



# 实验一

## LINUX环境和GCC工具链

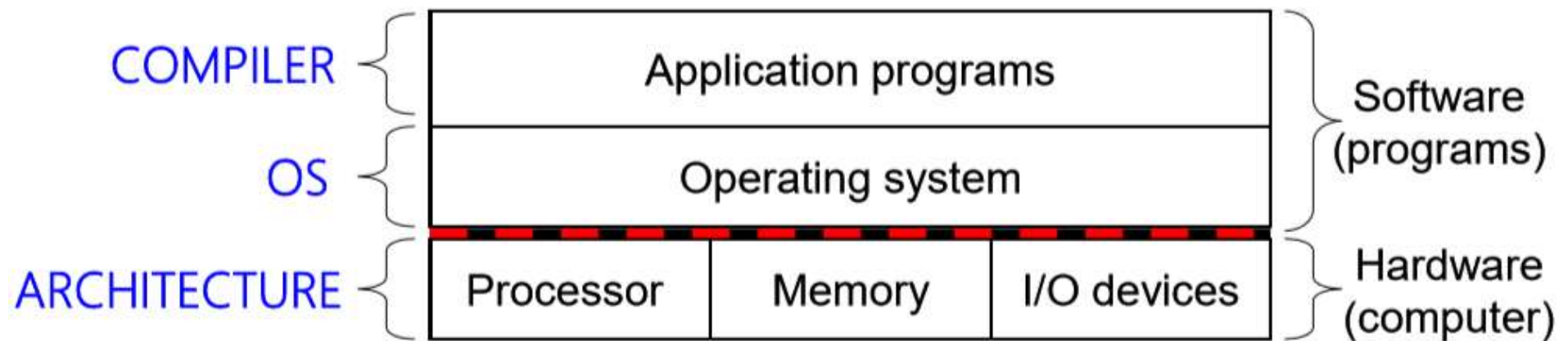
# Outline

---

- **Linux**操作系统概述和实验环境介绍
- **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

---

- 目前最流行的个人桌面操作系统。



# 操作系统—Linux

---

- **服务器端**，**Linux**市场份额一直在快速增长。**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、高并发的问题；
- 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）

---

- 尽管**UNIX**被免费提供，但获取源代码需要向**AT&T**交纳一定的许可证费用。**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**最初发布于**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。

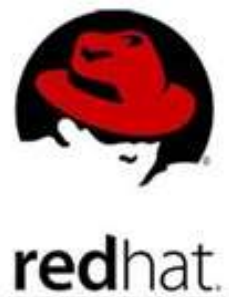


# Linux发行版举例（1）

## Ubuntu



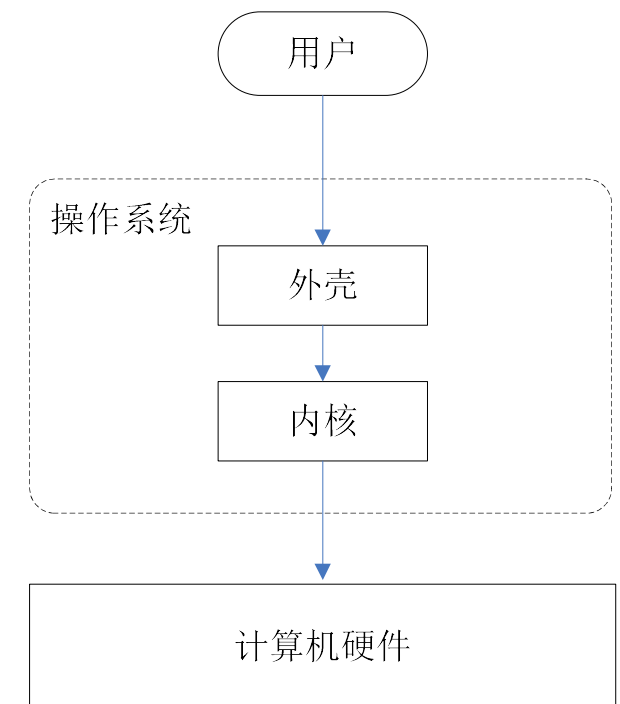
# Linux发行版举例（2）



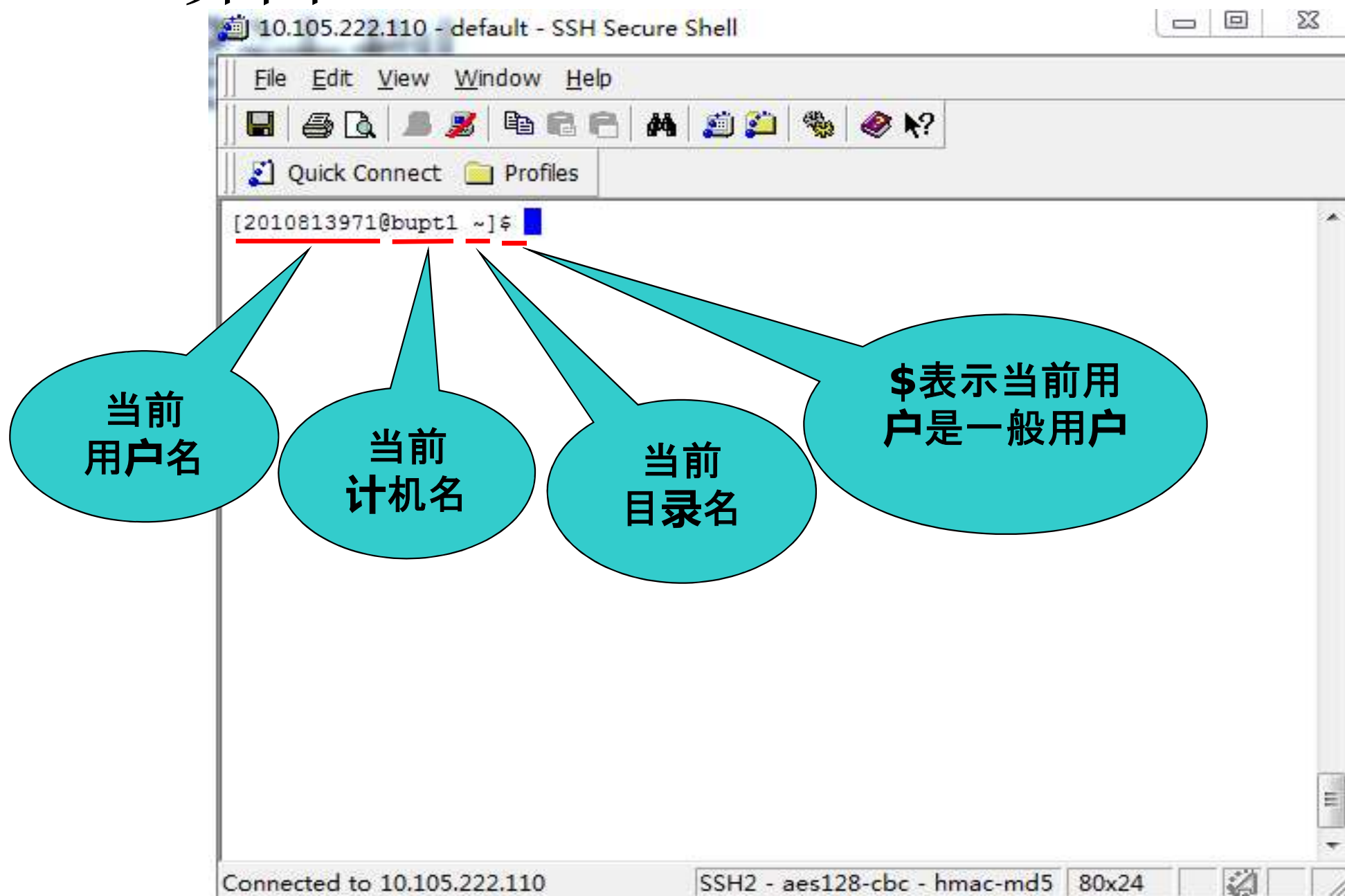
# Shell

## 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，由一个称为**Explorer**的模块解释用户的输入
- 又如**DOS**的命令行界面，**Command.com**就是外壳程序，类似于**Linux**的**shell**



