



21世纪高等学校电子信息类专业规划教材

Linux C 程序设计

——实例详解与上机实验

主 编 秦攀科
副主编 田 苗 张 新 张 雷
主 审 谢友宝



清华大学出版社
<http://www.tup.tsinghua.edu.cn>



北京交通大学出版社
<http://press.bjtu.edu.cn>



21 世纪高等学校电子信息类专业规划教材

Linux C 程序设计 ——实例详解与上机实验

主 编 秦攀科

副主编 田 苗 张 新 张 雷

主 审 谢友宝

清华大学出版社
北京交通大学出版社

· 北京 ·

内 容 简 介

本书为秦攀科主编的《Linux C 程序设计基础》一书的配套实验教材,在教材基础上对每一个知识点补充实例讲解,并为每章内容配备大量的上机实验练习供读者参考实践。本教材最突出的特色是以练促学,每一个语法知识点都提供了丰富的实例代码,在编写代码的过程中力求所有的实例代码都来源于实际开发的项目,使读者可以接触到第一线的源码获取实际的开发经验。

本书内容翔实,讲解透彻,具有很强的可读性,适合作为高等院校计算机专业教材,也适合程序设计的初学者使用,还可以作为计算机爱好者的自学参考书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13501256678 13801310933

图书在版编目(CIP)数据

Linux C 程序设计:实例详解与上机实验/秦攀科主编. —北京:清华大学出版社;北京交通大学出版社,2011.8

(21 世纪高等学校电子信息类专业规划教材)

ISBN 978-7-5121-0668-0

I. ① L… II. ① 秦… III. ① Linux 操作系统-程序设计-高等学校-教学参考资料 ② C 语言-程序设计-高等学校-教学参考资料 IV. ① TP316.89 ② TP312

中国版本图书馆 CIP 数据核字(2011)第 155045 号

责任编辑:郭东青

出版发行:清华大学出版社 邮编:100084 电话:010-62776969

北京交通大学出版社 邮编:100044 电话:010-51686414

印刷者:北京交大印刷厂

经 销:全国新华书店

开 本:185×260 印张:12.5 字数:312 千字

版 次:2011 年 8 月第 1 版 2011 年 8 月第 1 次印刷

书 号:ISBN 978-7-5121-0668-0/TP·653

印 数:1~4 000 册 定价:23.00 元

本书如有质量问题,请向北京交通大学出版社质监组反映。对您的意见和批评,我们表示欢迎和感谢。
投诉电话:010-51686043, 51686008; 传真:010-62225406; E-mail: press@bjtu.edu.cn。

前 言

本书为《Linux C 程序设计基础》的配套实验教材。对原教材每一个关键知识点均辅以实例进行讲解分析，从而使读者更加透彻地理解教材重点和难点，同时配以大量的实验练习供读者上机实践操作检验和巩固学习成果。内容分为三部分。

第一部分 Linux 系统基础入门

本部分是学习 Linux C 程序设计的基础部分，使读者快速地熟悉 Linux 系统的基本结构和操作命令，掌握 Linux C 程序的开发与编译环境。共包含两章内容：Linux 系统入门和 Linux C 程序设计简介。详细介绍了 Linux 操作系统基本结构、常用命令和 Linux 环境下 C 的开发环境和程序编译执行的工具。

第二部分 Linux C 程序设计基础

本部分为本书的核心与重点，结合实例重点介绍了 Linux 利用 C 语言进行程序设计的语法。共包括七章内容：数据类型、运算符和表达式，程序设计基本结构——顺序、选择与循环，数组与指针，函数，结构体，预处理命令和 Linux 文件系统与文件操作。

数据类型、运算符和表达式：介绍了 Linux C 语言的基本数据类型，通用运算符和表达式。程序设计基本结构——顺序、选择与循环：分别介绍了顺序程序设计、选择结构设计和循环控制。函数：从函数的基本概念入手，结合实例介绍了函数的定义、参数和调用方法。预处理命令：分别介绍了宏定义、“文件包含”处理和条件编译。数组与指针、结构体：详细介绍了 C 语言的“灵魂”指针及其操作和使用，结构体的定义和使用。

