

# GNU/Linux 入门简介

柏 爱 俊

baj@mail.ustc.edu.cn  
中国科学技术大学计算机系

December 9, 2007

## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

## GNU (GNU is Not Unix) 计划

由 Richard Stallman 在 1983 年 9 月 27 日公开发起的，目标是创建一套完全自由的操作系统，GNU 项目贡献了很多一流的软件，包括 GCC、Emacs 等，但一直没有开发出操作系统内核。

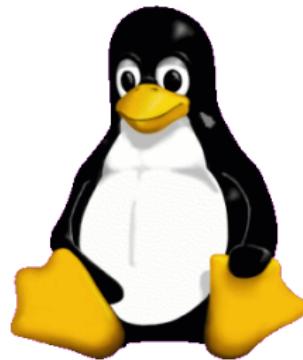
1991 年 8 月一位来自芬兰赫尔辛基大学的大二学

生 Linus Torvalds，对外发布了一套全新的操作系统内核，标志着 Linux 的诞生，Linux 发布不久就被 GNU 世界所接受，从而诞生了基于 Linux 的 GNU 系统（GNU/Linux）。

Linux 源码可以自由获得 (<http://www.kernel.org/>)，任何人都可以使用、更改或增强，经过十几年的发展，Linux 已成为目前使用最广泛的类 Unix 系统。

# Stallman & Linus





Linux 具有如下特点和优点：

- 真正意义上的多任务、多用户作系统
- 与 UNIX 系统在源代码级兼容，符合 IEEE POSIX 标准
- 支持数十种文件系统格式
- 开放源代码，用户可以自己对系统进行改进
- 提供了先进的网络支持：内置 TCP/IP 协议
- 优良的安全特性，对计算机病毒的天然免疫
- 还有很多，要在使用中体会……

Linux 具有如下特点和优点：

- 真正意义上的多任务、多用户作系统
- 与 UNIX 系统在源代码级兼容，符合 IEEE POSIX 标准
- 支持数十种文件系统格式
- 开放源代码，用户可以自己对系统进行改进
- 提供了先进的网络支持：内置 TCP/IP 协议
- 优良的安全特性，对计算机病毒的天然免疫
- 还有很多，要在使用中体会……

Linux 系统由内核和其他应用程序组成，不同的厂商可以发布内核与其他应用程序包的搭配，这样的组合称为“Linux 发行版”，不同发行版没有本质差别，差别仅在于内核的版本、软件包的数量和种类以及提供的服务。目前，世界上已经有了超过百种的不同发行版。



## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

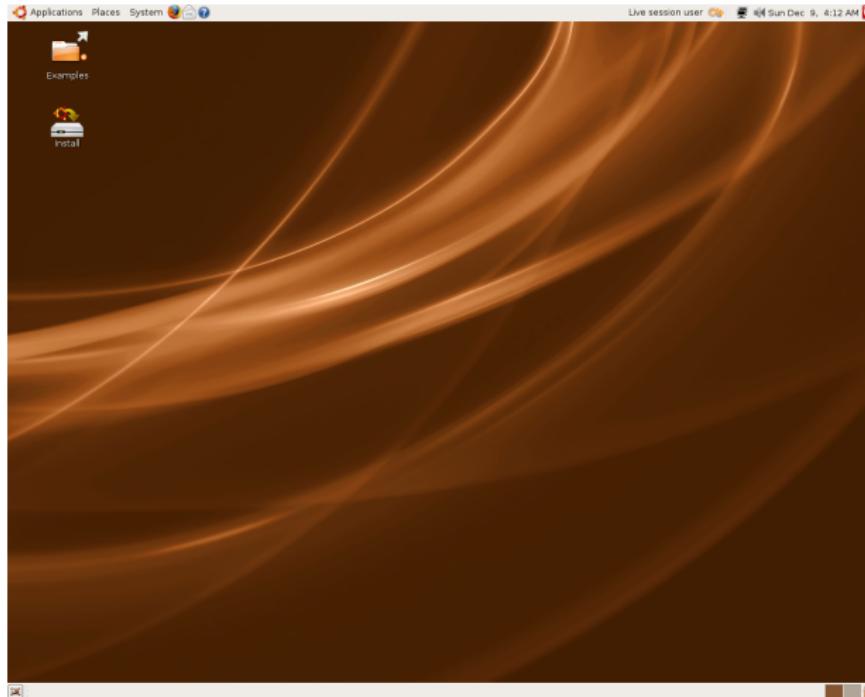
- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

# Ubuntu 7.10 的安装

把 Ubuntu 7.10 的光盘插入电脑光驱，在 BIOS 里面设置成从光驱启动，待出现安装画面后，选择 **Start or install Ubuntu**，稍等片刻，出现如下画面：



