Linux

什么是 Linux

Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX和 Unix 的多用户、多任务、支持多线程和多 CPU的操作系统。它能运行主要的 Unix 工具软件、应用程序和网络协议。它支持32 位和64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

Unix 和 Linux 有什么区别?

Linux 和 Unix 都是功能强大的操作系统，都是应用广泛的服务器操作系统，有很多相似之处，甚至有一部分人错误地认为 Unix 和 Linux 操作系统是一样的，然而，事实并非如此，以下是两者的区别。

- 开源性：Linux 是一款开源操作系统，不需要付费，即可使用；Unix 是一款对源码实行知识产权保护的传统商业软件，使用需要付费授权使用。
- 跨平台性：Linux 操作系统具有良好的跨平台性能，可运行在多种硬件平台上；Unix 操作系统跨平台性能较弱，大多需与硬件配套使用。
- 可视化界面：Linux 除了进行命令行操作，还有窗体管理系统；Unix 只是命令行下的系统。
- 硬件环境：Linux 操作系统对硬件的要求较低，安装方法更易掌握；Unix 对硬件要求比较苛刻，按照难度较大。
- 用户群体：Linux 的用户群体很广泛，个人和企业均可使用；Unix 的用户群体比较窄，多是安全性要求高的大型企业使用，如银行、电信部门等，或者 Unix 硬件厂商使用，如 Sun等。相比于 Unix 操作系统，Linux 操作系统更受广大计算机爱好者的喜爱，主要原因是 Linux 操作系统具有 Unix 操作系统的全部功能，并且能够在普通 PC计算机上实现全部的 Unix 特性，开源免费的特性，更容易普及使用！

什么是 Linux 内核?

Linux 系统的核心是内核。内核控制着计算机系统上的所有硬件和软件，在必要时分配硬件，并根据需要执行软件。

- 系统内存管理
- 应用程序管理
- 硬件设备管理
- 文件系统管理

Linux 的基本组件是什么?

就像任何其他典型的操作系统一样，Linux 拥有所有这些组件：内核，shell 和 GUI，系统实用程序和应用程序。Linux 比其他操作系统更具优势的是每个方面都附带其他功能，所有代码都可以免费下载。

Linux 的体系结构

从大的方面讲，Linux 体系结构可以分为两块：

- 用户空间(User Space)：用户空间又包括用户的应用程序(User Applications)、C 库(C Library)。
- 内核空间(Kernel Space)：内核空间又包括系统调用接口(System Call Interface)、内核(Kernel)、平台架构相关的代码(Architecture - Dependent Kernel Code)。

为什么 Linux 体系结构要分为用户空间和内核空间的原因?

- 现代 CPU 实现了不同的工作模式，不同模式下 CPU 可以执行的指令和访问的寄存器不同。
- Linux 从 CPU 的角度出发，为了保护内核的安全，把系统分成了两部分。用户空间和内核空间是程序执行的两种不同的状态，我们可以通过两种方式完成用户空间到内核空间的转移：1）系统调用；2）硬件中断。

BASH 和 DOS 之间的基本区别是什么?

BASH和 DOS控制台之间的主要区别在于3 个方面：

- BASH命令区分大小写，而 DOS命令则不区分;
- 在 BASH下，/ character 是目录分隔符，\作为转义字符。在 DOS下，/用作命令参数分隔符，\是目录分隔符
- DOS遵循命名文件中的约定，即8 个字符的文件名后跟一个点，扩展名为3 个字符。BASH没有遵循这样的惯例。

Linux 开机启动过程?

了解即可

- 主机加电自检，加载 BIOS 硬件信息
- 读取 MBR 的引导文件(GRUB、LILO)
- 引导 Linux 内核
- 运行第一个进程 init (进程号永远为1)
- 进入相应的运行级别
- 运行终端，输入用户名和密码

Linux 系统缺省的运行级别?

- 关机
- 单机用户模式
- 字符界面的多用户模式(不支持网络)
- 字符界面的多用户模式
- 未分配使用
- 图形界面的多用户模式
- 重启

Linux 使用的进程间通信方式?

- 管道(pipe)、流管道(s_pipe)、有名管道(FIFO)
- 信号(signal)
- 消息队列
- 共享内存

- 信号量
- 套接字(socket)

Linux 有哪些系统日志文件?

比较重要的是/var/log/messages 日志文件。

该日志文件是许多进程日志文件的汇总，从该文件可以看出任何入侵企图或成功的入侵。另外，如果胖友的系统里有 ELK 日志集中收集，它也会被收集进去。

Linux 系统安装多个桌面环境有帮助吗?

通常，一个桌面环境，如 KDE或 Gnome，足以在没有问题的情况下运行。尽管系统允许从一个环境切换到另一个环境，但这对用户来说都是优先考虑的问题。有些程序在一个环境中工作而在另一个环境中无法工作，因此它也可以被视为选择使用哪个环境的一个因素。

什么是交换空间?

交换空间是 Linux 使用的一定空间，用于临时保存一些并发运行的程序。当 RAM没有足够的内存来容纳正在执行的所有程序时，就会发生这种情况。

什么是 Root 帐户

root 帐户就像一个系统管理员帐户，允许你完全控制系统。你可以在此处创建和维护用户帐户，为每个帐户分配不同的权限。每次安装 Linux 时都是默认帐户。

什么是 LILO?

