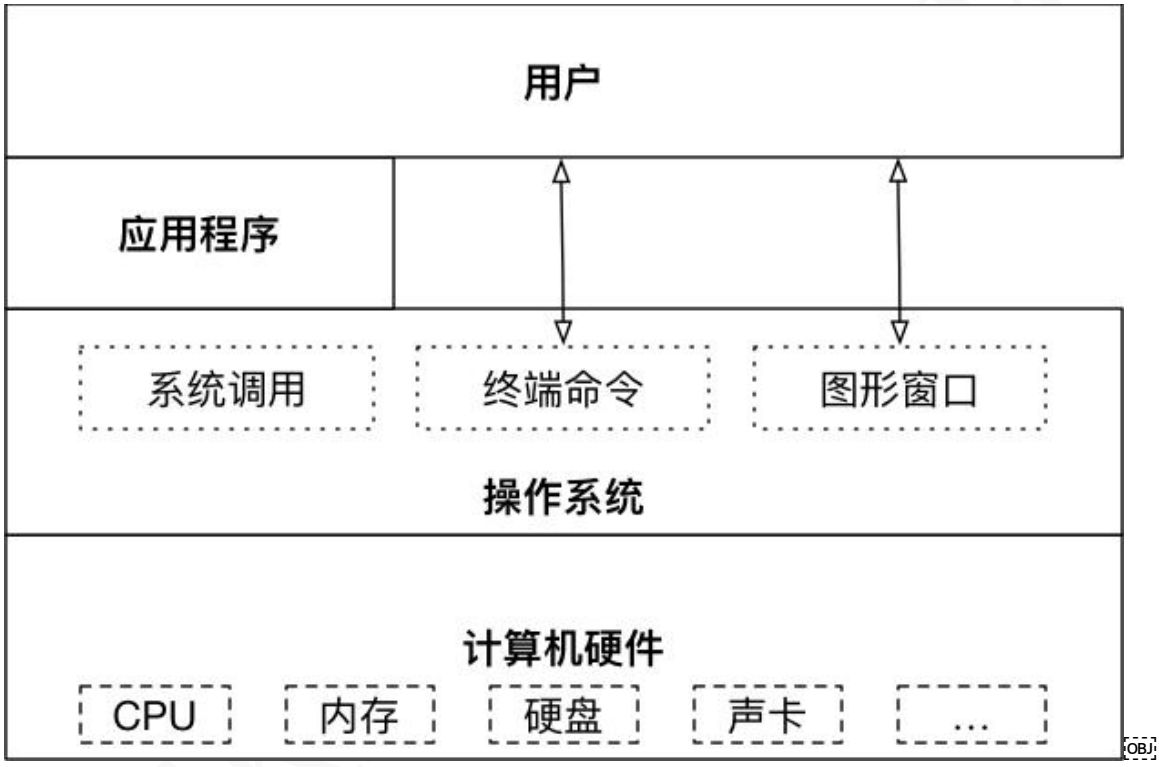
1 操作系统（科普章节）

目标

- 了解**操作系统**及作用

1. 操作系统（Operation System，OS）

操作系统作为接口的示意图



没有安装操作系统的计算机，通常被称为 **裸机**

- 如果想在 **裸机** 上运行自己所编写的程序，就必须用机器语言书写程序
- 如果计算机上安装了操作系统，就可以在操作系统上安装支持的高级语言环境，用高级语言开发程序

1.1 操作系统的作用

- 是现代计算机系统中 **最基本和最重要** 的系统软件
- 是 **配置在计算机硬件上的第一层软件**，是对硬件系统的首次扩展
- 主要作用是**管理好硬件设备**，并为用户和应用程序提供一个简单的接口，以便于使用
- 而其他的诸如编译程序、数据库管理系统，以及大量的应用软件，都直接依赖于操作系统的支持

1.2 不同应用领域的主流操作系统

- 桌面操作系统
- 服务器操作系统
- 嵌入式操作系统
- 移动设备操作系统

1> 桌面操作系统

- Windows 系列
  - 用户群体大
- macOS
  - 适合于 Mac 开发人员，UI
- Linux
  - 应用软件少

2> 服务器操作系统

- Linux
  - 安全、稳定、免费
  - 占有率高
- Windows Server
  - 付费
  - 占有率低

3> 嵌入式操作系统

- Linux

4> 移动设备操作系统

- iOS
- Android（基于 Linux）

1.3 虚拟机

虚拟机（Virtual Machine）指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统

- 虚拟系统通过生成现有操作系统的全新虚拟镜像，具有真实操作系统完全一样的功能
- 进入虚拟系统后，所有操作都是在这个全新的独立的虚拟系统里面进行，可以独立安装运行软件，保存数据，拥有自己的独立桌面，不会对真正的系统产生任何影响
- 而且能够在现有系统与虚拟镜像之间灵活切换的一类操作系统
-
-
-

修改虚拟机时间

2 Linux 常用命令

2.1 用户配置

1、**Linux 下有两种用户：超级用户（root）、普通用户。**

a) 超级用户：可以再 linux 系统下做任何事情，不受限制。

b) 普通用户：在 linux 下做有限的事情。

超级用户的命令提示符是“#”，普通用户的命令提示符是“$”。

命令：su [用户名]

功能：切换用户。

例如，要从 root 用户切换到普通用户 user，则使用 su user。

要从普通用户 user 切换到 root 用户则使用 su root（root 可以省略），此时系统会提示输入 root 用户的口令。

普通用户以 root 的身份去做事情，使用命令 sudo（可以让普通用户去做部分 root 事情）

2、**添加用户。**

命令：**useradd 用户名**

功能：添加一个普通用户。

例如，要想添加一个普通用户 user1 ， 则可以使用 useradd user1。使用该命令后，系统会在 目录“/home”下建立一个名为 user1 的目录。加 -m 才会创建目录

ubuntu 下需要 useradd -m test -s /bin/bash

-s 的作用是指定使用的脚本解析器

cat /etc/passwd 查看当前系统的用户

Tab 键可以联想

3、**设置密码**

命令：passwd 用户名

功能：设置或修改用户名的密码。

例如，我们要给刚才创建的 user1 用户设置一个密码 123456，则使用 passwd user1，然后系统会提示你输入新密码。注： root 用户才可以给新创建的用户配置密码

sudo passwd

4、**删除用户**

命令：userdel 用户名

注：删除用户后，其家目录并不会被删除，如果要删除家目录，需要 userdel -r 用户名

2.2 目录及文件操作

1、查看文件或目录

命令：ls［选项］［目录或文件］

功能：对于目录，该命令列出该目录下的所有子目录与文件。对于文件，将列出文件名以及其他信息。Linux 文件系统不是根据后缀名来执行文件的，而是根据此文件是否有可执行权限。

