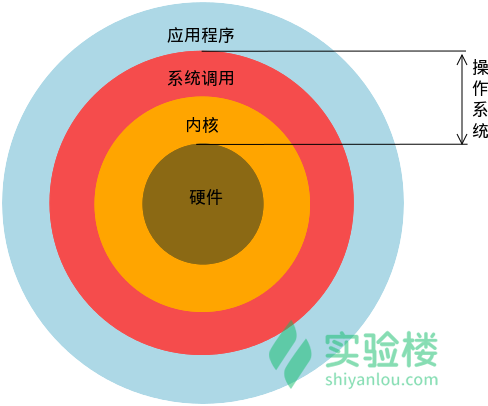
# Linux系统简介

## 1.2 知识点

* linux为何物
* linux历史简介
* linux重要人物
* linux与windows的不同
* 如何学习linux

### linux为何物

Linux 就是一个操作系统，就像你多少已经了解的 Windows（xp，7，8）和 Mac OS 。至于操作系统是什么，就不用过多解释了，如果你学习过前面的入门课程，应该会有个基本概念了，这里简单介绍一下操作系统在整个计算机系统中的角色。



我们的 Linux 主要是系统调用和内核那两层。当然直观地看，我们使用的操作系统还包含一些在其上运行的应用程序，比如文本编辑器、浏览器、电子邮件等。

### Linux历史

操作系统始于二十世纪五十年代，当时的操作系统能运行批处理程序。批处理程序不需要用户的交互，它从文件或者穿孔卡片读取数据，然后输出到另外一个文件或者打印机。

二十世纪六十年代初，交互式操作系统开始流行。它不仅仅可以交互，还能使多个用户从不同的终端同时操作主机。这样的操作系统被称作分时操作系统，它的出现对批处理操作系统是个极大的挑战。许多人尝试开发分时操作系统， 其中包括一些大学的研究项目和商业项目。当时有个项目叫做 Multics ，它的技术在当时很具有创新性。 Multics 项目的开发并不顺利，它花费了远超过预计的资金，却没有在操作系统市场上占到多少份额。而参加该项目的一个开发团体——贝尔实验室退出了这个项目。他们在退出后开发了他们自己的一个操作系统—— UNIX 。

UNIX 最初免费发布并因此在大学里受到欢迎。后来，UNIX 实现了 TCP/IP 协议栈，成为了早期工作站的操作系统的一个流行选择。

1990 年，UNIX 在服务器市场上尤其是大学校园中成为主流操作系统，许多校园都有 UNIX 主机，当然还包括一些研究它的计算机系的学生。这些学生都渴望能在自己的电脑上运行 UNIX 。不幸的是，从那时候开始，UNIX 开始变得商业化，它的价格也变得非常昂贵。而唯一低廉的选择就是 MINIX，这是一个功能有限的类似 UNIX 的操作系统，作者 Andrew Tanenbaum 开发它的目的是用于教学。

1991 年 10 月，Linus Torvalds（Linux 之父）在赫尔辛基大学接触 UNIX，他希望能在自己的电脑上运行一个类似的操作系统。可是 UNIX 的商业版本非常昂贵，于是他从 MINIX 开始入手，计划开发一个比 MINIX 性能更好的操作系统。很快他就开始了自己的开发工作。他第一次发行的版本迅速吸引了一些黑客。尽管最初的 Linux 并没有多少用处，但由于一些黑客的加入使它很快就具有了许多吸引人的特性，甚至一些对操作系统开发不感兴趣的人也开始关注它。

Linux 本身只是操作系统的内核。内核是使其它程序能够运行的基础。它实现了多任务和硬件管理，用户或者系统管理员交互运行的所有程序实际上都运行在内核之上。其中有些程序是必需的，比如说，命令行解释器（shell），它用于用户交互和编写 shell 脚本。 Linux 没有自己去开发这些应用程序，而是使用已有的自由软件。这减少了搭建开发环境所需花费的工作量。实际上，他经常改写内核，使得那些程序能够更容易地在 Linux 上运行。许多重要的软件，包括 C 编译器，都来自于自由软件基金 GNU 项目。GNU 项目开始于 1984 年，目的是为了开发一个完全类似于 UNIX 的免费操作系统。为了表扬 GNU 对 Linux 的贡献，许多人把 Linux 称为 GNU/Linux（GNU 有自己的内核）。