# Ubuntu 7.10 的安装

事实上，这时 Ubuntu 系统已经启动好了，你可以先体验一把。  
点击桌面上的 `install` 图标，开始安装：



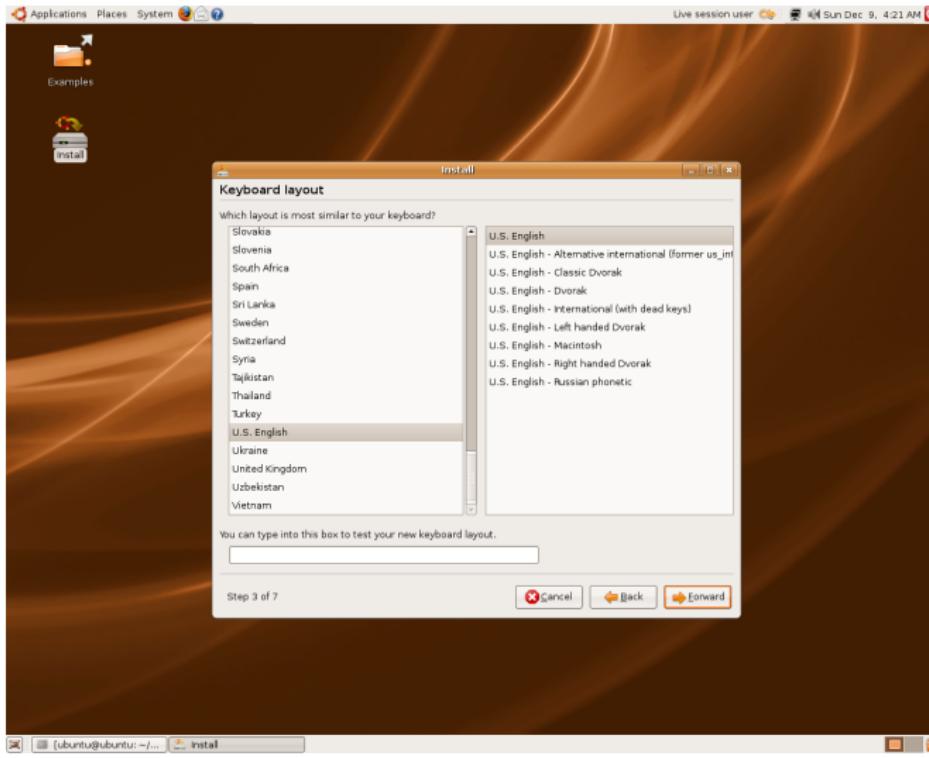
# Ubuntu 7.10 的安装

安装时，需要选择一些项目和填入一些信息，按照提示选择或填充就行。



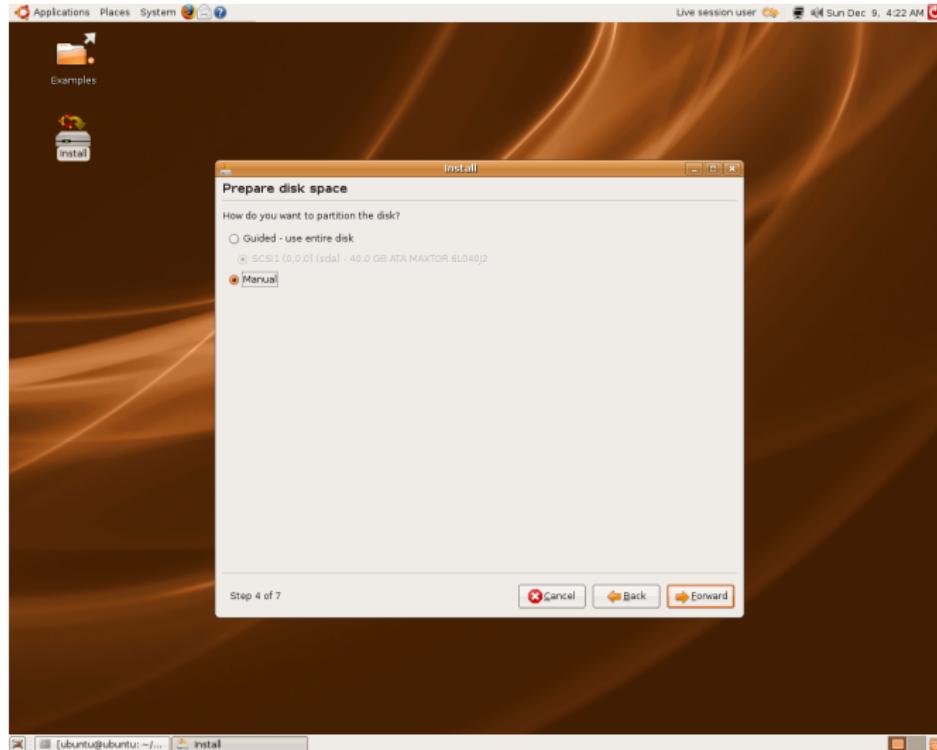
# Ubuntu 7.10 的安装

选择键盘布局:



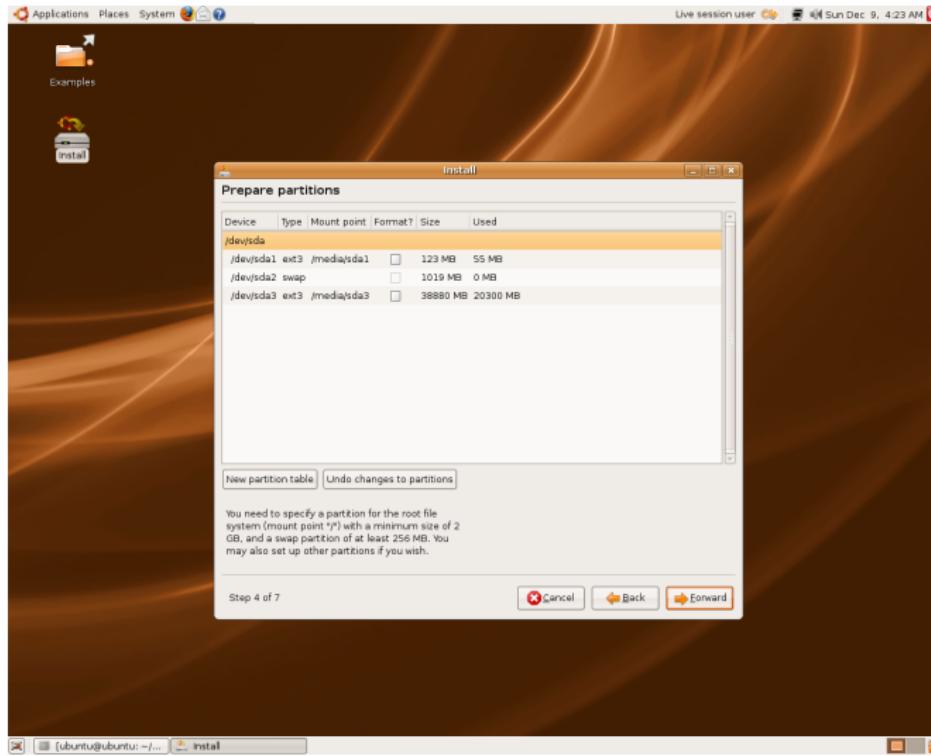
# Ubuntu 7.10 的安装

除非你系统上只装 Ubuntu，否则分区时请选择手动分区  
(**Manual**)。



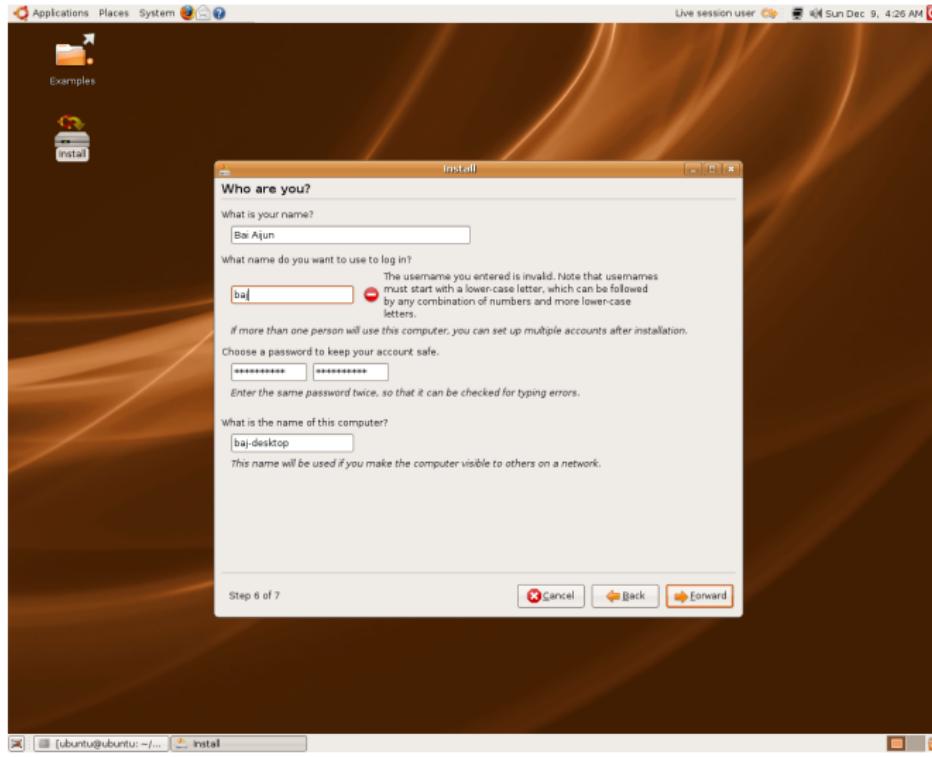
# Ubuntu 7.10 的安装

分区时，确保要有一个根分区（/）和一个交换分区（swap）。



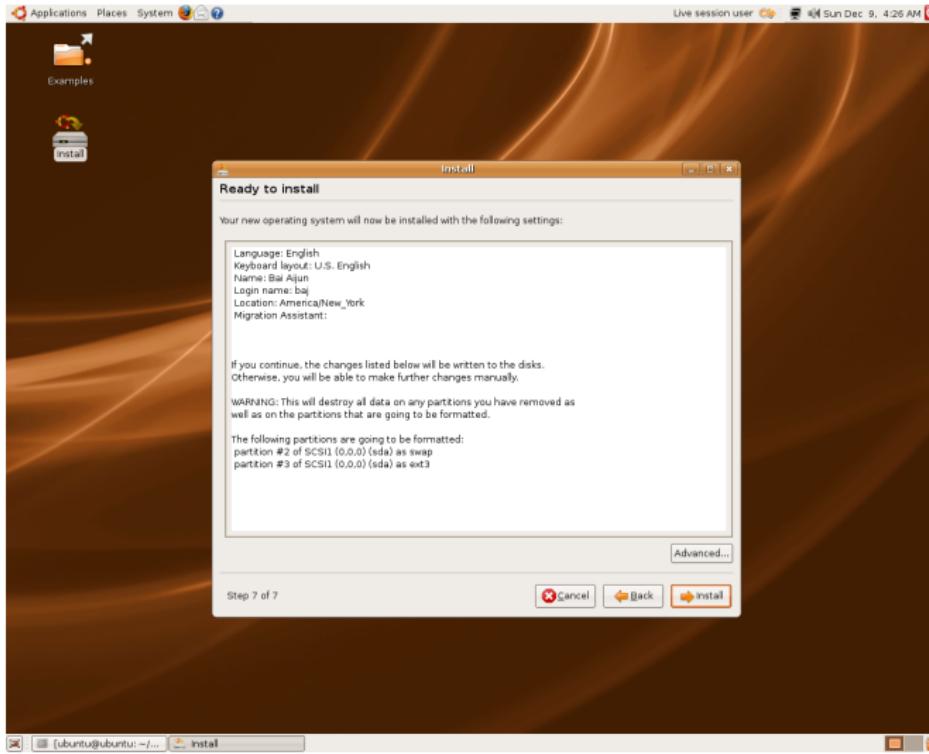
# Ubuntu 7.10 的安装

填入用户名和密码:



# Ubuntu 7.10 的安装

点击 **Install**, 开始安装:



## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

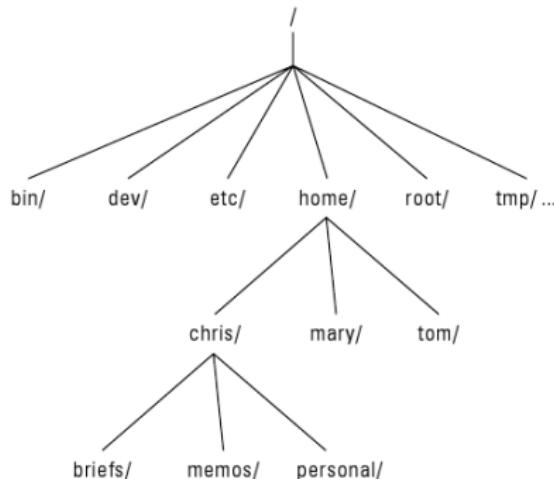
- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

