系统开发工具基础实验报告 2

姓名：刘浩洋

学号：24040021022

班级：软件工程

实验日期：2025 年 9 月 5 日

一、实验目的

1. 掌握 Linux 系统中 Shell 的基本命令，包括文件、目录和路径操作；
2. 熟悉 Shell 脚本编程基础，包括变量、数组、条件判断、循环结构；
3. 掌握 Vim 文本编辑器的基本使用方法，包括插入、删除、查找、替换等操作；
4. 能够独立编写并运行简单的 Shell 脚本，完成自动化任务；
5. 培养在虚拟机环境下使用 Ubuntu 进行系统开发工具实践的能力。

二、实验环境

- 操作系统：Windows 11 家庭中文版
- 虚拟化平台：VMware Workstation Pro 17
- 虚拟机系统：Ubuntu 24.04.3 LTS（64 位）
- Shell 环境：Bash 5.2.21
- 编辑器：Vim 9.0
- 网络环境：校园网有线连接

三、练习内容

本次实验围绕 Linux 系统开发工具展开，主要包括以下三方面内容：

1. **Shell 基础操作**：练习文件创建、复制、移动、删除，目录创建与切换，绝对与相对路径使用；
2. **Shell 脚本编程**：编写脚本实现打印输出、变量赋值、数组定义、条件选择（if/else）、循环控制（for/while）等功能；
3. **Vim 编辑器使用**：在 Vim 中编辑 Shell 脚本，练习进入插入模式、保存退出、查找替换、行号显示等常用功能。

四、20 个实例（shell 与 vim）

实例 1：创建实验目录结构

```
# 创建实验主目录
mkdir -p ~/lab1/{scripts,data,docs}
# 进入主目录
cd ~/lab1
# 查看目录结构
tree
# 创建测试文件
touch data/file1.txt data/file2.txt
# 查看文件列表
ls -l data/
# 显示当前路径
pwd
```

实例 2: 文件复制与重命名

```
# 复制文件到脚本目录
cp data/file1.txt scripts/
# 重命名文件
mv scripts/file1.txt scripts/test1.bak
# 批量复制
cp data/*.txt scripts/
# 查看目标目录
ls scripts/
# 显示文件详细信息
stat scripts/test1.bak
# 删除原文件（保留副本）
rm data/file1.txt
```

实例 3: 使用绝对与相对路径

```
# 显示当前目录
echo "当前目录: $(pwd)"
# 使用绝对路径访问
cd /home/$(whoami)/lab1/docs
# 返回上级目录
cd ..
# 进入scripts目录（相对路径）
cd scripts
# 显示绝对路径
realpath .
# 列出父目录内容
ls ../data
# 返回家目录
cd ~
```

实例 4: 查看与修改文件权限

```
# 查看scripts目录权限
ls -l scripts/
# 修改文件执行权限
chmod +x scripts/test1.bak
# 添加读写权限
chmod 664 scripts/test1.bak
# 查看修改后权限
stat scripts/test1.bak | grep "Access"
# 递归修改目录权限
chmod -R 755 scripts/
# 显示所有权限信息
ls -la scripts/
# 测试执行权限
./scripts/test1.bak
```

实例 5: 编写第一个 Shell 脚本（打印信息）

```
# 创建脚本文件
cat > scripts/hello.sh << 'EOF'
#!/bin/bash
# 打印欢迎信息
echo "========================"
echo "  欢迎使用Shell脚本! "
echo "  当前用户: $(whoami)"
echo "  主机名: $(hostname)"
echo "  系统时间: $(date)"
echo "========================"
EOF
# 添加执行权限
chmod +x scripts/hello.sh
# 运行脚本
./scripts/hello.sh
```

实例 6：Shell 变量定义与使用

```
# 创建变量脚本
cat > scripts/vars.sh << 'EOF'
#!/bin/bash
# 定义字符串变量
name="刘浩洋"
course="系统开发工具"
# 定义数值变量
year=2025
lab_num=1
# 输出变量值
echo "姓名：$name"
echo "课程：$course"
echo "年份：$year"
echo "实验编号：$lab_num"
EOF
chmod +x scripts/vars.sh
./scripts/vars.sh
```

实例 7：环境变量操作

```
# 查看常用环境变量
echo "用户：$USER"
echo "家目录：$HOME"
echo "路径：$PATH"
echo "主机名：$HOSTNAME"
# 设置临时环境变量
export MY_VAR="实验变量"
# 在子shell中验证
bash -c 'echo "子shell中MY_VAR=$MY_VAR"'
# 查看所有环境变量
env | grep MY_VAR
# 清除变量
unset MY_VAR
# 验证清除
echo "清除后：$MY_VAR"
```

