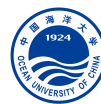


海纳百川
取则行远



中国海洋大学
OCEAN UNIVERSITY OF CHINA

Week3

左昊天

2024-09-06

Github 仓库地址

1 命令行环境

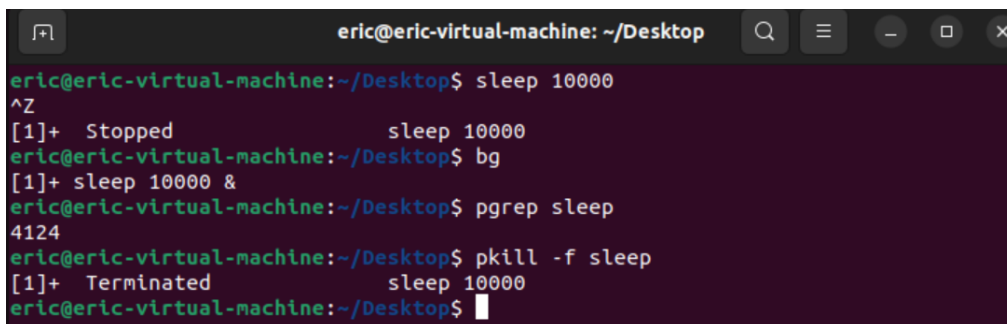
1.1 暂停和后台执行进程

```
eric@eric-virtual-machine:~$ sleep 1000
^Z
[1]+  Stopped                  sleep 1000
eric@eric-virtual-machine:~$ nohup sleep 2000&
[2] 2812
eric@eric-virtual-machine:~$ nohup: ignoring input and appending output to 'nohup.out'
jobs
[1]+  Stopped                  sleep 1000
[2]-  Running                  nohup sleep 2000 &
eric@eric-virtual-machine:~$ bg %1
[1]+ sleep 1000 &
eric@eric-virtual-machine:~$ jobs
[1]-  Running                  sleep 1000 &
[2]+  Running                  nohup sleep 2000 &
eric@eric-virtual-machine:~$ kill -STOP %1
eric@eric-virtual-machine:~$ jobs
[1]+  Stopped                  sleep 1000
[2]-  Running                  nohup sleep 2000 &
eric@eric-virtual-machine:~$ kill -SIGHUP %1
[1]+  Stopped                  sleep 1000
eric@eric-virtual-machine:~$ jobs
[1]+  Hangup                   sleep 1000
[2]-  Running                  nohup sleep 2000 &
eric@eric-virtual-machine:~$ kill -SIGHUP %2
eric@eric-virtual-machine:~$ kill %2
eric@eric-virtual-machine:~$ jobs
[2]+  Terminated              nohup sleep 2000
```

解释:

- 1、sleep 1000 休眠 1000s
- 2、control + z 将当前正在运行的前台进程移动到后台, 同时暂停该进程的执行
- 3、& 将命令放到后台执行
- 4、bg %1 恢复第一个任务的执行
- 5、kill -STOP %1 暂停第一个任务
- 6、kill %2 终止第二个任务

1.2 我们可以使用类似 `ps aux | grep` 这样的命令来获取任务的 `pid` , 然后您可以基于 `pid` 来结束这些进程。但我们其实有更好的方法来做这件事。在终端中执行 `sleep 10000` 这个任务。然后用 `Ctrl-Z` 将其切换到后台并使用 `bg` 来继续允许它。现在, 使用 `pgrep` 来查找 `pid` 并使用 `pkill` 结束进程而不需要手动输入 `pid`。(提示: : 使用 `-af` 标记)。



```
eric@eric-virtual-machine: ~/Desktop
eric@eric-virtual-machine:~/Desktop$ sleep 10000
^Z
[1]+  Stopped                  sleep 10000
eric@eric-virtual-machine:~/Desktop$ bg
[1]+  sleep 10000 &
eric@eric-virtual-machine:~/Desktop$ pgrep sleep
4124
eric@eric-virtual-machine:~/Desktop$ pkill -f sleep
[1]+  Terminated              sleep 10000
eric@eric-virtual-machine:~/Desktop$
```

