



# 实验一

## LINUX环境和GCC工具链

# Outline

---

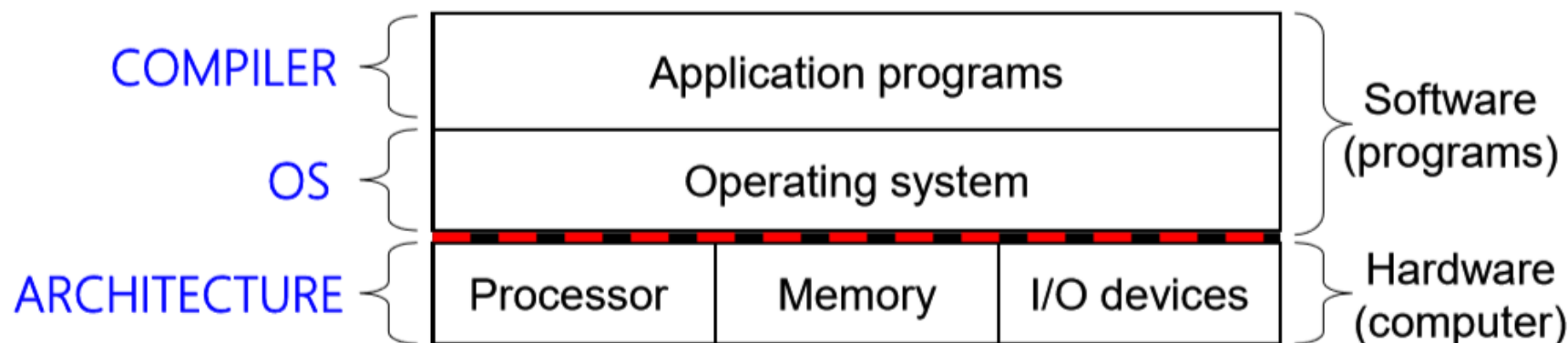
**Linux**概述及实验环境

Linux操作命令

GCC工具链

# 操作系统概念

- 操作系统（Operating System, OS）是管理和控制计算机硬件与软件资源的计算机程序，是直接运行在“裸机”上的最基本的系统软件。



# 操作系统—Windows、macOS

目前最流行的个人桌面操作系统：Windows、macOS



# 操作系统—Linux

---

**服务器端**，Linux市场份额一直在增长。Linux非常稳定，特别适合大型企业生产环境。

作为网络平台的后台服务器被使用：门户网站（搜狐、新浪、网易等）、电商平台（淘宝、QQ商城等）大部分使用Linux操作系统；

作为应用服务器、数据库服务器被使用：解决海量数据、高并发的问題；

作为嵌入式操作系统被使用：智能控制、自动化、物联网等领域。

# Linux历史-追溯到UNIX（1）

---

20世纪60年代是大型、复杂操作系统盛行的年代，比如IBM的OS/360和Honey-well的**Multics**系统。OS/360是历史上最成功的软件项目之一，而Multics却从来没被广泛应用过。贝尔实验室曾经是Multics项目的最初参与者，但由于诸多问题于1969年退出。鉴于Multics项目不愉快的经历，贝尔实验室的研究人员从1969年开始在DEC PDP-7计算机上完全用机器语言编写了一个简单的操作系统，该系统中的很多思想，比如层次文件系统、作为用户级进程的shell概念，都来自于Multics，只不过在一个更小、更简单的程序包里实现。1970年，Brian Kernighan给这个新系统命名为“**Unix**”，这是一个双关语，暗指“Multics”的复杂性。1973年用C重新编写内核，1974年发布。

# Linux历史-追溯到UNIX（2）

---

- 虽然UNIX免费提供，但获取源代码需要向AT&T交纳许可证费用。1977年，加州大学伯克利分校的计算机系统研究小组从AT&T获取了UNIX的源代码，经过改动和包装后发布伯克利UNIX（**Berkeley UNIX**），通常被称为BSD，代表Berkeley Software Distribution。
- 随着UNIX在商业的蓬勃发展，AT&T的许可证费用也水涨船高。伯克利于是决定从BSD中彻底除去AT&T的代码。到了1989年6月，一个完全没有AT&T代码的BSD版本诞生了。
- FreeBSD、OpenBSD等都是由BSD发展过来。
- 与此同时，另一些UNIX版本则沿用了AT&T的代码，这些UNIX系统的操作系统包括HP-UX、Solaris。

# Linux历史-追溯到UNIX (3)

---

- 因为AT&T的政策改变，在Version 7 Unix推出之后，发布新的使用条款，将UNIX源代码私有化，在大学中不再能使用UNIX源代码。**Tanenbaum**教授为了能在课堂上教授学生操作系统运作的细节，决定在不使用任何AT&T的源代码前提下，自行开发与UNIX兼容的操作系统，以避免版权上的争议。他以小型UNIX（mini-UNIX）之意，将它称为MINIX。
- MINIX发布于1987年，是荷兰阿姆斯特丹的Vrije大学计算机科学系的Andrew S. Tanenbaum教授所发展的一个类Unix操作系统。



# Linux历史（1）

到1991年，GNU计划已经开发出了许多工具软件。GNU C编译器已经出现，但还没有免费的GNU操作系统。MINIX也具有版权，需要购买才能得到源代码。而GNU的操作系统HURD一直在开发之中，并且在几年内都不能完成。



对于Linus来说，已经不能等待了。从1991年4月份起，他开始酝酿并着手编制自己的操作系统。

**Linus Torvalds**

芬兰、赫尔辛基大学

**旁注** Linux 项目

1991 年 8 月，芬兰研究生 Linus Torvalds 谨慎地发布了一个新的类 Unix 的操作系统内核，内容如下。

来自：torvalds@klaava.Helsinki.FI(Linus Benedict Torvalds)

新闻组：comp.os.minix

主题：在 minix 中你最想看到什么？

摘要：关于我的新操作系统的小调查

时间：1991 年 8 月 25 日 20:57:08 GMT

每个使用 minix 的朋友，你们好。

我正在做一个(免费的)用在 386(486)AT 上的操作系统(只是业余爱好，它不会像 GNU 那样庞大和专业)。这个想法自 4 月份就开始酝酿，现在快要完成了。我希望得到各位对 minix 的任何反馈意见，因为我的操作系统在某些方面与它相类似(其中包括相同的文件系统的物理设计(因为某些实际的原因))。

我现在已经移植了 bash(1.08)和 gcc(1.40)，并且看上去能运行。这意味着我需要几个月的时间来让它变得更实用一些，并且，我想要知道大多数人想要什么特性。欢迎任何建议，但是我无法保证我能实现它们。:-)

Linus (torvalds@kruuna.helsinki.fi)

就像 Torvalds 所说的，他创建 Linux 的起点是 Minix，由 Andrew S. Tanenbaum 出于教育目的开发的一个操作系统[113]。

接下来，如他们所说，这就成了历史。Linux 逐渐发展成为一个技术和文化现象。通过和 GNU 项目的力量结合，Linux 项目发展成了一个完整的、符合 Posix 标准的 Unix 操作系统的版本，包括内核和所有支撑的基础设施。从手持设备到大型计算机，Linux 在范围如此广泛的计算机上得到了应用。IBM 的一个工作组甚至把 Linux 移植到了一块腕表中！

# Linux历史（2）

---

- 1991年的10月5日，Linus在comp.os.minix新闻组上发布消息，正式向外宣布Linux（Linus' Minix, Linux）内核系统的诞生。10月5日对Linux社区来说是一个特殊的日子，许多后来Linux的新版本发布时都选择了这个日子
- 1994年3月14日，历经过无数的修订后，Linux推出了第一个正式的核心版本1.0并正式转向GPL协议
- Linux核心版本的发展走入了正轨。简单地说，Linux是对UNIX的重新实现。世界各地的Linux开发人员借鉴了UNIX的技术和用户界面，并且融入了很多独创的技术。Linux不属于BSD和AT&T风格的UNIX中的任何一种。因此严格来说，Linux是有别于UNIX的另一种操作系统

