**中国矿业大学计算机科学与技术学院**

**2024-2025(2)《Linux操作系统》课程作业(本科)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 专业班级: | | 计算机科学与技术2022-02班 | | | 姓名: | 杨晓琦 | | | 学号: | 08222213 | |  |
|  | | | | | | | | | | | |  |
| **序号** | **报告题目** | **基础理论掌握程度** | **综合知**  **识应用**  **能力** | **报告**  **内容** | **报告**  **格式** | **完成**  **状况** | **工作量** | **学习、**  **工作**  **态度** | **抄袭**  **现象** | **其它** | **综合**  **成绩** | **评阅日期** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| 任课教师： | |  | |  |  |  | |  | |  |  |  |

目录

[1 Linux平台下的基本命令 1](#_Toc194162921)

[1.1 查看版本 1](#_Toc194162922)

[1.1.1 uname命令 1](#_Toc194162923)

[1.1.2 /etc/issue命令 1](#_Toc194162924)

[1.2 帮助 1](#_Toc194162925)

[1.2.1 --help命令 1](#_Toc194162926)

[1.2.2 man命令 2](#_Toc194162927)

[1.2.3 info命令 2](#_Toc194162928)

[1.3 文件和目录 3](#_Toc194162929)

[1.3.1 ls命令 3](#_Toc194162930)

[1.3.2 pwd命令 3](#_Toc194162931)

[1.3.3 cd命令 3](#_Toc194162932)

[1.3.4 touch命令 4](#_Toc194162933)

[1.3.5 mkdir命令 4](#_Toc194162934)

[1.3.6 rmdir命令 4](#_Toc194162935)

[1.3.7 cp命令 5](#_Toc194162936)

[1.3.8 mv命令 5](#_Toc194162937)

[1.3.9 whereis命令 6](#_Toc194162938)

[1.3.10 which命令 6](#_Toc194162939)

[1.3.11 find命令 6](#_Toc194162940)

[1.3.12 grep命令 7](#_Toc194162941)

[1.3.13 cat与tac命令 7](#_Toc194162942)

[1.3.14 head与tail命令 7](#_Toc194162943)

[1.3.15 sort命令 8](#_Toc194162944)

[1.3.16 uniq命令 8](#_Toc194162945)

[1.3.17 cut命令 8](#_Toc194162946)

[1.3.18 comm命令 8](#_Toc194162947)

[1.4 系统信息 9](#_Toc194162948)

[1.4.1 hostname命令 9](#_Toc194162949)

[1.4.2 free命令 9](#_Toc194162950)

[1.5 日期和时间 10](#_Toc194162951)

[1.5.1 cal命令 10](#_Toc194162952)

[1.5.2 date命令 11](#_Toc194162953)

[1.6 信息传输 11](#_Toc194162954)

[1.6.1 echo命令 11](#_Toc194162955)

[1.6.2 wall命令 11](#_Toc194162956)

[1.6.3 write命令 11](#_Toc194162957)

[1.6.4 w命令 12](#_Toc194162958)

[1.6.5 who命令 12](#_Toc194162959)

[1.7 磁盘与文件系统 12](#_Toc194162960)

[1.7.1 fdisk命令 12](#_Toc194162961)

[1.7.2 mkfs命令 13](#_Toc194162962)

[1.7.3 df命令 13](#_Toc194162963)

[1.7.4 du命令 14](#_Toc194162964)

[1.7.5 lsblk命令 14](#_Toc194162965)

[1.7.6 mount命令 15](#_Toc194162966)

[1.7.7 fsck命令 15](#_Toc194162967)

[1.8 用户与组 15](#_Toc194162968)

[1.8.1 /etc/passwd 15](#_Toc194162969)

[1.8.2 /etc/shadow 16](#_Toc194162970)

[1.8.3 /etc/group 16](#_Toc194162971)

[1.8.4 /etc/gshadow 16](#_Toc194162972)

[1.9 管道符组合命令 17](#_Toc194162973)

[1.9.1 基本用法 17](#_Toc194162974)

[1.9.2 grep-文本搜索 17](#_Toc194162975)

[1.9.3 wc-统计 17](#_Toc194162976)

[1.9.4 sort-排序 18](#_Toc194162977)

[1.9.5 cut-提取特定字段 18](#_Toc194162978)

[1.9.6 xargs-将标准输入转换成命令行参数 18](#_Toc194162979)

[2 Linux Shell编程 19](#_Toc194162980)

[2.1 执行shell程序 19](#_Toc194162981)

[2.1.1 创建shell脚本 19](#_Toc194162982)

[2.1.2 编写shell脚本内容 20](#_Toc194162983)

[2.1.3 运行shell脚本 20](#_Toc194162984)

[2.2 if结构和条件分支 20](#_Toc194162985)

[2.2.1 四则简易计算器 20](#_Toc194162986)

[2.2.2 计算BMI并提供健康指南 21](#_Toc194162987)

[2.3 for、while和until循环的结合 23](#_Toc194162988)

[2.4 组合使用的综合案例(任务管理系统) 24](#_Toc194162989)

[2.5 shell程序的调试 27](#_Toc194162990)

[2.5.1 通过echo方式 27](#_Toc194162991)

[2.5.2 通过选项的方式 27](#_Toc194162992)

[2.5.3 通过test方式 28](#_Toc194162993)

[2.5.4 通过trap的方式 28](#_Toc194162994)

[2.5.5 使用调试工具-bashdb 29](#_Toc194162995)

# 1 Linux平台下的基本命令

要求：

⑴ 尽可能的多测试 LINUX 平台的各种命令，实验的命令必须有运行结果截图，命令不少于20个；⑵ 学习使用管道符组合使用命令；

⑶ 撰写内容必须符合《计算机学院考查类课程报告撰写规范》规范要求；

⑷ 提交电子版综合报告与源代码，连同两次作业放在一个压缩包里；

⑸ 提交网站：219.219.61.252（课程名称：**Linux操作系统2024-2025(2)**）；

# 1.1 查看版本

1.1.1 uname命令

uname [选项]...

表uname选项参数

|  |  |
| --- | --- |
| -r | 输出内核版本号 |
| -m | 输出主机的硬件架构名称 |
| -a | 显示全部信息 |

uname命令用来显示计算机及操作系统相关信息。如下图所示，对uname命令不同选项进行测试。

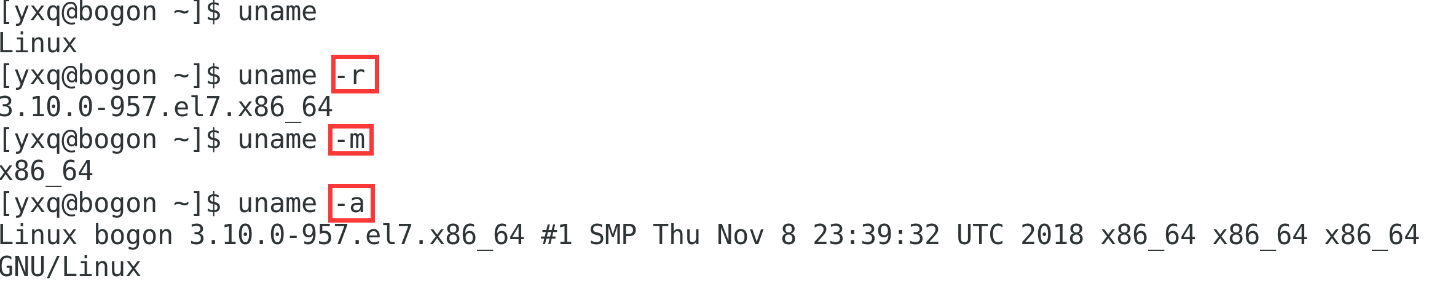


图1-1-1 uname

1.1.2 /etc/issue命令

该命令用于查看操作系统的发行版本号，\r，为当前运行的内核版本号，即3.10.0-957.el7.x86\_64

\m：为当前系统的机器类型（架构），即x86\_64。

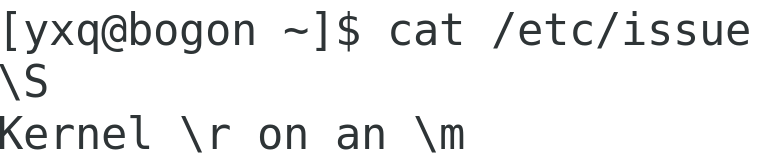


图1-1-2 cat /etc/issue

# 1.2 帮助

1.2.1 --help命令

该命令用于显示简要命令的格式、选项及其功能。如下图所示, 实现了查询ls命令的基本的使用方法和参数的介绍。

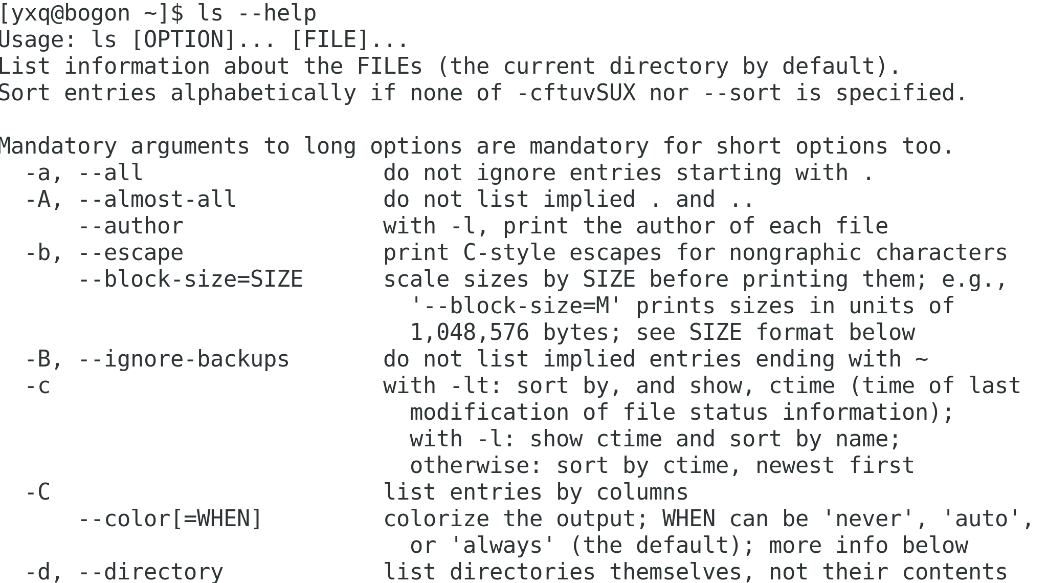
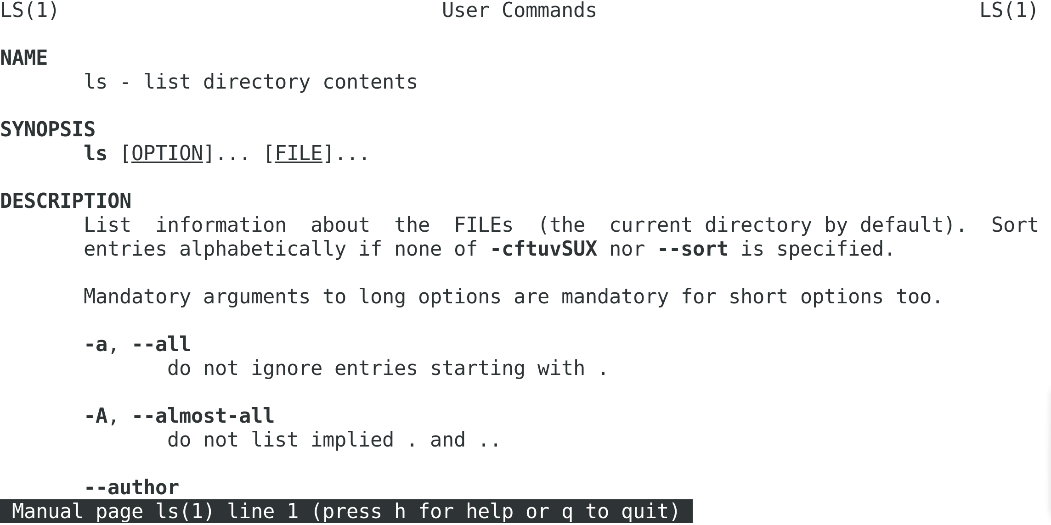