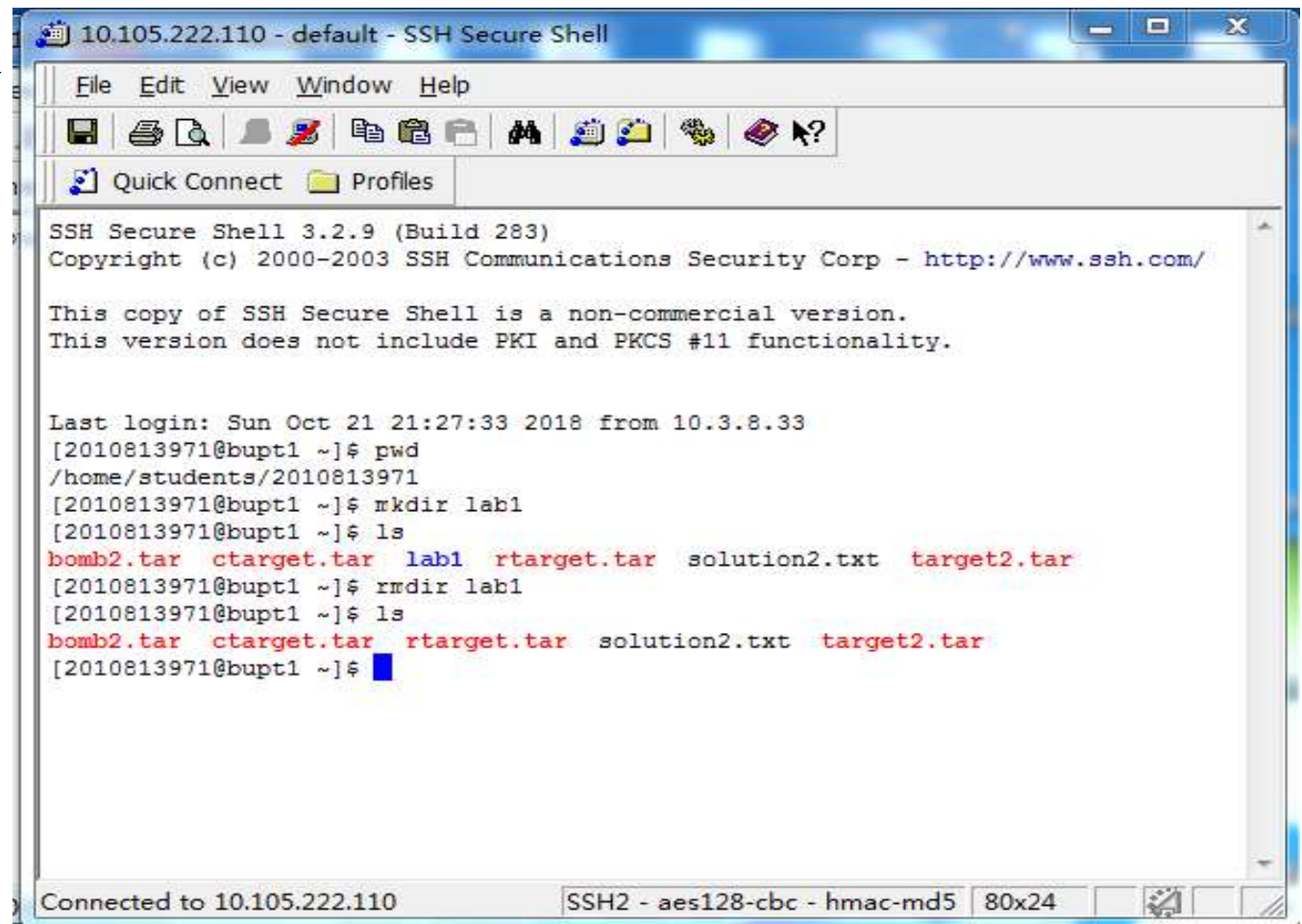
# Shell界面





# 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**



```
10.105.222.110 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

SSH Secure Shell 3.2.9 (Build 283)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.

Last login: Sun Oct 21 21:27:33 2018 from 10.3.8.33
[2010813971@bupt1 ~]$ pwd
/home/students/2010813971
[2010813971@bupt1 ~]$ mkdir lab1
[2010813971@bupt1 ~]$ ls
bomb2.tar ctaraget.tar lab1 rtaraget.tar solution2.txt target2.tar
[2010813971@bupt1 ~]$ rmdir lab1
[2010813971@bupt1 ~]$ ls
bomb2.tar ctaraget.tar rtaraget.tar solution2.txt target2.tar
[2010813971@bupt1 ~]$
```

