2014年日报（二）

Subtitle

2022/10/05

Table of Contents

2014年日报（二）

11月16日

VMware挂载物理磁盘

两年前在物理磁盘上装了Linux，后来更新win8的时候没有看Linux就直接更新了win8，导致Linux启动项丢失。想看看Linux里面是否保存了有价值的数据，可是碰巧光驱也不能用了，于是搁置了好几个月了。今天突然想到一个途径，既然Linux可以读取NTFS文件系统个，那么windows上如何读ext4
百度一下，果然又不少解决方案，放下第三方软件或者驱动不表，单说VMware的用途。

1. 用VMware挂载物理磁盘

注意磁盘类型，我查到的我的磁盘是SATA接口，可是选IDE才能用。

2. 用VMware挂载物理磁盘后直接启动Linux系统

~~新建一个虚拟机，使用原Linux的分区的物理磁盘。~~，这个方法失败了，大概是因为启动项被破坏了。懒得修复启动项，在虚拟磁盘上装个图形界面Ubuntu，然后在挂载物理磁盘，可以达到备份数据的目的。

参考资料

1. http://bbs.chinaunix.net/thread-2081416-1-1.html
2. http://blog.csdn.net/zhchch/article/details/1885793

11月17日

kvm: 23090: cpu0 unhandled wrmsr 0x391 data 2000000f

I installed kvm on CentOS 6.4. I created virtual server CentOS*test on it. When I start CEntOS*test I have message:

```
kvm: 23090: cpu0 unhandled wrmsr 0x391 data 2000000f
```

What is it mean? CentOS_test works fine, but this message concerned me.

It's a harmless warning message. Your guest operating system is probing its virtual CPU and restoring some CPU state during boot, which can not be done in a virtual environment. See this KVM mailing list post. It should eventually go away with a KVM/QEMU update.

From:
http://serverfault.com/questions/505152/kvm-23090-cpu0-unhandled-wrmsr-0x391-data-2000000f

我想同时维护两个git托管平台的项目

如果你的项目同时托管在了两个git平台上（比如github和CODE），想同步维护两个平台上的同一个项目，

也是很方便的。

1、设置账号绑定

我们预设使用场景是用户已经使用github为主要协作平台，CODE平台为同步平台

首先你需要在两个平台上都添加上同一份公钥。关于如何生成和添加公钥请参看CODE帮助管理公钥

添加完公钥后，需要在命令行中配置git账号的信息。两个平台最好使用同样的用户名和注册邮箱。如果注册邮箱不同，你可以把CODE的注册邮箱添进github的邮箱列表（github支持一个账号绑定多个邮箱）。

以下是设置账号的git命令：

```
git config set user.name 设置绑定用户名，此处可以与平台用户名不同。
```

```
git config set user.email xxx@xxx.com 此处邮箱需为CODE注册邮箱
```

```
git remote add code <项目地址> 项目地址填写形如： git@code.csdn.net:CSDN_Code/<项目名>.git 的SSH地址
```

2、git pull和git push

按照第一步完成设置后，一般而言，你本地代码仓将会有两个远端地址。指向github的远端主分支地址为origin master，指向CODE平台远端主分支地址为code master。

根据绑定账号时的设置，从两个平台回拉和推送代码的命令分别是：

```
从github回拉： git pull origin master

推送到github： git push origin master

从CODE回拉： git pull code master

推送到CODE： git push code master
```

为了保持本地项目处于最新状态，建议您在每次修改项目之前都是用git pull命令确认一下本地与远端的代码保持同步。

3、CODE平台的自动同步功能

如果你仅将CODE平台作为github项目的镜像站，而不打算在上面做任何独立提交和更新，可以使用CODE平台的“ 自动同步功能 ”。

该功能只支持git版本管理项目 该功能使用的是 – force模式，在CODE平台上的所作改动将被清除。
在CODE平台的操作不能被逆向同步到github等网站 由于自动同步这一功能的危险性和比较占用系统资源，如果您确实需要这一功能，请发送邮件到codesupport@csdn.net，注明您项目的github地址和CODE地址，申请开通自动同步。