第三部分 Linux C 高级程序设计

本部分为本书难点与重点，也是进行 Linux 系统程序设计的核心。共包含两章内容：进程与线程和网络通信。从进程和线程的基本概念入手介绍了 Linux 下多线程程序设计。主要内容包括：进程及其通信、高级进程通信和线程。网络编程详细介绍了 Linux 下使用 C 语言开发面向连接的（TCP）和面向非连接的（UDP）方式的网络编程及相关的函数和使用方法。

整体而言，本书内容翔实，讲解透彻。最突出的特色是以练促学，书中给出了丰富的实例供读者实战演练。适合 Linux 的初学者及希望利用 Linux 进行开发的程序设计人员阅读。

本书特色如下。

- 侧重基础，入门简单。

从 Linux 系统基础知识和 C 语言基本语法出发，侧重于每一个知识点的讲解，做到用简单的话阐述最复杂的道理。

- 内容翔实，讲解深入。

内容覆盖基础 C 语言程序设计和 Linux 系统程序设计，尤其是对 Linux 文件系统和文件操作、进程与线程、网络通信等相关系统编程知识进行详细的剖析和深入的讲解。

- 实例丰富，活学活用。

所有知识点都配合实例进行详细讲解，使读者能够更加透彻和全面地理解所述知识点。

本教材共分为 11 章，由秦攀科任主编，田苗、张新、张雷任副主编，霍元媛、杨睿、陈光宣、胡小三、石宁、李凌辉、许彬、王慧群和王志勇参加了编写，谢友宝主审。参与本书编写的还有：秦菊梅和何会来。本书所有程序均上机调试通过。

由于时间仓促，不妥之处欢迎读者批评指正，若有疑问或索取相关资料，可联系作者：
qinpanke@ gmail. com。

编 者
2011 年 7 月

目 录

第1章 Linux 系统入门	1
1.1 知识点实例上机	1
1.2 上机实验练习	7
第2章 Linux C 程序设计简介	11
2.1 知识点实例上机	11
2.1.1 Linux C 程序的编写、编译、连接与运行	11
2.1.2 make 工具与 makefile 文件	13
2.2 上机实验练习	15
第3章 数据类型、运算符和表达式	20
3.1 知识点实例上机	20
3.1.1 Linux C 数据类型	20
3.1.2 常量与变量	21
3.1.3 整型数据	22
3.1.4 实型数据	24
3.1.5 字符型数据	25
3.1.6 符号常量	27
3.1.7 类型转换	28
3.1.8 运算符与表达式	32
3.2 上机实验练习	46
第4章 程序设计基本结构——顺序、选择与循环	50
4.1 知识点实例上机	50
4.1.1 顺序结构程序设计	50
4.1.2 选择结构程序设计	60
4.1.3 循环结构程序设计	67
4.2 上机实验练习	75
第5章 数组与指针	83
5.1 知识点实例上机	83
5.1.1 数组	83
5.1.2 指针	88
5.2 上机实验练习	97

第6章 函数	103
6.1 知识点实例上机	103
6.1.1 函数的定义和声明	103
6.1.2 函数的参数和返回值	106
6.1.3 函数的调用	110
6.1.4 变量的作用范围与存储类型	112
6.2 上机实验练习	117
第7章 结构体	123
7.1 知识点实例上机	123
7.2 上机实验练习	127
第8章 预处理命令	131
8.1 知识点实例上机	131
8.2 上机实验练习	133
第9章 Linux 文件系统与文件操作	134
9.1 知识点实例上机	134
9.1.1 缓冲文件操作	134
9.1.2 非缓冲文件操作	144
9.1.3 临时文件的操作	147
9.2 上机实验练习	149
第10章 进程与线程	156
10.1 知识点实例上机	156
10.1.1 进程	156
10.1.2 线程	168
10.2 上机实验练习	176
第11章 网络通信	178
11.1 知识点实例上机	178
11.1.1 面向连接传输	178
11.1.2 面向非连接传输	184
11.1.3 阻塞与非阻塞	188
11.2 上机实验练习	193