1992－1993 年，Linux 内核具备了挑战 UNIX 的所有本质特性，包括 TCP/IP 网络，图形界面系统（X window )，Linux 同样也吸引了许多行业的关注。一些小的公司开始开发和发行 Linux，有几十个 Linux 用户社区成立。1994 年，Linux 杂志也开始发行。

Linux 内核 1.0 在 1994 年 3 月发布，内核的发布要经历许多开发周期，直至达到一个稳定的版本。

下面列举一些 Linux 诞生大事件：

* 1965 年，Bell 实验室、MIT、GE（通用电气公司）准备开发 Multics 系统，为了同时支持 300 个终端访问主机，但是 1969 年失败了；

那时候并没有鼠标、键盘，输入设备，只有卡片机。因此，如果要测试某个程序，则需要将读卡纸插入卡片机，如果有错误，还需要重新来过；Multics：Multiplexed Information and Computing Service；

* 1969 年，Ken Thompson（C语言之父）利用汇编语言开发了 File Server System（Unics，即 UNIX 的原型）；

因为汇编语言对于硬件的依赖性，因此只能针对特定硬件； 只是为了移植一款“太空旅游”的游戏；

* 1973 年，Dennis Ritchie 和 Ken Thompson 发明了 C 语言，而后写出了 UNIX 的内核；

将 B 语言改成 C 语言，由此产生了 C 语言之父；90% 的代码是 C 语言写的，10% 的代码用汇编语言写的，因此移植时只要修改那 10% 的代码即可；

* 1977 年，Berkeley 大学的 Bill Joy 针对他的机器修改了 UNIX 源码，称为BSD（Berkeley Software Distribution）；

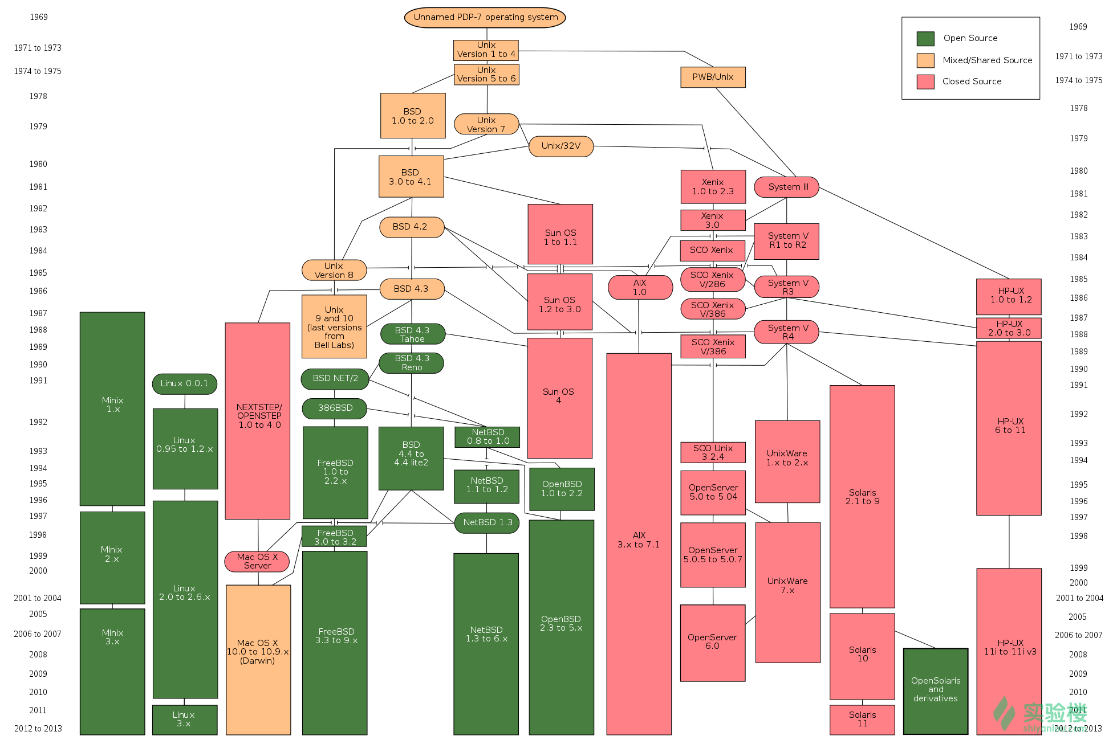
Bill Joy 是 Sun 公司的创始人；

* 1979 年，UNIX 发布 System V，用于个人计算机；
* 1984 年，因为 UNIX 规定“不能对学生提供源码”，Tanenbaum 老师自己编写兼容于 UNIX 的 Minix，用于教学；
* 1984 年，Stallman 开始 GNU（GNU's Not Unix）项目，创办 FSF（Free Software Foundation）基金会；

