[SDN: 2](#_Toc381081895)

[*2014.1.15* 2](#_Toc381081896)

[2014.2.11 2](#_Toc381081897)

[2014.2.16 2](#_Toc381081898)

[Python: 3](#_Toc381081899)

[2014.2.10 3](#_Toc381081900)

[2014.2.11 7](#_Toc381081901)

[2014.2.17 8](#_Toc381081902)

[网络: 8](#_Toc381081903)

[2014.2.10 8](#_Toc381081904)

[大数据： 8](#_Toc381081905)

[2014.2.10 8](#_Toc381081906)

[2014.2.12 9](#_Toc381081907)

[2014.2.13 9](#_Toc381081908)

[2014.2.17 9](#_Toc381081909)

[学习心得： 9](#_Toc381081910)

[2014.2.10 9](#_Toc381081911)

[Linux: 10](#_Toc381081912)

[2014.2.11 10](#_Toc381081913)

[2014.2.16 10](#_Toc381081914)

[2014.2.19 11](#_Toc381081915)

[2014.2.21 11](#_Toc381081916)

[2014.2.24 13](#_Toc381081917)

[其他 15](#_Toc381081918)

# SDN:

### 2014.1.15

<http://net.it168.com/a2014/0114/1585/000001585978_all.shtml>

SDN的形态

1：Openflow技术趋势

2：NFV技术发展趋势

3：I2RS技术发展趋势

4：overlay技术发展趋势

### 2014.2.11

看了当前sdn的控制器的文章，考虑有空自己用mininet搭建一个环境，测试下几个开源控制器的性能，然后自己可以跟一个控制器项目看看

### 2014.2.16

虚拟化和云的区别：

1：规模

2：API接口

3：多租户

4：提供自服务平台

5：按需

6：弹性扩展，快速部署

HA(高可用)

控制节点HA

Mysql HA Galera

消息队列HA RabbitMQ

API接口HA Haproxy

虚拟机HA

分布式文件系统

依赖Heat

类vmware

监控

Ceilometer

虚拟机内存监控和Agent

Libvirt的内存监控不准确

配置管理工具：

Puppet

Chef

Saltstack（python开发）

ansible

# Python:

### 2014.2.10

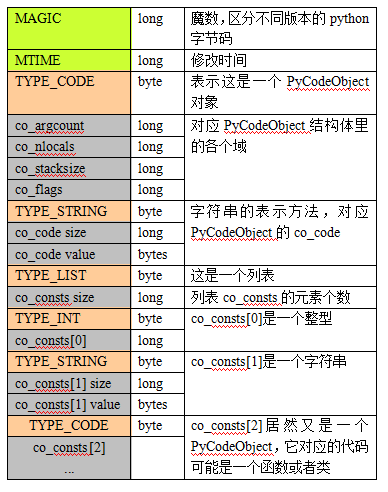
<http://blogread.cn/it/article/6606>

python程序的执行原理：

python是先把代码文件(.py)翻译成字节码，然后把自己码交给虚拟机，由虚拟机完成字节码的翻译。.pyc文件就是字节码。

字节码的产生时机：当import util时候会产生util.pyc会产生util的字节码

Python的字节码结构



如何看字节码：

Python提供了内置的compile可以编译python代码和查看PyCodeObject对象

例如：

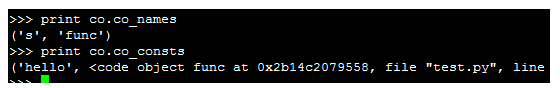
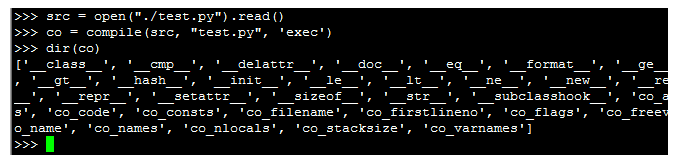
Test.py

S=’hello’

Def func():

Print S

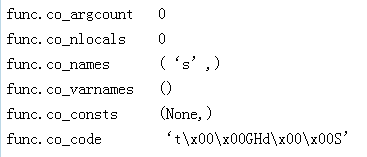
Func()



