系统开发工具基础实验报告

实验内容：实验二

姓名：张家宜　　学号：*20240013045*

日期：*2025* 年 *9* 月 *16* 日

目录

1 练习内容

1.1 Shell 工具与脚本

1.1.1 命令与 PATH

```
echo $PATH
which ls
```

$PATH 决定可执行文件的搜索路径
which/command -v 可查看命令解析结果

1.1.2 目录与导航

```
pwd
cd ..
ls -lah
```

查看当前目录、切换到上级、以更可读方式列目录（含隐藏文件）

1.1.3 重定向与追加

```
echo "hello" > out.txt
echo "world" >> out.txt
cat < out.txt
```

> 覆盖写入，>> 追加写入，< 将文件作为标准输入

1.1.4 管道组合

```
ls / | head -n 5
```

将一个命令的输出作为下一个命令的输入（|）

1.1.5 通配（globs）

```
echo *.txt
echo data/??.csv
```

* 匹配任意串，? 匹配单个字符，[abc] 字符类

1.1.6 引号与变量展开

```
echo "$HOME"
echo '$HOME'
```

双引号会展开变量
单引号原样输出

1.1.7 命令替换

```
echo "files: $(ls | wc -l)"
```

$(...) 将子命令输出嵌入当前命令参数

1.1.8 提权写文件

```
echo 3 | sudo tee /tmp/demo
```

重定向由 shell 执行
使用 sudo 时可借助 tee

1.1.9 find：按名称与类型查找

```
find . -name "*.txt" -type f
```

在当前目录递归查找普通文件

1.1.10 find：按时间过滤

```
find . -mtime -1 -type f
```

筛选最近一天内修改的文件

1.2 Shell 脚本

1.2.1 shebang 与可执行权限

```
#!/usr/bin/env bash
# 保存为 scripts/demo.sh
# 赋予可执行权限：chmod +x scripts/demo.sh
# 运行：./scripts/demo.sh foo bar
```

```
5
6 echo "script path: $0"
7 echo "args: $@"
8 echo "count: $#"
```

#!/usr/bin/env bash 通过 PATH 查找解释器，增强可移植性；$0 为脚本名，$@ 为全部参数，$ 为参数个数

1.2.2 参数校验与循环

```
1  #!/usr/bin/env bash
2  # 统计传入文件的行数；若无参数则给出用法并退出
3
4  if [ $# -lt 1 ]; then
5    echo "Usage: $0 FILE..." >&2
6    exit 1
7  fi
8
9  for f in "$@"; do
10   if [ -f "$f" ]; then
11     wc -l < "$f"
12   else
13     echo "skip: $f (not a file)"
14   fi
15 done
```

用 [ ] 做条件判断，-f 测试普通文件；"$@" 保留每个参数的整体性；exit 1 表示非零退出码用于指示错误

1.3 编辑器 (Vim)

1.3.1 多模式与保存退出

```
1 # 正常模式 <ESC>
2 # 插入模式 i / a / o / O
3 # 命令行模式 :
4 :w      " 保存
5 :q      " 退出
6 :wq     " 保存并退出
```

```
:q!      " 强制退出
```

1.3.2 基本移动

```
h  j  k  l          " 左下上右
w  b  e             " 以单词为单位前进/后退/到词尾
0  $                " 行首/行尾
gg  G               " 文件开头/文件结尾
```

在正常模式下用 hjkl 移动
w/b/e 按单词移动
0/$ 行首/行尾
gg/G 文件首/尾

1.3.3 查找与定位

```
/pattern            " 向下搜索
?pattern            " 向上搜索
n / N               " 下一个 / 上一个匹配
* / #               " 以光标下单词为关键词向下/向上搜索
```

使用正斜杠/问号进行前后向搜索
n/N 在匹配间跳转
* / # 以当前单词为关键词快速定位

1.3.4 操作符

```
d{motion}           " 删除到 {motion}
c{motion}           " 改写到 {motion}
y{motion} / p       " 复制 (yank) / 粘贴
u / <C-r>           " 撤销 / 重做
.                   " 重复上一次修改
```

dw 删除到下一个词起点，cw 改写该范围
. 可重复上一步修改

1.3.5 文本对象

```
ci"   ci'   ci(   ci[   " 改写引号/括号内文本
da"   da'   da(   da[   " 删除“包含定界符”的一段
```

结合 i/a 与引号、括号等精准作用于文本，如 ci" 与 da(

1.3.6 可视模式

```
v     V     <C-v>        " 字符 / 行 / 块可视
y / d / > / <             " 复制 / 删除 / 增加缩进 / 减少缩进
gu / gU                   " 转小写 / 转大写
```

进入可视模式选择区域后，可进行复制、删除、缩进与大小写转换等操作

1.3.7 缩进与格式化

```
>> / <<                   " 行级缩进 / 反缩进
=                         " 对选中区域自动缩进
gg=G                      " 对全文进行缩进格式化
```

1.3.8 缓冲区 / 窗口 / 标签页

```
:ls                       " 查看缓冲区
:b 2                      " 切到编号 2 的缓冲区
:split  /  :vsplit        " 水平 / 垂直分屏
:tabnew                   " 新建标签页
:bd                       " 关闭当前缓冲区
```

通过缓冲区在内存中同时打开多个文件，结合分屏与标签页组织多文件编辑

1.3.9 宏

```
q a         " 开始录制到寄存器 a
...         " 执行一系列编辑命令
q           " 结束录制
@a          " 回放宏 a
@@          " 重复上一次回放
```

把重复性编辑操作录成宏，批量回放以提升效率

1.3.10 ~/.vimrc

```
" ~/.vimrc
set number
set ignorecase smartcase
set tabstop=2 shiftwidth=2 expandtab
set hlsearch incsearch
syntax on
filetype plugin indent on
```

开启行号、智能大小写搜索、统一缩进与语法高亮等基础配置

2 解题感悟

通过本次实验，我体会到 Shell 在类 Unix（如 Linux）环境下能把零散的任务高效地“拼起来”借助重定向与管道把简单命令按需组合（例如 `cmd1 | cmd2 | cmd3`），再配合 `find`/`grep`（或 `rg`）等基础工具，就能快速完成查找、过滤、统计等常见工作；把重复流程写成脚本并加入可读的参数与退出码，不仅省时，还提升了可重复性与可维护性。

Vim 则是在终端环境下非常顺手的文本编辑器：“动词 + 动作”让移动与修改形成可预测、可组合的操作；文本对象与可视模式能够精确地选中并批量改写结构化内容；`~/.vimrc` 配置即可显著改善默认体验，而丰富的插件生态也支持按需扩展（如文件跳转、代码搜索等）。

参考文献

[1] Missing Semester 中文版：课程概览与 shell，https://missing-semester-cn.github.io/2020/course-shell/

[2] Missing Semester 中文版：Shell 工具和脚本，https://missing-semester-cn.github.io/2020/shell-tools/

[3] Missing Semester 中文版：编辑器 (Vim)，https://missing-semester-cn.github.io/2020/editors/

GitHub 链接

https://github.com/Misasasasasaka/report/tree/main/P2