Connected to 10.105.222.110 SSH2 - aes128-cbc - hmac-md5 80x24

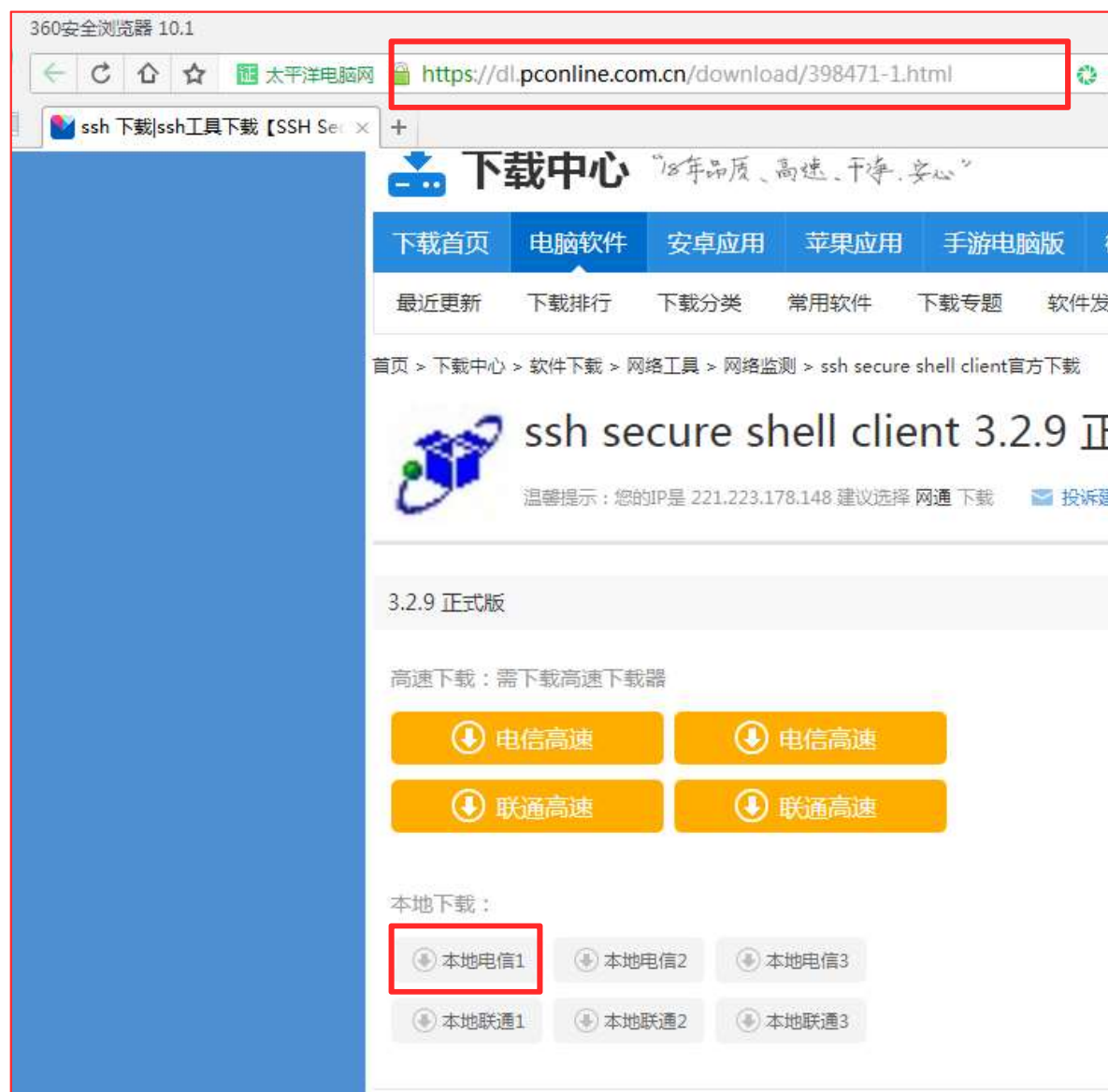
# Shell程序

---

- 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。



# SSH软件下载



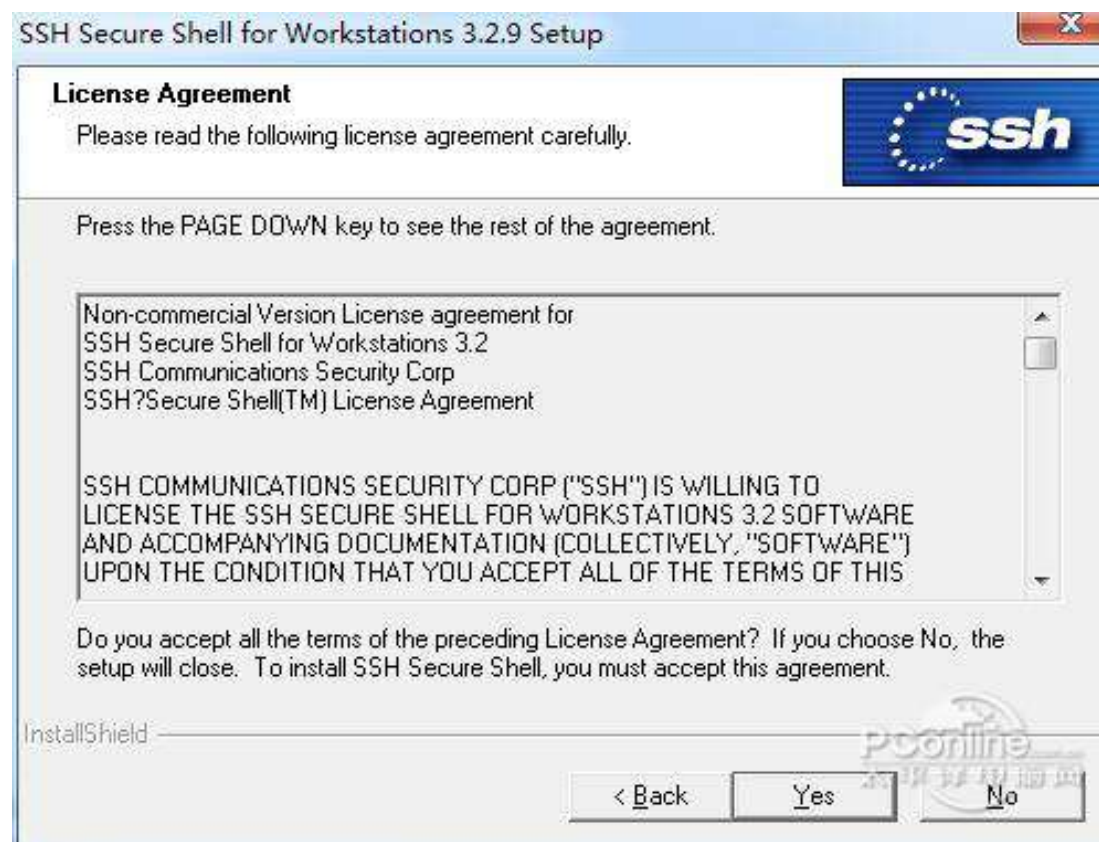
# SSH安装（1）

- 1、点击ssh（secure shell）工具的安装程序



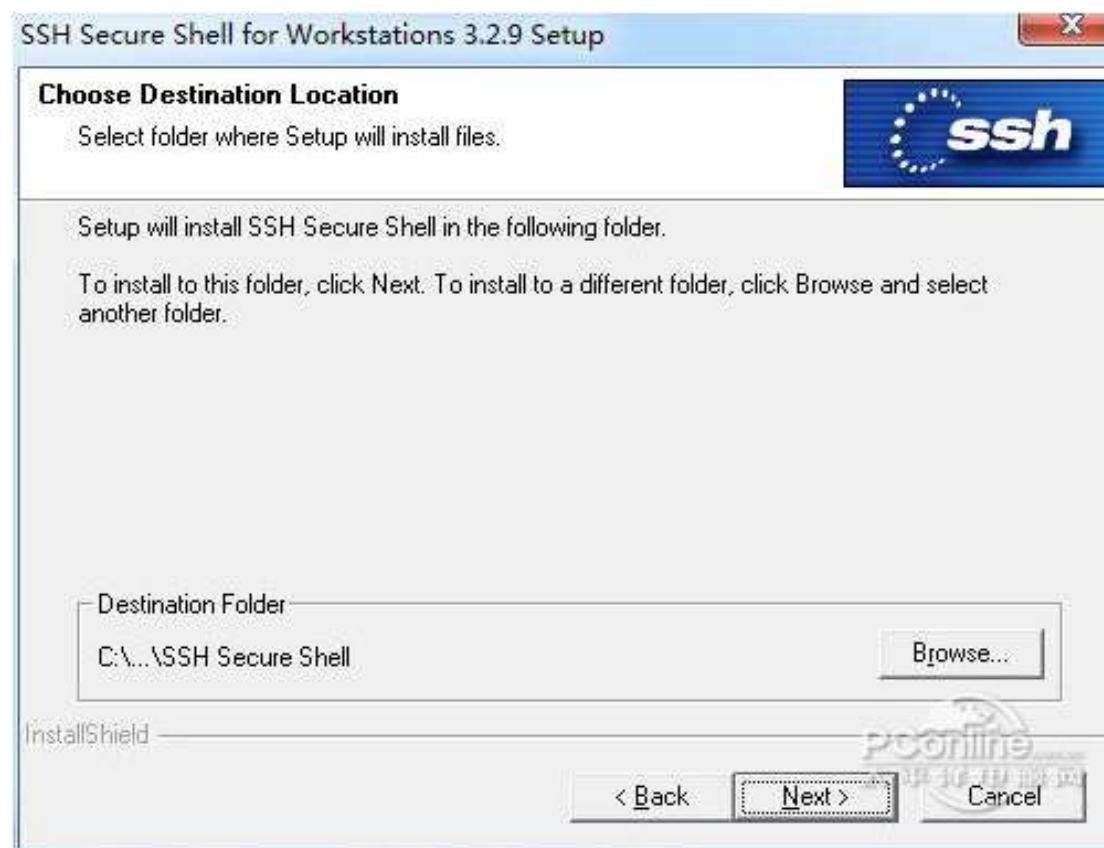
# SSH安装 (2)

