系统开发工具基础实验报告

实验内容：实验三

姓名：张家宜　　学号：2024020013045

日期：2025 年 9 月 21 日

目录

1 练习内容

1.1 命令行环境

1.1.1 任务控制

输入 Ctrl-C 时，shell 会发送一个 SIGINT 信号到进程

输入 Ctrl-\可以发送 SIGQUIT 信号

以下是一些 SIG 信号的说明

编号	信号名称	作用说明
2	SIGINT	Ctrl+C 终止前台进程（可捕获）
9	SIGKILL	强制杀死进程（不可捕获/忽略）
15	SIGTERM	默认 kill 发送，要求优雅终止
1	SIGHUP	终端挂起，常用于让守护进程重载配置
3	SIGQUIT	Ctrl+ 终止并生成 core dump

1.1.2 暂停进程

键入 Ctrl-Z 会让 shell 发送 SIGTSTP 信号

可以使用 fg 或 bg 命令恢复暂停的工作

jobs 命令会列出当前终端会话中尚未完成的全部任务

1.1.3 后台执行进程

fg 和 bg 命令分别表示在前台继续和在后台继续

命令中的 & 后缀可以让命令在直接在后台运行

1.1.4 示例

```
$ sleep 1000
^Z
[1]+  Stopped                 sleep 1000
$ jobs
```

```
[1]+  Stopped                 sleep 1000
$ bg %1
[1]+ sleep 1000 &
$ jobs
[1]+  Running                 sleep 1000 &
$ kill -STOP %1
[1]+  Stopped                 sleep 1000
$ jobs
[1]+  Stopped                 sleep 1000
$ kill -SIGHUP %1
[1]+  Stopped                 sleep 1000
$ jobs
[1]+  Hangup                  sleep 1000
```

1.1.5 终端多路复用

	操作说明	**命令**
会话	启动 tmux	`tmux`
	指定名称启动	`tmux new -s` *NAME*
	列出所有会话	`tmux ls`
	分离当前会话	`<C-b> d`
	重新连接最后会话	`tmux a`
	指定会话连接	`tmux a -t` *NAME*
窗口	新建窗口	`<C-b> c`
	关闭窗口	`<C-d>`
	跳转到第 N 个窗口	`<C-b> N`
	切换到前一个窗口	`<C-b> p`
	切换到下一个窗口	`<C-b> n`
	重命名当前窗口	`<C-b> ,`
	列出所有窗口	`<C-b> w`
面板	水平分割	`<C-b> "`
	垂直分割	`<C-b> %`
	切换方向	`<C-b> <` *方向键* `>`
	缩放/还原当前面板	`<C-b> z`

	回卷查看输出（空格选择，回车复制）	<C-b> [
	切换面板排布	<C-b> < 空格 >

1.1.6 别名

bash 中的别名语法如下：

```
alias alias_name="command_to_alias arg1 arg2"
```

一些示例：

```
alias ll="ls -lh"

alias gs="git status"
alias gc="git commit"
alias v="vim"

alias sl=ls

alias la="ls -A"
alias lla="la -l"

# 在忽略某个别名
\ls
# 或者禁用别名
unalias la

# 获取别名的定义
alias ll
# 会打印 ll='ls -lh'
```

1.1.7 配置文件（Dotfiles）

对于 bash 来说，在大多数系统下，可以通过编辑.bashrc 或.bash_profile 来进行配置。

1.1.8 远端设备

通过如下命令，可以使用 ssh 连接到其他服务器：

```
ssh foo@bar.mit.edu
```

使用 ssh-keygen 命令可以生成一对密钥：

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

使用 /.ssh/config 可以对 ssh 进行配置

```
Host vm
    User foobar
    HostName 172.16.174.141
    Port 2222
    IdentityFile ~/.ssh/id_ed25519
    LocalForward 9999 localhost:8888

# 在配置文件中也可以使用通配符
Host *.mit.edu
    User foobaz
```

