

**本科生实验报告**

**实验课程 Linux操作系统**

**学院名称 计算机与网络安全学院（牛津布鲁克斯学院）**

**专业名称 软件工程**

**学生姓名 郑昌良**

**学生学号 202013160217**

**指导教师 李军**

**实验地点 C075-05-35**

**实验成绩**

**二〇二二年四月 二〇二二年六月**

# 实验二 基本概念及操作

## 实验内容

1. 实验楼环境介绍
2. 常用 Shell 命令及快捷键
3. Linux 使用小技巧

## 实验目的

1. 掌握Linux 基本命令
2. 练习通配符的使用
3. 学会查看帮助文档

## 实验知识点

1. Linux 基本命令
2. 通配符的使用
3. 查看帮助文档

## Linux基本概念

### 1、Linux终端

通常我们在使用 Linux 时，并不是直接与系统打交道，而是通过一个叫做 Shell 的中间程序来完成的，在图形界面下为了实现让我们在一个窗口中完成用户输入和显示输出，Linux 系统还提供了一个叫做终端模拟器的程序（Terminal）。 下面是几个比较常见的终端模拟器： gnome-terminal，Konsole，xterm，rxvt，kvt，nxterm 和 eterm 。目前我们的实验中的终端程序是 xfce 桌面环境自带的 xfce-terminal。不过要注意的是这里所说的终端（Terminal）和控制台（Console）是有区别的。

终端本质上是对应着 Linux 上的 /dev/tty 设备，Linux 的多用户登录就是通过不同的 /dev/tty 设备完成的，Linux 默认提供了 6 个纯命令行界面的 “terminal”（准确的说这里应该是 6 个 virtual consoles）来让用户登录。在物理机系统上你可以通过使用[Ctrl]+[Alt]+[F1]～[F6]进行切换，不过在我们的在线实验环境中可能无法切换，因为特殊功能按键会被你的主机系统劫持。当你切换到其中一个终端后想要切换回图形界面，你可以按下[Ctrl]+[Alt]+[F7]来完成。

### 2、Shell

通常在图形界面中对实际体验带来差异的不是上述的不同发行版的各种终端模拟器，而是这个 Shell（壳）。有壳就有核，这里的核就是指 UNIX/Linux 内核，Shell 是指“提供给使用者使用界面”的软件（命令解析器），类似于 DOS 下的 command（命令行）和后来的 cmd.exe 。普通意义上的 Shell 就是可以接受用户输入命令的程序。它之所以被称作 Shell 是因为它隐藏了操作系统底层的细节。同样的 UNIX/Linux 下的图形用户界面 GNOME 和 KDE，有时也被叫做“虚拟 shell”或“图形 shell”。

UNIX/Linux 操作系统下的 Shell 既是用户交互的界面，也是控制系统的脚本语言。当然这一点也有别于 Windows 下的命令行，虽然该命令行也提供了很简单的控制语句。在 Windows 操作系统下，有些用户从来都不会直接使用 Shell，然而在 UNIX 系列操作系统下，Shell 仍然是控制系统启动、X11 启动和很多其它实用工具的脚本解释程序。

在 UNIX/Linux 中比较流行的常见的 Shell 有 bash、zsh、ksh、csh 等等，Ubuntu 终端默认使用的是 bash，默认的桌面环境是 GNOME 或者 Unity（基于 GNOME），但我们的环境中使用的分别是 zsh 和 xfce。

### 3、命令行操作体验

1．双击终端运行Shell程序，即可输入命令让系统执行命令。

2．使用Tab键可以进行命令补全，当然不止补全命令，补全目录、补全命令参数都是没问题的。

3． 使用 Ctrl+C可以终止命令运行。想想你有没有遇到过这种情况，当你在 Linux 命令行中无意输入了一个不知道的命令，或者错误地使用了一个命令，导致在终端里出现了你无法预料的情况，比如，屏幕上只有光标在闪烁却无法继续输入命令，或者不停地输出一大堆你不想要的结果。你想要立即停止并恢复到你可控的状态，那该怎么办呢？这时候你就可以使用Ctrl+c键来强行终止当前程序（你可以放心它并不会使终端退出）。

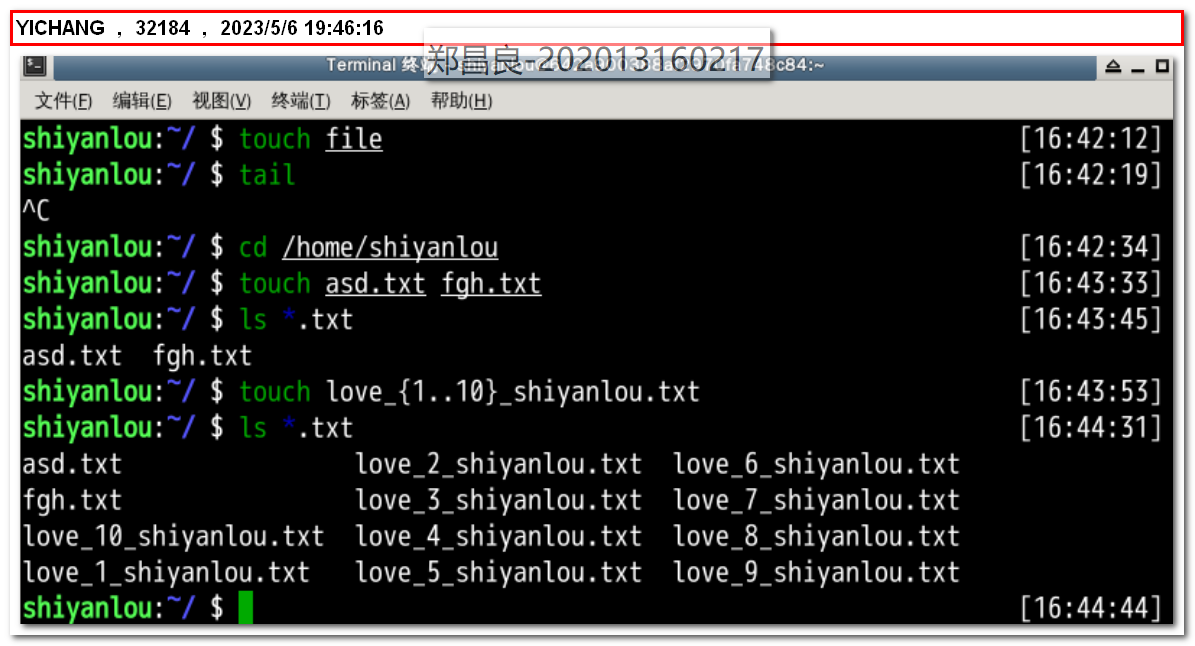
4. 利用历史输入命令

可以使用键盘上的方向上键↑，恢复你之前输入过的命令。

5. 使用通配符

通配符是一种特殊语句，主要有星号（\*）和问号（?），用来对字符串进行模糊匹配（比如文件名、参数名）。当查找文件夹时，可以使用它来代替一个或多个真正字符；当不知道真正字符或者懒得输入完整名字时，常常使用通配符代替一个或多个真正字符。

终端里面输入的通配符是由 Shell 处理的，不是由所涉及的命令语句处理的，它只会出现在命令的“参数值”里（它不能出现在命令名称里， 命令不记得，那就用Tab补全）。当 Shell 在“参数值”中遇到了通配符时，Shell 会将其当作路径或文件名在磁盘上搜寻可能的匹配：若符合要求的匹配存在，则进行代换（路径扩展）；否则就将该通配符作为一个普通字符传递给“命令”，然后再由命令进行处理。总之，通配符实际上就是一种 Shell 实现的路径扩展功能。在通配符被处理后， Shell 会先完成该命令的重组，然后继续处理重组后的命令，直至执行该命令。



6. 在命令行中使用帮助

在 Linux 环境中，如果你遇到困难，可以使用man命令，它是Manual pages的缩写。

Manual pages 是 UNIX 或类 UNIX 操作系统中在线软件文档的一种普遍的形式， 内容包括计算机程序（包括库和系统调用）、正式的标准和惯例，甚至是抽象的概念。用户可以通过执行man命令调用手册页。

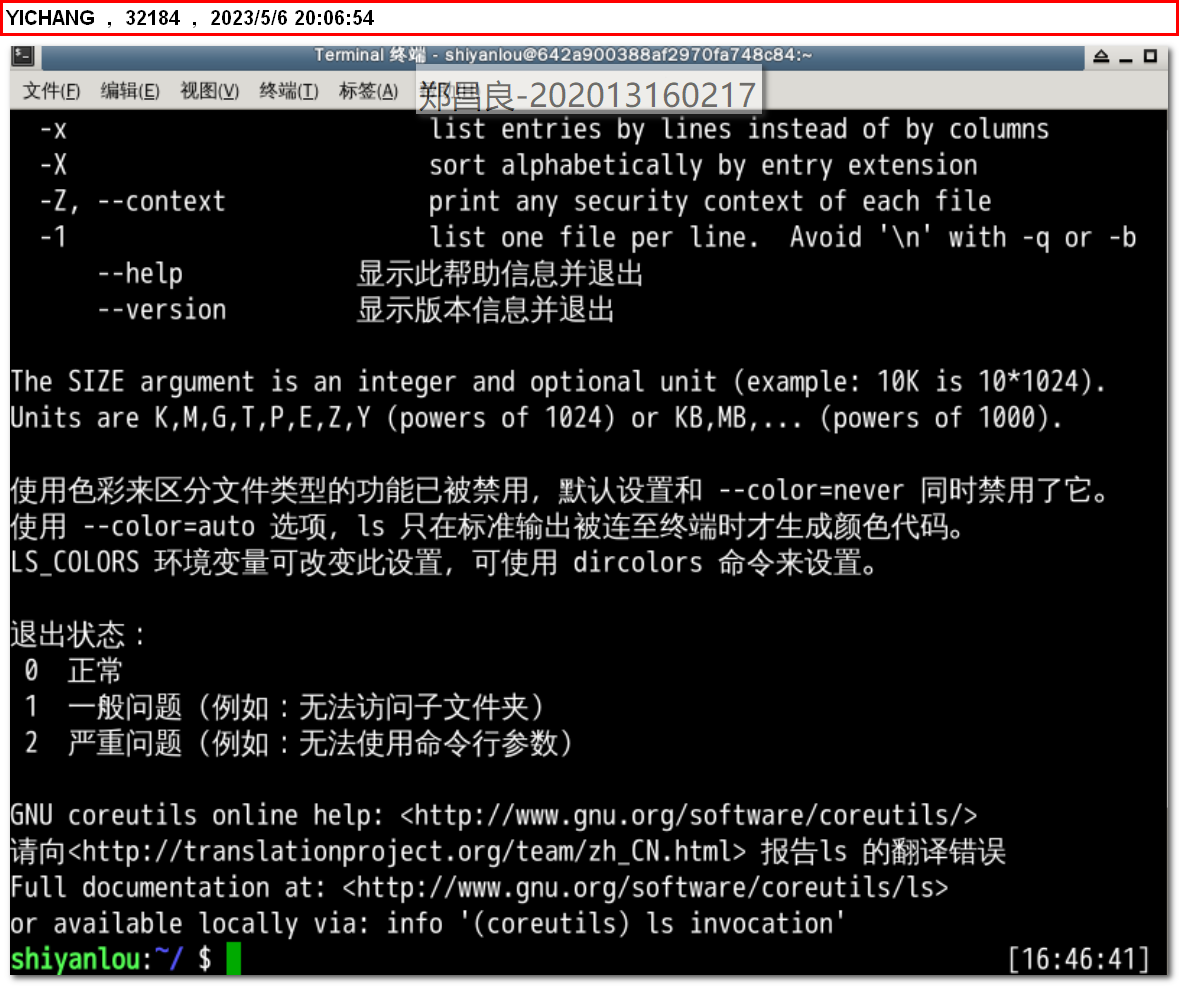
可以使用如下方式来获得某个命令的说明和使用方式的详细介绍：

man <command\_name>

通常 man 手册中的内容很多，你可能不太容易找到你想要的结果，不过幸运的是你可以在 man 中使用搜索/<你要搜索的关键字>，查找完毕后你可以使用n键切换到下一个关键字所在处，shift+n为上一个关键字所在处。使用Space（空格键）翻页，Enter（回车键）向下滚动一行，或者使用k,j（vim 编辑器的移动键）进行向前向后滚动一行。按下h键为显示使用帮助（因为 man 使用 less 作为阅读器，实为less工具的帮助），按下q退出。

想要获得更详细的帮助，你还可以使用info命令，不过通常使用man就足够了。如果你知道某个命令的作用，只是想快速查看一些它的某个具体参数的作用，那么你可以使用--help参数，大部分命令都会带有这个参数，如：