实例 8：Shell 数组操作

```
# 创建数组脚本
cat > scripts/array.sh << 'EOF'
#!/bin/bash
# 定义数组
fruits=("苹果" "香蕉" "橙子" "葡萄")
# 输出数组所有元素
echo "水果列表：${fruits[@]}"
# 输出数组长度
echo "数量：${#fruits[@]}"
# 访问特定元素
echo "第一个：${fruits[0]}"
echo "最后一个：${fruits[-1]}"
# 遍历数组
for fruit in "${fruits[@]}"; do
    echo " - $fruit"
done
EOF
chmod +x scripts/array.sh
./scripts/array.sh
```

实例 9: 使用 read 读取用户输入

```
# 创建输入脚本
cat > scripts/input.sh << 'EOF'
#!/bin/bash
echo "请输入您的姓名: "
read user_name
echo "请输入学号: "
read student_id
# 输出欢迎信息
echo "欢迎你, $user_name! "
echo "学号: $student_id"
echo "当前时间: $(date)"
# 验证输入是否为空
if [ -z "$user_name" ]; then
    echo "警告: 姓名为空! "
fi
EOF
chmod +x scripts/input.sh
./scripts/input.sh
```

实例 10：if 条件判断（数字比较）

```
# 创建判断脚本
cat > scripts/condition.sh << 'EOF'
#!/bin/bash
echo "请输入一个数字: "
read num
if [ $num -gt 0 ]; then
    echo "$num 是正数"
elif [ $num -lt 0 ]; then
    echo "$num 是负数"
else
    echo "$num 是零"
fi
# 判断奇偶
if [ $(($num % 2)) -eq 0 ]; then
    echo "$num 是偶数"
else
    echo "$num 是奇数"
fi
EOF
chmod +x scripts/condition.sh
./scripts/condition.sh
```

实例 11：case 多分支选择

```
# 创建菜单脚本
cat > scripts/menu.sh << 'EOF'
#!/bin/bash
echo "请选择操作："
echo "1) 查看日期"
echo "2) 查看磁盘"
echo "3) 查看内存"
echo "4) 退出"
read choice
case $choice in
    1)
        date
        ;;
    2)
        df -h
        ;;
    3)
        free -h
        ;;
    4)
        echo "再见！"
        exit 0
        ;;
    *)
        echo "无效选择"
        ;;
esac
EOF
chmod +x scripts/menu.sh
./scripts/menu.sh
```

实例 12：for 循环遍历文件

```
# 创建循环脚本
cat > scripts/for_files.sh << 'EOF'
#!/bin/bash
# 遍历scripts目录下的.sh文件
echo "脚本文件列表："
for file in scripts/*.sh; do
    if [ -f "$file" ]; then
        echo " - $(basename $file)"
        # 显示文件大小
        size=$(stat -c%s "$file")
        echo "   大小：$size 字节"
    fi
done
echo "遍历完成。"
EOF
chmod +x scripts/for_files.sh
./scripts/for_files.sh
```

实例 13：while 循环计数

```
# 创建计数脚本
cat > scripts/while_count.sh << 'EOF'
#!/bin/bash
count=1
echo "倒计时开始："
while [ $count -le 5 ]; do
    echo "$count..."
    sleep 1
    count=$((count + 1))
done
echo "发射！  "
# 验证循环变量
echo "最终count值：$count"
EOF
chmod +x scripts/while_count.sh
./scripts/while_count.sh
```

实例 14：until 循环使用

```
# 创建until脚本
cat > scripts/until_loop.sh << 'EOF'
#!/bin/bash
num=10
echo "从10倒数到1："
until [ $num -lt 1 ]; do
    echo "$num"
    num=$((num - 1))
done
echo "结束！"
# 验证最终值
echo "循环后num=$num"
EOF
chmod +x scripts/until_loop.sh
./scripts/until_loop.sh
```

实例 15：函数定义与调用

```
# 创建函数脚本
cat > scripts/functions.sh << 'EOF'
#!/bin/bash
# 定义函数
greet() {
    local name=$1
    echo "你好, $name! "
}

# 调用函数
greet "张三"
greet "李四"

# 带返回值的函数
add() {
    local a=$1
    local b=$2
    return $(($a + $b))
}

add 5 3
result=$?
echo "5 + 3 = $result"
EOF
chmod +x scripts/functions.sh
./scripts/functions.sh
```