图1-2-1 ls –help

1.2.2 man命令

man [选项] 命令

它包含了Linux中全部命令的手册。如下图所示，显示 ls 命令的手册页。



1-2-2 man ls

1.2.3 info命令

info 命令

如下图所示，info ls 是用于访问 ls 命令的详细信息页面的命令。

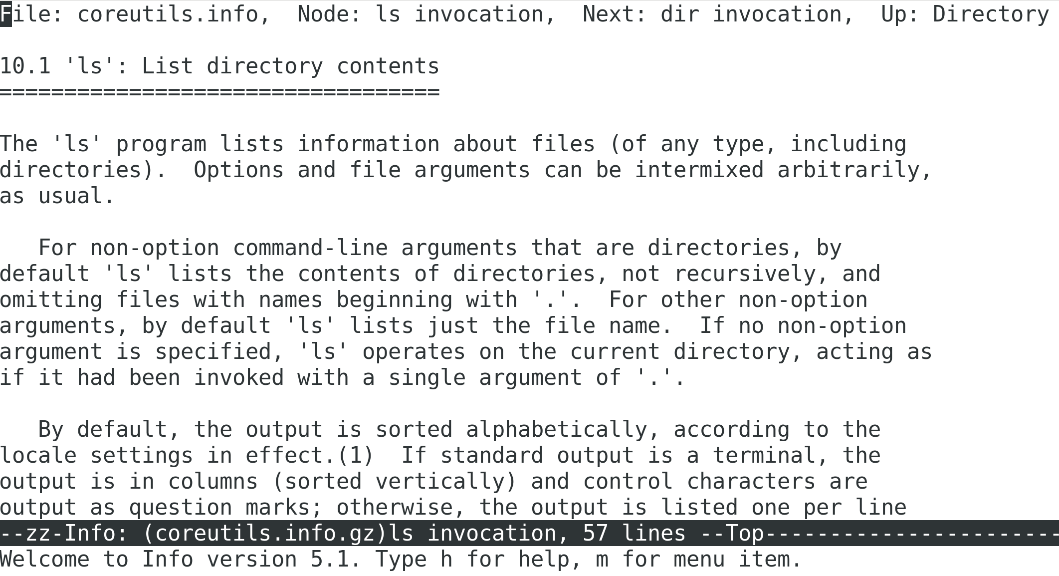


图1-2-3 info ls

# 1.3 文件和目录

1.3.1 ls命令

ls [选项] [目标目录]

ls命令用来显示文件信息。如下图所示，显示了当前目录即home目录以及子目录和文件，不包括隐藏文件。



图1-3-1 ls

1.3.2 pwd命令

pwd [选项]

pwd命令用来查询当前目录。如下图所示，显示$PWD变量值，它定义了当前的工作目录。

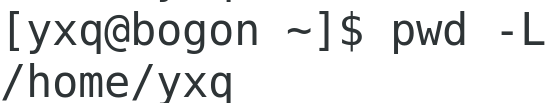


图1-3-2 pwd -L

1.3.3 cd命令

cd [选项] [目标目录]

cd命令用来切换工作目录。如下，实现了切换不同的目标目录。

表 cd命令切换不同目录

|  |  |
| --- | --- |
| cd(无) | 切换到当前用户的家目录 |
| cd/ | 切换到根目录 |
| cd- | 切换到上一次的目录 |
| cd. | 切换到当前所在目录 |
| cd.. | 切换到当前目标位置的上一层目录 |

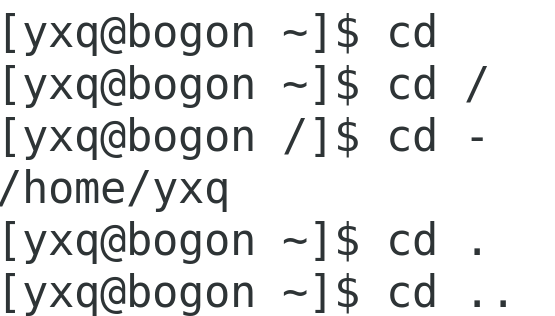


图1-3-3 cd

1.3.4 touch命令

touch [选项] 文件

touch命令将每个文件的访问时间和修改时间更改为当前时间，若文件不存在，则创建一个空文件。

如下图所示，yyxxqq文件不存在，则创建一个名为 yyxxqq 的空文件。



图1-3-4 touch

1.3.5 mkdir命令

mkdir [选项] [目录]

mkdir命令用于创建目录。如下图所示，在当前目录下创建 yang 空目录。

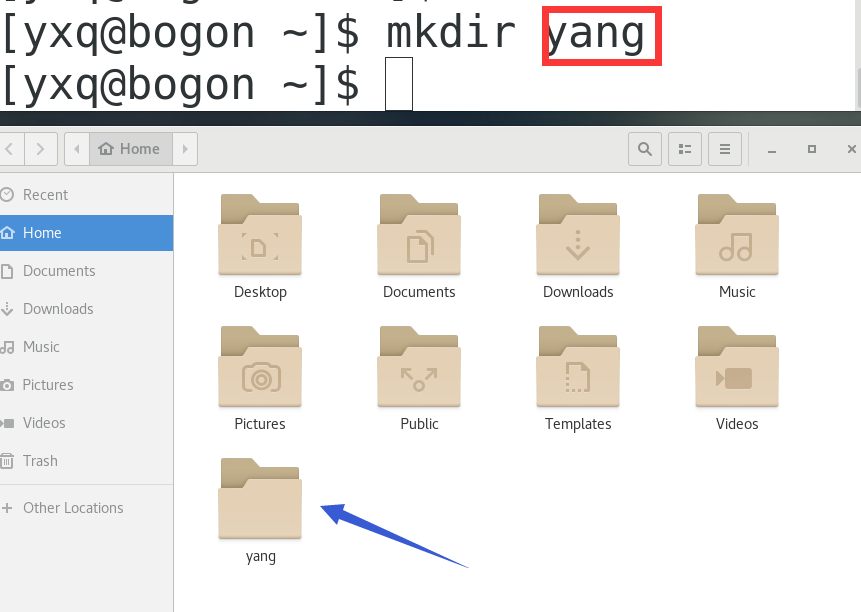


图1-3-5 mkdir

1.3.6 rmdir命令

rmdir [选项] [目录]

rmdir命令用于删除空目录。如下图所示，删除了 yang 的空目录。

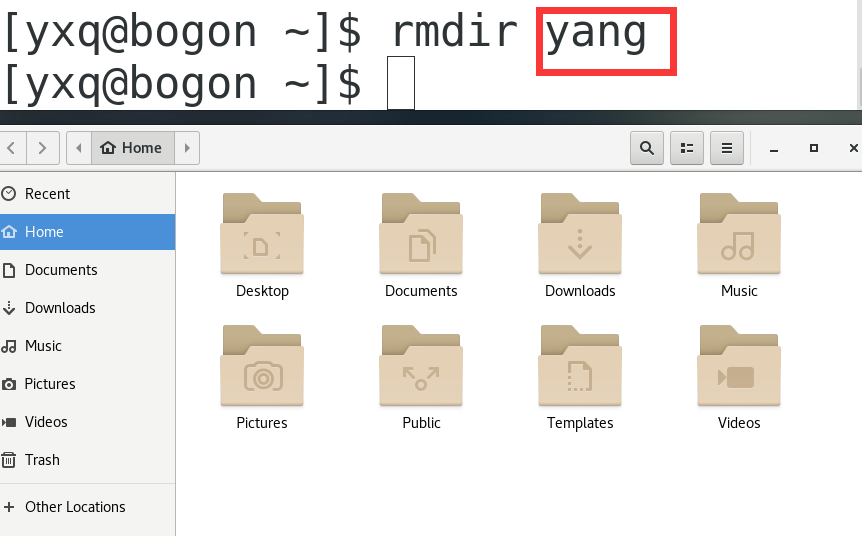
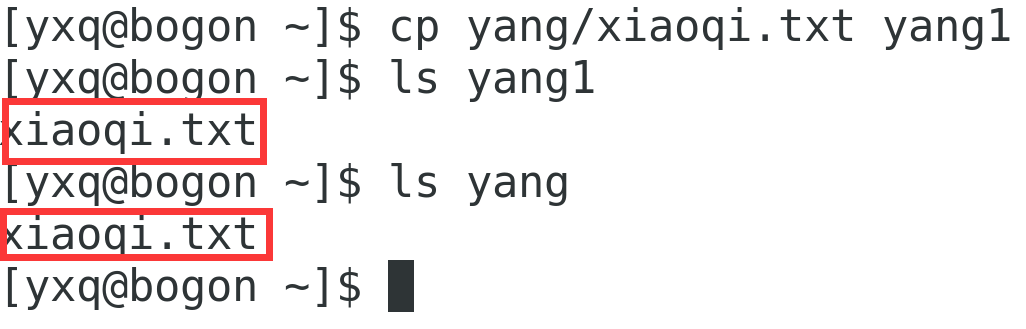


图1-3-6 rmdir

1.3.7 cp命令

cp [选项] [源文件或目录] [目标文件或目录]

cp命令用于复制文件或目录，不加目标文件名是原名复制，否则是改名复制。如下图所示，将当前目录下yang/xiaoqi.txt 文件复制到yang1目录下。



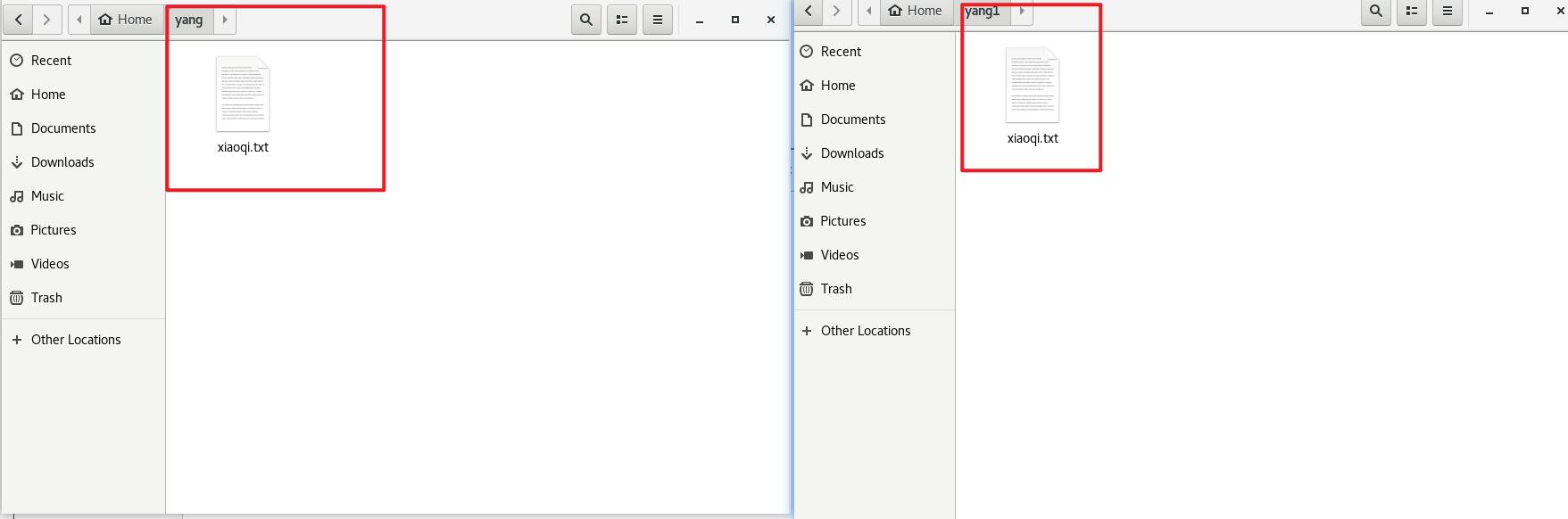
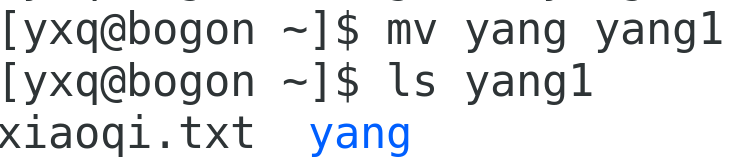


图1-3-7 cp

1.3.8 mv命令

mv [选项] [源文件或目录] [目标文件或目录]

mv命令用于剪切或改名。源文件和目标文件不在同一目录下是剪切，在同一目录下是改名。如下图所示，实现了先将yang目录剪切到yang1 目录下，随后再将同yang1/yang/xiaoqi.txt文件改名为xiaoqi1.txt。



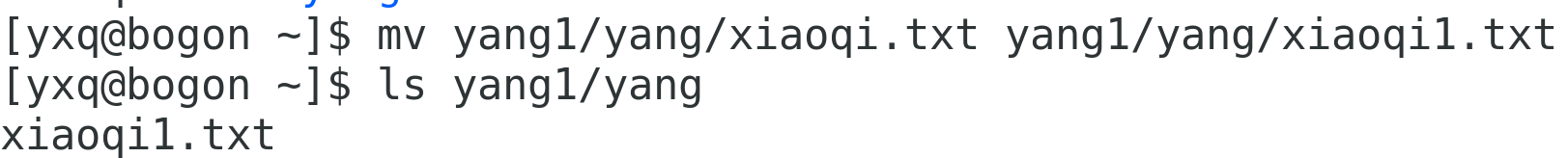


图1-3-8 mv

1.3.9 whereis命令

whereis [选项] [命令名]