From: https://code.csdn.net/help/CSDN_Code/code_support/new_7

11月18日

windows重置密码

windows server 2003 在服务器领域 就跟XP一样 也是一个经典，虽然现在有2008 以及2012 但是服务器上用的2003的还是很多。当然linux除外
在windows使用的过程中会出现忘记密码的情况很正常 但是对于XP 或者win7 windows2003没那么容易破解密码，因此解决server密码问题如下

在windows中摁5下shift之后会提示启用粘滞键 这是因为摁5下shift键之后就调
用%SystemDrive%\windows\system32\sethc.exe来启动粘滞键功能因此就需要
找到cmd.exe应用程序位置：%SystemDrive%\windows\system32\cmd.exe
cmd.exe就是在运行里输入cmd的时候所调用的命令窗口
将cmd.exe替换为sethc.exe，在登录的时候就可以摁5下粘滞键来启动cmd.exe
再输入net user administrator “ 密码 ”
没有密码登录的时候可以用光盘或者U盘引导 进入PE 然后再修改

From http://songjingang.blog.51cto.com/4151716/1035612

ext3文件系统超级块损坏

```
# mount /dev/sde1 /foo

mount wrong fs type bad option bad superblock on /dev/sde1or too many mounted file systems
这个错误信息标识 /dev/sde1 设备上的 ext3 文件系统的超级块损坏了，ext3 文件系统的元数据保存在超级块中。

ext3 文件系统还有一些备份的超级块，可以尝试用备份的超级块加载 ext3 文件系统和修复 ext3 文件系统。

备份的超级块信息可以通过以下命令获得，这个命令模拟 ext3 文件系统创建时的动作并打印出备份超级块的位置，给出的位置缺省是以4k为单位的，mount在使用时需要为它提供以1k为单位的偏移，需要乘4：

注意！一定要使用\"-n\"作为参数模拟 ext3 文件系统的创建而不是真的创建 ext3 文件系统

# mkfs.ext3 -n /dev/hda7

mke2fs 1.38 30-Jun-2005

Filesystem label=

OS type Linux

Block size=4096 log=2

Fragment size=4096 log=2

2198880 inodes 4393738 blocks

219686 blocks 5.00% reserved for the super user
```

```
First data block=0

135 block groups

32768 blocks per group， 32768 fragments per group

16288 inodes per group

Superblock backups stored on blocks：

32768， 98304， 163840， 229376， 294912， 819200， 884736， 1605632， 2654208，

4096000
```

使用备份的超级块来加载 ext3 文件系统的命令：

语法： mount.ext3 -o sb=n

```
# mount.ext3 -o sb=131072 /dev/hda7 /media/hda7
```

使用备份的超级块来修复 ext3 文件系统的命令

语法：fsck.ext3 -b superblock

```
# fsck.ext3 -b 32768 /dev/hda7
```

From:
http://zhidao.baidu.com/link?url=_1zLXwRjO9s8aCsU8l8mN6JRkbYBlQmWnojPnkBGSKKV7OLDpgckaJCcIKMpuqOcjqxxoguakgcNv2fr1ObRuq

11月19日

windows查看SN

sn.vbs

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\CIMV2")
Set colItems = objWMIService.InstancesOf("Win32_BIOS")
For Each objItem In colItems
WScript.Echo "SerialNumber: " & objItem.SerialNumber
Next
```

11月24日

ssh连接报错搜集

Read from socket failed: Connection reset by peer

10.69.3.51： 文件系统错误造成无法读取

ssh_exchange_identification: Connection closed by remote host

管理卡有报错：The Drive “Drive 0” has been disabled due to a detected fault，机器屏幕显示I/O error ,dev sda,

ping: sendto: No buffer space available

freebsd的系统，网上都说是修改内核参数。最后发现是因为路由设置错了。。

11月26日

ipmitool将终端同步至本地

ipmitool -I lanplus -H ip -U root -P password sol activate

- http://phorum.vbird.org/viewtopic.php?f=2&t=31725
- http://docs.linuxtone.org/ebooks/Dell/ipmitool.pdf
- http://my.oschina.net/davehe/blog/88801

11月27日

JavaScript作用域

```
<script>
  var yourname = "yourname";
   myname = "myname";

   changename();

  function changename(){
      alert("yourname is " + yourname + " , myname is " + myname); //输出 yourname is yourname , myname is myname
  }
</script>
```

在changename()函数里加var yourname = “360”;myname = “qq”;