常用的选项有：

-a 显示指定目录下所有子目录与文件。例如 列出“/root/home”目录下的所有子目录及文件，则使用

ls -a /root/home 。

就会把隐藏文件显示出来，Linux 下如果文件名以点开头就是隐藏文件。

配置文件会作为隐藏文件设计。

-l 列出指定目录下所有目录及文件的详细信息。例如 列出“/root/home”目录下的所有子目录及文件，则使用

ls -l /root/home 。每行列出的详细信息依次是：

文件类型与权限 连接数 文件所有者 文件所属组 文件大小 最近修改时间 文件名字。

使用 ls -l 命令显示的信息中 ，开头是由 10 个字母构成的字符串，其中第一个字符表示文件类型，它可以是下列类型之一：

- ：普通文件

d： 目录

l：符号链接

b：块设备文件

c：字符设备文件

p：命名管道

s：socket 文件

后面的 9 个字符表示文件的访问权限，分为 3 组，每组 3 位。第一组表示文件创建者的权限，第二组表示同组用户的权限，第三组表示其他用户的权限。每一组的三个字符分别表示对文件的读、写、执行权限。

各权限如下：r（读）、w（写）、x（执行）、_（没有设置权限）。

r 的值是 4

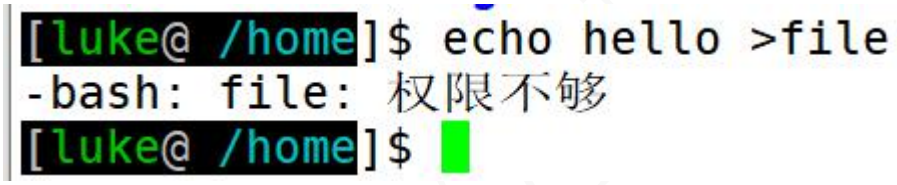
w 的值是 2

x 的值是 1

每一组可以用一个数字表示，例如 r_x ： 5 ， rw_:6 R__:4，那么这三组就可以用 3 个数字表示，例如 rwxr_xr_x:755 ， rw_r__r__:644。

ls 输出内容是有颜色的，比如：目录是蓝色，压缩文件是红色的显示，如果没有颜色，可以加上参数--color=never 表示输出没有彩色，而--color=auto 表示自动，--color=always 表示始终有颜色。

如果一个用户对目录没有写权限，那么它不能在这个目录新增，或者删除文件。



如果需要更加详细的参数描述，可以通过如下方式获得 ls 的帮助：man ls。

ls -lh 人类读起来比较开心的大小

2、改变工作目录。

命令：cd 目录名

功能：改变工作目录。将当前工作目录改变到指定的目录下，例如要切换当前目录到“/home/user/0718”目录，则使用 cd /home/user/0718。

常用的切换目录命令：

cd .. 到父目录

cd / 到根目录

cd ~ 到用户主目录(家目录)下~ 与直接执行 cd 效果一样

cd - 到上一次目录

3、显示当前工作目录。

命令： **pwd**

功能： 显示用户当前所在的目录。例如当我们使用命令 cd /home/user/0718 时，再使用命令 pwd 则命令行会显示

/home/user/0718。

4、**创建目录**

命令：**mkdir [选项] dirname**

功能：在当前目录下创建一个名为 “dirname”的目录。例如要在当前目录下创建一个名为“07181”的目录，则使用命令 mkdir 07181。系统就会在当前目录下，创建一个 07181 的目录，此时可以使用 ls -l 查看。

5、**删除目录**

命令：rmdir [选项] dirname

功能：在当前工作目录下删除目录名为“dirname”的子目录。此时该子目录必须是个空目录。我们刚才创建了一个空目录 07181，如果我们想把它删掉，则使用 rmdir 07181 。此时再使用 ls -l 列举一下，这时发现 07181 已经被删掉了。如果使用该命令删除一个非空的目录，则删除失败。

6、**拷贝文件或目录**

命令 cp [选项] 源文件或目录 目标文件或目录

功能：把指定的源文件复制到目标文件或把多个源文件复制到目标目录中。

常用参数：

-f 若目标目录中存在与源文件同名的文件，则直接覆盖，不提示。例如将当前目录下的 main.c 文件拷贝到“/home/user/0718”下，并且若存在同名的则进行覆盖，使用：

cp -f ./main.c /home/user/0718 。

如果在拷贝的同时将源文件重命名，例如将当前目录下的 main.c 文件拷贝到“/home/user/0718”目录下并命名为 main1.c，则使用：

cp -f ./main.c /home/user/0718/main1.c。

-i 和-f 相反，当目标文件中存在于源文件同名的文件，copy 时系统会提示是否进行覆盖。里如上例，若在拷贝过程中，目标文件中存在与源文件同名的文件，需要提示是否覆盖，则只需要将上例中的-f 改为 -I 即可。例如

cp -i ./main.c /home/user/0718 。

cp -i ./main.c /home/user/0718/main1.c。

-r 如果要拷贝的是一个目录，此时将同时拷贝该目录下的子目录和文件。此时目标文件必须为一个目录。例如，将”

/home/user/0718”目录下的所有文件及目录拷贝到“/home/user1”目录下，则使用

cp -r /home/user/0718 /home/user1 。(重点)

7、移动文件或目录。

命令：mv [选项] 源文件或目录 目标文件或目录

功能：视 mv 命令中第二个参数类型的不同（是目标文件还是目标目录），mv 命令将文件重命名或将其移至一个新的目录中。当第二个参数类型是文件时，mv 命令完成文件重命名，此时，源文件只能有一个（也可以是源目录名），它将所给的源文件或目录重命名为给定的目标文件名。当第二个参数是已存在的目录名称时，源文件或目录参数可以有多个，mv 命令将各参数指定的源文件均移至目标目录中。在跨文件系统移动文件时，mv 先拷贝，再将原有文件删除，而链至该文件的链接也将丢失。