# Linux发行版举例（1）

## Ubuntu





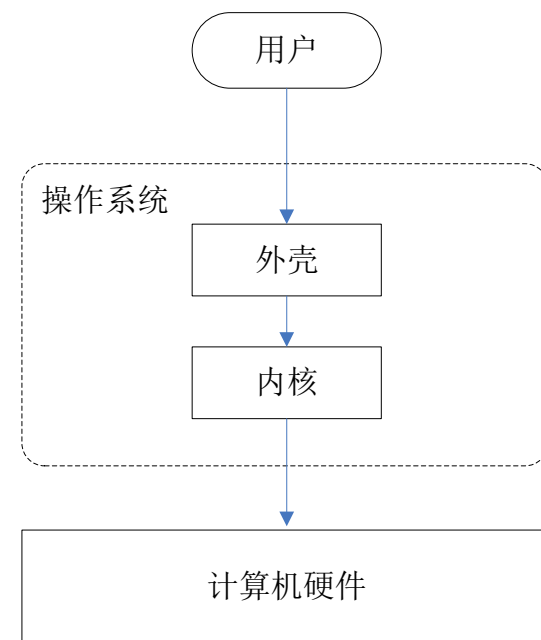
# Linux发行版举例（2）



# Shell

## 操作系统分内核与外壳

- 外壳程序对用户输入命令进行解释， 为用户提供一种通过操作系统使用计算机的操作环境
- Windows的图形界面（GUI）， 由一个称为Explorer的模块解释用户的输入
- Linux的命令行界面（CLI）， 称为Linux shell

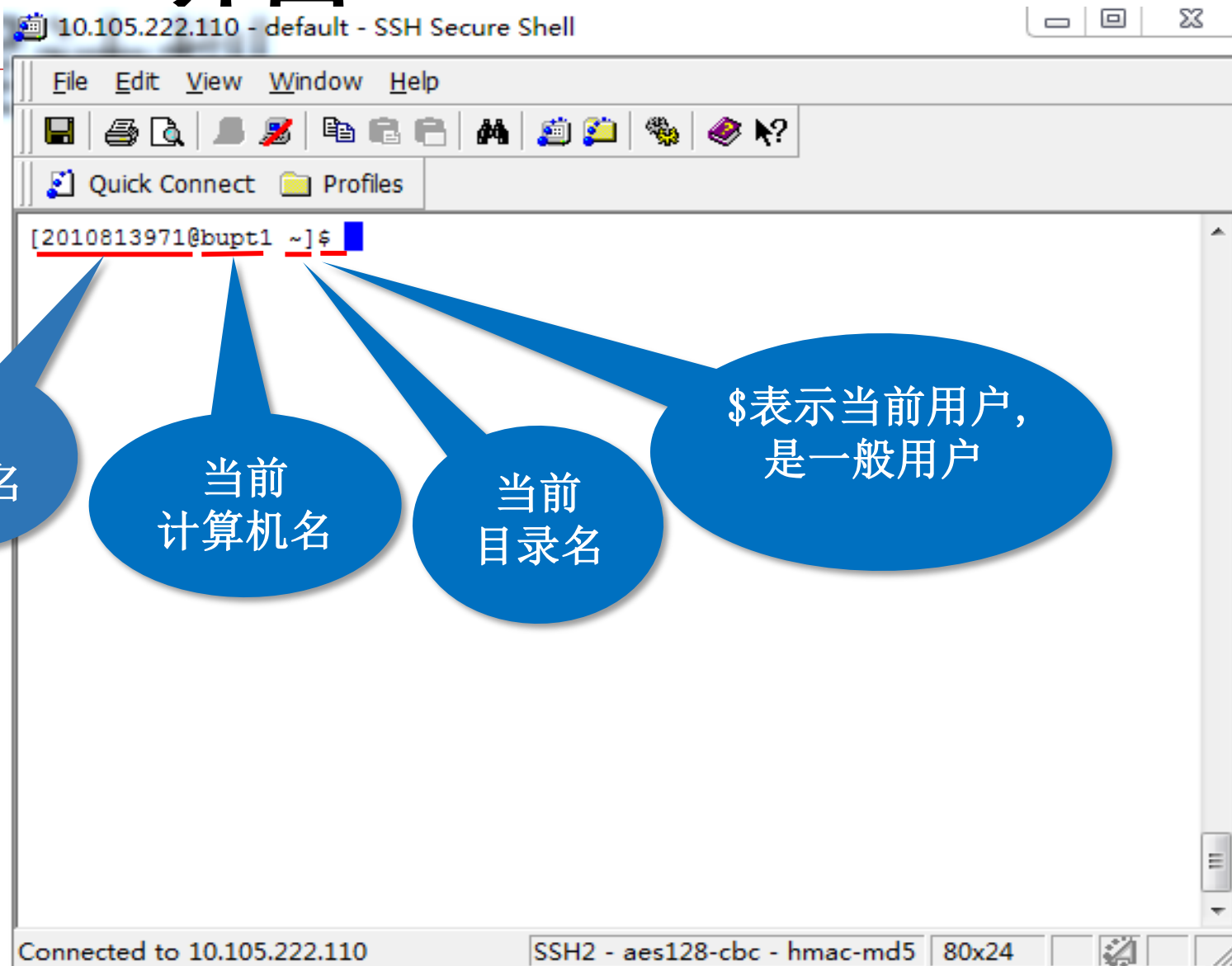


# Shell程序

---

- 又称shell脚本
- 把一系列的shell命令，按照一定的语法规则和控制结构，组织在一个文件中，然后由内核来一条接着一条地解释和执行这些命令，这个文件就是一个shell程序
- 类似于DOS/Windows中的.bat批处理文件

# Shell界面





# Shell命令

- 从命令行输入语句，每输入一次就能得到一次响应，这些语句就是shell命令

```
frank@ubuntu: ~/code
/home/frank
frank@ubuntu: ~$ cd code
frank@ubuntu: ~/code$ ls -l
total 1940
-rw-rw-r-- 1 frank frank 132 Sep 25 10:42 absdiff.c
-rw-rw-r-- 1 frank frank 186 Aug 17 02:16 absdiffgo.c
-rw-rw-r-- 1 frank frank 367 Sep 25 10:45 absdiffgo.s
frank@ubuntu: ~/code$
-rw-rw-r-- 1 frank frank 345 Sep 25 10:42 absdiff.s
-rwxrwxr-x 1 frank frank 8368 Sep 27 08:58 a.out
-rwxrwxr-x 1 frank frank 8384 Jul 31 09:58 bytecmp
-rw-rw-r-- 1 frank frank 147 Jul 31 09:55 bytecmp.c
-rwxrwxr-x 1 frank frank 8344 Aug 2 13:03 float
-rw-rw-r-- 1 frank frank 278 Aug 2 13:03 float.c
-rwxrwxr-x 1 frank frank 8416 Aug 17 07:39 hello
-rw-rw-r-- 1 frank frank 165 Aug 17 06:49 hello.c
-rw-rw-r-- 1 frank frank 1856 Aug 17 07:39 hello.o
-rw-rw-r-- 1 frank frank 765 Aug 17 07:39 hello.s
-rw-rw-r-- 1 frank frank 126 Jul 20 08:22 i2float.c
-rwxrwxr-x 1 frank frank 8368 Sep 27 08:55 sum
-rw-rw-r-- 1 frank frank 190 Sep 21 10:00 sum.c
-rw-rw-r-- 1 frank frank 1688 Sep 27 07:57 sum.o
-rw-rw-r-- 1 frank frank 948 Sep 27 08:54 sum.s
```

# Shell程序实例

- 用vi编辑器创建程序文件，键入若干命令并保存，修改文件属性为可执行。
- 执行该程序

## ashell脚本文件

```
pwd
mkdir zlab1
echo "hello" > ./zlab1/123
ls -l ./zlab1
```

```
frank@bupt1
pwd
mkdir zlab1
echo "hello" > ./zlab1/123
ls -l ./zlab1
~
~
~
~
~
~
~
frank@bupt1:~/code$ chmod 755 ashell
frank@bupt1:~/code$ ./ashell
/home/teacher/frank/code
total 4
-rw-r--r-- 1 frank teacher 6 Oct  8 12:20 123
frank@bupt1:~/code$
```