解释:

- 1、`pgrep sleep` 可以列出包含关键字 `sleep` 的进程的 `pid`
- 2、`pkill -f sleep` 可以终止包含关键字 `sleep` 的进程

1.3 终端多路复用

会话:

`tmux` 开始一个新的会话

`tmux new -s NAME` 以指定名称开始一个新的会话

`tmux ls` 列出当前所有会话

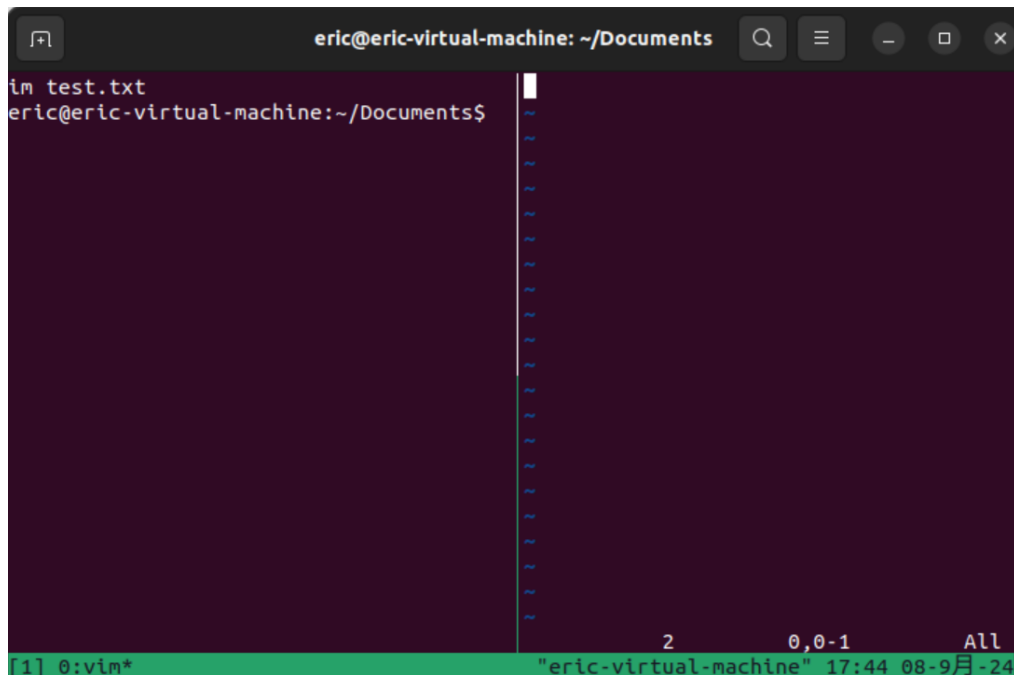
在 `tmux` 中输入 `<C-b> d` , 将当前会话分离

`tmux a` 重新连接最后一个会话。也可用 `-t` 来指定具体的会话

面板:

`<C-b> "` 水平分割

`<C-b> %` 垂直分割

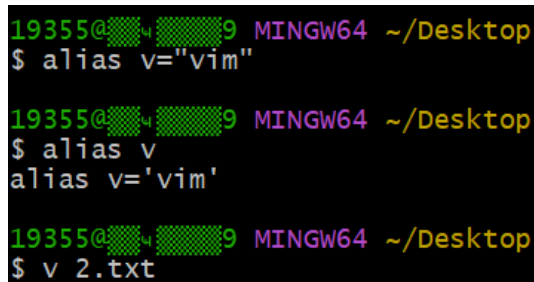


```
eric@eric-virtual-machine: ~/Documents
vim test.txt
eric@eric-virtual-machine:~/Documents$
```

