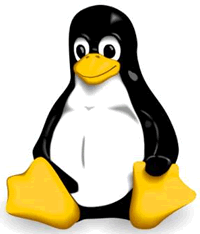
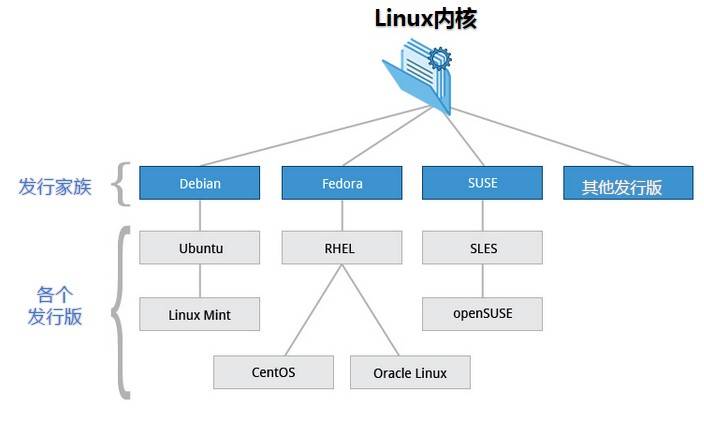
**Linux C与C++一线开发实践** 

# 一. Linux 简介



## 1. Linux和UNIX的关系及区别

Linux 是一套开放源代码程序的、可以自由传播的类 Unix 操作系统软件。但本质其实是Linux内核。

**UNIX**：60 年代末 ——> **Windows**：80 年代中期 ——> **Linux**：90 年代初。

**UNIX 系统**与**C语言**是一对孪生兄弟。

Linux 内核最初是由李纳斯•托瓦兹（Linus Torvalds）（他还搞出了全世界最大的程序员交友社区 GitHub）在赫尔辛基大学读书时出于个人爱好而编写的。

## 2. UNIX/Linux系统结构

|  |  |  |
| --- | --- | --- |
| 1.内核层 | 它直接附着在硬件平台之上，控制和管理系统内各种资源（硬件资源和软件资源） | http://c.biancheng.net/uploads/allimg/180926/2-1P926160U0153.jpg |
| 2.Shell层 | 输入命令行，由 Shell（命令解释器）解释执行并输出相应结果或者有关信息 |
| 3.应用层 | 提供基于 X Window 协议的图形环境。 |

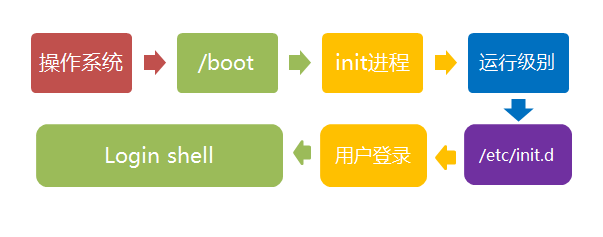
# 二. Linux 安装

centos 下载地址：

可以去官网下载最新版本：[https://www.centos.org/download/](https://www.centos.org/download/" \t "_blank)

安装教程： [Linux 安装 | 菜鸟教程 (runoob.com)](https://www.runoob.com/linux/linux-install.html) https://www.runoob.com/linux/linux-install.html

# 三. Linux 系统启动过程



## 1.内核的引导

首先读入 /boot 目录下的内核文件。

## 2.运行 init

init 程序首先是需要读取配置文件 /etc/inittab。

Linux允许为不同的场合，分配不同的开机启动程序，这就叫做"**运行级别**"（runlevel）。也就是说，启动时根据"运行级别"，确定要运行哪些程序。开机启动项在Windows叫做"服务"（service），在Linux就叫做"守护进程"（daemon）。

|  |  |
| --- | --- |
| 运行级别0 | 系统**停机**状态。系统默认运行级别不能设为0，否则不能正常启动。 |
| 运行级别1 | **单用户**工作状态，**root**权限，用于系统维护，禁止远程登陆。 |
| 运行级别2 | 多用户状态(没有NFS)。 |
| 运行级别3 | 完全的多用户状态(有NFS)，登陆后进入控制台命令行模式。 |
| 运行级别4 | 系统未使用，保留。 |
| 运行级别5 | X11控制台，登陆后进入图形GUI模式。 |
| 运行级别6 | 系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动。 |