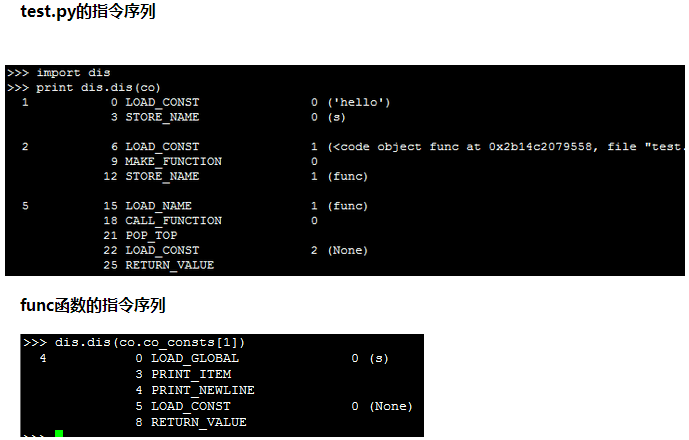
Dir列出的各个域都可以打印出来



Func=co.co\_consts[1]



Python内置的dis模块可以解析co\_code



第一列表示以下几个指令在py文件中的行号;

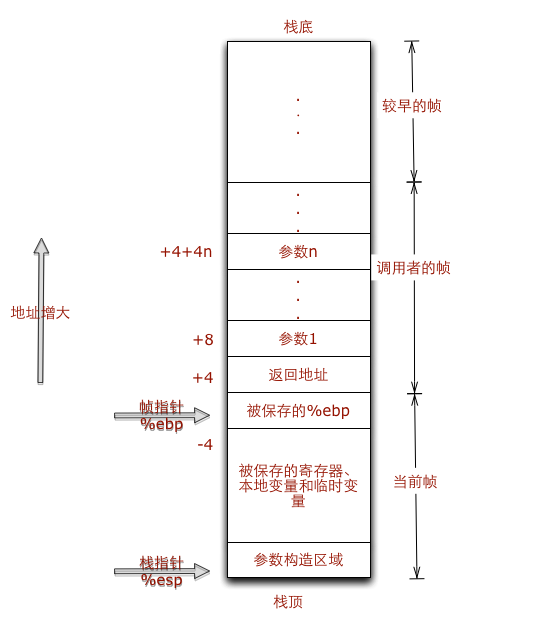
 第二列是该指令在指令序列co\_code里的偏移量;

 第三列是指令opcode的名称，分为有操作数和无操作数两种，opcode在指令序列中是一个字节的整数;

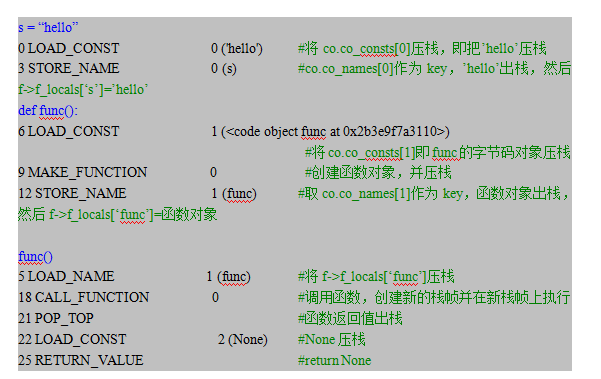
 第四列是操作数oparg，在指令序列中占两个字节，基本都是co\_consts或者co\_names的下标;

 第五列带括号的是操作数说明。

指令含义：与c语言类似：



各个co\_code的含义



### 2014.2.11

一

Csv(comma separated value): (<http://docs.python.org/2/library/csv.html>)就是一种处理csv格式的操作类

Codecs:字符的编码是按照某种在单字节字符和多字节字符之间进行转换的某种方法。从单字节到多字节叫做decodeing，从多字节到单字节叫做encoding；经常用到的是UTF-8和GB2312两种。Codecs模块中重要的函数式lookup，其他函数式：encoder，decoder，StreamReader，StreamWriter和StreamReaderWriter