[1] 0:vim* "eric-virtual-machine" 17:44 08-9月-24

1.4 别名

```
1 alias v="vim" #用于创建别名
2 alias v #用于查看别名的定义
3 v 2.txt #使用别名进行操作
```



```
19355@... MINGW64 ~/Desktop
$ alias v="vim"

19355@... MINGW64 ~/Desktop
$ alias v
alias v='vim'

19355@... MINGW64 ~/Desktop
$ v 2.txt
```

1.5 执行以下命令来获取您最常用的十条命令，尝试为它们创建别名。

命令：

```
1 history | awk '{ $1=""; print substr($0,2) }' | sort |
  uniq -c | sort -n | tail -n 10
```

```
$ history | awk '{s1="";print substr($0,2)}' | sort | uniq -c | sort -n | tail -n 10
  6 cd ../
  6 git branch
  6 git reflog
  6 jobs
  6 ssh-keygen -t ed25519 -C "Gitee SSH Key"
  7 git merge test
 10 cd
 10 git add .
 11 git push
 17 git status

19355@Mingw64: ~/Desktop
$ alias gb="git branch"

19355@Mingw64: ~/Desktop
$ alias j="jobs"

19355@Mingw64: ~/Desktop
$ alias gp="git push"

19355@Mingw64: ~/Desktop
$ alias gs="git status"

19355@Mingw64: ~/Desktop
$ gs
fatal: not a git repository (or any of the parent directories): .git
```

1.6 创建一个 dc 别名，它的功能是当我们错误的将 cd 输入为 dc 时也能正确执行。

```
eric@eric-virtual-machine:~/Desktop$ alias dc=cd
eric@eric-virtual-machine:~/Desktop$ dc /home/eric
eric@eric-virtual-machine:~$
```

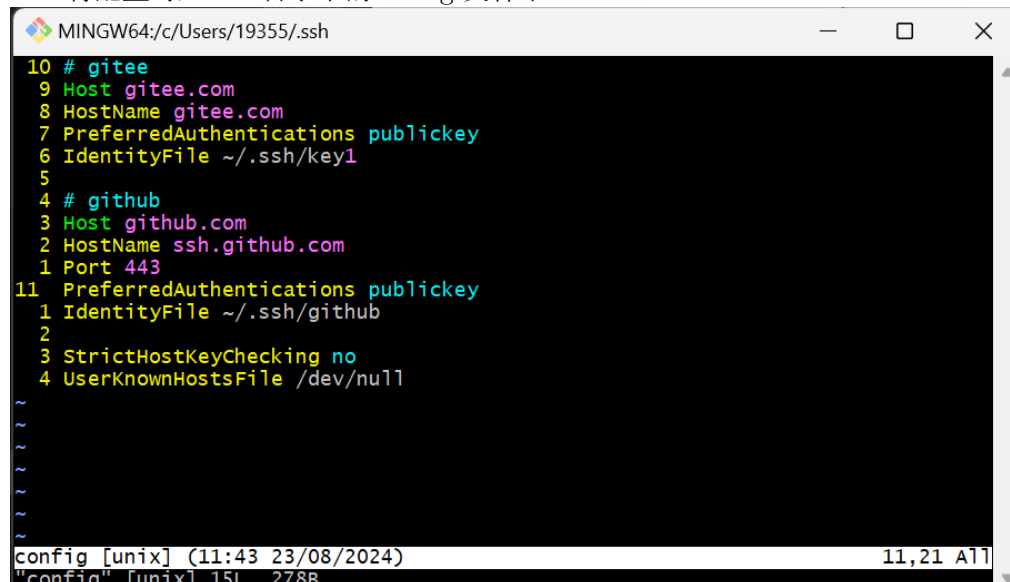
1.7 生成 ssh 秘钥

```
1 ssh-keygen -o -a 100 -t ed25519 -f ~/.ssh/id_ed25519
```

```
eric@eric-virtual-machine:~$ ssh-keygen -o -a 100 -t ed25519 -f ~/.ssh/id_ed25519
9
Generating public/private ed25519 key pair.
Created directory '/home/eric/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/eric/.ssh/id_ed25519
Your public key has been saved in /home/eric/.ssh/id_ed25519.pub
The key fingerprint is:
```