Linux 下常见文件类型有：

- ① **普通文件** – 最常用和常见的一类文件，不包含有文件系统的结构信息，表现为二进制流。
- ② **目录文件** – 目录文件是用于存放文件名及其相关信息的文件，是内核组织文件系统的基本节点。
- ③ **链接文件** – 文件可以有多个别名，表现为链接，又可细分为硬链接和符号链接。
- ④ **设备文件** – 硬件设备抽象出的标准接口，以文件的形式存在。
- ⑤ **管道文件** – 用于不同进程间的信息传递。

# 目录结构

Linux 文件系统被组织成树状结构，整棵树有唯一的根，称为根目录（记为“/”），根目录下包含一些子目录，每个子目录又可以包含一些子目录。下图展示了通常的目录结构。



# 一些特殊的目录

一些目录有约定俗成的含义。

- ① /bin — 包含一些常用的用户命令
- ② /boot — 存放 Linux 内核和引导程序 (GRUB 或 LILO)
- ③ /dev — 包含一些设备文件, 如内存为 ram\*
- ④ /etc — 主要存放一些管理配置文件
- ⑤ /home — 包含普通用户的“家”目录
- ⑥ /root — 超级用户的“家”目录
- ⑦ /proc — 包含系统运行时的内核信息
- ⑧ /tmp — 临时文件, 通常会定时清理
- ⑨ /usr — 包含很多“用户级”程序和文档
- ⑩ /var — 一些经常发生变动的文件, 比如日志文件

为了保护系统的安全性，Linux 系统对不同用户访问同一文件的权限做了不同的规定，权限可以分为三种：读的权限、写的权限和执行的权限，分别用 r、w 和 x 表示。

对于一个文件来说，它都有一个特定的所有者。同时，由于在 Linux 系统中，用户是按组分类的，一个用户属于一个或多个组，文件所有者以外的用户又可以分为文件所有者的同组用户和其它用户。因此，Linux 系统按文件所有者、文件所有者同组用户和其它用户三类规定不同的文件访问权限。

# 一个实际的例子

/bin/ls

/bin/ls 文件的权限为: -rwxr-xr-x root root

表明该文件所属用户为 root, 所属组为 root, root 用户对该文件可读、可写、可执行; root 组内的用户对该文件见可读、可执行; 其他用户仅有执行权限。

## 与 MS-Dos/Windows 系统的两点区别

- ① MS 系统中用盘符（如 C:）区分不同的分区，每个分区都是一个根目录，Linux 系统中只有一个根目录，不同的分区或硬盘通过挂载的方式存在于文件系统中，所以 /usr 也许在一个独立的硬盘上，而 /home 有可能在远程的服务器上。
- ② 目录分隔符不一样，MS-系统中是 \，Linux 系统中是 /。
- ③ Linux 系统中文件名是区分大小写的，这与 MS 系统不同。

## 一些有用的标记

- . — 代表当前工作目录
- .. — 代表 . 的上一级目录
- ~ — 用户的家目录
- .\* — 隐藏文件/目录都以 . 开头，可以用 ls .\* 查看

## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- **图形界面**
- Shell 基础

## 3 C/C++ 开发环境

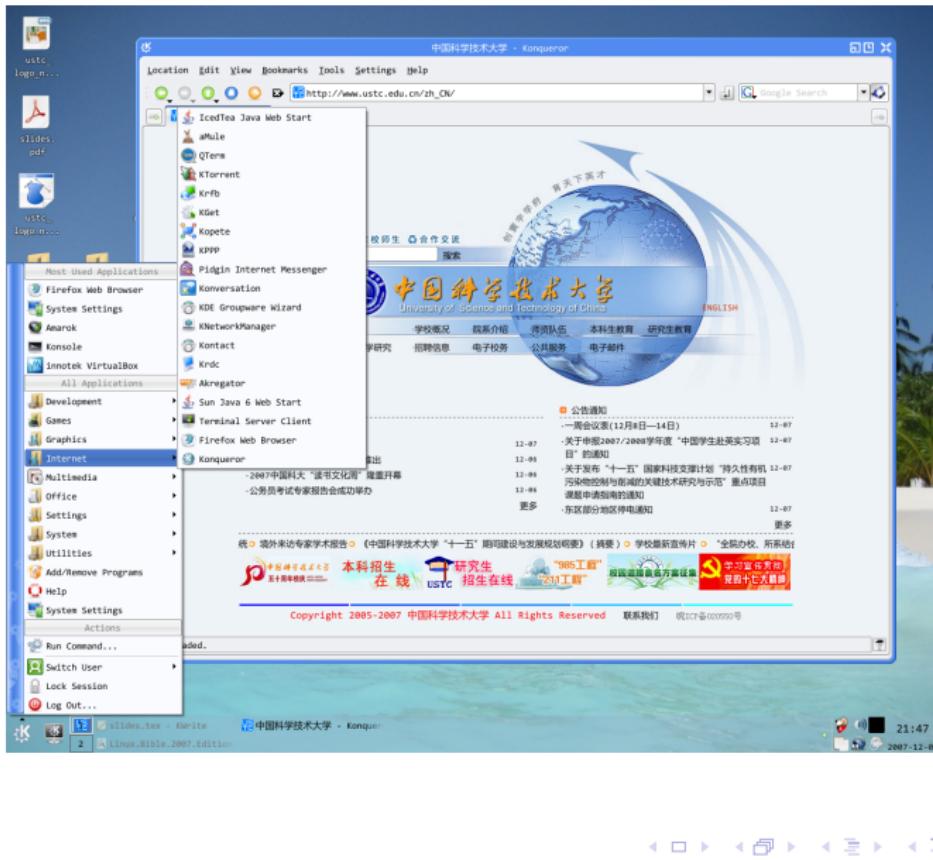
- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

X Window System 为 Linux 提供了图形界面，X 基于客户端/服务器（C/S）架构实现，对网络具有透明性。X 服务器是运行中的一个程序，X 客户端通过网络接口把各种消息、事件和指令发送给 X 服务器，X 服务器收到通知后完成图形绘制和显示。X 只提供了基本的绘图和鼠标操作的功能，但并不是一个好用的桌面环境，KDE 和 Gnome 是目前最流行的两大桌面环境，他们都是 X 客户端。

