Linux Basics Introduction

Linux is an open-source operating system whose kernel was created by Linus Torvalds in 1991.

Many companies and organizations develop their own Linux distributions based on the Linux operating system, such as Google, Red Hat, Ubuntu, etc.

Currently, the ROS system mainly runs on the Ubuntu operating system, which is based on the Debian Linux distribution.

Therefore, if you want to deeply learn and practice robotics, it's best to install the Ubuntu operating system on your computer using WSL, or a virtual machine like VirtualBox or VMware.

More importantly, most current artificial intelligence environments are based on the Ubuntu operating system, so for learning and practicing robotics, it's best to work in an Ubuntu environment.

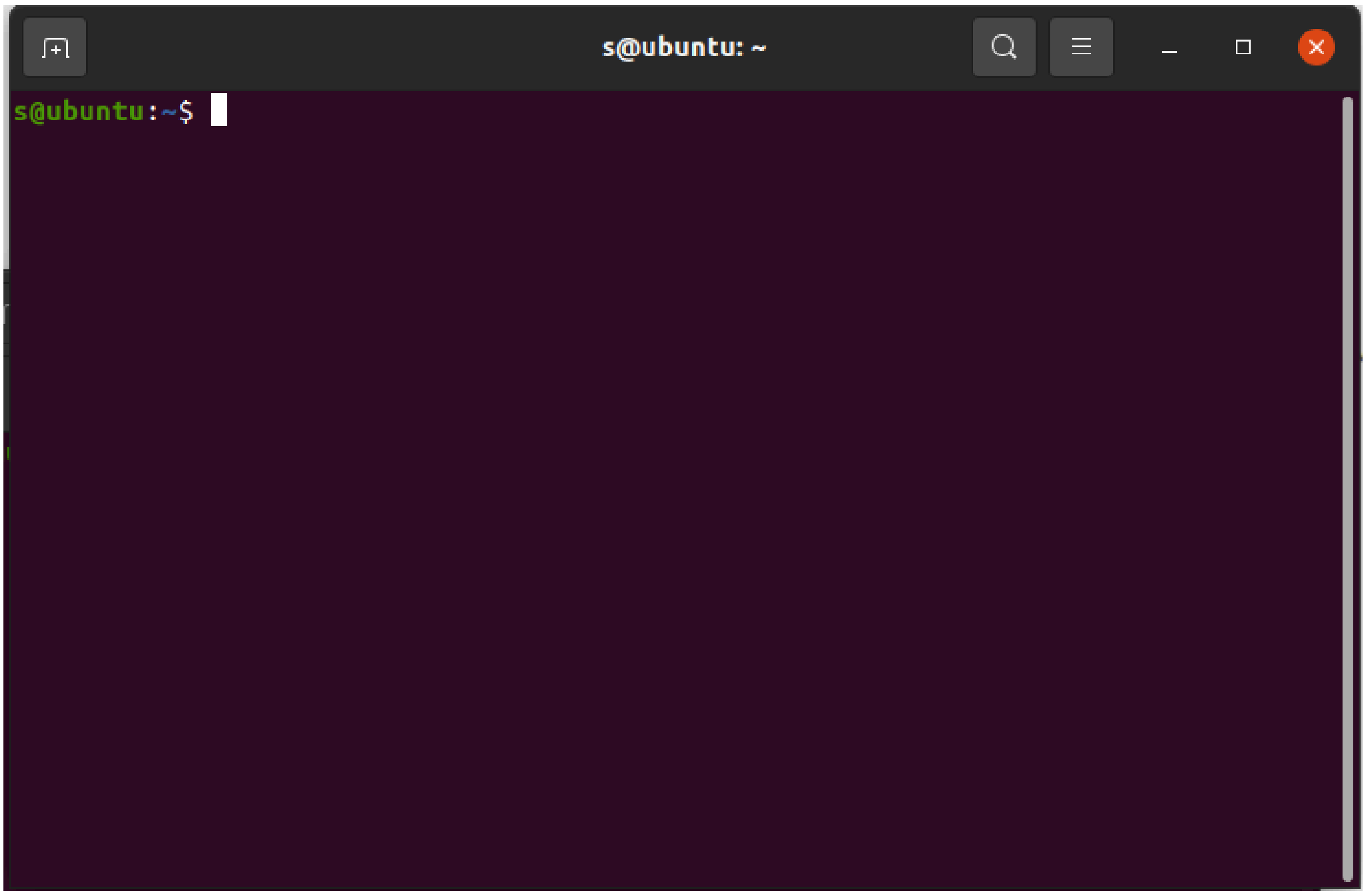
Terminal

First, we need to understand one thing: the terminal.

The terminal is a text interface where users can input commands, and the operating system will execute corresponding operations based on the commands entered by the user.

In the Ubuntu operating system, the terminal is a very important tool. Users can execute various commands in the terminal, such as installing software, configuring the system, running programs, etc.

Press the ctrl key, alt key, and t key simultaneously to summon a terminal interface:



```
s@ubuntu:~$ 
```