LILO是 Linux 的引导加载程序。它主要用于将 Linux 操作系统加载到主内存中，以便它可以开始运行。

什么是 BASH?

BASH是 Bourne Again SHell 的缩写。它由 Steve Bourne 编写，作为原始 Bourne Shell（由/ bin / sh 表示）的替代品。它结合了原始版本的 Bourne Shell 的所有功能，以及其他功能，使其更容易使用。从那以后，它已被改编为运行 Linux 的大多数系统的默认 shell。

什么是 CLI?

命令行界面（英语：command-line interface，缩写]：CLI）是在图形用户界面得到普及之前使用最为广泛的用户界面，它通常不支持鼠标，用户通过键盘输入指令，计算机接收到指令后，予以执行。也有人称之为字符用户界面

- CUI）。

通常认为，命令行界面（CLI）没有图形用户界面（GUI）那么方便用户操作。因为，命令行界面的软件通常需要用户记忆操作的命令，但是，由于其本身的特点，命令行界面要较图形用户界面节约计算机系统的资源。在熟记命令的前提下，使用命令行界面往往要较使用图形用户界面的操作速度要快。所以，图形用户界面的操作系统中，都保留着可选的命令行界面。

什么是 GUI?

图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口）是指采用图形方式显示的计算机操作用户界面。图形用户界面是一种人与计算机通信的界面显示格式，允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，以选择命令、调用文件、启动程序或执行其它一些日常任务。与通过键盘输入文本或字符命令来完成例行任务的字符界面相比，图形用户界面有许多优点。

开源的优势是什么?

开源允许你将软件（包括源代码）免费分发给任何感兴趣的人。然后，人们可以添加功能，甚至可以调试和更正源代码中的错误。它们甚至可以让它运行得

更好，然后再次自由地重新分配这些增强的源代码。这最终使社区中的每个人受益。

GNU 项目的重要性是什么?

这种所谓的自由软件运动具有多种优势，例如可以自由地运行程序以及根据你的需要自由学习和修改程序。它还允许你将软件副本重新分发给其他人，以及自由改进软件并将其发布给公众。

绝对路径用什么符号表示？当前目录、上层目录用什么表示？主目录用什么表示? 切换目录用什么命令?

绝对路径：如/etc/init.d
当前目录和上层目录：./ ../
主目录：~/
切换目录：cd

怎么查看当前进程？怎么执行退出？怎么查看当前路径?

查看当前进程：ps
执行退出：exit
查看当前路径：pwd

怎么清屏？怎么退出当前命令？怎么执行睡眠？怎么查看当前用户 id？查看指定帮助用什么命令?

清屏：clear
退出当前命令：ctrl+c 彻底退出
执行睡眠 ： ctrl+z 挂起当前进程fg 恢复后台
查看当前用户 id：”id“：查看显示目前登陆账户的 uid 和 gid 及所属分组及用户名
查看指定帮助：如 man adduser 这个很全 而且有例子；adduser --help 这个告诉你一些常用参数；info adduesr；

Ls 命令执行什么功能？可以带哪些参数，有什么区别?

ls 执行的功能：列出指定目录中的目录，以及文件
哪些参数以及区别：a 所有文件l 详细信息，包括大小字节数，可读可写可执行的权限等

建立软链接(快捷方式)，以及硬链接的命令。

软链接：ln -s slink source
硬链接：ln link source

目录创建用什么命令？创建文件用什么命令？复制文件用什么命令?

创建目录：mkdir
创建文件：典型的如 touch，vi 也可以创建文件，其实只要向一个不存在的文件输出，都会创建文件
复制文件：cp 7. 文件权限修改用什么命令？格式是怎么样的?
文件权限修改：chmod
格式如下：

$ chmod u+x file 给 file 的属主增加执行权限
$ chmod 751 file 给 file 的属主分配读、写、执行(7)的权限，给 file 的所在组分配读、执行(5)的权限，给其他用户分配执行(1)的权限
$ chmod u=rwx,g=rx,o=x file 上例的另一种形式
$ chmod =r file 为所有用户分配读权限
$ chmod 444 file 同上例
$ chmod a-wx,a+r file同上例
$ chmod -R u+r directory 递归地给 directory 目录下所有文件和子目录的属主分配读的权限

查看文件内容有哪些命令可以使用?

vi 文件名 #编辑方式查看，可修改
cat 文件名 #显示全部文件内容
more 文件名 #分页显示文件内容
less 文件名 #与 more 相似，更好的是可以往前翻页
tail 文件名 #仅查看尾部，还可以指定行数
head 文件名 #仅查看头部,还可以指定行数

随意写文件命令？怎么向屏幕输出带空格的字符串，比如”hello world”?

写文件命令：vi

向屏幕输出带空格的字符串:echo hello world

终端是哪个文件夹下的哪个文件？黑洞文件是哪个文件夹下的哪个命令?

终端 /dev/tty

黑洞文件 /dev/null

移动文件用哪个命令？改名用哪个命令?

mv mv

复制文件用哪个命令？如果需要连同文件夹一块复制呢？如果需要有提示功能呢?

cp cp -r？？？？

删除文件用哪个命令？如果需要连目录及目录下文件一块删除呢？删除空文件夹用什么命令?

rm rm -r rmdir

Linux 下命令有哪几种可使用的通配符？分别代表什么含义?