产品：GCC、Emacs、Bash Shell、GLIBC；倡导“自由软件”；GNU 的软件缺乏一个开放的平台运行，只能在 UNIX 上运行；自由软件指用户可以对软件做任何修改，甚至再发行，但是始终要挂着 GPL 的版权；自由软件是可以卖的，但是不能只卖软件，而是卖服务、手册等；

* 1985 年，为了避免 GNU 开发的自由软件被其他人用作专利软件，因此创建 GPL（General Public License）版权声明；
* 1988 年，MIT 为了开发 GUI，成立了研发 XFree86 的组织；
* 1991 年，芬兰赫尔辛基大学的研究生 Linus Torvalds 基于 gcc、bash 开发了针对 386 机器的 Linux 内核；
* 1994 年，Torvalds 发布 Linux-v1.0；
* 1996 年，Torvalds 发布 Linux-v2.0，确定了 Linux 的吉祥物：企鹅。

UNIX进化史（UNIX大家族族谱1969-2013）：



## Linux历史人物

### 1. Ken Thompson：C 语言之父和 UNIX 之父



### 2. Dennis Ritchie：C 语言之父和 UNIX 之父



### 3. Stallman：著名黑客，GNU 创始人，开发了 Emacs、gcc、bash shell



### 4. Bill Joy：BSD 开发者



### 5. Tanenbaum：Minix 开发者



### 6. Linus Torvalds：Linux 之父，芬兰赫尔辛基大学



## Linux和windows不同

### 1. 免费与收费

* 最新正版 Windows 10，需要付费购买；
* Linux 免费或少许费用。

### 2. 软件与支持

* Windows 平台：数量和质量的优势，不过大部分为收费软件；由微软官方提供重要支持和服务；
* Linux 平台：大都为开源自由软件，用户可以修改定制和再发布，由于基本免费没有资金支持，部分软件质量和体验欠缺；由全球所有的 Linux 开发者和自由软件社区提供支持。

### 3. 安全性

* Windows 平台：三天两头打补丁安装系统安全更新，还是会中病毒木马；
* Linux 平台：要说 Linux 没有安全问题，那当然是不可能的，这一点仁者见仁智者见智，相对来说肯定比 Windows 平台要更加安全，使用 Linux 你也不用装某杀毒、某毒霸。

### 4. 使用习惯

* Windows：普通用户基本都是纯图形界面下操作使用，依靠鼠标和键盘完成一切操作，用户上手容易，入门简单；
* Linux：兼具图形界面操作（需要使用带有桌面环境的发行版）和完全的命令行操作，可以只用键盘完成一切操作，新手入门较困难，需要一些学习和指导（这正是我们要做的事情），一旦熟练之后效率极高。

### 5. 可定制性

* Windows：这些年之前算是全封闭的，系统可定制性很差；
* Linux：你想怎么做就怎么做，Windows 能做到得它都能，Windows 做不到的，它也能。

### 6. 应用范畴

或许你之前不知道 Linux ，要知道，你之前在 Windows 使用百度、谷歌，上淘宝，聊 QQ 时，支撑这些软件和服务的，是后台成千上万的 Linux 服务器主机，它们时时刻刻都在忙碌地进行着数据处理和运算，可以说世界上大部分软件和服务都是运行在 Linux 之上的。

### 7. Windows 没有的

* 稳定的系统
* 安全性和漏洞的快速修补
* 多用户
* 用户和用户组的规划
* 相对较少的系统资源占用
* 可定制裁剪，移植到嵌入式平台（如安卓设备）
* 可选择的多种图形用户界面（如 GNOME，KDE）