# 实验三 用户及文件权限管理

## 实验内容

1、Linux 中创建、删除用户，及用户组等操作。

2、Linux 中的文件权限设置。

## 实验目的

1. 掌握Linux用户管理
2. 掌握Linux权限管理

## 三、 实验知识点

1、Linux 用户管理

2、Linux 权限管理

## 四、 实验内容

### 1、查看用户

$ who am i

# 或者

$ who mom likes



### 2、创建用户、用户组、删除用户和用户组

创建用户：

$ sudo adduser lilei



使用如下命令切换登录用户：

$ su -l lilei

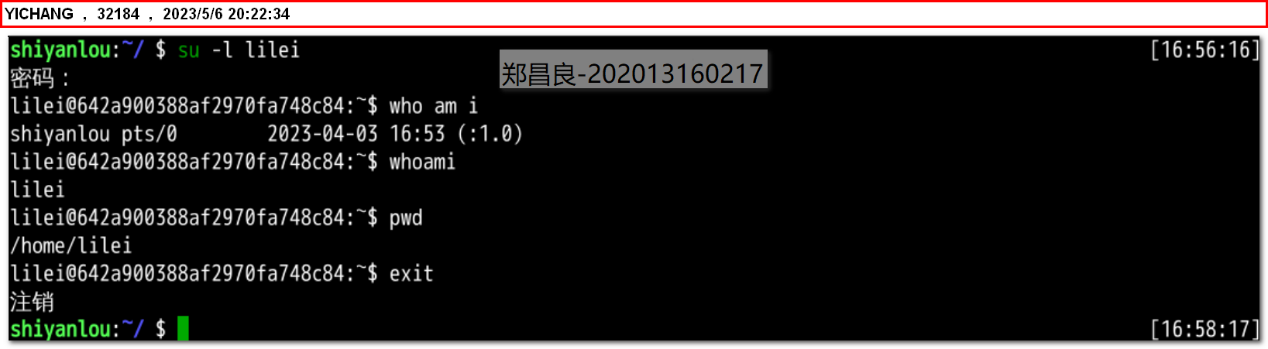
输入刚刚设置的 lilei 的密码，然后输入如下命令并查看输出：

$ who am i

$ whoami

$ pwd

退出当前用户跟退出终端一样，可以使用 exit 命令或者使用快捷键 Ctrl+D。



在 Linux 里面如何知道自己属于哪些用户组呢？

方法一、使用 groups 命令

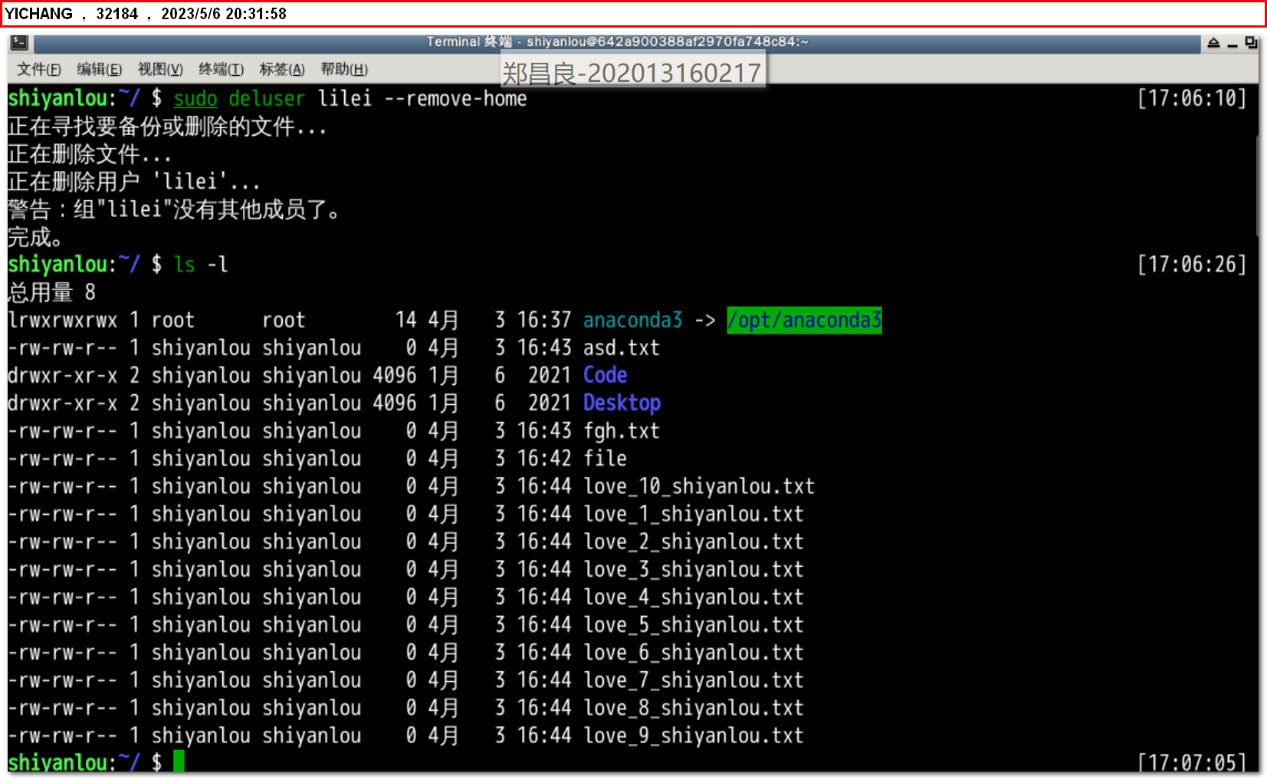
$ groups shiyanlou



方法二、查看/etc/group文件

删除用户组：

$ sudo deluser lilei --remove-home

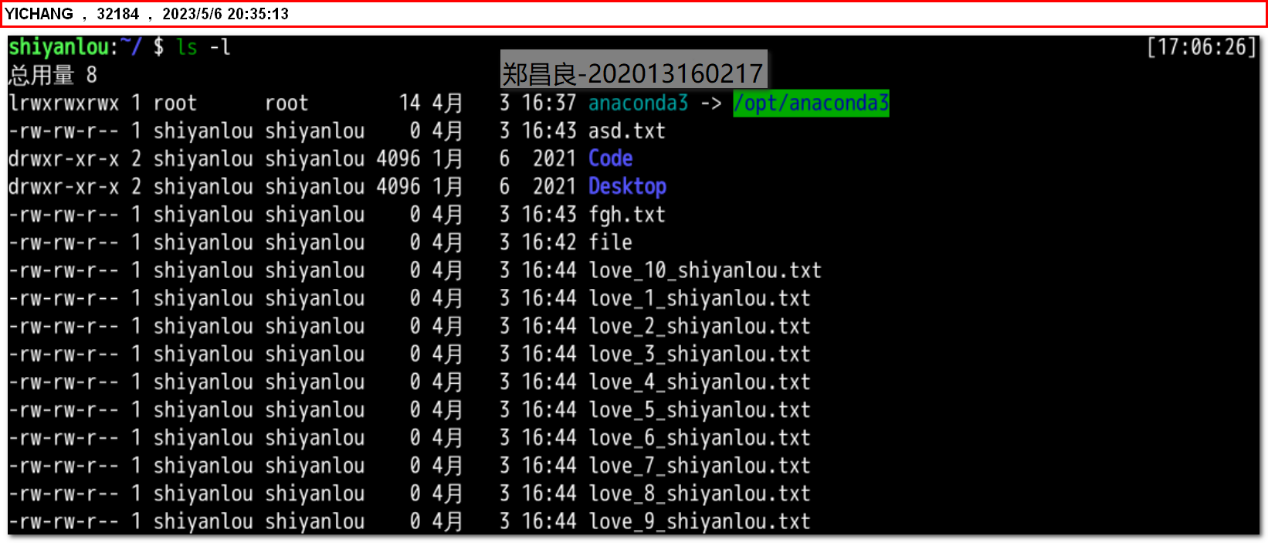


### 3、Linux文件权限

查看文件权限

使用较长格式列出文件：

$ ls -l

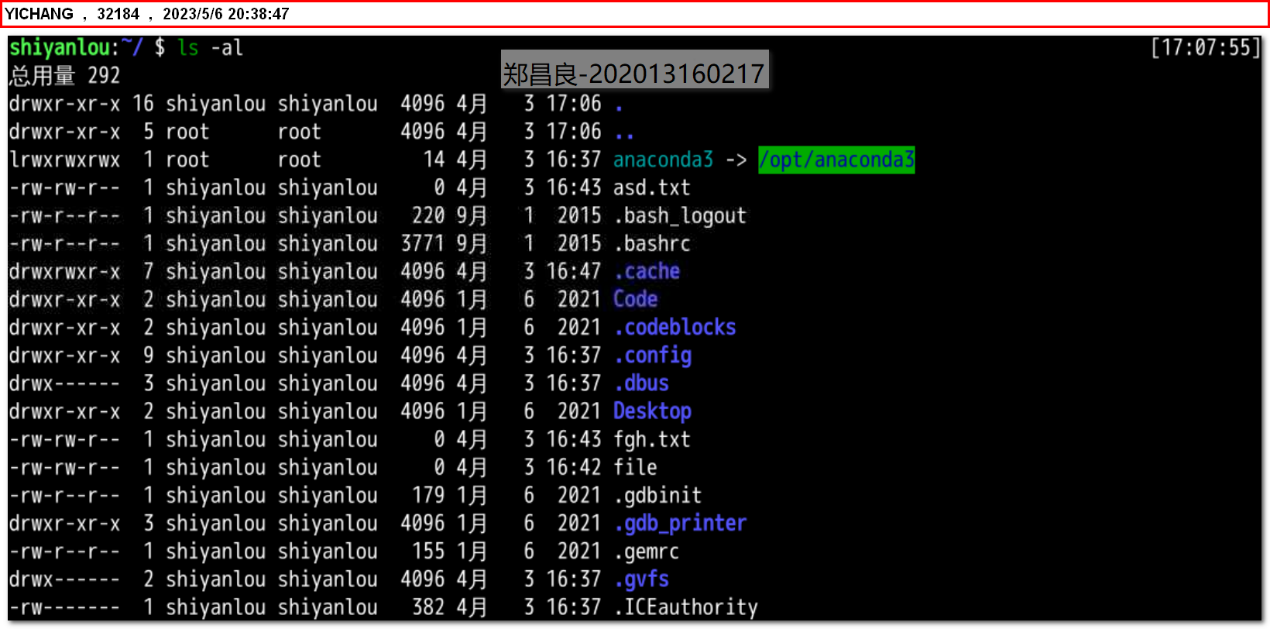


显示除了 .（当前目录）和 ..（上一级目录）之外的所有文件，包括隐藏文件（Linux 下以 . 开头的文件为隐藏文件）。

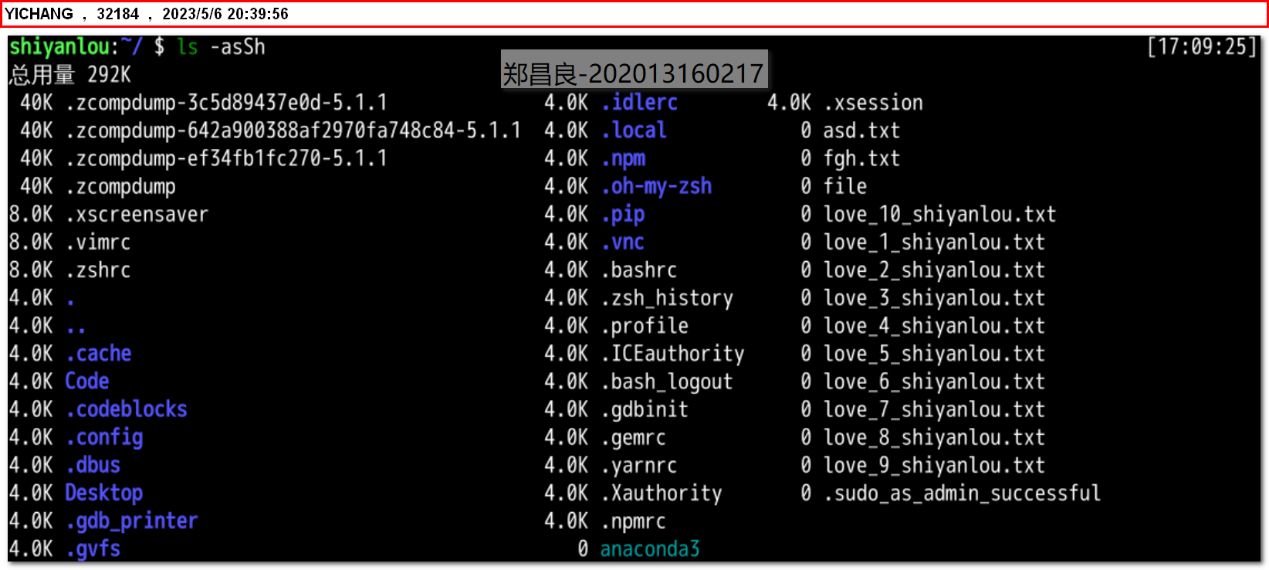
$ ls -a



$ ls -al



$ ls -asSh



变更文件所有者：

$ su - lilei

$ pwd

$ touch iphone11

$ ls -alh iphone11

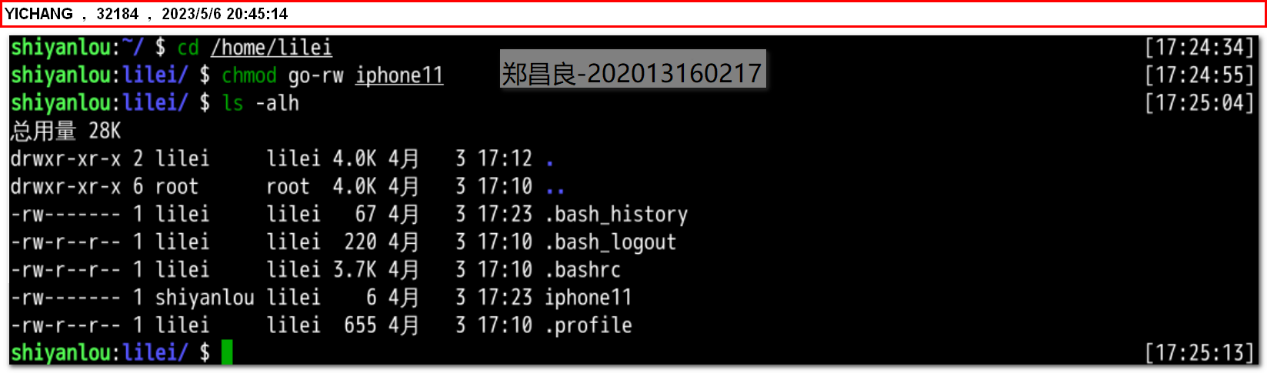


现在切换回到 shiyanlou 用户，使用以下命令变更文件所有者为 shiyanlou。

$ cd /home/lilei

$ ls iphone11

$ sudo chown shiyanlou iphone11



# 实验四 Linux 目录结构及文件基本操作

## 实验内容

1、Linux 的文件组织目录结构。

2、相对路径和绝对路径。

1. 对文件的移动、复制、重命名、编辑等操作。

## 二、实验目的

1、了解每个目录的大体内容

2、了解文件的属性

1. 使用touch，file，rm，mv 等基本命令

## 三、实验知识点

1、每个目录的大体内容

2、文件的属性

1. touch，file，rm，mv 等基本命令

## 四、Linux基本概念

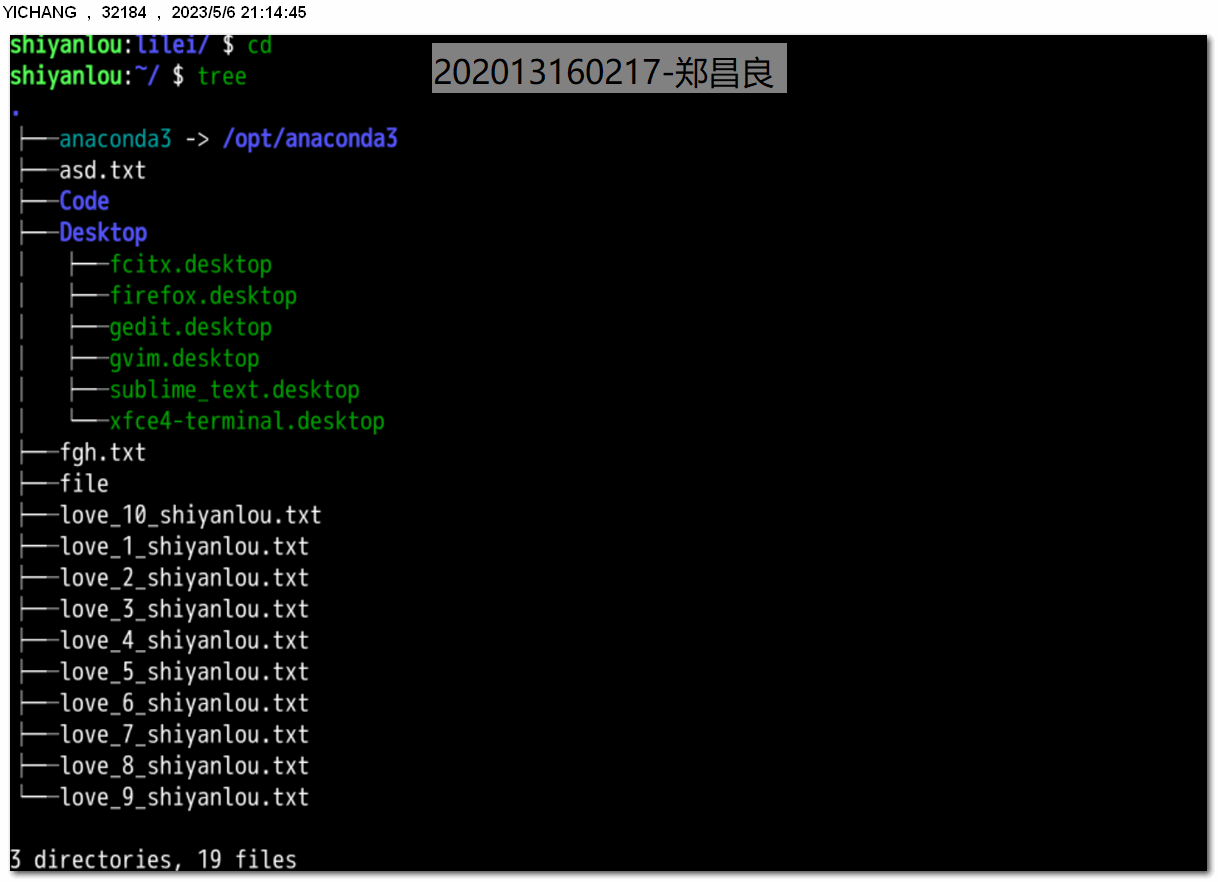
### FHS标准

FHS 定义了两层规范，第一层是， / 下面的各个目录应该要放什么文件数据，例如 /etc 应该放置设置文件，/bin 与 /sbin 则应该放置可执行文件等等。

第二层则是针对 /usr 及 /var 这两个目录的子目录来定义。例如 /var/log 放置系统日志文件，/usr/share 放置共享数据等等。

使用tree查看文件的树状图

$ tree /



### 目录路径

上一级：

cd ..