# Gnome 截图



# KDE 截图



# Gnome with compiz-fusion 截图



# KDE with compiz-fusion 截图



## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

# 什么是 Shell

类 Unix 系统与用户交互的命令行界面称之为 Shell，尽管 Shell 没有图形用户接口（GUI）直观，但很多专家级用户认为 Shell 比 GUI 更强大。

## 学习 Shell 的理由

- ① 学习一种登入类 Unix 系统并进行操作的通用技术
- ② 利用 Shell 的各种高级特性提高工作效率
- ③ 编制 Shell 脚本，完成各种复杂任务

下面以 Bourne Again Shell (bash) 为例介绍 Shell 的使用。

# 什么是 Shell

类 Unix 系统与用户交互的命令行界面称之为 Shell，尽管 Shell 没有图形用户接口（GUI）直观，但很多专家级用户认为 Shell 比 GUI 更强大。

## 学习 Shell 的理由

- ① 学习一种登入类 Unix 系统并进行操作的通用技术
- ② 利用 Shell 的各种高级特性提高工作效率
- ③ 编制 Shell 脚本，完成各种复杂任务

下面以 Bourne Again Shell (bash) 为例介绍 Shell 的使用。

类 Unix 系统与用户交互的命令行界面称之为 Shell，尽管 Shell 没有图形用户接口（GUI）直观，但很多专家级用户认为 Shell 比 GUI 更强大。

## 学习 Shell 的理由

- ① 学习一种登入类 Unix 系统并进行操作的通用技术
- ② 利用 Shell 的各种高级特性提高工作效率
- ③ 编制 Shell 脚本，完成各种复杂任务

下面以 Bourne Again Shell (bash) 为例介绍 Shell 的使用。

# 如何打开 Shell

## 使用终端模拟器

在图形界面下，可以使用桌面环境提供的终端模拟器来使用 Shell，比如 **xterm**、**gnome-terminal**、**konsole** 等（有点类似于 Windows 下的 **cmd.exe**，不过功能更强大）。

## 使用虚拟终端

很多系统都会默认启动 6 个虚拟终端（当然，这是可以自己设置的），可以用 **Ctrl+Alt+F [1-6]** 来切换使用不同的字符终端，**Ctrl+Alt+F7** 可以回到图形界面。

# 如何打开 Shell

## 使用终端模拟器

在图形界面下，可以使用桌面环境提供的终端模拟器来使用 Shell，比如 **xterm**、**gnome-terminal**、**konsole** 等（有点类似于 Windows 下的 **cmd.exe**，不过功能更强大）。

## 使用虚拟终端

很多系统都会默认启动 6 个虚拟终端（当然，这是可以自己设置的），可以用 **Ctrl+Alt+F [1-6]** 来切换使用不同的字符终端，**Ctrl+Alt+F7** 可以回到图形界面。

- **命令提示符** — 命令提示符表示系统空闲，现在可以键入命令。普通用户的默认命令提示符为 \$，root 的为 #。
- **命令种类** — 命令分为内建命令和外部命令，内建命令直接由 Shell 解释并执行，外部命令由 Shell 创建命令的进程完成执行。
- **命令选项** — 一般命令都提供一些选项用来改变执行行为，选项跟在命令后，一般以 - 或 -- 开头。
- **命令参数** — 有些命令需要参数，参数一般跟在选项后面，指明了命令执行的对象。
- **环境变量** — Shell 内部存储的关于环境的信息，比如 \$SHELL 就是当前使用的 Shell 名称（可以用 echo \$SHELL 查看）。
- **Shell 元字符** — 一些具有特殊含义的字符，如 > 表示重定向，| 表示管道。

- **命令提示符** — 命令提示符表示系统空闲，现在可以键入命令。普通用户的默认命令提示符为 \$，root 的为 #。
- **命令种类** — 命令分为内建命令和外部命令，内建命令直接由 Shell 解释并执行，外部命令由 Shell 创建命令的进程完成执行。
- **命令选项** — 一般命令都提供一些选项用来改变执行行为，选项跟在命令后，一般以 - 或 -- 开头。
- **命令参数** — 有些命令需要参数，参数一般跟在选项后面，指明了命令执行的对象。
- **环境变量** — Shell 内部存储的关于环境的信息，比如 \$SHELL 就是当前使用的 Shell 名称（可以用 echo \$SHELL 查看）。
- **Shell 元字符** — 一些具有特殊含义的字符，如 > 表示重定向，| 表示管道。

- **命令提示符** — 命令提示符表示系统空闲，现在可以键入命令。普通用户的默认命令提示符为 \$，root 的为 #。
- **命令种类** — 命令分为内建命令和外部命令，内建命令直接由 Shell 解释并执行，外部命令由 Shell 创建命令的进程完成执行。
- **命令选项** — 一般命令都提供一些选项用来改变执行行为，选项跟在命令后，一般以 - 或 -- 开头。
- **命令参数** — 有些命令需要参数，参数一般跟在选项后面，指明了命令执行的对象。
- **环境变量** — Shell 内部存储的关于环境的信息，比如 \$SHELL 就是当前使用的 Shell 名称（可以用 echo \$SHELL 查看）。
- **Shell 元字符** — 一些具有特殊含义的字符，如 > 表示重定向，| 表示管道。

- **命令提示符** — 命令提示符表示系统空闲，现在可以键入命令。普通用户的默认命令提示符为 \$，root 的为 #。
- **命令种类** — 命令分为内建命令和外部命令，内建命令直接由 Shell 解释并执行，外部命令由 Shell 创建命令的进程完成执行。
- **命令选项** — 一般命令都提供一些选项用来改变执行行为，选项跟在命令后，一般以 - 或 -- 开头。
- **命令参数** — 有些命令需要参数，参数一般跟在选项后面，指明了命令执行的对象。
- **环境变量** — Shell 内部存储的关于环境的信息，比如 \$SHELL 就是当前使用的 Shell 名称（可以用 echo \$SHELL 查看）。
- **Shell 元字符** — 一些具有特殊含义的字符，如 > 表示重定向，| 表示管道。