表 where选项参数

|  |  |
| --- | --- |
| -b | 只搜索二进制文件 |
| -m | 只搜索man手册和info |
| -s | 只搜索源代码 |

whereis命令用来搜索二进制命令文件、源码、man手册、info手册所在路径。如下图所示，实现了whereis不同选项进行文件的搜索。

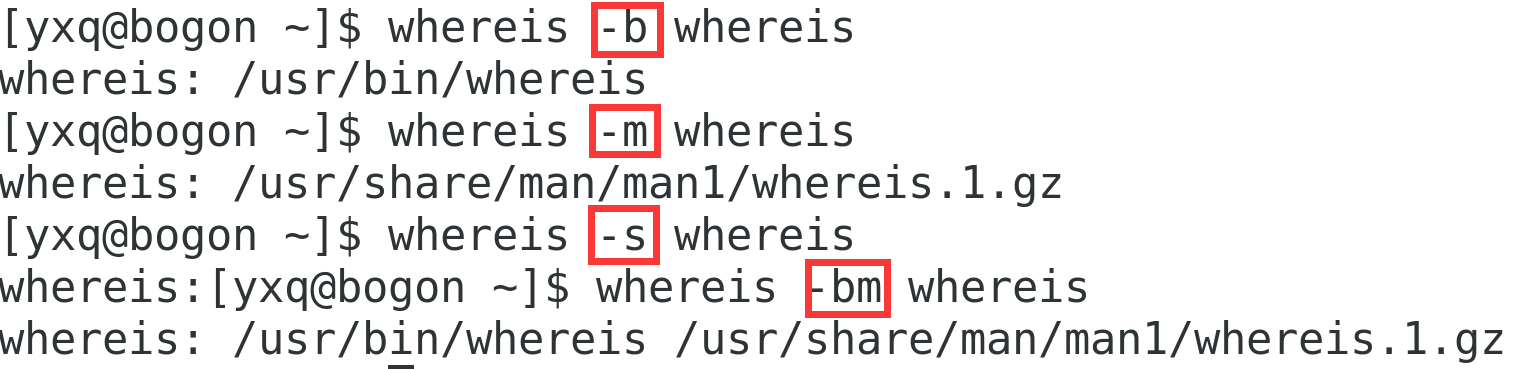


图1-3-9 whereis

1.3.10 which命令

which 命令名

which命令用来在环境变量$PATH设置的目录里查找符合条件的命令文件。如下图所示，将which与whereis进行比较，which可以查到命令的别名，但看不到帮助文档，而whereis可以查找外部命令及其帮助文件。



图1-3-10 which与whereis

1.3.11 find命令

find [路径] [表达式]

find命令用来在指定路径下查找文件(默认为当前目录)，如下图所示，查找/home目录下的所有文件。

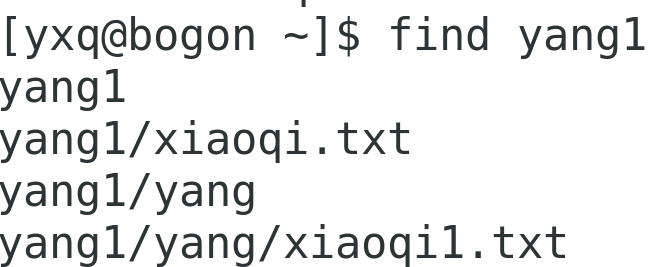


图1-3-11 find

1.3.12 grep命令

grep [选项] 模式字符串 文件名 [...]

grep命令用来搜索文件中含有指定模式字符串的行并显示该行。如下图所示，搜索 xiaoqi.txt文件中的08222213 字符串，并返回该行。



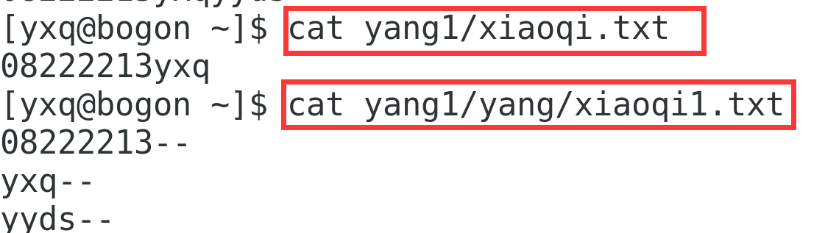
图1-3-12 grep

1.3.13 cat与tac命令

cat [选项]...[文件]...

Tac[文件]...

cat命令用来连接一个或多个文件并显示输出，而tac命令按行倒序显示文本内容。如下图所示，先cat命令正序显示xiaoqi.txt和xiaoqi1.txt文件，再用tac倒序显示。





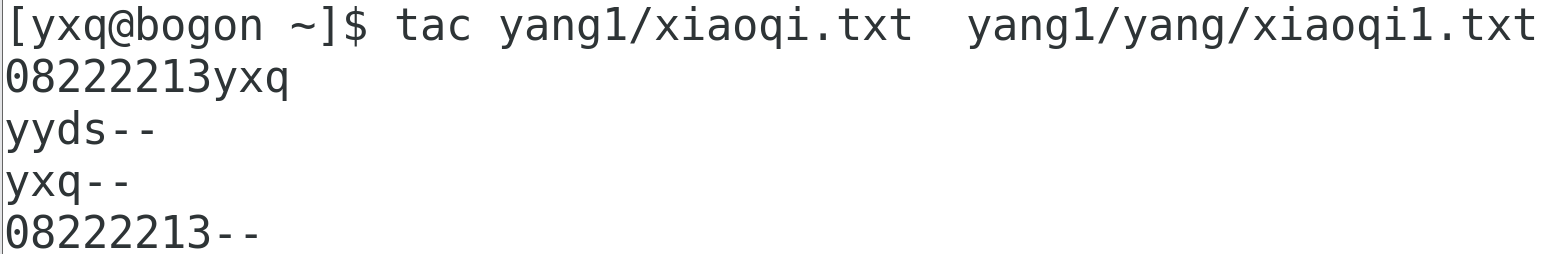


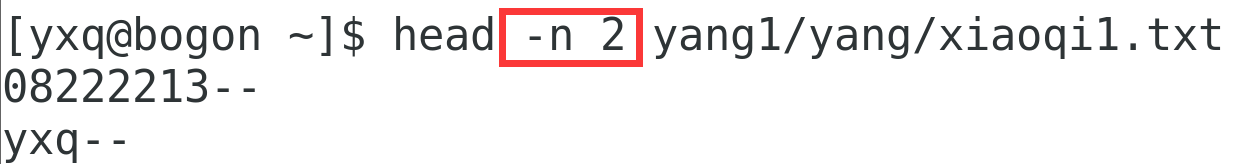
图1-3-13 cat与tac

1.3.14 head与tail命令

head [选项]...[文件]...

tail [选项]...[文件]...

head命令默认显示前10行，tail命令默认显示尾部10行。如下图所示，用head命令显示txt文件的前两行，用tail命令显示txt文件的最后两行。



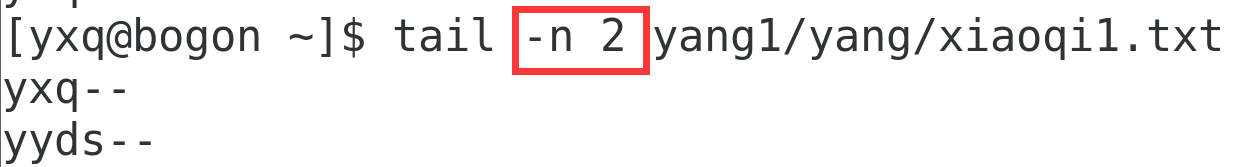


图1-3-14 head与tail

1.3.15 sort命令

sort [选项]...[文件]...

sort命令用来连接所有文件，然后排序，并输出结果。如下图所示，对txt文件每行进行排序。

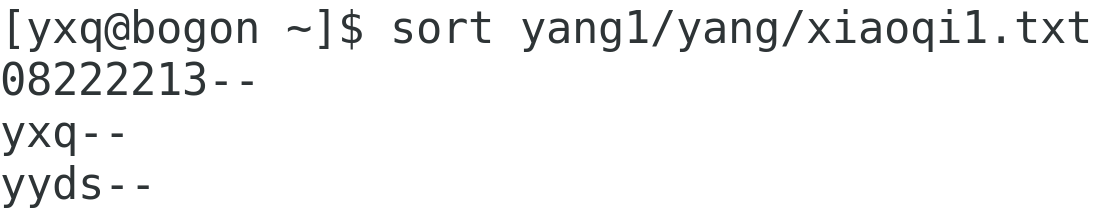


图1-3-15 sort

1.3.16 uniq命令

uniq [选项]...[输入文件[输出文件]]

uniq过滤输入文件中内容相同的相邻行，并写到输出文件中；不带选项时，内容相同的行仅输出一次。如下图所示，删除txt文件中重复行 08222213yxq 和 yyds 字段。

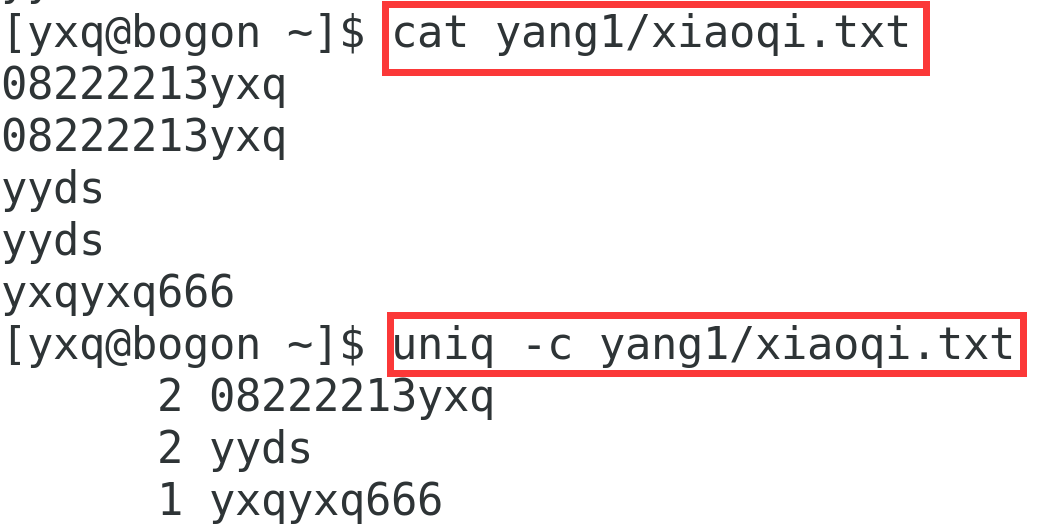


图1-3-16 uniq

1.3.17 cut命令

cut [选项]...[文件]...

cut命令将文件每行的指定部分打印到标准输出。如下图所示，打印txt文件每行的前四个字符。

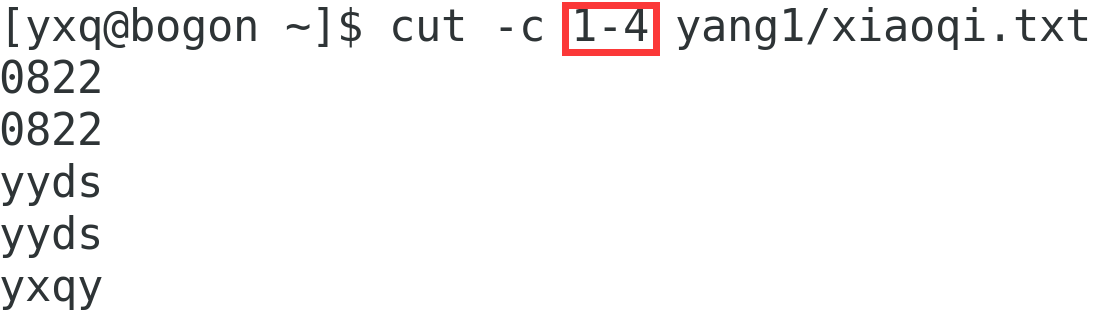


图 1-3-17 cut

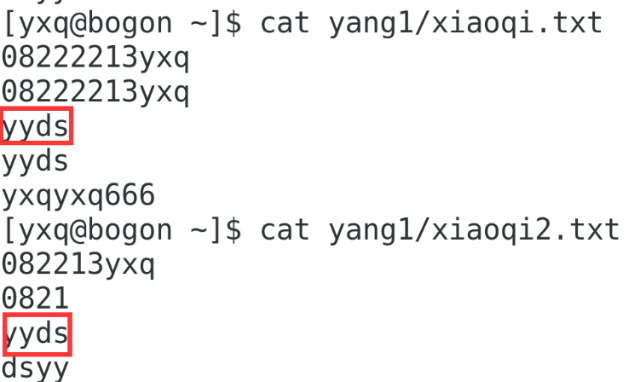
1.3.18 comm命令

comm [选项]... 文件1 文件2

表 comm选项参数

|  |  |
| --- | --- |
| -1 | 不输出文件1特有的行 |
| -2 | 不输出文件2特有的行 |
| -3 | 不输出两个文件共有的行 |

comm命令用来逐行比较已排序文本文件。如下图所示，-12选项用来输出两个文件的共有的行。



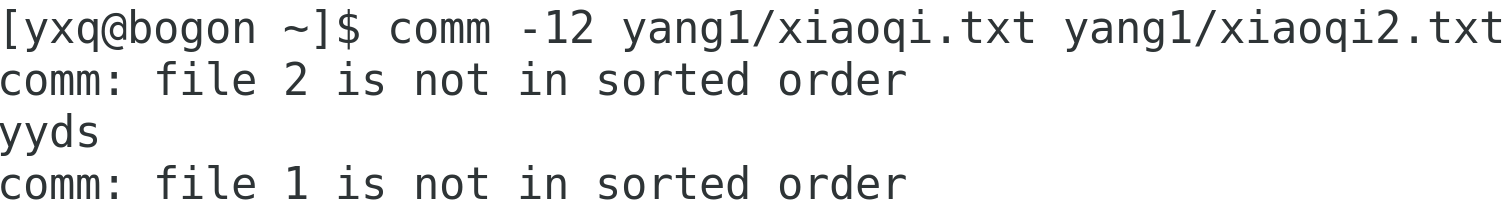


图1-3-18 comm

# 1.4 系统信息

1.4.1 hostname命令

hostname [-a|-A|-d|-f|-i|-l|-s|-y]