进入你的 home 目录：

cd ~

使用 pwd 获取当前路径：

pwd

绝对路径：

关于绝对路径，简单地说就是以根" / "目录为起点的完整路径，以你所要到的目录为终点，表现形式如： /usr/local/bin，表示根目录下的 usr 目录中的 local 目录中的 bin 目录。

相对路径：

相对路径，也就是相对于你当前的目录的路径，相对路径是以当前目录 . 为起点，以你所要到的目录为终点，表现形式如： usr/local/bin （这里假设你当前目录为根目录）。你可能注意到，我们表示相对路径实际并没有加上表示当前目录的那个 . ，而是直接以目录名开头，因为这个 usr 目录为 / 目录下的子目录，是可以省略这个 . 的（以后会讲到一个类似不能省略的情况）；如果是当前目录的上一级目录，则需要使用 .. ，比如你当前目录为 /home/shiyanlou 目录下，根目录就应该表示为 ../../ ，表示上一级目录（ home 目录）的上一级目录（ / 目录）。

下面我们以你的 home 目录为起点，分别以绝对路径和相对路径的方式进入 /usr/local/bin 目录：

# 绝对路径

$ cd /usr/local/bin

# 相对路径

$ cd ../../usr/local/bin



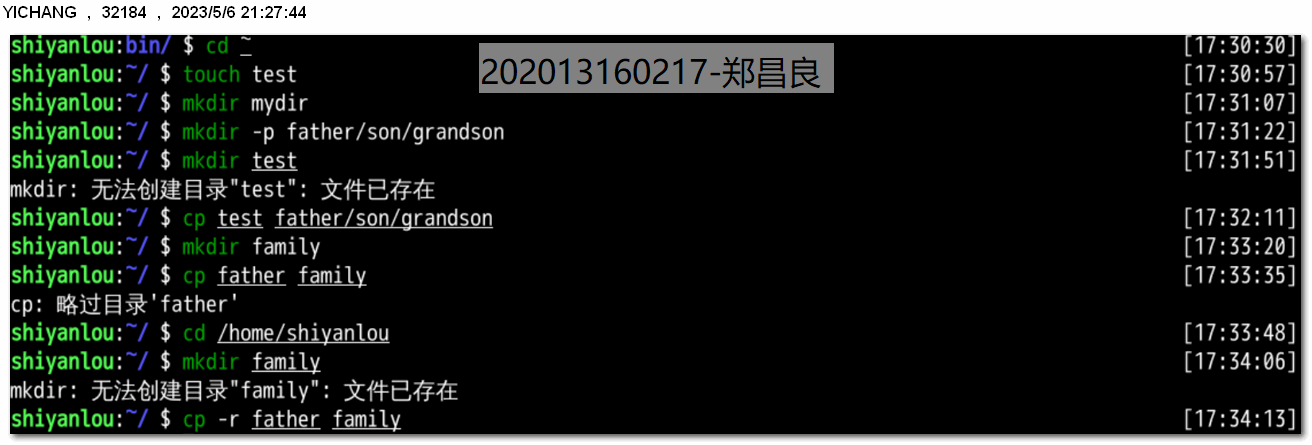
### 3、Linux文件操作

1. 新建目录

$ mkdir mydir

2. 使用 -p 参数，同时创建父目录（如果不存在该父目录），如下我们同时创建一个多级目录（这在安装软件、配置安装路径时非常有用）：

$ mkdir -p father/son/grandson



3. 复制文件

使用 cp 命令（copy）复制一个文件到指定目录。

将之前创建的 test 文件复制到 /home/shiyanlou/father/son/grandson 目录中：

$ cp test father/son/grandson

4. 复制目录

要成功复制目录需要加上 -r 或者 -R 参数，表示递归复制：

$ cd /home/shiyanlou

$ mkdir family

$ cp -r father family

5. 删除文件

使用 rm（remove files or directories）命令删除一个文件：

$ rm test

你如果想忽略这提示，直接删除文件，可以使用 -f 参数强制删除：

$ rm -f test

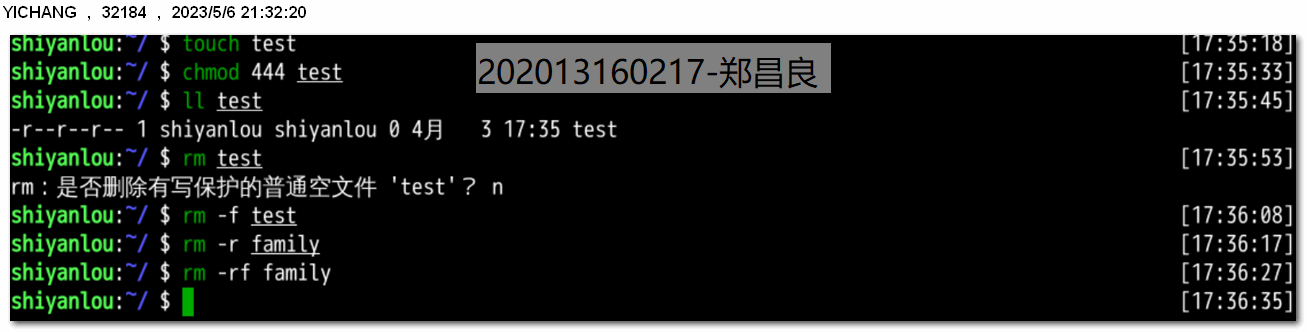
6. 删除目录

跟复制目录一样，要删除一个目录，也需要加上 -r 或 -R 参数：

$ rm -r family

遇到权限不足删除不了的目录也可以和删除文件一样加上 -f 参数：

$ rm -rf family



### 4、移动文件与文件重命名

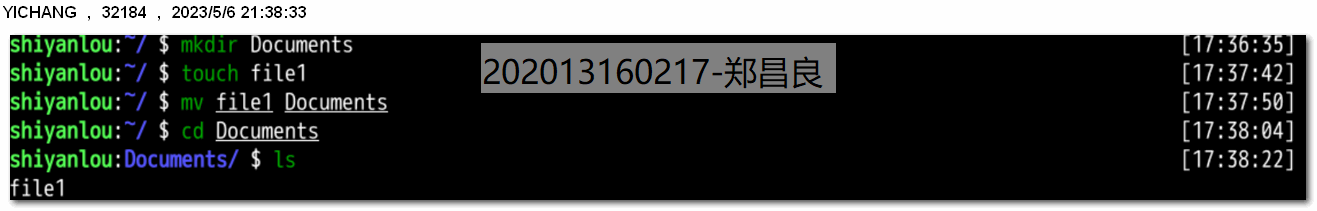
1. 移动文件

使用 mv（move or rename files）命令移动文件（剪切）。命令格式是 mv 源目录文件 目的目录。

$ mkdir Documents

$ touch file1

$ mv file1 Documents



2. 重命名文件

mv 命令除了能移动文件外，还能给文件重命名。命令格式为 mv 旧的文件名 新的文件名。

例如将文件“ file1 ”重命名为“ myfile ”：

$ mv file1 myfile

3. 批量重命名

要实现批量重命名，mv 命令就有点力不从心了，我们可以使用一个看起来更专业的命令 rename 来实现。不过它要用 perl 正则表达式来作为参数，关于正则表达式我们要在后面才会介绍到，这里只做演示，你只要记得这个 rename 命令可以批量重命名就好了，以后再重新学习也不会有任何问题，毕竟你已经掌握了一个更常用的 mv 命令。

rename 命令并不是内置命令，若提示无该命令可以使用 sudo apt-get install rename 命令自行安装。

$ cd /home/shiyanlou/

# 使用通配符批量创建 5 个文件:

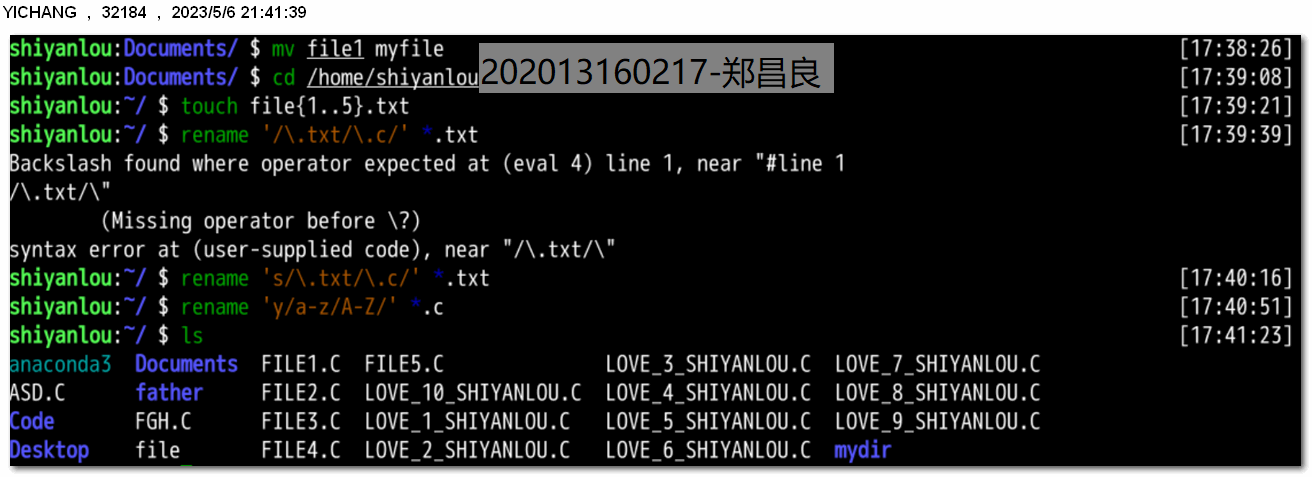
$ touch file{1..5}.txt

# 批量将这 5 个后缀为 .txt 的文本文件重命名为以 .c 为后缀的文件:

$ rename 's/\.txt/\.c/' \*.txt

# 批量将这 5 个文件，文件名和后缀改为大写:

$ rename 'y/a-z/A-Z/' \*.c



4. 查看文件

使用 cat，tac 和 nl 命令查看文件

$ cd /home/shiyanlou

$ cp /etc/passwd passwd

$ cat passwd



使用 more 和 less 命令分页查看文件

$ more passwd

使用 head 和 tail 命令查看文件

当我们想要查看最近新增加的用户，那么我们可以查看这个 /etc/passwd 文件，不过我们前面也看到了，这个文件里面一大堆乱糟糟的东西，看起来实在费神啊。因为系统新增加一个用户，会将用户的信息添加到 passwd 文件的最后，那么这时候我们就可以使用 tail 命令了：

$ tail /etc/passwd



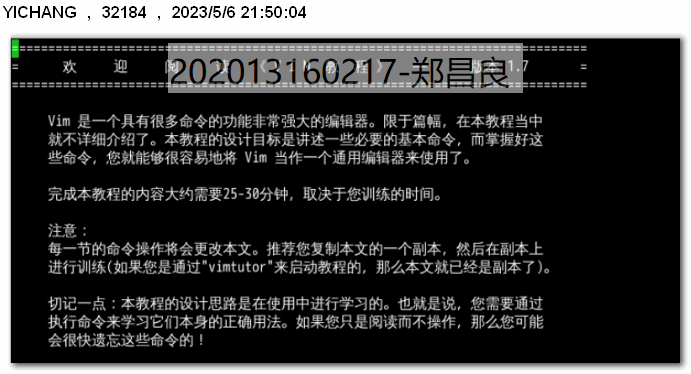
5. 查看文件类型

$ file /bin/ls



6. 编辑文件

在 Linux 下面编辑文件通常我们会直接使用专门的命令行编辑器比如（emacs，vim，nano）。



# 实验五 环境变量与文件查找

## 实验内容

本节实验介绍环境变量的作用与用法，及几种搜索文件的方法。学会这些技巧可以高效地使用 Linux 。

## 实验目的

1、掌握环境变量的设置

2、掌握环境变量的修改

## 三、实验知识点

1、环境变量的设置

2、环境变量的修改

## 四、Linux基本概念

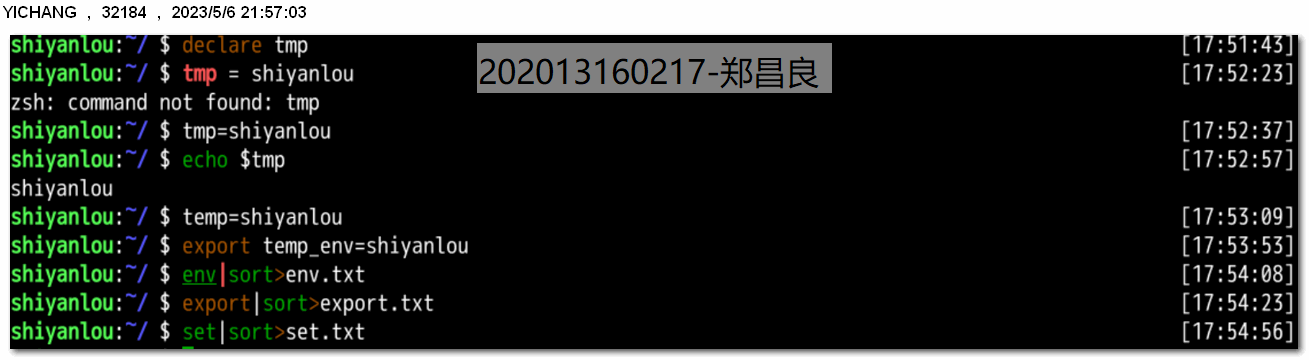
### 1、环境变量

使用 declare 命令创建一个变量名为 tmp 的变量：

$ declare tmp

读取变量的值，使用 echo 命令和 $ 符号

$ echo $tmp



通常我们会涉及到的变量类型有三种：

1. 当前 Shell 进程私有用户自定义变量，如上面我们创建的 tmp 变量，只在当前 Shell 中有效。
2. Shell 本身内建的变量。
3. 从自定义变量导出的环境变量。

可以更直观的使用 vimdiff 工具比较一下它们之间的差别：

$ temp=shiyanlou

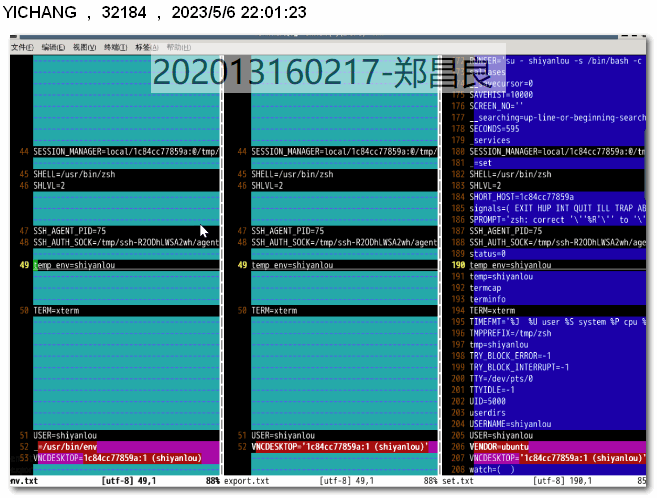
$ export temp\_env=shiyanlou

$ env|sort>env.txt

$ export|sort>export.txt

$ set|sort>set.txt

$ vimdiff env.txt export.txt set.txt



使环境变量永久生效

按变量的生存周期来划分，Linux 变量可分为两类：

永久的：需要修改配置文件，变量永久生效；

临时的：使用 export 命令行声明即可，变量在关闭 shell 时失效。

介绍两个重要文件 /etc/bashrc（有的 Linux 没有这个文件） 和 /etc/profile ，它们分别存放的是 shell 变量和环境变量。还有要注意区别的是每个用户目录下的一个隐藏文件：