## 3.系统初始化

在init的配置文件中有这么一行系统初始化操作：

si::sysinit:/etc/rc.d/rc.sysinit （bash shell的脚本）

它主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。

l5:5:wait:/etc/rc.d/rc 5

这一行表示以5为参数运行/etc/rc.d/rc，/etc/rc.d/rc是一个Shell脚本，它接受5作为参数，去执行/etc/rc.d/rc5.d/目录下的所有的rc启动脚本，/etc/rc.d/rc5.d/目录中的这些启动脚本实际上都是一些连接文件，而不是真正的rc启动脚本，真正的rc启动脚本实际上都是放在/etc/rc.d/init.d/目录下。

/etc/rc.d/rc5.d/中的rc启动脚本通常是K或S开头的连接文件，对于以 **S 开头**的启动脚本，将以**start**参数来运行。而如果发现存在相应的脚本也存在**K打头**的连接，而且已经处于运行态了(以/var/lock/subsys/下的文件作为标志)，则将首先以**stop**为参数停止这些已经启动了的守护进程，然后再重新运行。

（**这样做是为了保证是当init改变运行级别时，所有相关的守护进程都将重启。**）

至于在每个运行级中将运行哪些守护进程，用户可以通过chkconfig或setup中的"System Services"来自行设定。

## 4.建立终端

rc执行完毕后，返回init。这时基本系统环境已经设置好了，各种守护进程也已经启动了。

init接下来会打开6个终端，以便用户登录系统。在inittab中的以下6行就是定义了6个终端：

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

4:2345:respawn:/sbin/mingetty tty4

5:2345:respawn:/sbin/mingetty tty5

6:2345:respawn:/sbin/mingetty tty6

**2、3、4、5**的**运行级别**中都将以respawn方式运行mingetty程序，mingetty程序能打开终端、设置模式。

## 5.用户登录系统

|  |
| --- |
| 命令行登录 |
| ssh登录 |
| 图形界面登录 |

**注意事项：**

|  |  |  |
| --- | --- | --- |
| **运行级别**为**5**： | 他们的登录是通过一个图形化的登录界面。登录成功后可以直接进入 KDE、Gnome 等窗口管理器。 |  |
| 账号验证程序是 login： | 对用户名进行分析，如果用户名不是 root，且存在 /etc/nologin 文件，login 将输出 nologin 文件的内容，然后退出。 |  |
| /etc/securetty： | 登记了的终端才允许 root 用户登录。 | 如果不存在这个文件，则 root 用户可以在任何终端上登录。 |
| /etc/usertty： | 对用户作出附加访问限制。 | 如果不存在这个文件，则没有其他限制。 |

## 6.图形模式与文字模式的切换方式

默认我们登录的就是第一个窗口（tty1）。

Linux预设提供了六个命令窗口终端机（tty1,tty2 … tty6）让我们来登录。

按下**Ctrl + Alt + F1 ~ F6** 来切换它们。

界面切换快捷键：

|  |  |
| --- | --- |
| 图形界面 ——> 命令窗口界面 ： | **Ctrl + Alt + F1 ~ F6** |
| 命令窗口界面 ——> 图形界面： | **Ctrl + Alt + F7** |

## 7.Linux 关机

服务器上，很少遇到关机的操作。

正确的关机流程为：sync > shutdown > reboot > halt

sync 将数据由内存同步到硬盘中。

shutdown 关机指令，你可以man shutdown 来看一下帮助文档。例如你可以运行如下命令关机：

shutdown –h 10 ‘This server will shutdown after 10 mins’ 这个命令告诉大家，计算机将在10分钟后关机，并且会显示在登陆用户的当前屏幕中。

shutdown –h now 立马关机

shutdown –h 20:25 系统会在今天20:25关机

shutdown –h +10 十分钟后关机

shutdown –r now 系统立马重启

shutdown –r +10 系统十分钟后重启

reboot 就是重启，等同于 shutdown –r now

halt 关闭系统，等同于shutdown –h now 和 poweroff

要取消即将进行的关机：

# shutdown –c //cancel

halt 命令通知硬件来停止所有的 CPU 功能，但是仍然保持通电。你可以用它使系统处于低层维护状态。注意在有些情况会它会完全关闭系统。

# halt ### 停止机器

# halt -p ### 关闭机器、关闭电源

# halt --reboot ### 重启机器

poweroff 会发送一个 ACPI 信号来通知系统关机。

# poweroff ### 关闭机器、关闭电源

# poweroff --halt ### 停止机器

# poweroff --reboot ### 重启机器

reboot 命令 reboot 通知系统重启。

# reboot ### 重启机器

# reboot --halt ### 停止机器

# reboot -p ### 关闭机器

最后总结一下，不管是重启系统还是关闭系统，首先要运行 sync 命令，把内存中的数据写到磁盘中。

关机的命令有 shutdown –h now halt poweroff 和 init 0 , 重启系统的命令有 shutdown –r now reboot init 6。