“？”可替代单个字符。

“*”可替代任意多个字符。

方括号“[charset]”可替代 charset 集中的任何单个字符，如[a-z]，[abABC]

用什么命令对一个文件的内容进行统计？(行号、单词数、字节数)

wc 命令 - c 统计字节数 - l 统计行数 - w 统计字数。

Grep 命令有什么用？如何忽略大小写？如何查找不含该串的行?

是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹 配的行打印出来。
grep [stringSTRING] filename grep [^string] filename

Linux 中进程有哪几种状态？在 ps 显示出来的信息中，分别用什么符号表示的?

（1）、不可中断状态：进程处于睡眠状态，但是此刻进程是不可中断的。不可中断， 指进程不响应异步信号。
（2）、暂停状态/跟踪状态：向进程发送一个 SIGSTOP 信号，它就会因响应该信号 而进入 TASK_STOPPED 状态;当进程正在被跟踪时，它处于 TASK_TRACED 这个特殊的状态。
“正在被跟踪”指的是进程暂停下来，等待跟踪它的进程对它进行操作。

（3）、就绪状态：在 run_queue 队列里的状态

（4）、运行状态：在 run_queue 队列里的状态
（5）、可中断睡眠状态：处于这个状态的进程因为等待某某事件的发生（比如等待 socket 连接、等待信号量），而被挂起
（6）、zombie 状态（僵尸）：父亲没有通过 wait 系列的系统调用会顺便将子进程的尸体（task_struct）也释放掉
（7）、退出状态

D 不可中断 Uninterruptible（usually IO）
R 正在运行，或在队列中的进程
S 处于休眠状态
T 停止或被追踪
Z 僵尸进程
W 进入内存交换（从内核 2.6 开始无效）
X 死掉的进程

怎么使一个命令在后台运行?

一般都是使用 & 在命令结尾来让程序自动运行。(命令后可以不追加空格)

利用 ps 怎么显示所有的进程? 怎么利用 ps 查看指定进程的信息?

ps -ef (system v 输出)

ps -aux bsd 格式输出

ps -ef | grep pid

哪个命令专门用来查看后台任务?

job -l

把后台任务调到前台执行使用什么命令?把停下的后台任务在后台执行起来用什么命令?

把后台任务调到前台执行 fg

把停下的后台任务在后台执行起来 bg

终止进程用什么命令? 带什么参数?

kill [-s <信息名称或编号>][程序] 或 kill [-l <信息编号>]

kill-9 pid

怎么查看系统支持的所有信号?

kill -l

搜索文件用什么命令? 格式是怎么样的?

find <指定目录> <指定条件> <指定动作>

whereis 加参数与文件名

locate 只加文件名

find 直接搜索磁盘，较慢。

find / -name "string*"

查看当前谁在使用该主机用什么命令? 查找自己所在的终端信息用什么命令?

查找自己所在的终端信息：who am i

查看当前谁在使用该主机：who

使用什么命令查看用过的命令列表?

history

使用什么命令查看磁盘使用空间? 空闲空间呢?

df -hl
文件系统 容量 已用 可用 已用% 挂载点
Filesystem Size Used Avail Use% Mounted on /dev/hda2 45G 19G 24G 44% /
/dev/hda1 494M 19M 450M 4% /boot

使用什么命令查看网络是否连通?

netstat

使用什么命令查看 ip 地址及接口信息?

ifconfig

查看各类环境变量用什么命令?

查看所有 env
查看某个，如 home：env $HOME

通过什么命令指定命令提示符?

\u：显示当前用户账号
\h：显示当前主机名
\W：只显示当前路径最后一个目录
\w：显示当前绝对路径（当前用户目录会以~代替）
$PWD：显示当前全路径
$：显示命令行'$'或者'#'符号
#：下达的第几个命令
\d：代表日期，格式为week day month date，例如："MonAug1"
\t：显示时间为24小时格式，如：HH：MM：SS
\T：显示时间为12小时格式
\A：显示时间为24小时格式：HH：MM
\v：BASH的版本信息 如export PS1='[\u@\h\w#]$'

查找命令的可执行文件是去哪查找的? 怎么对其进行设置及添加?

whereis [-bfmsu][-B <目录>...][-M <目录>...][-S <目录>...][文件...]

补充说明：whereis 指令会在特定目录中查找符合条件的文件。这些文件的烈性应属于原始代码，二进制文件，或是帮助文件。

-b 只查找二进制文件。
-B<目录> 只在设置的目录下查找二进制文件。-f 不显示文件名前的路径名称。
-m 只查找说明文件。
-M<目录> 只在设置的目录下查找说明文件。-s 只查找原始代码文件。
-S<目录> 只在设置的目录下查找原始代码文件。-u 查找不包含指定类型的文件。
which 指令会在 PATH 变量指定的路径中，搜索某个系统命令的位置，并且返回第一个搜索结果。
-n 指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。
-p 与-n 参数相同，但此处的包括了文件的路径。-w 指定输出时栏位的宽度。
-V 显示版本信息

通过什么命令查找执行命令?

which 只能查可执行文件

whereis 只能查二进制文件、说明文档，源文件等

怎么对命令进行取别名?

alias la='ls -a'

du 和 df 的定义，以及区别?