## ■ 2、点击我接受



# SSH安装 (3)

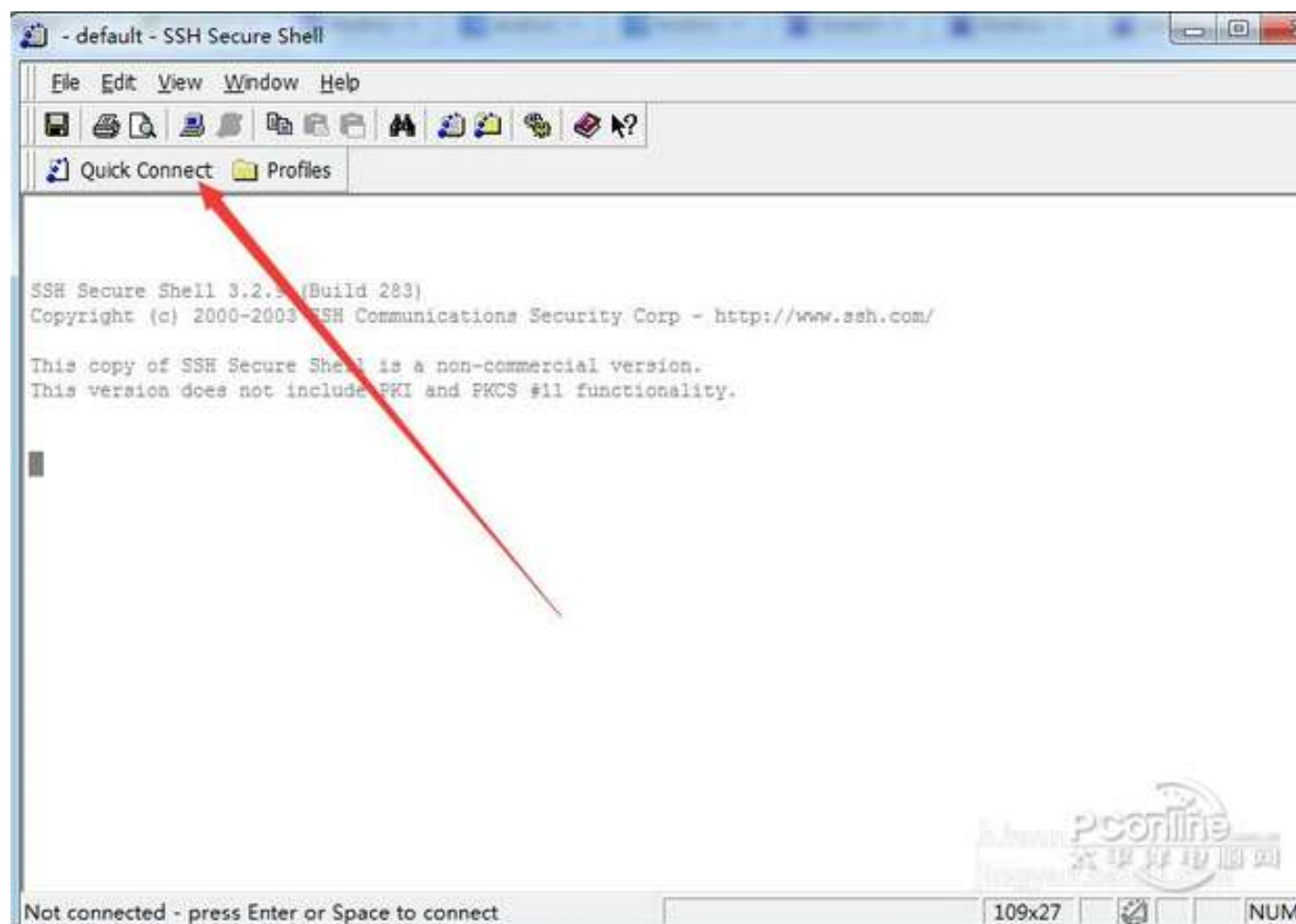
- 3、点击自定义安装，选择好保存路径



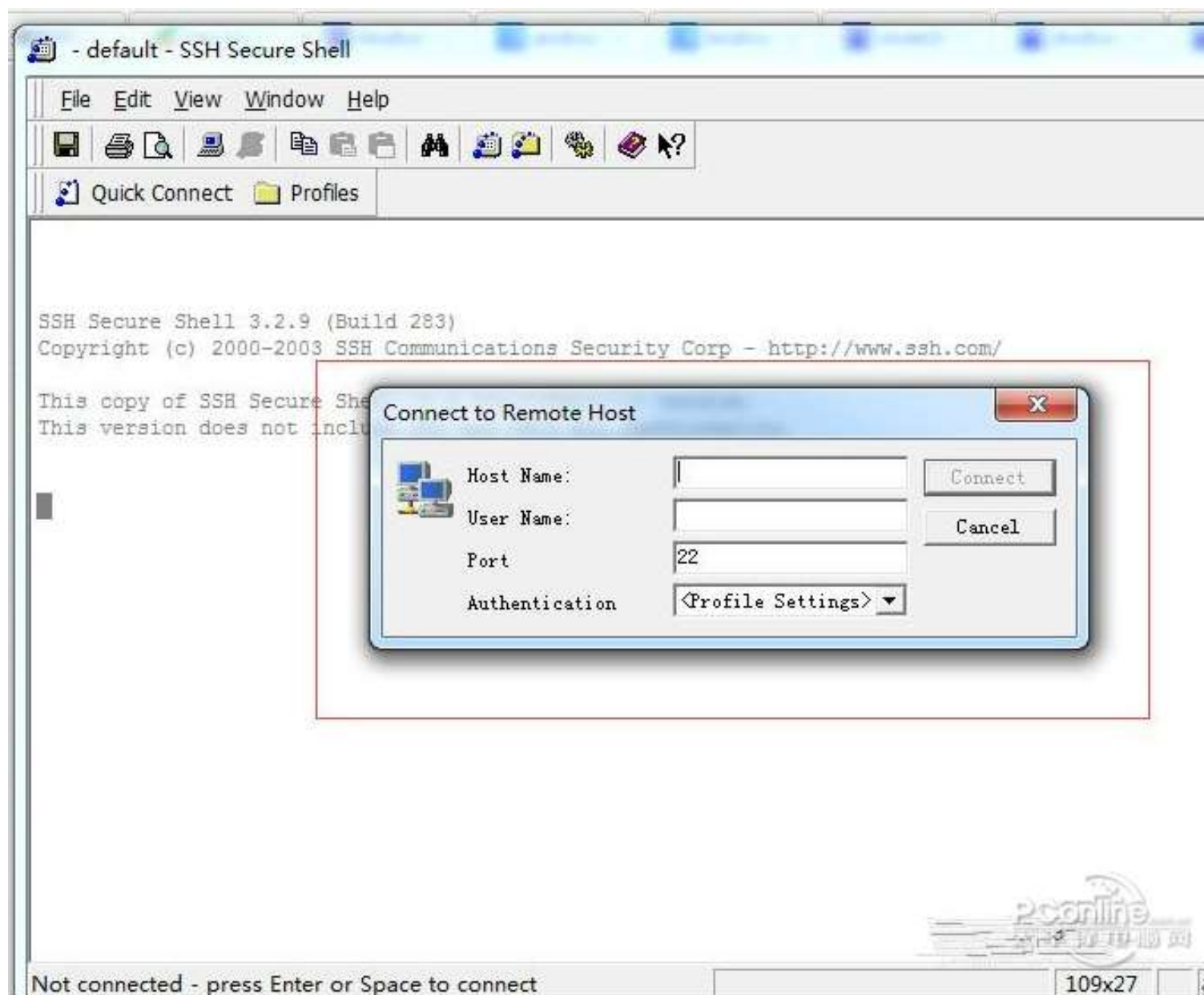


# 一、如何把Linux终端搬到了Windows下？

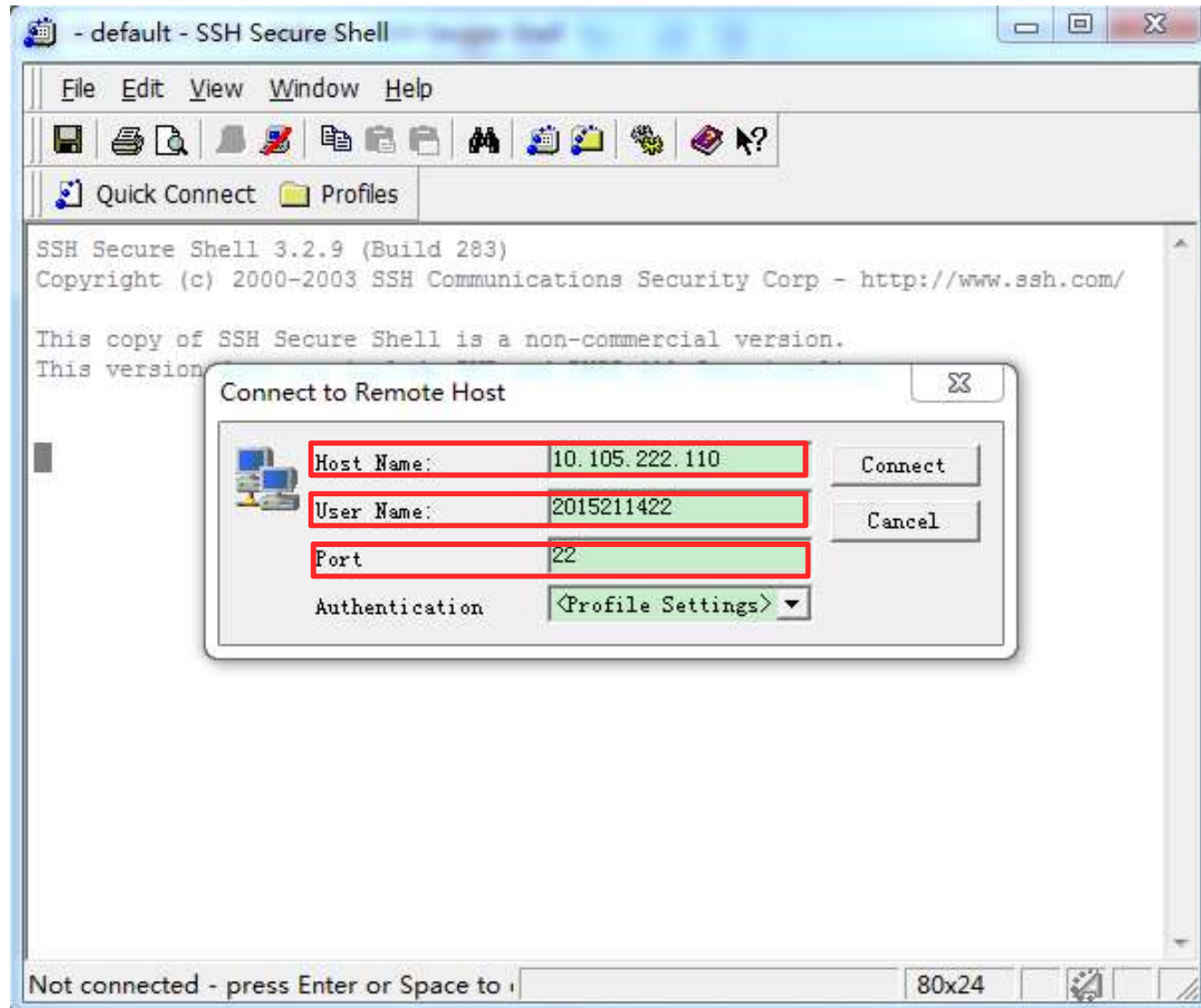