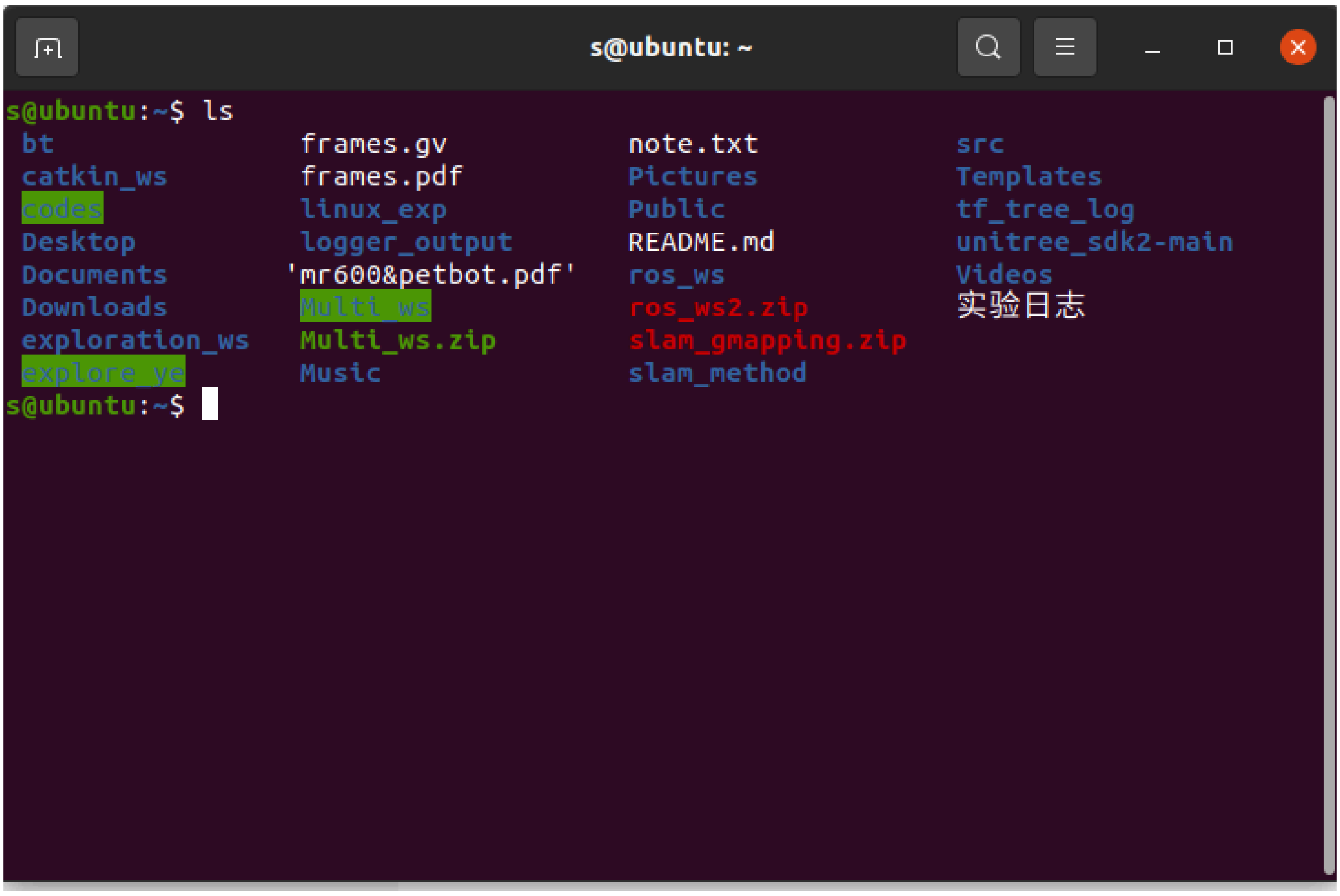
In this image, s is the username, the part after the @ separator is the hostname ubuntu, : is a separator symbol, and ~ represents the current user's home directory, which is also the current path where the terminal is located.

You can input some commands in this black box to perform corresponding operations. For example, input the ls command to list the files and folders in the current directory.

Or input the cd command to switch the current directory. For example, input

```
cd ~/
```

to switch to the current user's home directory, which is the default path when opening the terminal.



```
s@ubuntu:~$ ls
 bt                  frames.gv             note.txt             src
 catkin_ws           frames.pdf            Pictures             Templates
 codes               linux_exp             Public               tf_tree_log
 Desktop             logger_output         README.md            unitree_sdk2-main
 Documents          'mr600&petbot.pdf'     ros_ws               Videos
 Downloads           Multi_ws              ros_ws2.zip          实验日志
 exploration_ws      Multi_ws.zip          slam_gmapping.zip
 explore_ye          Music                 slam_method
s@ubuntu:~$ 
```

Linux Basic Commands Mini-Experiment (Follow Step by Step)

Experiment Safety Notice (Very Important)

In this experiment, you will encounter **commands that actually modify files and directories**. Please read carefully:

- `rm` **and** `mv` **directly delete or move files**
- `rm -rf` **is extremely dangerous; once the path is wrong, data cannot be recovered**
- **This experiment only allows operations under** `~` **(your home directory)**
- **It is strictly forbidden to execute** `rm -rf` **in system directories such as** `/`**,** `/home`**,** `/usr`
- **You will be responsible for repair or compensation costs if the system is damaged due to misoperation**

Feel free to ask questions if you don't understand

I. Experiment Objectives

Through a complete mini-experiment, master the following:

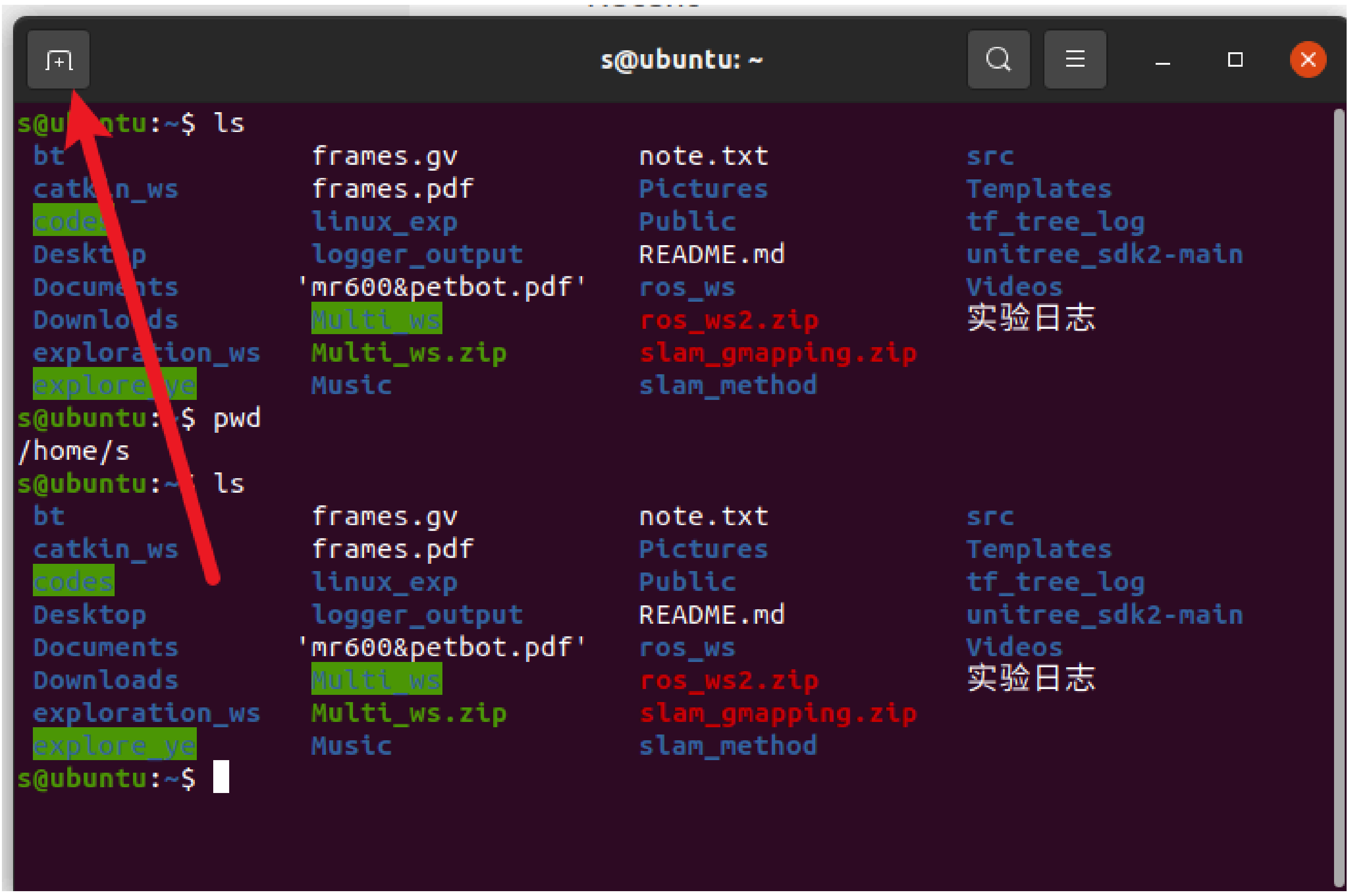
- **Path concepts** in the Linux terminal
- Common file/directory operation commands: `ls` `mkdir` `touch` `cp` `mv` `rm` `find` `cat`
- Using `gedit` to create and edit files
- Using different methods to execute Python programs, understanding:
  - Relative paths
  - Absolute paths
  - `~` (home directory)
- Understanding basic system and network commands: `ping`, `top`
- Learning to execute a **complete automation script**

