课程作业报告

课程名称：系统工具开发基础

学生姓名：张家玮

学号：23020007157

日期：2024 年 9 月 11 日

目录

Chapter 1

命令行环境

1.1 基础知识

命令行环境是一个强大的工具，用于与操作系统进行交互和管理各种任务。下面有一些比较基础实用的操作：

1.1.1 文件和目录操作

- pwd: 显示当前工作目录。
- cd: 更改当前工作目录。
- ls: 列出当前目录的文件和文件夹。
- cp：复制文件或目录。
- mv：移动或重命名文件或目录。
- mkdir：创建新目录。
- rm：删除文件。
- cat：显示文件内容。

1.1.2 文件查找

- find path -name pattern：在指定路径下查找匹配模式的文件。
- locate filename：查找文件（使用预构建的数据库）。
- grep pattern file：搜索文件中包含特定模式的行。

1.1.3 重定向和管道

- command >file：将命令输出重定向到文件，覆盖原文件内容。
- command >>file：将命令输出追加到文件末尾。
- command <file：从文件读取输入。
- command1 — command2：将 command1 的输出作为 command2 的输入。

1.1.4 系统信息

- ps aux：显示所有进程的信息。
- kill：终止指定进程。
- pkill：通过名称终止进程。

1.1.5 脚本和自动化

- bash ：执行 Bash 脚本。
- chmod +x：使脚本可执行。
- aliasname='command'：创建命令别名。
- unalias：删除命令别名。

1.2 任务控制

任务控制中分为前台进程和后台进程，即：

- 前台进程：当你在命令行运行一个命令时，默认情况下它会在前台运行，直到任务完成或被中断。
- 后台进程：你可以让命令在后台运行，这样你可以继续在终端中执行其他命令。