1、在窗口上左键点击快速登录，弹出登录远端主机的登录窗口



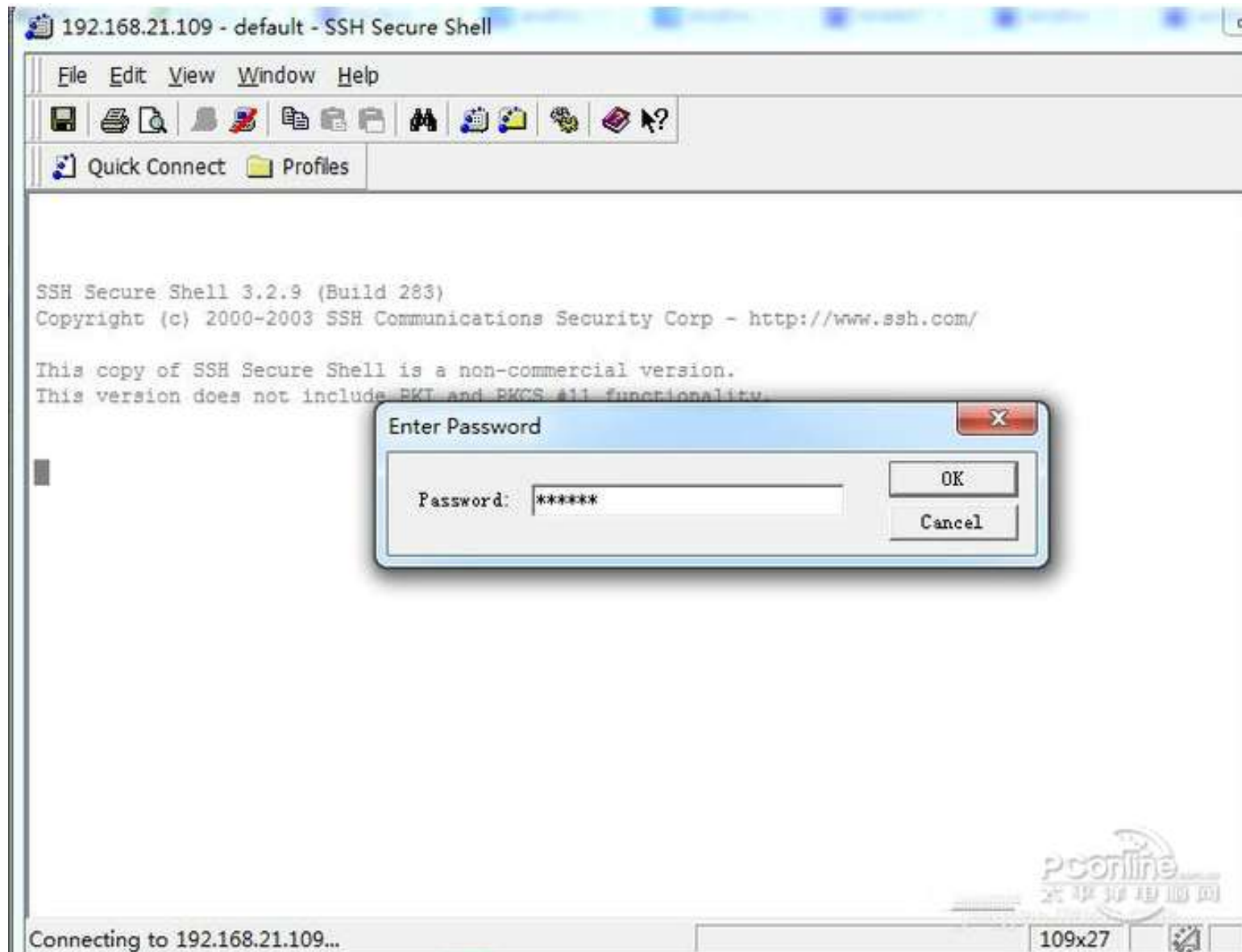
## 2、输入远端的IP地址和主机用户名



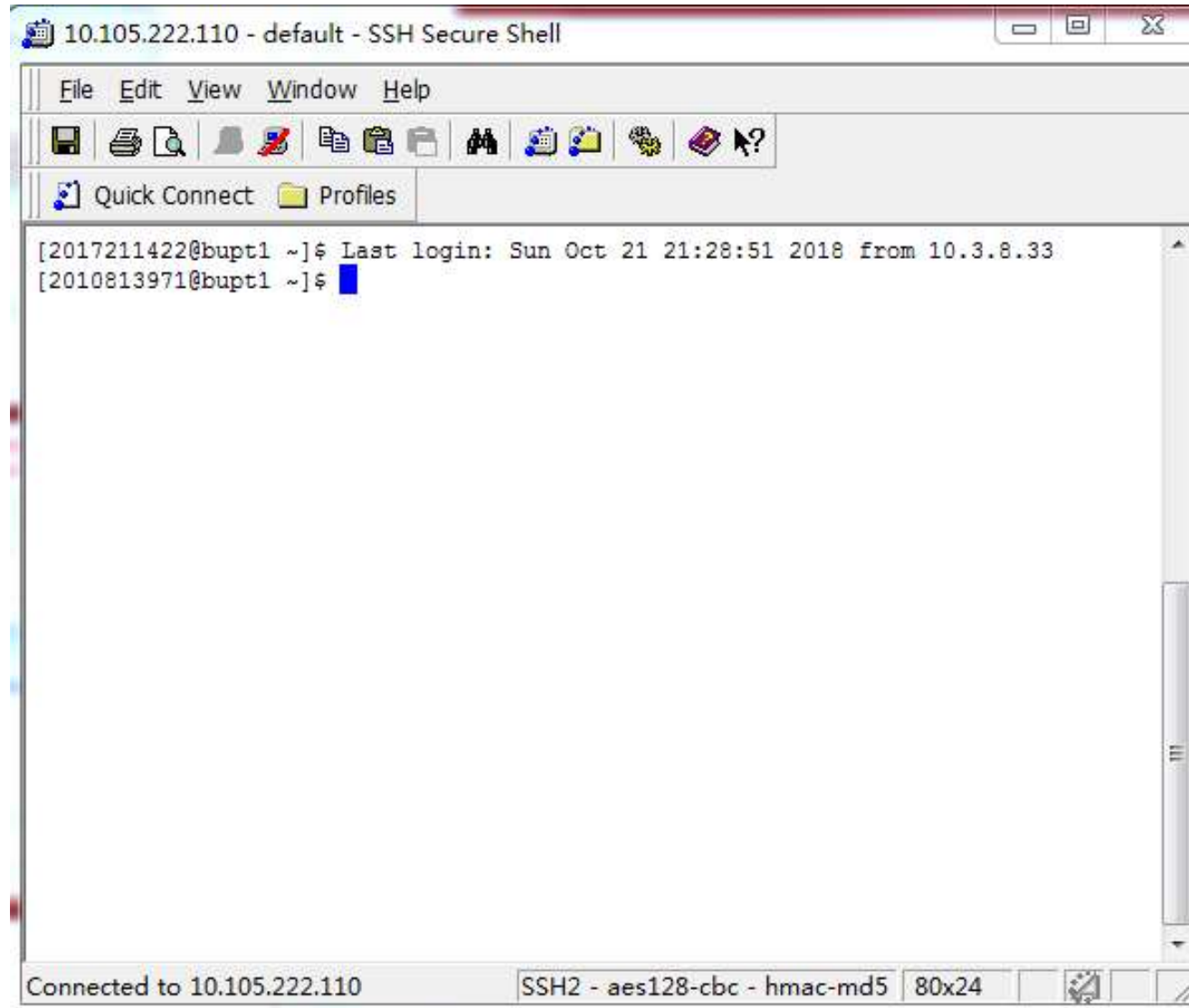
3、输入IP地址和用户名后，点击链接按键，客户端与主机链接成功后就会弹出输入密码的对话框（用户名为学号）