Tips

Copy and Paste: To copy in the terminal, use `ctrl + shift + c`. To paste, use `ctrl + shift + v`. If you press `ctrl + v` in the terminal, invisible characters will appear, and you'll need to press backspace twice to delete them.

Closing Programs: In the terminal, close programs using `ctrl + c`, and force terminate programs using `ctrl + z`.

Adding a New Terminal: Click the plus sign to add a new terminal:



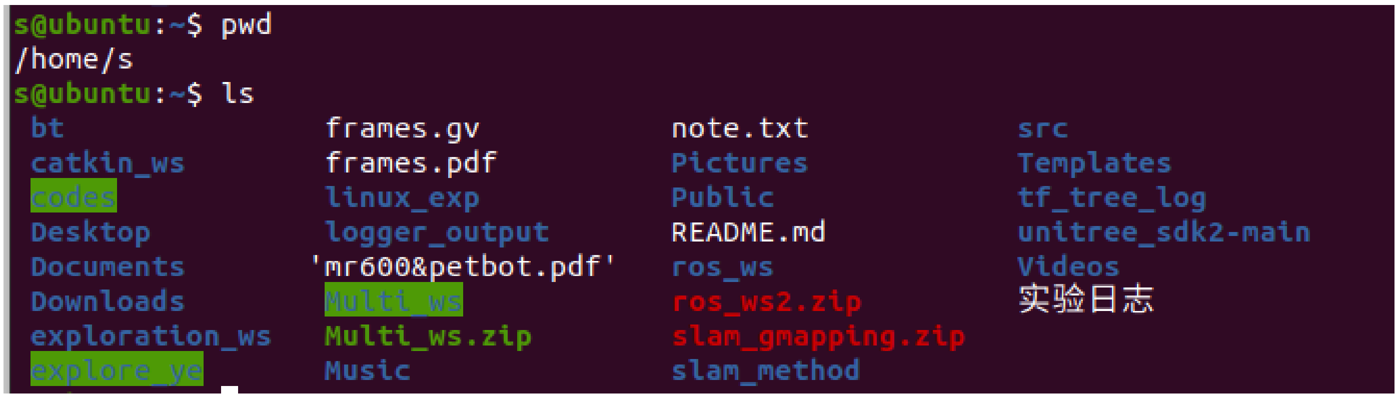
```
s@ubuntu:~$ ls
 bt                 frames.gv            note.txt             src
 catkin_ws          frames.pdf           Pictures             Templates
 codes              linux_exp            Public               tf_tree_log
 Desktop            logger_output        README.md            unitree_sdk2-main
 Documents         'mr600&petbot.pdf'    ros_ws               Videos
 Downloads          Multi_ws             ros_ws2.zip          实验日志
 exploration_ws     Multi_ws.zip         slam_gmapping.zip
 explore_ye         Music                slam_method
s@ubuntu:~$ pwd
/home/s
s@ubuntu:~$ ls
 bt                 frames.gv            note.txt             src
 catkin_ws          frames.pdf           Pictures             Templates
 codes              linux_exp            Public               tf_tree_log
 Desktop            logger_output        README.md            unitree_sdk2-main
 Documents         'mr600&petbot.pdf'    ros_ws               Videos
 Downloads          Multi_ws             ros_ws2.zip          实验日志
 exploration_ws     Multi_ws.zip         slam_gmapping.zip
 explore_ye         Music                slam_method
s@ubuntu:~$
```

II. Experiment 1: Manual Operations

1. Check Current Location

Confirm you are currently in your **user directory (~)**.

```
pwd
ls
```



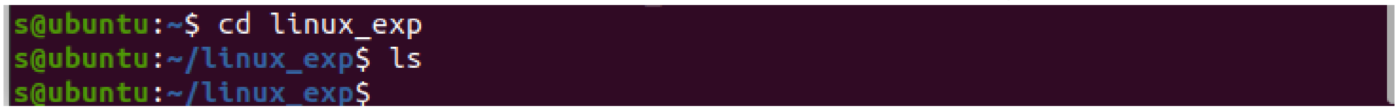
```
s@ubuntu:~$ pwd
/home/s
s@ubuntu:~$ ls
 bt                 frames.gv            note.txt             src
 catkin_ws          frames.pdf           Pictures             Templates
 codes              linux_exp            Public               tf_tree_log
 Desktop            logger_output        README.md            unitree_sdk2-main
 Documents         'mr600&petbot.pdf'    ros_ws               Videos
 Downloads          Multi_ws             ros_ws2.zip          实验日志
 exploration_ws     Multi_ws.zip         slam_gmapping.zip
 explore_ye         Music                slam_method
```

---

2. Create Experiment Workspace

```
mkdir linux_exp
cd linux_exp
ls
```

You can observe that the path before the $ symbol has changed.