参数：

-i 如果在移动的过程中存在重名的，则进行提示是否覆盖。

-f 若果在移动的过程中存在重名的，则直接进行覆盖，不会给出提示。

例如 要将 “/home/user/0718”下的 main.c 文件重命名为 main.cpp，则使用

mv /home/user/0718/main.c /home/user/0718/main.cpp。

要将“/home/user/0718”下的所有内容移动到“/home/user/0719”,则使用

mv -f /home/user/0718 /home/user/0719。

8、删除文件或目录

命令：rm [选项] 文件或目录

功能：在 linux 中创建文件很容易，系统中随时会有文件变得过时且毫无用处。用户可以用 rm 命令将其删除。该命令的功能为删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件，只是删除了链接，原有文件均保持不变。如果删除时没有-r 选项则不会删出目录。

参数：

-f 删除过程中不会给出提示。

-i 删除过程中会给出交互式提示。

-r 如果删除的是一个目录，则将该目录下的目录及子目录均删

除掉。

例如要删除 “/home/user/0718”目录下的 main.cc main.exe 则使用，

rm -f /home/user/0718/main.cc /home/user/0718/main.exe

若果要删除“/home/user/0718”这个目录，则使用 rm -rf /home/user/0718。

9、显示目录树形结构

命令：tree 在 ubuntu 需要执行 sudo apt install tree 进行安装

功能说明：以树状图列出目录的内容。

语　　法：tree [-aACdDfFgilnNpqstux][-I <范本样式>][-P <范本样式>][目录...]

补充说明：执行 tree 指令，它会列出指定目录下的所有文件，包括子目录里的文件

-a　显示所有文件和目录。不包含.和..

-C　在文件和目录清单加上色彩，便于区分各种类型。

-d　显示目录名称而非内容。

-D　列出文件或目录的更改时间。

-g　列出文件或目录的所属群组名称，没有对应的名称时，则显示群组识别码。

-i　不以阶梯状列出文件或目录名称。

-t　用文件和目录的更改时间排序。

-u　列出文件或目录的拥有者名称，没有对应的名称时，则显示用户识别码。

tree -h 显示每个文件大小

10、改变目录或文件的权限

功能：chmod 命令是非常重要的，用于改变文件或目录的访问权限。用户用它控制文件或目录的访问权限。

语法：该命令有两种用法。一种是包含字母和操作符表达式的文字设定法；另一种是包含数字的数字设定法。

说明：我们利用 ls -l 长格式列出文件或目录的基本信息如下：

文件类型与权限 链接数 文件所有者 文件属组 文件大小 最近修

改的时间 名字
对于权限，有第一组表示文件所有者的权限，第二组表示同组用户的权限，第三组表示其他用户的权限。每一组的三个字符分别表示对文件的读、写和执行权限。可以通过 chmod 来修改权限。

1. 文字设定法

chmod [who][+|-|=][mode] 文件名 //修改指定文件名中 who 的权限增加/去除/赋值为 mode
参数：
操作对象 who 可是下述字母中的任一个或者它们的组合：
u 表示“用户（user）”，即文件或目录的所有者。
g 表示“同组（group）用户”，即与文件所有者有相同组 ID 的所有用户。
o 表示“其他（others）用户”。
a 表示“所有（all）用户”。它是系统默认值。即 chmod +x 1.c 表示所有人都有可执行的权限。
操作符号可以是：
+ 添加某个权限。
- 取消某个权限。
= 赋予给定权限并取消其他所有权限（如果有的话）。
设置 mode 所表示的权限可用下述字母的任意组合(当组合的时候，who 不能少)：
r 可读
w 可写
x 可执行
文件名：以空格分开的要改变权限的文件列表，支持通配符。
在一个命令行中可给出多个权限方式，其间用逗号隔开。例如：chmod g+r，o+r example 使同组和其他用户对文件 example 有读权限。

2. 数字设定法

我们必须首先了解用数字表示的属性的含义：0 表示没有权限，1 表示可执行权限，2 表示可写权限，4 表示可读权限，然后将其相加。所以数字属性的格式应为 3 个从 0 到 7 的八进制数，其顺序是（u）（g）（o）。
例如，如果想让某个文件的所有者有"读/写"二种权限，需要把 4（可读）+2（可写）＝6（读/写).

数字设定法的一般形式为：chmod [mode] 文件名

例子：

（1）文字设定法：

例 1：$ chmod a+x sort

即设定文件 sort 的属性为：

文件所有者（u） 增加执行权限

与文件所有者同组用户（g） 增加执行权限

其他用户（o） 增加执行权限

例 2：$ chmod ug+w，o-x text

即设定文件 text 的属性为：

文件所有者（u） 增加写权限

与文件所有者同组用户（g） 增加写权限

其他用户（o） 删除执行权限

例 3：$ chmod a－x mm.txt

$ chmod －x mm.txt

$ chmod ugo－x mm.txt

以上这三个命令都是将文件 mm.txt 的执行权限删除，它设定的对象为所有使用者。

（2）数字设定法：

例 1： $ chmod 644 mm.txt

$ ls －l

即设定文件 mm.txt 的属性为：

-rw-r--r-- 1 inin users 1155 Nov 5 11:22 mm.txt

文件所有者（u）inin 拥有读、写权限

与文件所有者同组人用户（g） 拥有读权限

其他人（o） 拥有读权限

例 2： $ chmod 750 wch.txt

$ ls －l

-rwxr-x--- 1 inin users 44137 Nov 12 9:22 wchtxt

即设定 wchtxt 这个文件的属性为：

文件主本人（u）inin 可读/可写/可执行权

与文件主同组人（g） 可读/可执行权

其他人（o） 没有任何权限

11、文件查找

命令：find 起始目录 查找条件 操作

功能：在指定目录结构中搜索问价，并执行指定的操作。该命令的查找条件可以是一个逻辑运算符 not、and、or 组成的复合条件。

（1）and：逻辑与，在命令中用-a 表示，表示只有当所给的条件都满足时，查找条件才满足。例如在 “/home/user”目录下查找名为 0718 类型是一个目录的文件。则使用