$ cd /home/shiyanlou

$ ls -a



### 命令的查找路径与顺序

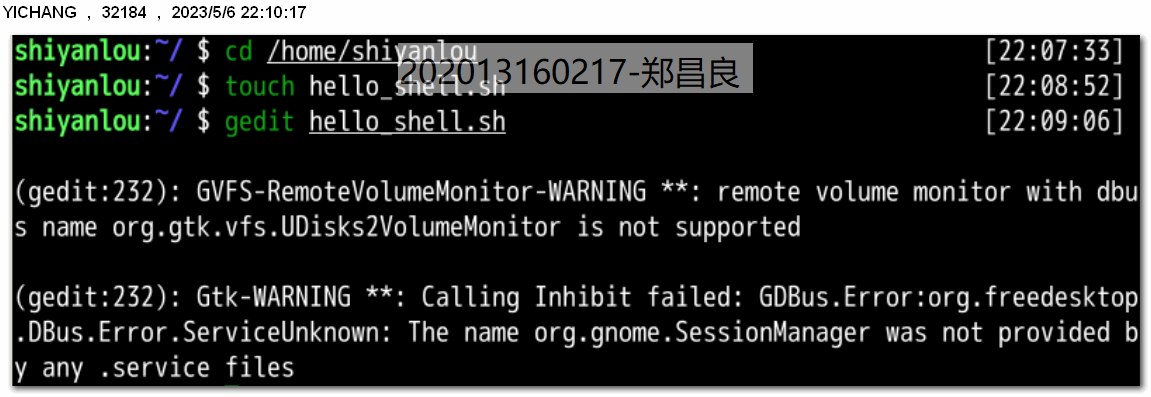
$ echo $PATH



$ cd /home/shiyanlou

$ touch hello\_shell.sh

$ gedit hello\_shell.sh

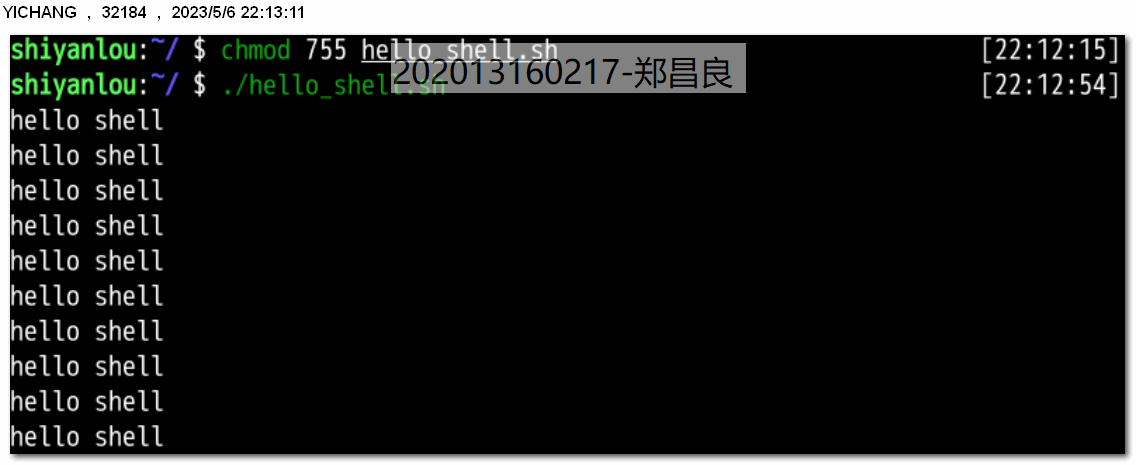


为文件添加可执行权限，否则执行会报错没有权限：

$ chmod 755 hello\_shell.sh

$ cd /home/shiyanlou

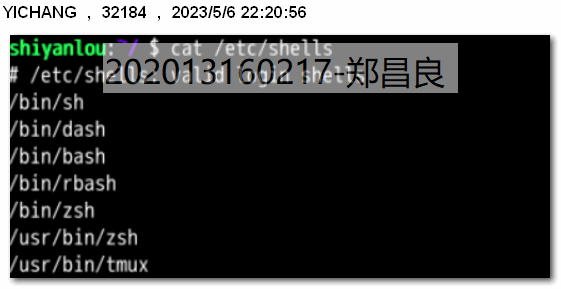
$ ./hello\_shell.sh



### 4、添加自定义路径到“PATH”环境变量

$ PATH=$PATH:/home/shiyanlou/mybin

可以使用 cat /etc/shells 命令查看当前系统已安装的 Shell。



$ echo "PATH=$PATH:/home/shiyanlou/mybin" >> .zshrc

### 修改和删除已有变量

比如我们可以修改前面添加到 PATH 的环境变量，将添加的 mybin 目录从环境变量里删除。为了避免操作失误导致命令找不到，我们先将 PATH 赋值给一个新的自定义变量 mypath：

$ mypath=$PATH

$ echo $mypath

$ mypath=${mypath%/home/shiyanlou/mybin}

# 或使用通配符,\*表示任意多个任意字符

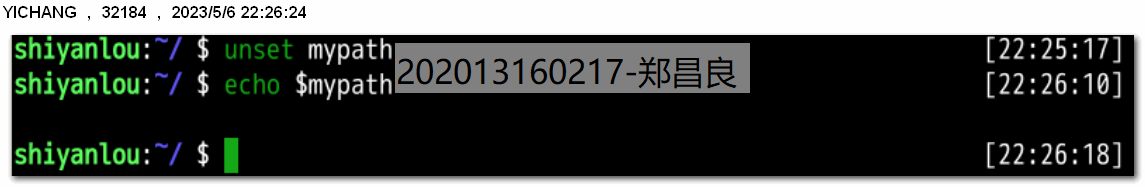
$ mypath=${mypath%\*/mybin}



删除变量

可以使用 unset 命令删除一个环境变量：

$ unset mypath



### 让环境变量立即生效

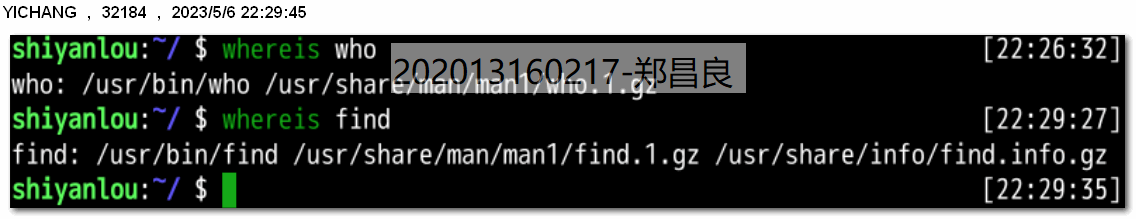
$ cd /home/shiyanlou

$ source .zshrc

### 搜索文件

$ whereis who

$ whereis find



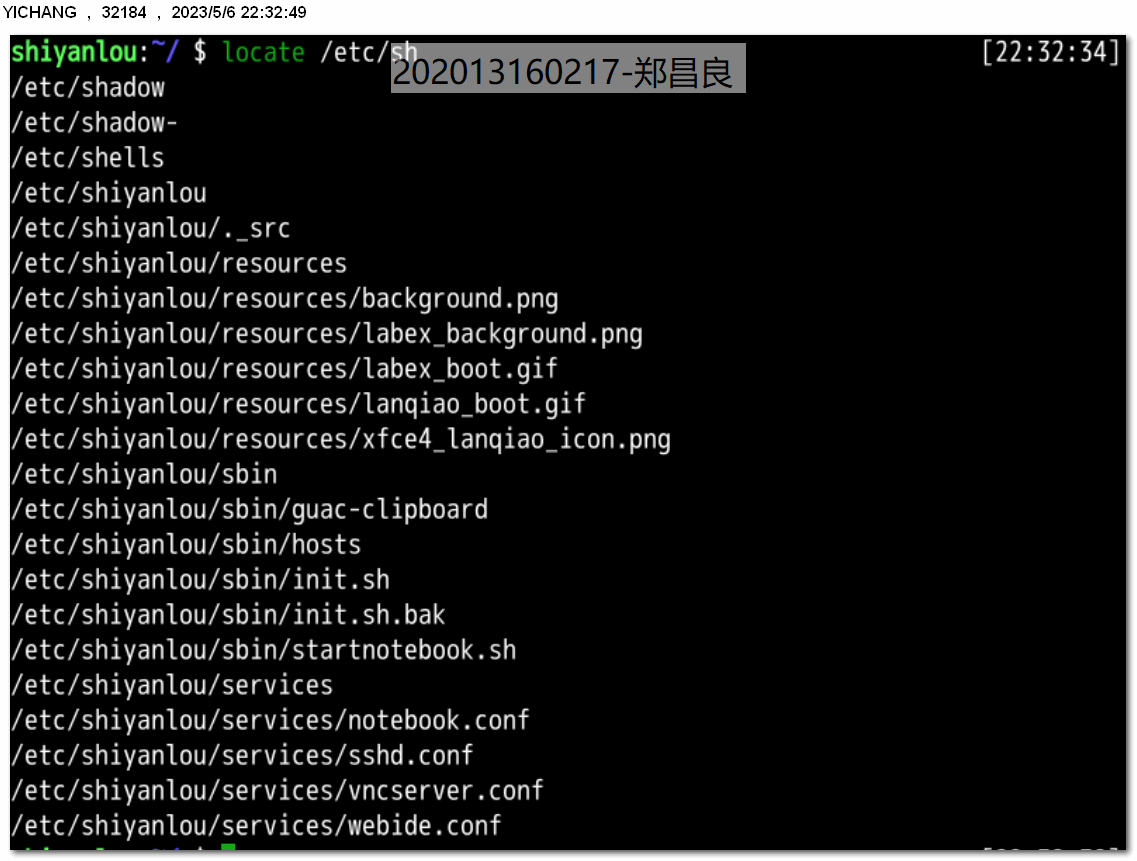
$ sudo apt-get update

$ sudo apt-get install locate

$ sudo updatedb

locate可以用来查找指定目录下的不同文件类型，如查找 /etc 下所有以 sh 开头的文件：

$ locate /etc/sh

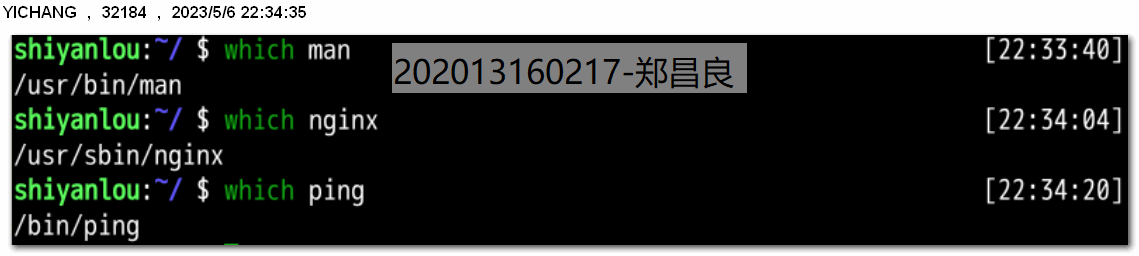


使用 which 来确定是否安装了某个指定的程序，因为它只从 PATH 环境变量指定的路径中去搜索命令并且返回第一个搜索到的结果。也就是说，我们可以看到某个系统命令是否存在以及执行的到底是哪一个地方的命令。

$ which man

$ which nginx

$ which ping



这条命令表示去 /etc/ 目录下面 ，搜索名字叫做 interfaces 的文件或者目录。

$ sudo find /etc/ -name interfaces



列出 home 目录中，当天（24 小时之内）有改动的文件：

$ find ~ -mtime 0



# 实验六 文件打包与解压缩

## 实验内容

介绍 Linux 上常用的压缩/解压工具，主要讲解 zip，tar 的使用

## 实验目的

1、掌握zip 命令

2、掌握tar 命令

3、掌握压缩与解压常用组合

## 实验知识点

1、zip 命令

2、tar 命令

3、压缩与解压常用组合

## 四、实验过程

### 1、zip压缩打包程序

1. 使用 zip 打包文件夹：

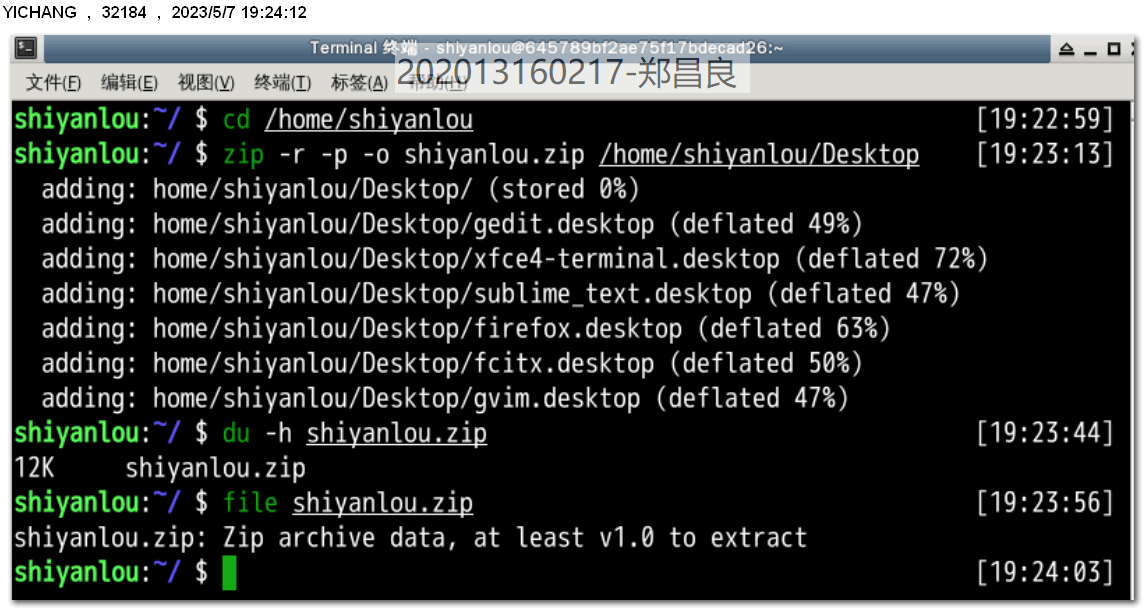
$ cd /home/shiyanlou

$ zip -r -q -o shiyanlou.zip /home/shiyanlou/Desktop

$ du -h shiyanlou.zip

$ file shiyanlou.zip

上面命令将目录 /home/shiyanlou/Desktop 打包成一个文件，并查看了打包后文件的大小和类型。第一行命令中，-r 参数表示递归打包包含子目录的全部内容，-q 参数表示为安静模式，即不向屏幕输出信息，-o，表示输出文件，需在其后紧跟打包输出文件名。



2. 设置压缩级别为 9 和 1（9 最大，1 最小），重新打包：

$ zip -r -9 -q -o shiyanlou\_9.zip /home/shiyanlou/Desktop -x ~/\*.zip

$ zip -r -1 -q -o shiyanlou\_1.zip /home/shiyanlou/Desktop -x ~/\*.zip

添加了一个参数用于设置压缩级别 -[1-9]，1 表示最快压缩但体积大，9 表示体积最小但耗时最久。最后那个 -x 是为了排除我们上一次创建的 zip 文件，否则又会被打包进这一次的压缩文件中，注意：这里只能使用绝对路径，否则不起作用。