- **命令提示符** — 命令提示符表示系统空闲，现在可以键入命令。普通用户的默认命令提示符为 \$，root 的为 #。
- **命令种类** — 命令分为内建命令和外部命令，内建命令直接由 Shell 解释并执行，外部命令由 Shell 创建命令的进程完成执行。
- **命令选项** — 一般命令都提供一些选项用来改变执行行为，选项跟在命令后，一般以 - 或 -- 开头。
- **命令参数** — 有些命令需要参数，参数一般跟在选项后面，指明了命令执行的对象。
- **环境变量** — Shell 内部存储的关于环境的信息，比如 \$SHELL 就是当前使用的 Shell 名称（可以用 echo \$SHELL 查看）。
- **Shell 元字符** — 一些具有特殊含义的字符，如 > 表示重定向，| 表示管道。

- **命令提示符** — 命令提示符表示系统空闲，现在可以键入命令。普通用户的默认命令提示符为 \$，root 的为 #。
- **命令种类** — 命令分为内建命令和外部命令，内建命令直接由 Shell 解释并执行，外部命令由 Shell 创建命令的进程完成执行。
- **命令选项** — 一般命令都提供一些选项用来改变执行行为，选项跟在命令后，一般以 - 或 -- 开头。
- **命令参数** — 有些命令需要参数，参数一般跟在选项后面，指明了命令执行的对象。
- **环境变量** — Shell 内部存储的关于环境的信息，比如 \$SHELL 就是当前使用的 Shell 名称（可以用 echo \$SHELL 查看）。
- **Shell 元字符** — 一些具有特殊含义的字符，如 > 表示重定向，| 表示管道。

## 如何从 Shell 获得帮助

- `help` 命令可以获得 Shell 内建命令的帮助, `help <name>` 具体查看 `<name>` 命令的帮助。
- 一般命令都有 `--help` 或 `-h` 或 `-help` 选项用来显示帮助。
- `man` 提供了在线文档系统, 比如 `man man` 将显示命令 `man` 的帮助文档, `man printf` 将获得 `printf` 函数帮助文档 (要求已经安装了 `manpages-dev`)。
- `info` 可以获得更为详细和结构化的帮助文档, `info info` 查看关于 `info` 的帮助文档。

## 命令行编辑技巧

- ↑ — 回显上一命令
- <TAB><TAB> — 自动补全命令
- history — 显示命令历史
- Ctrl+R — 从历史中检索命令
- !<str> — 执行上一个包含字符串 str 的命令
- Ctrl+C — 删除一行命令

# 文件系统常用命令

命令	描述	举例
cd	改变工作目录	cd, cd WrightEagle
pwd	打印工作目录	pwd
mkdir	创建目录	mkdir WE2008
touch	创建文件	touch start.sh
chmod	改变文件的权限	chmod +x start.sh
ls	显示目录内容	ls, ls WE2008/
cp	复制文件	cp start.sh .../start
cat	合并文件并输出	cat main.cpp action.cpp
rm	删除文件或目录	rm -fr WE2008/ start.sh
mv	移动文件或目录	mv .../start start.sh

# 系统管理常用命令

命令	描述	举例
uname	查看系统信息	uname -a
id	查看用户自己的身份	id
whoami	看看我是谁	whoami
passwd	更改密码	passwd, passwd root
uptime	查看系统运行时间	uptime
ps	查看运行中的进程	ps -e, ps aux
kill	根据 pid 杀死进程	kill 1647
killall	根据进程名杀死进程	killall WE2008

# 其他常用命令

命令	描述	举例
gzip	压缩/解压文件	gzip README, gzip -d README.gz
tar	打包/解包文件	tar czvf WE2008.tar.gz WE2008/
find	查找文件	find . -name '*.log' -print
grep	查找字符串	grep 'main' main.cpp
echo	回显一行内容	echo yes, echo \$[5+9]

## I/O 重定向

- < — 输入重定向
- > — 输出重定向，截断已有文件
- >> — 输出重定向，追加到已有文件

举例：

- cat a b >> c — 合并文件 a 和 b，追加到 c
- find . 2>/dev/null — 忽略出错信息
- ./start 1>/dev/null 2>&1 — 忽略所有输出

## I/O 重定向

- < — 输入重定向
- > — 输出重定向，截断已有文件
- >> — 输出重定向，追加到已有文件

举例：

- cat a b >> c — 合并文件 a 和 b，追加到 c
- find . 2>/dev/null — 忽略出错信息
- ./start 1>/dev/null 2>&1 — 忽略所有输出

## 管道

管道 (|) 把前一个命令的 stdout 连接到后一个命令的 stdin。

举例：

- ps -e | grep WE2008 — 查看 WE2008 是否在运行
- cat a | less — 分屏查看文件 a 的内容
- ls | sort — 排序显示当前目录下的内容
- cat a | sort > b — 将文件 a 排序后的结果保存成 b

## 管道

管道 (|) 把前一个命令的 stdout 连接到后一个命令的 stdin。

举例：

- ps -e | grep WE2008 — 查看 WE2008 是否在运行
- cat a | less — 分屏查看文件 a 的内容
- ls | sort — 排序显示当前目录下的内容
- cat a | sort > b — 将文件 a 排序后的结果保存成 b

# 有用的技术

## 文件通配符

文件通配符主要在 Shell 里面使用，主要用来匹配文件或目录名。

举例：

- ? — 可替代单个字符
- \* — 可替代任意字符
- [<charset>] — 匹配 <charset> 集合中的字符
- [<1>-<2>] — 匹配字符 <1> 到 <2> 之间的字符，如 [a-zA-Z]



# 有用的技术

## 文件通配符

文件通配符主要在 Shell 里面使用，主要用来匹配文件或目录名。

举例：

- ? — 可替代单个字符
- \* — 可替代任意字符
- [<charset>] — 匹配 <charset> 集合中的字符
- [<1>-<2>] — 匹配字符 <1> 到 <2> 之间的字符，  
如 [a-zA-Z]



