中國海洋大學

命令行环境与 Python 实验报告

姓名：乔宇恒

学号：23030021059

2024 年 9 月 11 日

目录

1 命令行环境

1.1 使用 pgrep 来查找 pid

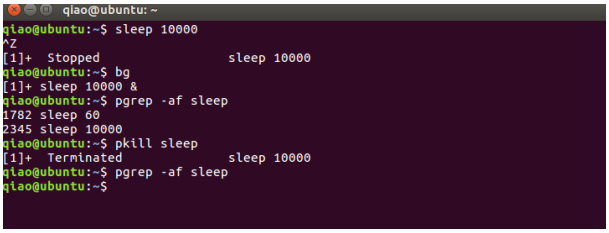

图 1: 使用 pgrep 来查找 pid

1.2 使用 pkill 结束进程而不需要手动输入 pid

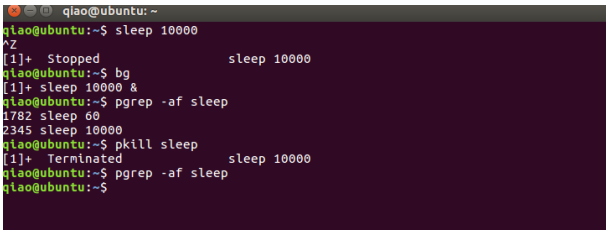

图 2: 使用 pkill 结束进程而不需要手动输入 pid

1.3 wait 命令

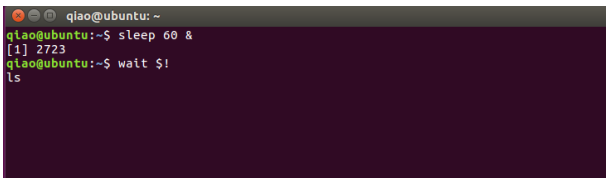

图 3: wait 命令

1.4 编写一个 bash 函数 pidwait

```
pidwait() {
    local pid="$1"

    # 检查 PID 是否为空
    if [ -z "$pid" ]; then
        echo "Usage: pidwait <pid>"
```

```
        return 1
    fi

    # 一直循环，直到进程结束
    while kill -0 "$pid" 2>/dev/null; do
        sleep 1  # 每隔 1 秒检查一次进程状态
    done

    echo "Process $pid has ended."
}
```

1.5 创建别名

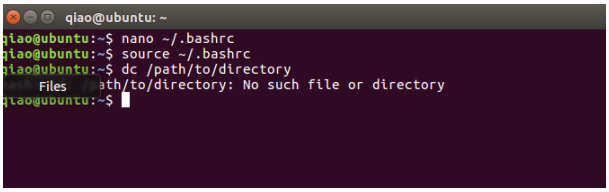

图 4: 创建别名

1.6 别名语法

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

1.7 文件操作

列出目录内容: ls
切换目录: cd /path/to/directory
查看当前目录: pwd
创建目录: mkdir directory_name
删除目录: rmdir directory_name 或 rm -r directory_name (删除非空目录)
删除文件: rm file_name
复制文件: cp source_file destination
移动/重命名文件: mv source_file destination

1.8 文件内容查看

查看文件内容: cat file_name
分页查看文件内容: less file_name 或 more file_name
显示文件头部: head file_name
显示文件尾部: tail file_name

1.9 系统信息

查看磁盘使用情况: df -h
查看内存使用情况: free -h
查看 CPU 信息: lscpu
查看系统版本: uname -a

1.10 进程管理

查看进程: ps aux 或 top
杀死进程: kill process_id
强制杀死进程: kill -9 process_id
查找进程: pgrep process_name

1.11 Windows 系统中的文件和目录操作

列出目录内容: dir
切换目录: cd path\to\directory
查看当前目录: cd 或 echo %cd%
创建目录: mkdir directory_name
删除目录: rmdir directory_name 或 rd directory_name
(删除非空目录使用 /s 选项)
删除文件: del file_name
复制文件: copy source_file destination
移动/重命名文件: move source_file destination