StringIO：可以认为是内存中的文件系统

了解了generator object，默认有一个next方法。一个斐波那契数列的python实现



了解 os，sys,的操作方法

二

<http://bitworking.org/news/Why_so_many_Python_web_frameworks> python搭建简单web

三

<http://blog.csdn.net/jgood/article/details/5471626> 图片蜘蛛人

四：

<http://www.wsgjp.com.cn/?Redir=%2fDesktop.gspx> 管家婆

待学习python sqlit

### 2014.2.17

安装python2.6安装python pip和setuptools

wget <https://raw.github.com/pypa/pip/master/contrib/get-pip.py>

sh get-pip.py

在linux下配置出了一个简单的server网页

# 网络:

### 2014.2.10

<http://blogread.cn/it/article/6786>

TCP有4个拥塞控制机制：

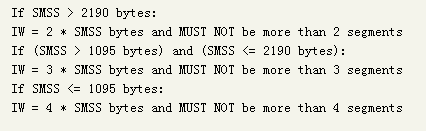
1：slow start, congestion avoidance, fast retransmit and fast recovery

Rwnd的大小取决于BDP，也就是带宽和时延的乘积

但是tcp使用16bit来记录窗口大小，最大也就64kb，超过64kb使用tcp\_window\_scaling机制具体参看<http://packetlife.net/blog/2010/aug/4/tcp-windows-and-window-scaling/>

Cwnd的大小取决于MSS(maximum segment size最大段大小)

min(4 \* MSS, max(2 \* MSS, 4380))



Google's advice ：《An Argument for Increasing TCP's Initial Congestion Window》

# 大数据：

### 2014.2.10

<http://blogread.cn/it/article/2455?f=wb2>

### 2014.2.12

<http://www.uml.org.cn/zjjs/201211065.asp> 分布式

### 2014.2.13

在分布是系统中三个核心：

Consistency/availability/partition tolerance

Consistency（一致性）：这个和数据库ACID的一致性类似，但这里关注的所有数据节点上的数据一致性和正确性，而数据库的ACID关注的是在在一个事务内，对数据的一些约束。

Availability（可用性）：关注的在某个结点的数据是否可用，可以认为某一个节点的系统是否可用，通信故障除外。

Partition Tolerance（分区可容忍性）：是否可以对数据进行分区。这是考虑到性能和可伸缩性。

### 2014.2.17

Google的十个核心技术可以分为四类：

分布式基础设施：GFS、Chubby（分布式锁）和protocol Buffer

分布式大规模数据处理：MapReduce和Sawzall

分布式数据库处理技术BigTable和数据库Sharding

Hadoop生态：

Ambari Zookeeper Hcatlog Oozie

(management) (Coordinstion) (tablemanagement) (Workflow Scheduler)

Hbase Pig(Data Flow) Hive(SQL)

(Big Table) MapReduce

HDFS

# 学习心得：

### 2014.2.10

<http://blogread.cn/it/article/4015?f=hot1> 学习的步骤

<http://chimera.labs.oreilly.com/books/1230000000545/index.html> 网络性能的书

# Linux:

## 启动篇：

### 2014.2.21

Linux启动的5个步骤：

1. 系统启动
2. Stage1：bootloader
3. Stage2：bootloader
4. Kernel
5. Init

启动顺序详细介绍：

1. 系统启动阶段：
2. 打开计算机
3. CPU跳到0xFFFF0处运行BIOS
4. BIOS找到一个可以启动的设备（例如：USB，硬盘，CD-rom）
5. 加载并执行MBR；（可用设备的头512字节）
6. stage1：bootloader：
7. bootloader运行512字节中的命令：

MBR的内容：前446字节：是bootloader

接下来的64字节是四个分区表

最后的8个字节是一个魔术字：0xaa55

具体参见：<http://www.wallcopper.com/os/1140.html>

/boot/grub/menu.conf

1. 执行446字节的内容，然后加载后面的30KB的内容
2. Grub接管控制，显示grub.boot菜单
3. Grub加载用户选择的kernel到内存中
4. Kernel的主要顺序