表 comm选项参数

|  |  |
| --- | --- |
| -s | 显示短主机名 |
| -i | 显示ip地址 |
| -f | 显示长主机名 |

hostname用来显示主机名，如下图所示，使用-i 选项显示主机ip地址。

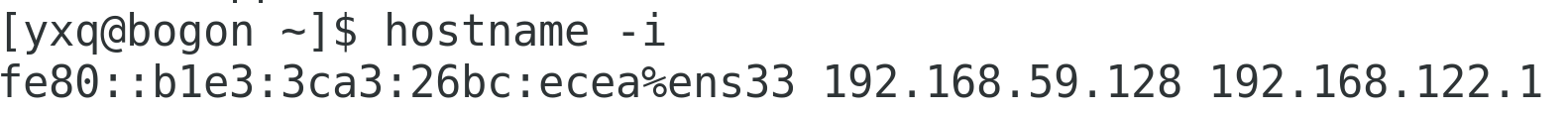


图1-4-1 hostname

1.4.2 free命令

free [选项]

表 free选项参数

|  |  |
| --- | --- |
| -m | 以MB显示输出 |
| -t | 显示内存加上swap总的容量 |
| -b | 以字节显示输出 |

free命令用来查看内存信息。如下图所示，对不同选项进行测试。

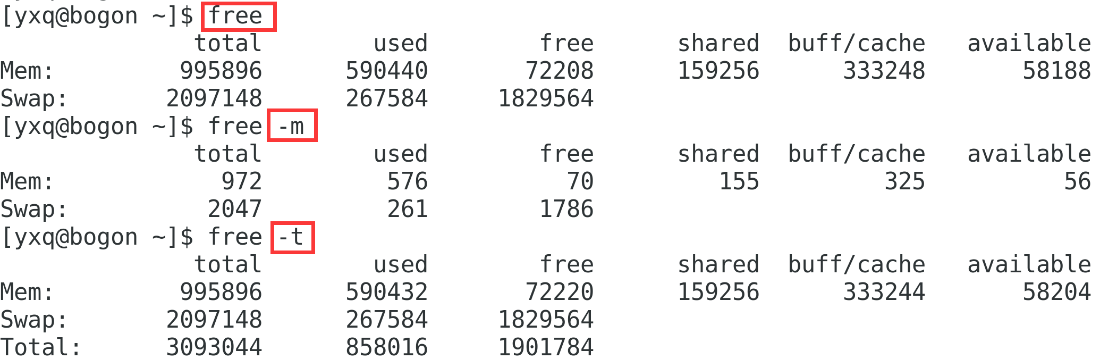


图1-4-2 free

# 1.5 日期和时间

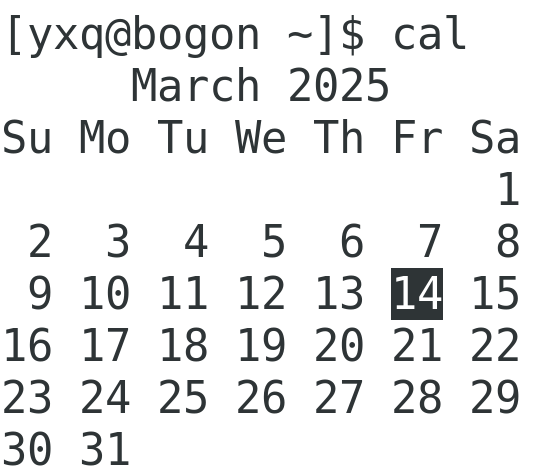
1.5.1 cal命令

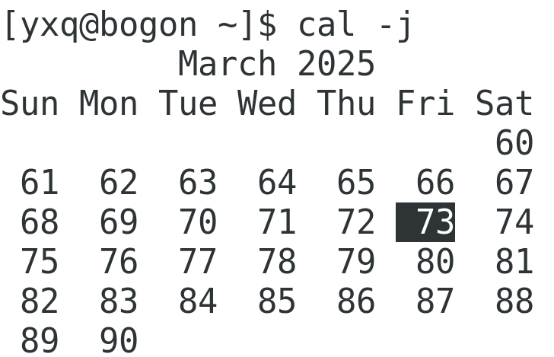
cal [选项] [月] [年]

表 cal选项参数

|  |  |
| --- | --- |
| -j | 以“朱利安日历格式”显示当前月份 |
| -1 | 默认，只显示当前月份的日历 |
| -3 | 显示系统的前一个月、当前月和下一个月的日历 |

cal命令用来显示日历。如下图所示，对cal不同选项进行测试。





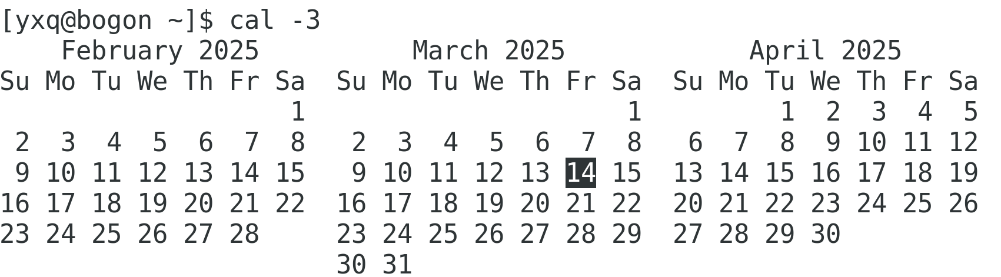
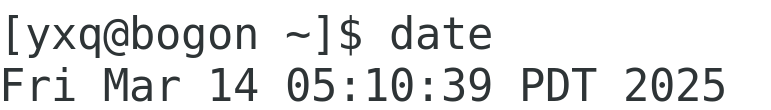


图1-5-1 cal

1.5.2 date命令

date [选项]...[+格式]

date命令用来设置和显示日期和时间date。如下图所示，显示系统当前日期和时间，再显示txt文件最后的修改时间。



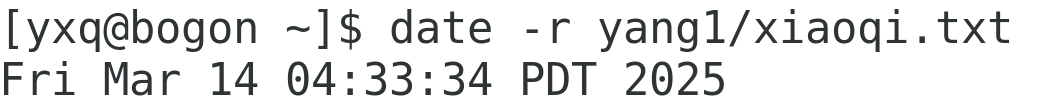


图1-5-2 date

# 1.6 信息传输

1.6.1 echo命令

echo [-neE] [参数...]

echo命令用来在屏幕上显示文字。如下图所示，如用-n参数进行不换行输出。

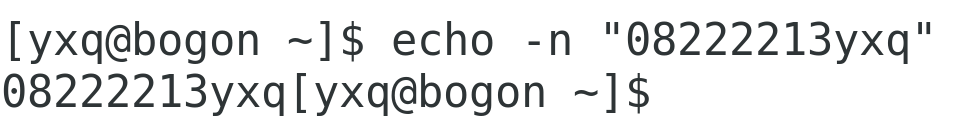


图1-6-1 echo

1.6.2 wall命令

wall [选项] [<文件>|<消息>]

wall命令用来向全部已登录用户发送消息wall。如下图所示，发送“yxq”进行测试。

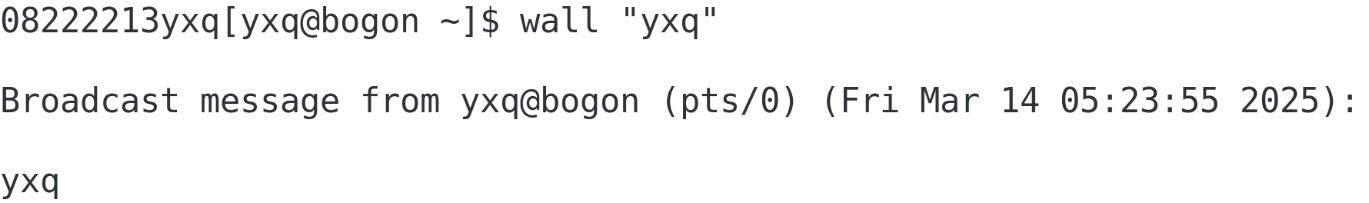


图1-6-2 wall

1.6.3 write命令

write[用户][终端名]

write命令用来向用户发送消息。如下图所示，向名为yxq的用户发送消息。

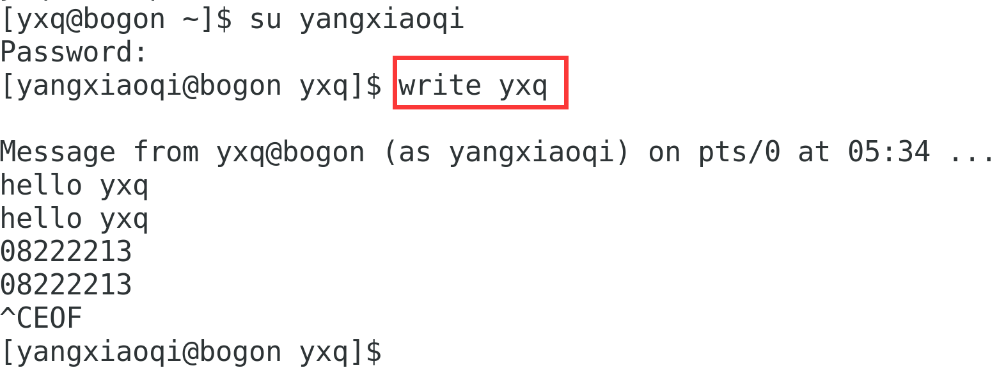


图1-6-3 write

1.6.4 w命令

w [选项]

w命令用来查看登录用户信息。如下图所示，查看root用户信息，​​USER​​ 列给出了用户名，然后是终端号、登录日期时间、空闲时间、CPU 使用率，以及用户正在执行的进程。

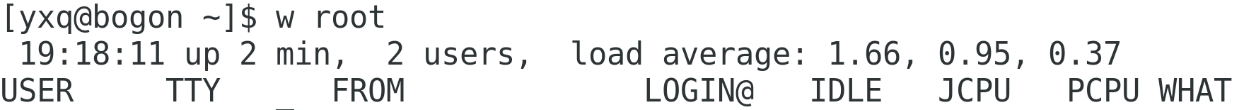


图1-6-4 w

1.6.5 who命令

who [选项]

who命令用来显示当前登入系统的用户。如下图所示，当前登录用户为yxq。

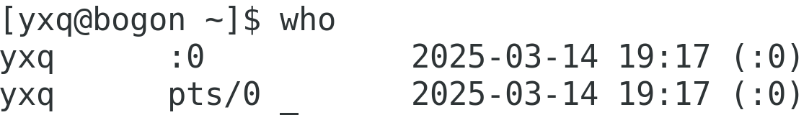


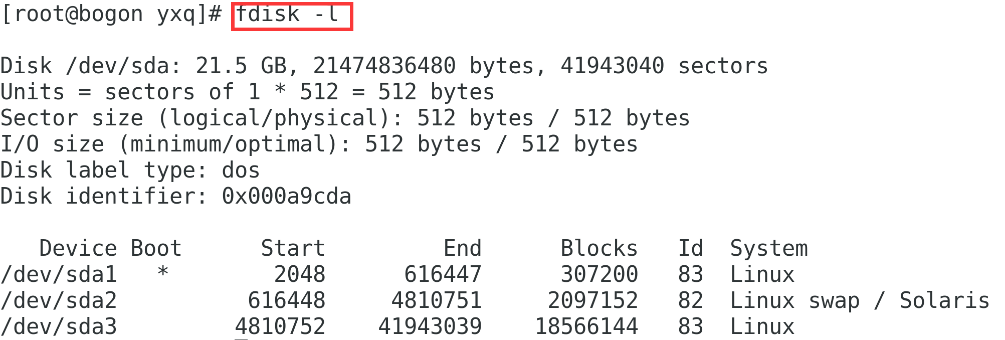
图1-6-5 who

# 1.7 磁盘与文件系统

1.7.1 fdisk命令

fdisk [选项]<磁盘>

fdisk命令是交互式的操作命令，用来显示分区。如下图所示，先使用fdisk /dev/sdc建立分区，再输入n新添加分区，接着输入e增加扩展分区，最后输入p显示所有分区。