# 如何远程登录Linux (1)

## ■ 使用SSH命令

### ◆ Windows

打开Windows PowerShell窗口，键入

`ssh 2021123456@10.120.11.12` ↻

输入密码 ↻

学号

服务器IP:10.120.11.12

用户名:学号

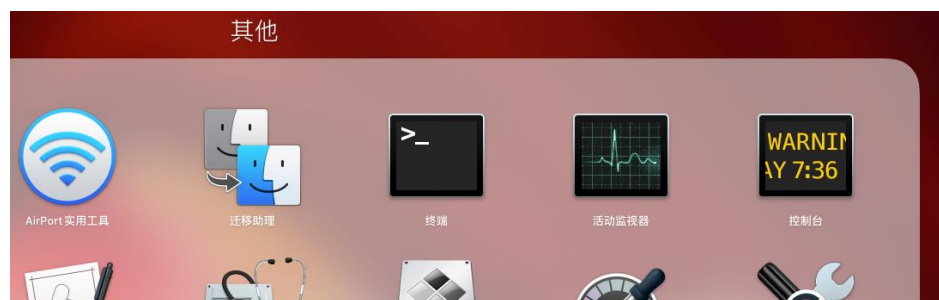
默认密码:bupt学号

### ◆ macOS

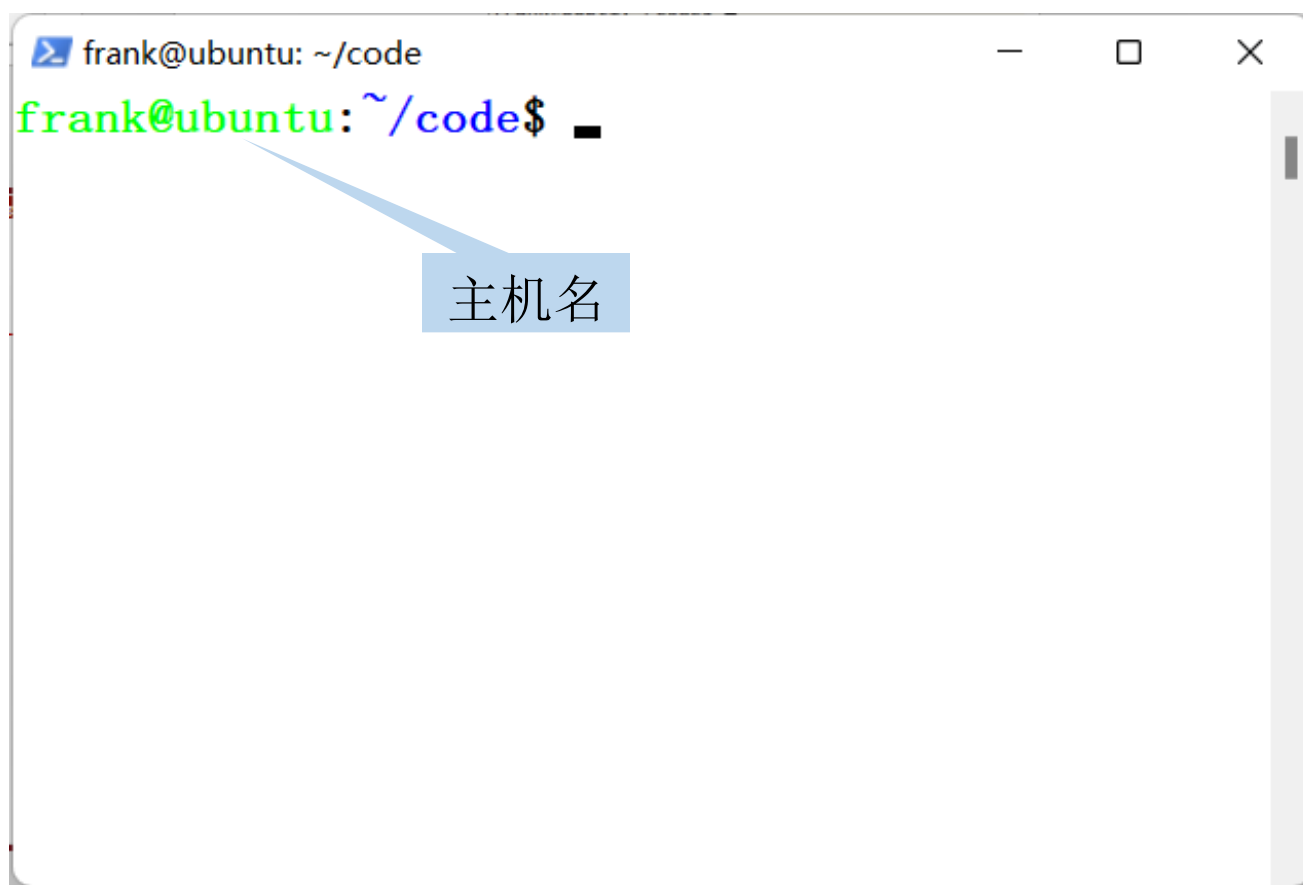
打开终端

`ssh 2021123456@10.120.11.12` ↻

输入密码 ↻



# 成功登录（使用Windows PowerShell）



A terminal window titled "frank@ubuntu: ~/code" with standard window controls. The prompt "frank@ubuntu: ~/code\$" is displayed in green and blue. A blue callout box with the text "主机名" (hostname) points to the "frank@ubuntu" portion of the prompt.

```
frank@ubuntu: ~/code$
```

主机名

# 注意事项

---

- 课后**尽快**修改账户的默认密码
  - 1、登录账户
  - 2、键入passwd命令
  - 3、按提示输入新密码
  - 4、记住自己的密码，忘记后需要系统管理员进行重置
  
- bupt1是为同学们提供课程实验的服务器，如果有同学在bupt1上**进行恶意行为**（如:进行提权、恶意占用CPU时间、内存空间和磁盘空间等），进而影响实验平台的正常运行，一经发现，该同学的实验课程**成绩为0分**！