1.2.1 将任务放到后台

这里给出一个实例，是我在最近开发mud游戏时使用的日志监管文件。

这样就可以成功让这个脚本在后台运行。

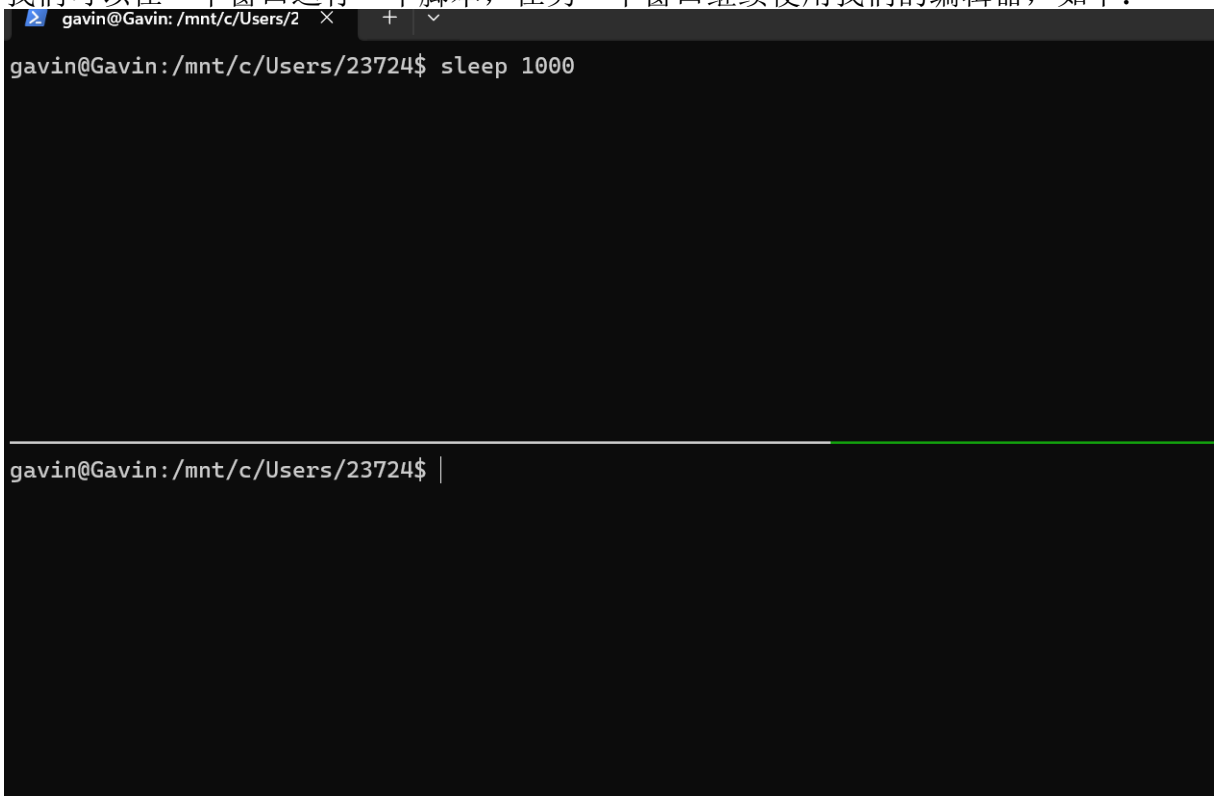
1.2.2 终止特定后台程序

我们可以同时运行多个后台程序，然后通过kill命令来终止我们想要终止的特定程序。如下图我们同时创建三个后台运行的程序然后再关闭我们想要关闭的脚本。

1.3 终端多路复用

1.3.1 多个终端

为了能够同时达到同时执行多个任务，运行编辑器，且在另一端执行程序，我们就可以使用终端多路复用。我们这里使用tmux这个终端多路复用器。
我们可以在一个窗口运行一个脚本，在另一个窗口继续使用我们的编辑器，如下：



1.3.2 在远程服务器上保持连接

当我们在远程服务器上运行了一个长时间运行的脚本，你希望保持会话，即使断开连接也能重新连接并查看脚本进展。就可以按照这样操作；

1. 启动 tmux 会话
2. 运行长时间运行的脚本
3. 按 Ctrl+b，然后按 d 将会话分离，你可以安全地断开与服务器的连接。
4. 重新连接会话,使用tmux attach-session命令来重新连接会话，从而达到目的。

1.4 别名

为了便于我们进行操作大多数Shell都有别名的操作，便于我们执行一些比较长的指令。在bush中别名的语法规则如下：

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

这里有我们定义好的一个别名：

```
[alias]
        graph = log --all --graph --decorate --oneline
```

这样我们就可以使用graph来直接执行我们设置好的长指令。

1.5 配置文件

配置文件是用来存储软件、工具或系统的配置参数的文件。通过配置文件我们可以自定义一些个性化的指令，配置环境等等，这种方式具有可移植性。
例如我们可以通过配置git的.gitconfig文件，来自定义一些功能，如加入颜色的修改。

```
[color]
        ui=auto
```

1.6 远端设备

这里我们操作了一下如何通过ssh协议，来远程利用ssh协议实现对我们GitHub仓库的各种操作。我们可以通过使用 ssh-keygen 命令可以生成一对密钥。可以看到我们生成的公钥和密钥的文件。

```
23724@Gavin MINGW64 ~
$ cd ~/.ssh

23724@Gavin MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

然后再打开公钥的文件，复制公钥进入Github中在设置里面添加对应的SSH密钥。

Chapter 2

Python基础

2.1 基础语法

在这里用程序展示一些关于python中一些简单语法，包括变量与数据类型，运算符，字符串操作，用户输入等等。

```
# 变量是存储数据的容器，如下我们定义个人信息
name = "Gavin" # 字符串类型
age = 20       # 整数类型
height = 1.83  # 浮点数类型
is_oucer = True# 布尔类型

print("Name:",name)
print("Age:",age)
print("Height:",height)
print("Is a student of OUC",is_oucer)

# python中运算符可以用于处理数学运算，字符串连接。
first="I am a"
second="student"
full=first+ " "+ second

print(full)

# 从用户那里获得输入，可以使用input()函数
user_name = input("what is your name?")

output = "Nice to meet you," + user_name +"!"
print(output)
```

2.2 控制流

在 Python 中，控制流的主要结构包括条件判断、循环和循环控制语句。

```python
# 条件判断if-elif-else，判断时可以使用and或or来进行
age = 20
is_student = False

if age >= 18 and is_student:
    print("You are an adult student.")
elif age >= 18 and not is_student:
    print("You are an adult, but not a student.")
else:
    print("You are not an adult.")

# 循环(for while)
codes = ["Java","C++","Python"]
for code in codes:
    print(code)

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

# break、continue 和 pass
numbers = [3, 0, 5, 8, 7, 2, 10, 0, 9]

for number in numbers:
    if number == 0:
        pass  # 0 是特殊情况，这里什么也不做，继续处理下一个数字
    elif number % 2 != 0:
        continue  # 奇数情况下跳过处理
    elif number == 7:
        print("Found 7, stopping the loop.")
        break  # 遇到 7 时停止循环
    print(f"Processing number: {number}")

print("Loop ended.")
```

2.3 函数

函数能够帮助我们完成很多的功能，实现代码重用，增加可读性，使代码参数化，模块化。

这里展示一个斐波那契数列的函数：