### 8. Linux 没有的

* 特定的支持厂商
* 足够的游戏娱乐支持度
* 足够的专业软件支持度

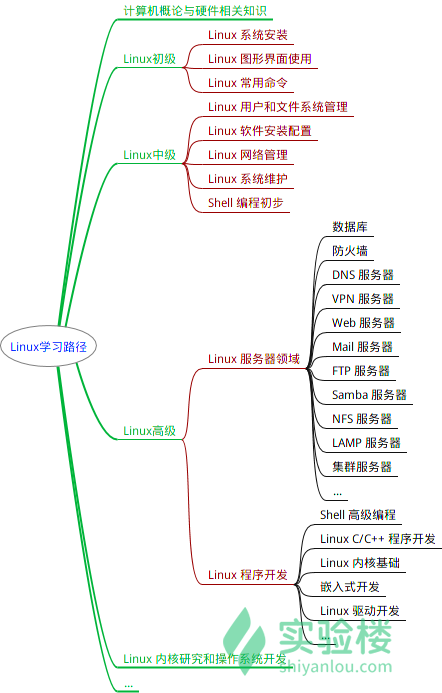
## 如何学习linux

### 1. 学习心态

* 明确目的：你是要用 Linux 来干什么，搭建服务器、做程序开发、日常办公，还是娱乐游戏；
* 面对现实：Linux 大都在命令行下操作，能否接受不用或少用图形界面；
* 是学习 Linux 操作系统本身还是某一个 Linux 发行版（[Ubuntu](http://www.ubuntu.com/)，[CentOS](http://www.centos.org/" \t "_blank)，[Fedora](http://fedoraproject.org/" \t "_blank)，[OpenSUSE](http://www.opensuse.org/" \t "_blank)，[Debian](http://www.debian.org/" \t "_blank)，[Mint](http://linuxmint.com/" \t "_blank) 等等），如果你对发行版的概念或者它们之间的关系不明确的话可以参看 [Linux 发行版](http://baike.baidu.com/view/897468.htm)。

### 2. 注重基础，从头开始

大致的学习路径如下：



# 基本概念和操作

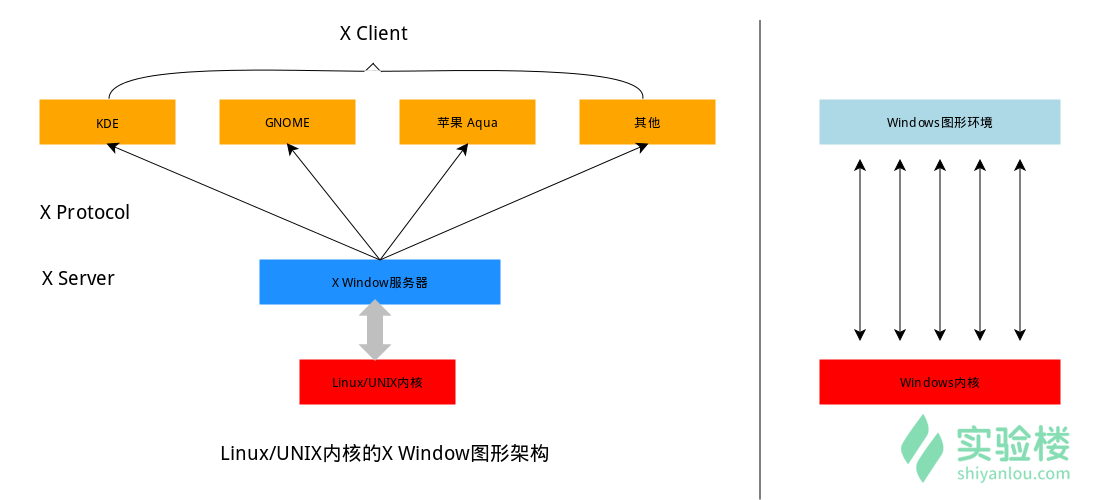
## 1.2 实验知识点

* Linux 基本命令
* 通配符的使用
* 查看帮助文档

## 桌面环境

### 1.Linux 桌面环境介绍

相对于现在的 Windows 系统，UNIX/Linux 本身是没有图形界面的，我们通常在 UNIX/Linux 发行版上看到的图形界面实际都只是运行在 Linux 系统之上的一套软件，类似 Windows95 之前的 Windows 的图形界面实则也只是运行在 DOS 环境的一套软件。而 Linux 上的这套软件以前是 XFree86，现在则是 xorg（X.Org），而这套软件又是通过 X 窗口系统（X Window System，也常被称为 X11 或 X）实现的，X 本身只是工具包及架构协议，而 xorg 便是 X 架构规范的一个实现体，也就是说它是实现了 X 协议规范的一个提供图形界面服务的服务器，就像实现了 http 协议提供 web 服务的 Apache 。如果只有服务器也是不能实现一个完整的桌面环境的，当然还需要一个客户端，我们称为 X Client，像如下几个大家熟知也最流行的实现了客户端功能的桌面环境 **KDE**，**GNOME**，**XFCE**，**LXDE** 。其中就有你看到的，实验楼目前使用的 **XFCE** 桌面环境，部分老用户可能可以回想起，实验楼之前使用的环境是 **LXDE** 。这也意味着在 Linux 上你可以自己选择安装不同的桌面环境，甚至可以定制自己的专属桌面。