# Outline

---

Linux概述及实验环境

**Linux操作命令**

GCC工具链

# Shell常用命令

---

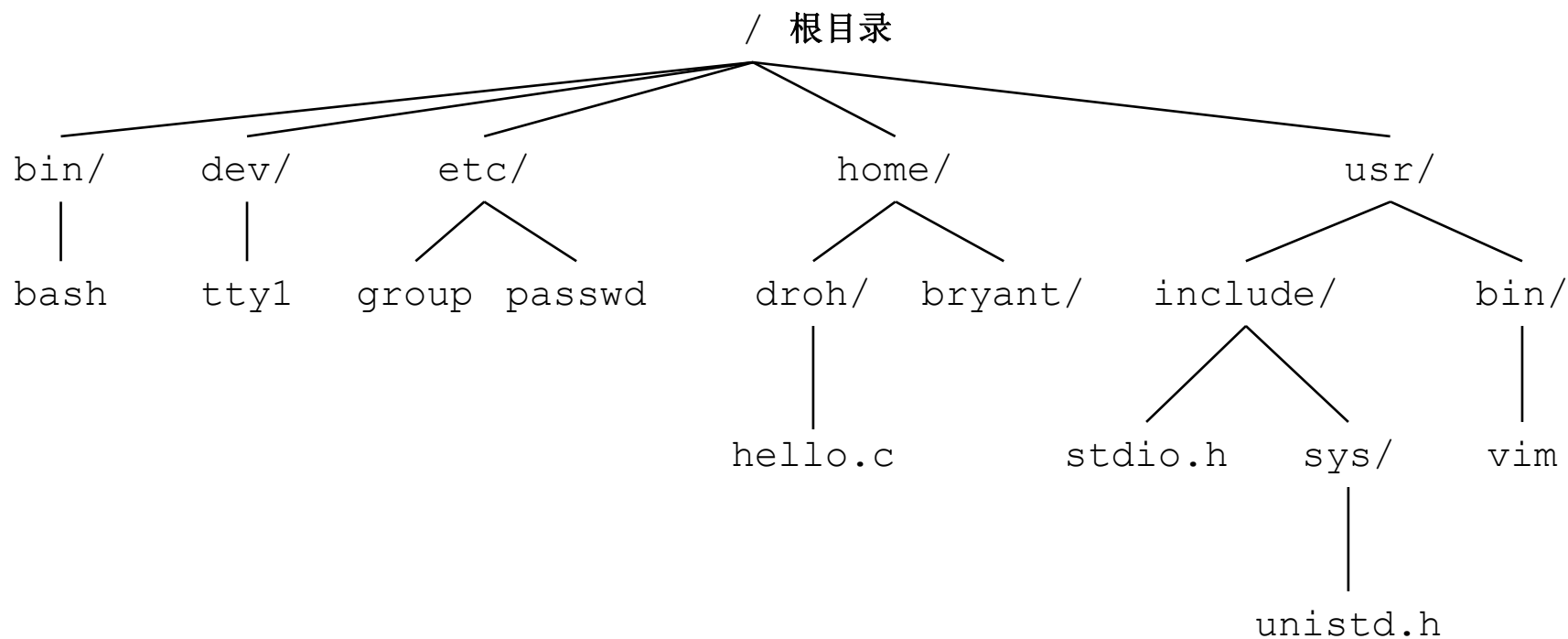
## ■ 目录操作命令

- ◆ 目录操作命令是指能够对目录进行查看、创建、删除，以及显示当前工作目录和改变当前目录等操作

## ■ 文件操作命令

- ◆ 在命令行环境下对文件进行操作将比在图形环境下操作文件更加快捷和高效
- ◆ 文件操作主要包括搜索文件、复制和移动文件、删除文件以及合并文件的内容等

# 目录层次结构

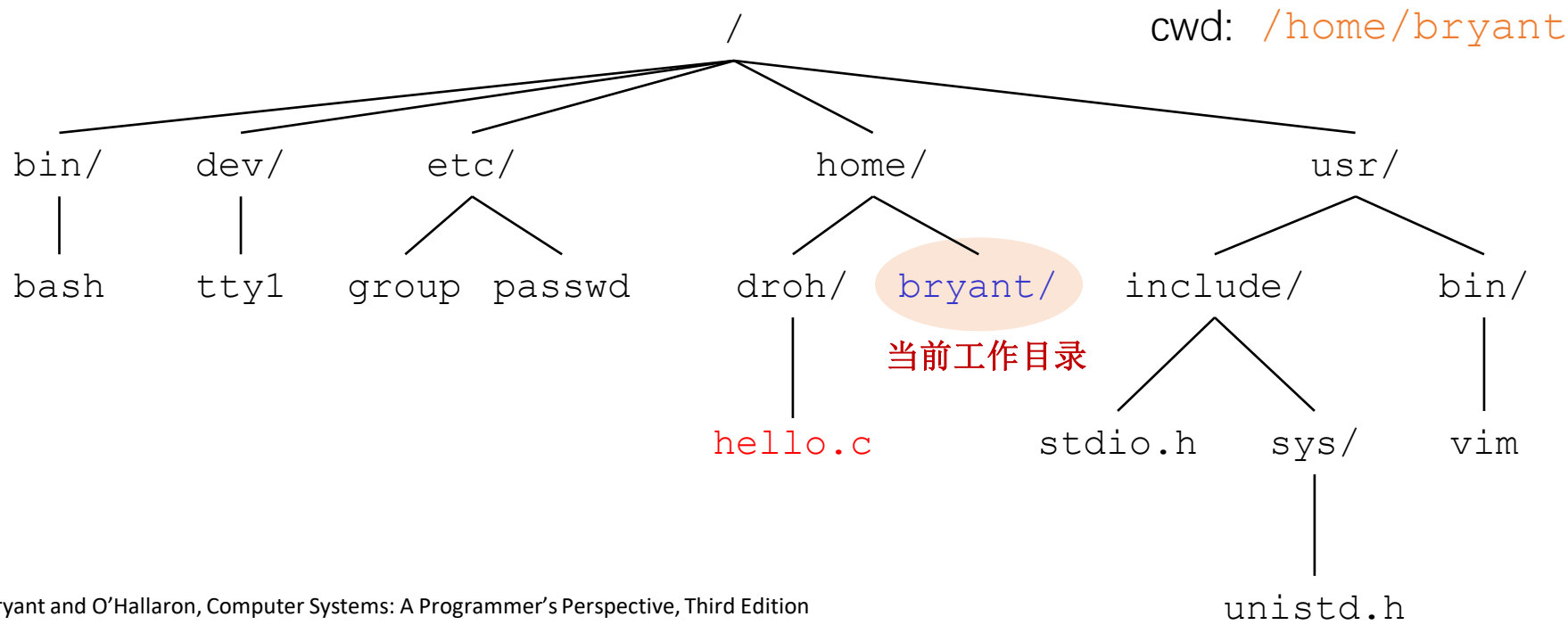




# 路径名

## ■ Locations of files in the hierarchy denoted by *pathnames*

- ◆ 绝对路径名（从根目录 / 开始表示的路径）
  - `/home/droh/hello.c`
- ◆ 相对路径名（从当前工作目录出发表示路径）
  - `../droh/hello.c`



# Linux文件系统（1）

---

■ 根目录下：/etc、/dev、/boot、/home、/lib、/lost+found、/mnt、/opt、/proc、/root、/bin、/sbin、/tmp、/var、/usr等

## 1. /etc

存放着许多系统所需的重要配置与管理文件

## 2. /dev

存放device file（设备文件），使用者可以经由内核用来存取系统中的硬件设备，当使用设备文件时内核会识别出输入输出请求，并传递到相对应设备驱动程序以便完成特定的动作

## 3. /boot

存放与系统启动的相关文件，不可任意删除

## 4. /home

登录用户的主目录（\$HOME）放在该目录下，以用户的名称作为/home目录下各个子目录的名称

# Linux文件系统（2）

---

- 根目录下：`/etc`、`/dev`、`/boot`、`/home`、`/lib`、`/lost+found`、`/mnt`、`/opt`、`/proc`、`/root`、`/bin`、`/sbin`、`/tmp`、`/var`、`/usr`等

## 5. `/lib`

存放许多系统启动时所需要的重要的共享函数库

## 6. `/usr/lib`

存放一些应用应用程序的共享函数库，例如Netscape、X server等。最重要的函数库为libc或glibc（glibc 2.x便是libc 6.x版本，标准C语言函数库）及文件名为library.a的静态函数库

## 7. `/mnt`

系统默认的挂载点（mount point），默认有  
`/mnt/cdrom`和`/mnt/floppy`

## 8. `/proc`

虚拟文件系统，它不占用硬盘空间，目录下文件均放置于内存中；`/proc`记录系统进程，硬件状态、内存使用等信息。

# Linux文件系统（3）

---

- 根目录下：/etc、/dev、/boot、/home、/lib、/lost+found、/mnt、/opt、/proc、/root、/bin、/sbin、/tmp、/var、/usr等

## 9. /root

系统管理用户root的主目录

## 10. /bin

存放一些系统启动时所需要的普通程序和系统程序及一些经常被其它程序调用的程序

## 11. /tmp

存放系统启动时产生的临时文件

## 12. /var

本目录存放被系统修改过的数据。在这个目录下的重要目录有/var/log、/var/spool、/var/run等，它们分别用于存放日志文件、新闻邮件、运行时信息。

# (1) 常用目录操作命令

命令	功能
<b>pwd</b>	打印当前工作目录
<b>cd</b>	改变当前所在目录
<b>ls</b>	查看目录下的内容
<b>dir</b>	类似 <b>ls</b> 命令
<b>mkdir</b>	创建目录
<b>rmdir</b>	删除空目录
<b>rm -r</b>	删除目录

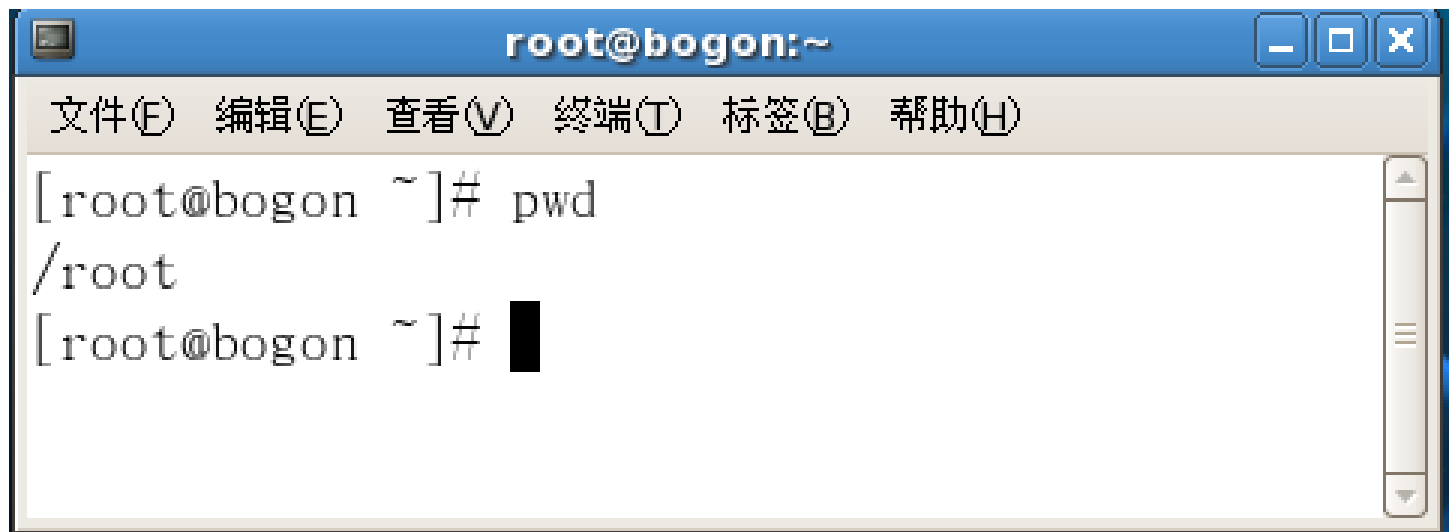
# pwd命令 (1)

