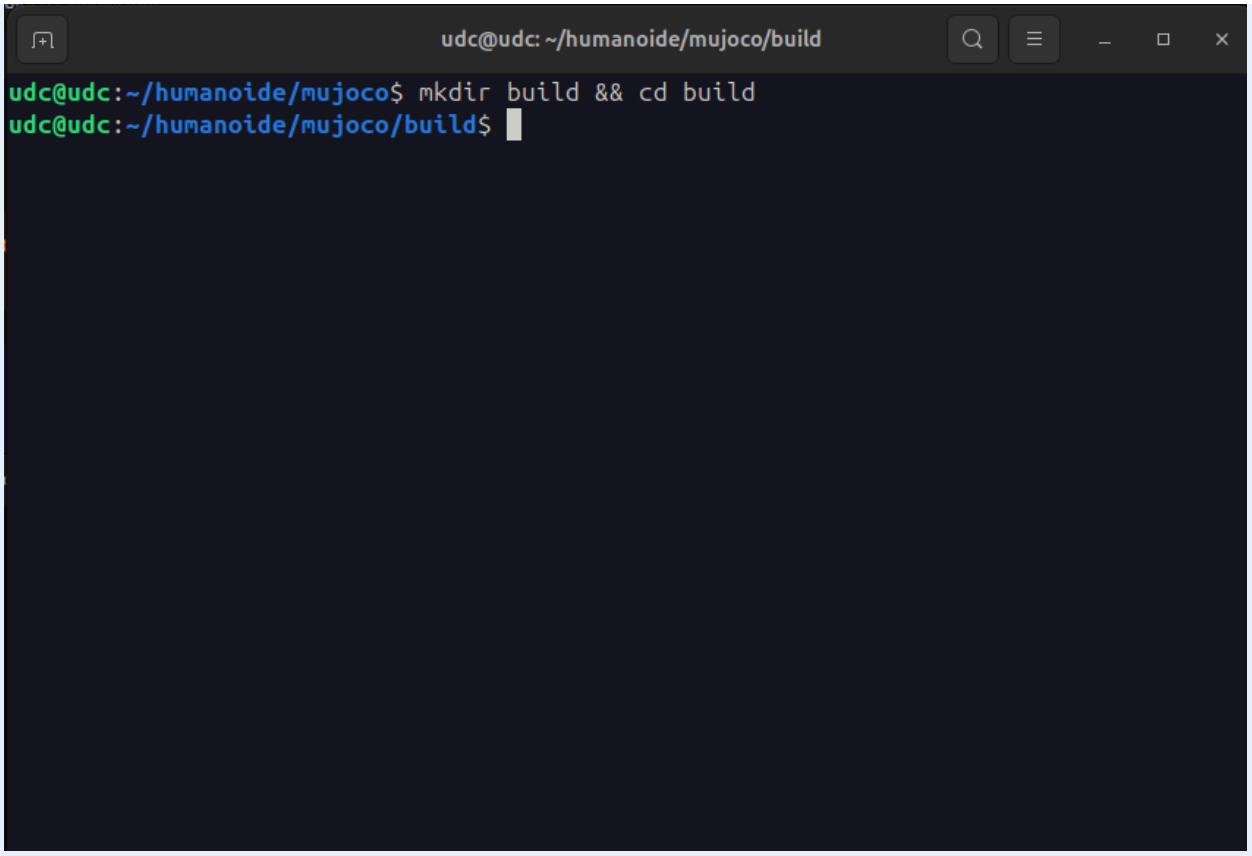
🖊 Pasos

- Desde la ubicación de:



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

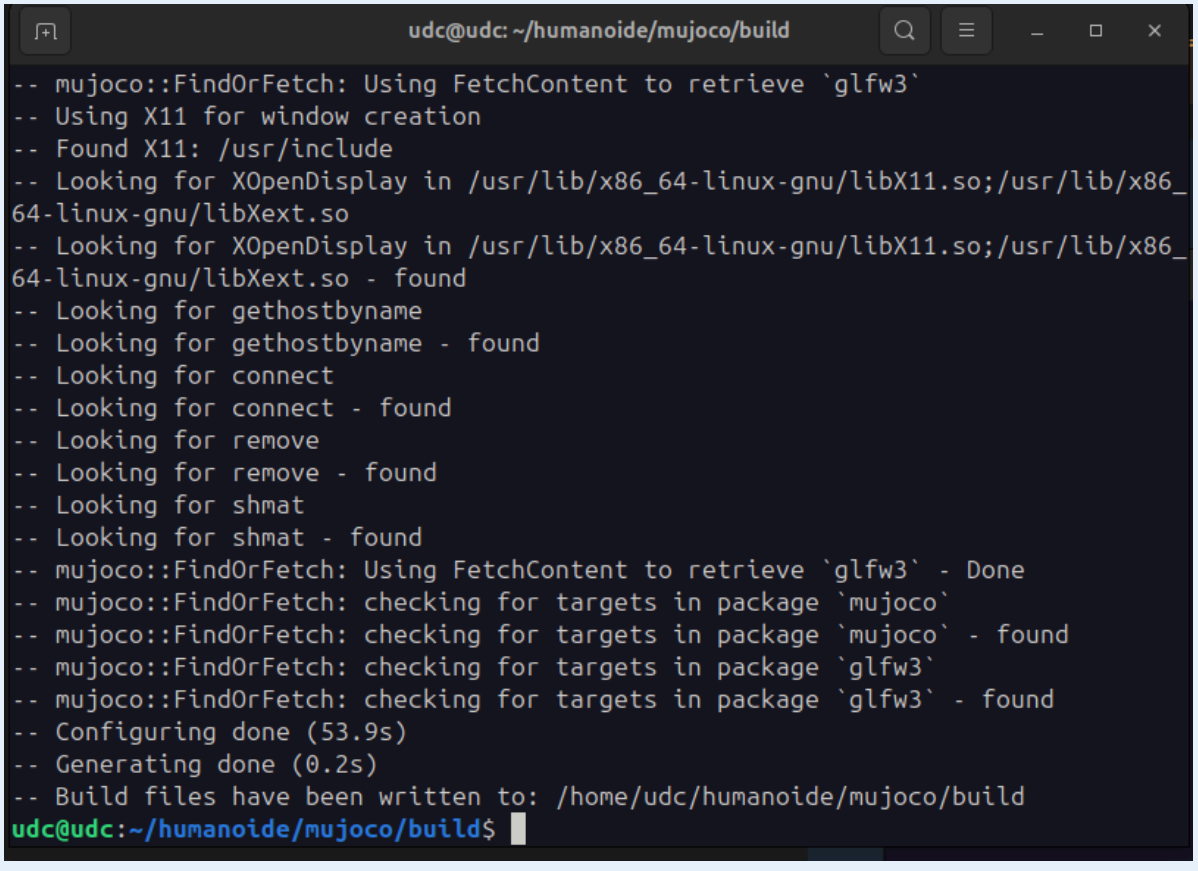
```
mkdir build && cd build
```



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

```
cmake ..
```

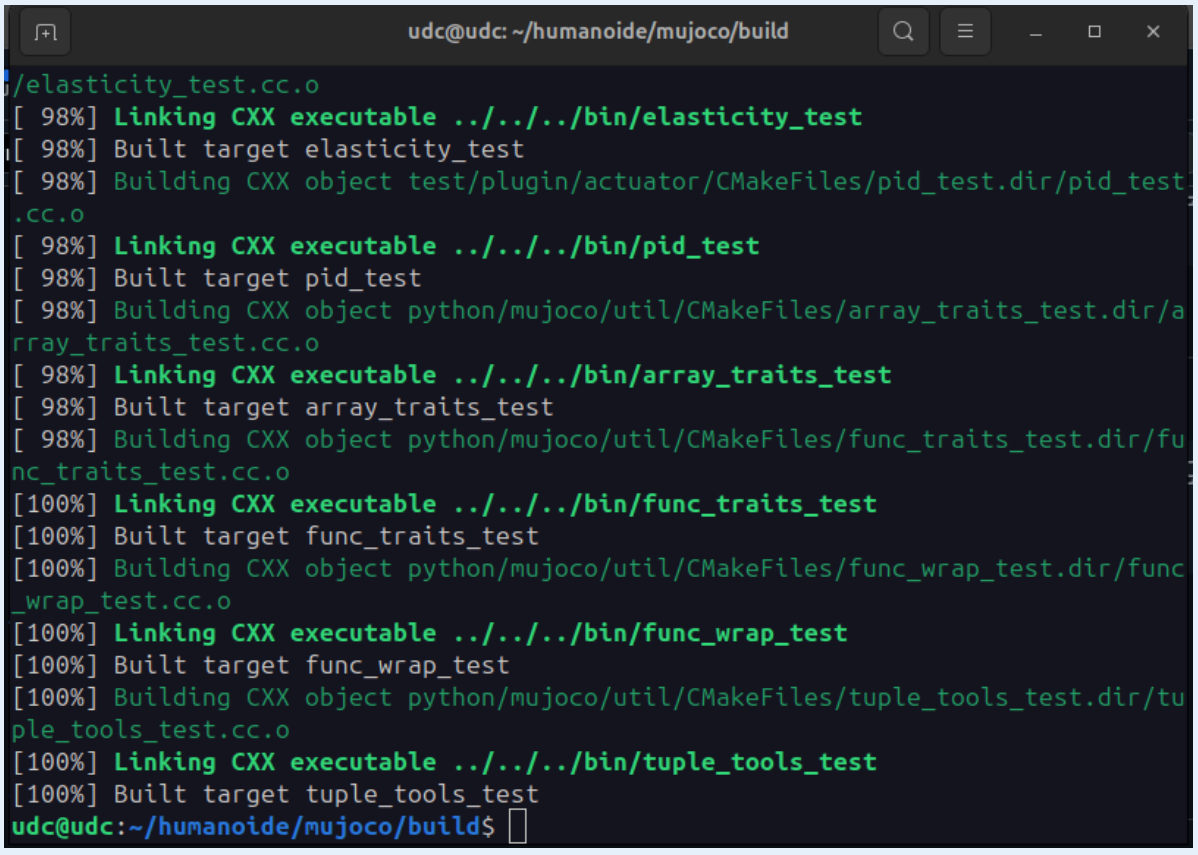
- Esperamos unos segundos y en la parte final debemos ver el siguiente mensaje, que nos confirma la construcción exitosa



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

```
make
```

- Nuevamente esperamos unos minutos y podremos visualizar lo siguiente:



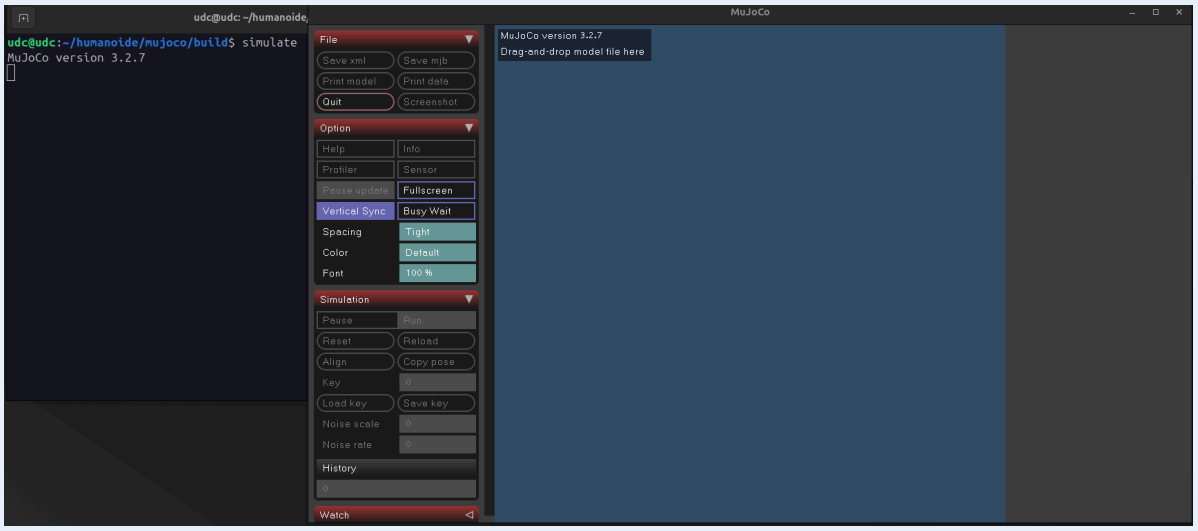
- Ahora colocamos otro

```
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
```