用 du 命令分别查看默认压缩级别、最低、最高压缩级别及未压缩的文件的大小：

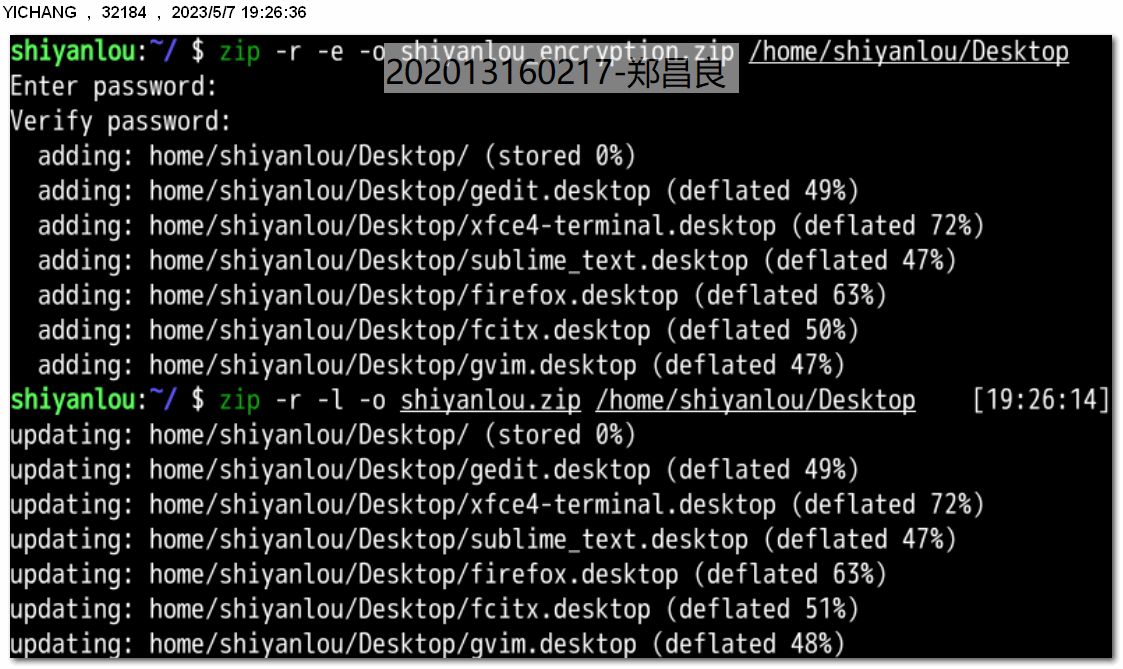
$ du -h -d 0 \*.zip ~ | sort



1. 创建加密 zip 包

$ zip -r -e -o shiyanlou\_encryption.zip /home/shiyanlou/Desktop

$ zip -r -l -o shiyanlou.zip /home/shiyanlou/Desktop



### 2、unzip 命令解压缩zip文件

将 shiyanlou.zip 解压到当前目录：

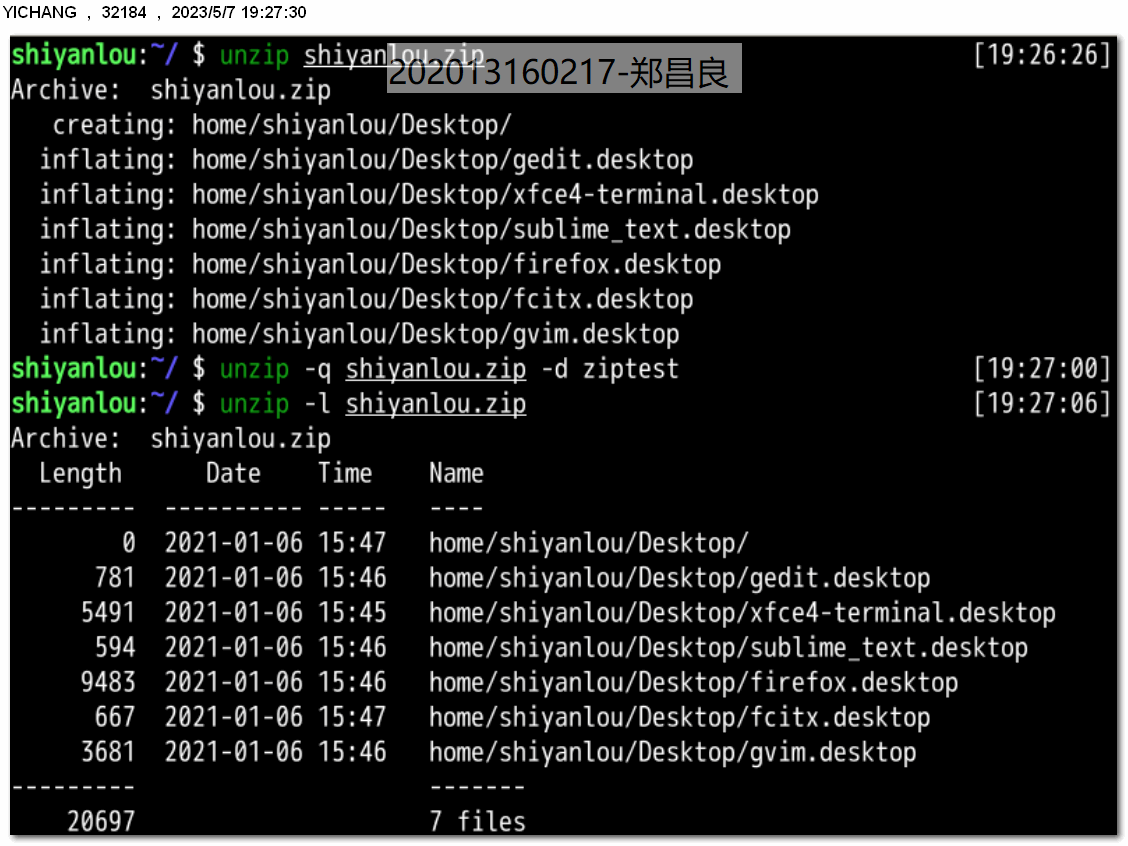
$ unzip shiyanlou.zip

使用安静模式，将文件解压到指定目录：

$ unzip -q shiyanlou.zip -d ziptest

上述指定目录不存在，将会自动创建。如果你不想解压只想查看压缩包的内容你可以使用 -l 参数：

$ unzip -l shiyanlou.zip



### 3、tar打包工具

1. 创建一个 tar 包：

$ cd /home/shiyanlou

$ tar -P -cf shiyanlou.tar /home/shiyanlou/Desktop

2、解包一个文件（-x 参数）到指定路径的已存在目录（-C 参数）：

$ mkdir tardir

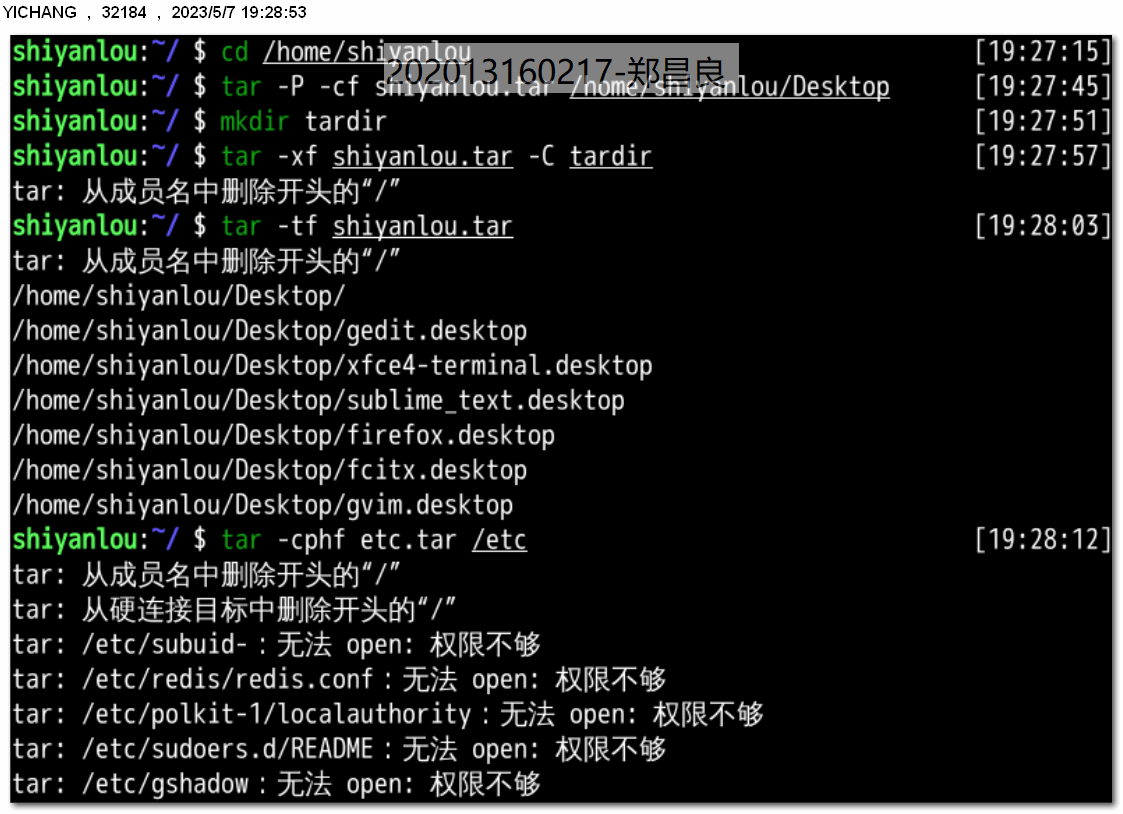
$ tar -xf shiyanlou.tar -C tardir

3、只查看不解包文件 -t 参数：

$ tar -tf shiyanlou.tar

4、保留文件属性和跟随链接（符号链接或软链接），有时候我们使用 tar 备份文件当你在其他主机还原时希望保留文件的属性（-p 参数）和备份链接指向的源文件而不是链接本身（-h 参数）：

$ tar -cphf etc.tar /etc



5、只需要在创建 tar 文件的基础上添加 -z 参数，使用 gzip 来压缩文件：

$ tar -czf shiyanlou.tar.gz /home/shiyanlou/Desktop

6、解压 \*.tar.gz 文件：

$ tar -xzf shiyanlou.tar.gz



# 实验七 文件系统操作与磁盘管理

## 实验内容

包含几个命令的几个参数的讲解

## 实验目的

1、掌握df，du，mount命令的使用

2、磁盘相关知识学习

## 三、实验知识点

1、df，du，mount命令的使用

2、磁盘相关知识学习

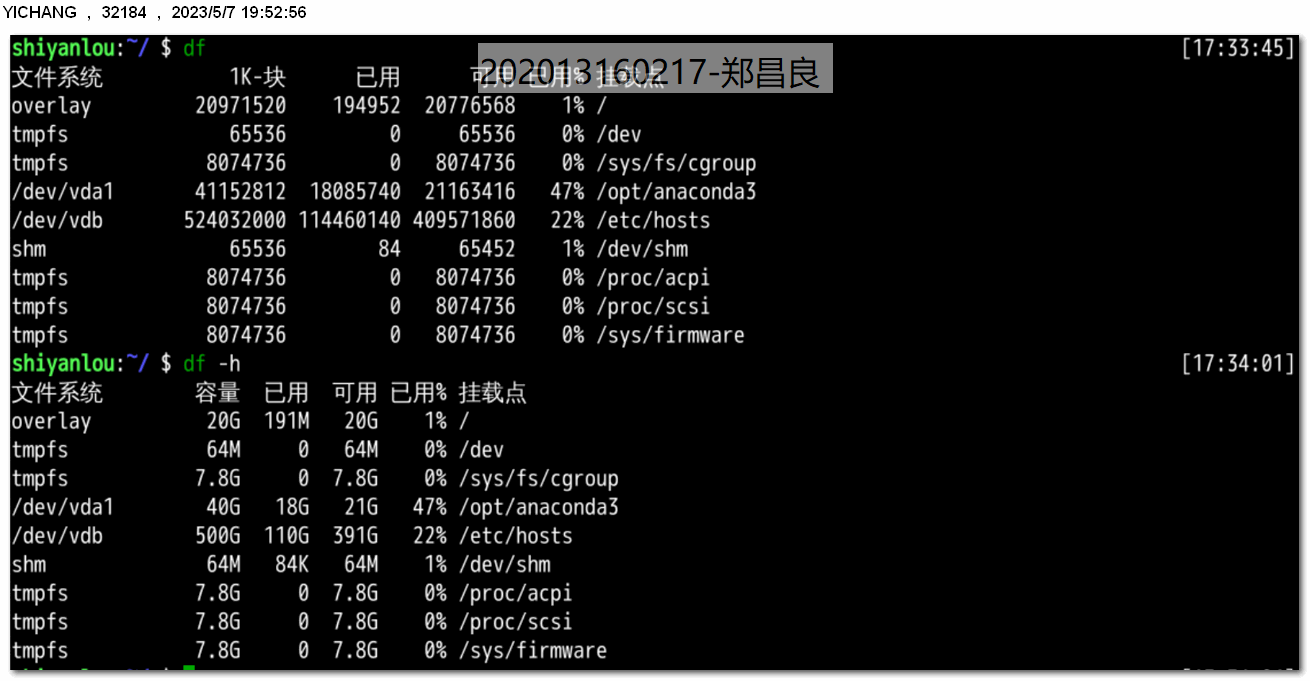
## 四、实验过程

### 1、基本操作

1. 查看磁盘和目录的容量

$ df

$ df -h



2、使用 du 命令查看目录的容量

-d参数指定查看目录的深度

$ du -h -d 0 ~

$ du -h -d 1 ~

常用参数

du -h #同--human-readable 以K，M，G为单位，提高信息的可读性。

du -a #同--all 显示目录中所有文件的大小。

du -s #同--summarize 仅显示总计，只列出最后加总的值。



3、创建虚拟磁盘

dd命令用于转换和复制文件，不过它的复制不同于cp。之前提到过关于 Linux 的很重要的一点，一切即文件，在 Linux 上，硬件的设备驱动（如硬盘）和特殊设备文件（如/dev/zero和/dev/random）都像普通文件一样，只是在各自的驱动程序中实现了对应的功能，dd 也可以读取文件或写入这些文件。这样，dd也可以用在备份硬件的引导扇区、获取一定数量的随机数据或者空数据等任务中。dd程序也可以在复制时处理数据，例如转换字节序、或在 ASCII 与 EBCDIC 编码间互换。

dd的命令行语句与其他的 Linux 程序不同，因为它的命令行选项格式为选项=值，而不是更标准的--选项 值或-选项=值。dd默认从标准输入中读取，并写入到标准输出中，但可以用选项if（input file，输入文件）和of（output file，输出文件）改变。

# 输出到文件

$ dd of=test bs=10 count=1 # 或者 dd if=/dev/stdin of=test bs=10 count=1

# 输出到标准输出

$ dd if=/dev/stdin of=/dev/stdout bs=10 count=1

将输出的英文字符转换为大写再写入文件：

$ dd if=/dev/stdin of=test bs=10 count=1 conv=ucase

使用 dd 命令创建虚拟镜像文件

$ dd if=/dev/zero of=virtual.img bs=1M count=256

$ du -h virtual.img



使用 mkfs 命令格式化磁盘（我们这里是自己创建的虚拟磁盘镜像）

使用下面的命令来将我们的虚拟磁盘镜像格式化为ext4文件系统：

$ sudo mkfs.ext4 virtual.img



使用 mount 命令挂载磁盘到目录树

$ sudo mount



$ mount -o loop -t ext4 virtual.img /mnt

# 也可以省略挂载类型，很多时候 mount 会自动识别

# 以只读方式挂载

$ mount -o loop --ro virtual.img /mnt

# 或者mount -o loop,ro virtual.img /mnt

# 实验八 Linux下的帮助命令

## 实验内容

我们时常使用 Linux 的命令，一些常用的命令即使不背我们也能记住，而一些较长的命令或是一些不常用的参数甚至是不常用的命令需要用时怎么也想不出来。本课程将带你好好利用 Linux 系统自带的帮助工具与文档。

## 二、实验目的

1、掌握内建命令与外部命令之分

2、学会help , man , info 命令的使用以及区别

## 实验知识点

1、内建命令与外部命令之分

2、help , man , info 命令的使用以及区别

## 四、实验过程

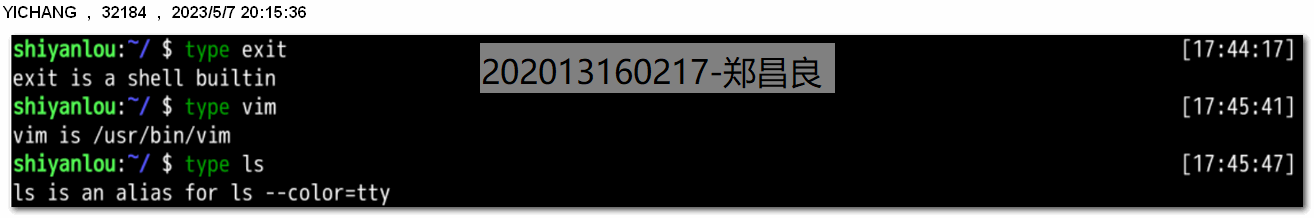
### 1、内建命令与外部命令

内建命令实际上是 shell 程序的一部分，其中包含的是一些比较简单的 Linux 系统命令，这些命令是写在 bash 源码的 builtins 里面的，由 shell 程序识别并在 shell 程序内部完成运行，通常在 Linux 系统加载运行时 shell 就被加载并驻留在系统内存中。而且解析内部命令 shell 不需要创建子进程，因此其执行速度比外部命令快。比如：history、cd、exit 等等。