**pwd(print working directory):打印工作目录**

**【功能】** 显示当前工作目录的整个路径。

**【用法】** 直接在Shell提示符#或\$后输入命令**pwd**，然后按回车。

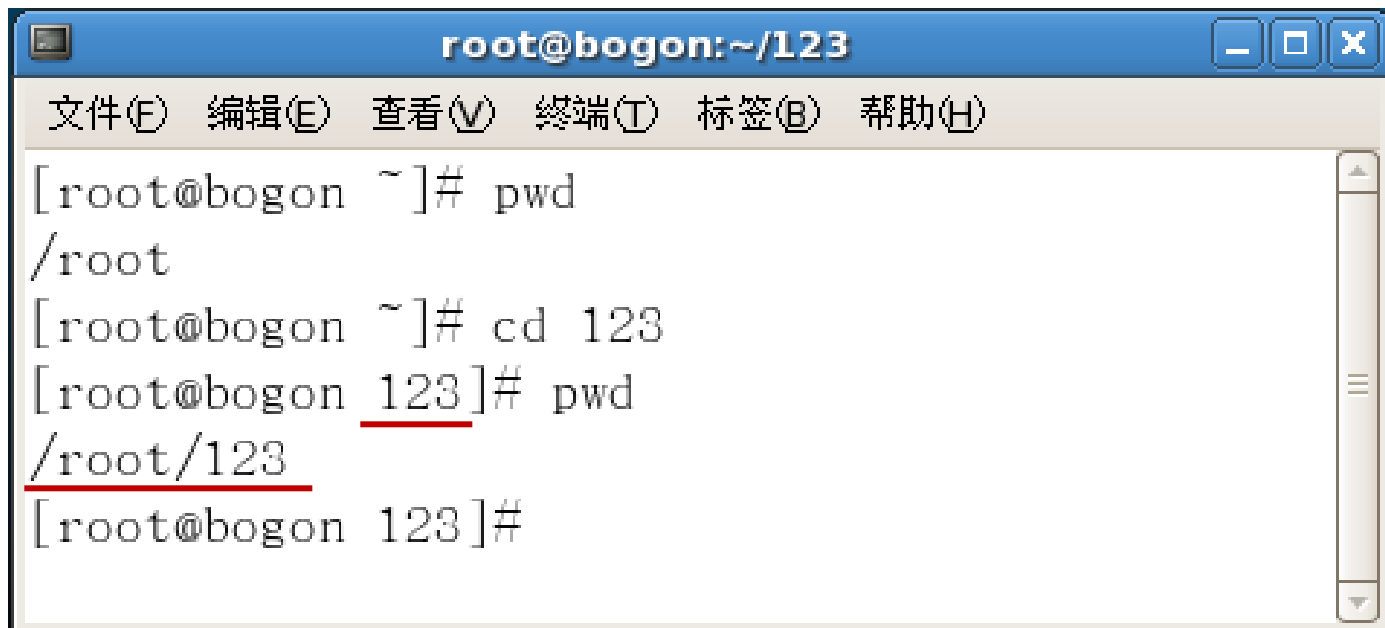
**【例如】**

A screenshot of a terminal window titled 'root@bogon:~'. The window has a menu bar with options: 文件(F), 编辑(E), 查看(V), 终端(T), 标签(B), 帮助(H). The terminal content shows the command 'pwd' being executed, resulting in the output '/root'. The prompt '[root@bogon ~]#' is visible before and after the command.

```
root@bogon:~  
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)  
[root@bogon ~]# pwd  
/root  
[root@bogon ~]#
```

# pwd命令 (2)

【注意】“**当前目录名**”跟“**显示当前工作目录**”是不同的。前者只是显示目录名字，后者显示整个路径。

A terminal window titled 'root@bogon:~/123' with standard window controls. The menu bar includes '文件(F)', '编辑(E)', '查看(V)', '终端(T)', '标签(B)', and '帮助(H)'. The terminal shows the following sequence of commands and outputs:

```
[root@bogon ~]# pwd
/root
[root@bogon ~]# cd 123
[root@bogon 123]# pwd
/root/123
[root@bogon 123]#
```

The output of the second 'pwd' command, '/root/123', is underlined in red to illustrate that it shows the full path.

# cd命令

---

【功能】 改变所在目录。

【用法】 可以直接写cd或cd ~，表示回到主目录，也可在后面加上要转移到的目的目录及其路径。

【格式】 cd [路径/目录名]

- ◆绝对路径：从根目录开始写，以“/”打头。
- ◆相对路径：当前目录的子级目录开始写。

【例如】

```
cd /root/123/456
```

```
cd 123/456
```

```
cd ..
```

```
cd /
```



# 示例：cd命令的使用

A terminal window titled 'root@bogon:~/123/456' with standard window controls. The menu bar includes '文件(F)', '编辑(E)', '查看(V)', '终端(T)', '标签(B)', and '帮助(H)'. The terminal text shows a sequence of 'cd' commands: starting at '~', then '456' (which fails with 'bash: cd: 456: 没有那个文件或目录'), then '/root/123/456', then '456', then '..' (moving to '123'), then '..' (moving back to '~'), then '123/456', and finally '456' again.

```
root@bogon:~/123/456
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
[root@bogon ~]# cd
[root@bogon ~]# cd 456
bash: cd: 456: 没有那个文件或目录
[root@bogon ~]# cd /root/123/456
[root@bogon 456]#
[root@bogon 456]# cd ..
[root@bogon 123]#
[root@bogon 123]# cd ..
[root@bogon ~]# cd 123/456
[root@bogon 456]#
```

# ls命令（1）

---

**【功能】** 查看某个目录中的内容。

**【用法】** 直接写ls，或在后面加上选项。

**【格式】** ls [选项] [目录]

**【例如】**

**ls**

**ls -a /root/123**

**ls -l /root/aaa.txt**

# ls命令 (2)

## ■ 各个选项及其功能:

选项	功能
无	显示指定目录中所有内容，不包括隐藏的。
<b>-a</b>	显示指定目录中所有内容，包括隐藏的。
<b>-l</b>	显示所有内容的细节，包括权限、所有者、群组、大小、创建日期、是否链接文件。
<b>-F</b>	显示文件类型， /、@、*。
<b>-r</b>	按字母表逆序显示。
<b>-R</b>	递归显示。
<b>-s</b>	按大小排序显示。
<b>-t</b>	按修改时间排序显示

# 示例：ls命令的使用（1）



A terminal window titled 'root@localhost' with a menu bar containing '文件(F)', '编辑(E)', '查看(V)', '终端(T)', '转到(G)', and '帮助(H)'. The terminal shows the execution of the 'ls' and 'ls -a' commands. The output of 'ls' lists '123', 'bohao', and 'install.log.syslog'. The output of 'ls -a' lists hidden files and directories in three columns.

```
root@localhost~  
文件(F) 编辑(E) 查看(V) 终端(T) 转到(G) 帮助(H)  
[root@localhost root]# ls  
123          bohao          install.log.syslog  
anaconda-ks.cfg  install.log  
[root@localhost root]# ls -a  
.          .gconfd          .mozilla  
..         .gnome           .nautilus  
123        .gnome2          .python  
anaconda-ks.cfg .gnome2_private .recently-used  
.bash_history .gnome-desktop .rhn-applet.conf  
.bash_logout .gstreamer      .tcshrc  
.bash_profile .gtkrc          .Trash  
.bashrc      .gtkrc-1.2-gnome2 .Xauthority  
bohao       .ICEauthority   .Xresources  
.cshrc       install.log     .xsession-errors  
.fonts.cache-1 install.log.syslog  
.gconf       .metacity  
[root@localhost root]#
```

# 示例：ls命令的使用（2）

- ls可以同时选择多个选项：


```

root@localhost: ~
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)

[root@localhost root]# ls -al
总用量 220
drwxr-x--- 16 root    root    4096  3月 26 14:15 .
drwxr-xr-x 19 root    root    4096  3月 26 13:31 ..
drwxr-xr-x  3 root    root    4096  3月 26 14:34 123
-rw-r--r--  1 root    root    1265  3月  6 07:52 anaconda-ks.cfg
-rw-----  1 root    root     194  3月 26 15:37 .bash_history
-rw-r--r--  1 root    root      24 2000-06-11 .bash_logout
-rw-r--r--  1 root    root    234 2001-07-06 .bash_profile
-rw-r--r--  1 root    root    176 1995-08-24 .bashrc
drwxr-xr-x  2 root    root    4096  3月 21 23:12 bohao
-rw-r--r--  1 root    root     210 2000-06-11 .cshrc
-rw-r--r--  1 root    root   62569  3月  8 00:28 .fonts.cache-1
drwx-----  5 root    root    4096  3月 26 13:33 .gconf
drwx-----  3 root    root    4096  3月 26 15:25 .gconfd
drwx-----  5 root    root    4096  3月  5 23:57 .gnome
drwxr-xr-x  5 root    root    4096  3月 21 23:17 .gnome2
  