# 四.系统目录结构

https://www.runoob.com/wp-content/uploads/2014/06/4_20.png

目录的解释：

|  |  |
| --- | --- |
| **/bin** | Binaries (二进制文件) 的缩写, 这个目录存放着最经常使用的命令。 |
| **/boot** | 启动 Linux 时使用的一些核心文件，包括一些连接文件以及镜像文件。 |
| **/dev** | Device(设备) 的缩写, 该目录下存放的是 Linux 的外部设备，在 Linux 中访问设备的方式和访问文件的方式是相同的。 |
| **/etc** | Etcetera(等等) 的缩写,这个目录用来存放所有的系统管理所需要的配置文件和子目录。 |
| **/home** | 用户的主目录，在 Linux 中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。 |
| **/lib** | Library(库) 的缩写这个目录里存放着系统最基本的动态连接共享库，其作用类似于 Windows 里的 DLL 文件。 |
| **/lost+found** | 这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。 |
| **/media** | 识别一些设备，例如U盘、光驱等等，当识别后，Linux 会把识别的设备挂载到这个目录下。 |
| **/mnt** | 临时挂载别的文件系统的，我们可以将光驱挂载在 /mnt/ 上，然后进入该目录就可以查看光驱里的内容了。 |
| **/opt** | optional(可选) 的缩写，这是给主机额外安装软件所摆放的目录。 |
| **/proc** | Processes(进程) 的缩写，伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，这个目录是一个虚拟的目录（在内存里），它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。 |
| **/root** | 管理员，用户主目录。 |
| **/sbin** | Superuser Binaries (超级用户的二进制文件) 的缩写，这里存放的是系统管理员使用的系统管理程序。 |
| **/selinux** | Redhat/CentOS 所特有的目录，类似于 windows 的防火墙。 |
| **/srv** | 存放一些服务启动之后需要提取的数据。 |
| **/sys** | 该目录下安装了 2.6 内核中新出现的一个文件系统 sysfs（针对进程信息的 proc 文件系统、针对设备的 devfs 文件系统以及针对伪终端的 devpts 文件系统） 。 |
| **/tmp** | temporary(临时) 的缩写这个目录是用来存放一些临时文件的。 |
| **/usr** | unix shared resources(共享资源) 的缩写。类似于 windows 下的 program files 目录。 |
| **/usr/bin** | 系统用户使用的应用程序。 |
| **/usr/sbin** | 超级用户使用的比较高级的管理程序和系统守护程序。 |
| **/usr/src** | 内核源代码默认的放置目录。 |
| **/var** | variable(变量) 的缩写，这个目录中存放着在不断扩充着的东西，我们习惯将那些经常被修改的目录放在这个目录下。包括各种**日志文件**。 |
| **/run** | 是一个临时文件系统，存储系统启动以来的信息。当系统重启时，这个目录下的文件应该被删掉或清除。 |

# 五. Linux 忘记密码解决方法

[Linux 忘记密码解决方法 | 菜鸟教程 (runoob.com)](https://www.runoob.com/linux/linux-forget-password.html)

https://www.runoob.com/linux/linux-forget-password.html

# 六. Linux 远程登录

Window 系统上 Linux 远程登录客户端有 SecureCRT, Putty, SSH Secure Shell ，XShell等。

通过ssh服务（建立在应用层和传输层基础上的安全协议）实现的远程登录功能，

默认 **ssh 服务端**口号为 **22**。

[Linux 远程登录 | 菜鸟教程 (runoob.com)](https://www.runoob.com/linux/linux-remote-login.html) https://www.runoob.com/linux/linux-remote-login.html

**终端利用ssh登录远程服务器：**

安装ssh：

yum install ssh

启动ssh：

service sshd start

登录远程服务器：

ssh -p 50022 my@127.0.0.1

输入密码：

my@127.0.0.1:

**-p** 后面是端口

**my** 是服务器用户名

**127.0.0.1** 是服务器 ip

回车输入密码即可登录。

# 七．Linux 文件基本属性

为了保护系统的安全性，Linux 系统对不同的用户访问同一文件（包括目录文件）的权限做了不同的规定。

通常使用以下两个命令来修改文件或目录的所属用户与权限：

|  |  |  |
| --- | --- | --- |
| **chown** (change owner) | 修改所属用户与组。 | https://www.runoob.com/wp-content/uploads/2014/06/1_151733904241.png |
| **chmod** (change mode) | 修改用户的权限。 |