这里讲这么多，主要是为了帮助你更好地理解 Linux 的桌面环境是一个怎样的概念，以及它跟 Windows 操作系统桌面的差异，实际它们之间是有着本质的差别的，希望清楚这些内容之后可以让你忽略那些体验上的差异，专心把精力放到 Linux 系统本身或者你正准备学习的某一门技术之上，这门实验课后面的内容和实验楼的其它课程便是为了帮助你实现以上两个目标。

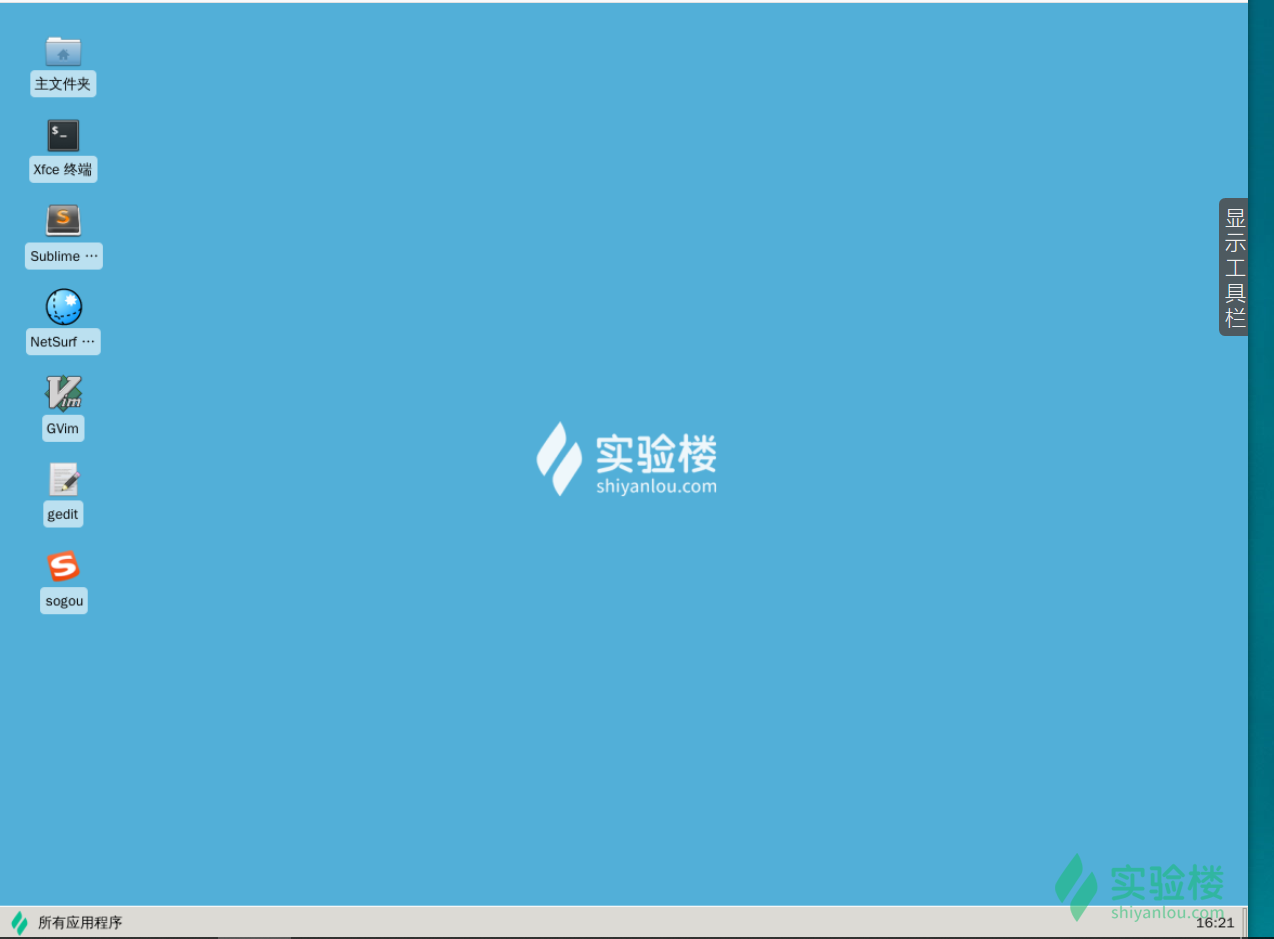
### 2.Linux 桌面环境的使用

如果你对当前实验桌面环境（xfce）的使用有任何困难，建议你先学习[实验楼入门基础课程](http://shiyanlou.com/courses/63)，对于其它桌面环境的使用，不在本课程的范围之内，有兴趣的用户可以在自己的计算机中安装完整的 Linux 发行版或其 Live CD 体验。

## Linux终端

### 终端的概念

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

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

### Shell

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

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

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

### 命令行

#### 写在前面

在linux中，最最重要的就是命令，这就包含了2个过程，输入和输出

* 输入：输入当然就是打开终端，然后按键盘输入，然后按回车，输入格式一般就是这类的

#创建一个名为 file 的文件，touch是一个命令

$ touch file

#进入一个目录，cd是一个命令

$ cd /etc/

#查看当前所在目录

$ pwd

* 输出：输出会返回你想要的结果，比如你要看什么文件，就会返回文件的内容。如果只是执行，执行失败会告诉你哪里错了，如果执行成功那么会没有输出，因为linux的哲学就是：没有结果就是最好的结果

#### 开始

如图，双击桌面上的 Xfce 终端 图标打开终端后系统会自动运行 Shell 程序，然后我们就可以输入命令让系统来执行了：

#### 1) 重要快捷键：

真正学习命令行之前，你先要掌握几个十分有用、必需掌握的小技巧：

