

系统开发工具基础第三周报告

韩昊鑫 24020007039

September 2025

Contents

0.1	命令行	1
0.1.1	简单指令	1
0.1.2	tmux 会话	1
0.1.3	窗口	1
0.1.4	别名	1
0.1.5	别名的使用	2
0.1.6	配置文件	2
0.1.7	远程登录	2
0.1.8	课后实验	3
0.2	Python 学习	3
0.2.1	变量和输出	3
0.2.2	运算符	3
0.2.3	条件语句	4
0.2.4	循环语句	4
0.2.5	循环嵌套	4
0.2.6	循环内部操作	4
0.2.7	函数	5
0.2.8	汇总	5
0.3	Python 视觉	6
0.3.1	01	6
0.3.2	02	6
0.3.3	03	6
0.3.4	04	6
0.3.5	05	6
0.4	github 链接	6

0.1 命令行

0.1.1 简单指令

```
sleep 1000  
nohup sleep 2000 &  
jobs  
kill -SIGHUP %1
```

0.1.2 tmux 会话

1. tmux 开始一个新的会话
2. tmux new -s NAME 以指定名称开始一个新的会话
3. tmux ls 列出当前所有会话
4. 在 tmux 中输入 <C-b> d , 将当前会话分离
5. tmux a 重新连接最后一个会话。您也可以通过 -t 来指定具体的会话

0.1.3 窗口

1. <C-b> c 创建一个新的窗口, 使用 <C-d> 关闭
2. <C-b> N 跳转到第 N 个窗口, 注意每个窗口都是有编号的
3. <C-b> p 切换到前一个窗口
4. <C-b> n 切换到下一个窗口
5. <C-b> , 重命名当前窗口

<C-b> 是指 Ctrl + b

0.1.4 别名

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

= 两边是没有空格的, 因为 alias 是一个 shell 命令, 它只接受一个参数。
这相当于 C++ 中的 define/using 语句

0.1.5 别名的使用

```
# 创建常用命令的缩写
alias ll="ls -lh"
alias gs="git status"
alias gc="git commit"
alias v="vim"
# 别名可以组合使用
alias la="ls -A"
alias lla="la -l"
# 禁用别名
unalias la
# 获取别名的定义
alias ll, 会打印 ll='ls -lh'
```

0.1.6 配置文件

在默认情况下 shell 并不会保存别名。为了让别名持续生效，您需要将配置放进 shell 的启动文件里，像是 .bashrc 或 .zshrc。

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

还有一些其他的工具也可以通过点文件进行配置：

1. bash - ~/.bashrc, ~/.bash_profile
2. git - ~/.gitconfig
3. vim - ~/.vimrc 和 ~/.vim 目录
4. ssh - ~/.ssh/config
5. tmux - ~/.tmux.conf

0.1.7 远程登录

目前使用的工具是 Xshell8，虚拟机连接到 Xshell8 即可，虚拟机终端输入 ifconfig 找到 ip 地址，并输入到 Xshell8 中进行连接测试即可

```
wt@wt-Linux:~$  
wt@wt-Linux:~$ sleep 100000  
^Z  
[1]+  已停止                  sleep 100000  
wt@wt-Linux:~$ bg  
[1]+  sleep 100000 &  
wt@wt-Linux:~$ jobs  
[1]+  运行中                  sleep 100000 &  
wt@wt-Linux:~$ pgrep sleep  
4668  
wt@wt-Linux:~$ pkill -f sleep  
[1]+  已终止                  sleep 100000  
wt@wt-Linux:~$ jobs  
wt@wt-Linux:~$
```

Figure 1: test after class

0.1.8 课后实验

0.2 Python 学习

0.2.1 变量和输出

Python 无需定义变量类型，直接 `a = 12` 即可，简单输出直接 `print()` 即可

0.2.2 运算符

大部分同 C++ 一样 `+*/`

`%` 取模返回除法的余数

`*` 幂返回 `x` 的 `y` 次幂

`//` 取整除返回商的整数部分（向下取整）

还有比较运算符，赋值运算符，位运算符，逻辑运算符等

成员运算符 `in` 和 `not in`

身份运算符 `is` 和 `is not`

```

wt@wt-Linux:~$ sleep 600 &
[1] 18368
wt@wt-Linux:~$ pgrep sleep | wait; ls
公共 视频 文档 音乐 brightness nihao2 nohup.out
模板 图片 下载 桌面 nihao nihao666 snap
wt@wt-Linux:~$ pidwait()
> {
> while kill -0 $1
> do
> sleep 1
> done
> ls
> }
wt@wt-Linux:~$ sleep 600 & pidwait $(pgrep sleep 600)
[2] 18376
pgrep: only one pattern can be provided
Try 'pgrep --help' for more information.
kill: 用法: kill [-s 信号说明符 | -n 信号编号 | -信号说明符] pid
.. 或 kill -l [信号说明符]
公共 视频 文档 音乐 brightness nihao2 nohup.out
模板 图片 下载 桌面 nihao nihao666 snap