第 1 章 Linux 系统入门

1.1 知识点实例上机

回顾教材知识点，上机操作并调试实例代码进一步掌握 Linux 常用命令和 shell 脚本的重点并达到熟练运用的目的。

- 上机实践，深入了解 Linux 操作系统的使用与基础概念，实现熟练使用 Linux。
- 上机实践 Linux 常用命令：目录操作、文件管理、文档编辑、备份压缩、系统设置管理和磁盘管理维护。
- 上机实践 shell 脚本的编写，shell 脚本中变量应用和程序流程控制方法。

本章主要包括：操作系统基础和基本操作、Linux 常用命令和 shell 脚本。前两部分内容读者可参阅《Linux C 程序设计基础》进行上机实践操作，本章重点是 shell 脚本知识点的实例开发与讲解。

shell 脚本

shell 脚本与 Windows/DOS 下的批处理相似，也就是将各类命令预先放入到一个文件中，方便一次性执行一个程序文件，主要是方便管理员进行设置或者管理用的。首先通过一个简单的 shell 脚本的编写与执行了解和熟悉 shell 脚本，然后深入学习 shell 脚本中的变量和 shell 脚本的程序控制命令，进一步巩固和掌握 shell 脚本程序设计方法。

1. 一个简单的 shell 脚本

【例 1-1】使用 shell 输出打印字符

(1) 创建程序文件 script_test，如图 1-1 所示。

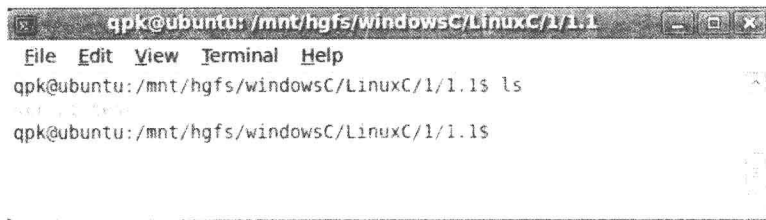


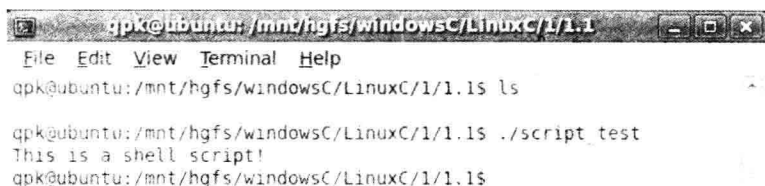
图 1-1 创建 shell 脚本文件

(2) 编辑脚本文件 script_test 如下：

```
1 echo This is a shell script!
```



(3) 运行脚本，如图 1-2 所示。



```
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.1
File Edit View Terminal Help
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.1$ ls
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.1$ ./script_test
This is a shell script!
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.1$
```

图 1-2 运行 shell 脚本

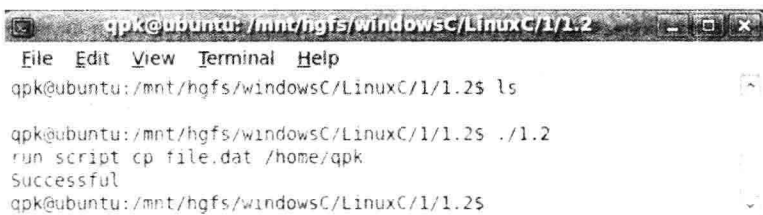
2. shell 脚本中的变量

shell 脚本中的变量分为系统环境变量和用户自定义变量。

【例 1-2】使用系统环境变量 HOME 将当前文件夹下的文件复制到用户 home 路径下。

```
1 echo run script"cp file.dat $HOME"
2 cp file.dat $HOME
3 echo Successful
```