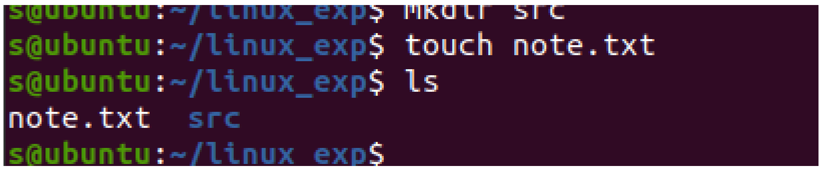
```
s@ubuntu:~$ cd linux_exp
s@ubuntu:~/linux_exp$ ls
s@ubuntu:~/linux_exp$
```

---

3. Create Files and Directories

In Windows, we use right-click to create files/folders, and in the terminal, we have corresponding commands:

```
mkdir src
touch note.txt
ls
```



```
s@ubuntu:~/linux_exp$ mkdir src
s@ubuntu:~/linux_exp$ touch note.txt
s@ubuntu:~/linux_exp$ ls
note.txt  src
s@ubuntu:~/linux_exp$
```

---

4. Use gedit to Create a Python File

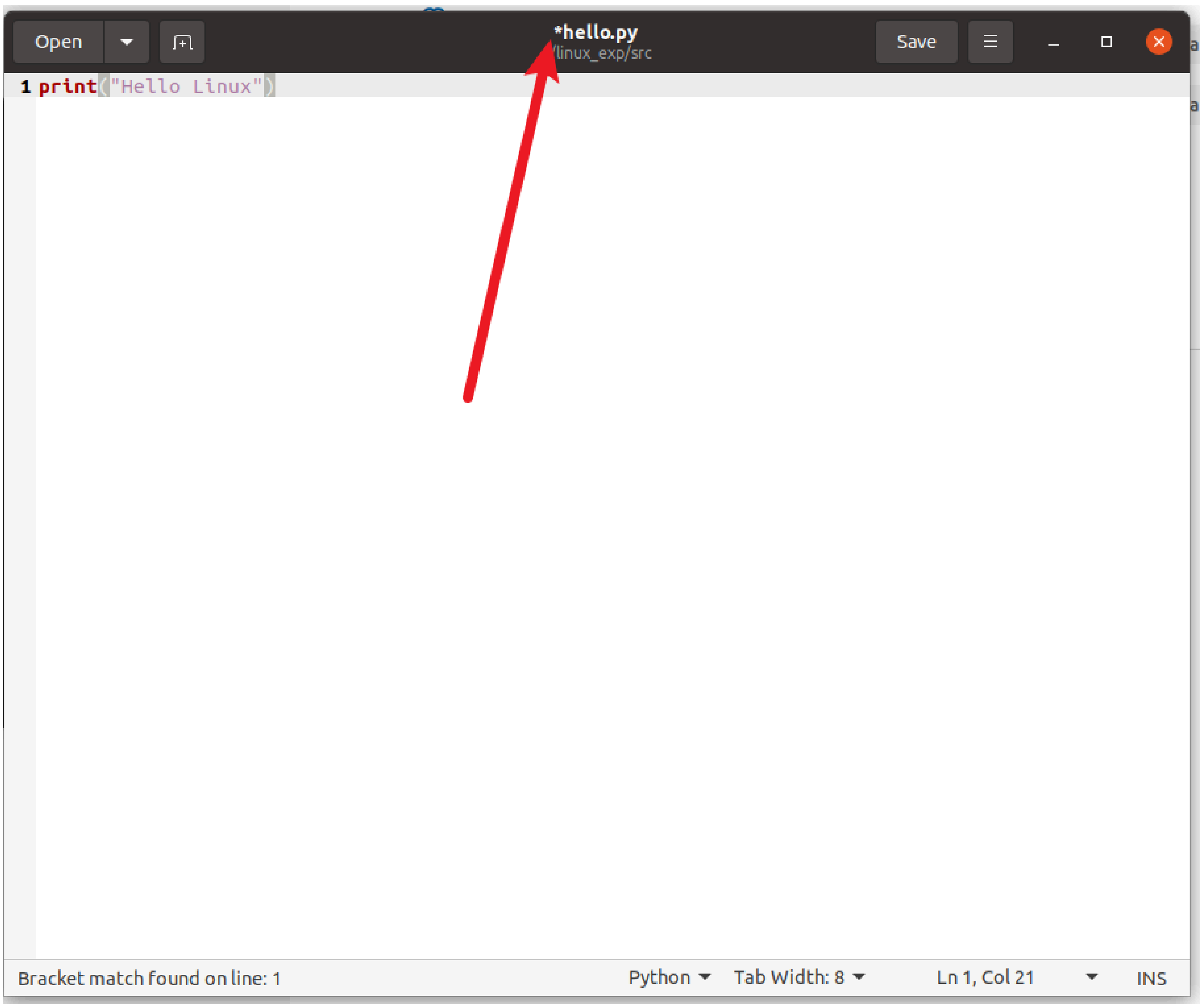
gedit is a text editor:

```
gedit src/hello.py
```

In the opened editor, input and save:

```
print("Hello Linux")
```

The effect is shown below:

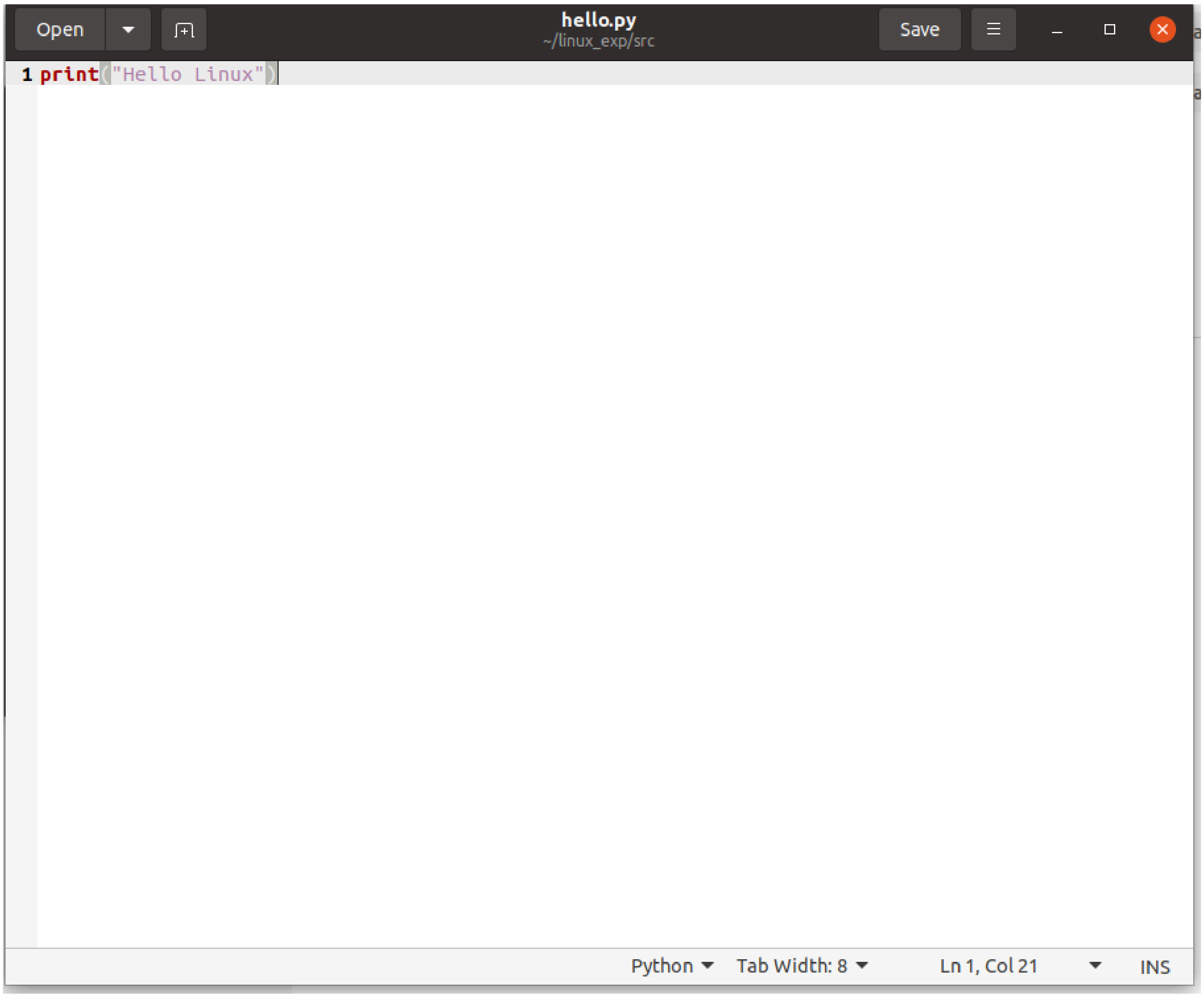


```python
print("Hello Linux")
```

Bracket match found on line: 1

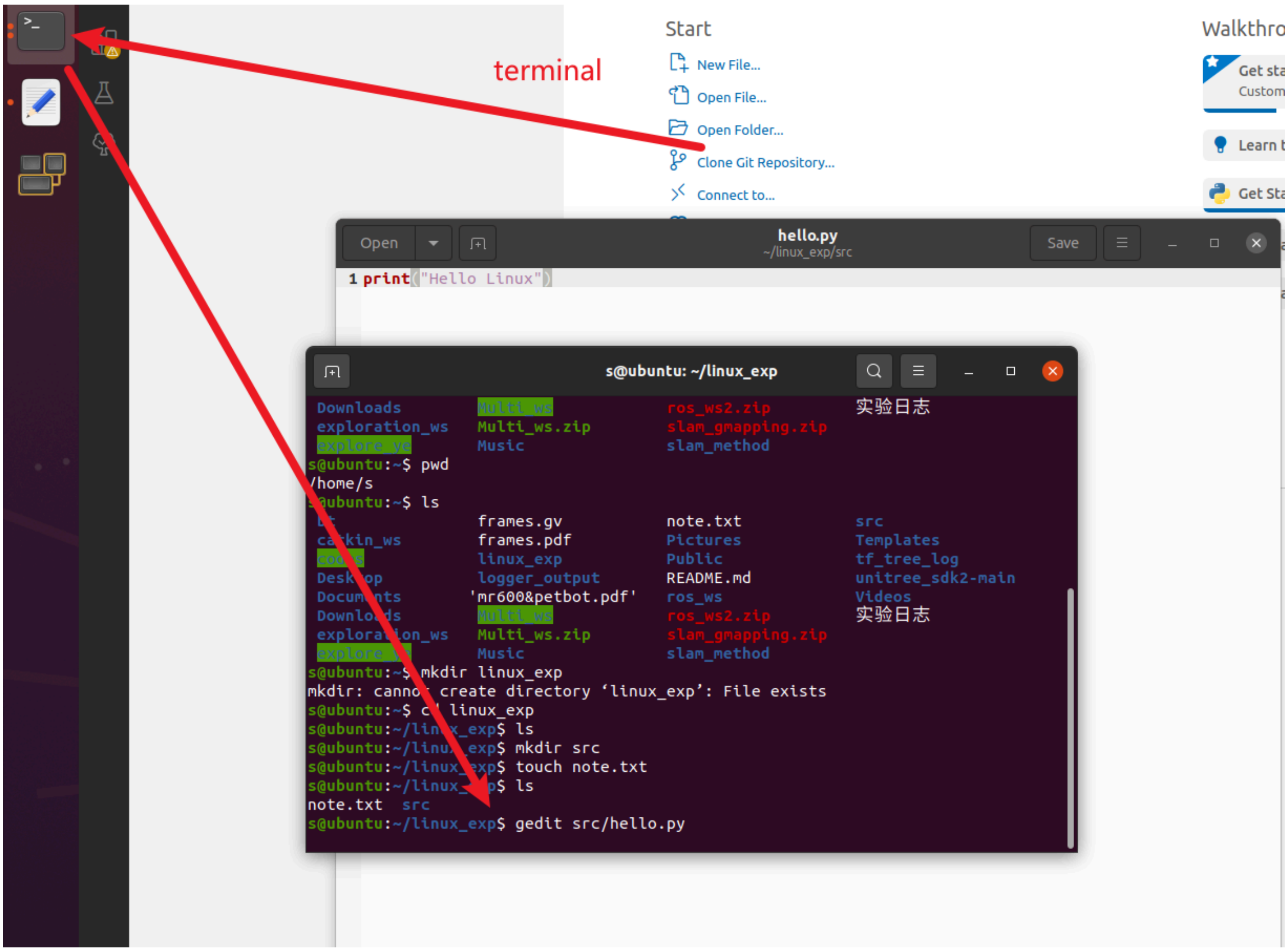
Note that you need to save with `ctrl + s`. The * symbol here indicates there are unsaved modifications.

After saving, the effect is as shown:



```
print("Hello Linux")
```

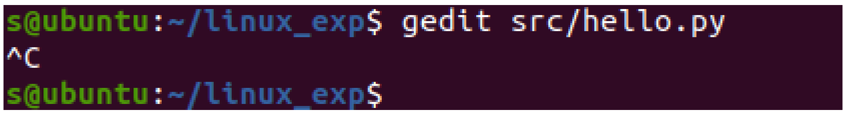
Click the terminal icon on the left.



```
1 print("Hello Linux")
```

```
Downloads          Multi_ws             ros_ws2.zip          实验日志
exploration_ws     Multi_ws.zip         slam_gmapping.zip
explore_ye         Music                slam_method
s@ubuntu:~$ pwd
/home/s
s@ubuntu:~$ ls
[illegible]                 frames.gv            note.txt             src
catkin_ws          frames.pdf           Pictures             Templates
codes              linux_exp            Public               tf_tree_log
Desktop            logger_output        README.md            unitree_sdk2-main
Documents         'mr600&petbot.pdf'    ros_ws               Videos
Downloads          Multi_ws             ros_ws2.zip          实验日志
exploration_ws     Multi_ws.zip         slam_gmapping.zip
explore_ye         Music                slam_method
s@ubuntu:~$ mkdir linux_exp
mkdir: cannot create directory ‘linux_exp’: File exists
s@ubuntu:~$ cd linux_exp
s@ubuntu:~/linux_exp$ ls
s@ubuntu:~/linux_exp$ mkdir src
s@ubuntu:~/linux_exp$ touch note.txt
s@ubuntu:~/linux_exp$ ls
note.txt  src
s@ubuntu:~/linux_exp$ gedit src/hello.py
```

You can see that below the gedit command, there is no new place to input commands. This is because the gedit text editor is running. You can press `ctrl` and `c` simultaneously in the terminal to close it.