find /home/user -name 0718 -a -type d

（2）or：逻辑或，在命令中用-o 表示，表示只要所给的条有一个满足，查找条件就满足。例如在“/home/user”目录下查找名字为 main.cc 或名字为 main.c 的文件。则使用

find /home/user -name main.cc -o -name main.c。

（3）not：逻辑非，在命令中用！表示查找不满足所给条件的文件。例如在“/home/user“下查找名字不是 main.c 的文件，则使用

find /home/user ! -name main.cc 。

常用的查找条件有：

（1）根据名称和文件属性查找。

-name ’字串’ 查找文件名匹配所给字串的所有文件，字串内可用通配符*、?、[ ]。

＊ 代表零个或者任意多个字符

？有且只有一个字符

[] 连续的一部分字符

-gid n 查找属于 ID 号为 n 的用户组的所有文件。

-uid n 查找属于 ID 号为 n 的用户的所有文件。

-group ’字串’ 查找属于用户组名为所给字串的所有的文件。

-user ’字串’ 查找属于用户名为所给字串的所有的文件。

-empty 查找大小为 0 的目录或文件。

-perm 权限 查找具有指定权限的文件和目录，权限的表示可以如 711，644。

-size n[bckMG] 查找指定文件大小的文件，n 后面的字符表示单位，缺省为 b，代表 512 字节的块。后面要带加号或者减号。

-type x 查找类型为 x 的文件，x 为下列字符之一：

b 块设备文件
c 字符设备文件
d 目录文件
p 命名管道（FIFO）
f 普通文件
l 符号链接文件（symbolic links）
ln -s 建立软连接
ln 建立硬链接
s socket 文件

（2）根据时间查找

- amin n 查找 n 分钟以前被访问过的所有文件。（+表示 n 分钟之前，-表示 n 分钟之内，**+号和-号都不能省略**）
- cmin n 查找 n 分钟以前文件状态被修改过的所有文件。
- **mmin n 查找 n 分钟以前文件内容被修改过的所有文件。**
- atime n 查找 n 天以前被访问过的所有文件。
- ctime n 查找 n 天以前文件状态被修改过的所有文件。
- **mtime n 查找 n 天以前文件内容被修改过的所有文件。**

（3）可执行的操作。

- exec 命令名称 {} :对符合条件的文件执行所给的 Linux 命令，而不询问用户是否需要执行该命令。{}表示命令的参数即为所找到的文件；命令的末尾必须以“ \;”结束。

例如，在“/home/user“目录下查找名为 main.c 文件并显示这些文件的详细信息，则使用

```
find /home/user -name main.c -exec ls -l {} \;

find /home/luke -name main.c |xargs ls -l
```

rm 和 find 组合就和 ls 类似

工作时一般不采用 exec，而是采用 xargs，例如上面操作可以改为 find /home/user -name main.c|xargs ls -l

下面是 find 与具有两个操作对象的命令组合

```
find . -type f -exec cp {} /home/user \;

find . -type f | xargs -
i cp {} /home/user
```

12、列出文件系统的整体磁盘空间使用情况

df [选项] [文件名]

-h：--human-readable，以人们易读的 GB、MB、KB 等格式显示，可以直接 df -h 显示整个磁盘使用情况

13、显示每个文件和目录的磁盘使用空间

du [选项] [文件名]

-h：--human-readable，以人们易读的 GB、MB、KB 等格式显示操作发现目录的每一级都显示，如果只想显示当前目录，du -h --max-depth=0 /home/luke

2.3 文件查看及处理命令

1、查看文件内容

命令： **cat [选项] [文件]**

功能： 查看目标文件的内容

参数：

-b 对非空输出行编号

-E 在每行结束处显示$

-n 对输出的所有行编号

-s 不输出多行空行。

例如 要查看当前目录下的 main.py 的内容

则使用，**cat main.py。**

标准的输入输出与重定向：

文件描述符是一个整数,它代表一个打开的文件,标准的三个描述符号:

标准输入：一般指键盘,描述符为：0

标准输出：一般指屏幕输出,描述符为：1

错误输出：也是屏幕,描述符为：2

重定向符号:

<重定向输入 、 **>重定向输出** 、 **>>添加输出** 、 2>错误重定向 、 &>错误和信息重定向

```
Cat >file1<file2
./print.py >text5.txt 2>&1
```

cat 常常与重定向一起使用。其中**>表示创建，>>表示追加**,<<表

示以什么结束

如果 cat 的命令行中没有参数，它就会从标准输入中读取数据，并将其送到标准输出。

linux 中创建空文件的四种方式：

方式 1： echo > a.txt（会有一个字节）

方式 2： touch b.txt

方式 3： cat > c.txt 按 ctrl+c 组合键退出；或 Ctrl+d

方式 4： vi d.txt 进入之后：wq 退出。

2、显示文件内容的前几行

命令：head -n 行数值 文件名

功能：显示目标文件的前几行。

例如 要显示 当前目录下 main.cc 的前 10 行，则使用 Head -n 10 main.cc。

3、显示文件的后几行

命令：tail -n 行数值 文件名。

功能：显示目标文件的最后几行。

例如 要显示“/home/user/0718/”目录下的 main.cc 文件的最后 10 行。则使用 tail -n 10 /home/user/0718/main.cc

4、单页浏览文件

more 或者 less 命令

5、对文件内容进行排序

sort 文件名

6、查看文件内容类型

file 文件名 **根据文件内容，判别文件类型**

7、报告或删除文件中重复的行

uniq 文件名

-c 在输出行前面加上每行在输入文件中出现的次数。

-d 仅显示重复行。

-u 仅显示不重复的行。

8、统计指定文件中的行数、字数、字节数

wc 文件名

-c 统计字节数。

-l 统计行数。

-m 统计字符数。这个标志不能与 -c 标志一起使用。

-w 统计字数。一个字被定义为由空白、跳格（tab）或换行字

符分隔的**字符串**

```
 wc main.c
[luke@ ~/day3]$ wc day3_history.txt
 318  990 5372 day3_history.txt
```

依次行数，单词数，字符数

9、汉字编码转换（乱码不存在）

iconv

输入/输出格式规范：

-f, --from-code=名称 原始文本编码

-t, --to-code=名称 输出编码

举例：iconv -f utf-8 -t gb2312 hanzi>hanzi1

10、**搜索文件内容 grep**

命令：grep [选项] [查找模式] [文件名 1，文件名 2，…]

功能：grep 过滤器查找指定字符模式的文件，并显示含有此模式的所有行。被寻找的模式称为正则表达式。

grep 的通配符

零个或任意多个字符 .* 场景：字符串中间部分

一个字符 .