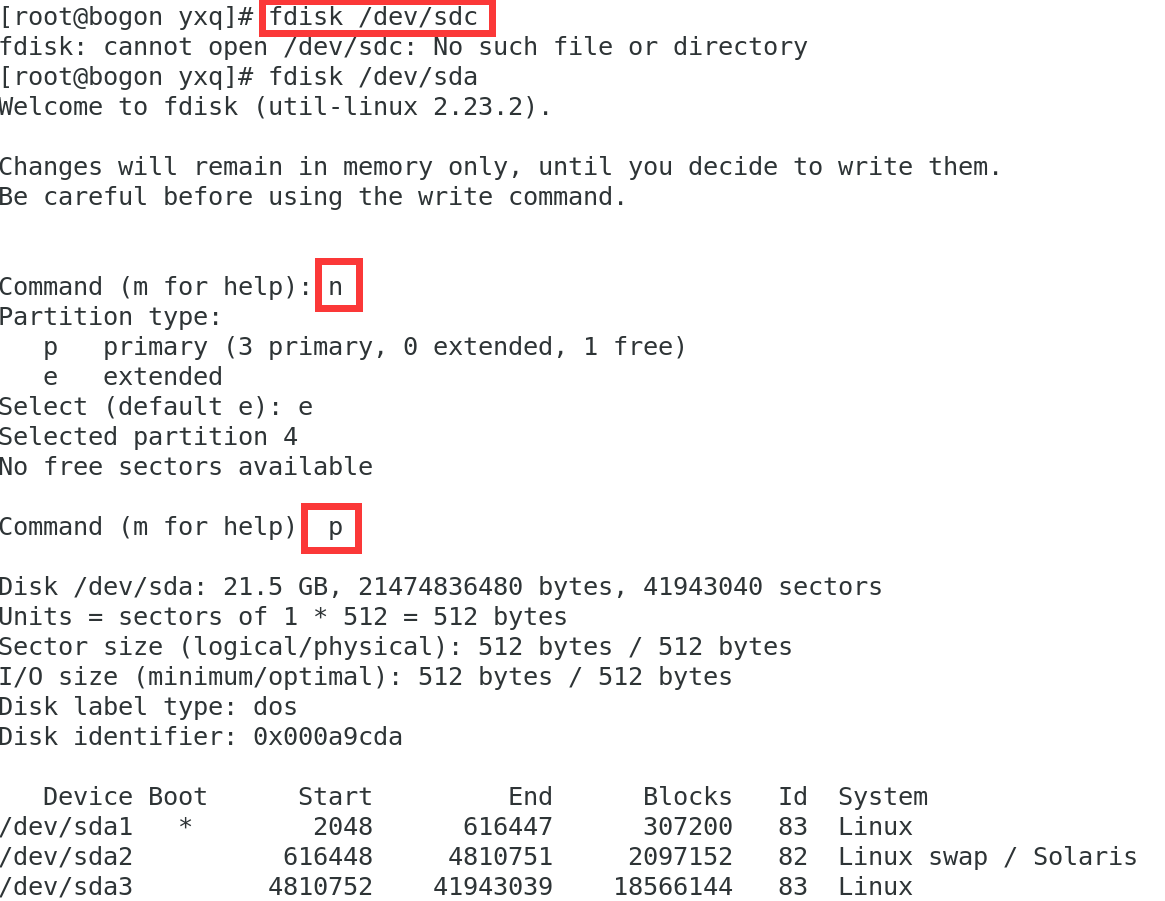


图1-7-1 fdisk

1.7.2 mkfs命令

mkfs [选项]<设备>[<大小>]

mkfs命令用来进行分区格式化。如下图所示，格式化/dev/sda1分区，创建xfs文件系统。

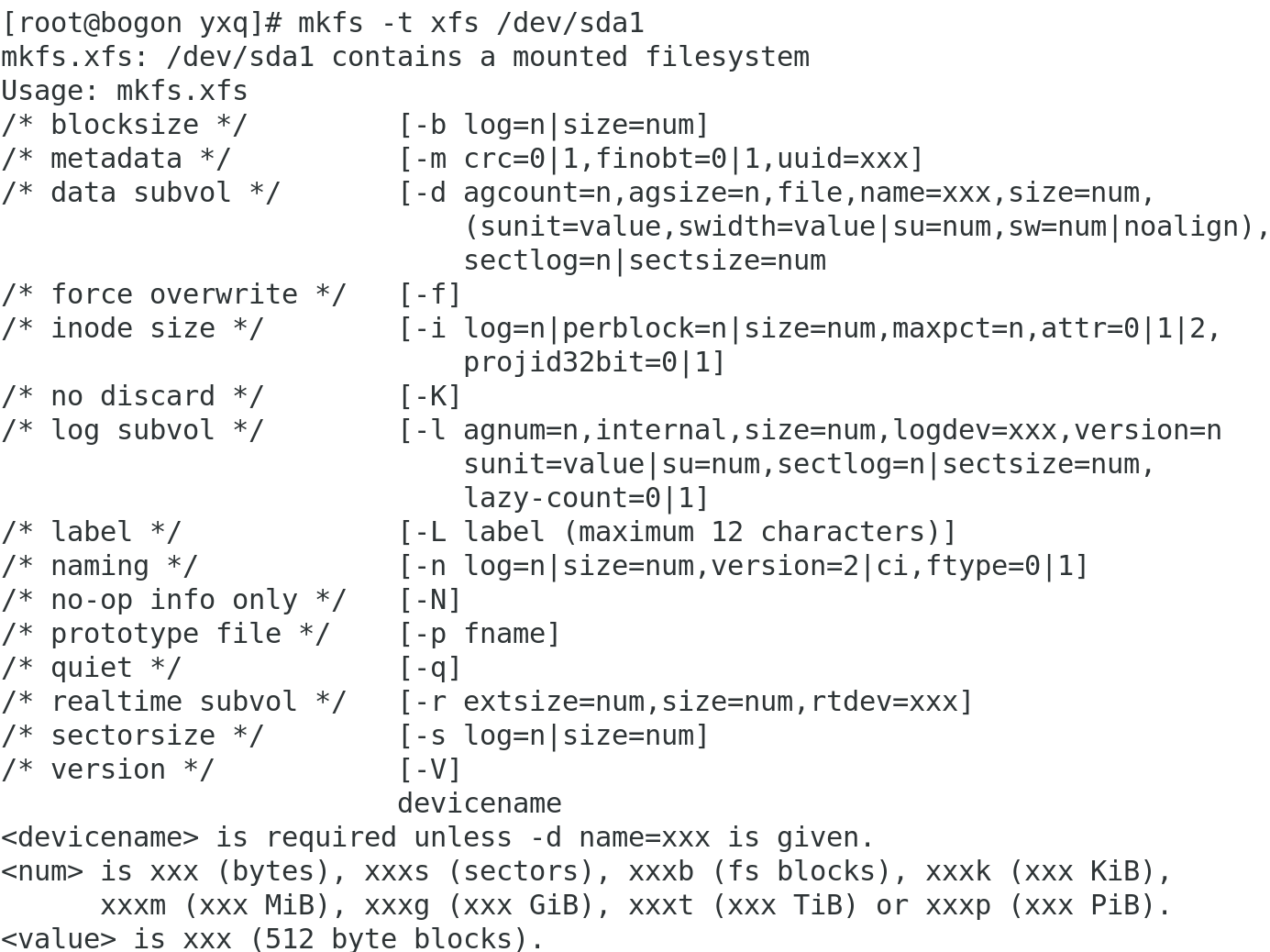


图1-7-2 mkfs

1.7.3 df命令

df [选项]...[文件|目录]...

df命令用来查看文件系统及其使用情况。如下图所示，显示每个文件所在的文件系统的信息。

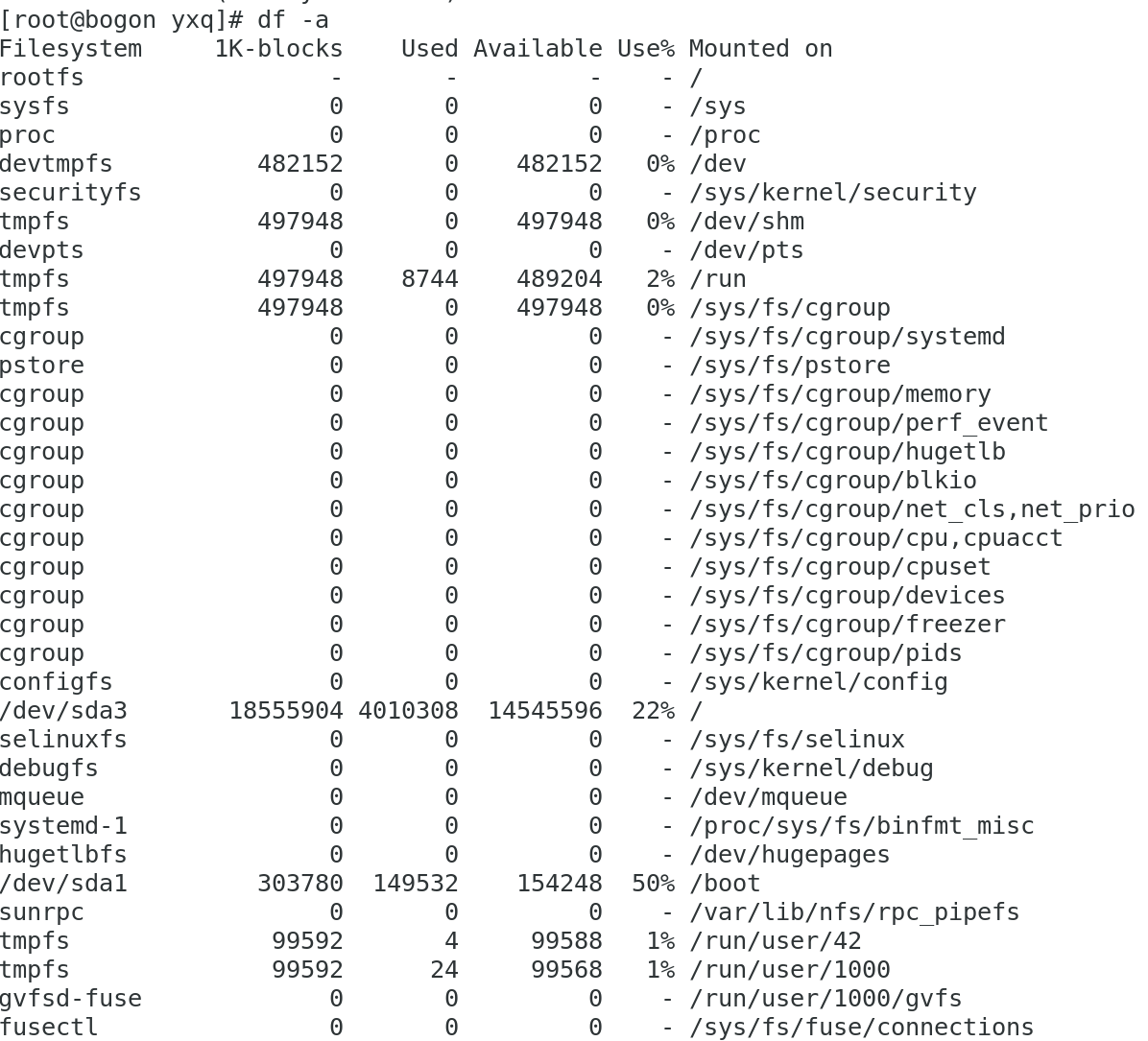


图1-7-3 df

1.7.4 du命令

du命令查看文件或目录存储空间用量。如下图所示，使用-h以1024进制的单位显示大小。

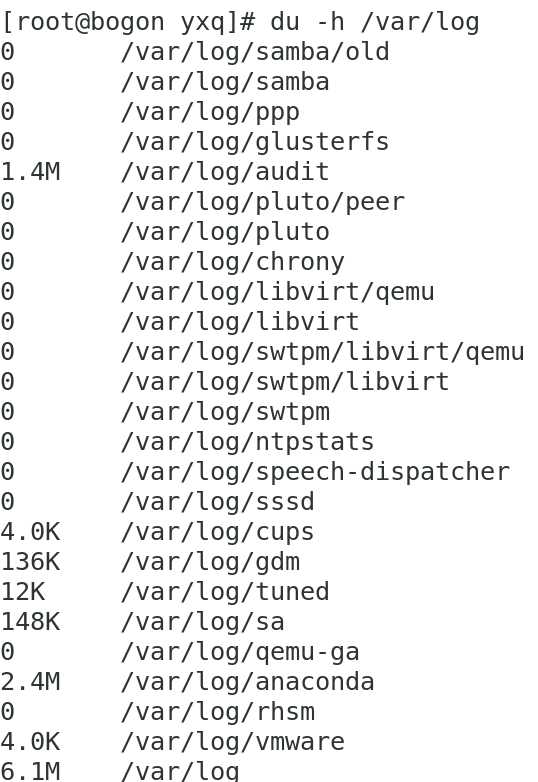


图1-7-4 du

1.7.5 lsblk命令

lsblk [选项][<设备名>...]