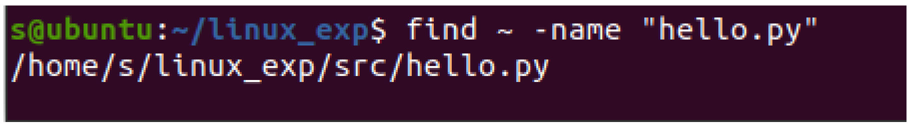
```
s@ubuntu:~/linux_exp$ gedit src/hello.py
^C
s@ubuntu:~/linux_exp$
```

---

5. Use find to Search for Files

We can search for files we want in folders

```
find ~ -name "hello.py"
```

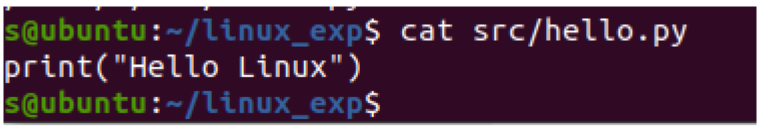


```
s@ubuntu:~/linux_exp$ find ~ -name "hello.py"
/home/s/linux_exp/src/hello.py
```

You can observe the path of this file. It starts with `/`, meaning this is an absolute path.

6. View File Contents

```
cat src/hello.py
```



```
s@ubuntu:~/linux_exp$ cat src/hello.py
print("Hello Linux")
s@ubuntu:~/linux_exp$
```

7. Copy, Move, Delete Files (Use Caution)

Copy

```
cp src/hello.py hello_copy.py
ls
```

Move/Rename

```
mv hello_copy.py hello_moved.py
ls
```

Delete

```
rm hello_moved.py
ls
```

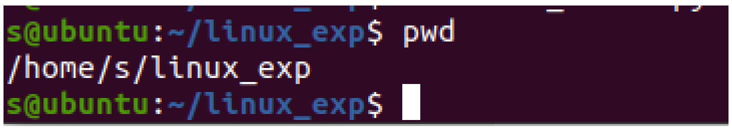
You can observe the effects of these commands on files.

Note: **Do not use `rm -rf /` or randomly delete directories**

III. Experiment 2: Execute Python, Understand Paths

Make sure you are still in the `~/linux_exp` directory:

```
pwd
```



```
s@ubuntu:~/linux_exp$ pwd
/home/s/linux_exp
s@ubuntu:~/linux_exp$ 
```

Method 1: Relative Path

```
python3 src/hello.py
```

Method 2: Home Path

```
python3 ~/linux_exp/src/hello.py
```

Method 3: Absolute Path

```
python3 /home/<username>/linux_exp/src/hello.py
```

> Please replace `<username>` with the current terminal username, which is the string before @

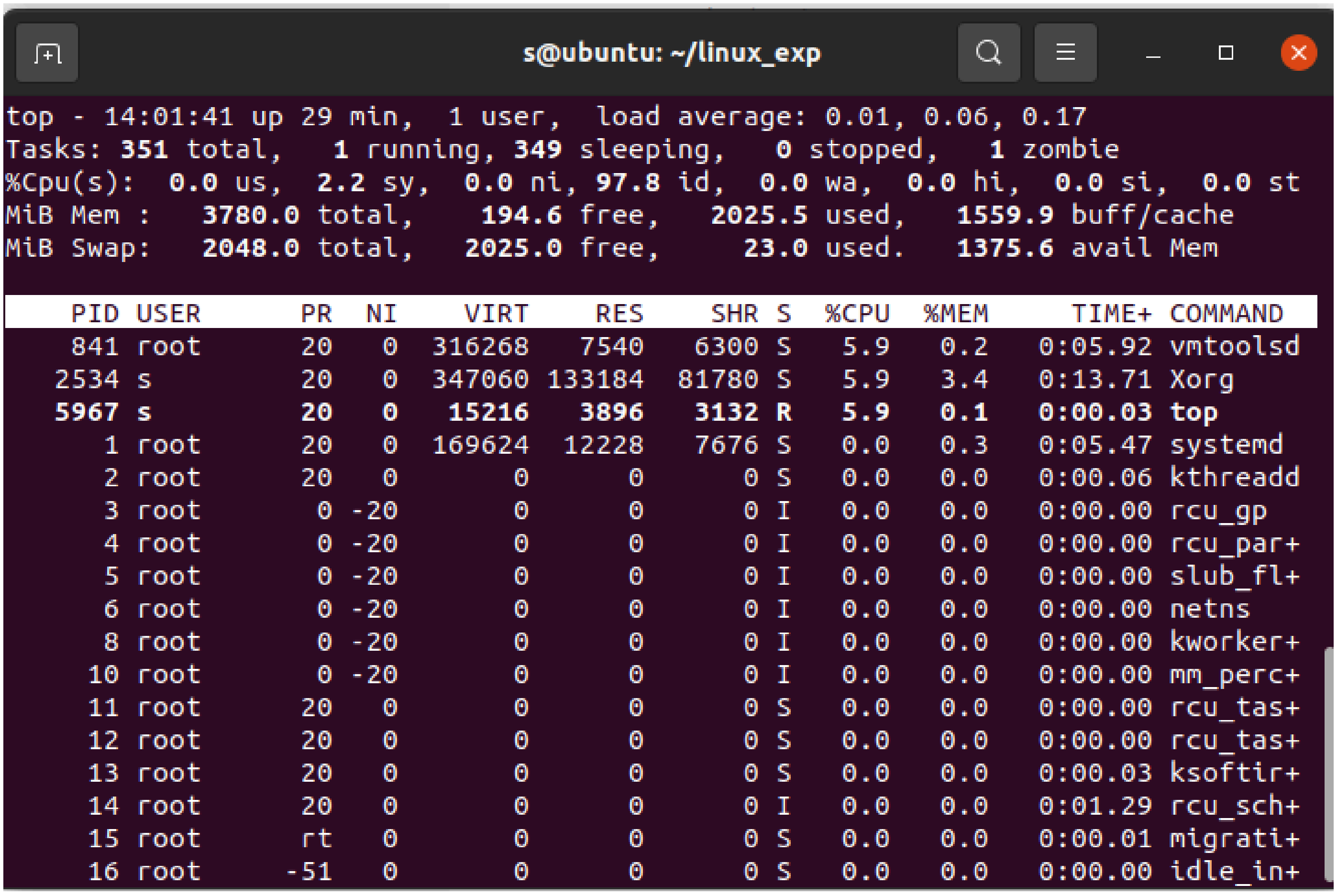
Reflection

- Why do all three methods work?
- After you `cd ~`, which ones will still work?