字符范围 []

常用的一些正则表达式

^ :以什么开头 ，例如 ls -l | grep ^d 显示当前目录下的所有子目录的详细信息。

$:以什么结尾 。例如 ls -l | grep c$ 显示当前目录下以 c 结尾的文件。

常用的参数：

-F 每个模式作为固定的字符串对待

-c 只显示匹配行的数量。

-i 比较式不区分大小写。

-n 在输出前加上匹配串所在的行号。

如何找到所有 py 文件里是否调用了 print 函数?

```
find . -name "*.py"|xargs grep print
```

2.4 其他命令

1、管道与命令替换

管道：是重定向的一种，就像一个导管一样，将一个程序或命令

的输出作为另一个程序或命令的输入。eg: #ls -l /etc | wc -w

命令替换：和重定向有点相似，但区别在于命令替换是将一个命令的输出作为另一个命令的参数。常用的格式为 ：command1 `command2`或 command1 $(command2)

举例：

首先列出当前的所有信息，并重定向到 aa 文件中：

#ls | cat > aa 或 ls > aa

然后，通过命令替换，列出 aa 文件中所有的文件信息

#ls -l `cat aa` 或者用 ls -l $(cat aa)

2、文件或目录的创建掩码

umask 指文件（0666）或目录（0777）创建时在全部权限中要去掉的一些权限，普通用户缺省时 umask 的值为 002，超级用户为 022。

002 表示创建目录时所有者的权限不去掉， 所属组权限不去掉，其他组权限写属性去掉

创建一文件以后，普通用户缺省的权限为 664

超级用户： 644

创建一目录以后，普通用户缺省的权限为 775

超级用户： 755

可以通过 umask 查看默认的缺省的掩码值。通过 umask 001 修改掩码值。

3、文档管理

命令：tar [主选项+辅选项] 目标文档 源文件或目录

功能：tar 可以为文件和目录创建档案。利用 tar，用户可以为某一特定文件创建档案（备份文件），也可以在档案中改变文件，或者向档案中加入新的文件。tar 最初被用来在磁带上创建档案，现在，用户可以在任何设备上创建档案，如软盘。

利用 tar 命令，**可以把一大堆的文件和目录全部打包成一个文件，**这对于备份文件或将几个文件组合成为一个文件以便于网络传输是非常有用的。

常用参数：

c：创建新的档案文件。

r: 要把存档的文件追加到档案文件的末尾。

tar rf *.tar test

x：从档案文件中释放文件。

f：使用档案文件或设备。

v：在归档过程中显示处理的文件。

z：用 gzip 来压缩/解压缩文件，后缀名为.gz，加上该选项后可以将档案文件进行压缩。

例如，把“/home/user/0718”下的所有后缀为.c 的归档到 source.tar，则使用 tar cvf source.tar /home/user/0718/*.c

若果在归档的过程中还要进行压缩，则使用

tar czvf source.tar.gz /home/user/0718/*.c。

如果要将归档的文件 source.tar 释放掉，则使用

tar xf source.tar / tar xf source.tar

如果将归档后的压缩文件释放掉，则使用

tar xf source.tar.gz

4、文件压缩解压

命令：gzip/bzip2 [选项]压缩或解压缩的文件名

功能：gzip 用来将文件压缩成后缀为.gz 的压缩文件，或者将后缀为.gz 的文件进行解压。Bzip2 用来将文件压缩成后缀名为.bz2 的压缩文件，或者将后缀为.bz2 的压缩文件解压。

常用参数：

-d: 将压缩文件进行解压。

-v： 在压缩或解压过程中显示解压或压缩的文件。

例如，将 main.c 进行压缩，则使用

gzip/bzip2 -v main.c。

则就会将 main.c 压缩成 main.c.gz 或者 main.c.bz2 。

如果将刚才的压缩文件解压，则使用

gzip -dv main.c.gz,或者 bzip2 -dv main.c.bz2。

5、scp 远程 copy 文件命令

scp filename username@ip:path

filename:文件名称

username:copy 到的目标主机的用户名

ip:目标主机 IP

path:目标主机路径

scp file3 king@192.168.4.52:~/ 从本机 copy 到 4.52 机器

scp king@192.168.4.52:~/file3 . 从 4.52 机器 copy 到本机

scp file8 luke@192.168.2.100:~

如果要 scp 文件夹，需要加-r

无秘钥登录设置

ssh-keygen 一直回车即可

ssh-copy-id python5@42.192.117.114 这一步要输入密码，密码是在 day2 笔记里

6、用来查看和配置网络设备

ifconfig 当网络环境发生改变时可通过此命令对网络进行相应的配置，只有 root 权限才可以配置网络

直接执行 ifconfig 查看网络信息

通过 ifconfig 网卡名 down 用来关闭网络，例如 ifconfig ens33 down

通过 ifconfig 网卡名 up 用来启动网络，例如 ifconfig ens33 up

重启网络服务

sudo /etc/init.d/networking restart

7、查看与设置路由

route

当网络不通时，通过执行 route 查看路由，查看网关配置是否正确

8、ubuntu 设置固定 IP

首先点击右上角的《上下箭头》，点 Edit Connections,点 edit，点击 ipv4 setting

9、安装中文帮助文档

1.安装中文帮助包 sudo apt-get install manpages-zh

2.查看安装路径 dpkg -L manpages-zh | less

3.回到家目录 cd ~

4.编辑 bashrc vim .bashrc

5.添加新命令 alias cman='man -M /usr/share/man/zh_CN' 这个路径是 2 中查到的，默认就是这个

6.更新 bashrc source .bashrc

7.结束 cman 可以查看中文帮助，man 查看英文帮助（这个作用不大，因为只有少部分有中文）

•
•
•

3 VIM 编辑器

3.1 如何安装

```
sudo apt install vim
```

3.2 VIM 的两种状态

VIM(vimsual)是 Linux/UNIX 系列 OS 中通用的全屏编辑器。
vim 分为两种状态，**即命令状态和编辑状态，在命令状态下，所键入的字符系统均作命令来处理，如:q 代表退出，而编辑状态则是用来编辑文本资料的。当你进入 vim 时，会首先进入命令状态。在命令状态下，按”i”(插入)或”a”(添加)可以进入编辑状态，在编辑状态，按 ESC 键进入命令状态。**
在命令状态下，有如下一些常用命令：
新增：