说明：kernel image的类型有两张：zImage：大小小于512KB，bzImage：大小大于512KB

1. Start() ./arch/i386/boot/head.S
2. Startup\_32() ./arch/i386/boot/compress/head.S

Decompress\_kernel() ./arch/i386/boot/compress/misc.c

1. startup\_32() ./arch/i386kernel/head.S
2. start\_kernel() ./init/main.c
3. cpu\_idle() ./init/main.c
4. 调用init脚本启动第一个进程
5. Init的作用：
6. 是kernel执行的第一个进程
7. 他是所有的进程的root
8. /etc/rc.d/rc.sysinit是第一个进程
9. /etc/inittab是第一个进程的配置文件
10. 在关机的时候，这个root进程会按照顺序关闭进程

问题：MRB的512扇区的程序是怎么写入的？

阅读documentation/x86/boot.txt介绍了，引导的流程。

### 2014.2.25

Initramfs：

相关文章：<http://lwn.net/Articles/14448/>

<http://blog.linux.org.tw/~jserv/archives/001840.html>

<http://blog.linux.org.tw/~jserv/archives/001954.html>

<http://www.cnblogs.com/wwang/archive/2010/10/27/1862222.html>

文档：

Documentation/filesystems

我的理解：

Kernel在加载完成后，需要运行一个用户态的程序，但是用户的程序存在文件系统中，因此内核必须找到一个可以挂在的文件系统才可以完成系统的引导过程。还有一些驱动程序也是在文件系统中，所以如果想挂载文件系统，那么文件系统又需要底层设备，而设备又需要驱动，那么这就是死锁逻辑，所以在挂载真正系统前需要一些东西来引导。

## 进程篇：

## 内存篇：

## 网络篇：

## Fs篇：

## Driver篇：

## 虚拟化篇：

### 2014.2.16

半虚拟化：

在guest的kernel中将所有的敏感操作全部替换掉（需要修改guest kernel的代码）

Pv-ops in linux kernel

Hypercall in xen

Int on 32bit

Syscall on 64bit

这种pv-ops的形式的缺点是：需要有guest kernel的代码并修改。但是windows没有源代码所以没法使用。Windows公司向linux kernel提交了一个patch，使得windows可以跑在PV模式下面。使得跑windows虚拟化方案。

硬件虚拟化：

引入新的CPU模式用来运行guest

Guest mode vs host mode

Non-root mode vs root mode on x86

Vmxon指令，vmxof指令可以在guest mode和host mode之间转换

硬件虚拟化+PV（部分虚拟化）

1：减少vm-exit

2：更快的IO

3：精确的time-keeping

4：搞笑的spinlock

5：接口

Virtio

MSR扩展

其他的虚拟化：

VT-d（DMA、IRQ remapping）

Pass-through （sr-iov）

Ive-migation

虚拟化的书籍：1）intel 系统虚拟化 2）kvm的vtr

轻量级虚拟化对互联网公司更有利。Container 基于cgroup的docker

Vmware Exsi

## 算法篇：

Hlist中的prev只想前一个next的指针，这样直接用\*(node->pprev)=node就可以直接修改或者访问前驱的next的节点了

## 知识篇：

1：安装linux（LFS和组合）

2：linux的使用（shell等等）

3：linux的启动步骤

Grub，init进程

4：一个ko的书写

### 2014.2.11

内核的学习方法：

1. 全面了解内核框架
2. 找一个感兴趣了研究
3. 融入内核社区
4. 坚持

Vim+cscope+ctags

使用Readme /Kconfig/Makefile是将关注的代码范围缩小

git blame可以比较git中的差异

如果你看不懂kernel代码了，不妨用git blame看看它为什么要做这些的修改。  
很多问题可以在patch的changelog中找到

<http://www.cnblogs.com/biyeymyhjob/archive/2012/07/20/2601655.html>

### 2014.2.19

编译linux内核：

1. make defconfig
2. make –jn
3. make module
4. make module\_install
5. make install