lsblk命令可以查看块设备信息命令。如下图所示，显示了所有块设备信息。

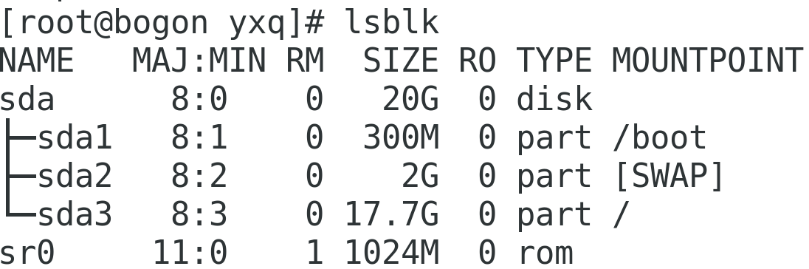


图1-7-5 lsblk

1.7.6 mount命令

mount[-t 文件系统][-o 特殊选项] 设备文件名 挂载点

mount该命令用于挂载文件系统。如下图所示，查询系统中已经挂载的设备。



图1-7-6 mount

1.7.7 fsck命令

fsck [选项][<文件系统>...]

fsck命令可以进行文件系统的修复。文件系统并不总是处于最佳状态，如果突然关闭系统，数据可能会损坏，fsck命令用于检查文件系统的一致性，并尝试修复它。

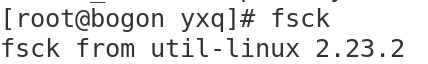


图1-7-7 fsck

# 1.8 用户与组

1.8.1 /etc/passwd

用户和属性信息存放的文件及位置为/etc/passwd。如下图所示，使用cat命令读取/etc/passwd文件，可以查看用户及其属性信息的具体内容。

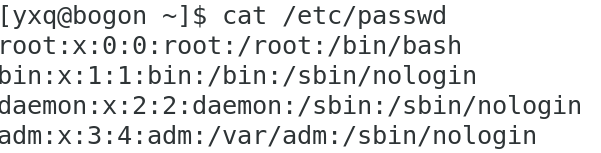


图1-8-1 /etc/passwd

1.8.2 /etc/shadow

用户密码和相关属性存放的文件及位置为/etc/shadow。如下图所示，使用cat命令读取/etc/shadow文件，可以查看用户密码及其相关属性的具体内容。

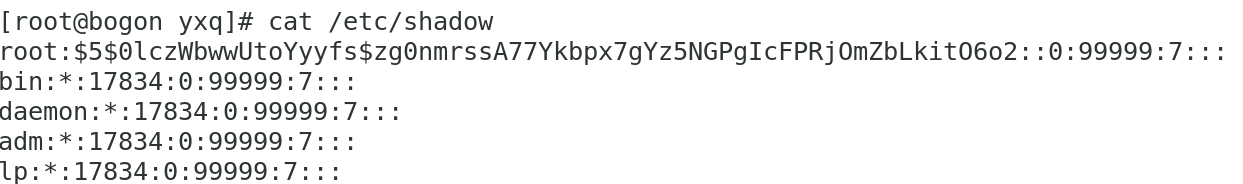


图1-8-2 /etc/shadow

1.8.3 /etc/group

组和属性关系存放的文件及位置为/etc/group。如下图所示，使用cat命令读取/etc/group文件，可以查看组及其属性信息的具体内容。

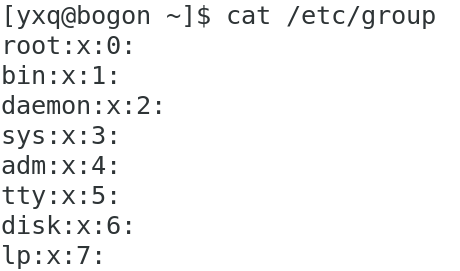


图1-8-3 /etc/group

1.8.4 /etc/gshadow

组密码和相关属性存放的文件及位置为/etc/gshadow。如下图所示，使用cat命令读取/etc/gshadow文件，可以查看组密码及其相关属性信息的具体内容。



图1-8-4 /etc/gshadow

# 1.9 管道符组合命令

1.9.1 基本用法

command1 | command2 | command3 ...

command1 的输出会传递给 command2 作为输入，command2 的输出会继续传递给 command3，以此类推，每个命令都只会处理它接收到的数据流。

1.9.2 grep-文本搜索

cat /etc/passwd | grep root

使用此命令实现在 /etc/passwd 文件中查找包含 root 的行。

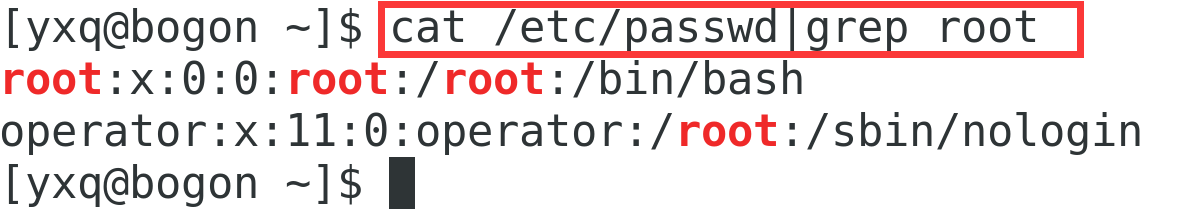


图1-9-2 在文件中查找包含root关键字

1.9.3 wc-统计

使用此命令实现统计 /etc/passwd 文件中包含 root 的行数。

cat /etc/passwd | grep root | wc -l

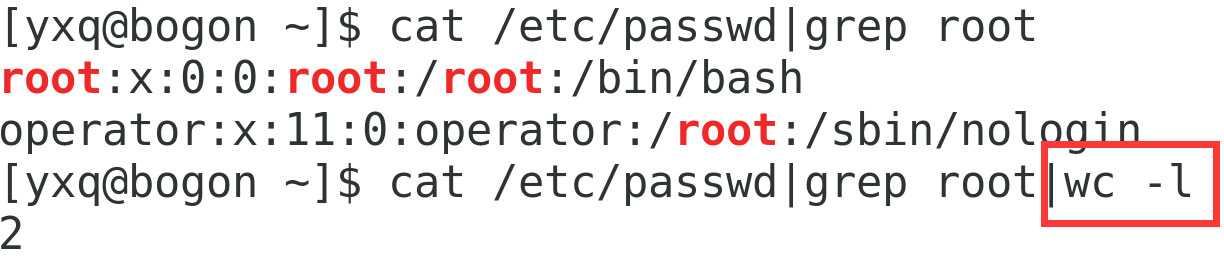


图1-9-3 统计文件中特定行的行数

1.9.4 sort-排序

使用此命令列出所有正在运行的进程，筛选出 bash 相关的进程，并按内存使用量排序。

ps aux | grep bash | sort -k 4 -nr

* ps aux：列出所有正在运行的进程
* grep bash：查看bash进程
* sort -k 4 -nr：根据第4列（内存使用）降序排序

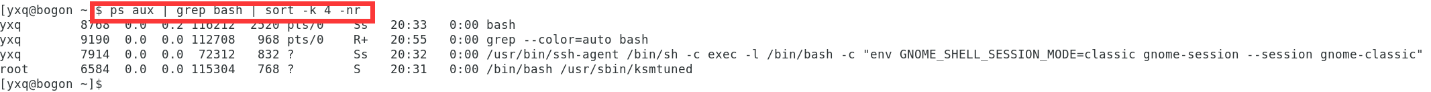


图1-9-4 查看当前活动的进程并筛选出特定的程序

1.9.5 cut-提取特定字段

使用此命令列出 /var/log/messages 文件中最新的 5 条 error 日志，并显示日期和错误信息。

grep "error" /var/log/messages | tail -n 5 | cut -d ' ' -f 1,2,3,4,5

* grep "error" /var/log/message：在文件中查找所有包含 "error" 的行
* tail -n 5：只显示文本最后5行
* | cut -d ' ' -f 1,2,3,4,5：只保留每行的前五列数据

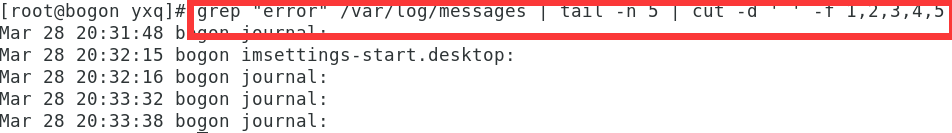


图 1-9-5 筛选、排序并提取特定字段

1.9.6 xargs-将标准输入转换成命令行参数

ls \*.txt | xargs wc -l

使用该命令列出当前目录下所有 .txt 文件并统计每个文件的行数。

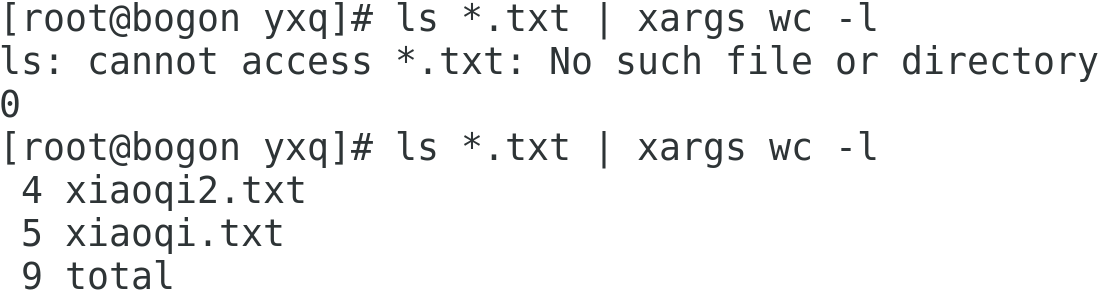


图1-9-6 结合 xargs 处理更复杂的命令

# 2 Linux Shell编程

要求：

⑴ 知道如何执行Shell程序；

⑵ 在Shell脚本中要体现条件控制（如if结构和条件分支），if语句不少于4个；

⑶ 在Shell脚本中要体现循环（for、while和until循环），循环嵌套不少于1层；

⑷ 掌握Shell程序的调试；

⑸ 每个Shell 脚本中有效语句不少于20行；

⑹ 符合编程规范，符合《计算机学院考查类课程报告撰写规范》要求；

⑺ 提交电子版报告与源代码，连同两次作业放在一个压缩包里；

⑻ 提交网站：219.219.61.252（课程名称：**Linux操作系统2024-2025(2)**）；

# 2.1 执行shell程序

2.1.1 创建shell脚本

使用vi/vim命令创建文件demo.sh

vi demo.sh

1. 进入命令模式

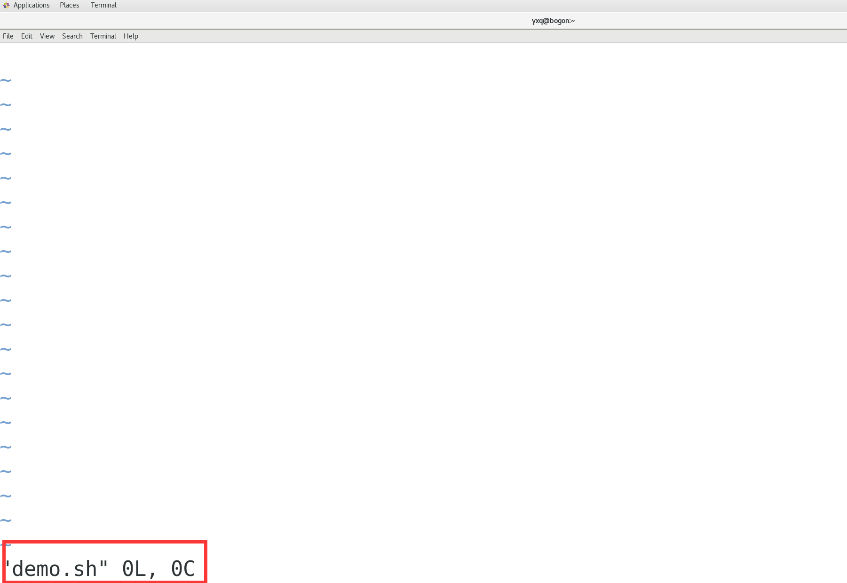


图2-1 进入vim编辑器

在命令模式下按下 **i** 就进入了输入模式，使用 Esc 键可以返回到普通模式。

②进入输入模式

在输入模式中，可以使用以下按键：

|  |  |
| --- | --- |
| 字符按键以及Shift组合 | 输入字符 |
| ENTER回车键 | 换行 |
| BACK SPACE退格键 | 删除光标前一个字符 |
| DEL删除键 | 删除光标后一个字符 |
| 方向键 | 在文本中移动光标 |

③进入底线命令模式

在命令模式下按下 **:**就进入了底线命令模式。

底线命令模式中，基本的命令如下：

|  |  |
| --- | --- |
| :w | 保存文件 |
| :q | 退出 vim 编辑器 |
| :wq | 保存文件并退出 vim 编辑器 |
| :q! | 强制退出vim编辑器，不保存修改 |