# 有用的技术

## 正则表达式

正则表达式（RE）提供了精确匹配字符串的方法，很多 Linux 下的工具都完全支持 RE，如 Vim、grep、sed、awk 等。

举例：

- \<a>[ ]\*=[^=]\ — 匹配所有变量 a 被赋值的地方
- \<.\*n\> — 匹配以 n 结尾的单词
- ^a.\*b\$ — 匹配以 a 开头，以 b 结尾的行



# 有用的技术

## 正则表达式

正则表达式（RE）提供了精确匹配字符串的方法，很多 Linux 下的工具都完全支持 RE，如 Vim、grep、sed、awk 等。

举例：

- \<a>[ ]\*= [^=] \ — 匹配所有变量 a 被赋值的地方
- \<.\*n\> — 匹配以 n 结尾的单词
- ^a.\*b\$ — 匹配以 a 开头，以 b 结尾的行



## 作业控制

- 有些命令执行时可能会花费比较多的时间，如果直接敲命令执行，就要等一些时间才能返回命令提示符，可以选择让这些命令在后台执行（只要在命令后加上 &），使 Shell 立即返回命令提示符，以便进行其它操作。
- Ctrl+Z 挂起一个执行中的命令，fg 以前台形式恢复命令的执行，bg 以后台形式恢复命令执行。
- 可以用 Ctrl+C 终断一个命令的执行。
- 有些命令屏蔽了终断信号，以致于 Ctrl+C 不起作用，这时可以用 Ctrl+\ 强制杀死。

## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- 管理一个项目
- 有用的工具

程序员最常用的编辑器是 Emacs 和 Vim（Vim 代表 Vi Improved，是对 Vi 的改进），其中 Vim 比较容易上手，下面介绍 Vim 的基本使用。

## Vim 的特点

- 纯字符界面，不需要图形系统的支持。
- 灵活的定制特性，方便扩展功能。
- 很多高级特性，提高工作效率。
- 还有很多……

程序员最常用的编辑器是 Emacs 和 Vim（Vim 代表 Vi Improved，是对 Vi 的改进），其中 Vim 比较容易上手，下面介绍 Vim 的基本使用。

## Vim 的特点

- 纯字符界面，不需要图形系统的支持。
- 灵活的定制特性，方便扩展功能。
- 很多高级特性，提高工作效率。
- 还有很多……

程序员最常用的编辑器是 Emacs 和 Vim（Vim 代表 Vi Improved，是对 Vi 的改进），其中 Vim 比较容易上手，下面介绍 Vim 的基本使用。

## Vim 的特点

- 纯字符界面，不需要图形系统的支持。
- 灵活的定制特性，方便扩展功能。
- 很多高级特性，提高工作效率。
- 还有很多……

## 用 Vim 打开文件:

vi main.cpp 打开 main.cpp 文件

vi a +12 打开 a, 光标定位到第十二行

vimdiff a b 比较文件 a 和 b 的区别

## Vim 的工作模式

Vim 区分命令模式和编辑模式。Vim 刚启动时处于命令模式，命令模式下所有的按键都试图被解释成命令，可以完成一些常见操作，比如删除、复制、粘贴、替换等，此时输入编辑命令便切换到编辑模式，如 i 或 a；编辑模式下，所有按键都被视为输入，直接反映到所编辑文件的内容上；编辑模式下按 ESC 切换到命令模式。

## 用 Vim 打开文件:

vi main.cpp 打开 main.cpp 文件

vi a +12 打开 a, 光标定位到第十二行

vimdiff a b 比较文件 a 和 b 的区别

## Vim 的工作模式

Vim 区分命令模式和编辑模式。Vim 刚启动时处于命令模式，命令模式下所有的按键都试图被解释成命令，可以完成一些常见操作，比如删除、复制、粘贴、替换等，此时输入编辑命令便切换到编辑模式，如 i 或 a；编辑模式下，所有按键都被视为输入，直接反映到所编辑文件的内容上；编辑模式下按 ESC 切换到命令模式。

## 光标定位

- h, j, k, l — ←, ↓, ↑, →
- w, b — 移到下/前一个单词
- 0, \$ — 移动到一行开始/结束的地方
- % — 跳转到下一个匹配的括号

## 退出 Vim

- :w — 保存修改后的文件
- :q — 退出 Vim
- :wq — 保存并退出
- :q! — 强制退出

## 光标定位

- h, j, k, l — ←, ↓, ↑, →
- w, b — 移到下/前一个单词
- 0, \$ — 移动到一行开始/结束的地方
- % — 跳转到下一个匹配的括号

## 退出 Vim

- :w — 保存修改后的文件
- :q — 退出 Vim
- :wq — 保存并退出
- :q! — 强制退出

## 编辑操作

- i, a — 在光标左/右边插入字符，切换到编辑模式
- o, O — 在当前行下/上新建一行，切换到编辑模式
- x, X — 删除光标下/前的字符
- dw, db — 从光标处删到下/前一个单词
- d\$, d0 — 从光标处删到行尾/首
- dd — 删除一行
- yy — 复制一行
- p, P — 在当前行下/上粘贴寄存器内容

## 查找和替换

- `/`, `?` — 向前/后搜索单词, 如 `/hello`
- `:g/hello` — 打印所有出现 `hello` 的行
- `:%s/hallo/hello/g` — 把所有的 `hallo` 替换成 `hello`

## 其它命令

- `u`, `Ctrl+R` — 撤销/恢复上一操作
- `:<num>` — 跳转到第 `<num>` 行
- `:$` — 跳转到最后一行
- `gd`, `gD` — 跳转到光标下变量局部/全局声明处
- `Ctrl+G` — 显示当前编辑的行的信息, 如行号等
- `Ctrl+P` — 编辑模式下, 自动补全单词

## 查找和替换

- /, ? — 向前/后搜索单词, 如 /hello
- :g/Hello — 打印所有出现 Hello 的行
- :%s/Hallo/Hello/g — 把所有的 Hallo 替换成 Hello