运行结果如图 1-3 所示。



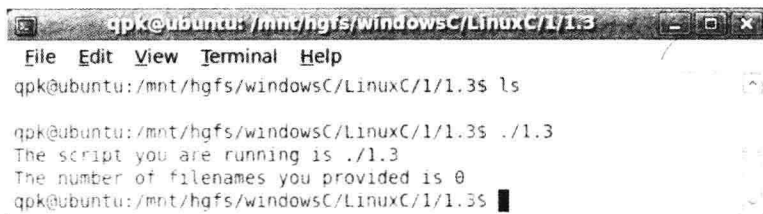
```
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.2
File Edit View Terminal Help
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.2$ ls
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.2$ ./1.2
run script cp file.dat /home/qpk
Successful
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.2$
```

图 1-3 shell 使用系统环境变量

【例 1-3】使用用户定义变量。

```
1 echo The script you are running is $0
2 echo The number of filenames you provided is $#
```

运行结果如图 1-4 所示。



```
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.3
File Edit View Terminal Help
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.3$ ls
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.3$ ./1.3
The script you are running is ./1.3
The number of filenames you provided is 0
qpk@ubuntu: /mnt/hgfs/windowsC/LinuxC/1/1.3$
```

图 1-4 shell 使用用户定义变量

程序说明：

shell 将脚本名分配给位置变量 \$0；将命令行中包含的文件名分别分配给位置变量 \$1，\$2。



3. shell 脚本中的程序流程控制命令

可以运用程序流程控制命令来决定 shell 脚本中各个语句执行的顺序。控制程序流程的基本命令有三种：二路跳转、多路跳转和重复执行。

(1) if 语句。

在 shell 脚本中常用的 if 语句有：if-then 语句、if-then-else 语句和 if-then-elif-else-if 语句。

if-then 语句常用于二路跳转，其语法如下：

```
if expression
then
    then-command
fi
```

【例 1-4】 if-then 语句实现二路跳转。

```
1  #!/bin/bash
2  if [-e ./test]
3  then
4  echo There is a file named test in the current directory.
5  fi
```

运行结果如图 1-5 所示。

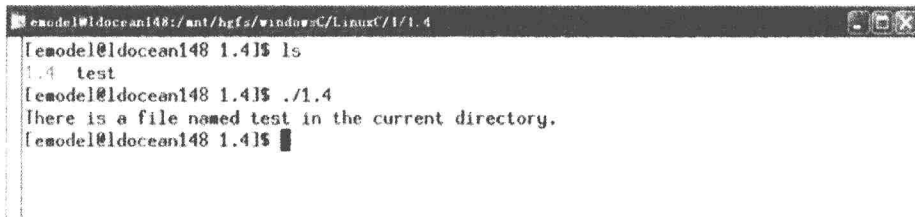


图 1-5 if-then 语句

if-then-else 语句同样可以实现二路跳转，且比较简洁易懂，其语法如下：

```
if expression
then
    then-command
else
    else-command
fi
```

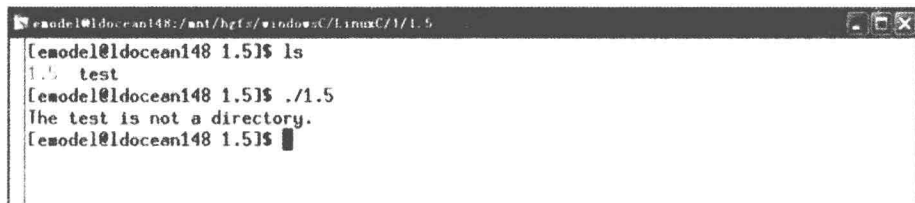
【例 1-5】 if-then-else 语句。

```
1  #!/bin/bash
2  if [-d ./test]
3  then
4  echo The test is a directory
```



```
5 else
6 echo The test is not a directory.
7 fi
```

