

天津医科大学实验课教案首页

(共 3 页、第 1 页)

课程名称：Linux 系统概论		实验名称：实验 8 shell 脚本编程	
教师姓名：伊现富	职称：讲师	教学日期：2018 年 7 月 2&9 日 13:30-15:30	
授课对象：生物医学工程学院 2016 级生信班（本）			实验人数：28
实验类型（验证型、综合型、设计型、创新型）：验证型			实验分组：一人一机
学时数：4		教材版本：Linux 系统概论上机指南（自编教材）	

实验目的与要求：

- 掌握 shell 脚本的结构和运行方法。
- 掌握 shell 函数的使用方法。
- 掌握 shell 脚本中的流程控制。
- 掌握 shell 脚本的调试方法。

实验内容及学时分配：

- (5') 编程起步：回顾 shell 脚本的基本结构和运行方法，总结调试 shell 脚本的主要方法。
- (10') 流程控制：回顾 if-then、case 和 while、until、for 等条件流程控制和迭代流程控制的基本语法。
- (5') shell 函数：回顾声明和调用 shell 函数的基本语法，总结嵌套和递归、作用域等相关知识点。
- (80') 实验操作：练习 shell 脚本的编写，掌握 shell 脚本的基本使用。

主要仪器和实验材料：

- 主要仪器：一台安装有 CentOS 计算机。

实验重点、难点及解决策略：

- 重点难点：shell 中的流程控制语法，shell 函数的使用。
- 解决策略：通过实例进行讲解，通过演示进行学习，通过练习熟练掌握。

思考题：

- 如何给变量赋值、访问变量的值？
- 总结各种条件流程控制和迭代流程控制的语法结构。
- 列举常见的文件测试并解释其含义。
- 字符串和整数值比较的运算符有哪些？
- 如何声明并调用 shell 函数？
- 如何对 shell 脚本进行调试？

参考资料：

- Linux 基础及应用习题解析与实验指导（第二版），谢蓉 编著。中国铁道出版社，2014。

主任签字：

年 月 日

教务处制

一、编程起步 (5 分钟)

1. 脚本结构与运行方法

(1) 脚本结构

- `#!` (`shebang` 结构) 调用 `shell`
- `#` 进行注释
- 命令和控制结构

(2) 运行方法

- 创建文件: `vim script.sh`
- 修改权限: `chmod u+x script.sh`
- 运行脚本: `./script.sh`, `sh script.sh`

2. 变量使用

- 赋值: 使用 `=` (赋值运算符), `=` 两边不能有空格, 在 `=` 后面跟一个换行符赋空值
- 访问: 默认情况下将变量视作文本字符串, 在变量名前加 `$` 可以访问变量的值
- 变量名: 只能包含字母、数字和下划线, 必须以字母或下划线开头, 大小写敏感 (惯例使用大写)

3. 脚本调试 (5 分钟)

- `sh -n script`: 检查语法, 不执行脚本。
- `sh -x script`: 执行脚本并显示所有变量的值。
- `set -x; command; set +x`: 脚本局部调试。
- `echo`: 打印变量的值、提示信息等来调剂脚本。

二、流程控制 (10 分钟)

1. 简介

- 流程控制允许程序做出判断: 程序计算条件的值, 并根据这些条件执行相应的操作
- 条件流程控制: 根据特定的约束是否满足来决定是否执行某个代码段
- 迭代流程控制: 代码块重复或迭代, 直到满足某个条件为止

2. 条件流程控制

• if-then

– if-then

```
if some_condition
then
    something happens
fi
```

– if-then-else

```
if some_condition
then
    something happens
else
    something happens
fi
```

– if-then-elif-else

```
if some_condition
then
    something happens
elif other_condition
    something happens
else
    something happens
fi
```

• 比较运算符

- 字符串比较
- 整数值比较

测试	助记	功能
<code>-d</code>	Directory	文件存在并且是目录
<code>-e</code>	Exist	指定的文件存在
<code>-f</code>	File	文件存在并且是普通文件
<code>-G</code>	Group	执行命令的用户属于文件所属组的成员
<code>-nt</code>	Newer Than	file1 -nt file2, 前一个文件比后一个文件新
<code>-ot</code>	Older Than	file1 -ot file2, 前一个文件比后一个文件老
运算符	示例	功能
<code>=</code>	<code>stringA=stringB</code>	stringA 与 stringB 相同
<code>!=</code>	<code>stringA!=stringB</code>	stringA 与 stringB 不同
<code>></code>	<code>stringA>stringB</code>	stringA 大于 stringB
<code><</code>	<code>stringA<stringB</code>	stringA 小于 stringB
<code>-z</code>	<code>-z string</code>	string 为空
<code>-n</code>	<code>-n string</code>	string 非空

运算符	助记	功能
<code>-eq</code>	Equal	等于
<code>-ne</code>	Not Equal	不等于
<code>-gt</code>	Greater Than	大于
<code>-lt</code>	Lesser Than	小于
<code>-ge</code>	Greater or Equal	大于等于
<code>-le</code>	Lesser or Equal	小于等于

- test
 - 语法


```
# 注意中括号内的空格
if [ $COLOR="purple" ]
if (test $COLOR="purple")

# 测试文件是否存在
if [ -e filename ]
if (test -e filename)
```
 - 文件测试
 - 逻辑运算符 (&&、||)
- case


```
case expression in
    pattern1)
        action1
        ;;
    pattern2)
        action2
        ;;
    *)
        default action
        ;;
esac
```

3. 迭代流程控制

- | | | |
|--|--|--|
| <ul style="list-style-type: none"> • while <pre>while condition do action done</pre> | <ul style="list-style-type: none"> • until <pre>until condition do action done</pre> | <ul style="list-style-type: none"> • for <pre>for VAR in LIST do action done</pre> |
|--|--|--|

三、 shell 函数 (5 分钟)

1. 声明与调用

<pre># 函数声明 name () { commands } # 函数调用, 不带 () name name parameter(s)</pre>	<pre>#!/bin/bash # A simple function repeat () { echo "I don't know \$1 \$2" } repeat Your Name # I don't know Your Name</pre>
---	--

2. 嵌套和递归

- 递归函数: 就像调用其他函数一样、调用自己的函数
- 函数嵌套: 在一个函数中调用包含在另一个函数中的功能

3. 作用域

- 全局作用域: 在脚本的任何地方都可以访问变量
- 局部作用域: 只能在声明变量的作用域内访问它们
- 使用 **local** 关键字定义局部变量

四、 实验操作 (80 分钟)

1. 简单的 shell 脚本 (运行方法)
2. shell 函数 (传递参数, 嵌套, 作用域)
3. shell 脚本实例 (read, if-then, test, case, for)