|  |  |
| --- | --- |
| ll 或者 ls -l | 显示一个文件的**属性**以及文件所属的**用户和组**。 |

[root@www /]# ls -l

total 64

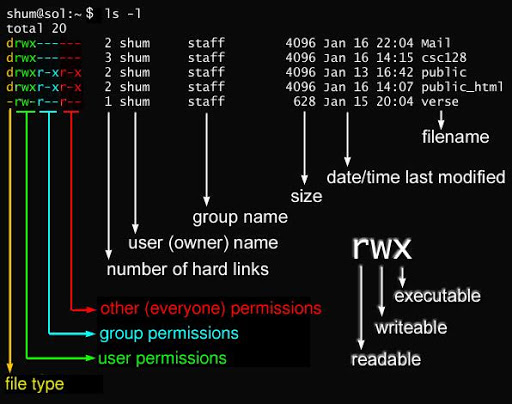
dr-xr-xr-x 2 root root 4096 Dec 14 2012 bin

dr-xr-xr-x 4 root root 4096 Apr 19 2012 boot

……

文件的第一个属性：

|  |  |
| --- | --- |
| **d** | 目录 |
| **-** | 文件 |
| **l** | 链接文档（link file） |
| **b** | 可供储存的接口设备(可随机存取装置) |
| **c** | 串行端口设备，例如键盘、鼠标(一次性读取装置) |



接下来的字符中，以三个为一组，且均为 **rwx** 的三个参数的组合。

|  |  |
| --- | --- |
| **r** | 可读(read) |
| **w** | 可写(write) |
| **x** | 可执行(execute) |
| **-** | 没有权限 |

## 1.更改文件属性

|  |  |
| --- | --- |
| chgrp [-R] 属组名 文件名 | 加上-R的参数，那么该目录下的所有文件的属组都会更改。 |
| chown [–R] 属主名 文件名  chown [-R] 属主名：属组名 文件名 |

例子：

## 2.chgrp：更改文件属组

chgrp [-R] 属组名 文件名

* -R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上-R的参数，那么该目录下的所有文件的属组都会更改。

## 3.chown：更改文件属主，也可以同时更改文件属组

将install.log文件的拥有者改为bin这个账号：

[root@www ~]# chown bin install.log

[root@www ~]# ls -l

-rw-r--r-- 1 bin users 68495 Jun 25 08:53 install.log

将install.log文件的拥有者与群组改回为root：

[root@www ~]# chown root:root install.log

[root@www ~]# ls -l

-rw-r--r-- 1 root root 68495 Jun 25 08:53 install.log

**chmod：更改文件9个属性**

|  |  |  |
| --- | --- | --- |
| **owner** | **group** | **others** |
| read(4)/write(2)/execute(1) | read(4)/write(2)/execute(1) | read(4)/write(2)/execute(1) |

**rwx rwx ---** 的每种身份(owner/group/others)各自的三个权限(r/w/x)**分数**:

owner = 4+2+1 = 7

group = 4+2+1 = 7

others = 0+0+0 = 0

所以，该文件的权限数字就是 **770 。**

## 4.变更权限的指令 chmod

chmod [-R] xyz 文件或目录

* **xyz** : 就是刚刚提到的数字类型的权限属性，为 **rwx** 属性数值的相加。
* **-R** : 进行递归(recursive)的持续变更，以及连同次目录下的所有文件都会变更

举例，将 **.bashrc** 这个文件所有的权限都设定启用：

[root@www ~]# chmod 777 .bashrc

[root@www ~]# ls -al .bashrc

-rwxrwxrwx 1 root root 395 Jul 4 11:45 .bashrc

再者， rwxr-xr-- 的分数：（4+2+1）（4+0+1）（4+0+0）=7 5 4

## 5.符号类型改变文件权限

还有一个改变权限的方法：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 将文件权限设置为rwxr-xr-- ： **chmod u=rwx,g=rx,o=r 文件名** | | | | |
| chmod | u(user) g(group) o(others) a(all) | +(加入) -(除去) =(设定) | r w x | 文件或目录 |