du 显示目录或文件的大小

df 显示每个<文件>所在的文件系统的信息，默认是显示所有文件系统。
（文件系统分配其中的一些磁盘块用来记录它自身的一些数据，如 i 节点，磁盘分布图，间接块，超级块等。这些数据对大多数用户级的程序来说是不可见的，通常称为 Meta Data。） du 命令是用户级的程序，它不考虑 Meta Data，而 df 命令则查看文件系统的磁盘分配图并考虑 Meta Data。
df 命令获得真正的文件系统数据，而 du 命令只查看文件系统的部分情况。

awk 详解。

awk '{pattern + action}' {filenames}
#cat /etc/passwd |awk -F ':' '{print $1"\t"$7}' //-F 的意思是以':'分隔 root /bin/bash
daemon /bin/sh 搜索/etc/passwd 有 root 关键字的所有行

#awk -F: '/root/' /etc/passwd root❌0:0:root:/root:/bin/bash

当你需要给命令绑定一个宏或者按键的时候，应该怎么做呢?

可以使用bind命令，bind可以很方便地在shell中实现宏或按键的绑定。

在进行按键绑定的时候，我们需要先获取到绑定按键对应的字符序列。

比如获取F12的字符序列获取方法如下：先按下Ctrl+V,然后按下F12 .我们就可以得到F12的字符序列 ^[[24~。

接着使用bind进行绑定。

[root@localhost ~]# bind '"\e[24~":"date"'

注意：相同的按键在不同的终端或终端模拟器下可能会产生不同的字符序列。

【附】也可以使用showkey -a命令查看按键对应的字符序列。

如果一个linux新手想要知道当前系统支持的所有命令的列表，他需要怎么做?

使用命令compgen -c，可以打印出所有支持的命令列表。

[root@localhost ~]$ compgen -c
l.
ll
ls
which
if
then
else
elif
fi
case

esac
for
select
while
until
do
done
…

如果你的助手想要打印出当前的目录栈，你会建议他怎么做?

使用Linux 命令dirs可以将当前的目录栈打印出来。

[root@localhost ~]# dirs
/usr/share/X11
【附】： 目录栈通过pushd popd 来操作。

你的系统目前有许多正在运行的任务，在不重启机器的条件下，有什么方法可以把所有正在运行的进程移除呢?

使用linux命令 'disown -r '可以将所有正在运行的进程移除。

bash shell 中的hash 命令有什么作用?

linux命令'hash'管理着一个内置的哈希表，记录了已执行过的命令的完整路径, 用该命令可以打印出你所使用过的命令以及执行的次数。

[root@localhost ~]# hash
hits command
2 /bin/ls
2 /bin/su

哪一个bash内置命令能够进行数学运算。

bash shell 的内置命令let 可以进行整型数的数学运算。

#! /bin/bash
…
…
let c=a+b
…
…

怎样一页一页地查看一个大文件的内容呢?

通过管道将命令”cat file_name.txt” 和 ’more’ 连接在一起可以实现这个需要.

[root@localhost ~]# cat file_name.txt | more

数据字典属于哪一个用户的?

数据字典是属于’SYS’用户的，用户‘SYS’ 和 ’SYSEM’是由系统默认自动创建的

怎样查看一个linux命令的概要与用法？假设你在/bin目录中偶然看到一个你从没见过的的命令，怎样才能知道它的作用和用法呢?

使用命令whatis 可以先出显示出这个命令的用法简要，比如，你可以使用whatis zcat 去查看‘zcat’的介绍以及使用简要。

[root@localhost ~]# whatis zcat
zcat [gzip] (1) – compress or expand files

使用哪一个命令可以查看自己文件系统的磁盘空间配额呢?

使用命令repquota 能够显示出一个文件系统的配额信息

【附】只有root用户才能够查看其它用户的配额。

说一下异步和非阻塞的区别?

- 异步和非阻塞的区别:

1. 异步：调用在发出之后，这个调用就直接返回，不管有无结果；异步是过程。
2. 非阻塞：关注的是程序在等待调用结果（消息，返回值）时的状态，指在不能立刻得到结果之前，该调用不会阻塞当前线程。

- 同步和异步的区别：

1. 步：一个服务的完成需要依赖其他服务时，只有等待被依赖的服务完成后，才算完成，这是一种可靠的服务序列。要么成功都成功，失败都失败，服务的状态可以保持一致。
2. 异步：一个服务的完成需要依赖其他服务时，只通知其他依赖服务开始执行，而不需要等待被依赖的服务完成，此时该服务就算完成了。被依赖的服务是否最终完成无法确定，一次它是一个不可靠的服务序列。

- 消息通知中的同步和异步：

1. 同步：当一个同步调用发出后，调用者要一直等待返回消息（或者调用结果）通知后，才能进行后续的执行。
2. 异步：当一个异步过程调用发出后，调用者不能立刻得到返回消息（结果）。在调用结束之后，通过消息回调来通知调用者是否调用成功。

- 阻塞与非阻塞的区别：

1. 阻塞：阻塞调用是指调用结果返回之前，当前线程会被挂起，一直处于等待消息通知，不能够执行其他业务，函数只有在得到结果之后才会返回。
2. 非阻塞：非阻塞和阻塞的概念相对应，指在不能立刻得到结果之前，该函数不会阻塞当前线程，而会立刻返回。

同步与异步是对应的，它们是线程之间的关系，两个线程之间要么是同步的，要么是异步的。

阻塞与非阻塞是对同一个线程来说的，在某个时刻，线程要么处于阻塞，要么处于非阻塞。

阻塞是使用同步机制的结果，非阻塞则是使用异步机制的结果。

滑动窗口的概念以及应用?