运行结果如图 1-6 所示。



```
[emodel@ldocean148 1.5]$ ls
1.5 test
[emodel@ldocean148 1.5]$ ./1.5
The test is not a directory.
[emodel@ldocean148 1.5]$
```

图 1-6 if-then-else 语句

if-then-elif-else-fi 语句的语法如下：

```
if expression
then
    ;elif expression
    then
        then-command-list
    :
    ;else
        else-command-list
fi
```

if-then-elif-else-fi 语句能够实现更复杂的多路跳转，完成各种更加复杂的任务。

【例 1-6】 if-then-elif-else-fi 语句的使用。

```
1 #!/bin/bash
2 echo"Enter a file:"
3 read file
4 if [-d$file]
5 then
6 echo"$file is a directory"
7 elif [-f file]
8 then
9 echo"file is a ordinary file"
10 else
11 echo"file is a special file"
12 fi
```

运行结果如图 1-7 所示。



```
ermodel@ldocean148: /mnt/hgfs/windowsC/LinuxC/1/1.6
[ermodel@ldocean148 1.6]$ ls
1.6 test
[ermodel@ldocean148 1.6]$ ./1.6
Enter a file:
1.6
1.6 is a ordinary file
[ermodel@ldocean148 1.6]$
```

图 1-7 if-then-elif-else-fi 语句

(2) 循环语句。

循环语句用于确定程序中某些部分是否应该执行多次。bash shell 中，常用的循环语句有 for 语句、while 语句和 until 语句，关键字 for、in、do、done 等经常会出现这些语句中。

① for 语句。

for 语句的语法如下：

```
for variable [in argument-list]
do
    command-list
done
```

【例 1-7】下面的脚本 for_example 展示了 for 语句的用法，其内容如下：

```
1 [ericcgx@node01 ericcgx]$ more for_example
2 #!/bin/bash
3 for course in Maths Chinese English Biology
4 do
5 echo "$course"
6 done
7 exit 0
```

运行这个脚本，结果如图 1-8 所示。

```
ermodel@ldocean148: /mnt/hgfs/windowsC/LinuxC/1/1.7
[ermodel@ldocean148 1.7]$ ./1.7
Maths
Chinese
English
Biology
[ermodel@ldocean148 1.7]$
```

图 1-8 for 语句用法

② while 语句。

while 语句为判断一个表达式的真假而重复执行一系列语句，其语法如下：

```
while expression
do
    command-list
done
```



【例 1-8】下面的脚本 while_example 展示了 while 语句的用法，其内容如下：

```
1  #!/bin/bash
2  Passwd=888888
3  echo "Guess the password!"
4  echo -n "Enter your answer:"
5  read Answer
6  while [ "$Passwd" != "$Answer" ]
7  do
8  echo "Wrong answer, try again!"
9  echo -n "Enter your answer:"
10 read Answer
11 done
12 echo "The answer is right, congratulation!"
13 exit 0
```

执行这个脚本，结果如图 1-9 所示。

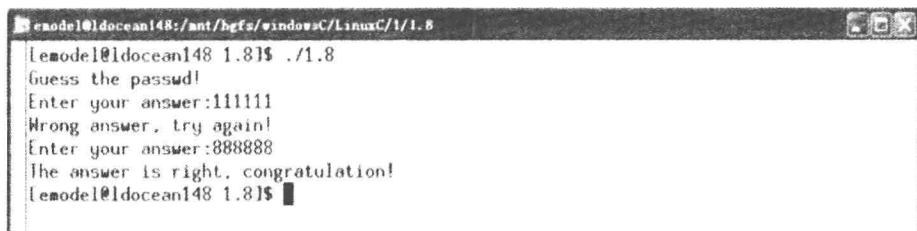


图 1-9 while 语句

程序说明：当运行这个脚本时，变量 Passwd 被初始化为 888888，用户输入的猜测结果则存储在局部变量 Answer 中，如果猜测的不是 888888，while 循环的条件就为 true，就一直在 do 和 done 之间执行操作。程序提示用户的猜测结果为错，并要求再来一次。直到用户猜到了 888888，那样循环条件就变为 false，程序跳出 while 循环，然后执行随后的命令，即用 echo 命令打印出祝贺信息。