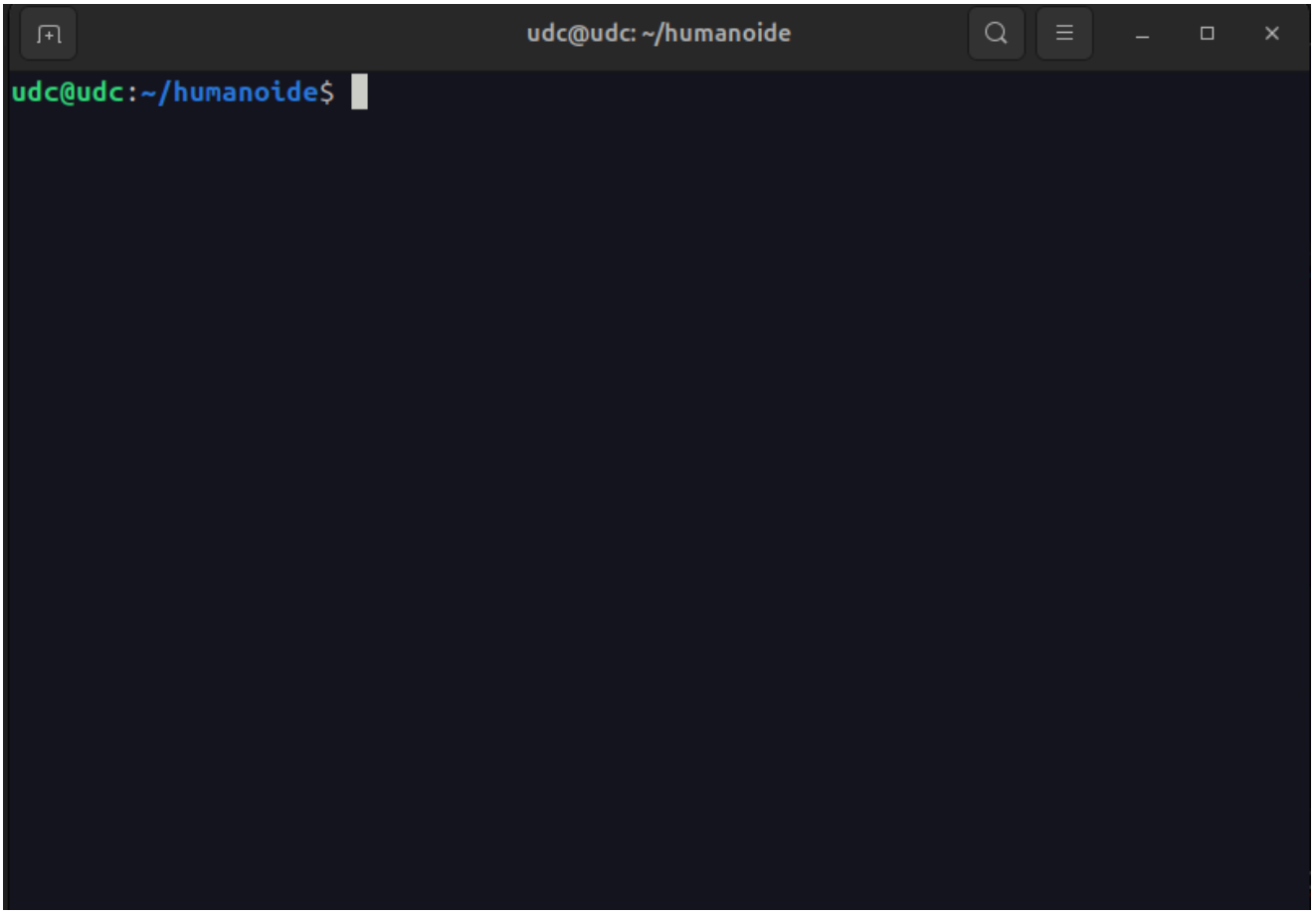
- Visualizaremos lo siguiente



---

3 - Clonación y construccion de Unitre_sdk2

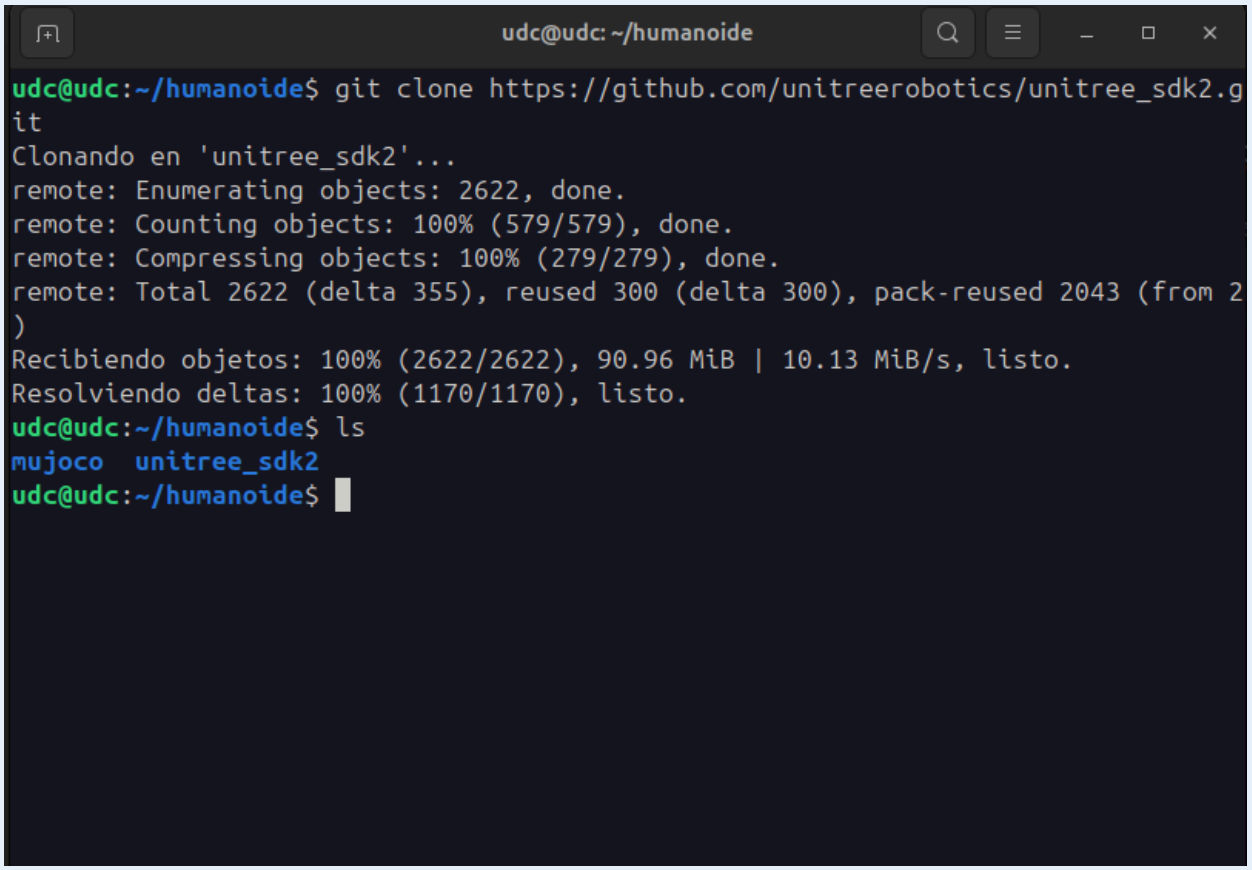
Este recurso debe estar en la carpeta `humanoide` donde clonamos el recurso de `mujoco`



🖊 Pasos

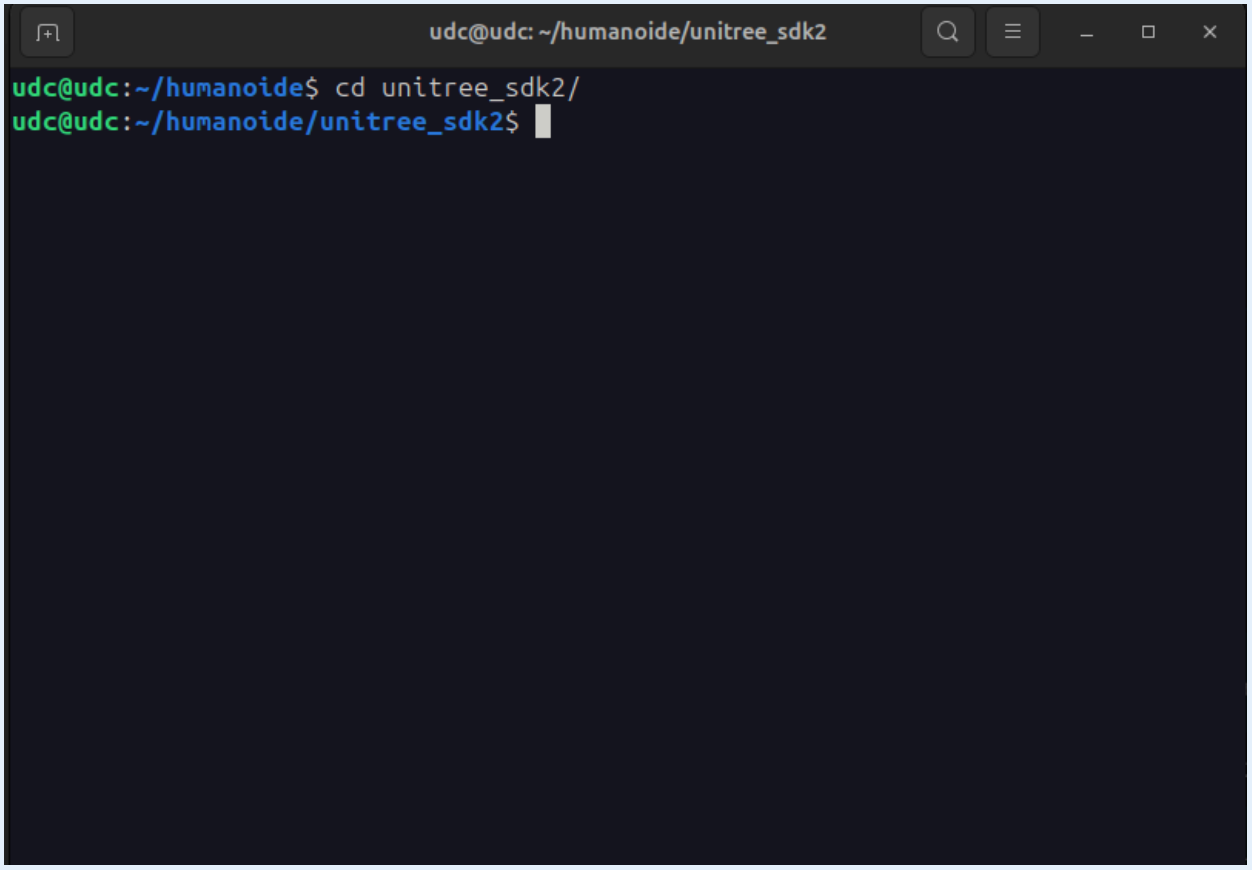
Clonar el recurso de `unitree_sdk2`, con `ls` debemos estar visualizando en la carpeta `humanoide` dos carpetas `mujoco` y `unitree_sdk2`.