1.8 ssh 配置

将配置写入.ssh 目录下的 config 文件中



```
10 # gitee
9 Host gitee.com
8 HostName gitee.com
7 PreferredAuthentications publickey
6 IdentityFile ~/.ssh/key1
5
4 # github
3 Host github.com
2 HostName ssh.github.com
1 Port 443
11 PreferredAuthentications publickey
1 IdentityFile ~/.ssh/github
2
3 StrictHostKeyChecking no
4 UserKnownHostsFile /dev/null
~
~
~
~
~
~
config [unix] (11:43 23/08/2024) 11,21 All
"config" [unix] 15L, 278B
```

2 Python 基础

2.1 输出语句

2.1.1 单行输出

```
1 print("这是输出的语句")
```

print 中可以用单引号或双引号，但若想输出的内容中存在同样的引号，会和输出语句的引号产生歧义。

解决方法 1: print 语句和内容用不同的引号

解决方法 2: 在内容的引号前加转义字符\

```
1 print("I'm fine.")
2 print('Let\'s go!')
```

2.1.2 多行输出

使用三引号进行多行输出

```
1 print('''我相信能再次看到蓝天，
2 鲜花挂满枝头''')
```

```
这是输出的语句
I'm fine.
Let's go!
我相信能再次看到蓝天，
鲜花挂满枝头。
```

2.2 解一元二次方程

乘方为 `**` 根号可以用 `**(1/2)` 或使用 `math` 库

```
1 import math
2 a=int(input("请输入a的值"))
3 b=int(input("请输入b的值"))
4 c=int(input("请输入c的值"))
5 print((-b+math.sqrt(b**2-4*a*c))/(2*a))
6 print((-b-math.sqrt(b**2-4*a*c))/(2*a))
```

```
请输入a的值1
请输入b的值-3
请输入c的值2
2.0
1.0
```

2.3 for 循环与字典

2.3.1 普通循环

```
1 total=0
2 for i in range(1,11):
3     total=total+i
4 print(total)
```

注：1、python 的缩进非常严格，通过缩进判断循环是否结束。

2、range 的范围左闭右开

```
\系统开发工具基础\1\tool_class_2024_sum\week3\python\for1.py'
55
```

2.3.2 对字典的循环

```
1 a = {"小王":"123456789","小李":"15462847","小张":
      "4535415"}
```

```
2 for name, phone in a.items():
3     print(name+phone)
```

```
小王123456789
小李15462847
小张4535415
```

关于字典：

- 1、字典是键值对
- 2、a.keys() 返回所有键，a.values() 返回所有值，a.items() 返回所有键值对

2.4 while 循环

```
1 i=0
2 total=0
3 while i<11:
4     total=total+i
5     i=i+1
6 print(total)
```

2.5 格式化字符串

方法一：

```
1 name="小王"
2 print(f"你好{name}")
```

方法二：

```
1 name="小王"
2 print("你好{0}".format(name))
```

```
你好小王
你好小王
```

2.6 函数

```
1 def calculate(a):
2     s=a*a
3     return s
```



```
4 a=float(input("请输入正方形的边长"))
5 s=calculate(a)
6 print(f"边长为{a}的正方形的面积为: {s}")
```

```
请输入正方形的边长3
边长为3.0的正方形的面积为: 9.0
```

2.7 类

2.7.1 创建类以及类的实例化

```
1 class Student:
2     def __init__(self,name,number,grade): #构造函数
3         self.name = name
4         self.number = number
5         self.grade = grade
6     def showStudent(self):
7         print("姓名:" + self.name + " 学号:" + self.number +
8               " 年级:" + self.grade)
9
10 wang = Student("小王","1","9") #对象的初始化
11 wang.showStudent()
```