第一个ko

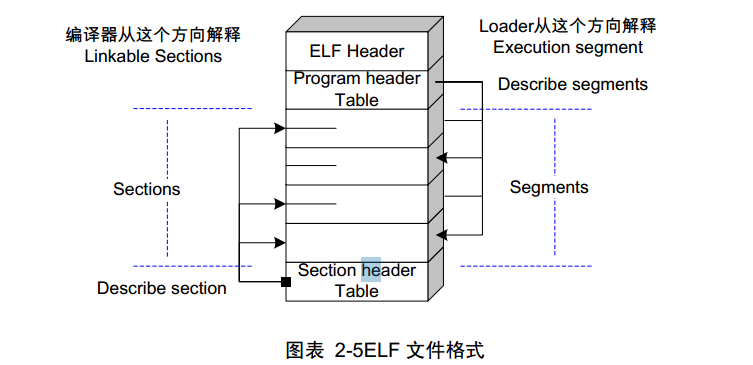
1. makefile的规则
2. 代码的框架

do\_initcalls函数：

1. include/init.h里面

ELF文件的格式：

1. relocatbale:由编译器和汇编器生成，由linker处理它
2. Executable:所有的重定位和符号解析都完成了，只有共享库在运行啥时候才能解析
3. Share object：包含linker需要的符号信息和运行时刻所需的代码



Ld –verbose :用户态的ld script

Arch/i386/kernel/vmlinux.lds.S:内核态的ld script

### 2014.2.24

进程描述的结构体

struct task\_struct

struct pid

struct upid

struct thread\_info

1)process creation

态线程处理函数：clone fork vfork exec\*

内核态的线程：内核态使用的线性地址大于PAGE\_OFFSE.

普通进程用的是4G的线性空间

/boot/config-`uname -r`.x86\_64 这个是内核编译时候的配置文件

kernel\_thread

问题：

内核在什么时机会自己创建进程/线程（中断，用户调用）

内核创建的线程/进程的方式与用户态创建线程进程分别是什么？区别是什么？

kthreadd、migration（任务前一的线程），

ksoftirqd（软中断线程），watchdog（开门狗线程）

内核的线程和用户的线程生成的区别：

了解如下函数的执行过程

do\_fork()

copy\_process

选择一个pid

初始化 vfork修改CLONE\_VFORK标记为1

wake\_up\_new\_task

CLONE\_VFORK\_SET?-----wait\_for\_completion

copy\_process的过程

check flages

dump\_task\_

fork\_init

#define current get\_current()

execve()->sys\_execve()

使用系统调用来执行一个可执行程序？？

do\_execve

打开可执行文件 open\_exec()

bprm\_init

sched

sched\_exec()

prepare\_binprm()

copy环境变量

search\_binary\_handler()

process termination

exit()->do\_exit

exit() and \_exit()

无论用户态还是内核态：那么多的系统调用，我们通过什么好的方式来获得我们需要的系统调用呢？

Linux中文件系统的mode：

-：表示文件

d:表示目录

b：表示块儿设备

c：表示字符设备

l：表示连接

s：表示套接字：用来给不同机器之间不同进程间消息交互的。

在centos中，yum install crash 如果要使用这个工具需要包含kerenl-debuginfo

Yum install kernel-debuginfo

中断上下文与进程上下文的区别：进程上下文从用户发起，进入系统调用，或者没有进入内核，就在用户态执行。而中断上下文是在当前的进程上下文中进行的，借用当前上下文的栈，直接执行的。

### 2014.2.25

Kdump,perf, make menuconfig下面的kernel hacking里找有内存方面的

Intel的x86开发手册

<http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html>

2.6.32内核中，排队自旋锁

自旋锁在初始化的时候会吧自己初始化为0

Spin\_lock:会首相把自己的slock付给本的变量，并把slock+1，然后不停的比较本地变量的高8bit和低8bit（高8bit表示我的顺序，低8bit表示当前的顺序），同时还要不停的将那个实际的slock的低8bit复制到我的本地变量来。

Spin\_unlock:的作用是，直接讲低8bit进行加1处理。

所以核心思想就是：高8bit存储的是我自己是第几位，低8bit存储的是当前轮到第几位了。如果这两个相等了，则就是轮到我了

# 其他

大神的进google经历<http://www.cnblogs.com/figure9/archive/2013/01/09/2853649.html>

<http://blog.csdn.net/Hackbuteer1/article/list/1>

四城，大成网，齐家

四城的建材团购一般价格给力

地板：必美

吊顶：泰力达

<http://blog.csdn.net/sabalol/article/details/1645512>