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



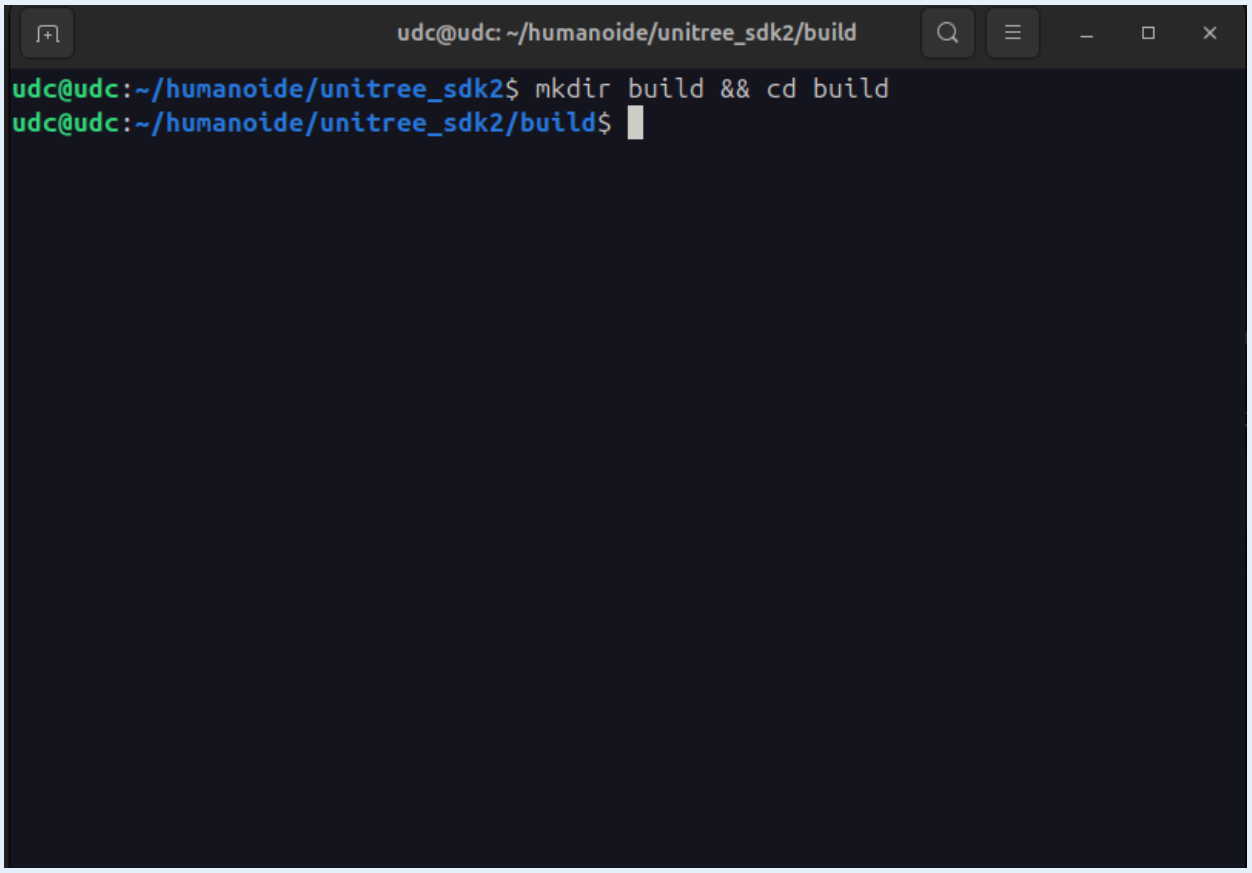
- Ingresamos a la carpeta `unitree_sdk2`

```
cd unitree_sdk2
```



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

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



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

```
cmake ..
```

Después ingresamos el siguiente comando:

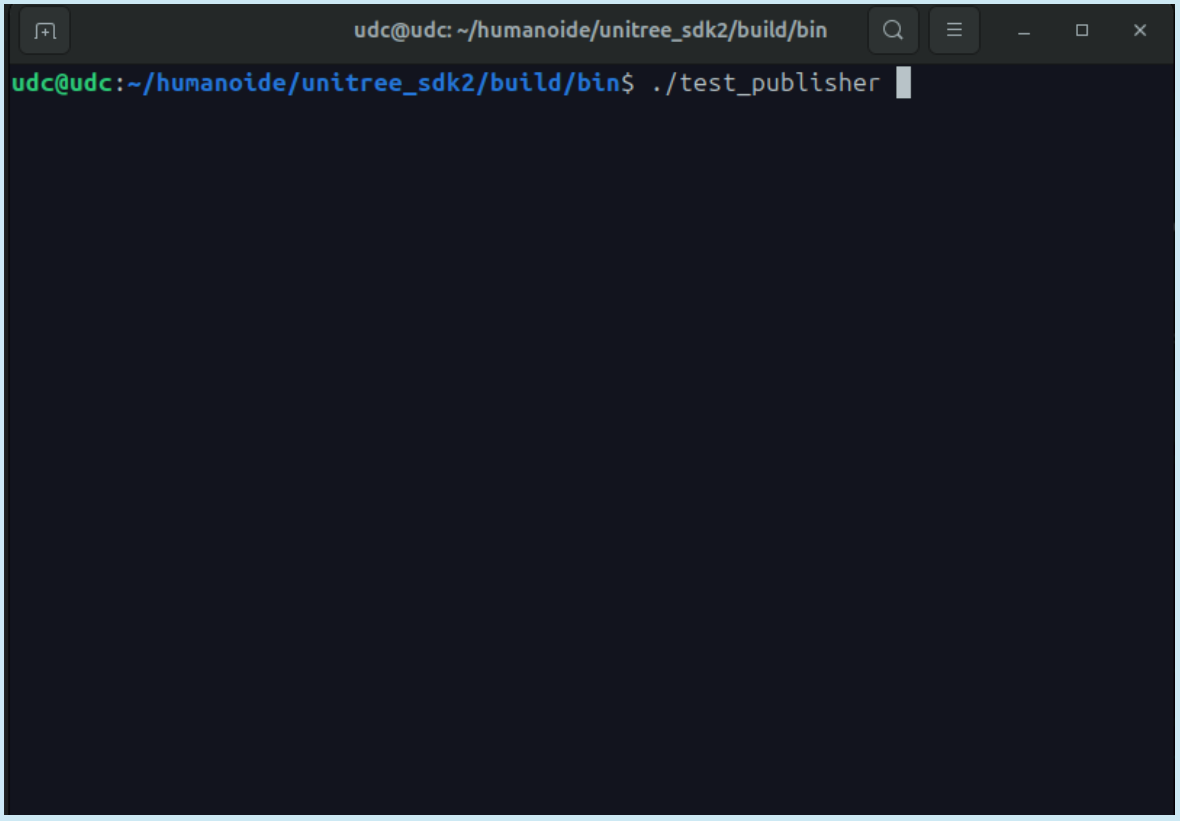
```
sudo make install
```

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

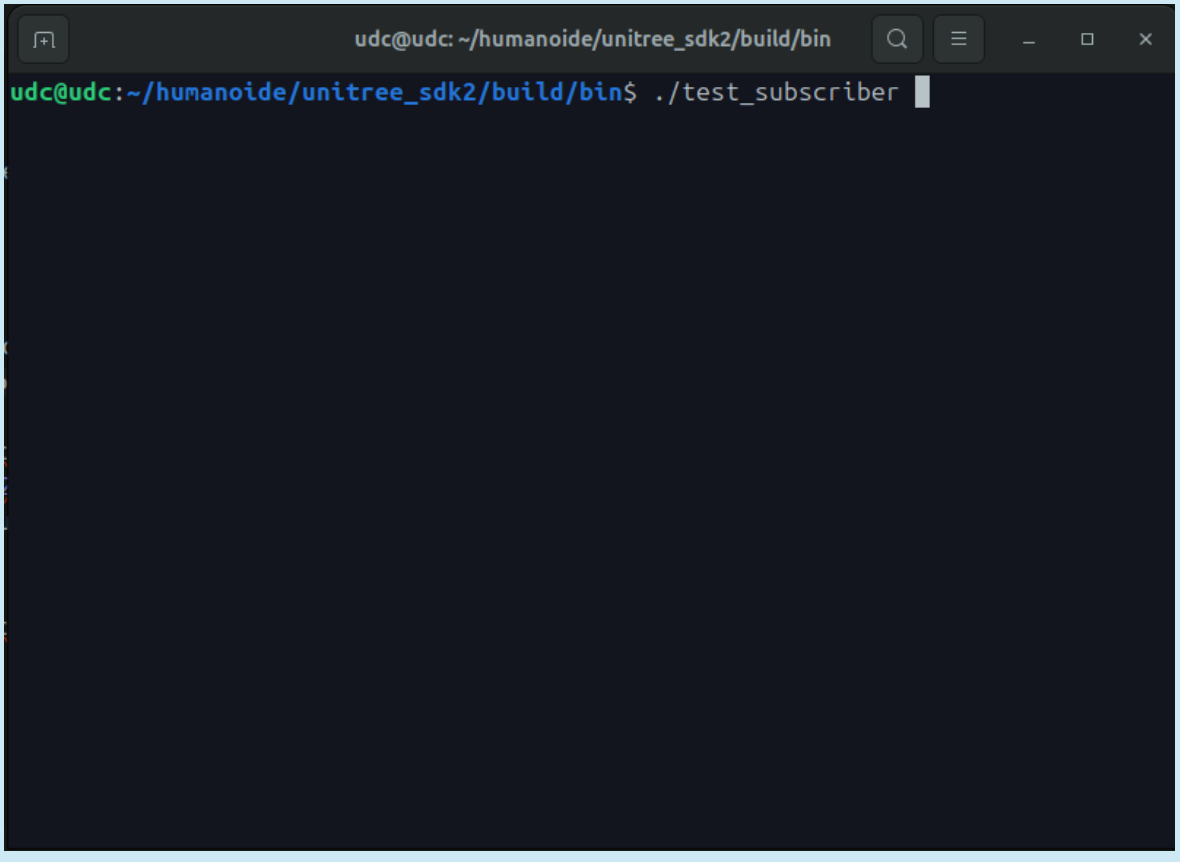
Ejecutaremos un script de prueba `./test_publisher`



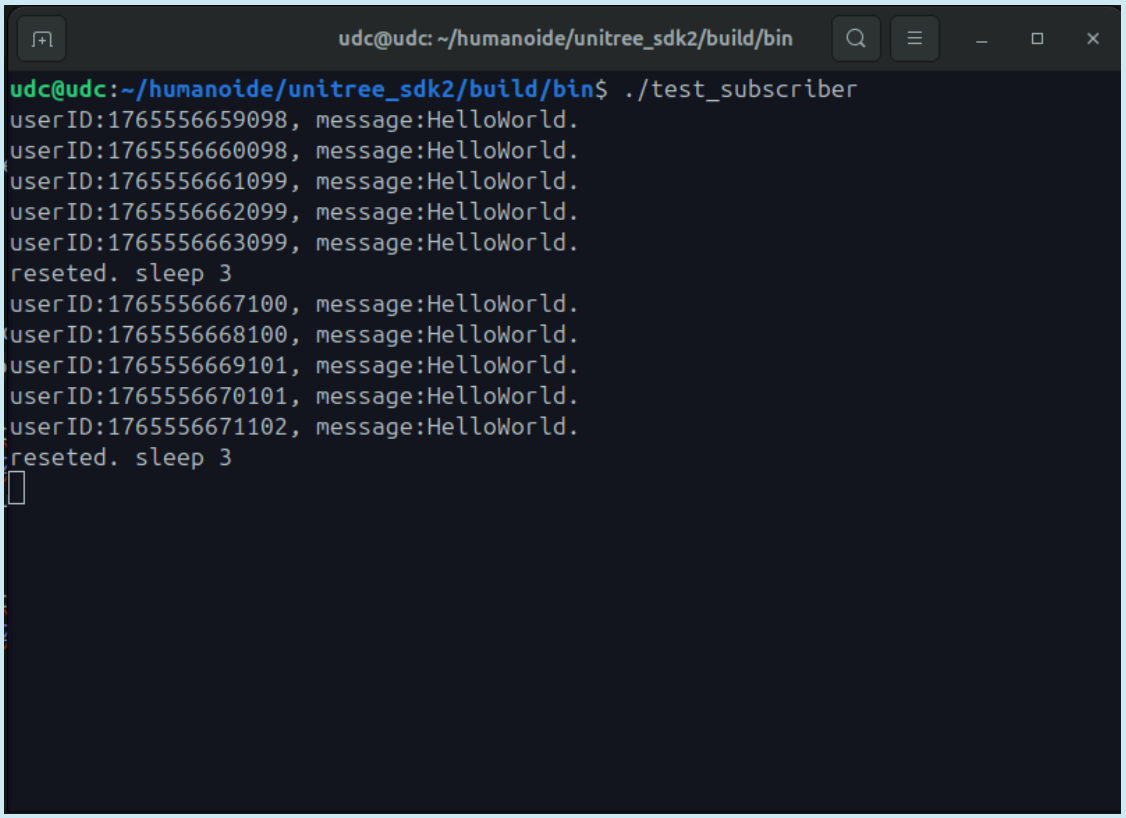
- Estara esprando una conexion que la realizaremos desde la segunda terminal