4、输入密码，点击确认按钮，如果用户名和密码输入正确的话，就会进入主界面（密码为**bupt+学号**，如**bupt2015211422**）



# 成功登陆



The screenshot shows a terminal window titled "10.105.222.110 - default - SSH Secure Shell". The window has a menu bar with "File", "Edit", "View", "Window", and "Help". Below the menu bar is a toolbar with various icons. The main text area displays the following text:

```
[2017211422@bupt1 ~]$ Last login: Sun Oct 21 21:28:51 2018 from 10.3.8.33  
[2010813971@bupt1 ~]$
```

The status bar at the bottom of the window shows "Connected to 10.105.222.110", "SSH2 - aes128-cbc - hmac-md5", and "80x24".

# SSH配置数据

主机名(H):	<input type="text" value="10.120.11.12"/>
端口(O):	<input type="text" value="22"/>
防火墙(F):	<input type="text" value="None"/>
用户名(U):	<input type="text" value="2018123456"/>
...	

# MAC OS登录Linux

```
#!/usr/bin/expect -f
```

```
set port 22
```

```
set user 2018123456
```

```
set host 10.120.11.12
```

```
set password xxxxxx
```

```
set timeout 30
```

```
spawn ssh -p$port $user@$host
```

```
expect {
```

```
"*assword:*" { send "$password\r" }
```

```
"yes/no" { send "yes\r";exp_continue }
```

```
}
```

```
interact
```

打开终端窗口，用vi命令创建名为bupt1文件  
拷贝右边内容并做相应修改

保存文件名

修改文件属性，命令为chmod 744 bupt1

执行命令 ./bupt1

登录Linux

执行命令 logout

返回MAC OS终端



# Outline

---

- **Linux**操作系统概述和实验环境介绍
- **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

# Linux文件系统

- 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(装置文件), 使用者可以经由核心用来存取系统中的硬设备, 当使用装置文件时内核会辨识出输入输出请求, 并传递到相对应装置的驱动程序以便完成特定的动作;

## 3. /boot

存放与系统激活的相关文件, 不可任意删除

## 4. /home

登录用户的主目录(\$HOME)放在此目录下, 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

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

## 5. /lib

存放许多系统激活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

- 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

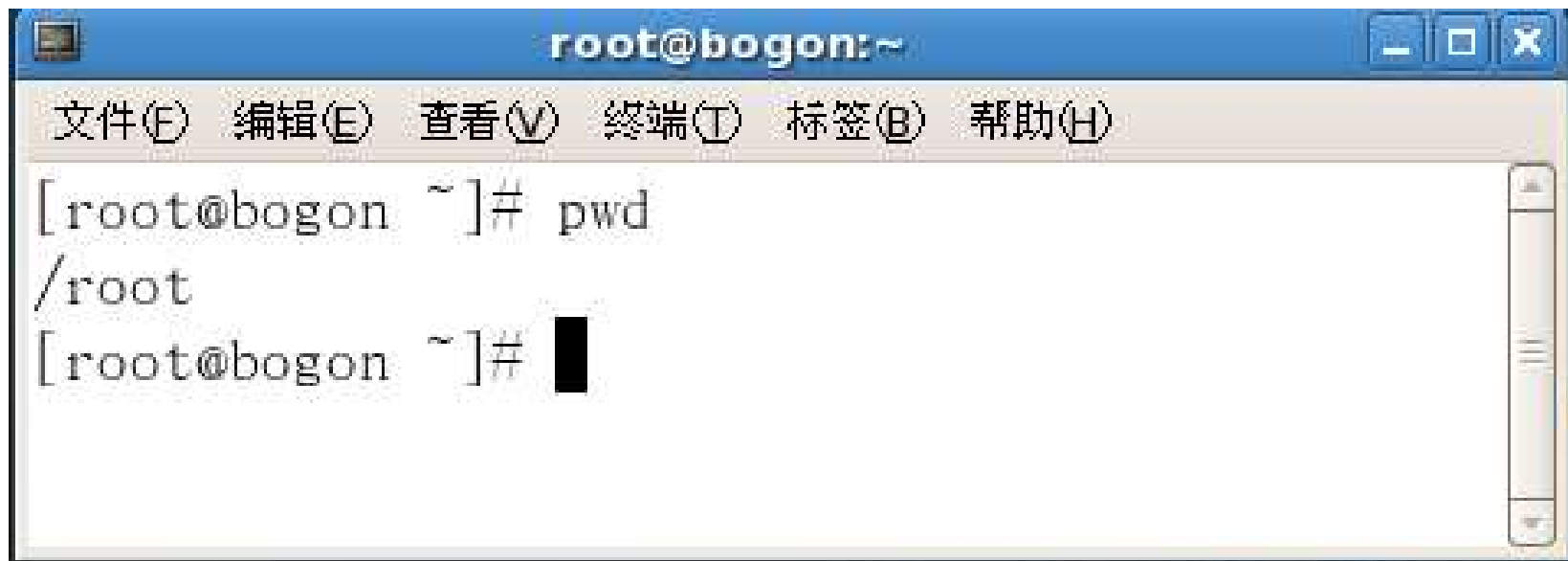
本目录存放被系统修改过的数据。在这个目录下的重要目录有 /var/log、/var/spool、/var/run等, 它们分别用于存放记录文件、新闻邮件、运行时信息。