a	从光标后面开始添加文本
A	从光标所在行的末尾开始添加文本

插入：

i	从光标前面开始插入文本
I	从光标所在行的开始处插入文本

2.2 VIM 内常用的命令操作

2.2.1 删除与修改

x	删除光标处的字符
dd	删除光标所在的整行
3dd	删除光标所在行以及下面的两行（删除 3 行）
D	删除光标到行尾的文本，常用语删除注释语句 (d$)
yy	复制光标所在的整行

[n]yy 从光标开始往下复制 n 行,[n]表示一个整数

p	将复制后的文本粘贴到光标处
u	撤销上次操作

先 dd 后再去 p 的效果就是剪切的效果

2.2.2 光标移动

^	光标移动到行首
$	光标移动到行尾
Ctrl+d	向下翻半页
Ctrl+f	向下翻一页
Ctrl+u	向上翻半页
Ctrl+b	向上翻一页
gg	**光标定位到文档头**
G	**光标定位到文档尾**
H	光标定位到当前页首
L	光标定位到当前页的最后一行的行首
w	光标往后移一个字
b	光标往前移一个字
[n]+	光标向后移动 n 行,[n]表示一个整数 10+
[n]-	光标向前移动 n 行,[n]表示一个整数 10-
[n]G	光标定位到第 n 行行首, [n]表示一个整数 20G

也可以:50 到达第 50 行

2.2.3 查找与替换

/[str] 查找字符串 str，[str]表示要查找的字符串
回车后会加亮显示所有找到的字符串，接着
命令 n 移动到下一个找到的字符串，命令 N 移动到
上一个找到的字符串 eg /hello

部分替换 (只能替换光标之所在的行)

:s/[src]/[dst] /i 忽略大小写 /g 全部匹配

eg :s/hello/world/ig 替换一行

:3,6s/[src]/[dst]/ig（3-6 行中找） eg :3,6 s/hello/world

全部替换

:%s/[src]/[dst]/g 将文档中所有 src 的字符串替换为 dst 字符串

:%s/^ //g 将文档每一行的行首的空格去掉

2.2.4 块操作

v 可视化块选择状态，选中块之后，可以对块进行删除(d),复制(y),剪切(x)

Ctrl +v 竖向选择模式，主要用于批量注释代码，输入步骤如下：

1、首先按 ctrl+v，竖选选中要注释的行

2、输入 I（注意是大写的 I），然后输入//

3、再输入 esc，就会看到选中的行被注释了。

2.2.5 文档保存及退出

结束编辑：

:q 在未修改文档的情况下退出

:q! 放弃文档的修改，强行退出

:w 文档存盘

:wq 文档存盘退出

其他：

:help 命令 查看该命令的帮助提示（不常用，当不小心按 F1 时，通过:q 进行退出）

:%!xxd 十六进制模式

:%!xxd -r 返回文本模式 中间有一个空格的

如果在编辑过程中不小心按了 Ctrl+s,vim 会处于僵死状态，按 Ctrl+q 可以恢复。

在命令模式下输入:new 2.c //表示再打开一个 vim,是横向的用 vnew 2.c 表示纵向

也可以通过:split vsplit sp vsp，两个窗口之间进行切换的方式：Ctrl+w,w

2.3 VIM 外使用到的命令

sed：管道查找替换程序

sed 's/aa/bb/' a.txt > b.txt //将 a.txt 中的 aa 替换成 bb 并重定向输出到 b.txt 中。

sed -i "s/print/mysprint/g" *.py

Vimdiff 命令讲解

•

•

•
•

2.4 修改配置

.bashrc 修改 PS1，alias，PATH 配置

永久变色

export PS1="\[\e[37;40m\][\[\e[32;40m\]\u\[\e[37;40m\]@\[\e[36;40m\]\w\[\e[0m\]]\\$ "

做短命令

alias ll='ls -lh'

执行代码不需要./是因为对应的可执行代码在 PATH 环境变量路径下

1、没有~/.local/bin，就创建.local,和 bin

2、把 qscp 放到 bin 下面

3、export PATH=$PATH:~/.local/bin 添加到~/.bashrc

4、执行 source ~/.bashrc

4 Shell 编程(暂时不讲，了解)

1 初识 Shell 脚本

如果我们有一系列经常使用的 Linux 命令，我们可以把它们存储在一个文件中。Shell 可以读取这个文件并执行其中的命令。这样的文件被称为脚本文件。

执行 shell 脚本

要创建一个 shell 脚本，我们要使用任何编辑器比如 vi 在文本文件中编写它，保存的文件最好是.sh 后缀的。

如：vi aa.sh

chmod +x aa.sh 然后 ./aa.sh 或 bash aa.sh 或 sh aa.sh

2 shell 脚本的编写语法

2.1 变量

2.1.1 程序开始与注释

1.程序往往以下面的行开始#!/bin/bash（redhat/suse 下，所以系统默认的 shell 是 bash shell。）

2.注释用#

2.1.2 shell 变量

shell 变量没有数据类型，都是字符串，即使数值也是字符串

创建变量：变量名称=值。如果值有空格则必须用""或者’ ’ 引用起来

Eg: a=“hello” （=号两边不能有空格）

引用变量：echo $a 或 echo ${a} 或 echo “${a}” 注意‘’ “” 的区别（单引号：消除所有字符的特殊意义；双引号：消除除$、”” 、’ ’ 三种以外其它字符的特殊意义）

1>: #echo →hello 等同于#echo ${a}
#echo “${a}”

2>: #echo “hello b$aa” →hello b,因为此时把 aa 作为一个整体变量，而且没有定义，所以输出前面的字符串

3>: #echo “hello b${a}a” →hello bhelloa

4>: #echo”${a}a” →helloa

5>: #echo ‘${a}a’ →${a}a,因为’ ’ 会消除特殊字符的意义。

6>: #echo ‘\${a}a’ →\${a}a.