```python
def fibonacci(n):
    sequence = [0, 1]
    for i in range(2, n):
        sequence.append(sequence[-1] + sequence[-2])
    return sequence[:n]

print(fibonacci(10))
```

2.4 数据结构

Python 中的常见数据结构包括列表、元组、字典、集合等。

1. 列表:列表是一种有序的、可变的序列，支持重复元素。
2. 元组:元组是有序的不可变序列，通常用于存储不需要修改的数据。
3. 字典:字典是键值对的集合，键是唯一的，值可以是任意数据类型。字典用于快速查找、插入和删除。
4. 集合:集合是无序的、不重复的元素集合，常用于去重和集合运算。集合是无序的、不重复的元素集合，常用于去重和集合运算。

具体练习代码可以查看GitHub仓库。

2.5 文件操作

在 Python 中，使用 open() 函数打开文件，并使用 close() 函数关闭文件。通常使用 with 语句自动管理文件的打开和关闭，这样即使出现异常，也能确保文件被正确关闭。

使用 open() 函数时，可以指定文件模式，常用模式包括：

- 'r'：只读模式（默认）
- 'w'：写入模式（会清空文件内容，如果文件不存在则创建）
- 'a'：追加模式（在文件末尾添加内容）
- 'b'：二进制模式（与 'r'、'w'、'a' 结合使用）
- 'x'：独占创建模式（文件存在则报错）

下面有两个实例：

2.5.1 逐行读取文件并打印

```python
# 逐行读取
with open('example.txt', 'r') as file:
    for line in file:
        print(line.strip())
# 一次性读取
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

2.5.2 写入文件

```python
# 将一些文本写入到 'example.txt' 文件中
# 输出内容将覆盖已有文件内容
with open('example.txt', 'w') as file:
    file.write("Hello, World!\n")
    file.write("This is a new file.\n")

# 将一些文本追加到 'example.txt' 文件末尾
with open('example.txt', 'a') as file:
    file.write("Appending a new line.\n")
```

Chapter 3

Python视觉应用

Python在视觉应用领域有广泛应用，主要是使用一些Python中的视觉处理库来进行这些处理。
下面有一些练习的实例：

3.1 图像的灰度化与显示

```python
import cv2          # OpenCV库，用于图像处理
import matplotlib.pyplot  # Matplotlib库，用于图像显示

# 1. 读取图像
image = cv2.imread('ouc.png')
# 2. 转换为灰度图像
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 3. 显示原图像和灰度图像
matplotlib.pyplot.subplot(1, 2, 1)  # 创建一个 1x2 的子图布局，当前选择第 1 个子图
matplotlib.pyplot.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # 转换并显示原图像
matplotlib.pyplot.title('Original Image')  # 设置子图的标题

matplotlib.pyplot.subplot(1, 2, 2)  # 选择第 2 个子图
matplotlib.pyplot.imshow(gray_image, cmap='gray')  # 直接显示灰度图像（不需要转换）
matplotlib.pyplot.title('Gray Image')  # 设置子图的标题

matplotlib.pyplot.show()  # 显示图像窗口
```

3.2 图像的模糊化

```python
import cv2
import matplotlib.pyplot

# 读取图像
image = cv2.imread('ouc.png')

# 将图像转换为 RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# 应用高斯模糊
blurred_image = cv2.GaussianBlur(image_rgb, (15, 15), 0)

# 显示模糊处理后的图像
matplotlib.pyplot.imshow(blurred_image)
matplotlib.pyplot.axis('off')
matplotlib.pyplot.show()
```

Chapter 4

学习心得

4.1 感受

在学习命令行环境的过程中，首先还是因为这是一个没有图形用户界面（GUI）的系统，所以我们不能得到反馈所有在开始学习时都会很抽象，我们需要记住各种指令，但是这些指令在掌握了后会变得十分高效，命令行给我们提供了强大的控制力，让我们来管理系统和各种程序的行为。我们还能使用一个具有魔力的东西，脚本来让我们进行重复性操作。至于Python的基础操作，在以及学习了一端程序之后，这个上手起来就十分便捷了，从B站以及菜鸟教程中了解到相关的语法规则再进行相关的练习就好了。python的视觉应用，现在主要是根据一些教程，实际操作了一些程序，学习的并没有更加深入，后续应该结合一些实例来练习一下。

4.2 学习中遇到的困难

困难主要是在学习命令行环境中遇到，主要还是感觉不够熟练需要长期练习，不断使用。别的在远端设备中操作遇到问题，通过谷歌搜索，查询到类似问题的帖子，参考别人的解决方案成功解决问题。

Chapter 5

参考资料

- Missing Semester CN
- Google
- Wikipedia
- B站
- 菜鸟教程
- 计算机视觉编程

如果想要查看相关实例练习的源代码可以查询我的GitHub仓库
GitHub Repository