# (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。

# pwd命令

- 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】



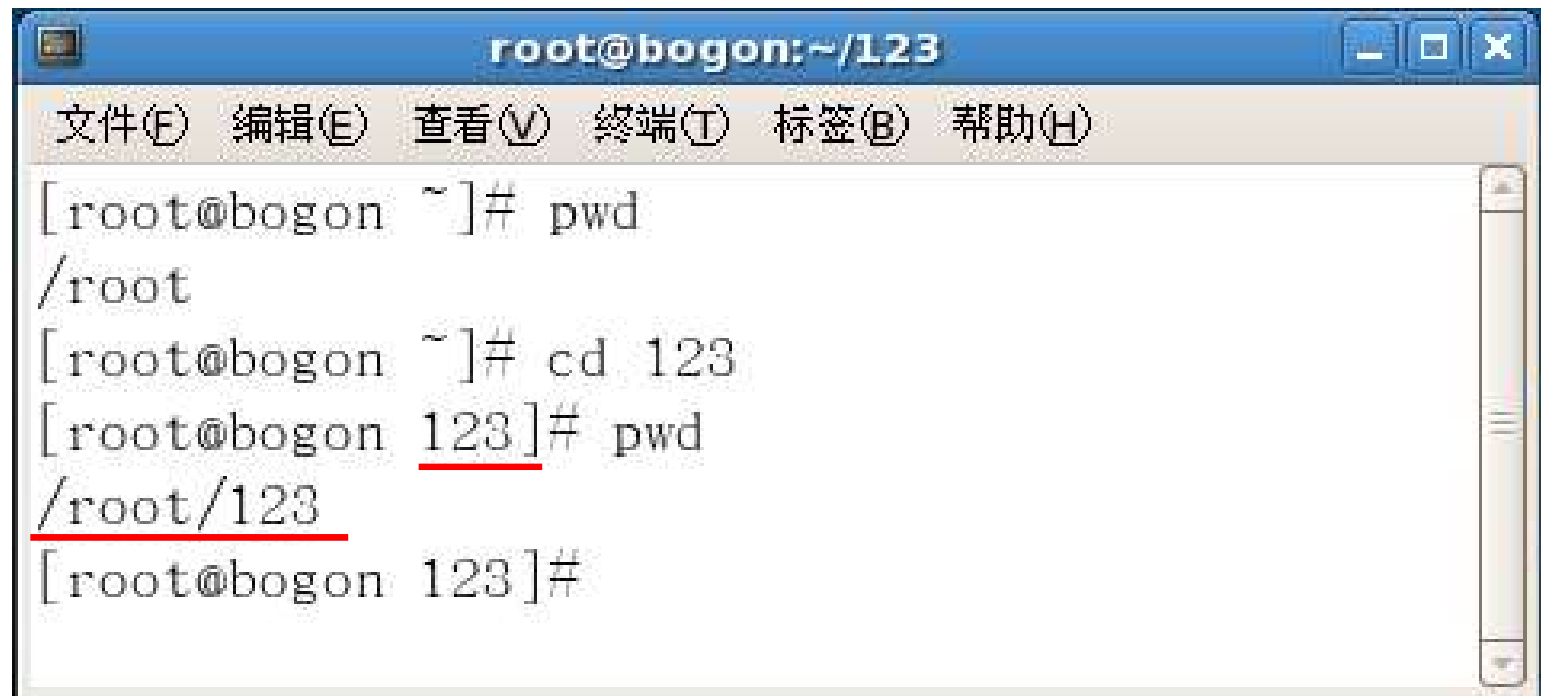
A terminal window titled "root@bogon:~" with a menu bar containing "文件(F)", "编辑(E)", "查看(V)", "终端(T)", "标签(B)", and "帮助(H)". The terminal shows the command "pwd" being executed, resulting in the output "/root". The prompt "[root@bogon ~]#" is visible before and after the command.

```
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（续）

- 【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 a menu bar containing '文件(F)', '编辑(E)', '查看(V)', '终端(T)', '标签(B)', and '帮助(H)'. The terminal shows the following commands and output:

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

# 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 a menu bar containing '文件(F)', '编辑(E)', '查看(V)', '终端(T)', '标签(B)', and '帮助(H)'. The terminal shows a sequence of commands and their outputs: '[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', and '[root@bogon 456]#'.

```
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

- 【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（续）

## ■ 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。



```

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。（续）

- ls可以同时选择多个选项：

```

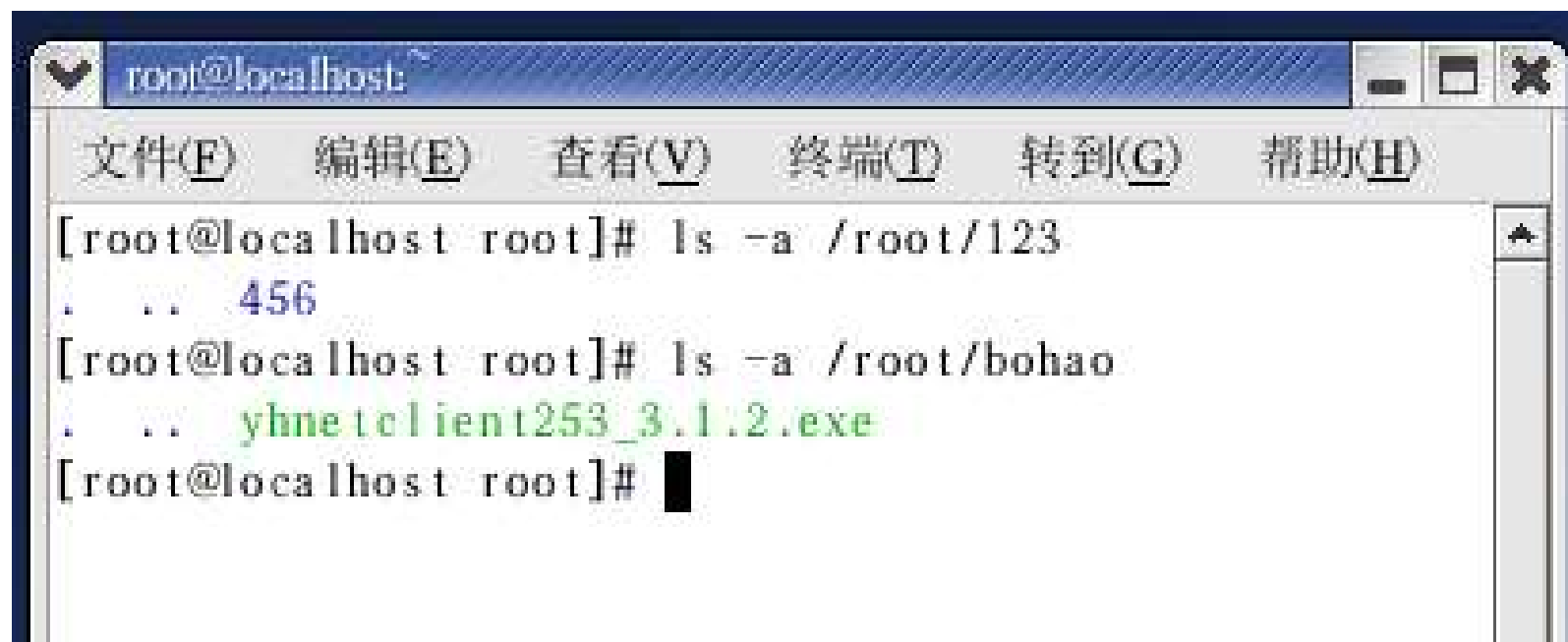
root@localhost
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
[root@localhost root]# ls -al
总用量 220
drwxr-x--- 16 root    root    4096  3月 26 14:15 .
drwxr-xr-x 19 root    root    4096  3月 26 13:31 ..
drwxr-xr-x  3 root    root    4096  3月 26 14:34 123
-rw-r--r--  1 root    root   1265  3月  6 07:52 anaconda-ks.cfg
-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 .fontconfig.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。（续）