删除变量：unset 变量名 eg: unset a

还可以设置变量为只读变量 readonly a=3

也可以允许用户从键盘输入，实现程序交互：read a

echo $? 用于显示上一条命令的执行结果（0 表示成功，1 表示失败），

或者函数返回值。

转义符 a=What\’s\ your \ \"topic\"\? (➔#a=”What’ s your \” topic\” ?”)

#echo $a

命令代换 echo `date`（小飘号） 或 echo $(date) 显示当前系统时间，

即用系统变量时，用 echo $(命令)的形式等价于 echo `

命令`

eg:echo `pwd` ➔echo $(pwd)

表达式计算:

expr 4 + 5	expr $a + $b
echo `expr 4 + 5`	echo `expr $a + $b`
echo $(expr 4 + 5)	echo $(expr $a + $b)
echo $((4 + 5))	echo $(($a + $b))
echo $[4 + 5]	echo $[$a + $b]

举例：写 1.sh 要求读入 1 个目录名，在当前目录下创建该目录，并复制 etc

下的 conf 文件到该目录，统计 etc 下所有目录的数目到 etcdir.txt 中

```
==>#!/bin/bash
   #this is my first shell project
   read dir
   mkdir ${dir}
   cp -rf /etc/*.conf ${dir}
   ls -l /etc/* | grep ^d | wc -l > etcdir.txt
```

2.1.3 标准变量或环境变量

系统预定义的变量，一般在/etc/profile 中进行定义

HOME 用户主目录 PATH 文件搜索路径

PWD 用户当前工作目录 PS1、PS2 提示符

UNAME HOSTNAME LOGNAME echo $PWD

用 echo $PATH 显示，用 env 看环境所有变量，用 env | grep "name"查找

用"export"进行设定或更改为全局变量，

用 unset 变量名 ➔取消全局变量

的定义

例: 定义本地变量 name="Red Hat Linux" export name 把 name 变为全

局变量

sh 进入子 shell echo ${name}全局变量可以作用于子进程，而本地变量不

可以。

或直接输出 export name="Red Hat Linux"

bash 退出子 shell,进入父 shell

设置环境变量：比如把/etc/apache/bin 目录添加到 PATH 中：

1. #PATH=$PATH:/etc/apache/bin
2. vi /etc/profile 在里面添加 PATH=$PATH:/etc/apache/bin
3. vi ~/.bash_profile 在里面修改 PATH 行，把/etc/apache/bin 加进去，此种方法针对当前用户有效。

2.1.4 特殊变量

$1,$2…$n 传入的参数　$0 表示 shell 程序名称 →每一项相当于 main 函数中 argv[i]

$#传递到脚本的参数列表,或表示参数个数　→等价于 main 函数中的 argc-1

$@传入脚本的全部参数　→argv[1] ---- argv[n-1]

$* 显示脚本全部参数

$? 前个命令执行情况　0 成功　1 失败

$$ 脚本运行的当前进程号

$! 运行脚本最后一个命令

举例：

vi 1.sh

```
#!/bin/bash
echo $1
echo $2
echo $3
echo $#
echo $@
echo $*
echo $$
exit 3
```

./1.sh 1 2 hello "hello world"

echo $?

2.2 运算符与表达式

算术运算符(+、-、*、/、%)

逻辑运算符(&&、||、 > 、= = 、<、！=)

赋值运算符(=、+=、-=、*=、/=、%=、&=、^=、|=、<<=、>>=)

计算表达式有四种：1、$(()) 2、$[] 3、let var= 4、expr 4 + 5

echo $[$v1 < $v2] 计算逻辑表达式(用 1 表示 true,用 0 表示 false)

echo $[($v1<$v2)&&($v1>$v2)] 计算逻辑表达式

v3=2

let v3*=$(($v1+$v2))

echo $v3 或 echo ${v3}

举例：写 2.sh 要求输入 2 个数 计算 2 个数的和

```
#!/bin/bash
#this is my second shell project
echo "please input the first number:"
read a
echo "please input the second number:"
read b
c=$(($a + $b))
echo "The result of $a + $b is $c"
```

2.3 Test 命令的用法

VAR=2 test $VAR -gt 1 echo $?

1）判断表达式 and or

test 表达式 1 -a 表达式 2 两个表达式都为真

test 表达式 1 -o 表达式 2 两个表达式有一个为真

测试是否是闰年：test $(($iYear % 400)) -eq 0 -o $(($iYear % 4)) -eq 0 -a $(($iYear % 100)) -ne 0

2）判断字符串

test -n 字符串 字符串的长度非零

-z 字符串长度为零 ==字符串相等 ! = 字符串不等

a=”abc” test $a == ”abc” echo $?(0) test $a == ”

afd” echo $?(1)

3）判断整数

test 整数 1 -eq 整数 2 整数相等

-ge 大于等于 -gt 大于 -le 小于等于 -lt 小于 -ne 不等于

4）判断文件

test File1 -ef File2 两个文件具有同样的设备号和 i 结点号

test File1 -nt File2 文件 1 比文件 2 新

test File1 -ot File2 文件 1 比文件 2 旧

test -d File 文件存在并且是目录

test -e File 文件存在

test -f File 文件存在并且是正规文件

test -r File 文件存在并且可读

test -w File 文件存在并且可写

test -x File 文件存在并且可执行

举例：

```
a=2
test $a -ge 3
echo $?
```

8. 数组

定义 1：a=(1 2 3 4 5)下标从 0 开始 各个数据之间用空格隔开

定义 2：a[0]=1;a[1]=2;a[2]=3

定义 3：a=([1]=1 [2]=2)