```

# 示例：ls命令的使用（3）

- 显示指定目录中的内容：

A screenshot of a terminal window titled 'root@localhost ~'. The window has a menu bar with options: 文件(F), 编辑(E), 查看(V), 终端(T), 转到(G), and 帮助(H). The terminal shows two commands being executed. The first command is '[root@localhost root]# ls -a /root/123', which outputs '. .. 456'. The second command is '[root@localhost root]# ls -a /root/bohao', which outputs '. .. yhnetclient253\_3.1.2.exe'. The prompt '[root@localhost root]#' is followed by a black cursor block.

```
root@localhost ~
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
[root@localhost root]# ls -a /root/123
.  ..  456
[root@localhost root]# ls -a /root/bohao
.  ..  yhnetclient253_3.1.2.exe
[root@localhost root]#
```

# dir命令

---

【功能】类似ls，但选项较少，用于查看某个目录中的内容。

【格式】**dir** [选项] [目录]

【例如】

**dir**

**dir -a /root/123**

**dir -l /root/aaa.txt**

# mkdir命令

---

【功能】创建目录，只能在已存在的目录中创建新的目录。

【格式】**mkdir** [选项] [目录]

**-p**: 在创建目录时，如果父目录不存在，则同时创建该目录及该目录的父目录。

【例如】

```
mkdir -p /root/abc/123
```



# rmmdir命令

---

【功能】删除空目录。

【格式】rmmdir [选项] [目录]

**-p:** 在删除目录时，一起删除父目录，但父目录中必须没有其他目录及文件。

【例如】

```
rmmdir -p /root/abc/123
```

# rm -r命令

---

【功能】 删除目录。

【格式】 **rm -r [目录]**

【例如】

**rm -r /root/abc/123**

## (2) 文件操作命令

命令	功能
<b>cat</b>	查看文件的内容。
<b>more</b>	分页查看，空格键下一页，b键上一页。
<b>less</b>	分页查看，类似more，也可方向键滚动显示。
<b>head</b>	查看文件的前面部分，默认是前10行。
<b>tail</b>	查看文件的后面部分，默认是后10行。
<b>cp</b>	复制文件。
<b>mv</b>	移动文件或重命名。
<b>rm</b>	删除文件。
<b>find</b>	检索文件。
<b>touch</b>	创建一个空白文件。
<b>ln</b>	创建链接文件。

# cat 命令

---

【功能】显示文件的内容，或合并文件内容，并重定向输出。

【用法】按Ctrl+D退出cat命令。

【格式】cat [选项] [文件名]

■ 选项：

- ◆-n：对所有输出行标注行号
- ◆-b：对所有非空行标注行号
- ◆>：重定向输出
- ◆<：重定向输入

**cat 用于查看全文内容，不能分页查看**

# cat使用示例

---

**cat** //将键盘输入的内容重新显示到屏幕

**cat > aa** //将键盘输入的内容重定向输出到指定文件中

**cat aa** //将指定文件的内容显示到屏幕。

**cat < aa** //从指定文件中获取内容作为输入信息，然后显示到屏幕。

**cat aa bb > cc** //合并a， b中的内容，然后重定向输出到c。

**cat -n aa bb > cc**

# more命令

---

【功能】 分页查看，空格键下一页，b键上一页。

【格式】 `more [选项] 文件名`

选项：

**-num:** 一次显示的行数

**+num:** 从第几行开始显示

【例如】

`more .bash_history`

`more -10 .bash_history`

`more +5 .bash_history`

# less命令

---

【功能】分页查看，类似more，也可方向键滚动显示，按q键结束浏览。读取速度较快。

【格式】less [选项] 文件名

【例如】

```
less .bash_history
```

# head、tail命令

---

**【功能】** 显示文件的前部分/后部分，默认情况下，只显示前10行/后10行。

**【格式】** head/tail [选项] 文件名

选项：

**-num:** 显示指定文件的前/后 num行

**【例如】**

**head .bash\_history**

**tail .bash\_history**

**tail -n .bash\_history**



# cp命令

---

**【功能】** 复制文件或目录到指定的目录或文件。

**【格式】** `cp [选项] 源文件或目录 目标文件或目录`

选项:

**-i:** 交互式，询问是否覆盖。

**-r:** 递归复制。

**【例如】**

```
cp aa.txt /root/123
```

```
cp -i aa.txt /root/123
```

```
cp -i aa.txt /root/123/bb.txt
```

# mv命令

---

**【功能】** 移动文件或目录到指定的文件或目录

**【格式】** `mv [选项] 源文件或目录 目标目录或文件`

**【例如】**

`mv aa.txt /root/123`

# rm命令

---

【功能】删除指定的一个或多个文件。

【格式】rm [选项] 文件名

【例如】

```
rm -i aa.txt
```

```
rm -i aa.txt bb.txt
```

# find命令

---

【功能】检索某个或某些特定的文件，可根据名称、类型等来检索。

【格式】`find [选项] [路径] 参数`

【例如】

`find -name "*.txt"`

`find -type d`

`find -type f`

# touch命令

---

【功能】 改变文件的时间记录，或创建一个空白文件。

【格式】 touch [选项] 文件名

选项：

无：如目标文件不存在，则建立新的文件。

-c：如目标文件不存在，不会建立新的文件。

-a：改变文件的读取时间记录。

-d：设定时间和日期。

【例如】

**touch dd.txt**

**touch -c ee.txt**

# ln命令

---

【功能】创建链接文件。

【格式】ln [选项] 源文件名 链接文件  
选项：

无：硬链接文件。

-s：软链接文件，符号链接文件

【例如】

```
ln aa /root/123/bb
```

```
ln -s aa /root/123/cc
```

# 编辑和修改shell命令

---

- 输入命令后，可通过左右方向键、**Home**键、**End**键等来移动光标，然后再配合**Delete**和**Backspace**键可对命令进行编辑和修改。

# 使用帮助命令

格式	使用范围	举例
<b>man</b> 命令名	所有命令	<b>man cat</b> <b>man man</b>
<b>whatis</b> 命令名	所有命令	<b>whatis echo</b> <b>whatis cat</b>
<b>help</b> 命令名	部分命令	<b>help echo</b> 不行: <b>help cat</b>
命令名 <b>--help</b>	部分命令	<b>cat --help</b> 不行: <b>echo --help</b>

按q键退出



# 历史命令和Tab自动补全

---

- **方向键**：使用上、下方向键，可自动选择以前使用过的命令。
  - ◆ 历史命令被保存在一个名为**.bash\_history**的文件中，它是当前主目录中的文件，可查看和阅读。
- **Tab键**：使用Tab键，可自动补全命令或路径。若有多个符合条件的选项，则可再次按Tab，输入下一个字符，再次按Tab，直至补全。

例如：`ls /root/Desktop/`  
`vi /etc/yum.r.../rhel...`

# Outline

---

Linux概述及实验环境

Linux操作命令

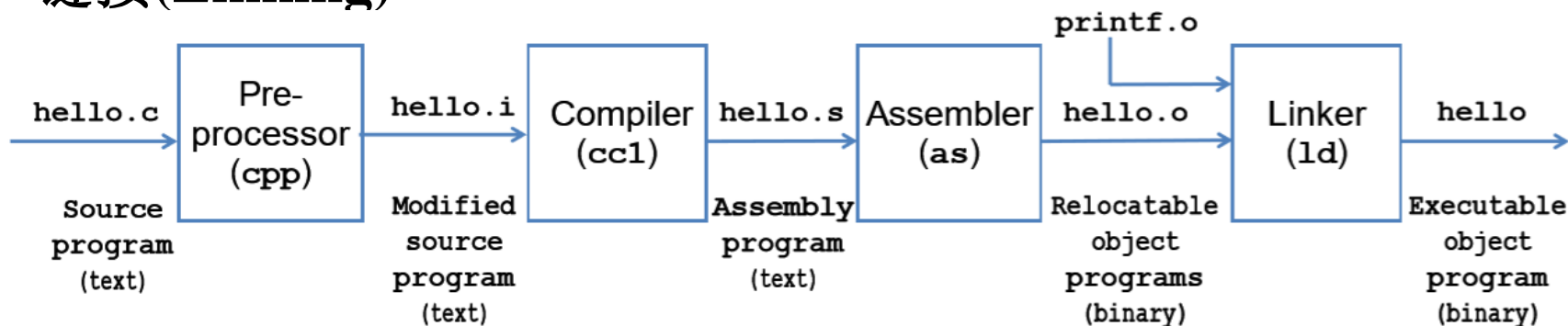
**GCC工具链**

# GCC编译器

- GCC编译器能将C、C++语言源程序、汇编程序**编译、链接**成可执行文件。在**Linux**系统中，**可执行文件没有统一的后缀**，系统从文件的属性来区分可执行文件和不可执行文件。