```
<script>
  var yourname = "yourname";
   myname = "myname";

   changename();

  function changename(){
```

```
        alert("yourname is " + yourname + " , myname is " + myname); //输出 yourname is undefined , myname is myname
        var yourname = "360";
        myname = "qq";
    }
</script>
```

12月15日

KVM虚拟机PPTP拨号619错误

在宿主机中加载相关模块

```
/sbin/modprobe ip_nat_pptp
/sbin/modprobe ip_conntrack_pptp
```

可以写到/etc/rc.local里开机启动。

参考：http://www.2cto.com/os/201305/213208.html

curl中-d选项不能和-I连用

curl中为什么-d选项不能和-I连用：http://justwinit.cn/post/6849/

curl使用chrome的cookie

```
Hit F12 to open the developer console.
Look at the Network tab.
Do whatever you need to on the web site to trigger the action you're interested in
Right click the relevant request, and select "Copy as cURL"
This will give you the curl command for the action you triggered, fully populated with cookies and all.
You can of course also copy the flags as a basis for new curl commands.
```

参考：http://stackoverflow.com/questions/21919156/how-do-i-copy-cookies-from-chrome

shell 密码星号显示

```
#!/bin/sh


getchar() {
    stty cbreak -echo
    dd if=/dev/tty bs=1 count=1 2>/dev/null
    stty -cbreak echo
}


printf "Please input your passwd: "


while : ; do
```

```
    ret=`getchar`
    if [ "$ret" = "" ]; then
        echo
        break
    fi
    str="$str$ret"
#    printf "*"
done
echo "your password is:$str"
```

另一种方法，不显示密码

```
#! /bin/bash
read -s content
echo $content
```

12月17日

shell删除字符串最后一位

```
DATA=${DATA%?}  #删除最后一位的逗号
```

12月18日

shell取字符串第一个字符

```
shell中怎么取字符串的变量的其中的一个字符

s=abcdef

取第一个字符 ${s:0:1}
第三个字符 ${s:2:1}
```

检测服务器是否开启虚拟化

CPU：Intel i5-2320

系统环境：CentOS 6.3

对于Linux系统，则只能在BIOS中才能确认是否开启了CPU的虚拟化功能。

cat /proc/cpuinfo |egrep 'vmx|svm'\\

如果有vmx字样输出，则表示该CPU为Intel的CPU，且硬件支持虚拟化；如果有svm字样输出，则表示该CPU为AMD的CPU，且硬件支持虚拟化。该命令只能检测硬件是否支持虚拟化，并不能检测BIOS中是否开启了CPU的虚拟化功能，在上述环境中，不论是否开启BIOS中CPU的虚拟化功能，都有vmx字样输出。

screen用法

```
[root@manage ~]# screen -S fsck
[detached]            #ctrl-a,d暂时离开
[root@manage ~]# screen -ls
There is a screen on:
    20012.fsck    (Detached)
1 Socket in /var/run/screen/S-root.

[root@manage ~]# screen -r fsck
[screen is terminating]     #exit退出
[root@manage ~]#
```

标题

理解DOMString、Document、FormData、Blob、File、ArrayBuffer数据类型
http://www.zhangxinxu.com/wordpress/?p=3725

12月19日

sed匹配多行

```
[root@HADOOP-219 autoticket]# cat file.html |sed -n "/标题/,/<\/dd>/p"
            <dt>标题</dt>
            <dd>服务器（10.77.136.106） /data4 只读</dd>
[root@HADOOP-219 autoticket]# cat file.html |sed -n "/标题/,/<\/dd>/p"|awk -F '[>|<]' 'NR==2{print $2}'
dd
[root@HADOOP-219 autoticket]# cat file.html |sed -n "/标题/,/<\/dd>/p"|awk -F '[>|<]' 'NR==2{print $3}'
服务器（10.77.136.106） /data4 只读
```

12月20日

shell删除字符串后n位

用了一个比较笨的办法
待处理字符串具有下面的形式：

```
HTTP_分类号_TP3-2012-2014-1-10
SDN_分类号_TP3-2012-2014-1-10
版本控制_分类号_TP3-2012-2014-1-10
数据结构_分类号_TP3-2012-2014-1-10
虚拟化技术-2013-2014-1-100
IBM-2013-2014-1-1
Shell_分类号_TP3-2012-2014-1-10
编译_分类号_TP3-2012-2014-1-10
```

想去掉关键词后面的年份等标示，简化目录名称，代码如下：