外部命令是 Linux 系统中的实用程序部分，因为实用程序的功能通常都比较强大，所以其包含的程序量也会很大，在系统加载时并不随系统一起被加载到内存中，而是在需要时才将其调入内存。虽然其不包含在 shell 中，但是其命令执行过程是由 shell 程序控制的。外部命令是在 Bash 之外额外安装的，通常放在/bin，/usr/bin，/sbin，/usr/sbin 等等。比如：ls、vi 等。

type exit

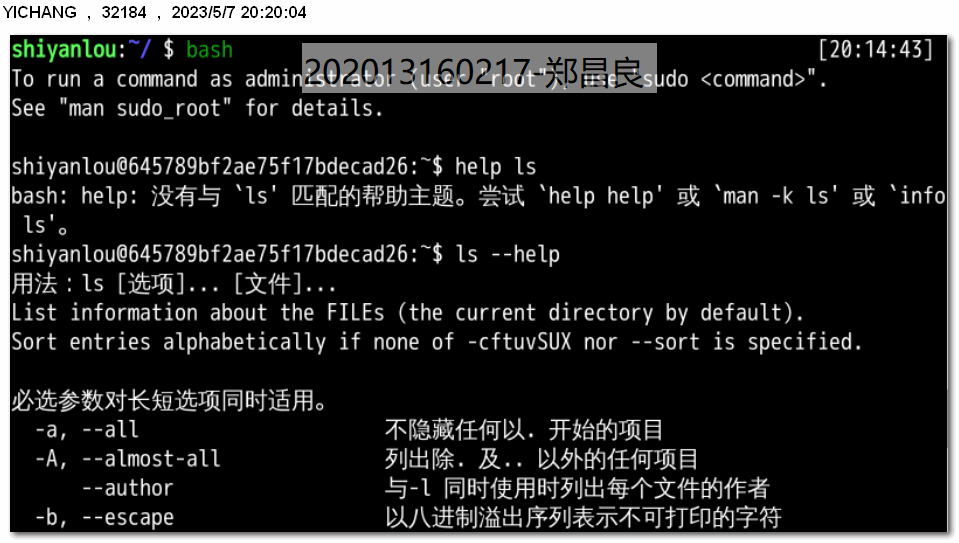
type vim



### 2、帮助命令的使用

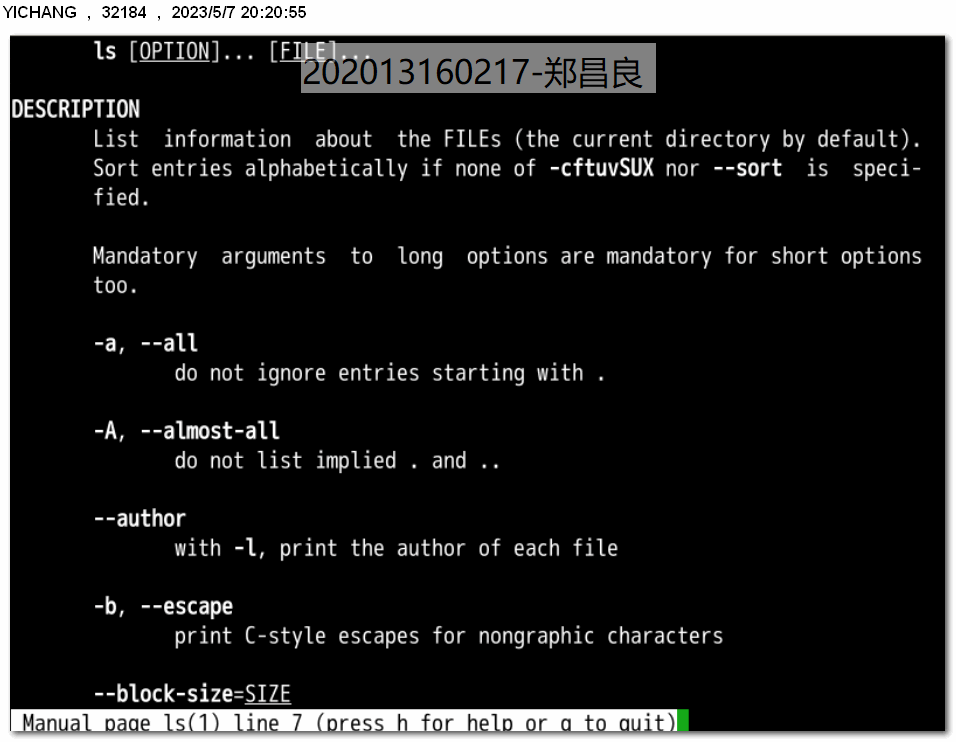
help ls

ls --help



man ls

man 　没有内建与外部命令的区分，因为 man 工具是显示系统手册页中的内容，也就是一本电子版的字典，这些内容大多数都是对命令的解释信息，还有一些相关的描述。通过查看系统文档中的 man 也可以得到程序的更多相关信息和 Linux 的更多特性。



### 2、info命令

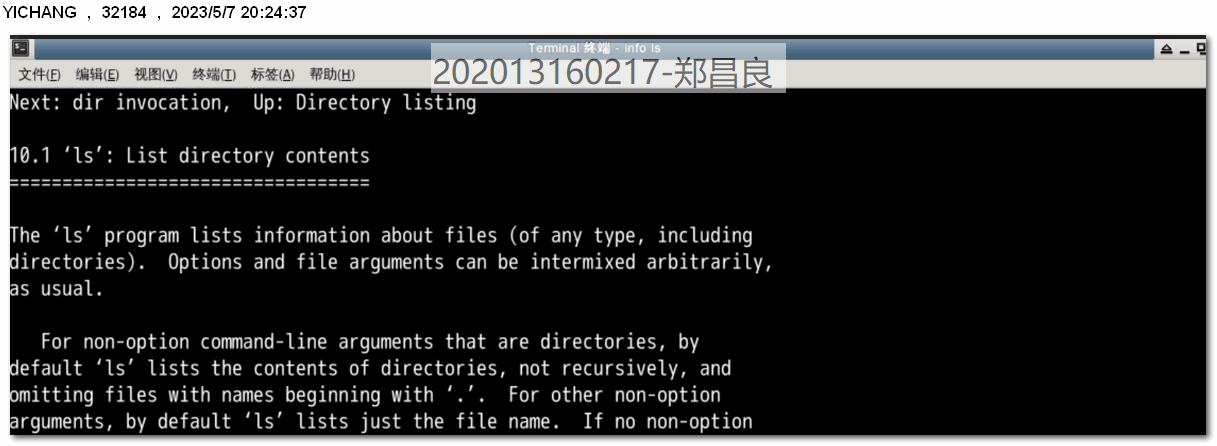
# 安装 info

$ sudo apt-get update

$ sudo apt-get install info

# 查看 ls 命令的 info

$ info ls



# 实验九 Linux任务计划crontab

## 实验内容

我们时常会有一些定期定时的任务，如周期性的清理一下／tmp，周期性的去备份一次数据库，周期性的分析日志等等。而且有时候因为某些因素的限制，执行该任务的时间会很尴尬。本课程将带你很好的利用 Linux 系统的计划工具

## 实验目的

1、掌握crontab 语法

## 三、实验知识点

1、crontab 语法

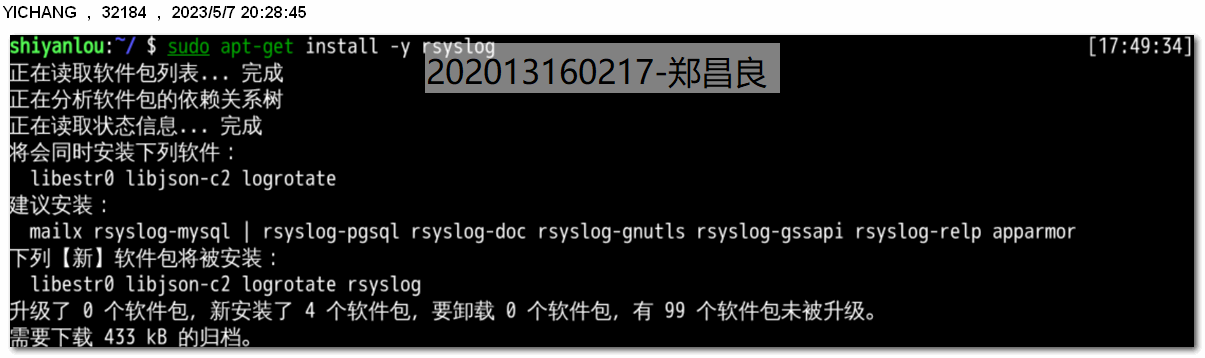
## 四、实验过程

### 1、crontab准备

sudo apt-get install -y rsyslog

sudo service rsyslog start

sudo cron －f &





### 2、crontab使用

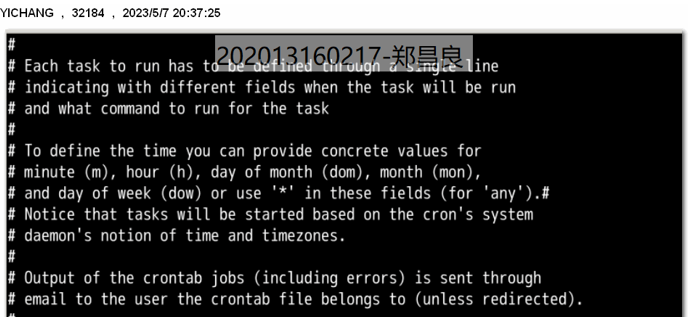
通过下面一个命令来添加一个计划任务

crontab -e



crontab -l





虽然我们添加了任务，但是如果 cron 的守护进程并没有启动，它根本都不会监测到有任务，当然也就不会帮我们执行，我们可以通过以下 2 种方式来确定我们的 cron 是否成功的在后台启动，默默的帮我们做事，若是没有就得执行上文准备中的第二步了

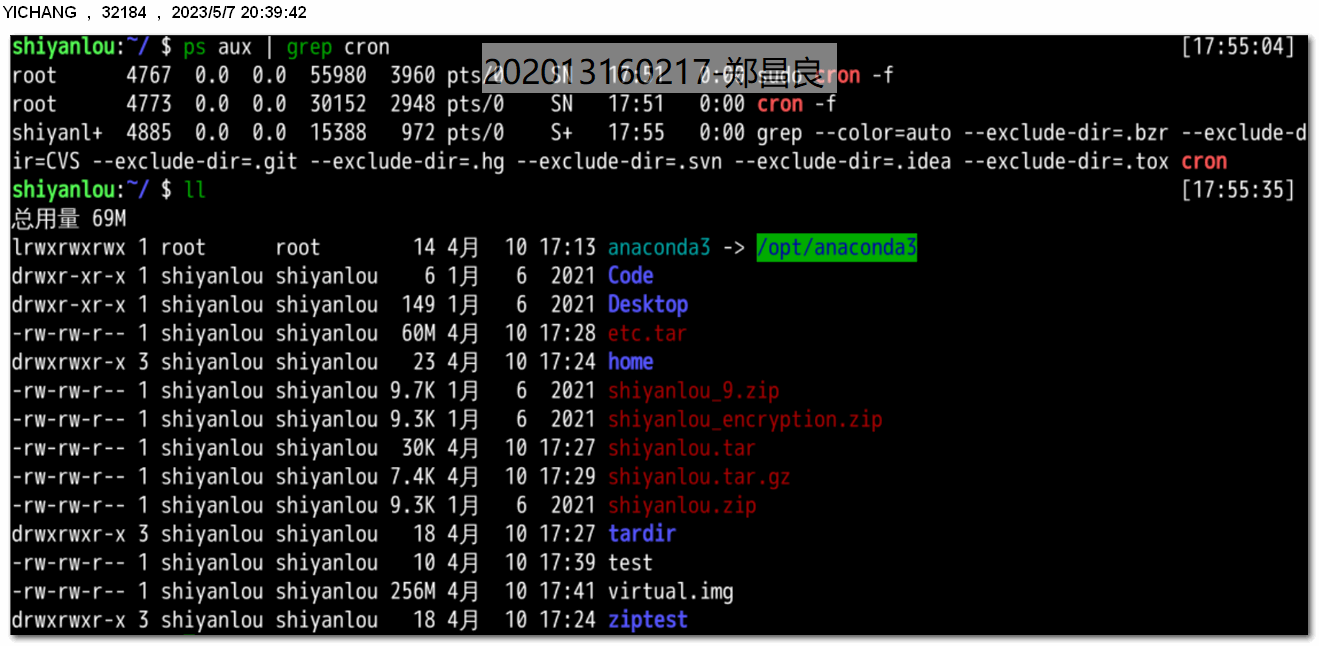
ps aux | grep cron

or

pgrep cron

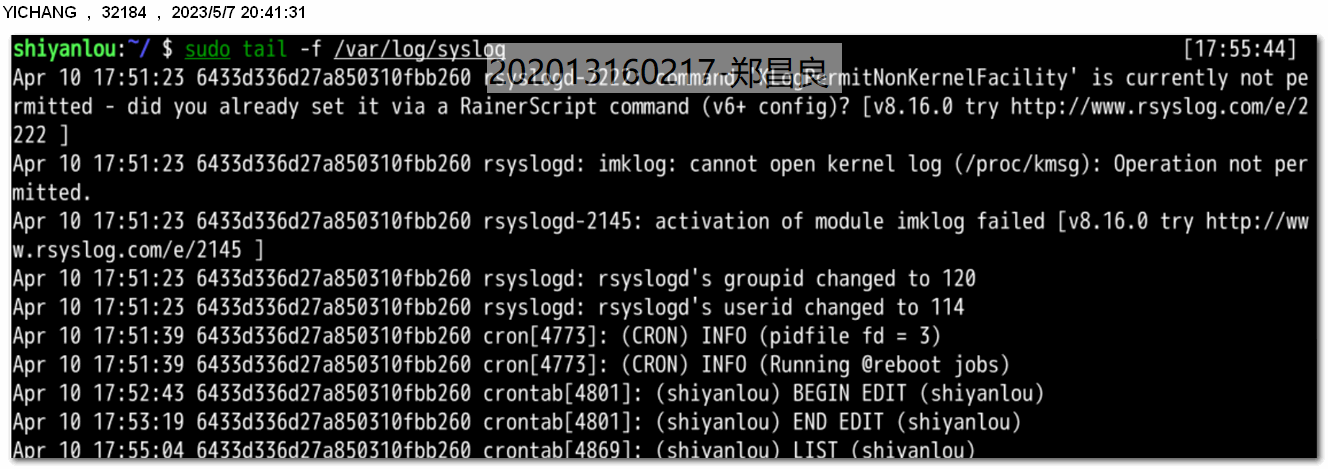
通过下图可以看到任务在创建之后，执行了几次，生成了一些文件，且每分钟生成一个：

ll



我们通过这样一个命令可以查看到执行任务命令之后在日志中的信息反馈

sudo tail -f /var/log/syslog



当我们并不需要这个任务的时候我们可以使用这么一个命令去删除任务

crontab -r

# 实验十 命令执行顺序控制与管道

## 实验内容

顺序执行、选择执行、管道、cut 命令、grep 命令、wc 命令、sort 命令等，高效率使用 Linux 的技巧。

## 实验目的

1、掌握cut，grep，wc，sort命令的使用

2、加深管道的理解

## 实验知识点

1、cut，grep，wc，sort命令的使用

2、管道的理解

## 四、实验过程

### 1、顺序执行多条命令

当我们需要使用apt-get安装一个软件，然后安装完成后立即运行安装的软件（或命令工具），又恰巧你的主机才更换的软件源还没有更新软件列表（比如之前我们的环境中，每次重新开始实验就得 sudo apt-get update，现在已经没有这个问题了）,那么你可能会有如下一系列操作：