滑动窗口概念不仅存在于数据链路层，也存在于传输层，两者有不同的协议，但基本原理是相近的。其中一个重要区别是，一个是针对于帧的传送，另一个是字节数据的传送。

滑动窗口（Sliding window）是一种流量控制技术。早期的网络通信中，通信双方不会考虑网络的拥挤情况直接发送数据。由于大家不知道网络拥塞状况，同时发送数据，导致中间节点阻塞掉包，谁也发不了数据，所以就有了滑动窗口机制来解决此问题。参见滑动窗口如何根据网络拥塞发送数据仿真视频。

滑动窗口协议是用来改善吞吐量的一种技术，即容许发送方在接收任何应答之前传送附加的包。接收方告诉发送方在某一时刻能送多少包（称窗口尺寸）。

CP中采用滑动窗口来进行传输控制，滑动窗口的大小意味着接收方还有多大的缓冲区可以用于接收数据。发送方可以通过滑动窗口的大小来确定应该发送多少字节的数据。当滑动窗口为0时，发送方一般不能再发送数据报，但有两种情况除外，一种情况是可以发送紧急数据，例如，允许用户终止在远端机上的运行进程。另一种情况是发送方可以发送一个1字节的数据报来通知接收方重新声明它希望接收的下一字节及发送方的滑动窗口大小。

Epoll原理.

开发高性能网络程序时，windows开发者们言必称Iocp，linux开发者们则言必称Epoll。大家都明白Epoll是一种IO多路复用技术，可以非常高效的处理数以百万计的Socket句柄，比起以前的Select和Poll效率提高了很多。

先简单了解下如何使用C库封装的3个epoll系统调用。

```
int epoll_create(int size);
int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);
int epoll_wait(int epfd, struct epoll_event *events,int maxevents, int timeout);
```

使用起来很清晰，首先要调用`epoll_create`建立一个epoll对象。参数size是内核保证能够正确处理的最大句柄数，多于这个最大数时内核可不保证效果。 epoll_ctl可以操作上面建立的epoll，例如，将刚建立的`socket`加入到epoll中让其监控，或者把 epoll正在监控的某个socket句柄移出epoll，不再监控它等等。

`epoll_wait`在调用时，在给定的timeout时间内，当在监控的所有句柄中有事件发生时，就返回用户态的进程。

从调用方式就可以看到epoll相比select/poll的优越之处是,因为后者每次调用时都要传递你所要监控的所有socket给select/poll系统调用，这意味着需要将用户态的socket列表copy到内核态，如果以万计的句柄会导致每次都要copy几十几百KB的内存到内核态，非常低效。而我们调用`epoll_wait`时就相当于以往调用select/poll，但是这时却不用传递socket句柄给内核，因为内核已经在epoll_ctl中拿到了要监控的句柄列表。

所以，实际上在你调用`epoll_create`后，内核就已经在内核态开始准备帮你存储要监控的句柄了，每次调用`epoll_ctl`只是在往内核的数据结构里塞入新的socket句柄。

在内核里，一切皆文件。所以，epoll向内核注册了一个文件系统，用于存储上述的被监控socket。当你调用epoll_create时，就会在这个虚拟的epoll文件系统里创建一个file结点。当然这个file不是普通文件，它只服务于epoll。

epoll在被内核初始化时（操作系统启动），同时会开辟出epoll自己的内核高速cache区，用于安置每一个我们想监控的socket，这些socket会以红黑树的形式保存在内核cache里，以支持快速的查找、插入、删除。这个内核高速cache区，就是建立连续的物理内存页，然后在之上建立slab层，通常来讲，就是物理上分配好你想要的size的内存对象，每次使用时都是使用空闲的已分配好的对象。

```
static int __init eventpoll_init(void)  {
    ... ...

    /* Allocates slab cache used to allocate "struct epitem" items */
    epi_cache = kmem_cache_create("eventpoll_epi", sizeof(struct epitem),
            0, SLAB_HWCACHE_ALIGN|EPI_SLAB_DEBUG|SLAB_PANIC,
            NULL, NULL);

    /* Allocates slab cache used to allocate "struct eppoll_entry" */
    pwq_cache = kmem_cache_create("eventpoll_pwq",
            sizeof(struct eppoll_entry), 0,
            EPI_SLAB_DEBUG|SLAB_PANIC, NULL, NULL);
 ... ...
 }
```

epoll的高效就在于，当我们调用`epoll_ctl`往里塞入百万个句柄时，`epoll_wait`仍然可以飞快的返回，并有效的将发生事件的句柄给我们用户。这是由于我们在调用`epoll_create`时，内核除了帮我们在epoll文件系统里建了个file结点，在内核cache里建了个红黑树用于存储以后epoll_ctl传来的socket外，还会再建立一个list链表，用于存储准备就绪的事件，当epoll_wait调用时，仅仅观察这个list链表里有没有数据即可。有数据就返回，没有数据就sleep，等到timeout时间到后即使链表没数据也返回。所以，epoll_wait非常高效。