使用GCC编译程序时，编译过程可以被细分为四个阶段：

- 预处理(Pre-Processing)
- 编译(Compiling)
- 汇编(Assembling)
- 链接(Linking)



# 文件类型

---

- GCC通过后缀来区别输入文件的类别:
  - ◆ **c**为后缀的文件: C语言源代码文件
  - ◆ **a**为后缀的文件:是由目标文件构成的库文件
  - ◆ **.cc**或**.cxx**为后缀的文件:是C++源代码文件
  - ◆ **h**为后缀的文件:头文件
  - ◆ **i**为后缀的文件:是已经预处理过的C源代码文件
  - ◆ **ii**为后缀的文件:是已经预处理过的C++源代码文件
  - ◆ **o**为后缀的文件:是编译后的目标文件
  - ◆ **s**为后缀的文件:是汇编语言源代码文件
  - ◆ **S**为后缀的文件:是经过预编译的汇编语言源代码文件

# 起步（演示）

---

- **hello.c:**

```
#include<stdio.h>
```

```
int main(void)
```

```
{
```

```
printf>Hello world!\n);
```

```
return 0;
```

```
}
```

- **编译和运行这段程序:**

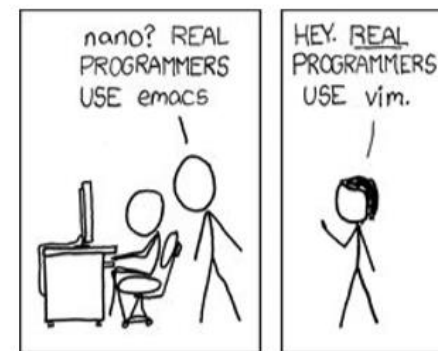
```
# gcc hello.c -o hello
```

```
# ./hello
```

- **输出: Hello world!**

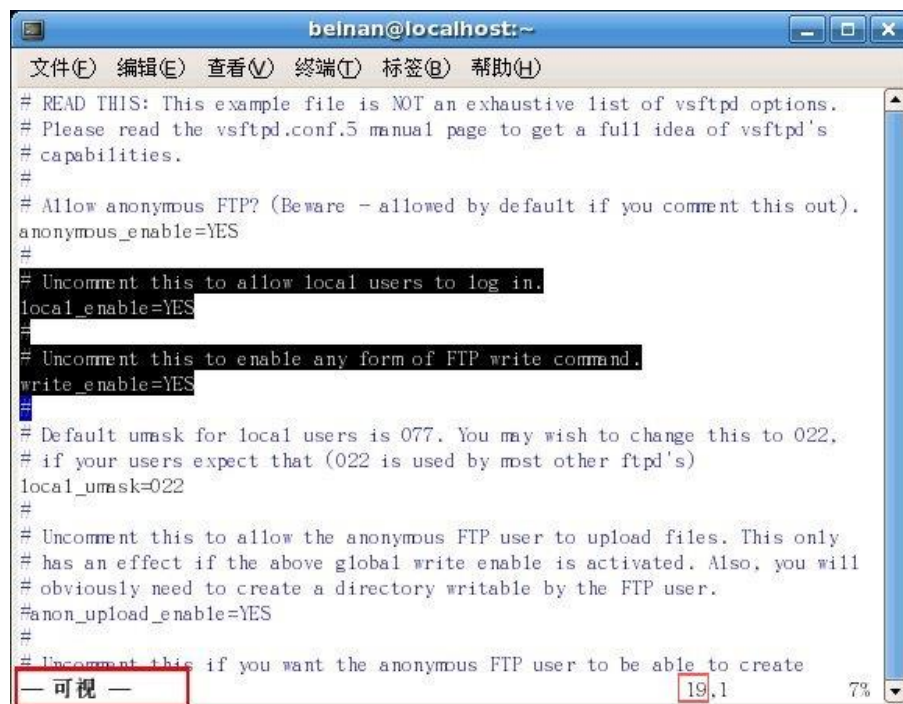
# vi编辑器

- **vi**是**Visual Interface**的简称，它可以执行输出、删除、查找、替换、块操作等众多文本操作
- 用户可以根据自己的需要对**vi**进行定制，这是其他编辑程序所没有的。
- **vi**不是一个排版程序，它不像**WORD**或**WPS**那样可以对字体、格式、段落等其他属性进行编排，它只是一个文本编辑程序
- **vi**是全屏幕文本编辑器，它没有菜单，只有命令
- **vim**则是**vi**的强化版**vi plus**，专业全称**vi improved**



# vi的启动

- 在系统提示符后输入vi和想要编辑（或建立）的文件名，便可进入vi
- 如果只输入vi，而不带文件名，也可以进入vi



The screenshot shows a terminal window titled "beinan@localhost:~". The window contains the text of a file being edited in the vi editor. The text is as follows:

```
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able to create
```

At the bottom of the window, there is a status bar with the text "— 可视 —" (Visible) on the left, "19,1" in the middle, and "7%" on the right. The "19,1" and "7%" are highlighted with red boxes.

# vi的退出

- 要退出vi，在命令模式下键入如图所示命令。
- 其中:wq和:x是存盘退出，而:q是直接退出。可以用:w命令保存文件后再用:q退出，或用:wq或:x命令退出。
- 如果你不想保存改变后的文件，你就需要用:q!命令，这个命令将不保存文件而直接退出vi

:w	保存；
:w filename	另存为 filename
:wq!	保存退出
:wq! filename	注：以 filename 为文件名保存后退出
:q!	不保存退出
:x	应该是保存并退出，功能和:wq!相同



# vi的工作模式

- vi可以分为三种状态，分别是命令模式（**command mode**）、插入模式（**Insert mode**）和底行模式（**last line mode**）
- 命令模式：控制屏幕光标的移动，字符、字或行的删除，移动复制某区段，或进入插入模式（键入“i”等）、或底行模式（键入“:”）。
- 插入模式：只有在该模式下，才能进行文字的输入，按「ESC」键可回到命令行模式。
- 底行模式：将文件保存或退出vi，或设置编辑环境，如寻找字符串、列出行号.....等。
- 键入“:”进入底行模式，此时vi会在屏幕的最后一行显示一个“:”等待用户输入命令；用户按“/或?”键进入向下/向上搜索模式。



# vi编辑器中的常用命令

类型	命令	说明
命令 模式下	<b>x</b>	删除光标所在的字符
	<b>dd</b>	删除光标所在的行
	<b>u</b>	撤销上次操作（undo）
	<b>/str</b>	查找str； <b>n</b> ： 查找下一个； <b>N</b> ： 查找上一个
	<b>ctrl + f</b>	向前翻转一页
	<b>ctrl + b</b>	向后翻转一页

类型	命令	说明
进入插入模 式	<b>i</b>	从光标所在位置前开始插入文本
	<b>I</b>	该命令是将光标移到当前行的行首，然后在起前插入文本
	<b>a</b>	用于在光标当前所在位置之后追加新文本
	<b>A</b>	将光标移到所在行的行尾，从哪里开始插入新文本
	<b>o</b>	在光标所在行的下面新开一行，并将光标置于该行行首，等待输入
	<b>O</b>	在光标所在行的上面插入一行，并将光标置于该行行首，等待输入

# 文件相关命令

- 使用下表中的命令可以在Vi中进行文件相关的操作

文件相关	<code>:w</code>	在光标所在行的下面新开一行，并将光标置于该行行首，等待输入
	<code>:w file</code>	在光标所在行的上面插入一行，并将光标置于该行行首，等待输入
	<code>:n1,n2w file</code>	将从 n1 开始到 n2 结束的行写到 file 文件中
	<code>:nw file</code>	将第 n 行写到 file 文件中
	<code>:1,w file</code>	将从第 1 行起到光标当前位置的所有内容写到 file 文件中
	<code>:\$w file</code>	将从光标当前位置起到文件结尾的所有内容写到 file 文件中
文件相关	<code>:r file</code>	打开另一个文件 file
	<code>:e file</code>	新建 file 文件
	<code>:f file</code>	把当前文件改名为 file 文件

# GCC基本用法

---

- gcc基本的用法是：
- gcc [options] [filenames]
  - ◆ Options: 编译器所需要的编译选项
  - ◆ filenames: 要编译的文件名

# 编译选项（1）

---

gcc编译器的编译选项大约有100多个，其中多数我们根本就用不到，这里只介绍其中最基本、最常用的参数。

- **-o output\_filename**: 确定可执行文件的名称为 **output\_filename**。如果不给出这个选项，gcc就给出预设的可执行文件**a.out**。