IV. Experiment 3: System and Network Commands (Observation Only)

View Processes (Press q to exit)

```
top
```

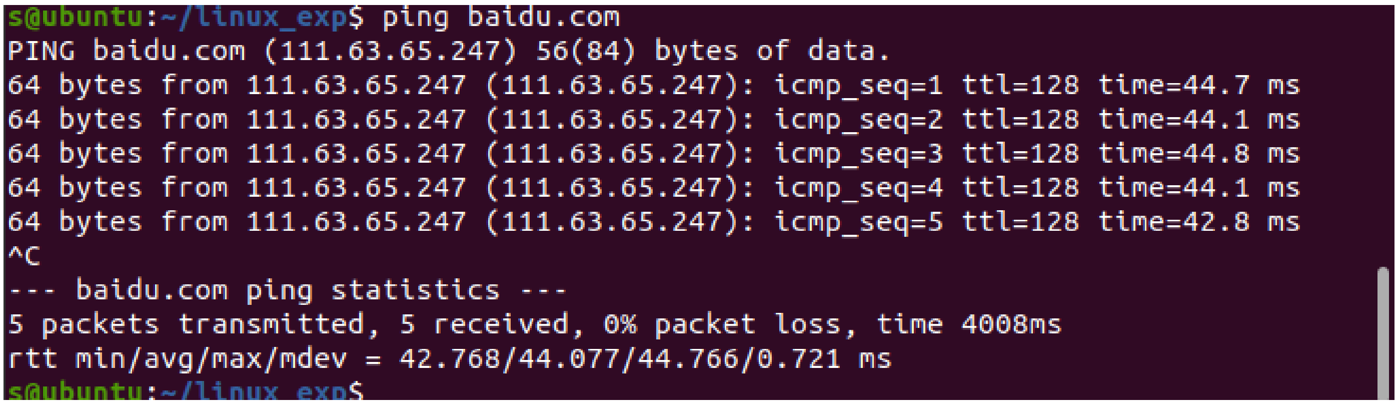

s@ubuntu: ~/linux_exp

```
top - 14:01:41 up 29 min,  1 user,  load average: 0.01, 0.06, 0.17
Tasks: 351 total,   1 running, 349 sleeping,   0 stopped,   1 zombie
%Cpu(s):  0.0 us,  2.2 sy,  0.0 ni, 97.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3780.0 total,    194.6 free,   2025.5 used,   1559.9 buff/cache
MiB Swap:   2048.0 total,   2025.0 free,     23.0 used.   1375.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    841 root      20   0  316268   7540   6300 S   5.9   0.2   0:05.92 vmtoolsd
   2534 s         20   0  347060 133184  81780 S   5.9   3.4   0:13.71 Xorg
   5967 s         20   0   15216   3896   3132 R   5.9   0.1   0:00.03 top
      1 root      20   0  169624  12228   7676 S   0.0   0.3   0:05.47 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.06 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par+
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_fl+
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_perc+
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tas+
     12 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tas+
     13 root      20   0       0      0      0 S   0.0   0.0   0:00.03 ksoftir+
     14 root      20   0       0      0      0 I   0.0   0.0   0:01.29 rcu_sch+
     15 root      rt   0       0      0      0 S   0.0   0.0   0:00.01 migrati+
     16 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_in+
```

Test Network (Ctrl + C to stop)

Sometimes we use this command to test whether the host can access the internet.

```
ping baidu.com
```



```
s@ubuntu:~/linux_exp$ ping baidu.com
PING baidu.com (111.63.65.247) 56(84) bytes of data.
64 bytes from 111.63.65.247 (111.63.65.247): icmp_seq=1 ttl=128 time=44.7 ms
64 bytes from 111.63.65.247 (111.63.65.247): icmp_seq=2 ttl=128 time=44.1 ms
64 bytes from 111.63.65.247 (111.63.65.247): icmp_seq=3 ttl=128 time=44.8 ms
64 bytes from 111.63.65.247 (111.63.65.247): icmp_seq=4 ttl=128 time=44.1 ms
64 bytes from 111.63.65.247 (111.63.65.247): icmp_seq=5 ttl=128 time=42.8 ms
^C
--- baidu.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 42.768/44.077/44.766/0.721 ms
s@ubuntu:~/linux_exp$
```

V. Experiment 4: One-Click Automation Script (Key Point)

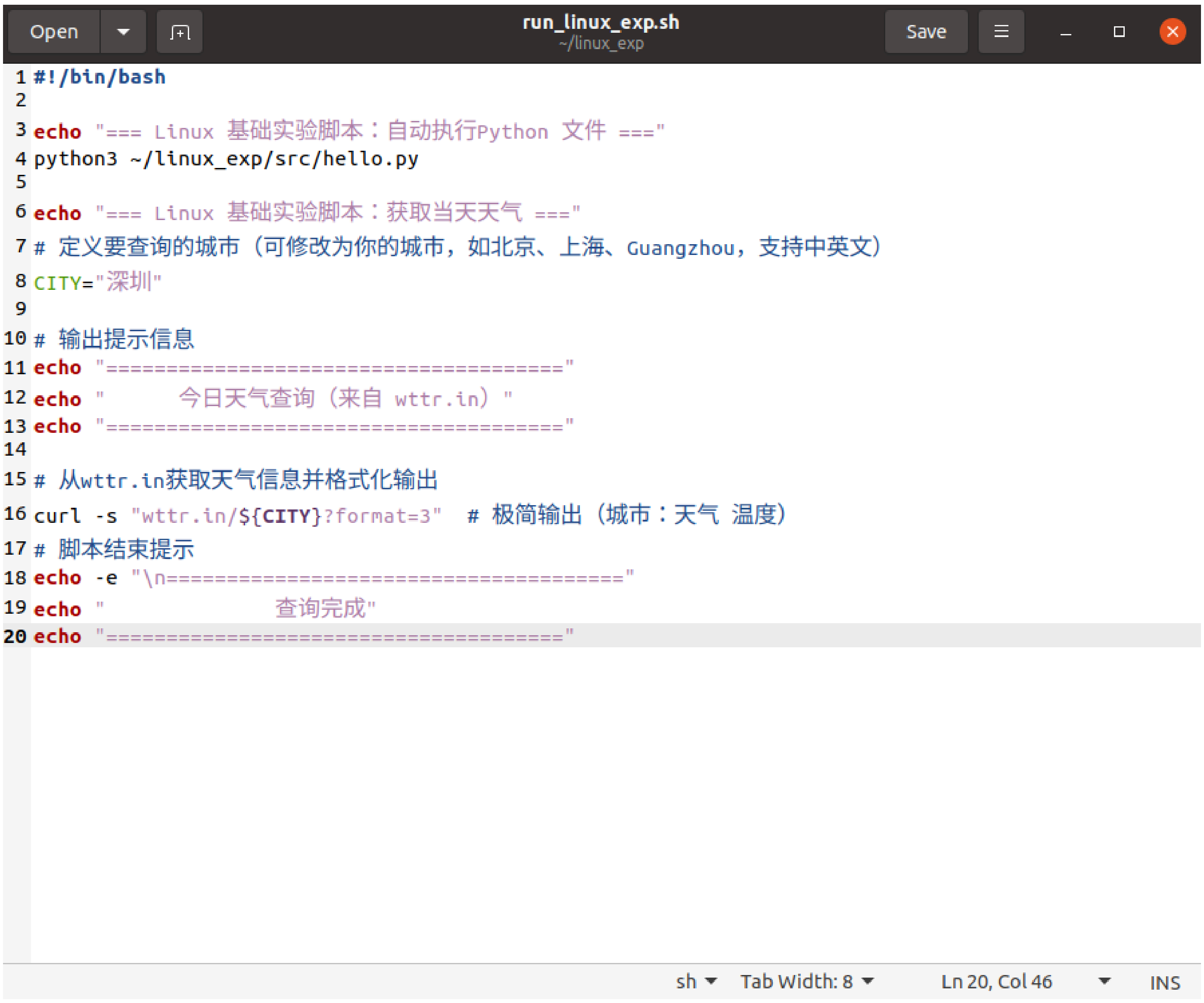
We can write a script to automatically get today's weather in Shenzhen from the internet and execute the Python code we wrote above.