Terminal #2

Ejecutaremos un script de prueba `./test_subscriber`



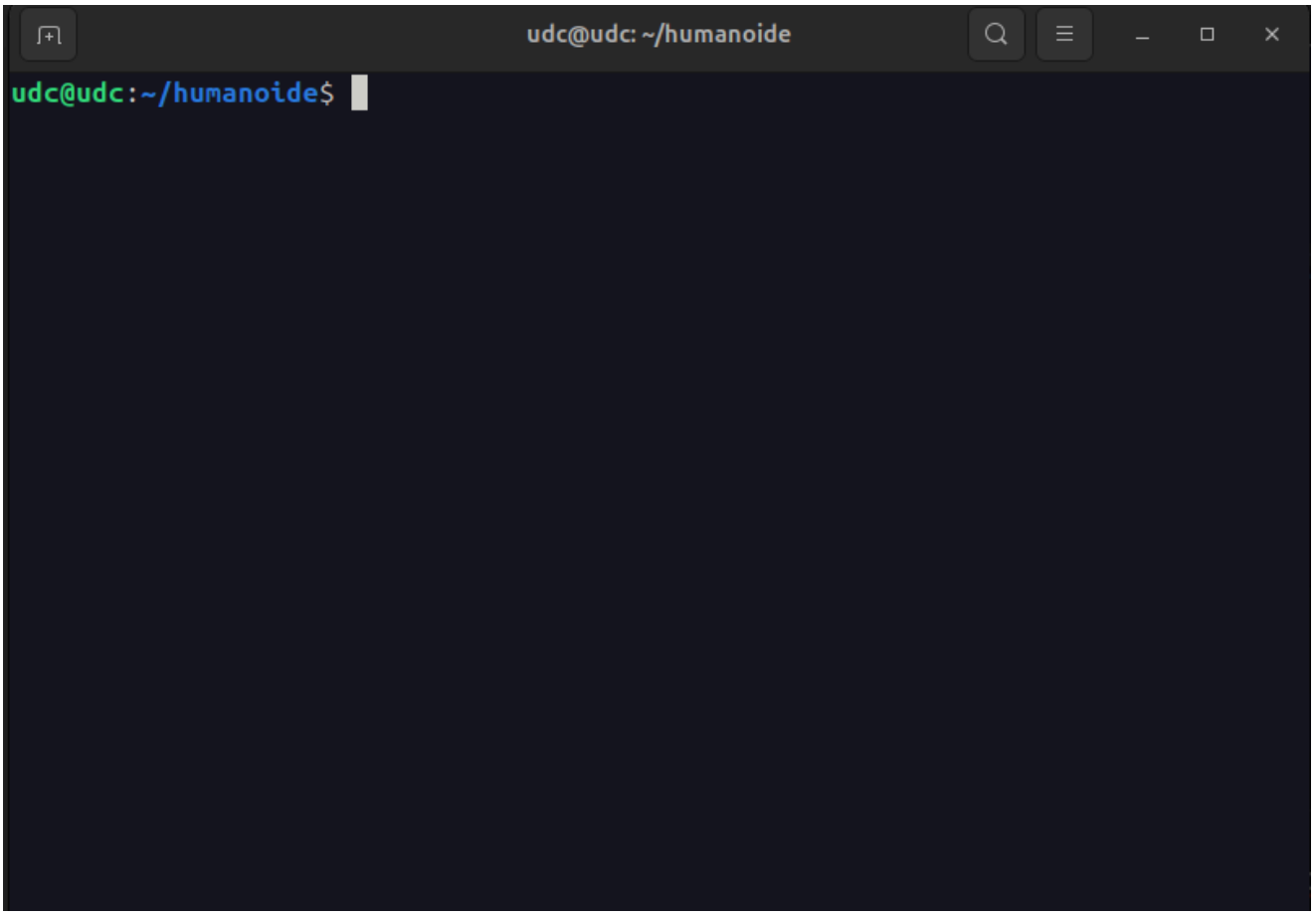
- Resultado de la espera



---

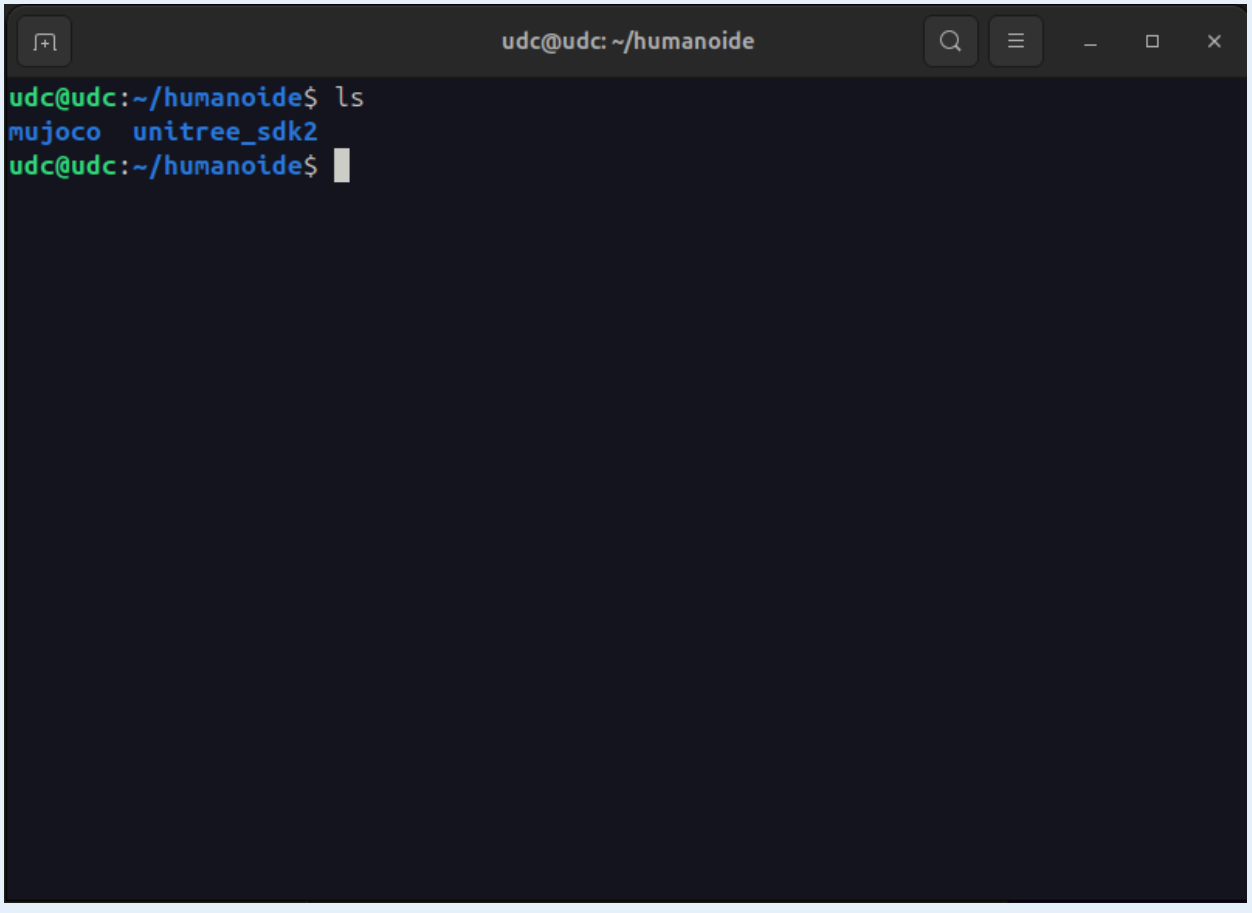
4 - Clonación e instalación de `unitree_sdk2_python`