- 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 the following commands and output:

```
[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**

## (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>b</b> 键上一页。
<b>less</b>	分页查看，类似 <b>more</b> ，也可方向键滚动显示。
<b>head</b>	查看文件的前面部分，默认是前十行。
<b>tail</b>	查看文件的后面部分，默认是后十行。
<b>cp</b>	复制文件。
<b>mv</b>	移动文件。
<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，只显示前十行/后十行。
- **【格式】** **head/tail** [选项] 文件名
- 选项：
  - -n 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



按**q**键退出

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>

# 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...`

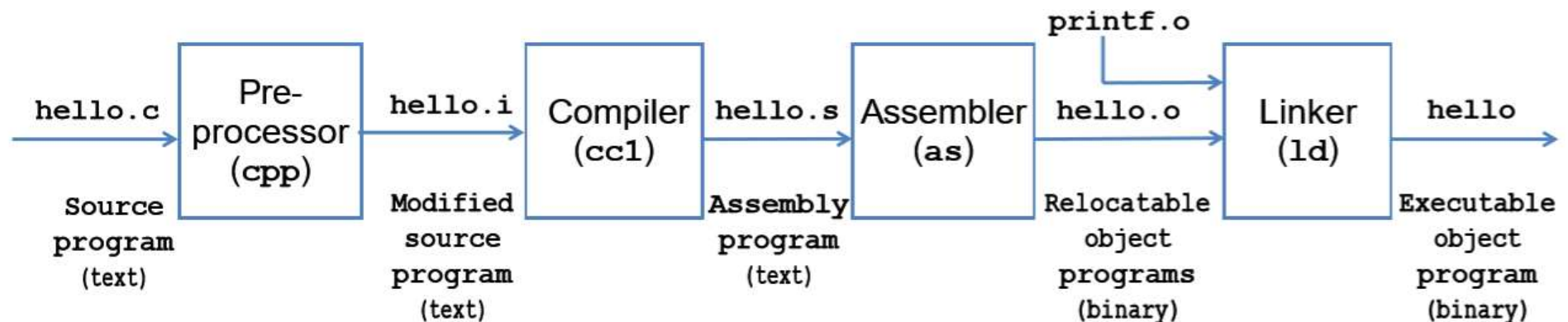
# Outline

---

- **Linux**操作系统概述和实验环境介绍
- **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
  - ✓ **C**, **.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**

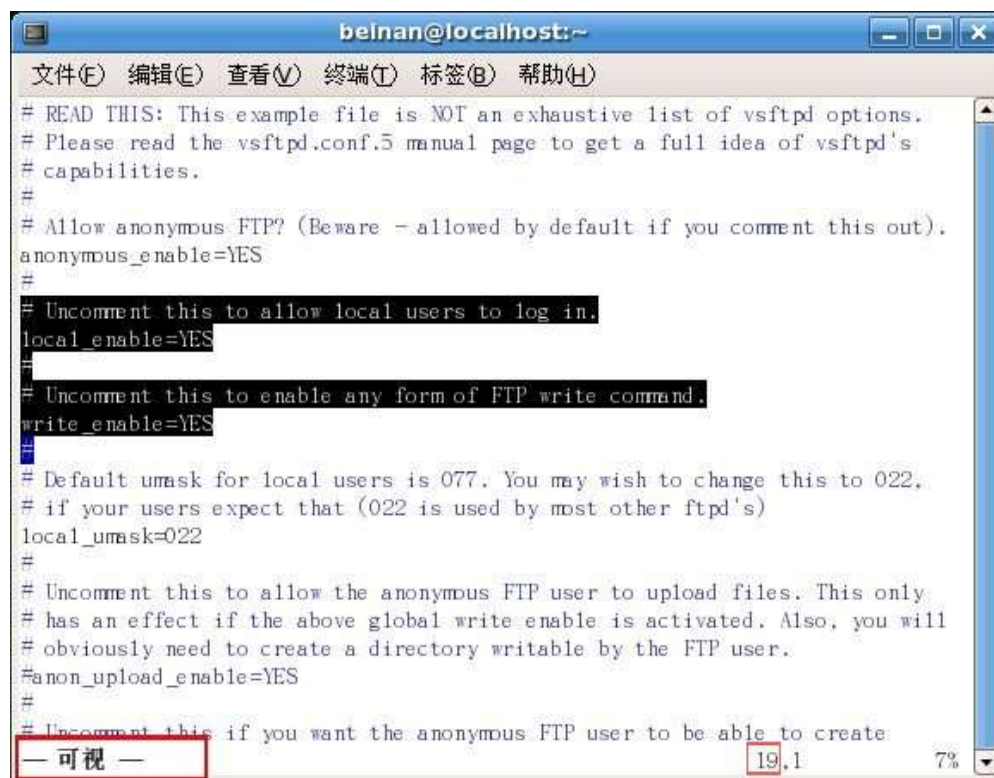
- 输出: **Hello world!**

# 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

# 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**



```
beinan@localhost:~  
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  
— 可视 — 19,1 7%
```

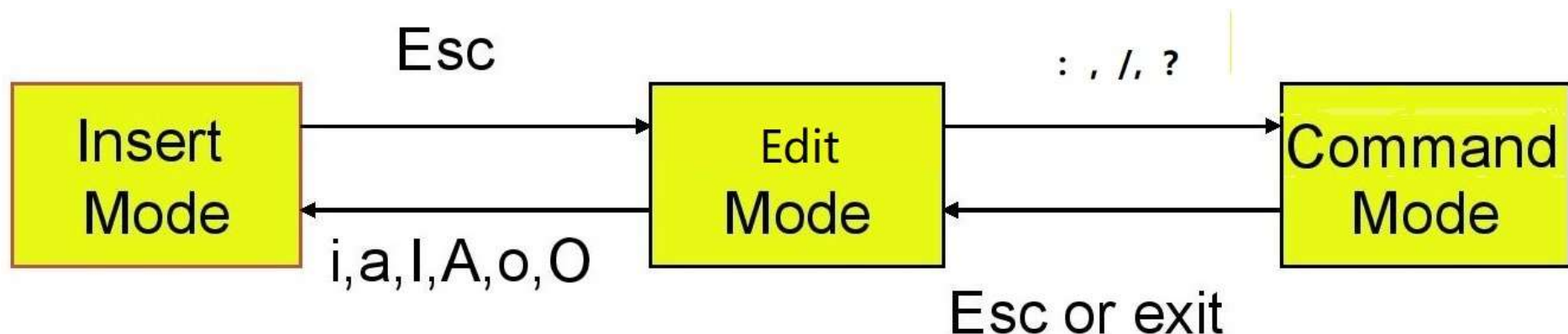
# Vi编辑器的退出

- 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**

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

# Vi的工作模式

- **Vi**有**3**种基本工作模式：编辑模式、插入模式和命令模式。
- 打开**Vi**后，首先进入编辑模式，然后**Vi**等待编辑命令输入而不是文本输入，这时输入的字母都将作为编辑命令来解释。
- 在编辑模式下输入插入命令**i**、附加命令**a**、打开命令**o**、修改命令**c**、替换命令**s**等都可以进入插入模式。在插入模式下，用户输入的任何字符都被**Vi**当作文件内容保存起来，并将其显示在屏幕上。在文本输入过程中（插入模式下），若想回到命令模式下，按**ESC**键即可。
- 在编辑模式下，用户按“**:**”键进入命令模式，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

- 在编辑模式下，输入如表所示的命令均可进入插入模式

类型	命令	说明
进入插入模式	i	从光标所在位置前开始插入文本
	I	该命令是将光标移到当前行的行首，然后在起前插入文本
	a	用于在光标当前所在位置之后追加新文本
	A	将光标移到所在行的行尾，从哪里开始插入新文本
	o	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
	O	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>	当前文件内容写盘
	<code>:w file</code>	当前文件内容写到file文件中
	<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

- **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**。(演示)

# 编译选项

- **-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**)所必要的符号信息，要想对编译出的程序进行调试，就必须加入这个选项
- **-Og (-O1)**，对程序进行优化编译、链接，采用这个选项，整个源代码会在编译、连接过程中进行优化处理，这样生成的可执行文件的执行效率可以提高。但编译、链接的速度要慢一些
- **-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

# 编译选项 (**foo.c**)

```
#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;
```

```
}
```

# 编译选项

**1. gcc foo.c -o foo**

**time ./foo**

**2. gcc -Og foo.c -o foo**

**time ./foo**

- 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很大幅度的改善

# 编译选项

- **-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**cpp**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**” ”**，**cpp**在当前目录中搜寻头文件。这个选项的作用是告诉**cpp**，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

- **-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，连接程序**ld**在系统的预设路径中（如**/usr/lib**）寻找所需要的库文件，这个选项告诉连接程序，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**开发组织并发布的**UNIX/Linux**下的程序调试工具。它使你能在程序运行时观察程序的内部结构和内存的使用情况. 以下是**gdb**所提供的一些功能:
  - 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
- `next`: 执行一条C语句（或函数）
- `step`: 执行一条C语句

# 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
- **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

# 调试例子(文件名aa.c)

```
#include <stdio.h>
#define MAXLINE_LENGTH 80
char Buffer[MAXLINE_LENGTH];
int * main(void){
    int a=0;
    int result=0;
    for (a=0;a<100;a++)
        result=a+result;
    int nextInChar;
    int nextLocation;
    printf("Input> ");
    nextLocation = 0;
    while ((nextInChar = getchar()) != '\n' &&
        nextInChar != EOF) {
        Buffer[nextLocation++] = nextInChar;
    }
    printf("Input complete ");
}
```

# 调试操作举例

1. 使用**gcc**命令进行编译（**务必需要-g选项**）

**gcc -g aa.c -o aa**

2. 启动**gdb**进行调试

**gdb aa**

3. 使用**run**命令运行程序

**r (or run)**

4. 根据逻辑找出问题，可附加使用**list**查看代码

**l (or list)**

# 调试操作举例

## 5. 使用**break**命令设置断点

**b (or break) lineNumber or functionName**

## 6. 使用**run**命令运行程序

**r (or run)**

## 7. 使用**watch**命令指定需要跟踪的变量

**watch Buffer**

## 7. 按照终端提示输入**input**字符串

命令	效果
<b>开始和停止</b> quit run kill	退出 GDB 运行程序(在此给出命令行参数) 停止程序
<b>断点</b> 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
<b>执行</b> 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
<b>检查代码</b> 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
<b>检查数据</b> 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
<b>有用的信息</b> 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