##### [Tab]

使用Tab键来进行命令补全，Tab键一般是在字母Q旁边，这个技巧给你带来的最大的好处就是当你忘记某个命令的全称时可以只输入它的开头的一部分，然后按下Tab键就可以得到提示或者帮助完成：

##### [Ctrl+c]

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

尝试输入以下命令：

$ tail

然后你会发现你接下来的输入都没有任何反应了，只是将你输入的东西显示出来，现在你可以使用Ctrl+c，来中断这个你目前可能还不知道是什么的程序（在后续课程中我们会具体解释这个tail命令是什么）。

又或者输入：

$ find /

显然这不是你想的结果，可以使用Ctrl+c结束。

虽然这个按着很方便，但不要随便按，因为有时候，当你看到终端没有任何反应或提示，也不能接受你的输入时，可能只是运行的程序需要你耐心等一下，就不要急着按Ctrl+c了。

##### 其他一些常用快捷键

| **按键** | **作用** |
| --- | --- |
| Ctrl+d | 键盘输入结束或退出终端 |
| Ctrl+s | 暂停当前程序，暂停后按下任意键恢复运行 |
| Ctrl+z | 将当前程序放到后台运行，恢复到前台为命令fg |
| Ctrl+a | 将光标移至输入行头，相当于Home键 |
| Ctrl+e | 将光标移至输入行末，相当于End键 |
| Ctrl+k | 删除从光标所在位置到行末 |
| Alt+Backspace | 向前删除一个单词 |
| Shift+PgUp | 将终端显示向上滚动 |
| Shift+PgDn | 将终端显示向下滚动 |

#### 2) 学会利用历史输入命令

很简单，你可以使用键盘上的方向上键↑，恢复你之前输入过的命令，你一试便知。

#### 3) 学会使用通配符

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

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

首先回到用户家目录：

$ cd /home/shiyanlou

然后使用 touch 命令创建 2 个文件，后缀都为 txt：

$ touch asd.txt fgh.txt

可以给文件随意命名，假如过了很长时间，你已经忘了这两个文件的文件名，现在你想在一大堆文件中找到这两个文件，就可以使用通配符：

$ ls \*.txt

在创建文件的时候，如果需要一次性创建多个文件，比如：**“love\_1\_linux.txt，love\_2\_linux.txt，... love\_10\_linux.txt”**。在 Linux 中十分方便：