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

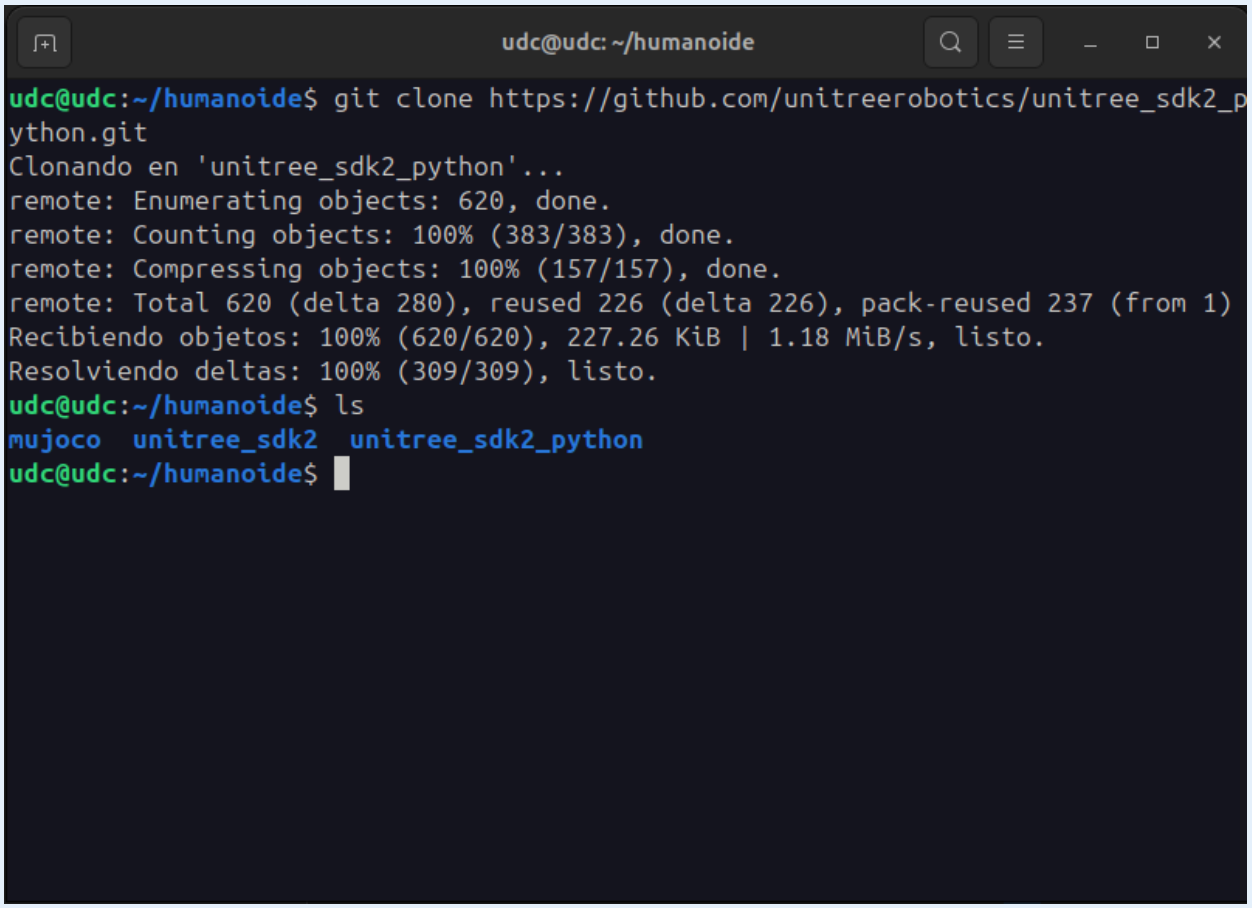


Pasos

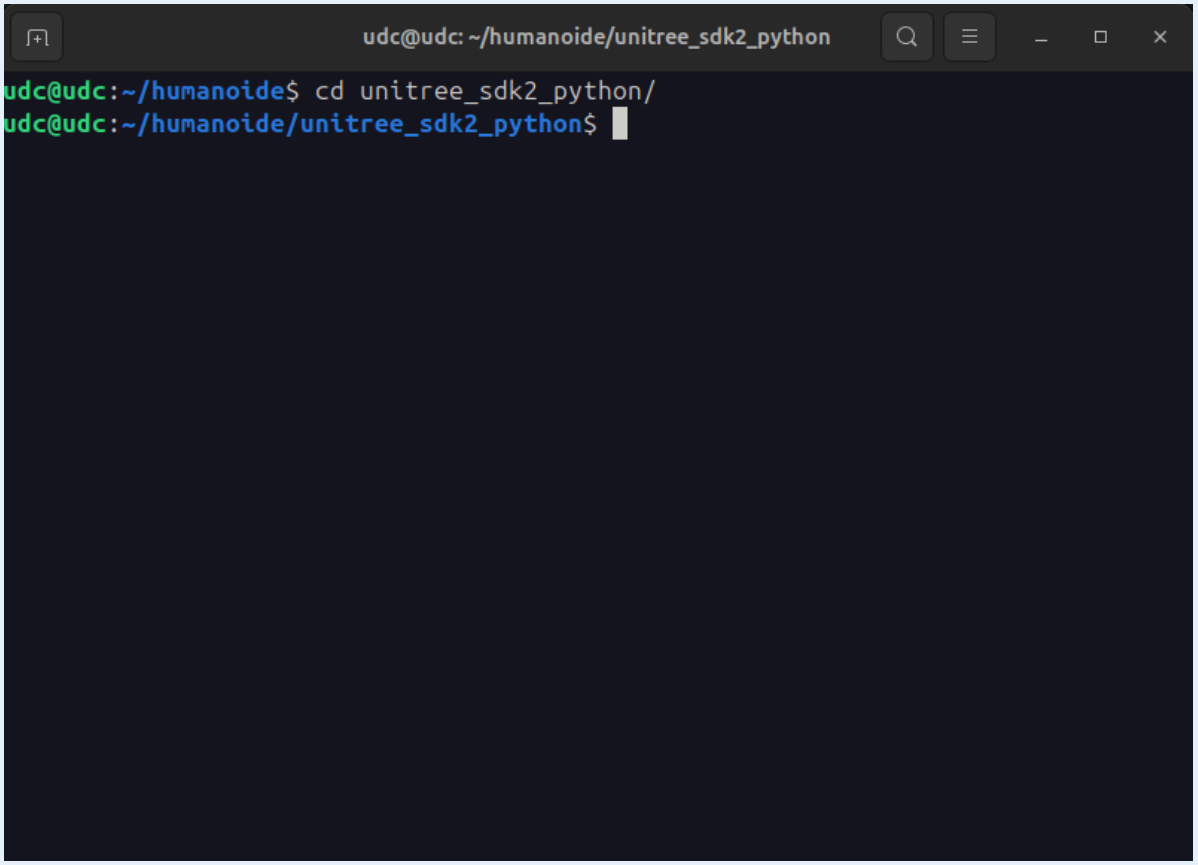
Clonar el recurso de `unitree_sdk2_python`, con `ls` debemos estar visualizando en la carpeta `humanoide` dos carpetas `mujoco` y `unitree_sdk2`.



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

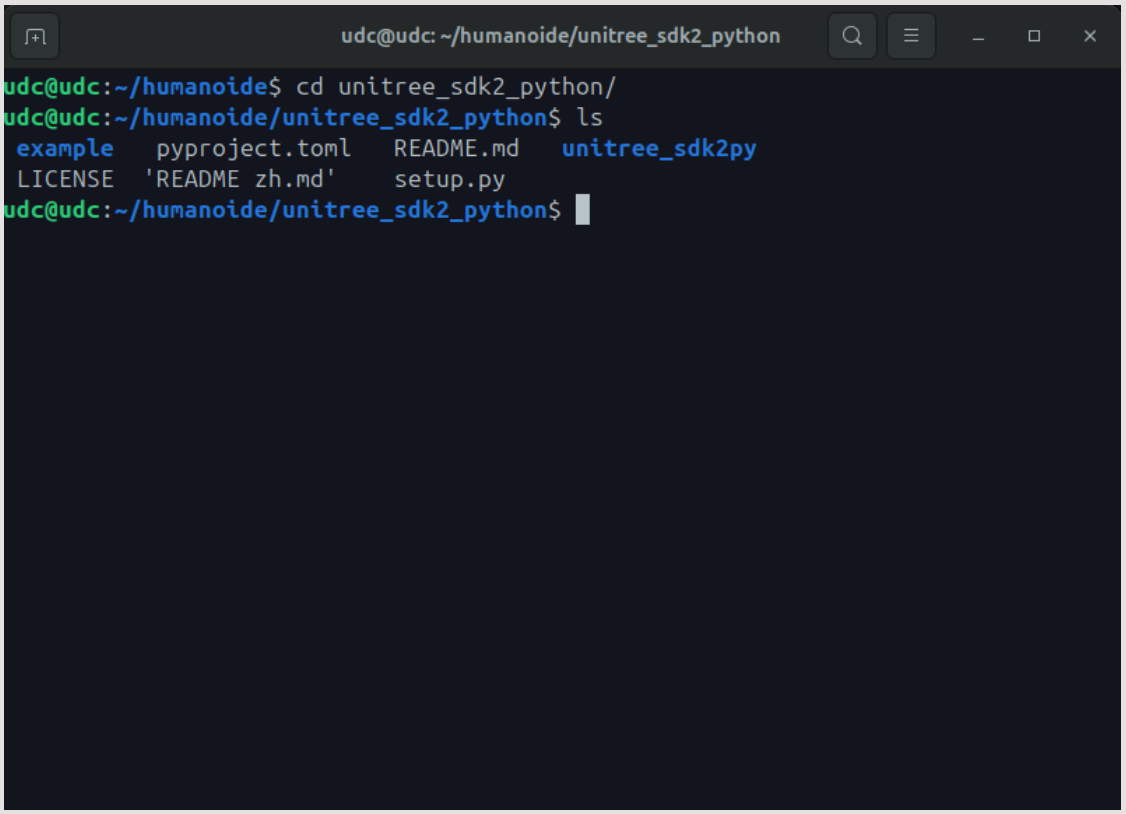


- Ingresamos a `unitree_sdk2_python`



> ⚠ **Tener en cuenta**

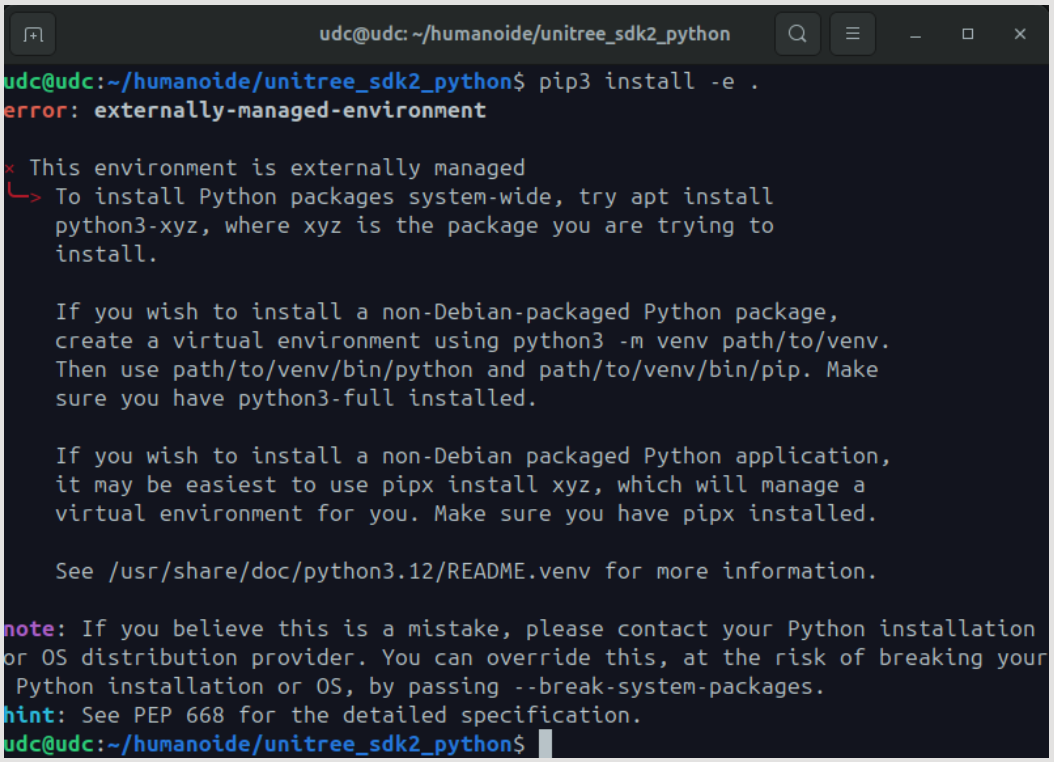
Ese recurso trabaja con python, utiliza archivo de configuración distinto a los anteriores, en este caso podemos saber cuál es, un `pyproject.toml`



- Para construir el entorno usaremos el siguiente comando

```
pip3 install -e .
```

- De primer momento se puede presentar un error:



- 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/`

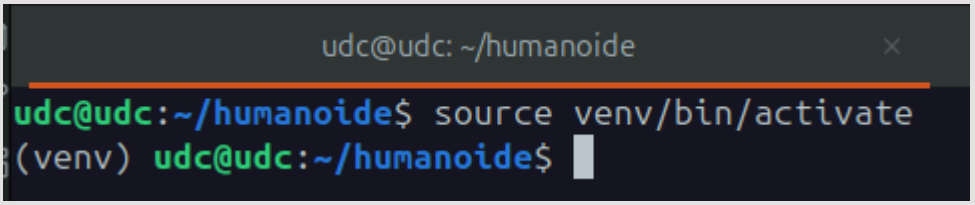
- Desde ese punto creamos el entorno, usando el siguiente comando en terminal:

```
python3 -m venv venv
```

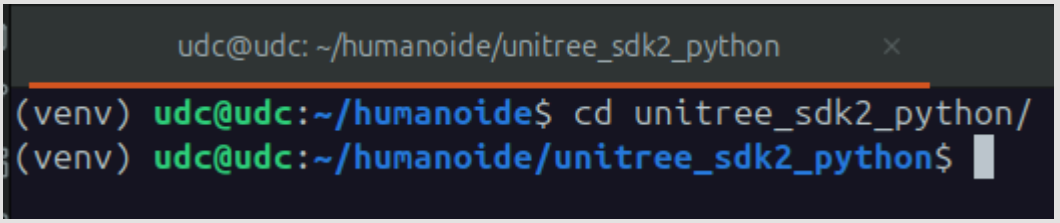
- Después activamos el entorno virtual *(**OJO**, siempre toca activarlo en cada instancia de terminal si queremos trabajar con el código de la carpeta)*
- Activamos en entorno virtual:

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

- Veremos el siguiente cambio:



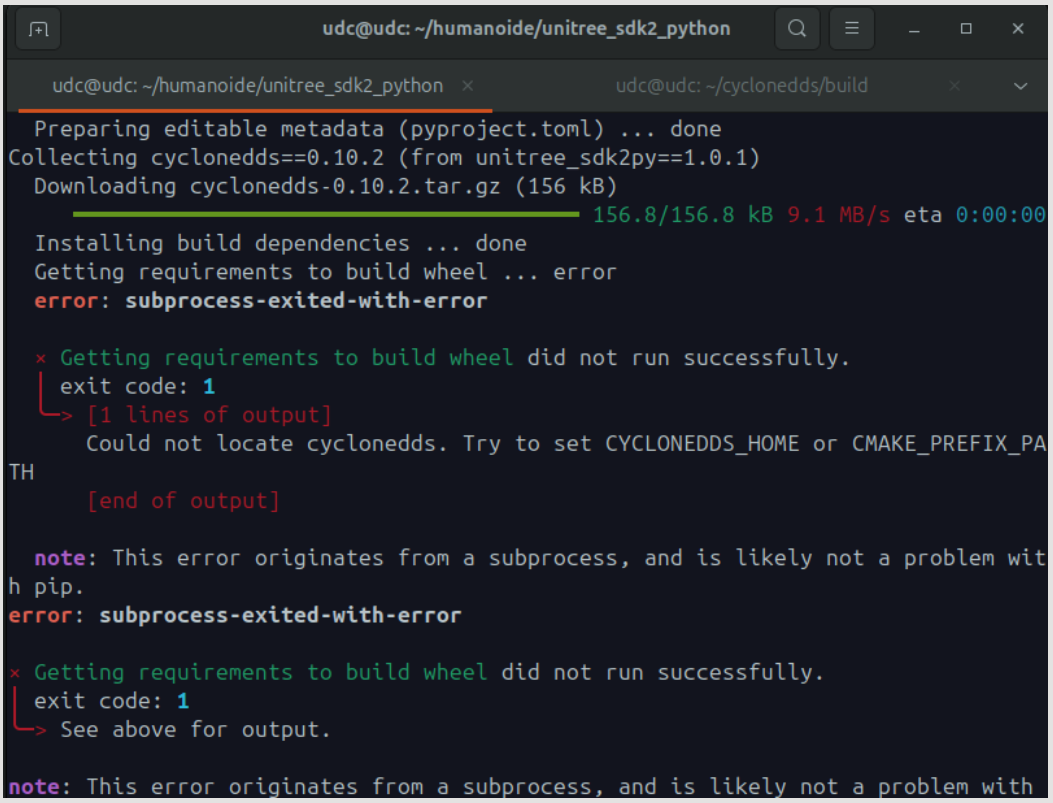
- Vamo a la carpeta de `unitree_sdk2_python` y vamos a usar el comando que nos permite configurar lo necesario para el entorno:




```
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

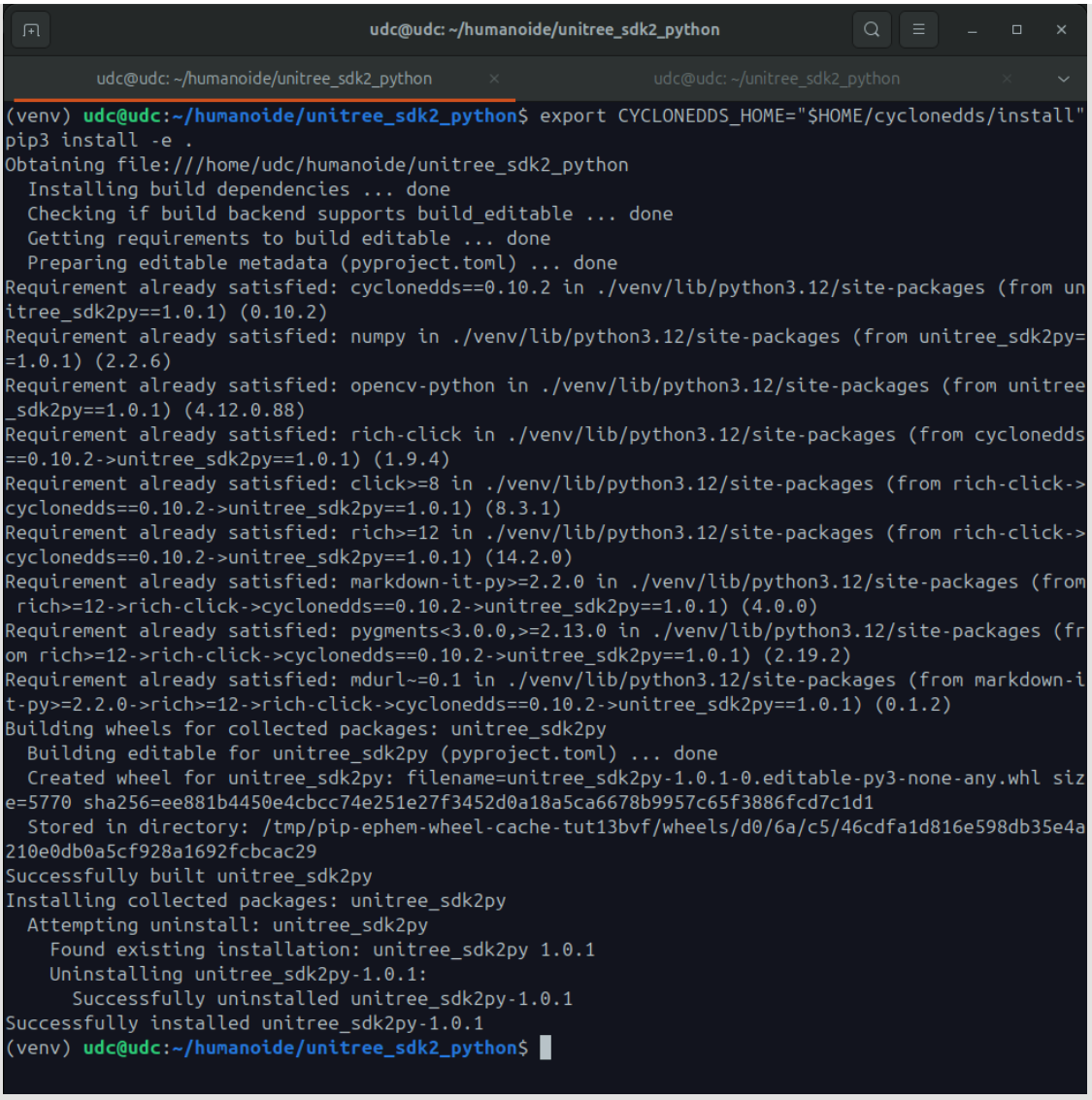


- Mostrando que necesitamos otro recurso, para ello abrimos una nueva terminal, colocaremos lo siguiente y ejecutamos:

```
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 .
```



- Se instalo lo necesario correctamente.

---

5 - Clonación e instalación de `unitree_mujoco`

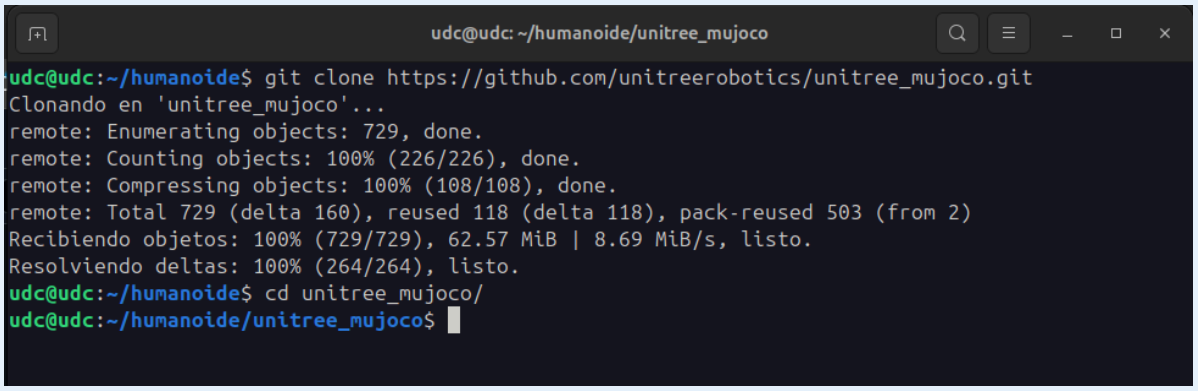
Regresamos a la carpeta `humanoide` y volvemos a la clonacion e instalación del ultimo recurso.

Pasos

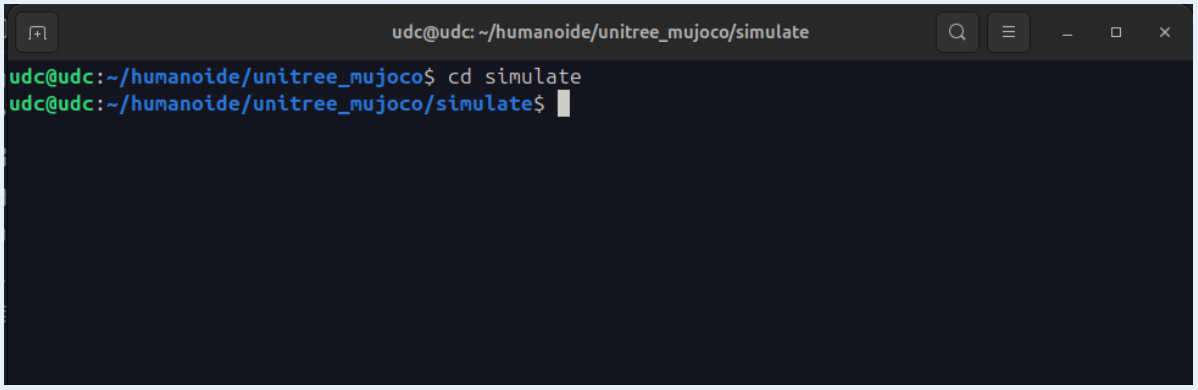
- Clonar el recurso

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

- Ingresamos al recurso

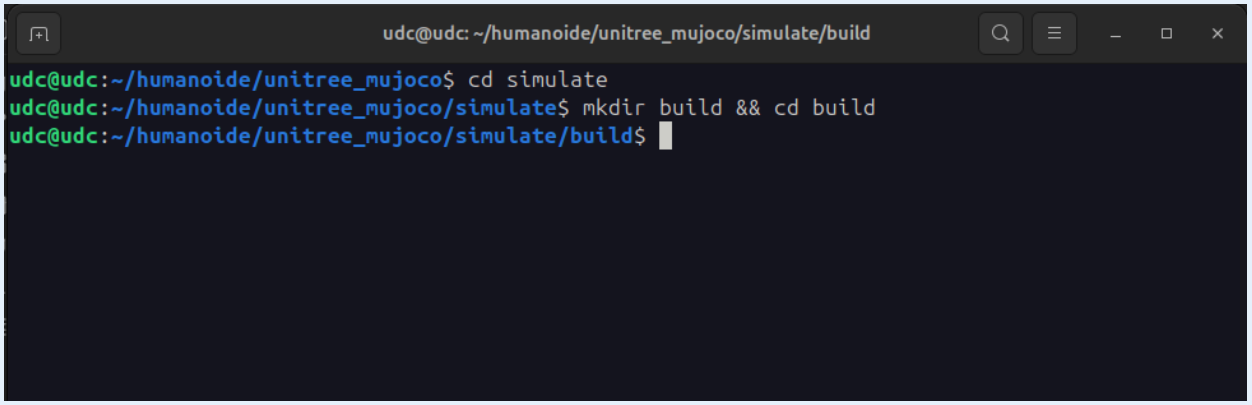


- Nos movemos a la carpeta `simulate`



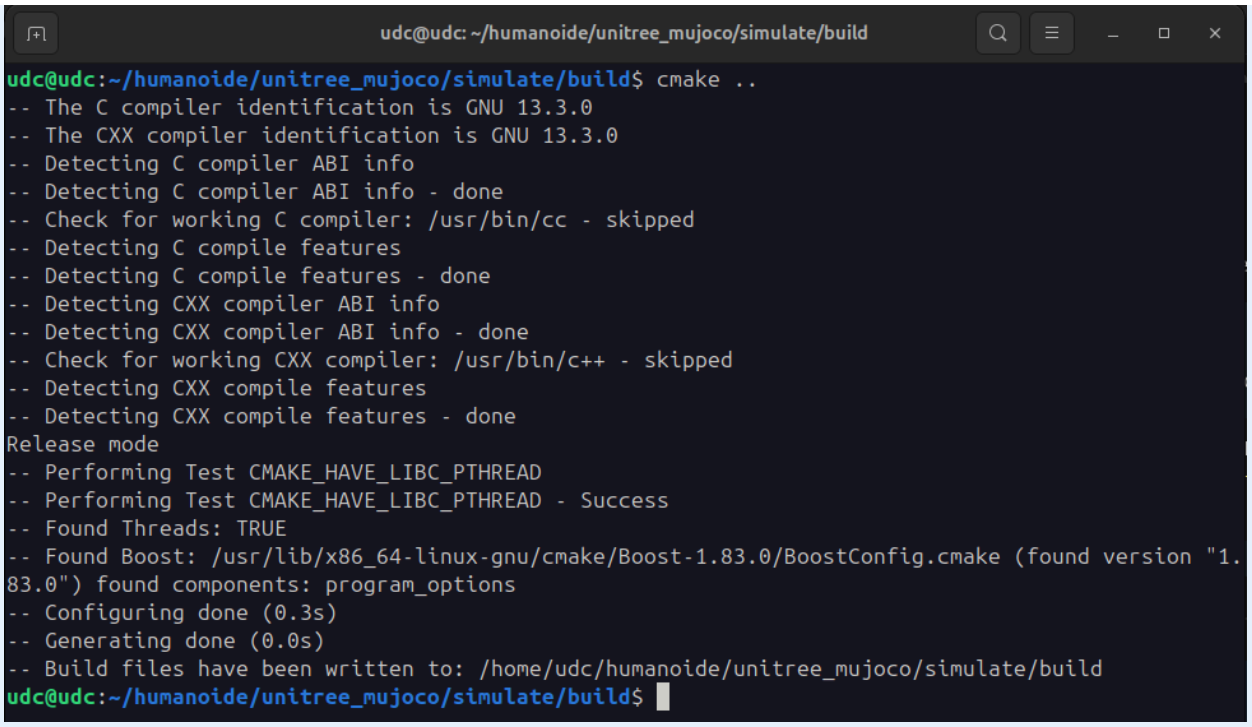
- Vamos a crear una carpeta llamada `build` y movernos a ella desde una solar orden:

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



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