而且，通常情况下即使我们要监控百万计的句柄，大多一次也只返回很少量的准备就绪句柄而已，所以，epoll_wait仅需要从内核态copy少量的句柄到用户态而已，因此就会非常的高效！

然而,这个准备就绪list链表是怎么维护的呢？当我们执行epoll_ctl时，除了把socket放到epoll文件系统里file对象对应的红黑树上之外，还会给内核中断处理程序注册一个回调函数，告诉内核，如果这个句柄的中断到了，就把它放到准备就绪list链表里。所以，当一个socket上有数据到了，内核在把网卡上的数据copy到内核中后就来把socket插入到准备就绪链表里了。

如此，一个红黑树，一张准备就绪句柄链表，少量的内核cache，就帮我们解决了大并发下的socket处理问题。执行`epoll_create`时，创建了红黑树和就绪链表，执行epoll_ctl时，如果增加socket句柄，则检查在红黑树中是否存在，存在立即返回，不存在则添加到树干上，然后向内核注册回调函数，用于当中断事件来临时向准备就绪链表中插入数据。执行epoll_wait时立刻返回准备就绪链表里的数据即可。

最后看看epoll独有的两种模式LT和ET。无论是LT和ET模式，都适用于以上所说的流程。区别是，LT模式下，只要一个句柄上的事件一次没有处理完，会在以后调用epoll_wait时每次返回这个句柄，而ET模式仅在第一次返回。

当一个socket句柄上有事件时，内核会把该句柄插入上面所说的准备就绪list链表，这时我们调用`epoll_wait`，会把准备就绪的socket拷贝到用户态内存，然后清空准备就绪list链表，最后，`epoll_wait`需要做的事情，就是检查这些socket，如果不是ET模式（就是LT模式的句柄了），并且这些socket上确实有未处理的事件时，又把该句柄放回到刚刚清空的准备就绪链表了。所以，非ET的句柄，只要它上面还有事件，epoll_wait每次都会返回。而ET模式的句柄，除非有新中断到，即使socket上的事件没有处理完，也是不会每次从epoll_wait返回的。

因此epoll比select的提高实际上是一个用空间换时间思想的具体应用.对比阻塞IO的处理模型, 可以看到采用了多路复用IO之后, 程序可以自由的进行自己除了IO操作之外的工作, 只有到IO状态发生变化的时候由多路复用IO进行通知, 然后再采取相应的操作, 而不用一直阻塞等待IO状态发生变化,提高效率.

负载均衡原理是什么?

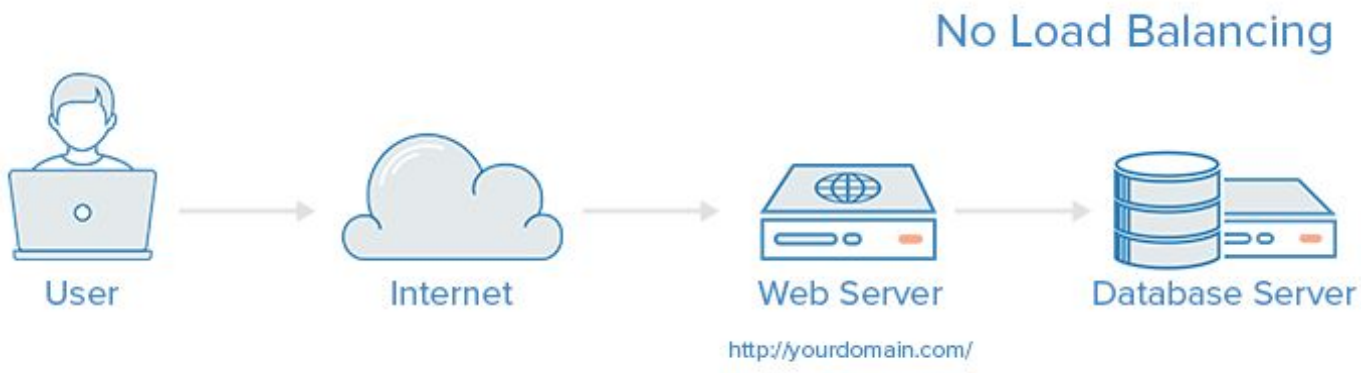
负载均衡Load Balance）是高可用网络基础架构的关键组件，通常用于将工作负载分布到多个服务器来提高网站、应用、数据库或其他服务的性能和可靠性。负载均衡，其核心就是网络流量分发，分很多维度。

负载均衡（Load Balance）通常是分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。

负载均衡是建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

通过一个例子详细介绍:

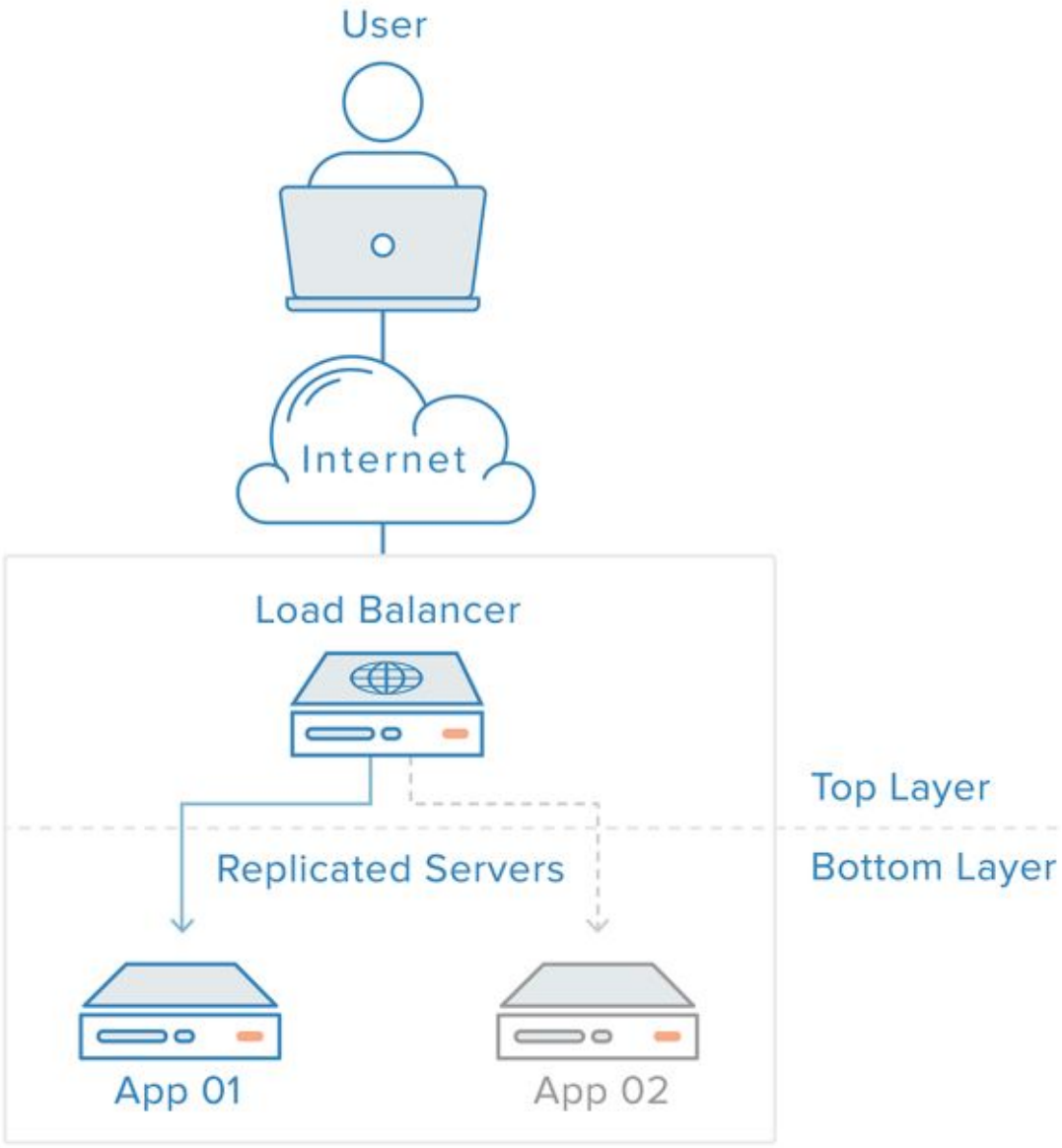
- 没有负载均衡 web 架构



在这里用户是直连到 web 服务器，如果这个服务器宕机了，那么用户自然也就没办法访问了。 另外，如果同时有很多用户试图访问服务器，超过了其能处理的极限，就会出现加载速度缓慢或根本无法连接的情况。

而通过在后端引入一个负载均衡器和至少一个额外的 web 服务器，可以缓解这个故障。 通常情况下，所有的后端服务器会保证提供相同的内容，以便用户无论哪个服务器响应，都能收到一致的内容。

- 有负载均衡 web 架构



用户访问负载均衡器，再由负载均衡器将请求转发给后端服务器。在这种情况下，单点故障现在转移到负载均衡器上了。 这里又可以通过引入第二个负载均衡器来缓解。

那么负载均衡器的工作方式是什么样的呢,负载均衡器又可以处理什么样的请求?

负载均衡器的管理员能主要为下面四种主要类型的请求设置转发规则：

- HTTP (七层)
- HTTPS (七层)
- TCP (四层)
- UDP (四层)

负载均衡器如何选择要转发的后端服务器?

负载均衡器一般根据两个因素来决定要将请求转发到哪个服务器。首先，确保所选择的服务器能够对请求做出响应，然后根据预先配置的规则从健康服务器池（healthy pool）中进行选择。

因为，负载均衡器应当只选择能正常做出响应的后端服务器，因此就需要有一种判断后端服务器是否健康的方法。为了监视后台服务器的运行状况，运行状态检查服务会定期尝试使用转发规则定义的协议和端口去连接后端服务器。 如果，服务器无法通过健康检查，就会从池中剔除，保证流量不会被转发到该服务器，直到其再次通过健康检查为止。

负载均衡算法

负载均衡算法决定了后端的哪些健康服务器会被选中。 其中常用的算法包括：

- Round Robin（轮询）：为第一个请求选择列表中的第一个服务器，然后按顺序向下移动列表直到结尾，然后循环。
- Least Connections（最小连接）：优先选择连接数最少的服务器，在普遍会话较长的情况下推荐使用。
- Source：根据请求源的 IP 的散列（hash）来选择要转发的服务器。这种方式可以一定程度上保证特定用户能连接到相同的服务器。