引用 ${a[1]}

${#a[@]}数组长度 →${#a[*]}

${a[@]:1:2}从下标 1 开始后面显示 2 个

${a[@]}或${a[*]}输出数组的所有元素

例子

```
#a=(2 5 7 10)
#echo ${a[2]} //输出下标为 2 的数据
#echo ${#a[*]}//输出数组的长度
#echo ${a[@]:2} 截取下标从 2 到最后
#echo ${a[@]:1:2}//截取从下标 1 开始后面连续 2
```

```
#!/bin/bash
a=(3 10 6 5 9 2 8 1 4 7)
x=0
while [ $x -lt ${#a[*]} ]
do
    echo ${a[$x]}//或者 echo ${a[x]}
    x=$(($x + 1))
done

#!/bin/bash
a=(3 10 6 5 9 2 8 1 4 7)
i=0
while  ((  i<10  ))     //类似 C 语言的写法
do
    echo ${a[i]}
    i=$(($i+1))
done
```

2.4 If 语句

if [condition] then action fi 只有当 condition 为真时，该语句才执行操作，否则不执行操作，并继续执行 “fi” 之后的任何行。

if [condition] then action elif [condition2] then action2 . . . elif [condition3] then else actionx fi

在使用时，将 “if” 和 “then” 放在不同行，如同行放置，则 if 语句必须要; 结束

举例: 用参数传 1 个文件名，该文件如果是文件并且可读可写就显示该文件，如果是目录就进入该目录，并判断 ls.sh 存在否，如果不存在就建立 1 个 ls.sh 的文件并运行该文件。

该文件的内容是 ls -li /etc > etc.list

```
#!/bin/bash
 if [ -f $1 -a -r $1 -a -w $1 ]   //判断是普通文件并可读可写➔if test  -f $1  -a  -r $1  -a  -w $1
 then
```

```
        cat $1    //显示文件内容
    elif [ -d $1 ]   //否则如果是目录
    then
        cd $1    //进入目录
        if [ -e ls.sh ]  //如果 ls.sh 该文件存在
        then
            chmod +x ls.sh   //赋予可执行的权限
            ./ls.sh     //执行
        else
            touch ls.sh   //如果不存在则创建 ls.sh
            echo "#!/bin/bash" >> ls.sh   //将程序写入 ls.sh
    中保存
            echo "ls -li /etc > etc.list" >> ls.sh   //将要
    执行的命令写入 ls.sh 中保存
            chmod +x ls.sh    //赋予可执行的权限
            ./ls.sh
        fi

    fi
```

2.5 Case 语句

case 常用的语法形式如下：

```
case  $1 in
     "1")
        echo you inputed "1"
        ;;
    "2")
        echo you inputed "2"
        ;;
     *)
        echo you inputed other number
        ;;
esac
```

例子 1

```
    echo "Is it morning? Please answer yes or no."
    read YES_OR_NO
    case "$YES_OR_NO" in
          yes|y|Yes|YES)
          echo "Good Morning!";;
          [nN]*) /* 表示 n 或 N 开头的任意字段 */
          echo "Good Afternoon!";;
          *)
          echo "Sorry, $YES_OR_NO not recognized. Enter
      yes or no."
          exit 1;;
    esac
```

例子 2：编写一个加减乘除取模计算器

```
   echo "please input the first number:"
       read a
       echo "please input the second number:"
       read b
       echo "please input your operator:"
       read c
       case $c in
       "+")
       echo "the result of $a + $b is $(($a + $b))"
       ;;
       "-")
       echo "the result of $a - $b is $(($a - $b))"
       ;;
       "*")
       echo "the result of $a * $b is $(($a * $b))"
       ;;
       "/")
       echo "the result of $a / $b is $(($a / $b))"
       ;;
       *)
       echo "no true operator!"
       ;;
```

```
esac
```

2.6 for 循环

例子 1：

```
for x in one two three four
do
    echo number $x
done
```

例子 2：

```
for x in /etc/????.???? /var/lo* /home/* ${PATH} //列举
do
   echo $x
done
```

例子 3：/etc/r*中的文件和目录

```
for myfile in /etc/r*
do
   if [ -d "$myfile" ]
   then
       echo "$myfile(dir)"
   else
       echo "$myfile"
   fi
done
```

例子 4：

```
for x in /var/log/*
do
   echo `basename $x` is a file living in /var/log
done
```

例子 5：//冒泡排序

```
#!/bin/bash
a=(3 10 6 5 9 2 8 1 4 7)
for (( i=1; i<10; i++ ))
do
    for (( j=0; j<10-i; j++ ))
```

```
        do
            if [ ${a[j]} -gt ${a[j+1]} ]
            then
                temp=${a[j]}
                a[j]=${a[j+1]}  //或者 a[j]=${a[$(($j+1))]}
                a[j+1]=$temp
            fi
        done
    done

    for (( i=0; i<10; i++ ))
    do
        echo ${a[i]}
    done
```

2.7 While 语句

```
    myvar=0
    while [ $myvar -ne 10 ]
    do
        echo $myvar
        myvar=$(($myvar+1))
    done
```

举例：

```
    #!/bin/bash
        #this is my first shell project
    loopcount=0
    result=0
    while [ $loopcount -lt 100 ]
    do
        loopcount=$(($loopcount + 1))
        result=$(($loopcount + $result))
    done
    echo "The result of \'1+2+3+...+100\' is $result"
```

2.9 until 语句

```
myvar=0
until [ $myvar -eq 10 ]
do
    echo $myvar
    myvar=$(($myvar+1))
done
```

3 Shell 函数

函数名(){ 命令 1 ... ... }

function 函数名() { ... ... }

```
#declare a function named hello
function hello()
{
   echo "Hello,$1 today is `date` "
   return 11
}
echo "now going to the function hello"
hello  "I LOVE CHINA"
echo $?
echo "back from the function "
```

例 2：

实现两个数相加

C 语言实现:

```
#include <stdio.h>
int add(int a, int b)
{
     return a + b;
}
int main()
{
     int a = 10;
     int b = 20;
     int c = add(a, b);
```

```
        printf("%d\n", c);
        return 0;
    }
```

Shell 实现：

```
    #!/bin/bash
    function add()
    {
        return $(($1+$2))
    }
    a=10
    b=20
    add a b
    echo $?
```

•