$ touch love\_{1..10}\_shiyanlou.txt

Shell 常用通配符：

| **字符** | **含义** |
| --- | --- |
| \* | 匹配 0 或多个字符 |
| ? | 匹配任意一个字符 |
| [list] | 匹配 list 中的任意单一字符 |
| [^list] | 匹配 除list 中的任意单一字符以外的字符 |
| [c1-c2] | 匹配 c1-c2 中的任意单一字符 如：[0-9] [a-z] |
| {string1,string2,...} | 匹配 string1 或 string2 (或更多)其一字符串 |
| {c1..c2} | 匹配 c1-c2 中全部字符 如{1..10} |

#### 4) 学会在命令行中获取帮助

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

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

你可以使用如下方式来获得某个命令的说明和使用方式的详细介绍：

$ man <command\_name>

比如你想查看 man 命令本身的使用方式，你可以输入：

man man

通常情况下，man 手册里面的内容都是英文的，这就要求你有一定的英文基础。man 手册的内容很多，涉及了 Linux 使用过程中的方方面面。为了便于查找，man 手册被进行了分册（分区段）处理，在 Research UNIX、BSD、OS X 和 Linux 中，手册通常被分为8个区段，安排如下：

| **区段** | **说明** |
| --- | --- |
| 1 | 一般命令 |
| 2 | 系统调用 |
| 3 | 库函数，涵盖了C标准函数库 |
| 4 | 特殊文件（通常是/dev中的设备）和驱动程序 |
| 5 | 文件格式和约定 |
| 6 | 游戏和屏保 |
| 7 | 杂项 |
| 8 | 系统管理命令和守护进程 |

要查看相应区段的内容，就在 man 后面加上相应区段的数字即可，如：

$ man 1 ls

会显示第一区段中的ls命令 man 页面。

所有的手册页遵循一个常见的布局，为了通过简单的 ASCII 文本展示而被优化，而这种情况下可能没有任何形式的高亮或字体控制。一般包括以下部分内容：

**NAME（名称）**

该命令或函数的名称，接着是一行简介。

**SYNOPSIS（概要）**

对于命令，正式的描述它如何运行，以及需要什么样的命令行参数。对于函数，介绍函数所需的参数，以及哪个头文件包含该函数的定义。

**DESCRIPTION（说明）**

命令或函数功能的文本描述。

**EXAMPLES（示例）**

常用的一些示例。

**SEE ALSO（参见）**

相关命令或函数的列表。

也可能存在其它部分内容，但这些部分没有得到跨手册页的标准化。常见的例子包括：OPTIONS（选项），EXIT STATUS（退出状态），ENVIRONMENT（环境），BUGS（程序漏洞），FILES（文件），AUTHOR（作者），REPORTING BUGS（已知漏洞），HISTORY（历史）和 COPYRIGHT（版权）。

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

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

$ ls --help

## 作业：

本课程会在每一节实验结束部分介绍一个有趣的 Linux 命令，这一节介绍一个可以输出图形字符的命令banner，先看效果：

你可以先使用如下命令安装：

$ sudo apt-get update

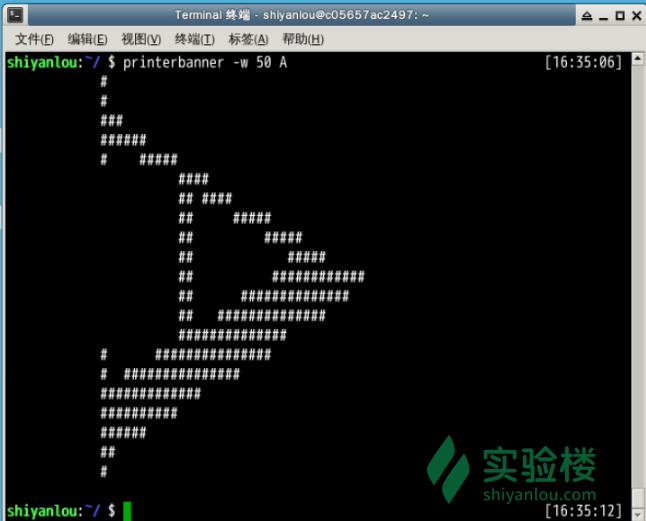
$ sudo apt-get install sysvbanner

然后：

$ banner shiyanlou

或者你觉得这字体不好看，那么你还可以使用默认已经安装的一个命令printerbanner：

$ printerbanner -w 50 A



-w参数指定打印宽度，因为我们的环境在屏幕中显示比较小，必须要加上宽度限制。

还有两个类似的命令toilet,figlet，作为作业安装试用。