## 其它命令

- u, Ctrl+R — 撤销/恢复上一操作
- :<num> — 跳转到第 <num> 行
- :\$ — 跳转到最后一行
- gd, gD — 跳转到光标下变量局部/全局声明处
- Ctrl+G — 显示当前编辑的行的信息, 如行号等
- Ctrl+P — 编辑模式下, 自动补全单词

## 1 初探 Linux

- 背景知识
- 安装 Linux

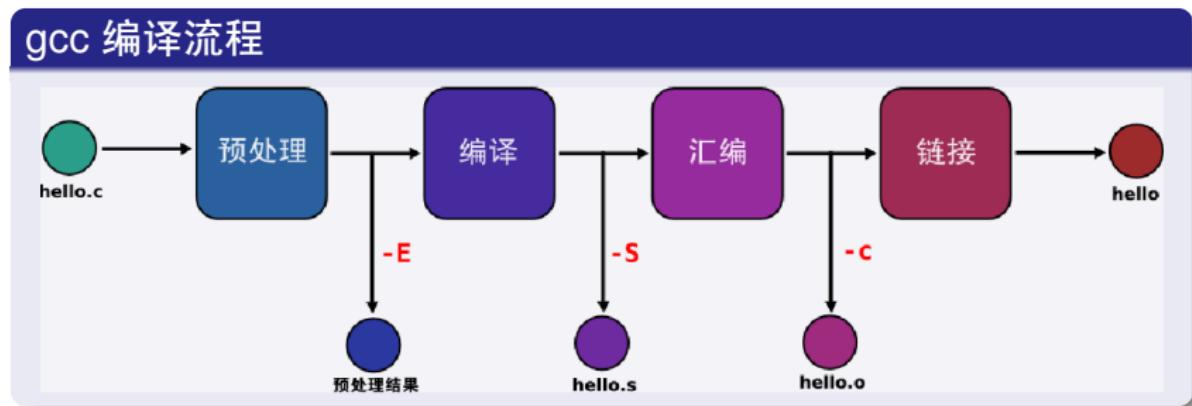
## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器**
- 管理一个项目
- 有用的工具

目前 Linux 下最常用的 C 语言编译器是 GCC ,它是 GNU 项目中符合 ANSI C 标准的编译系统, 能够编译用 C、C++ 和 Object C 等语言编写的程序。



# 编译和运行

## helloworld 的编译和运行

```
$ vi hello.c
$ cat hello.c
#include <stdio.h>
int main (int argc, char* argv[])
{
    printf("Hello World!");
    return 0;
}
$ gcc hello.c -o hello
$ ls
hello hello.c
$ ./hello
Hello World!
```

## 默认文件类型

.c	C 源代码
.cc .cxx .cpp .C	C++ 源代码
.h	C/C++ 头文件
.hh .hpp .H	C++ 头文件

gcc 可以编译 C++ 程序，但是不能自动链接到 stdc++ 类库，GCC 提供了一个命令 g++ 处理这个问题。

## C++ 程序编译示例

```
$ g++ hello.cc -o hello
```

## 默认文件类型

.c	C 源代码
.cc .cxx .cpp .C	C++ 源代码
.h	C/C++ 头文件
.hh .hpp .H	C++ 头文件

gcc 可以编译 C++ 程序，但是不能自动链接到 stdc++ 类库，GCC 提供了一个命令 g++ 处理这个问题。

## C++ 程序编译示例

```
$ g++ hello.cc -o hello
```

## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- **管理一个项目**
- 有用的工具

前面只简单介绍了只有一个源文件的项目的编译过程，如果一个项目有几十、几百个源文件，还按照前面的方法逐一编译，其效率是无法接受的。这时可以使用 GNU Make 工具帮助管理代码。

Make 的执行需要事先已经写好 Makefile。Makefile 由一些规则组成，每条规则分为三个部分：

- ① 目标 — 本条规则需要完成的目标
- ② 依赖关系 — 目标所依赖的先决条件
- ③ 命令 — 依赖关系满足时，执行的命令

# Makefile 实例

前面的 `hello.c` 的 Makefile 可以这样写：

## Makefile

```
hello: hello.c  
    gcc hello.c -o hello
```

这样，就可以直接敲命令 `make` 来自动完成编译了。

### ♥ 小提示

命令要另起一行写，并且以 `<TAB>` 开头

# Makefile 实例

前面的 `hello.c` 的 Makefile 可以这样写：

## Makefile

```
hello: hello.c  
    gcc hello.c -o hello
```

这样，就可以直接敲命令 `make` 来自动完成编译了。

## ♥ 小提示

命令要另起一行写，并且以 `<TAB>` 开头

# Makefile 高级特性

```
GCC = gcc
CPPFLAGS = -g
LDFLAGS = -g
EXEC = hello
SRC = hello.c
OBJ = $(SRC:.c=.o)
$(EXEC) : $(OBJ)
    $(GCC) $(LDFLAGS) $^ -o $@
%.o: %.c %.h
    $(GCC) $(CPPFLAGS) -c $< -o $@
clean:
    rm -f *.o $(EXEC)
```

自动扩展的宏：

`$@` 完全的目标名

`$<` 第一个先决条件

`$^` 所有先决条件

## 1 初探 Linux

- 背景知识
- 安装 Linux

## 2 Linux 基础

- 文件系统
- 图形界面
- Shell 基础

## 3 C/C++ 开发环境

- 编辑器
- 编译器
- 管理一个项目
- 有用的工具**

# 一些有用的工具

gdb	GNU 调试器，功能很强大
valgrind	调试和分析程序，经常用来检查有无内存泄漏
gprof	分析函数调用过程，可自动给出调用关系图
ctags	建立源文件索引，方便检索，可内嵌 Vim
automake	自动生成跨平台的 Makefile
cvs / svn	版本控制系统

- 《*Linux.Bible.2007.Edition*》  
*by Christopher Negus*  
— 参考了很多书中的内容
- 《*GNU/Linux 下的 C/C++ 编程环境简介*》  
*by 詹剑 <zhanjian@ustc.edu>*  
— 参考了很多开发环境方面的内容

# 谢谢！