# 编译选项（2）

---

- **-c**: 只编译，不连接成为可执行文件，编译器只是由输入的.c等源代码文件生成.o为后缀的目标文件
- **-g**: 产生调试工具（GNU的gdb）所需要的符号信息，要想对编译出的程序进行调试，就必须加入这个选项
- **-Og（-O1）**，生成符合原始C代码整体结构的机器代码对整个源代码在编译、连接过程中进行优化处理，生成的可执行文件执行效率得到提高。
- **-O2**，比-O更好的优化编译、连接，当然整个编译、连接过程会更慢

# 举例1

bar.c文件

```
#include <stdio.h>
#define MAXLINE_LENGTH 80
char Buffer[MAXLINE_LENGTH];
int main(void){
    int nextInChar;
    int nextLocation;
    printf("Input> ");
    nextLocation = 0;
    while ((nextInChar = getchar()) != '\n' && nextInChar != EOF) {
        Buffer[nextLocation++] = nextInChar;
    }
    Buffer[nextLocation++] = 0;
    printf("Input string is: ");
    puts(Buffer);
    printf("\n");
    return nextLocation-1;
}
```

根据GCC编译输出的提示信息，找出错误

# 举例2 (1)

---

foo.c文件

```
#include<stdio.h>
```

```
int main(void) {
```

```
    double counter, result, temp;
```

```
    for (counter=0; counter < 2000.0*2000.0*2000.0/20.0+2020; counter += (5-1)/4) {
```

```
        temp=counter/1979;
```

```
        result=counter;
```

```
    }
```

```
    printf(“Result is %lf\n”, result);
```

```
    return 0;
```

```
}
```



# 举例2 (2)

1. `gcc foo.c -o foo`  
`time ./foo`

```
frank@bupt3:~/code$ time ./foo
Result is 400002019.000000

real    0m1.458s
user    0m1.456s
sys     0m0.000s
```

2. `gcc -Og foo.c -o foo`  
`time ./foo`

```
frank@bupt3:~/code$ time ./foo
Result is 400002019.000000

real    0m0.585s
user    0m0.584s
sys     0m0.000s
```

对比两次执行的输出结果不难看出，程序的性能的确得到了大幅度的改善

# 编译选项（3）

---

- **-I dirname**: 将**dirname**所指出的目录加入到程序头文件目录列表中。

C程序中的头文件包含两种情况：

**#include<A.h>**

**#include“B.h”**

对于<>, 预处理程序在系统预设的头文件目录（如 **/usr/include**）中搜寻相应的文件；而对于“ ”, 预处理程序在当前目录中搜寻头文件。

这个选项的作用是告诉预处理程序，如果在当前目录中没有找到需要的文件，就到指定的**dirname**目录中去寻找。

例：**gcc foo.c -I /home/include -o foo**

# 编译选项（4）

---

- **-L dirname**: 将**dirname**所指出的目录加入到库文件的目录列表中。
- 在默认状态下，链接程序**ld**在系统的预设路径中（如**/usr/lib**）寻找所需要的库文件。这个选项告诉链接程序，首先到**-L**指定的目录中去寻找，然后再到系统预设路径中寻找。

# GDB的概述

---

- **GDB**是一款GNU开发并发布的UNIX/Linux下的**程序调试工具**。它使程序员能在程序运行时观察程序的内部结构和内存的使用情况。**GDB**提供的功能包括：
  - ◆ 监视程序中变量的值
  - ◆ 能设置断点以使程序在指定的代码行上停止执行
  - ◆ 能一行行的执行代码

# GDB的使用方法

---

- GDB的命令格式

- ◆ `gdb [option] [executable-file[core-file or process-id]]`

# GDB的常用命令（1）

---

## ■ 加载和退出命令

- ◆ **`gdb filename`**: 在shell下直接加载文件进行调试
- ◆ **`file filename`**: 在gdb下通过file命令加载程序进行调试
- ◆ **`kill`**: 终止正在调试的程序
- ◆ **`quit`**: 退出gdb调试环境

## ■ 断点控制

- ◆ **`break`** 函数名或行号 [**if 条件**]
- ◆ **`info break`** : 显示程序中设置的断点;
- ◆ **`delete breakpoint`** 断点号: 删除指定的断点
- ◆ **`clear`** 断点号: 作用同上
- ◆ **`disable breakpoint`** 断点号: 禁用指定的断点
- ◆ **`enable breakpoint`** 断点号: 允许指定的断点

# GDB的常用命令（2）

---

## ■ 程序的控制指令

- ◆ **run**: 程序开始执行，一直运行到断点才终止
- ◆ **continue**: 运行到下一个断点
- ◆ **step**: 执行一条C语句
- ◆ **stepi**: 执行一条**指令**
- ◆ **nexti**: 类似于stepi，但以函数调用为单位执行

# GDB的常用命令（3）

---

## ■ 变量、参数的设置与查看

- ◆ **list**: 列出产生执行文件的源代码的一部分
- ◆ **watch** 变量名: 当变量改变时, 显示变量修改前后的值;
- ◆ **print** 变量名: 打印变量值
- ◆ **x**: **examine**查看命令, 查看内存地址中的内容
- ◆ **whatis** 变量名或函数名: 显示变量或函数的类型
- ◆ **ptype**:显示数据结构的定义
- ◆ **set args**:设置程序的运行参数
- ◆ **show args**:显示程序的运行参数



# 调试操作举例 (1)

```
#define index 20      demo.c
int a[50];
int foo(int *p){
    int i;
    for (i = 0; i < 50; i++)
        p[i] = 1 << i;
    return p[index];
}

int main(){
    int result=100;
    result = foo(a)+result;
    return result;
}
```

```
0000000000000061a <main>:
61a:488d3d1f0a2000 lea 0x200a1f(%rip),%rdi
621:e8d4ffffffffff callq 5fa <foo>
626:83c064          add     $0x64,%eax
629:c3            retq
```

```
000000000000005fa <foo>:
5fa:b90000000000 mov     $0x0,%ecx
5ff:eb10          jmp     611<foo+0x17>
601:4863d1        movslq  %ecx,%rdx
604:b80100000000 mov     $0x1,%eax
609:d3e0          shl     %cl,%eax
60b:890497        mov     %eax, (%rdi,%rdx,4)
60e:83c101        add     $0x1,%ecx
611:83f931        cmp     $0x31,%ecx
614:7eeb          jle     601<foo+0x7>
616:8b4750        mov     0x50(%rdi),%eax
619:c3            retq
```

# 调试操作举例（2）

---

1. 使用gcc命令进行编译（-g选项生成调试信息）

**gcc -Og -g -fno-stack-protector demo.c -o demo**

2. 启动gdb进行调试

**gdb demo**

3. 使用break命令设置断点

**b (or break) <line\_number> or <function\_name>**

条件断点的设置

**b ) <line\_number> or <function\_name> if condition**

例如：b main 在main函数处设置断点

4. 使用run命令运行程序

**r (or run)**

5. 使用list查看源代码

**l (or list)**

# 调试操作举例（2）

---

6. 使用**print**命令显示指定寄存器内容  
**print /x \$rax** 以十六进制显示**rax**的内容
7. 使用**continue**命令从断点处开始执行直到下一个断点
8. 使用**stepi**命令执行一条指令
9. 使用**nexti**命令执行一条指令或一个函数调用
10. **quit**命令退出gdb

命令	效果
开始和停止 quit run kill	退出 GDB 运行程序(在此给出命令行参数) 停止程序
断点 break multstore break * 0x400540 delete 1 delete	在函数 multstore 入口处设置断点 在地址 0x400540 处设置断点 删除断点 1 删除所有断点
执行 stepi stepi 4 nexti continue finish	执行 1 条指令 执行 4 条指令 类似于 stepi, 但以函数调用为单位 继续执行 运行到当前函数返回
检查代码 disas disas multstore disas 0x400544 disas 0x400540,0x40054d print /x \$rip	反汇编当前函数 反汇编函数 multstore 反汇编位于地址 0x400544 附近的函数 反汇编指定地址范围内的代码 以十六进制输出程序计数器的值
检查数据 print \$rax print /x \$rax print /t \$rax print 0x100 print /x 555 print /x (\$rsp+ 8) print *(long *) 0x7fffffff818 print *(long *) (\$rsp+ 8) x/2g 0x7fffffff818 x/20bmultstore	以十进制输出 \$rax 的内容 以十六进制输出 \$rax 的内容 以二进制输出 \$rax 的内容 输出 0x100 的十进制表示 输出 555 的十六进制表示 以十六进制输出 \$rsp 的内容加上 8 输出位于地址 0x7fffffff818 的长整数 输出位于地址 \$rsp+8 处的长整数 检查从地址 0x7fffffff818 开始的双(8 字节)字 检查函数 multstore 的前 20 个字节
有用的信息 info frame info registers help	有关当前栈帧的信息 所有寄存器的值 获取有关 GDB 的信息

图 3-39 GDB 命令示例。说明了一些 GDB 支持机器级程序调试的方式