实例 16: 使用 Vim 创建脚本文件

```
# 使用Vim创建新脚本
vim scripts/vim_test.sh
# 在Vim中执行以下操作:
# 1. 按 i 进入插入模式
# 2. 输入以下内容:
<<VIM_CONTENT>>
#!/bin/bash
echo "这是通过Vim编辑的脚本"
echo "编辑时间: $(date)"
echo "作者: 刘浩洋"
# 显示参数个数
echo "参数数量: $#"
# 遍历参数
for arg in "$@"; do
    echo "参数: $arg"
done
<<VIM_CONTENT>>
# 3. 按 Esc 退出插入模式
# 4. 输入 :wq 保存并退出
# 5. 添加执行权限
chmod +x scripts/vim_test.sh
# 6. 运行测试
./scripts/vim_test.sh 1 2 3
```

实例 17: Vim 查找与替换

```
# 复制脚本用于编辑
cp scripts/hello.sh scripts/vim_edit.sh
# 使用Vim打开
vim scripts/vim_edit.sh
# 在Vim中:
# 1. 按 / 搜索 "欢迎"
# 2. 按 n 跳转到下一个匹配
# 3. 按 N 跳转到上一个匹配
# 4. 按 i 修改文本为 "欢迎来到Linux世界"
# 5. 使用 :%s/echo/printf/g 替换所有echo为printf
# 6. 使用 :set number 显示行号
# 7. 使用 :set nonumber 隐藏行号
# 8. 保存退出 :wq
# 查看修改后内容
cat scripts/vim_edit.sh
```

实例 18: Vim 多文件编辑

```
# 创建多个文件
touch scripts/file{1..3}.txt
# 使用Vim同时打开
vim scripts/file1.txt scripts/file2.txt scripts/file3.txt
# 在Vim中:
# 1. 编辑file1.txt, 输入 "这是文件1"
# 2. 按 :w 保存
# 3. 按 :n 切换到下一个文件
# 4. 输入 "这是文件2"
# 5. 按 :n 切换到最后一个
# 6. 输入 "这是文件3"
# 7. 按 :wa 保存所有文件
# 8. 按 :qall 退出所有
# 验证内容
cat scripts/file*.txt
```

实例 19：Shell 脚本调试

```
# 创建可能出错的脚本
cat > scripts/debug.sh << 'EOF'
#!/bin/bash
set -x  # 启用调试模式
name="测试"
echo "开始执行"
if [ $name = "test" ]; then
    echo "匹配"
else
    echo "不匹配"
fi
# 模拟错误
ls /nonexistent/path
echo "脚本结束"
set +x  # 关闭调试
EOF
chmod +x scripts/debug.sh
# 运行并观察输出
./scripts/debug.sh
# 使用bash -n检查语法
bash -n scripts/debug.sh
```

实例 20：综合脚本（文件备份）

```
# 创建备份脚本
cat > scripts/backup.sh << 'EOF'
#!/bin/bash
# 备份data目录到backup
backup_dir="backup_$(date +%Y%m%d_%H%M%S)"
mkdir "$backup_dir"
# 复制所有.txt文件
for file in data/*.txt; do
    if [ -f "$file" ]; then
        cp "$file" "$backup_dir/"
        echo "已备份：$(basename $file)"
    fi
done
# 显示备份内容
echo "备份完成，文件列表："
ls "$backup_dir"
# 打包备份目录
tar -czf "${backup_dir}.tar.gz" "$backup_dir"
echo "已压缩为 ${backup_dir}.tar.gz"
rm -rf "$backup_dir"
EOF
chmod +x scripts/backup.sh
./scripts/backup.sh
```

五、实验结果

- 成功在 Ubuntu 24.04.3 虚拟机中完成全部 20 个 Shell 与 Vim 实例操作；
- 所有脚本均能正确运行，输出符合预期；
- 掌握了文件、目录、路径的基本操作命令；
- 熟练使用 Vim 编辑器创建和修改 Shell 脚本；
- 实现了变量、数组、条件、循环、函数等 Shell 编程结构；
- 所有实验代码已提交至 GitHub 仓库，可通过链接访问。

六、解题感悟

通过本次实验，我深刻体会到 Linux Shell 的强大与灵活。Shell 不仅是命令行工具，更是一种高效的自动化编程语言。从最初对 Vim 的不适应，到能够熟练使用其进行脚本编辑，我认识到掌握基础工具对系统开发的重要性。Shell 脚本的简洁语法和强大功能让我能够快速实现文件管理、系统监控等任务。同时，在虚拟机环境中操作也增强了我对 Linux 系统架构的理解。未来我将继续深入学习 Shell 高级编程和系统管理知识，为后续课程和项目开发打下坚实基础。

七、GitHub 链接

实验代码已上传至 GitHub，仓库地址为：`https://github.com/ouc-lhy/for-lesson/tree/master/lesson2`