2.1.2 编写shell脚本内容

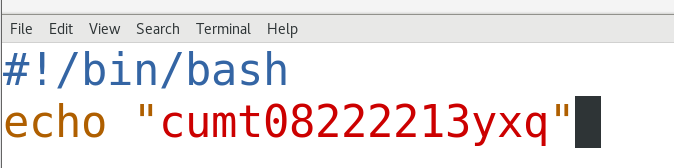


图2-2 编写shell程序

其中：

**#!** 告诉系统其后路径所指定的程序即是解释此脚本文件的 Shell 程序

echo 命令用于向窗口输出文本

2.1.3 运行shell脚本

①作为可执行程序

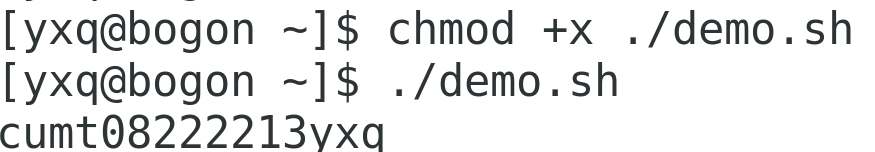


图2-3 作为可执行程序运行shell脚本

1. 作为解释器参数

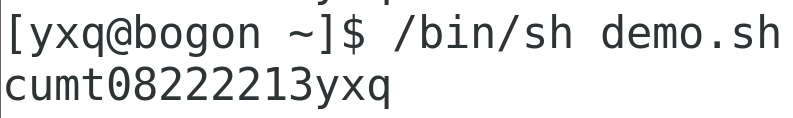


图2-4 作为解释器参数运行shell脚本

# 2.2 if结构和条件分支

2.2.1 四则简易计算器

源代码如下：

#!/bin/bash  
  
echo "欢迎使用CUMT简易计算器！"  
echo "请选择您要进行的运算："  
echo "1. 加法 (+)"  
echo "2. 减法 (-)"  
echo "3. 乘法 (\*)"  
echo "4. 除法 (/)"  
read -p "请输入您的选择（1-4）：" choice  
  
if [ "$choice" -lt 1 ] || [ "$choice" -gt 4 ]; then  
    echo "无效的选择，请输入1到4之间的数字。"  
    exit 1  
fi  
  
read -p "请输入第一个数字：" num1  
read -p "请输入第二个数字：" num2  
  
if [ "$choice" -eq 1 ]; then  
    result=$(echo "$num1 + $num2" | bc)  
    echo "结果：$num1 + $num2 = $result"  
elif [ "$choice" -eq 2 ]; then  
    result=$(echo "$num1 - $num2" | bc)  
    echo "结果：$num1 - $num2 = $result"  
elif [ "$choice" -eq 3 ]; then  
    result=$(echo "$num1 \* $num2" | bc)  
    echo "结果：$num1 \* $num2 = $result"  
elif [ "$choice" -eq 4 ]; then  
    if [ "$num2" -eq 0 ]; then  
        echo "错误：除数不能为零！"  
        exit 1  
    fi  
    result=$(echo "scale=2; $num1 / $num2" | bc)  
    echo "结果：$num1 / $num2 = $result"  
fi  
  
echo "感谢使用CUMT简易计算器！"

这段代码通过使用if结合和多条件分支实现了用户可以输入两个数字并选择加减乘除四种运算返回正确结果的功能。

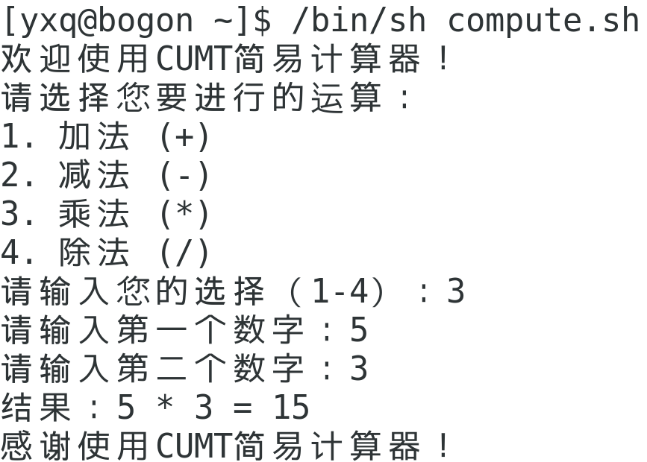


图2-5 compute.sh运行结果

2.2.2 计算BMI并提供健康指南

源代码如下：

#!/bin/bash  
  
echo "欢迎使用高级BMI计算器！我们将为您提供个性化的健康建议。"  
  
read -p "请输入您的体重（公斤）：" weight  
read -p "请输入您的身高（米）：" height  
read -p "请输入您的年龄：" age  
read -p "请输入您的性别（男/女）：" gender  
echo "请选择您的日常活动水平："  
echo "1. 低（很少或没有运动）"  
echo "2. 中（轻度运动或每周3-5天的锻炼）"  
echo "3. 高（高强度运动或每天锻炼）"  
read -p "请输入您的选择（1-3）：" activity\_level  
  
bmi=$(echo "scale=2; $weight / ($height \* $height)" | bc)  
  
echo "您的BMI指数是: $bmi"

if (( $(echo "$bmi < 18.5" | bc -l) )); then  
    bmi\_suggestion="您的体重过轻。建议您增加营养摄入，保持均衡饮食，并定期进行体检。"  
elif (( $(echo "$bmi >= 18.5 && $bmi < 24.9" | bc -l) )); then  
    bmi\_suggestion="您的体重正常。继续保持健康的饮食习惯和适量的运动。"  
elif (( $(echo "$bmi >= 25 && $bmi < 29.9" | bc -l) )); then  
    bmi\_suggestion="您的体重过重。建议您减少高热量食物的摄入，适当增加体育活动。"  
elif (( $(echo "$bmi >= 30" | bc -l) )); then

 bmi\_suggestion="您属于肥胖范围。建议您咨询医生或营养师，制定合理的减肥计划。"  
else  
    bmi\_suggestion="无法识别的BMI值，请检查您的输入是否正确。"  
fi  
  
echo "$bmi\_suggestion"  
  
if [ "$gender" == "男" ]; then  
    if [ "$activity\_level" -eq 1 ]; then

echo "作为男性，如果您平时运动较少，我们建议您逐渐增加每日步行量，并尝试加入一些力量训练。"  
    elif [ "$activity\_level" -eq 2 ]; then  
        echo "作为男性，如果您有中等程度的活动量，可以考虑多样化您的运动类型，如游泳、骑自行车等。"  
    elif [ "$activity\_level" -eq 3 ]; then  
     echo "作为男性，如果您经常进行高强度运动，请确保您的饮食中含有足够的蛋白质和碳水化合物以支持您的能量需求。"  
    else  
 echo "无效的活动水平选择，请输入1到3之间的数字。"  
        exit 1  
    fi  
elif [ "$gender" == "女" ]; then  
    if [ "$activity\_level" -eq 1 ]; then  
        echo "作为女性，如果您平时运动较少，我们建议您开始轻松的散步，并逐渐增加强度。"  
    elif [ "$activity\_level" -eq 2 ]; then  
        ech echo "作为女性，如果您有中等程度的活动量，可以考虑参加瑜伽或舞蹈课程，这有助于增强柔韧性和心肺功能。"

    elif [ "$activity\_level" -eq 3 ]; then  
        echo "作为女性，如果您经常进行高强度运动，请特别注意铁质和其他微量元素的补充。"  
    else  
        echo "无效的活动水平选择，请输入1到3之间的数字。"  
        exit 1  
    fi  
else  
    echo "无效的性别输入，请输入“男”或“女”。"

    exit 1  
fi

if [ "$age" -lt 18 ]; then  
    echo "青少年时期是养成健康生活习惯的关键期，请保持良好的饮食和运动习惯。"  
elif [ "$age" -ge 18 ] && [ "$age" -le 60 ]; then  
    echo "成年人应定期监测健康状况，保持均衡饮食和适量运动。"  
else  
    echo "老年人应注意预防跌倒，保持适度的体力活动，如有需要请咨询医生。"  
fi  
  
echo "感谢使用BMI计算器"

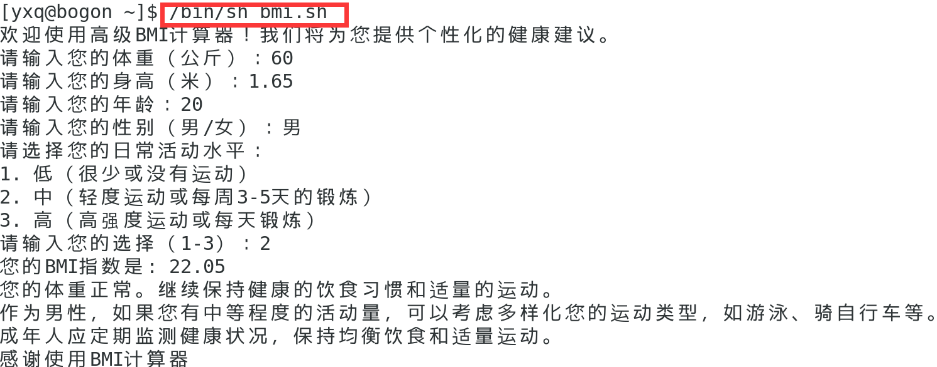


图2-6 bmi.sh运行结果

这段代码同样使用if结构和多条件分支，它通过用户输入的体重、身高、年龄、性别以及日常活动水平，计算出BMI值，并根据这些信息提供个性化的健康建议。它不仅根据BMI范围给出基本的健康提示，还会根据用户的性别、活动水平和年龄提供更具体的建议，帮助用户更好地管理健康。

# 2.3 for、while和until循环的结合

通过多层嵌套实现了一个简单的对文件的备份

源代码如下：

#!/bin/bash

backup\_dir="/tmp/backup"

text\_backup="$backup\_dir/text"

image\_backup="$backup\_dir/images"

other\_backup="$backup\_dir/others"

mkdir -p "$text\_backup" "$image\_backup" "$other\_backup"

read -p "请输入要备份的目录路径：" target\_dir

if [ ! -d "$target\_dir" ]; then

echo "错误：目录不存在！"

exit 1

fi

for file in "$target\_dir"/\*; do

if [ -f "$file" ]; then

filename=$(basename "$file")

extension="${filename##\*.}"

filesize=$(du -k "$file" | cut -f1)

while [ "$filesize" -gt 0 ]; do

echo "正在处理文件：$filename (大小: ${filesize}KB)"

user\_input=""

until [[ "$user\_input" == "y" || "$user\_input" == "n" ]]; do

read -p "是否备份文件 $filename？(y/n): " user\_input

if [[ "$user\_input" != "y" && "$user\_input" != "n" ]]; then

echo "无效输入，请输入 y 或 n。"

fi

done

if [ "$user\_input" == "y" ]; then

# 根据文件类型进行分类备份

if [[ "$extension" == "txt" || "$extension" == "log" ]]; then

cp "$file" "$text\_backup"

echo "已将文本文件 $filename 备份到 $text\_backup"

elif [[ "$extension" == "jpg" || "$extension" == "png" ]]; then

cp "$file" "$image\_backup"

echo "已将图片文件 $filename 备份到 $image\_backup"

else

cp "$file" "$other\_backup"

echo "已将其他类型文件 $filename 备份到 $other\_backup"

fi

else

echo "跳过文件 $filename 的备份。"

fi

# 结束while循环

filesize=0

done

fi

done

echo "备份完成！"

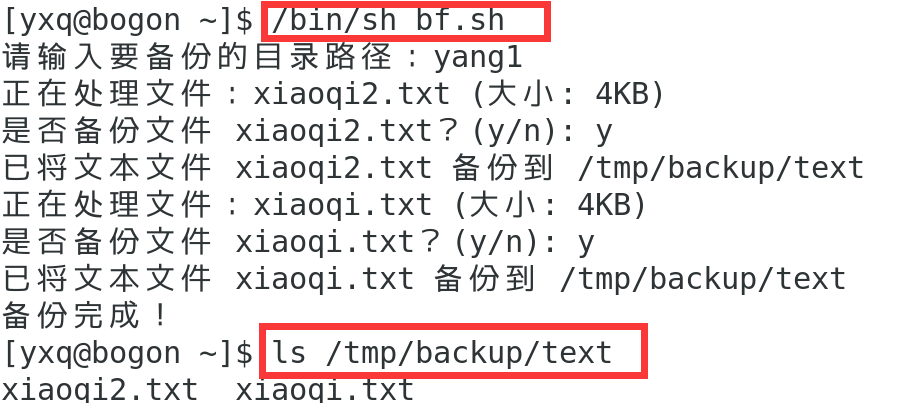


图2-7 bf.sh运行结果

这段代码通过for、while和until的结合，使用for循环遍历指定目录中的所有文件；使用while循环处理文件大小是否满足条件。使用until循环等待用户输入有效的确认选项将文件按大小和类型（如文本文件、图片文件等）分类备份到不同的目录中。

# 2.4 组合使用的综合案例(任务管理系统)

源代码如下：

#!/bin/bash

# 初始化任务列表

tasks=()

priorities=()

statuses=()

