Shell 与 Vim

耿惠生 23020007028

September 2024

1 练习内容

1.1 Shell

1.1.1 ls 拓展操作

ls -lah –sort=time –color
-l：以长格式列出文件，显示详细信息
-a：包括所有文件和隐藏文件
-h：以人类可读的格式显示文件大小
–sort=time：根据最近访问的时间排序文件
–color：以彩色文本显示输出结果

1.1.2 课后习题 2

vim：# marco.sh
MARCO_DIR=””
marco() {
MARCO_DIR=$(pwd)
echo ”Current directory saved: $MARCO_DIR”
}
polo() {
if [-z ”$MARCO_DIR”]; then
echo ”No directory saved. Please run ’marco’ first.”
else
cd ”$MARCO_DIR” || echo ”Failed to change directory to $MARCO_DIR”

echo ”Changed directory to: $MARCO_DIR” fi
}

1.1.3 课后习题 3

#!/bin/bash
COMMAND=”error.sh”
LOG_FILE=”output.txt”
RUN_COUNT=0
while true; do
$COMMAND » ”$LOG_FILE” 2>&1
if [$? -ne 0]; then
echo ”Command failed after $RUN_COUNT runs.” break
fi
RUN_COUNT=$((RUN_COUNT + 1))
done
echo ”Output and errors from the command:”
cat ”$LOG_FILE”

1.1.4 课后习题 4

find /home/learn/shell -type f -name ”*.html” -print0 | zip -@ output.zip

-type f : 只查找文件

`-name "*.html"`: 查找所有扩展名为 `.html` 的文件

`-print0`: 以 null 字符分隔输出文件名，这样可以正确处理文件名中的空格和特殊字符

`zip -@ output.zip`: 从标准输入读取文件名并将它们压缩到 `output.zip` 文件中

1.1.5 课后习题 5

find /home/learn/shell -type f -atime -7

`-atime -7`: 查找最近 7 天内访问过的文件。您可以根据需要调整数字以查找不同时间范围内的文件

1.1.6 检查文件是否存在

```
FILE="/home/learn/file.txt"
if [ -f "$FILE" ]; then
echo "$FILE 存在。"
else
echo "$FILE 不存在。"
fi
```

1.1.7 遍历文件

```
for FILE in *; do
if [ -f "$FILE" ]; then
echo " 文件: $FILE"
fi
done
```

1.1.8 计算 1 到 10 的总和

```
sum=0
for i in {1..10}; do
sum=$((sum + i))
done
echo "1 到 10 的总和是: $sum"
```

1.1.9 简单输入输出

```
if [ $ -eq 0 ]; then
echo " 请提供参数。"
exit 1
fi
echo " 您提供的参数是: $1"
```

1.1.10 简单输出日期

```
#!/bin/bash
echo " 当前日期和时间: $(date)"
```

1.2 Vim

1.2.1 打开文件

vim learn.sh

1.2.2 插入模式

按 `i` 进入插入模式，可以开始编辑文本

1.2.3 保存文件

在普通模式下，输入 `:w` 并按 `Enter` 保存文件

1.2.4 退出 Vim

输入 `:q` 并按 `Enter` 退出 Vim。如果文件有修改，需先保存

1.2.5 保存并退出

输入 `:wq` 或 `:x` 并按 `Enter` 保存文件并退出

1.2.6 强制退出

输入 `:q!` 并按 `Enter` 强制退出，不保存修改

1.2.7 删除一行

在普通模式下，输入 `dd` 删除当前行

1.2.8 复制粘贴

复制当前行：输入 `yy`
粘贴：在光标位置输入

1.2.9 撤销和重做

撤销上一步操作：输入 `u`
重做上一步操作：输入 `Ctrl + r`

1.2.10 搜索文本

输入 `/search_term` 并按 `Enter` 搜索指定的文本，使用 `n` 跳转到下一个匹配项

2 使用感悟

2.1 Shell 和脚本

2.1.1 提高效率

通过编写脚本，可以将一些重复性的任务自动化，从而节省时间和精力。这对于我们处理大量数据或执行复杂操作的场景非常有用。

2.1.2 调试困难

由于 shell 脚本是逐行执行的，调试起来有难度。我需要仔细检查每一行代码，确保没有语法错误和逻辑错误。

2.2 Vim

2.2.1 模式切换

vim 拥有多个模式，包括命令模式、插入模式和可视模式。这需要我们熟练掌握每个模式的切换方法。

2.2.2 快捷键

vim 拥有很多快捷键，这让我们在不用鼠标的情况下就可以完成文本编辑。熟练掌握这些快捷键可以提高我们文本编辑的速度。

3 github 地址

`https://github.com/211-sss/homework`