1.1.9 通过 SSH 复制文件

1. **ssh + tee** 通过管道传输标准输入，在远程使用 tee 保存文件。示例：

```
cat localfile | ssh user@host "tee serverfile"
```

2. **scp** 直接复制文件或目录，支持递归与权限保持。示例：

```
scp -r local_dir user@host:/path/
```

3. **rsync** 基于差异同步，可断点续传，支持符号链接和权限同步。示例：

```
rsync -avz --partial localfile user@host:/path/
```

1.2 Python 入门基础

1.2.1 数据类型

Python 内置了丰富的数据类型，可分为不可变类型和可变类。

常见分类如下：

数值类型（不可变）

- **int**：整数类型。例如：

```
a = 42
b = -7
```

- **float**：浮点数。示例：

```
pi = 3.14159
```

- **complex**：复数，形式为 `a + bj`。

```
z = 2 + 3j
```

布尔类型（不可变）

- **bool**：值为 `True` 或 `False`。`True` 等价于 1，`False` 等价于 0。

```
flag = True
if flag:
    print("It's True")
```

字符串（不可变）

- **str**：文本数据，可使用单引号或双引号定义。
- 支持切片和索引：`s[0]`, `s[1:4]`。
- 常用方法：`upper()`、`lower()`、`replace()`、`split()`、`join()`。

```
s = "Hello World"
print(s.upper())
print(s[0:5])
```

序列类型

- **list**（可变）：有序集合，可存放不同类型元素。

```
nums = [1, 2, 3]
nums.append(4)
```

- **tuple**（不可变）：有序且不可修改。

```
coords = (10, 20)
```

- **range**（不可变）：生成整数序列，常与 `for` 循环搭配。

```
for i in range(1, 6):
    print(i)
```

集合类型

- **set**：无序且元素唯一，可进行集合运算（交集、并集、差集）。

```
s1 = {1, 2, 3}
s2 = {3, 4}
print(s1 & s2)  # 交集
```

映射类型（可变）

- **dict**：键值对存储，键必须可哈希且唯一。

```
person = {"name": "Alice", "age": 20}
person["age"] = 21  # 修改值
```

- 常用方法：`keys()`、`values()`、`items()`、`get()`。

1.2.2 顺序结构

Python 程序默认自上而下执行:

```
print("第一行")
print("第二行")
```

1.2.3 判断结构

```
x = 10
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

1.2.4 循环结构

```
# for循环
for i in range(5):
    print(i)

# while循环
count = 0
while count < 5:
    print(count)
    count += 1
```

1.2.5 函数

```
def add(a, b):
    return a + b

result = add(3, 4)
print(result)
```

参数类型包括位置参数、默认参数、关键字参数、可变参数 (*args, **kwargs)；支持匿名函数：

```
double = lambda x: x * 2
print(double(5))
```

1.2.6 运算符与表达式

- **算术运算符**：+ 加、- 减、* 乘、/ 真除、// 整除、% 取余、** 幂运算。

```
a, b = 7, 3
print(a + b, a // b, a ** b)
```

- **比较运算符**：==、!=、>、<、>=、<=。
- **逻辑运算符**：and、or、not。

```
x = 5
print(x > 0 and x < 10)
```

- **赋值运算符**：=、+=、-=、*= 等。
- **成员运算符**：in、not in。

```
print(3 in [1,2,3])
```

- **身份运算符**：is、is not 用于判断对象标识是否相同。

1.2.7 输入

- **标准输入**：input() 读取字符串，可配合 int() 等函数转换类型。

```
name = input("请输入姓名：")
age = int(input("请输入年龄："))
```

1.2.8 输出

- **标准输出**：print() 可指定分隔符和结尾字符。

```
print("Hello", "World", sep=", ", end="!\n")
```

- **字符串格式化**:

```
print(f"{name} 今年 {age} 岁")
print("Name: {}, Age: {}".format(name, age))
```

1.2.9 模块与包

- **模块**: 每个 .py 文件都是一个模块，可使用 import 导入。

```
import math
print(math.sqrt(16))
```

- **选择性导入**:

```
from math import pi, sin
print(pi, sin(pi/2))
```

- **别名导入**:

```
import numpy as np
```

- **自定义模块**: 将函数或类写入独立 .py 文件，通过 import 文件名调用。
- **包**: 含有 __init__.py 文件的文件夹即为包，可包含多个模块形成层次结构。

```
from mypackage import mymodule
```

1.3 Python 视觉应用

1.3.1 图像对比度增强

使用 PIL.ImageEnhance 对图像进行对比度增强。

```
from PIL import Image, ImageEnhance

img_original = Image.open("a.jpg")
img_original.show("Original Image")

# 增强对比度，系数 3.8
img = ImageEnhance.Contrast(img_original)
```

```
img.enhance(3.8).show("Image With More Contrast")

input()
```

说明：读取原始图片并显示，然后通过设置系数 3.8 将对比度显著提高。

1.3.2 颜色通道分离

利用 NumPy 和 PIL 分离并显示 RGB 三个通道。

```
from PIL import Image
import numpy as np

img = np.array(Image.open('a.jpg'))

# 仅保留红色通道
img_red = img.copy()
img_red[:, :, (1, 2)] = 0

# 仅保留绿色通道
img_green = img.copy()
img_green[:, :, (0, 2)] = 0

# 仅保留蓝色通道
img_blue = img.copy()
img_blue[:, :, (0, 1)] = 0

# 拼接原图与三种单通道图像
img_ORGB = np.concatenate((img, img_red, img_green, img_blue), axis
    =1)
img_converted = Image.fromarray(img_ORGB)
img_converted.show()

input()
```

说明：分别提取 R、G、B 三个单独通道，并将它们与原图水平拼接显示。

1.3.3 图像裁剪与缩放

使用 OpenCV 进行裁剪和缩放。

```
import cv2

img = cv2.imread("a.jpg")

# 按坐标裁剪图像
imgCropped = img[150:383, 125:290]
shape = imgCropped.shape
print(shape[0])

# 调整尺寸：高度增加 20%，宽度翻倍
imgCropped = cv2.resize(imgCropped, (shape[0]*12//10, shape[1]*2))

# 显示原图与裁剪后的图
cv2.imshow("Image cropped", imgCropped)
cv2.imshow("Image", img)
cv2.waitKey(0)
```

说明：从原图中截取指定区域并进行比例缩放，然后在窗口中显示处理结果。

2 解题感悟

命令行环境里面的方法让我在 shell 里面也可以快捷的控制任务，简化操作，是对上节课的拓展，很实用。

Python 是当下流行的编程语言，它的语法简单，功能强大，小到日常办公，大到科学研究都可以用到。标准库“开箱即用”，模块与包的组织清晰，配合 `NumPy` 等科学计算库，用少量且可读性强的代码就能把数据处理、文件批量操作、可视化等任务快速落地，既提高效率也便于后续复用与维护。

同时，Python 图像处理也让我们可以快速的对图片进行细微的编辑或者大批量编辑，这次用 `PIL.ImageEnhance` 做对比度增强，用 `NumPy` 分离与拼接颜色通道，再用 `OpenCV` 完成裁剪与尺寸调整，把这些步骤写成小脚本后即可一键复现同样的流程，对单张图精细微调与成批处理目录都很方便。

参考文献

[1] Missing Semester 中文版，https://missing-semester-cn.github.io/

[2] Python 基础教程，https://www.runoob.com/python/python-tutorial.html

[3] Python 视觉应用，https://blog.csdn.net/sgzqc/article/details/124871774

GitHub 链接

https://github.com/Misasasasasaka/report/tree/main/P3