$ sudo apt-get update

# 等待——————————然后输入下面的命令

$ sudo apt-get install some-tool //这里some-tool是指具体的软件包

# 等待——————————然后输入下面的命令

$ some-tool

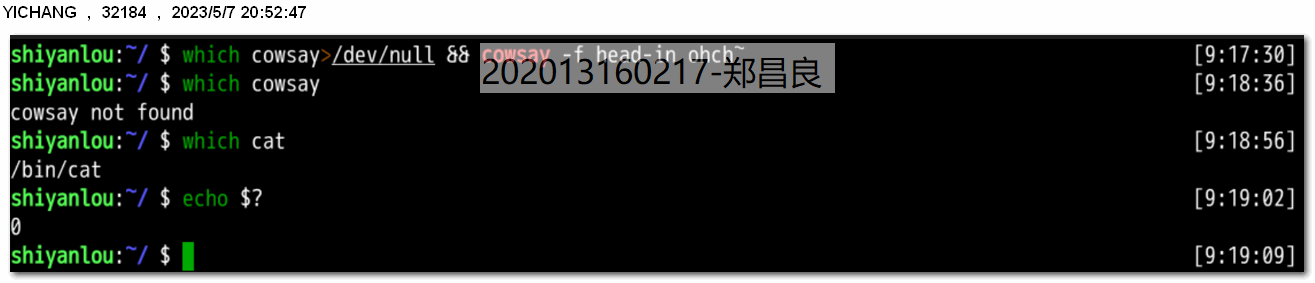
简单的顺序执行你可以使用;来完成，比如上述操作你可以：

$ sudo apt-get update;sudo apt-get install some-tool;some-tool

### 2、有选择的执行命令

$ which cowsay>/dev/null && cowsay -f head-in ohch~

&&就是用来实现选择性执行的，它表示如果前面的命令执行结果（不是表示终端输出的内容，而是表示命令执行状态的结果）返回 0 则执行后面的，否则不执行，你可以从$?环境变量获取上一次命令的返回结果：



||在这里就是与&&相反的控制效果，当上一条命令执行结果为 ≠0($?≠0)时则执行它后面的命令：

$ which cowsay>/dev/null || echo "cowsay has not been install, please run 'sudo apt-get install cowsay' to install"

除了上述基本的使用之外，我们还可以结合着&&和||来实现一些操作，比如：

$ which cowsay>/dev/null && echo "exist" || echo "not exist"



### 3、管道试用

先试用一下管道，比如查看/etc目录下有哪些文件和目录，使用ls命令来查看：

$ ls -al /etc



有太多内容，屏幕不能完全显示，这时候可以使用滚动条或快捷键滚动窗口来查看。

### 4、cut 命令，打印每一行的某一字段

打印/etc/passwd文件中以:为分隔符的第 1 个字段和第 6 个字段分别表示用户名和其家目录：

$ cut /etc/passwd -d ':' -f 1,6

# 前五个（包含第五个）

$ cut /etc/passwd -c -5

# 前五个之后的（包含第五个）

$ cut /etc/passwd -c 5-

# 第五个

$ cut /etc/passwd -c 5

# 2到5之间的（包含第五个）

$ cut /etc/passwd -c 2-5





### 5、grep 命令，在文本中或 stdin 中查找匹配字符串

搜索/home/shiyanlou目录下所有包含"shiyanlou"的文本文件，并显示出现在文本中的行号：

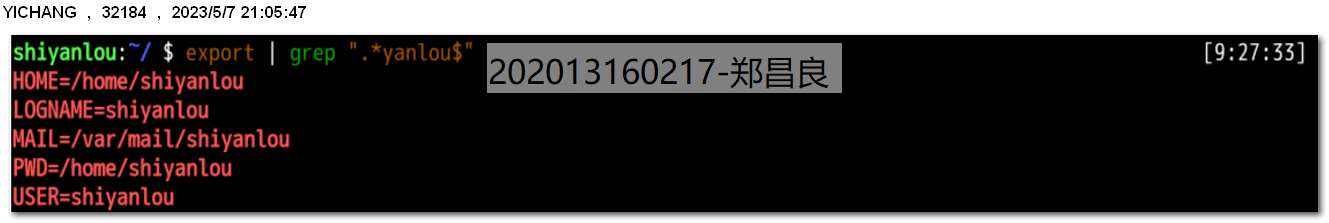
$ grep -rnI "shiyanlou" ~



-r 参数表示递归搜索子目录中的文件,-n表示打印匹配项行号，-I表示忽略二进制文件。

# 查看环境变量中以"yanlou"结尾的字符串

$ export | grep ".\*yanlou$"



### 6、wc 命令，简单小巧的计数工具

wc 命令用于统计并输出一个文件中行、单词和字节的数目，比如输出/etc/passwd文件的统计信息：

$ wc /etc/passwd

# 行数

$ wc -l /etc/passwd

# 单词数

$ wc -w /etc/passwd

# 字节数

$ wc -c /etc/passwd

# 字符数

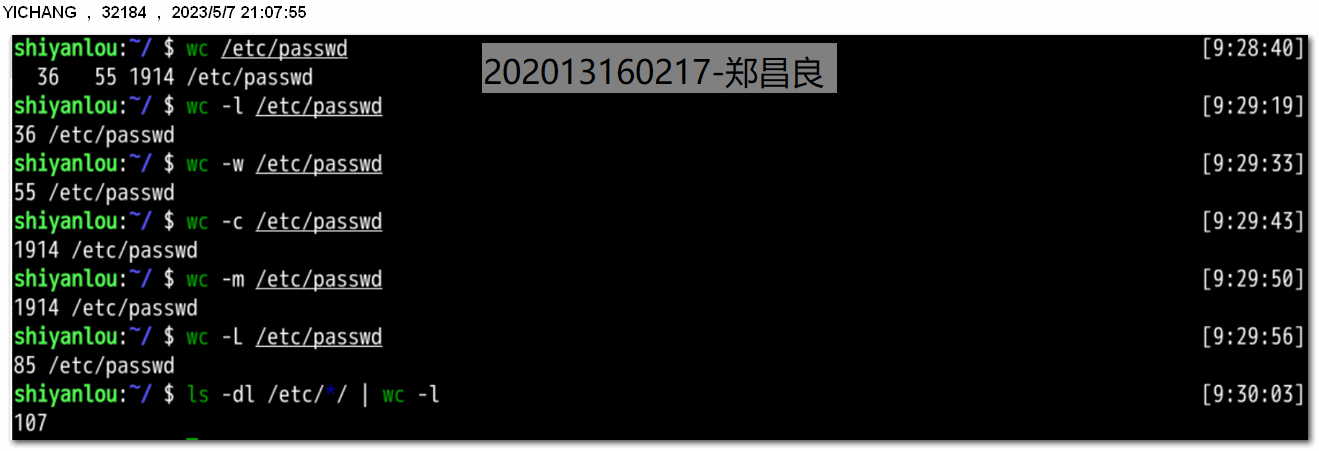
$ wc -m /etc/passwd

# 最长行字节数

$ wc -L /etc/passwd

再来结合管道来操作一下，下面统计 /etc 下面所有目录数：

$ ls -dl /etc/\*/ | wc -l



### 7、sort排序命令

# 默认为字典排序：

$ cat /etc/passwd | sort

反转排序：

$ cat /etc/passwd | sort -r

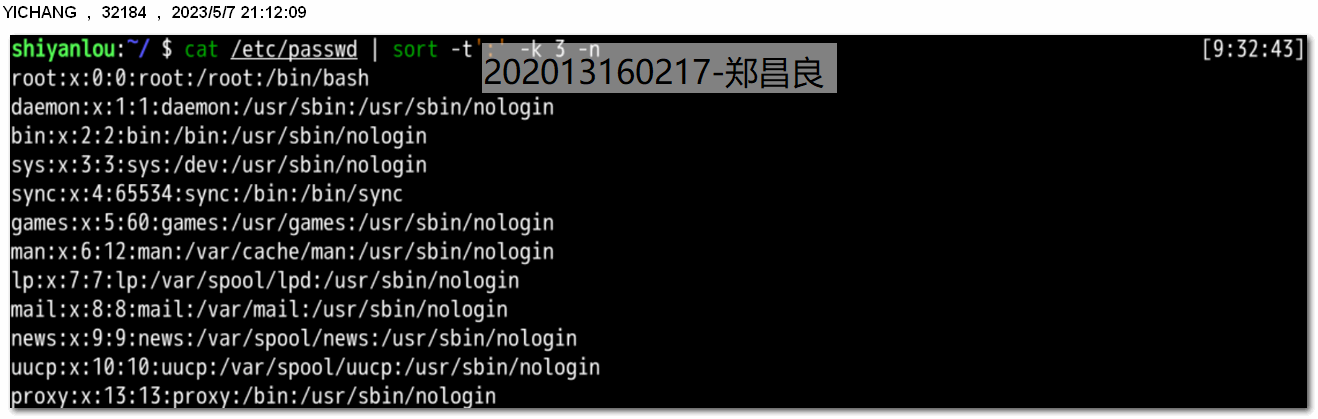
按特定字段排序：

$ cat /etc/passwd | sort -t':' -k 3



上面的-t参数用于指定字段的分隔符，这里是以":"作为分隔符；-k 字段号用于指定对哪一个字段进行排序。这里/etc/passwd文件的第三个字段为数字，默认情况下是以字典序排序的，如果要按照数字排序就要加上-n参数：

$ cat /etc/passwd | sort -t':' -k 3 -n



8、uniq去重命令

过滤重复行：

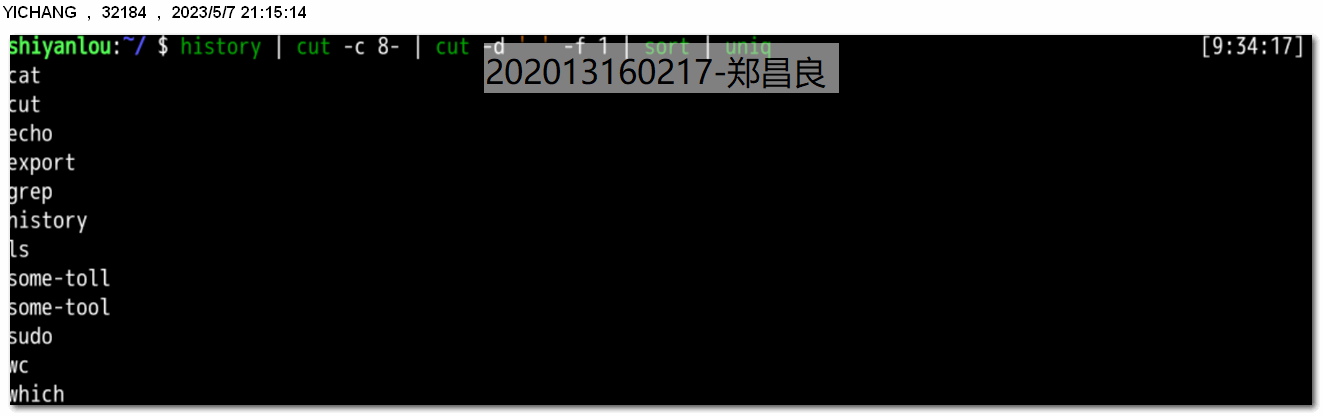
使用history命令查看最近执行过的命令（实际为读取${SHELL}\_history 文件,如我们环境中的~/.zsh\_history 文件），不过你可能只想查看使用了哪个命令而不需要知道具体干了什么，那么你可能就会要想去掉命令后面的参数然后去掉重复的命令：

$ history | cut -c 8- | cut -d ' ' -f 1 | uniq

因为uniq命令只能去连续重复的行，不是全文去重，所以要达到预期效果，我们先排序：

$ history | cut -c 8- | cut -d ' ' -f 1 | sort | uniq

# 或者$ history | cut -c 8- | cut -d ' ' -f 1 | sort -u



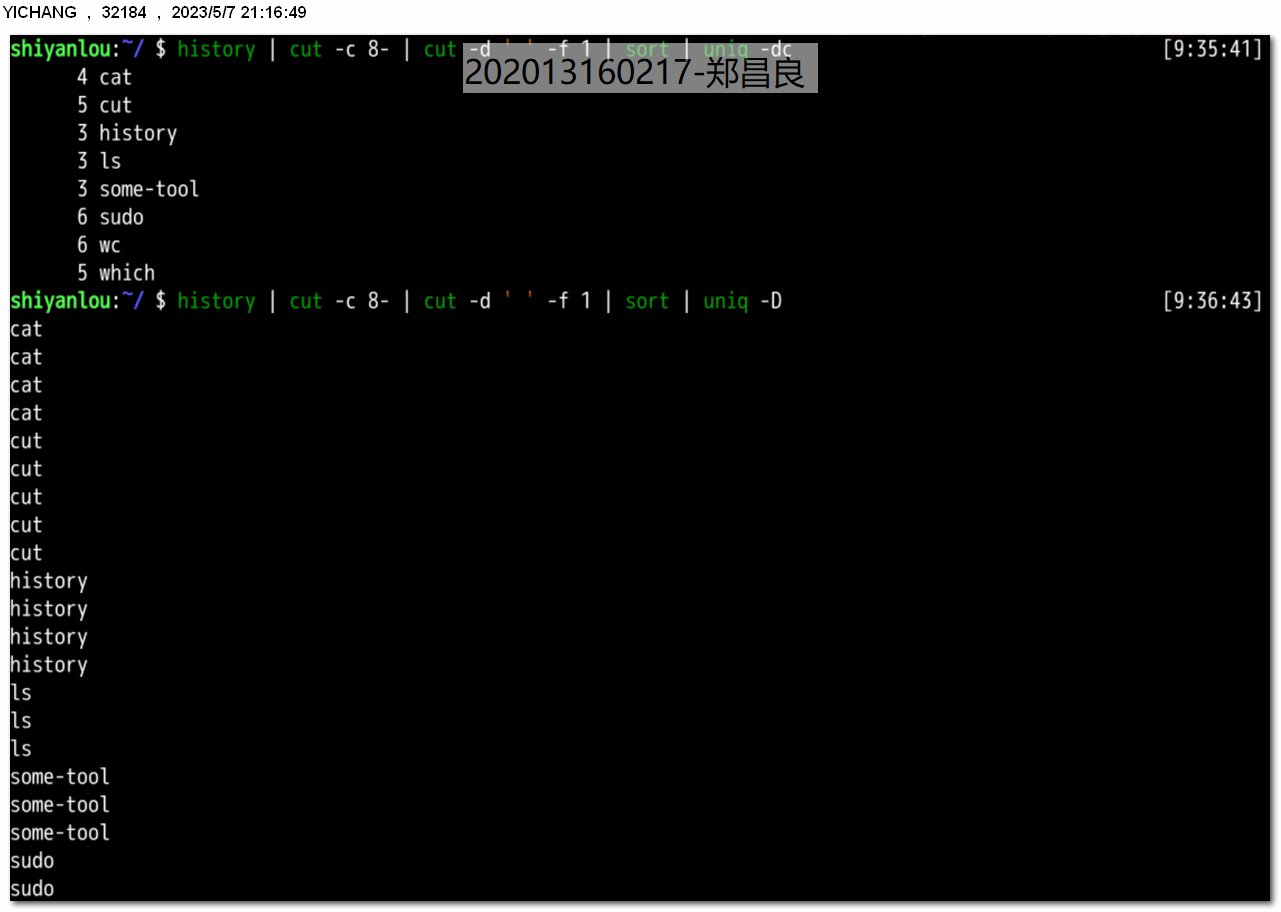
输出重复行：

# 输出重复过的行（重复的只输出一个）及重复次数

$ history | cut -c 8- | cut -d ' ' -f 1 | sort | uniq -dc

# 输出所有重复的行

$ history | cut -c 8- | cut -d ' ' -f 1 | sort | uniq -D



# 实验十一 简单文本处理

## 实验内容

练习tr（注意不是 tar），col，join，paste。

## 实验目的

1、学会使用常见文本处理命令

2、学会如何简单处理文本

## 三、实验知识点

1、常见文本处理命令

2、如何简单处理文本

## 四、实验过程

### 1、tr命令

tr 命令可以用来删除一段文本信息中的某些文字。或者将其进行转换。

# 删除 "hello shiyanlou" 中所有的'o','l','h'

$ echo 'hello shiyanlou' | tr -d 'olh'