```
cmake ..
```



Después ingresamos el siguiente comando:

```
make
```

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

- Para la solución debemos colocar la ruta absoluta de los archivos u carpeta, *¿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 codigo, 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 esta ubicado en `/home/udc/humanoide/unitree_mujoco/simulate`

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

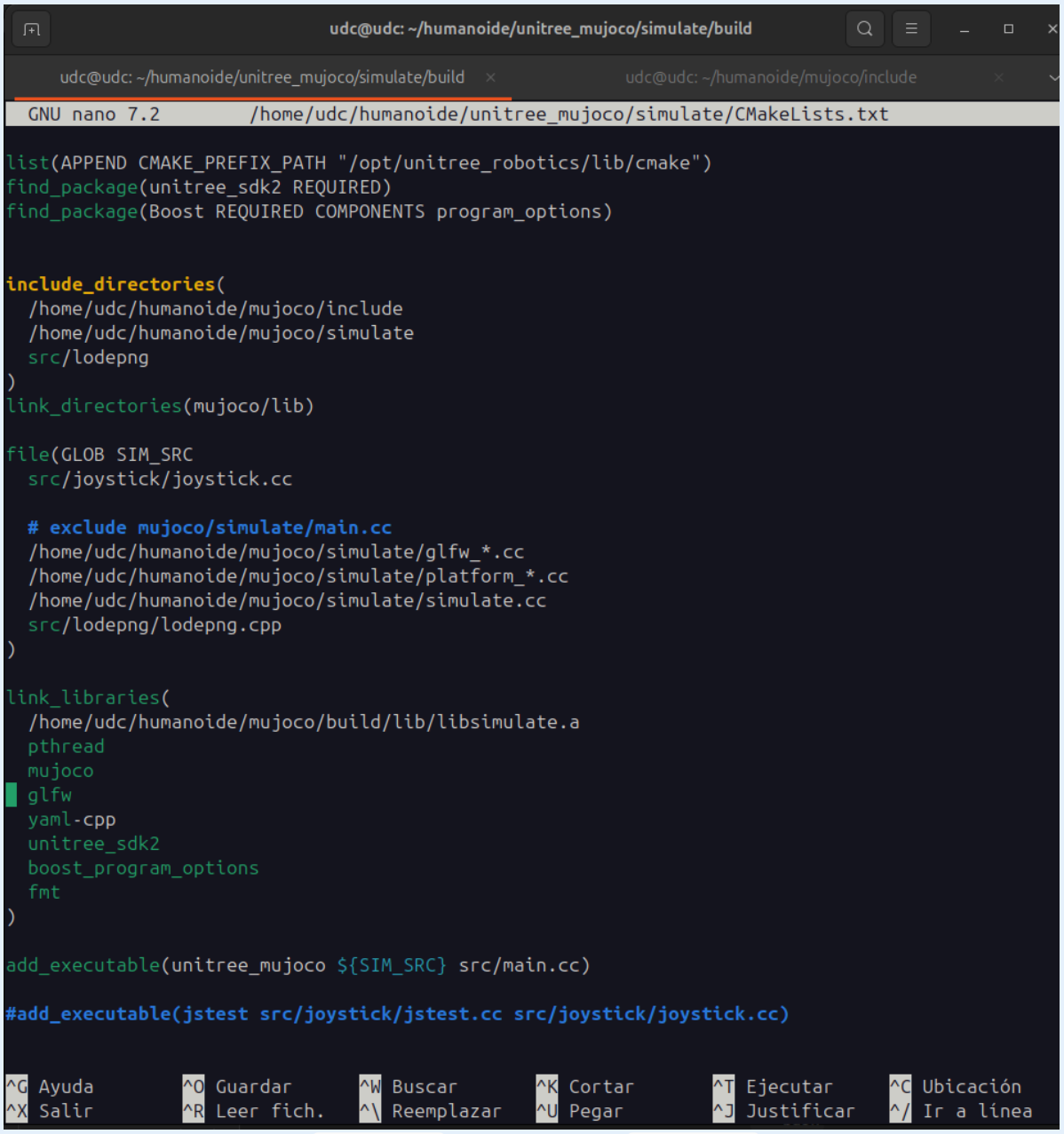
O

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

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

```
nano CMakelists.txt
```

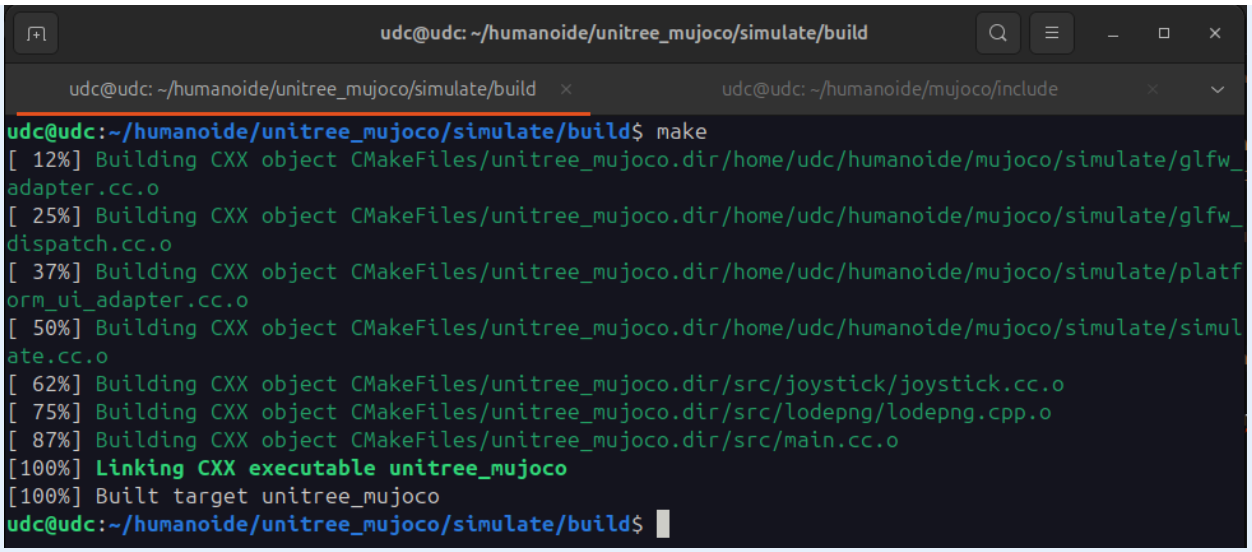
- Modificamos el archivo como se ve en la imagen



Guardamos con la tecla `CTRL + O` y salimos con `CTRL + X`

- Volvemos a ejecutar:

```
make
```



---

6 - Funcionalidad del entorno

Para esta funcionalidad debes contar con dos instanacias de terminal

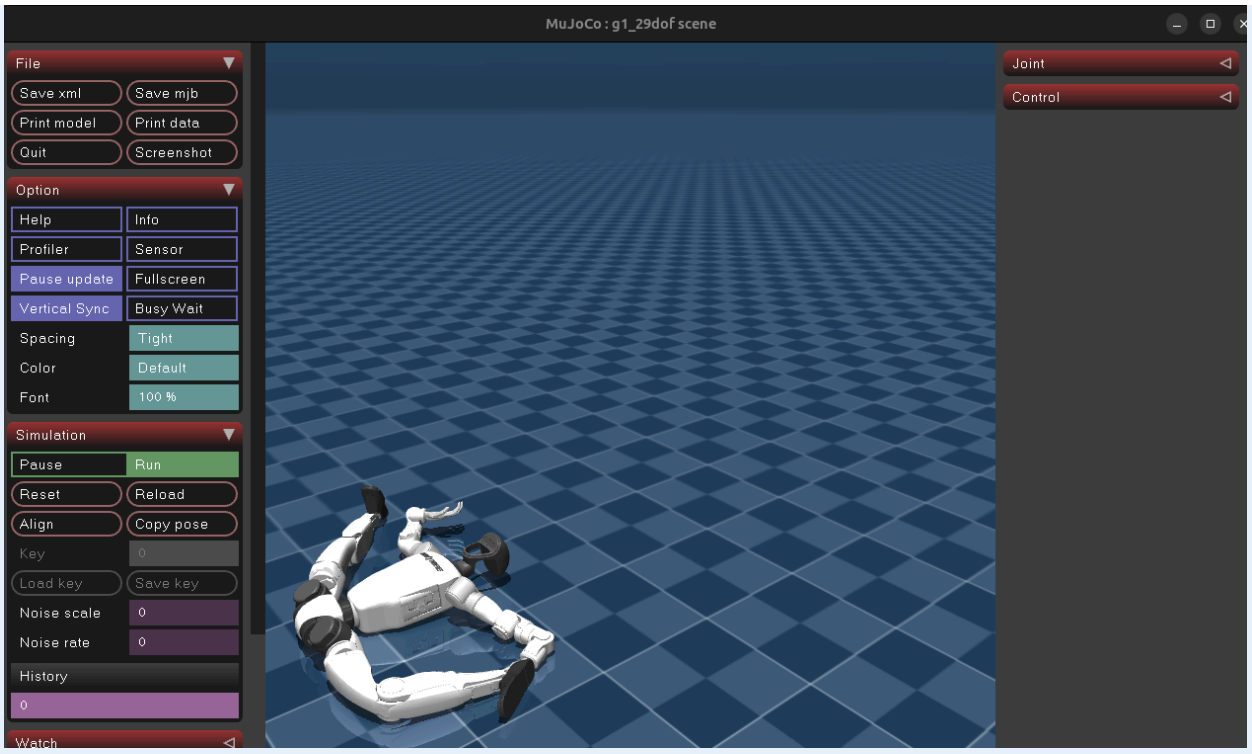
Terminal #1

La primera terminal se debe ubicar en la siguiente ruta `~/humanoide/unitre_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.

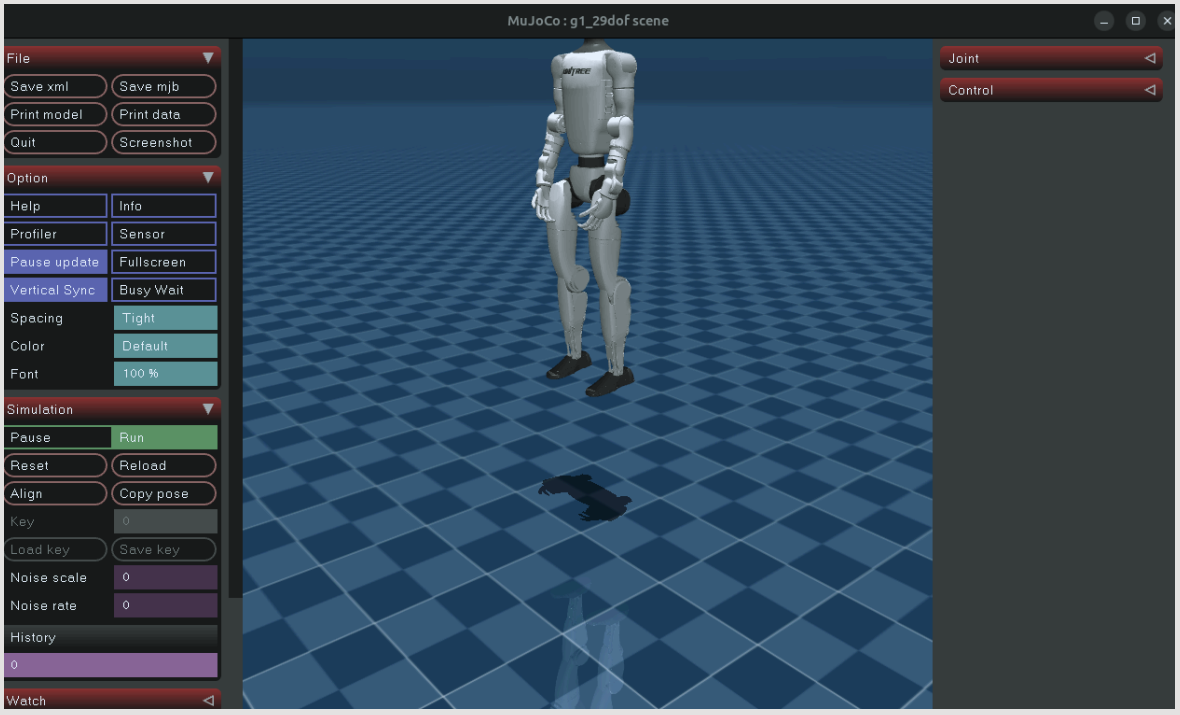
```
nano ~/humanoide/unitree_mujoco/simulate/config.yaml
```