注：构造函数的下划线是两个下划线并非一个！

```
姓名: 小王 学号: 1 年级: 9
```

2.7.2 类的继承

```
1 class People:
2     def __init__(self,name,age): #构造函数
3         self.name = name
4         self.age = age
5
6 class Student(People):
7     def __init__(self,name,age,number,grade): #构造函数
8         super().__init__(name,age)
9         self.number = number
10        self.grade = grade
11    def showStudent(self):
```

```
12         print("姓名: "+self.name+" 年龄: "+self.age+" 学  
          号: "+self.number+" 年级: "+self.grade)  
13  
14 wang = Student("小王","14","1","9") #对象的初始化  
15 wang.showStudent()
```

解释:

1、`super()`会返回当前类的父类，可以用此函数调用父类的构造函数。

2、`class Student(People)`中括号的内容表示继承的父类。

姓名: 小王 年龄: 14 学号: 1 年级: 9

2.8 文件的读写

打开文件的代码:

```
1 open("./test.txt","r",encoding="utf-8")
```

1、其中第一个引号中表示文件的地址

2、第二个引号中表示要对文件进行什么操作

3、`encoding` 表示文件的编码方式

第二个引号中的内容	可对文件进行的操作
r	只读
w	只写（覆盖之前的内容）
a	只写（在已有内容后面添加）
r+	读和写（覆盖之前的内容）
a+	读和写（追加写）

2.8.1 文件的读

`.read()` 返回文件全部内容

`.readline()` 返回文件的一行内容

`.readlines()` 返回文件所有内容组成的列表

2.8.2 实例

```
1 f=open("test.txt","a",encoding="utf-8")  
2 f.write("——— 流浪地球")  
3 f.close()
```

```
4
5 f=open("test.txt","r",encoding="utf-8")
6 print(f.read())
7 f.close()
```

我相信能再次看到蓝天，
鲜花挂满枝头。—— 流浪地球

2.9 猜数字

```
1 import random
2 random_int = random.randint(1,10)
3 num = int(input("请输入一个1-10的数"))
4 while num != random_int:
5     if num > random_int:
6         print("猜大了")
7     else:
8         print("猜小了")
9     num = int(input("请输入一个1-10的数"))
10
11 print("恭喜你猜对了")
```

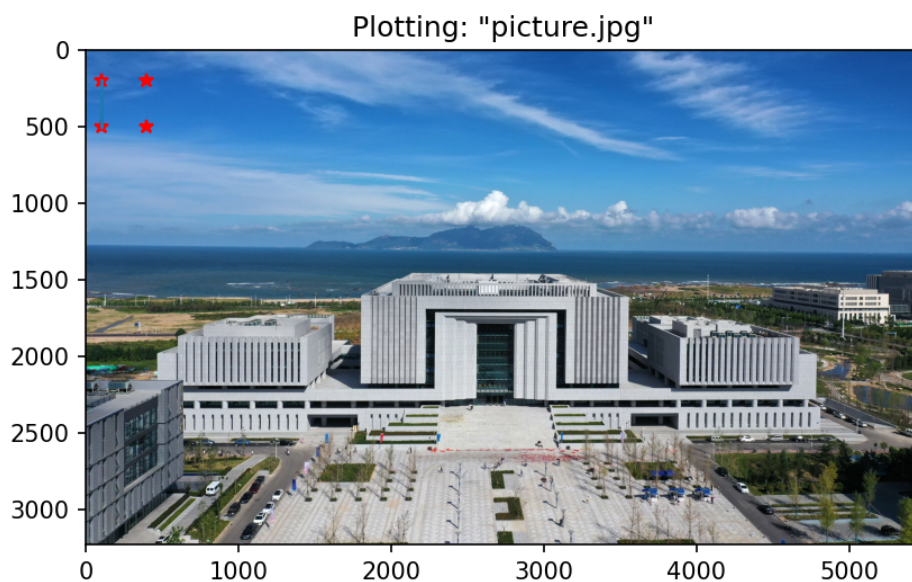
请输入一个1-10的数5
猜大了
请输入一个1-10的数2
猜小了
请输入一个1-10的数3
恭喜你猜对了