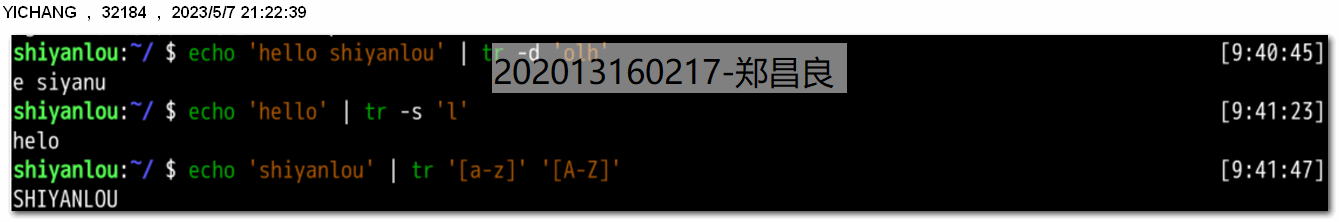
# 将"hello" 中的ll,去重为一个l

$ echo 'hello' | tr -s 'l'

# 将输入文本，全部转换为大写或小写输出

$ echo 'input some text here' | tr '[:lower:]' '[:upper:]'

# 上面的'[:lower:]' '[:upper:]'你也可以简单的写作'[a-z]' '[A-Z]',当然反过来将大写变小写也是可以的



### 2、col命令

# 查看 /etc/protocols 中的不可见字符，可以看到很多 ^I ，这其实就是 Tab 转义成可见字符的符号

$ cat -A /etc/protocols

# 使用 col -x 将 /etc/protocols 中的 Tab 转换为空格,然后再使用 cat 查看，你发现 ^I 不见了

$ cat /etc/protocols | col -x | cat -A





### 3、join命令

$ cd /home/shiyanlou

# 创建两个文件

$ echo '1 hello' > file1

$ echo '1 shiyanlou' > file2

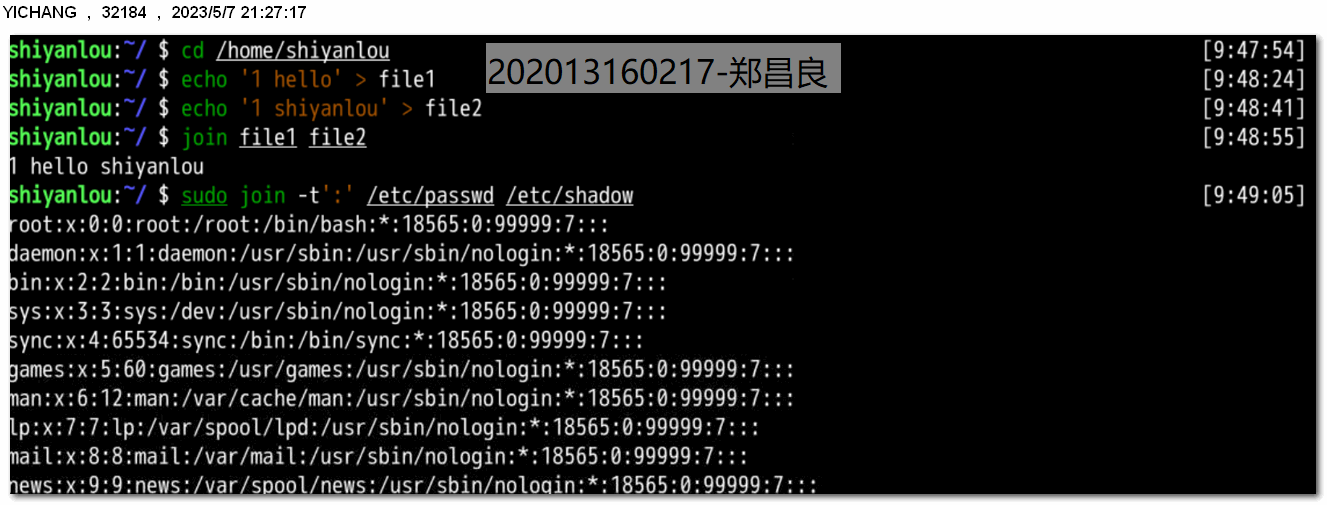
$ join file1 file2

# 将/etc/passwd与/etc/shadow两个文件合并，指定以':'作为分隔符

$ sudo join -t':' /etc/passwd /etc/shadow

# 将/etc/passwd与/etc/group两个文件合并，指定以':'作为分隔符, 分别比对第4和第3个字段

$ sudo join -t':' -1 4 /etc/passwd -2 3 /etc/group





### 4、paste命令

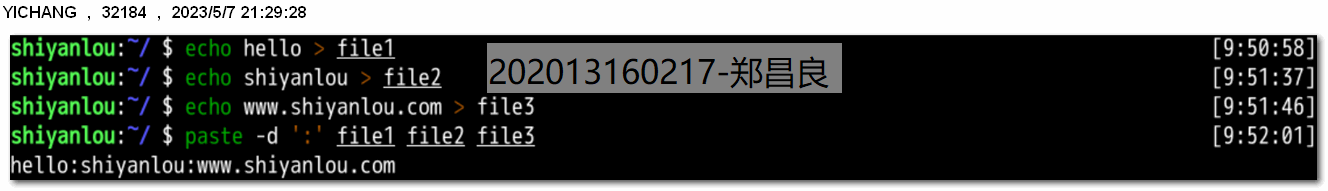
$ echo hello > file1

$ echo shiyanlou > file2

$ echo www.shiyanlou.com > file3

$ paste -d ':' file1 file2 file3

$ paste -s file1 file2 file3



# 实验十二 数据流重定向

## 实验内容

将标准输出导向一个文件或追加到一个文件中。这其实就是重定向，将原本输出到标准输出的数据重定向到一个文件中，因为标准输出(/dev/stdout)本身也是一个文件，我们将命令输出导向另一个文件自然也是没有任何问题的。

## 实验目的

1、掌握重定向怎么用

2、了解文件描述符(0,1,2)

## 三、实验知识点

1、重定向怎么用

2、文件描述符(0,1,2)

## 四、实验过程

### 1、简单的重定向

$ mkdir Documents

$ cat > Documents/test.c <<EOF

#include <stdio.h>

int main(){

printf("hello world\n");

return 0;

}

EOF

将一个文件作为命令的输入，标准输出作为命令的输出：

$ cat Documents/test.c

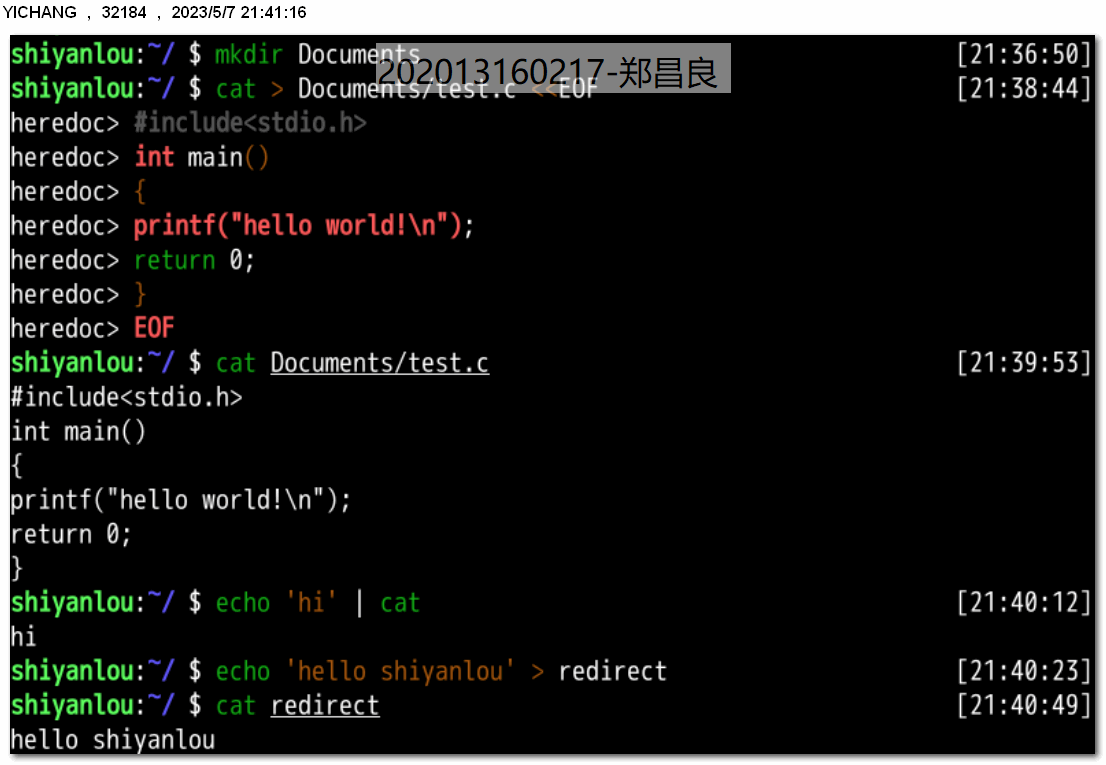
将 echo 命令通过管道传过来的数据作为 cat 命令的输入，将标准输出作为命令的输出：

$ echo 'hi' | cat

将 echo 命令的输出从默认的标准输出重定向到一个普通文件;

$ echo 'hello shiyanlou' > redirect

$ cat redirect



### 2、标准错误重定向

# 使用cat 命令同时读取两个文件，其中一个存在，另一个不存在

$ cat Documents/test.c hello.c

# 你可以看到除了正确输出了前一个文件的内容，还在末尾出现了一条错误信息

# 下面我们将输出重定向到一个文件

$ cat Documents/test.c hello.c > somefile

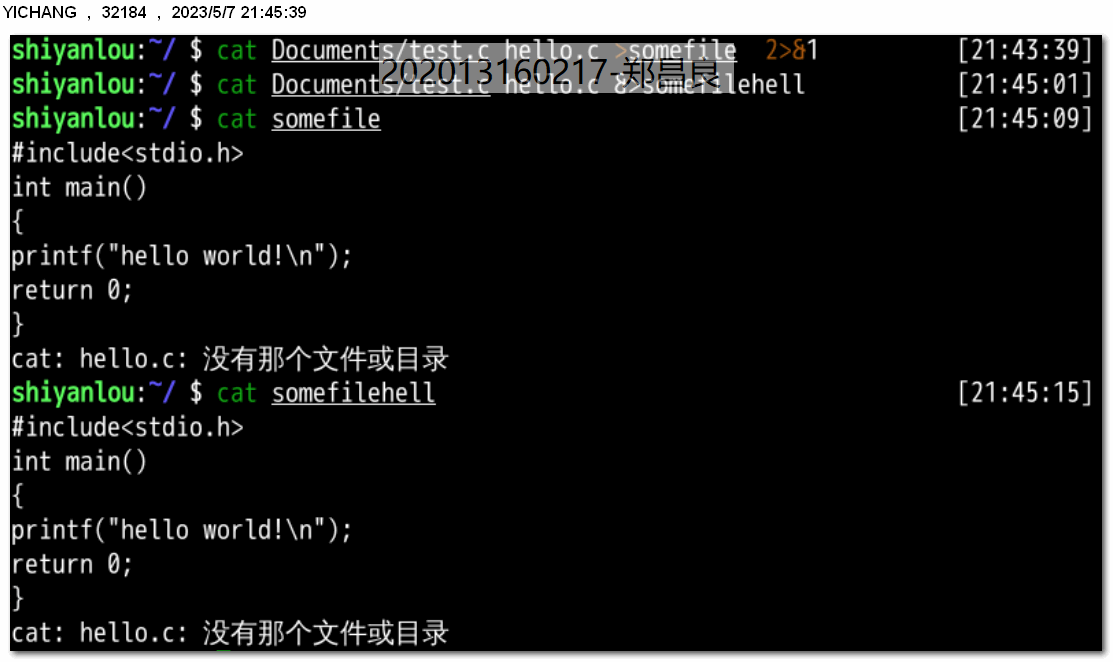


# 将标准错误重定向到标准输出，再将标准输出重定向到文件，注意要将重定向到文件写到前面

$ cat Documents/test.c hello.c >somefile 2>&1

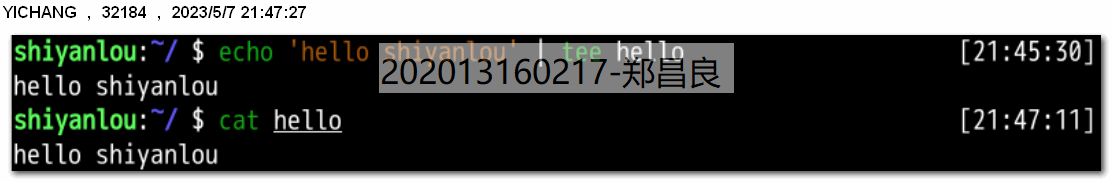
# 或者只用bash提供的特殊的重定向符号"&"将标准错误和标准输出同时重定向到文件

$ cat Documents/test.c hello.c &>somefilehell



### 3、使用tee命令同时重定向到多个文件

$ echo 'hello shiyanlou' | tee hello



### 4、永久重定向

# 先开启一个子 Shell

$ zsh

# 使用exec替换当前进程的重定向，将标准输出重定向到一个文件

$ exec 1>somefile

# 后面你执行的命令的输出都将被重定向到文件中,直到你退出当前子shell，或取消exec的重定向（后面将告诉你怎么做）

$ ls

$ exit

$ cat somefile



### 5、创建输出文件描述符

使用下面命令查看当前 Shell 进程中打开的文件描述符：

$ cd /dev/fd/;ls -Al

使用exec命令可以创建新的文件描述符：

$ zsh

$ exec 3>somefile

# 先进入目录，再查看，否则你可能不能得到正确的结果，然后再回到上一次的目录

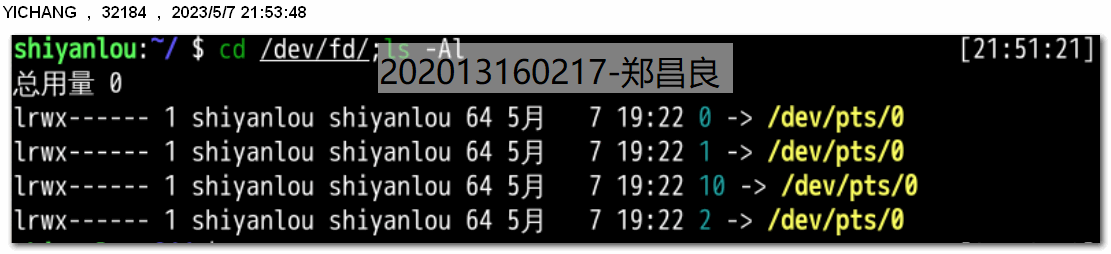
$ cd /dev/fd/;ls -Al;cd -

# 注意下面的命令>与&之间不应该有空格，如果有空格则会出错

$ echo "this is test" >&3

$ cat somefile

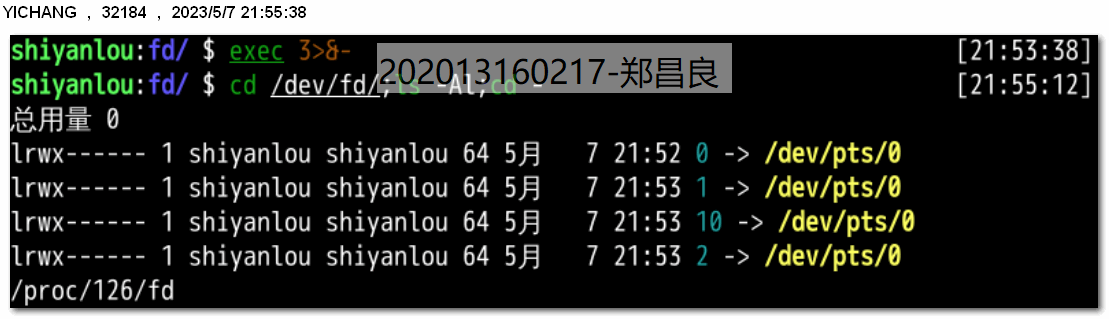
$ exit



### 6、关闭文件描述符

$ exec 3>&-

$ cd /dev/fd;ls -Al;cd -



### 完全屏蔽命令的输出

可以利用/dev/null屏蔽命令的输出：

$ cat Documents/test.c 1>/dev/null 2>&1

### 使用 xargs 分割参数列表

$ cut -d: -f1 < /etc/passwd | sort | xargs echo

上面这个命令用于将/etc/passwd文件按:分割取第一个字段排序后，使用echo命令生成一个列表。