1️⃣ Create Script File

```
cd ~/linux_exp
gedit run_linux_exp.sh
```

Write the following content and save:

```
#!/bin/bash

echo "=== Linux Basic Experiment Script: Automatically Execute Python File ==="
python3 ~/linux_exp/src/hello.py

echo "=== Linux Basic Experiment Script: Get Today's Weather ==="
# Define the city to query (can be modified to your city, such as Beijing,
Shanghai, Guangzhou; supports Chinese and English)
CITY="Shenzhen"

# Output prompt information
echo "======================================"
echo "      Today's Weather Query (from wttr.in)"
echo "======================================"

# Get weather information from wttr.in and format output
curl -s "wttr.in/${CITY}?format=3"  # Minimal output (City: Weather Temperature)
# Script completion prompt
echo -e "\n======================================"
echo "              Query Complete"
echo "======================================"
```



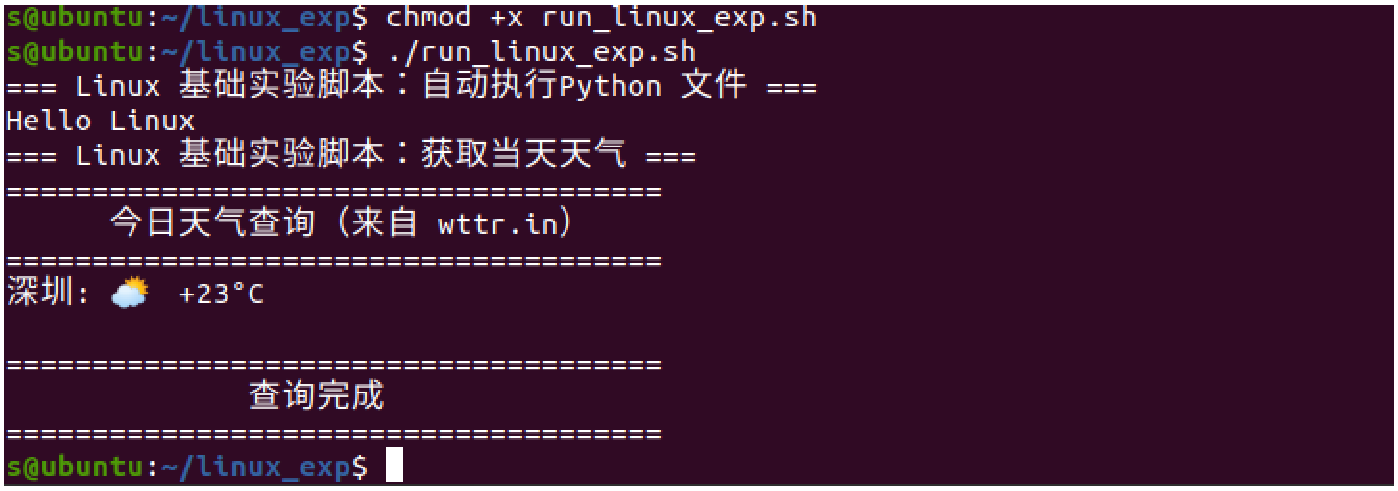
```
#!/bin/bash

echo "=== Linux 基础实验脚本：自动执行Python 文件 ==="
python3 ~/linux_exp/src/hello.py

echo "=== Linux 基础实验脚本：获取当天天气 ==="
# 定义要查询的城市（可修改为你的城市，如北京、上海、Guangzhou，支持中英文）
CITY="深圳"

# 输出提示信息
echo "======================================="
echo "      今日天气查询（来自 wttr.in） "
echo "======================================="

# 从wttr.in获取天气信息并格式化输出
curl -s "wttr.in/${CITY}?format=3"  # 极简输出（城市：天气 温度）
# 脚本结束提示
echo -e "\n======================================="
echo "              查询完成"
echo "======================================="
```

2 Add Execute Permission

```
chmod +x run_linux_exp.sh
```

3 Execute Script

```
./run_linux_exp.sh
```



```
s@ubuntu:~/linux_exp$ chmod +x run_linux_exp.sh
s@ubuntu:~/linux_exp$ ./run_linux_exp.sh
=== Linux 基础实验脚本：自动执行Python 文件 ===
Hello Linux
=== Linux 基础实验脚本：获取当天天气 ===
======================================
      今日天气查询（来自 wttr.in）
======================================
深圳: ⛅  +23°C

======================================
              查询完成
======================================
s@ubuntu:~/linux_exp$
```

We can see that scripts are particularly powerful tools that can accomplish many tasks through Linux commands.

VI. Experiment Summary

In this experiment, you actually used and understood:

📌 **Three Path Writing Methods**

- **Absolute path** `/home/<username>/linux_exp/src/hello.py`
- **Relative path** `src/hello.py`
- **Home path** `~/linux_exp/src/hello.py`

👉 **Whether a command succeeds depends on: where you are + how you write the path**

VII. Checklist (Self-Check)

- ☐ I know what `pwd` does
- ☐ I won't randomly use `rm -rf`
- ☐ I can understand every line of commands in the script
- ☐ I understand why the same Python file can be executed in multiple ways

VIII. Linux Command Learning Resources

- linux-command-manual
- geeksforgeeks - Linux Commands
- The Linux command line for beginners