# 主菜单

function main\_menu {

echo "=== 任务管理系统 ==="

echo "1. 添加任务"

echo "2. 查看所有任务"

echo "3. 更新任务状态"

echo "4. 统计任务状态"

echo "5. 退出系统"

read -p "请选择操作（1-5）：" choice

}

# 添加任务

function add\_task {

while true; do

read -p "请输入任务名称：（输入'q'返回主菜单）" task\_name

if [[ "$task\_name" == "q" ]]; then

break

fi

# 设置任务优先级

priority=""

until [[ "$priority" == "高" || "$priority" == "中" || "$priority" == "低" ]]; do

read -p "请输入任务优先级（高/中/低）：" priority

if [[ "$priority" != "高" && "$priority" != "中" && "$priority" != "低" ]]; then

echo "无效输入，请输入 高、中 或 低。"

fi

done

# 添加任务到列表

tasks+=("$task\_name")

priorities+=("$priority")

statuses+=("未完成")

echo "任务 '$task\_name' 已成功添加！"

done

}

# 查看所有任务

function view\_tasks {

if [ ${#tasks[@]} -eq 0 ]; then

echo "当前没有任务！"

return

fi

echo "=== 当前任务列表 ==="

for i in "${!tasks[@]}"; do

echo "[$i] 名称: ${tasks[i]} | 优先级: ${priorities[i]} | 状态: ${statuses[i]}"

done

}

# 更新任务状态

function update\_task\_status {

if [ ${#tasks[@]} -eq 0 ]; then

echo "当前没有任务可以更新！"

return

fi

view\_tasks

read -p "请输入要更新的任务编号：（输入'q'返回主菜单）" task\_index

if [[ "$task\_index" == "q" ]]; then

return

elif ! [[ "$task\_index" =~ ^[0-9]+$ ]] || [ "$task\_index" -ge ${#tasks[@]} ]; then

echo "无效的任务编号！"

return

fi

# 更新任务状态

status=""

until [[ "$status" == "已完成" || "$status" == "未完成" ]]; do

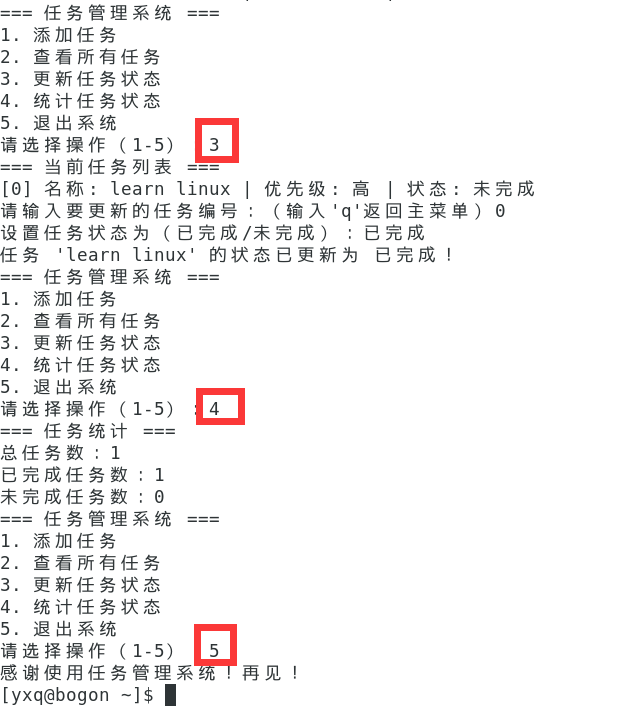
read -p "设置任务状态为（已完成/未完成）：" status

if [[ "$status" != "已完成" && "$status" != "未完成" ]]; then

echo "无效输入，请输入 已完成 或 未完成。"

fi

 done  
  
    statuses["$task\_index"]="$status"  
    echo "任务 &apos;${tasks[$task\_index]}&apos; 的状态已更新为 $status！"  
}  
  
# 统计任务状态  
function stats\_tasks {  
    total=${#tasks[@]}  
    completed=0  
    uncompleted=0  
  
    for status in "${statuses[@]}"; do  
        if [[ "$status" == "已完成" ]]; then  
            ((completed++))  
        else  
            ((uncompleted++))  
        fi  
    done  
   
  
    echo "=== 任务统计 ==="  
    echo "总任务数：$total"  
    echo "已完成任务数：$completed"  
    echo "未完成任务数：$uncompleted"  
}  
  
# 主程序  
while true; do  
    main\_menu  
  
    case "$choice" in  
        1)  
            add\_task  
            ;;  
        2)  
            view\_tasks  
            ;;  
        3)  
            update\_task\_status  
            ;;  
        4)  
            stats\_tasks  
            ;;  
        5)  
            echo "感谢使用任务管理系统！再见！"  
            break  
            ;;  
        \*)  
            echo "无效选择，请输入1-5之间的数字。"  
            ;;  
    esac  
done



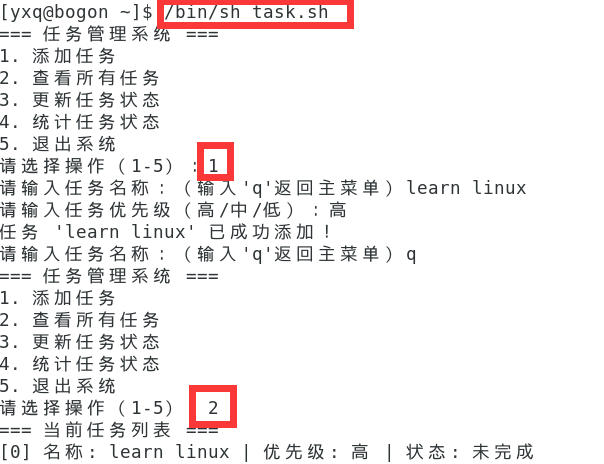


图2-8 task.sh运行结果

这个任务管理系集成了条件控制（如if结构和多条件分支）和多种循环（for、while和until循环），提供了任务的创建、查看、更新及统计功能。用户可以添加具有不同优先级的任务，标记任务为已完成或未完成，并查看所有任务的详细信息以及整体的任务完成情况统计。

# 2.5 shell程序的调试

2.5.1 通过echo方式

echo最简单的调试方法，可以在任何怀疑出错的地方用echo打印变量来进行调试自己的程序。

echo $变量名

2.5.2 通过选项的方式

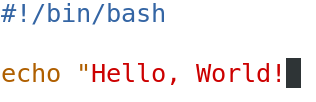
① -n选项

bash -n hello.sh

这个选项可以读取shell脚本，但不实际执行，用于测试Shell脚本中是否存在语法错误。

如下图所示，输出字符串“hello,world!”时，未输出后引号”，使用该命令来检查语法错误。

提示：第 3 行的单引号没有正确闭合。第 4 行缺少了相应的语句或结构，导致提前结束。



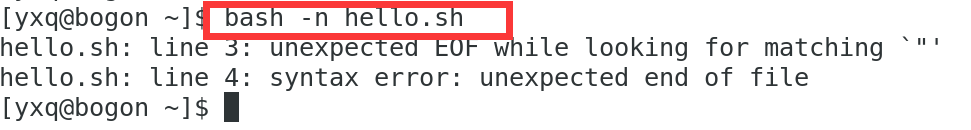


图 2-9 bash -n hello.sh进行调试

② -c选项

bash -c 脚本

该选项使Shell解析器从字符串而非文件中读取并执行命令**,** 当需要调试一小段脚本的执行结果时，非常方便。

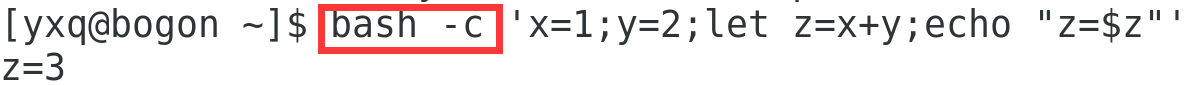


图2-10 使用-c选项进行简单调试

③ -v选项

bash -v hello.sh

该选项用于：

* 如果脚本没有问题，脚本中每条命令被依次打印出来，然后是该命令的执行结果。
* 如果有错误，错误会按照正常的错误处理流程输出，还可以看到导致错误的那条命令是什么。

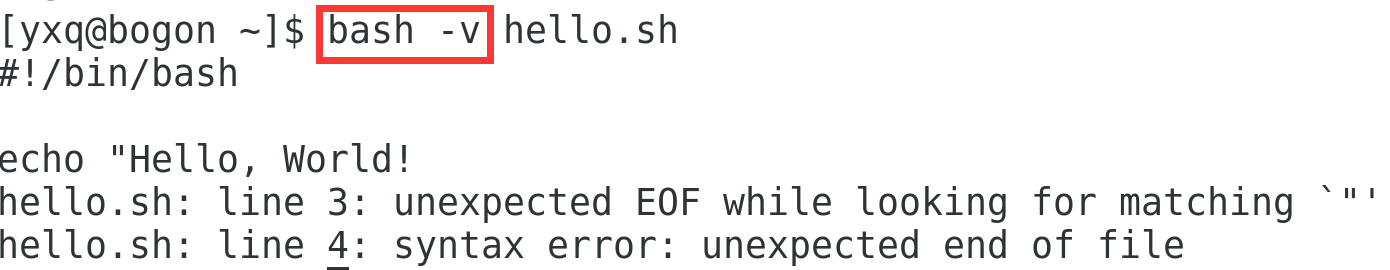


图2-11 -v选项进行调试

④ -x选项

bash -x hello.sh

提供跟踪执行信息,将执行脚本的过程中把实际执行的每个命令显示出来，行首显示+, +后面显示经过替换之后的命令行内容。

如下图所示，将上述错误的hello.sh脚本修改正确，使用该选项。

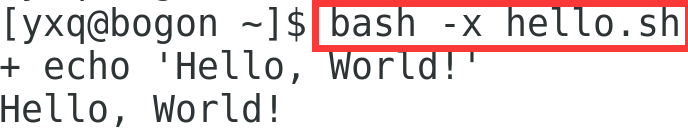


图 2-12 -x选项进行调试

2.5.3 通过test方式

[表达式]

返回值为0表示真，为1表示假。如下图所示，对两个数进行比较。

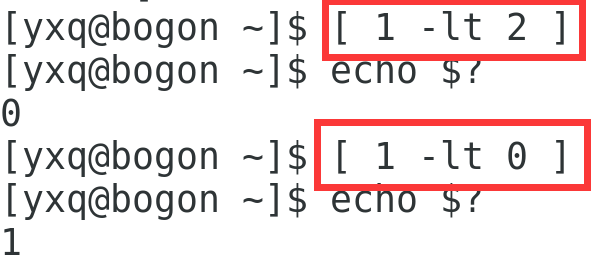


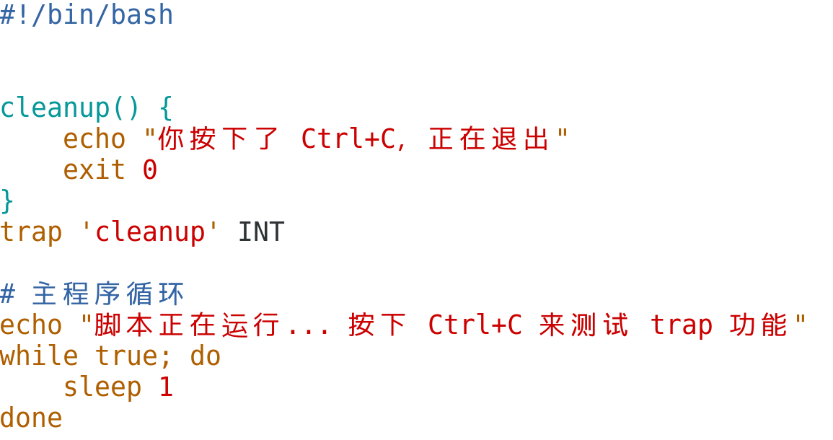
图2-13 test方法进行调试

2.5.4 通过trap的方式

用于捕获指定的信号并执行预定义的命令

trap ‘cleanup’ INT

如下图所示，使用trap ‘cleanup’ INT，当按下 Ctrl+C（即发送 SIGINT 信号）时捕获该信号，并cleanup函数。



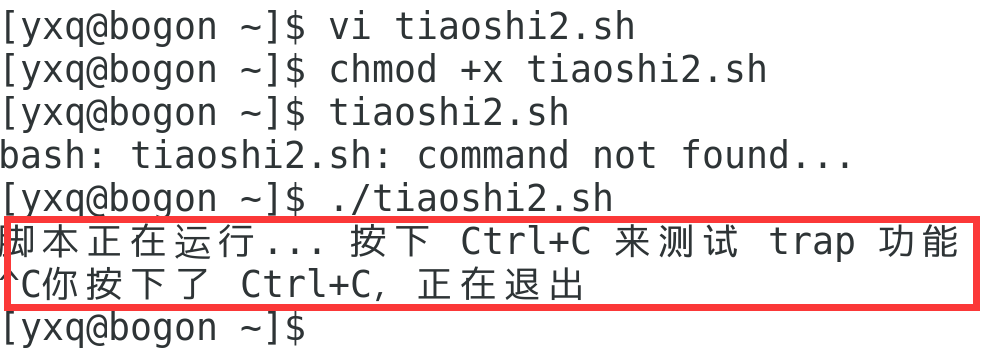


图2-14 通过trap进行调试

2.5.5 使用调试工具-bashdb

使用shell调试器bashdb，这是一个类似于GDB的调试工具，可以完成对shell脚本的断点设置，单步执行，变量观察等许多功能。

bashdb --debug shell脚本