③ until 语句。

until 语句的句法和 while 语句类似，但是语义却不同。while 语句只要测试语句的值为 true，则不断执行循环体，而 until 语句中，只要测试语句的值还是 false，则不断执行循环体。until 语句的语法如下：

```
until expression
do
    command-list
done
```

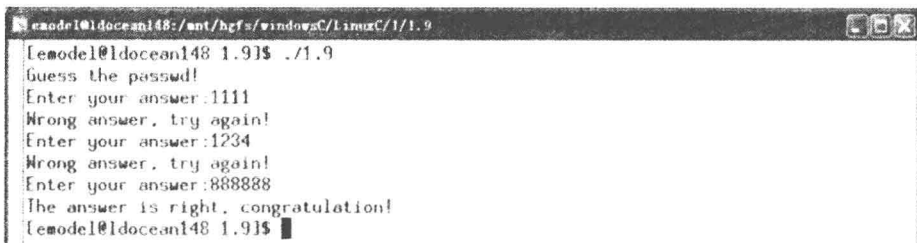
【例 1-9】下面的脚本 until_example 完成了和上述脚本 while_example 一样的工作，其内容如下：

```
1  #!/bin/bash
```



```
2 Passwd=888888
3 echo "Guess the password!"
4 echo -n "Enter your answer:"
5 read Answer
6 until ["$Passwd"="$Answer"]
7 do
8 echo "Wrong answer, try again!"
9 echo -n "Enter your answer:"
10 read Answer
11 done
12 echo "The answer is right, congratulation!"
13 exit 0
```

运行结果如图 1-10 所示。



```
lemodel@ldocean148: /mnt/hgfs/windowsC/LinuxC/1/1.9
lemodel@ldocean148 1.91$ ./1.9
Guess the passwd!
Enter your answer:1111
Wrong answer, try again!
Enter your answer:1234
Wrong answer, try again!
Enter your answer:888888
The answer is right, congratulation!
lemodel@ldocean148 1.91$
```

图 1-10 until 语句

1.2 上机实验练习

一、Linux 命令上机实验

1. 使用下面的命令显示有关你的计算机系统信息：uname（显示操作系统的名称），uname-n（显示系统域名），uname-p（显示系统的 CPU 名称）。回答下列问题：

- (1) 你的操作系统名字是什么？
- (2) 你的计算机系统的域名是什么？
- (3) 你的计算机系统的 CPU 名字是什么？

2. 使用 whoami 命令找到用户名。然后使用 who-a 命令来看看你的用户名和同一系统其他用户的列表。

3. 用 pwd 显示你的主目录（home directory）名字，给出 pwd 显示的结果。

4. 在系统中，执行 cd professional/courses 命令，回答下列问题。

- (1) 你的主目录的绝对路径是什么？给出获得该绝对路径的命令及命令输出。
- (2) acm 目录的绝对路径是什么？
- (3) 给出 acm 目录的两个相对路径。
- (4) 执行 cd major/cs381/labs 命令。然后执行一个命令显示当前目录的绝对路径，给出这个会话过程。



(5) 给出三个不同的命令，返回到你的主目录。

5. 在你的主目录下建立如图 1-11 所示的目录树。Your Home Directory 表示你的主目录，不需要再建立。给出完成这项工作的所有会话。（会话是指你命令的输入和结果的输出，你提交的作业要包含这些内容。）