1.12 Windows 系统中的进程管理

查看进程: tasklist
杀死进程: taskkill /PID process_id
强制杀死进程: taskkill /F /PID process_id
查找进程: tasklist | findstr process_name

2 Python

2.1 Python 简介

Python 是一种广泛使用的高级编程语言，以其简洁、易读的语法和强大的功能著称。它支持多种编程范式，包括面向对象、过程式和函数式编程，并且提供了丰富的标准库和第三方模块，使得开发各种应用程序变得高效且灵活。作为一种跨平台的解释型语言，Python 被广泛应用于数据分析、机器学习、Web 开发、自动化脚本等多个领域。

2.2 Python 基础操作

2.2.1 正则表达式

```
import re

result = re.match(r'\d+', '123abc')
if result:
    print(result.group())  # 输出: 123
```

从字符串的起始位置匹配正则表达式模式，如果匹配成功，返回一个匹配对象；否则返回 None。

2.2.2 变量赋值

```
x = 10
y = 20
z = x + y
print(z)  # 输出: 30
```

2.2.3 条件语句

```
x = 5
if x > 0:
    print("x 是正数")
elif x == 0:
    print("x 是零")
else:
    print("x 是负数")
```

2.2.4 循环

```
for i in range(5):
    print(i)  # 输出: 0 1 2 3 4
```

2.2.5 列表操作

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])  # 输出: 1
numbers.append(6)
print(numbers)  # 输出: [1, 2, 3, 4, 5, 6]
```

2.2.6 字典操作

```
person = {'name': 'Alice', 'age': 25}
print(person['name'])  # 输出: Alice
person['age'] = 26
print(person)  # 输出: {'name': 'Alice', 'age': 26}
```

2.2.7 函数定义

```
def add(x, y):
    return x + y

result = add(5, 3)
print(result)  # 输出: 8
```

2.2.8 列表推导式

```
numbers = [1, 2, 3, 4, 5]
squares = [n**2 for n in numbers]
print(squares)  # 输出: [1, 4, 9, 16, 25]
```

2.2.9 文件读取

```
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

2.2.10 异常处理

```
try:
    x = int(input("输入一个数字: "))
    print(x)
except ValueError:
    print("输入无效, 请输入数字")
```

2.2.11 类与对象

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.name}")

p = Person('Alice', 25)
p.greet()  # 输出: Hello, my name is Alice
```

2.2.12 列表切片

```
numbers = [1, 2, 3, 4, 5]
print(numbers[1:4])  # 输出: [2, 3, 4]
```

2.2.13 集合操作

```
numbers = {1, 2, 3, 4}
numbers.add(5)
print(numbers)  # 输出: {1, 2, 3, 4, 5}
```

2.2.14 迭代器

```
my_list = [1, 2, 3]
iterator = iter(my_list)
print(next(iterator))  # 输出: 1
print(next(iterator))  # 输出: 2
```

2.2.15 列表排序

```
numbers = [5, 3, 1, 4, 2]
sorted_numbers = sorted(numbers)
print(sorted_numbers)  # 输出: [1, 2, 3, 4, 5]
```

2.3 Python 视觉应用

```
from PIL import Image, ImageFilter

# 打开图像
image_path = "input_image.jpg"
image = Image.open(image_path)

# 1. 显示原始图像
image.show()

# 2. 灰度化处理
gray_image = image.convert("L")
gray_image.show()

# 3. 图像缩放
resized_image = image.resize((image.width // 2, image.height // 2))
# 将图像缩小一半
resized_image.show()

# 4. 边缘检测
edges_image = image.filter(ImageFilter.FIND_EDGES)
edges_image.show()

# 5. 保存处理后的图像
gray_image.save("output_gray_image.jpg")
resized_image.save("output_resized_image.jpg")
edges_image.save("output_edges_image.jpg")

print("图像处理完成, 处理后的图像已保存。")
```

3 实验感悟

学习了命令行环境的基本操作，了解了 Python 语言入门基础知识和在视觉方面的应用

4 Github 网址

https://github.com/Eternity-JOE/Git