3 Python 视觉应用

3.1 绘制图像、点、线

```
1 from PIL import Image
2 from pylab import *
3 # 读取图像到数组中
4 im = array(Image.open('picture.jpg'))
5 # 绘制图像
6 imshow(im)
7 # 一些点
```

```
8 x = [100,100,400,400]
9 y = [200,500,200,500]
10 # 使用红色星状标记绘制点
11 plot(x,y,'r*')
12 # 绘制连接前两个点的线
13 plot(x[:2],y[:2])
14 # 添加标题，显示绘制的图像
15 title('Plotting: "picture.jpg"')
16 show()
```



3.2 创建缩略图

```
1 import matplotlib.pyplot as plt
2 from PIL import Image
3
4 pil_im = Image.open('picture.jpg')
5 pil_im.thumbnail((128,128))
6 plt.imshow(pil_im)
7 plt.axis('off')
8 plt.show()
```

原图片：



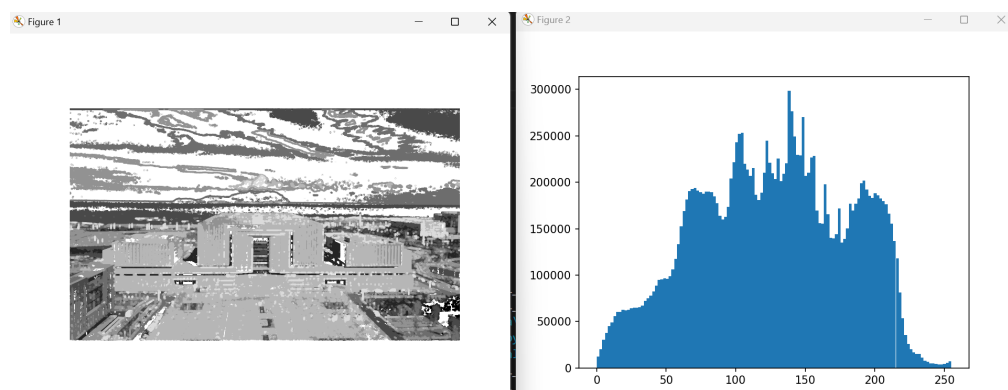
创建的缩略图：



3.3 生成图像轮廓和直方图

```
1 from PIL import Image
2 from pylab import *
3 # 读取图像到数组中
```

```
4 im = array(Image.open('picture.jpg').convert('L'))
5 # 新建一个图像
6 figure()
7 # 不使用颜色信息
8 gray()
9 # 在原点的左上角显示轮廓图像
10 contour(im, origin='image')
11 axis('equal')
12 axis('off')
13 figure()
14 hist(im.flatten(), 128)
15 show()
```



遇到的问题: vs code 中的 python 无法识别相对路径

解决方法: 在 launch.json 配置文件中添加 "cwd": "\${fileDirname}"

3.4 交互式批注

在绘图窗口的图像区域点击三次, 程序将这些点击的坐标 $[x, y]$ 自动保存在 x 列表里。

```
1 from PIL import Image
2 from pylab import *
3 im = array(Image.open('picture.jpg'))
4 imshow(im)
5 print('Please click 3 points')
6 x = ginput(3)
7 print('you clicked:', x)
8 show()
```



```
Please click 3 points
you clicked: [(np.float64(1485.1774193548388), np.float64(955.0064516129028)), (np.float64(2617.822580645161), np.float64(1889.070967741935)), (np.float64(4066.7258064516127), np.float64(1550.748387096774))]
```