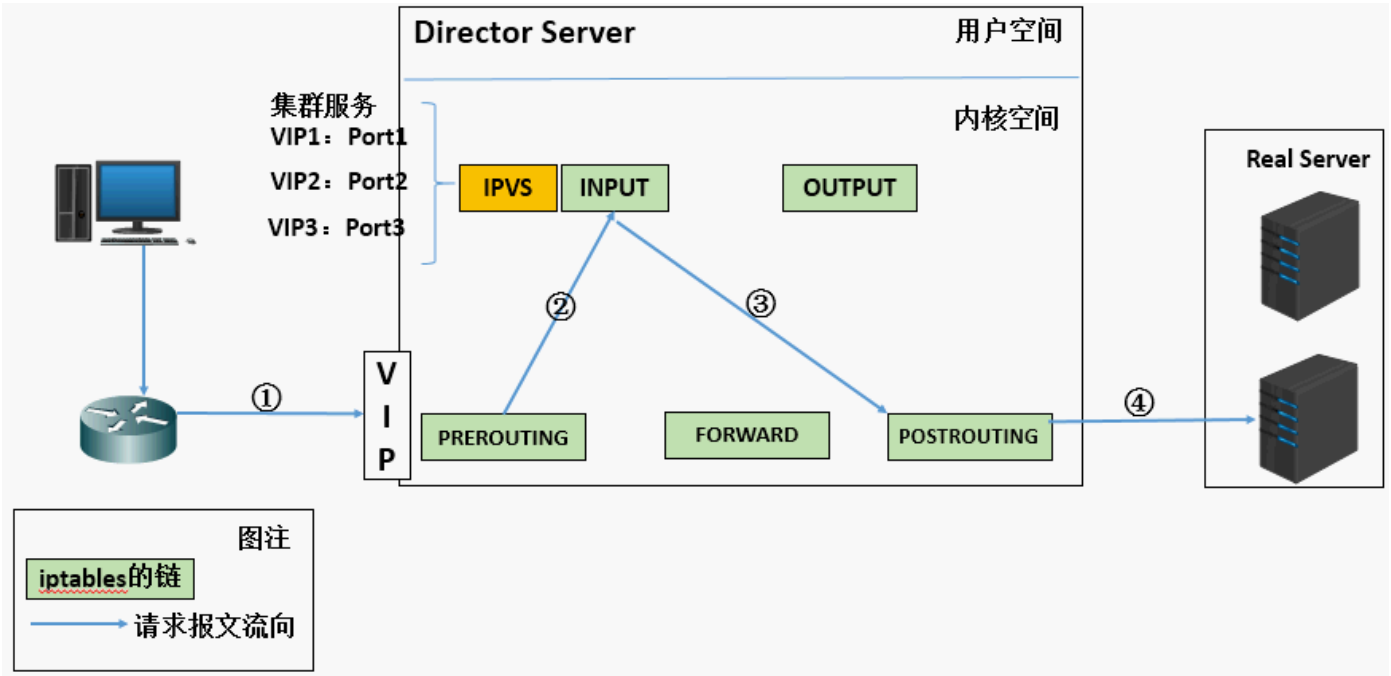
如果你的应用需要处理状态而要求用户能连接到和之前相同的服务器。可以通过 Source 算法基于客户端的 IP 信息创建关联，或者使用粘性会话（sticky sessions）。

除此之外，想要解决负载均衡器的单点故障问题，可以将第二个负载均衡器连接到第一个上，从而形成一个集群。

LVS相关了解.

LVS是 Linux Virtual Server 的简称，也就是Linux虚拟服务器。这是一个由章文嵩博士发起的一个开源项目，它的官方网站是LinuxVirtualServer现在 LVS 已经是 Linux 内核标准的一部分。使用 LVS 可以达到的技术目标是：通过 LVS 达到的负载均衡技术和 Linux 操作系统实现一个高性能高可用的 Linux 服务器集群，它具有良好的可靠性、可扩展性和可操作性。 从而以低廉的成本实现最优的性能。LVS 是一个实现负载均衡集群的开源软件项目，LVS架构从逻辑上可分为调度层、Server集群层和共享存储。

LVS的基本工作原理:



1. 当用户向负载均衡调度器（Director Server）发起请求，调度器将请求发往至内核空间
2. PREROUTING链首先会接收到用户请求，判断目标IP确定是本机IP，将数据包发往INPUT链
3. IPVS是工作在INPUT链上的，当用户请求到达INPUT时，IPVS会将用户请求和自己已定义好的集群服务进行比对，如果用户请求的就是定义的集群服务，那么此时IPVS会强行修改数据包里的目标IP地址及端口，并将新的数据包发往POSTROUTING链
4. POSTROUTING链接收数据包后发现目标IP地址刚好是自己的后端服务器，那么此时通过选路，将数据包最终

发送给后端的服务器

LVS的组成:

LVS 由2部分程序组成，包括 `ipvs` 和 `ipvsadm`。

1. ipvs(ip virtual server)：一段代码工作在内核空间，叫ipvs，是真正生效实现调度的代码。
2. ipvsadm：另外一段是工作在用户空间，叫ipvsadm，负责为ipvs内核框架编写规则，定义谁是集群服务，而谁是后端真实的服务器(Real Server)

详细的LVS的介绍可以参考LVS详解.