```

Figure 2: test after class

```

wt@wt-Linux:~$ alias dccc
wt@wt-Linux:~$ history | awk '{ $1=""; print substr($0,2) }' | sort | uniq -c | sor
t -n | tail -n 10
  2 sudo apt update
  2 sudo mkdir -p /mnt/hgfs
  2 sudo reboot
  2 vmhgfs-fuse .host:/Shared /mnt/hgfs
  4 sleep 100000
  5 pkill -f sleep
  6 bg
  6 ls /mnt/hgfs
  6 pgrep sleep
 17 jobs
wt@wt-Linux:~$

```

Figure 3: test after class

0.2.3 条件语句

同 C++ 一样，只不过没有括号

0.2.4 循环语句

同上

0.2.5 循环嵌套

依旧同上

0.2.6 循环内部操作

break continue 同 C++ 一样，py 多了一个 pass 语句，表示什么也不做，相当于 C++ 中的

0.2.7 函数

换了个样子的 C++ 函数，结构简单

0.2.8 汇总

```
def fun(a): 1个用法
    for i in range(a):
        a += 4
    return a

print('\n')
b = "hello world"
print(b[2:5])
print(b.upper())
print(b.replace(__old: 'hello', __new: 'python'))
print(b)
x = 'he' in b
print(x)
"""这是注释"""
#这也是注释
'''这还是注释'''
aa = "ni"
bb = "hao"
cc = "hello"
dd = aa + bb + "," + cc
print(dd)
a = 5
a = fun(a)
print(a)
print('')
while(a > 0):
    a -= 5
    print(a)

a += 2
print('')
for i in range(a):
    for j in range(3):
        if i+j > 2:
            print(i + j)
```

Figure 4: Python

```
C:\Users\10376\PycharmProjects\001\.venv\Scripts\

llo
HELLO WORLD
python world
hello world
True
nihao,hello
25

20
15
10
5
0

3

进程已结束，退出代码为 0
```

Figure 5: Python

0.3 Python 视觉

0.3.1 01

0.3.2 02

0.3.3 03

0.3.4 04

0.3.5 05

0.4 github 链接

下面是 github 链接:

<https://github.com/C-learning-beginner/git-test.git>

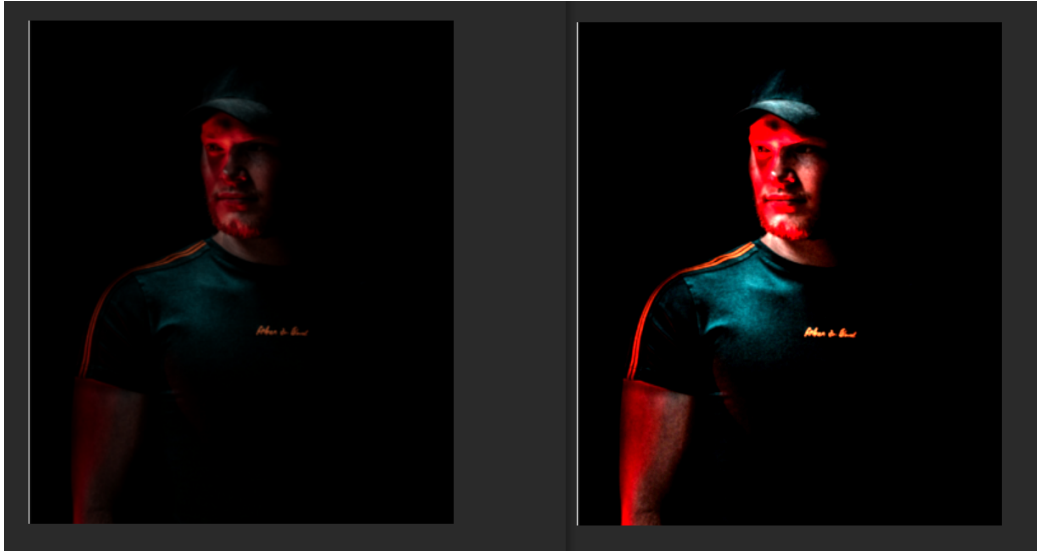


Figure 6: Python 视觉 01

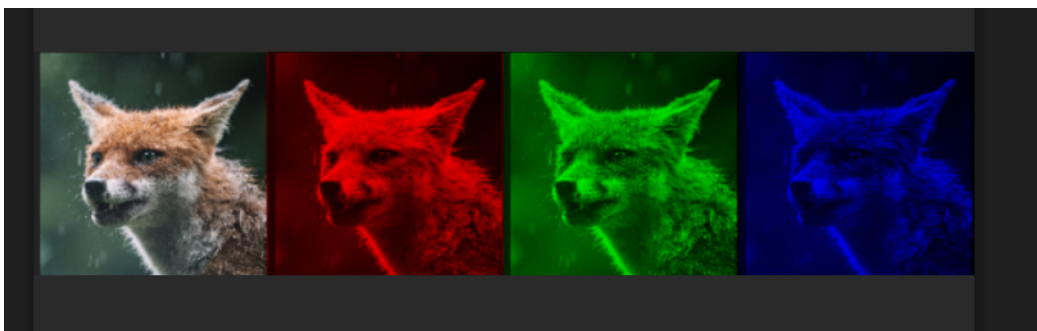


Figure 7: Python 视觉 02

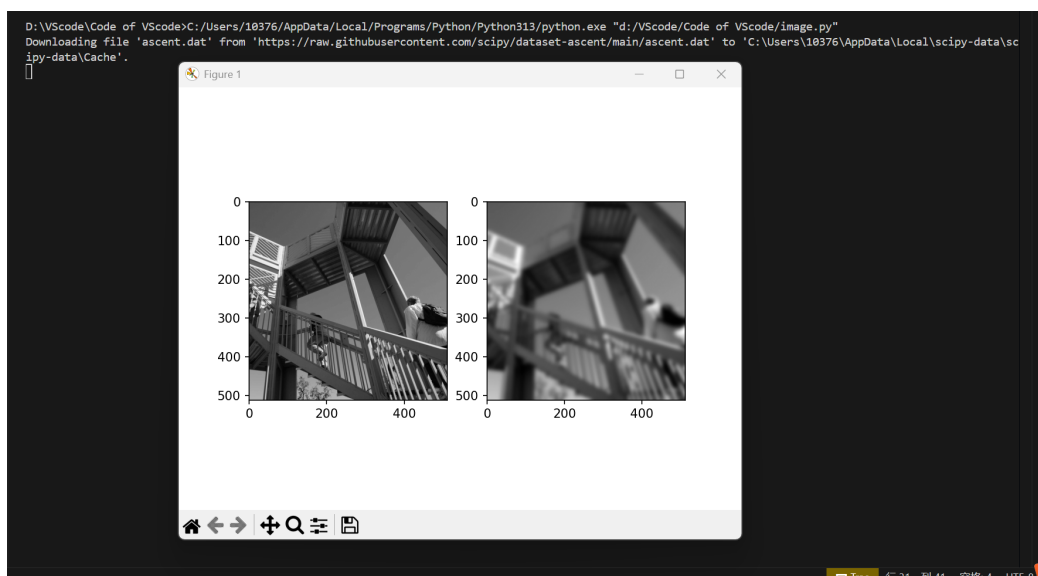


Figure 8: Python 视觉 03

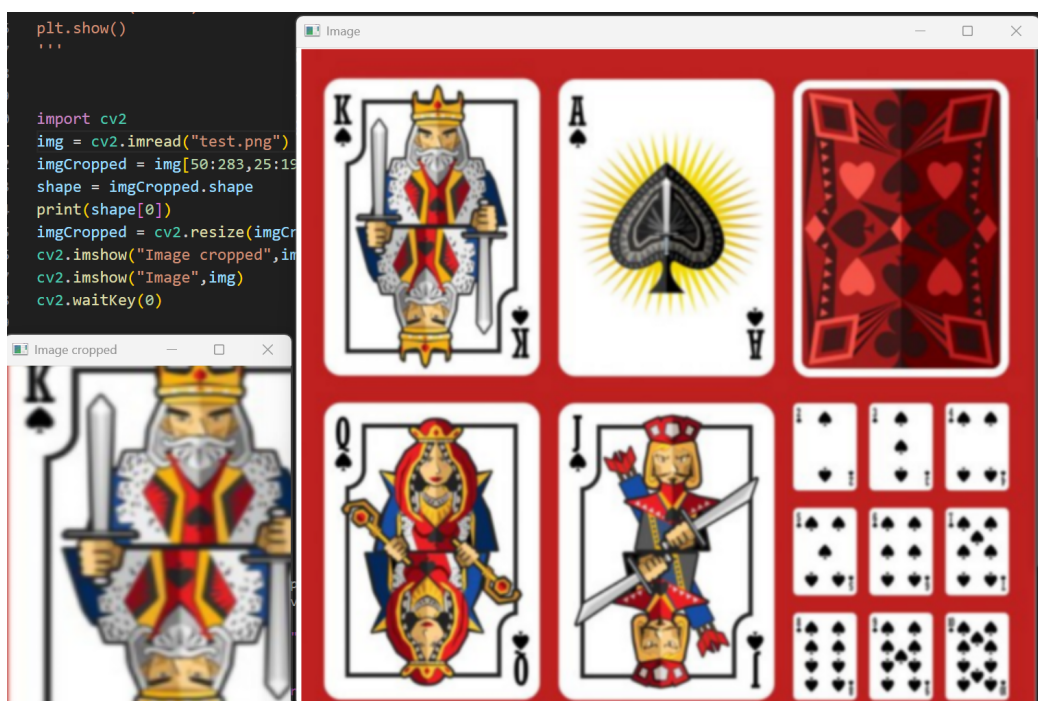


Figure 9: Python 视觉 04



Figure 10: Python 视觉 05