```
  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.

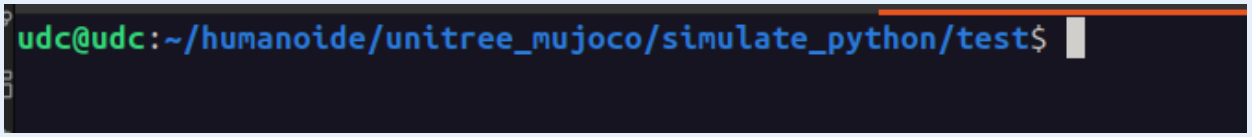
- Para subir, presionar las teclas `7` o ⬆️
- Para bajar, presionar las teclas `8` o ⬇️
- Para soltarlo de la banda, presionar `9` o `Delete/Backspace`.

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

🖊 Terminal #2

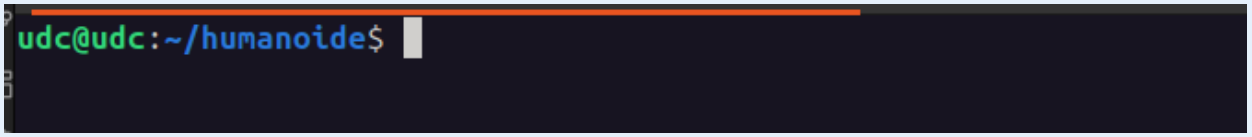
La segunda instanacia de terminal se debe ubicar en la siguiente ubicación `~/humanoide/unitre_mujoco/simulate_python/test`

> **⚠ Tener en cuenta**
>
> Tener la primera instancia de terminal como se describio anteriormente, debe estar activo.



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

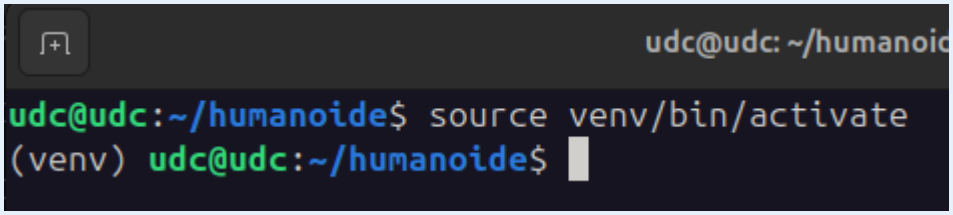
- Activamos el entorno que habiamos craeado, nos movemos a la carpeta `humanoide`



- Activamos el entorno

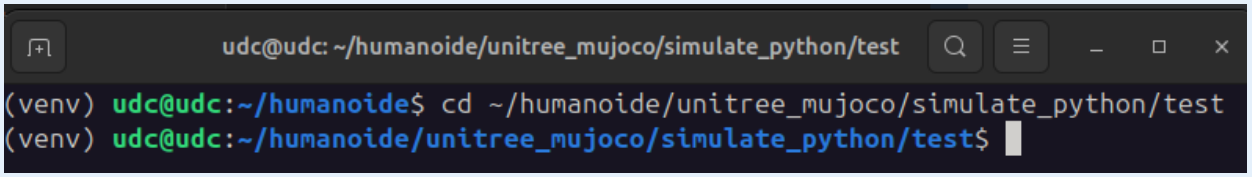
```
source venv/bin/activate
```

- Visualizamos que el entorno se activo



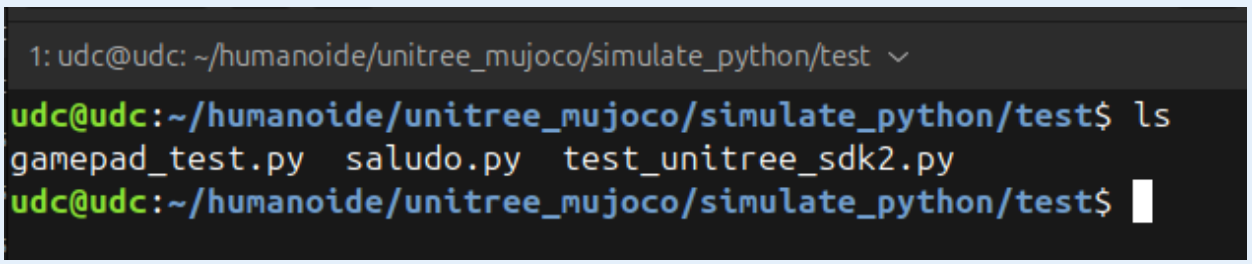
- Y nos ubicamos nuevamente en la carpeta `~/humanoide/unitree_mujoco/simulate_python/test`

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



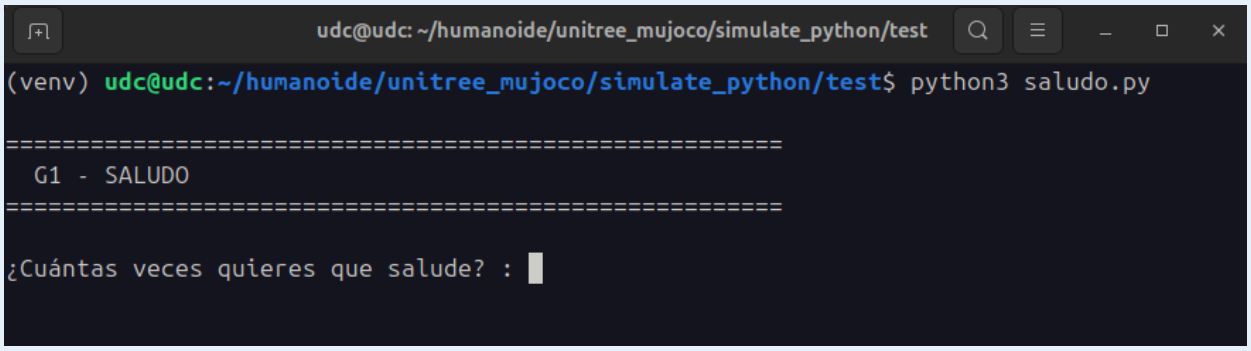
⚠ **Creamos un archivo, para el ejemplo `saludo.py` dentro de la carpeta `test`, y le añadimos el codigo que se encuentra al final de la documentación**

- Visualizo el contenido de la carpeta



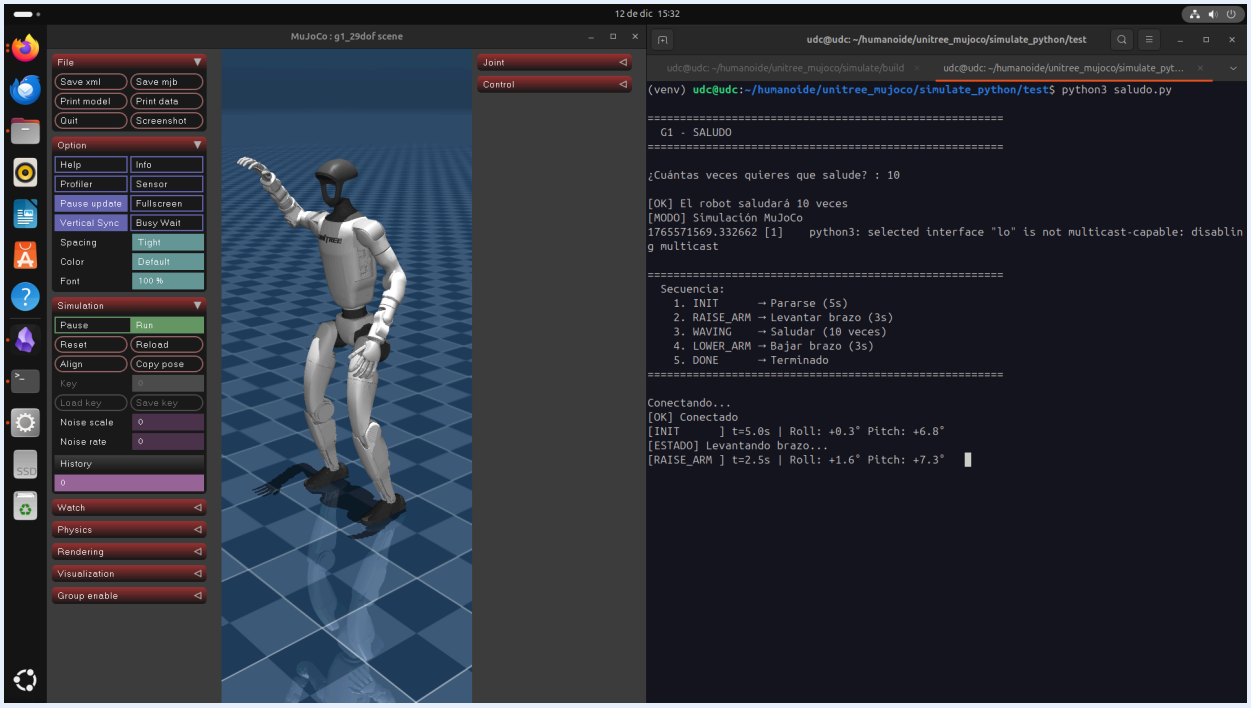
- Y ejecutamo de la siguiente manera

```
python3 saludo.py
```



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