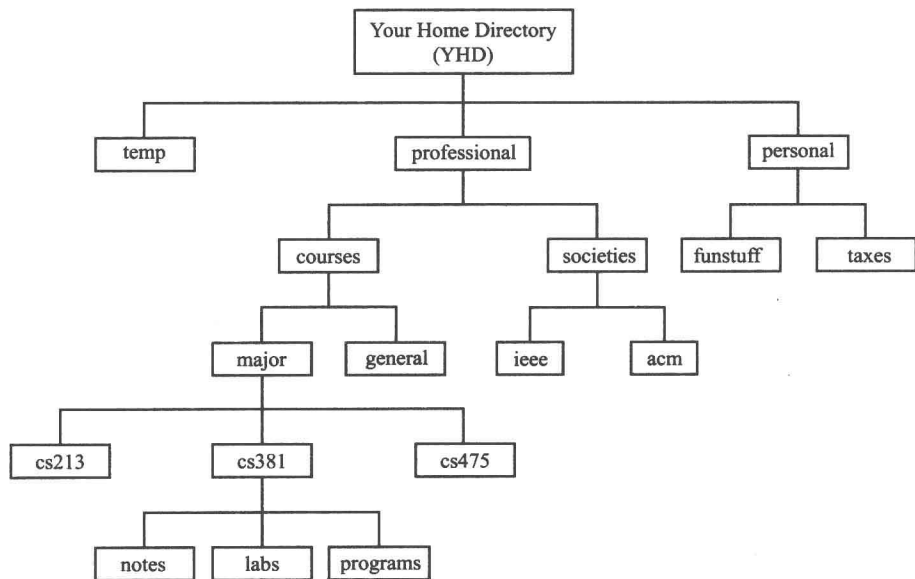


图 1-11 目录树

二、观察、思考并上机练习 shell 脚本

1. 编写一个名为 mul 的脚本程序，参数为一个大于 20 的正整数。先检查参数是否符合要求。如果不符合要求，请给出提示；如果符合要求，分别输出其与 1 到 10 的乘积。

```
#gedit mul.sh
#!/bin/bash
if [ $1 -gt 20 ]
then
    n=1
    m=1
    while [ $n -le 10 ]
    do
        m=$(expr $1 \* $n)
        echo "$n    $m"
        n=$(expr $n + 1)
    done
else
    echo "number is wrong"
fi
#bash mul.sh 34
```



2. 编写一个名为 move 的脚本程序，格式 `move <file1> <file2>`。如果 file1 不存在，给出提示；否则移动 file1 至 file2。

```
#gedit move.sh
#!/bin/bash
if test -f file1
then mv file1 file2
else
    echo "file1 is not exists"
fi
# bash move.sh file1 file2
```

3. 编写一个 shell 脚本，能够显示下面序列的前 25 个数字。0, 1, 1, 2, 3, 5, 8, 13..., 前两个数字之和为第三个数字，即著名的 Fibonacci 序列。

```
#gedit shell.sh
#!/bin/bash
n=0
echo "$n"
m=1
echo "$m"
t=1
a=2
while [ $a -lt 25 ]
do
    t=$(expr $n+$m)
    echo "$t"
    a=$(expr $a+1)
    n=$m
    m=$t
done
# bash shell.sh
```

4. 编写一个名为 square 的脚本程序，参数为一大于 10 的正整数。先检查参数是否符合要求。如果不符合要求，请给出提示；如果符合要求，输出从 1 到该正整数的平方值。

```
#gedit square.sh
#!/bin/bash
if [ $1 -gt 10 ]
then
    n=1
    m=1
    while [ $n -le $1 ]
    do
        m=$(expr $n \ * $n)
```



```
        echo "$n    $m"
        n=$(expr $n+1)
    done
else
    echo "number is wrong"
fi
bash square.sh 45
```

三、实验报告

1. 源程序。
2. 错误原因及其修改记录。
3. 实验结果记录。
4. 实验体会。