# chmod u=rwx,g=rx,o=r test1 // 修改 test1 权限

# ls -al test1

-rwxr-xr-- 1 root root 0 Nov 15 10:32 test1

将权限去掉而不改变其他已存在的权限（a-x）：

# chmod a-x test1

# ls -al test1

-rw-r--r-- 1 root root 0 Nov 15 10:32 test1

# 八. Linux 系统信息、系统资源、时间

## 1.系统信息

**系统标识 uname**

struct utsname {

char sysname[]; /\* 当前操作系统的名称 \*/

char nodename[]; /\* 网络上的名称（主机名） \*/

char release[]; /\* 操作系统内核版本 \*/

char version[]; /\* 操作系统发行版本 \*/

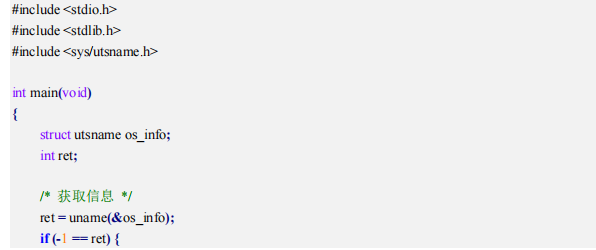
char machine[]; /\* 硬件架构类型 \*/

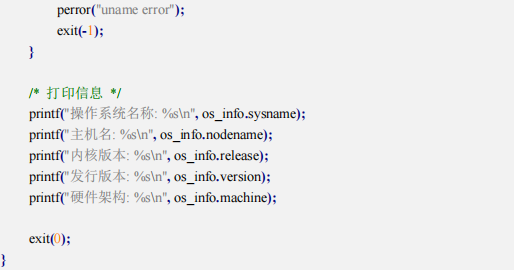
#ifdef \_GNU\_SOURCE

char domainname[];/\* 当前域名 \*/

#endif

};





**sysinfo 函数**

struct sysinfo {

long uptime; /\* 自系统启动之后所经过的时间（以秒为单位） \*/

unsigned long loads[3]; /\* 1, 5, and 15 minute load averages \*/

unsigned long totalram; /\* 总的可用内存大小 \*/

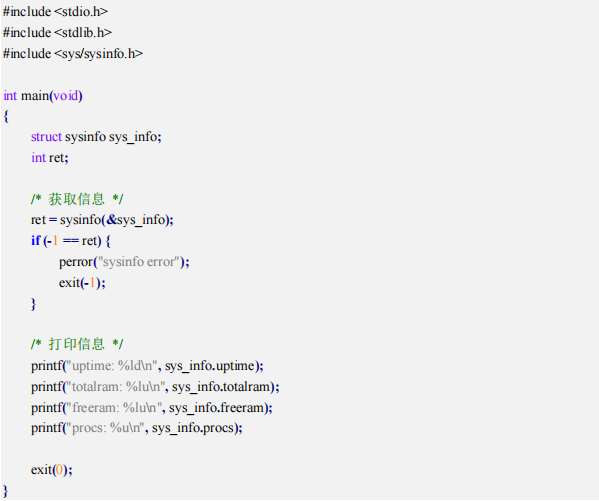
unsigned long freeram; /\* 还未被使用的内存大小 \*/

unsigned long sharedram; /\* Amount of shared memory \*/

unsigned long bufferram; /\* Memory used by buffers \*/

unsigned long totalswap; /\* Total swap space size \*/

unsigned long freeswap; /\* swap space still available \*/

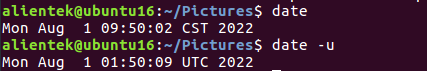


**gethostname 函数**

**sysconf()函数**

## 2.时间

系统时区、UTC时间：



# 点亮LED灯

## 1.应用层控制硬件

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#define LED\_TRIGGER "/sys/class/leds/sys-led/trigger"

#define LED\_BRIGHTNESS "/sys/class/leds/sys-led/brightness"

#define USAGE() fprintf(stderr, "usage:\n" \

" %s <on|off>\n" \

" %s <trigger> <type>\n", \

argv[0], argv[0])

int main(int argc, char \*argv[])

{

int fd1, fd2;

/\* 校验传参 \*/

if (2 > argc)

{

USAGE();

exit(-1);

}

/\* 打开文件 \*/

fd1 = open(LED\_TRIGGER, O\_RDWR);

if (0 > fd1)

{

perror("open error");

exit(-1);

}

fd2 = open(LED\_BRIGHTNESS, O\_RDWR);

if (0 > fd2)

{

perror("open error");

exit(-1);

}

/\* 根据传参控制 LED \*/

if (!strcmp(argv[1], "on"))

{

write(fd1, "none", 4); //先将触发模式设置为 none

write(fd2, "1", 1); //点亮 LED

}

else if (!strcmp(argv[1], "off"))

{

write(fd1, "none", 4); //先将触发模式设置为 none

write(fd2, "0", 1); //LED 灭

}

else if (!strcmp(argv[1], "trigger"))

{

if (3 != argc)

{

USAGE();

exit(-1);

}

if (0 > write(fd1, argv[2], strlen(argv[2])))

perror("write error");

}

else

USAGE();

exit(0);

}

## 2.应用层控制GPIO