```
for id in `ls`;do var=${id%??????????????} && mv $id $var;done
```

处理后形式：

```
HTTP_分类号_TP3
Linux_分类号_TP3
Python_分类号_TP3
TCP_分类号_TP3
mpi_分类号_TP3
编译_分类号_TP3
分布式计算_分类号_TP3
前端_分类号_TP3
数学_分类号_TP3
```

12月21日

curl获取http返回值

通过curl的-w参数我们可以自定义curl的输出，%{httpcode}代表http状态码 <code bash> # curl -I -m 10 -o /dev/null -s -w %{httpcode} www.letuknowit.com</code> 上面的输出是不含换行的，如果需要换行的话，加上\n

```
# curl -I -m 10 -o /dev/null -s -w %{http_code}  www.letuknowit.com
200# curl -I -m 10 -o /dev/null -s -w %{http_code}"\n"  www.letuknowit.com
200
```

curl -w参数

From: http://seofangfa.com/shell/curl-w-write-out.html

顾名思义，write-out的作用就是输出点什么。curl的-w参数用于在一次完整且成功的操作后输出指定格式的内容到标准输出。

输出格式由普通字符串和任意数量的变量组成，输出变量需要按照%{variable_name}的格式，如果需要输出%，double一下即可，即%%，同时，\n是换行，\r是回车，\t是TAB□curl会用合适的值来替代输出格式中的变量，所有可用变量如下：

url_effective 最终获取的url地址，尤其是当你指定给curl的地址存在301跳转，且通过-L继续追踪的情形。

httpcode http状态码，如200成功,301转向,404未找到,500服务器错误等。(The numerical response code that was found in the last retrieved HTTP(S) or FTP(s) transfer. In 7.18.2 the alias responsecode was added to show the same info.)

http_connect The numerical code that was found in the last response (from a proxy) to a curl CONNECT request. (Added in 7.12.4)

time_total 总时间，按秒计。精确到小数点后三位。 （The total time, in seconds, that the full operation lasted. The time will be displayed with millisecond resolution.□

time_namelookup DNS解析时间,从请求开始到DNS解析完毕所用时间。(The time, in seconds, it took from the start until the name resolving was completed.)

time*connect 连接时间,从开始到建立TCP连接完成所用时间,包括前边DNS解析时间，如果需要单纯的得到连接时间，用这个time*connect时间减去前边time_namelookup时间。以下同理，不再赘述。(The time, in seconds, it took from the start until the TCP connect to the remote host (or proxy) was completed.)

time_appconnect 连接建立完成时间，如SSL/SSH等建立连接或者完成三次握手时间。(The time, in seconds, it took from the start until the SSL/SSH/etc connect/handshake to the remote host was completed. (Added in 7.19.0))

time_pretransfer 从开始到准备传输的时间。(The time, in seconds, it took from the start until the file transfer was just about to begin. This includes all pre-transfer commands and negotiations that are specific to the particular protocol(s) involved.)

time*redirect 重定向时间，包括到最后一次传输前的几次重定向的DNS解析，连接，预传输，传输时间。(The time, in seconds, it took for all redirection steps include name lookup, connect, pretransfer and transfer before the final transaction was started. time*redirect shows the complete execution time for multiple redirections. (Added in 7.12.3))

time*starttransfer 开始传输时间。在发出请求之后，Web 服务器返回数据的第一个字节所用的时间(The time, in seconds, it took from the start until the first byte was just about to be transferred. This includes time*pretransfer and also the time the server needed to calculate the result.)

size_download 下载大小。(The total amount of bytes that were downloaded.)

size_upload 上传大小。(The total amount of bytes that were uploaded.)

size_header 下载的header的大小(The total amount of bytes of the downloaded headers.)

size_request 请求的大小。(The total amount of bytes that were sent in the HTTP request.)

speed_download 下载速度，单位-字节每秒。(The average download speed that curl measured for the complete download. Bytes per second.)

speed_upload 上传速度,单位-字节每秒。(The average upload speed that curl measured for the complete upload. Bytes per second.)

content_type 就是content-Type，不用多说了，这是一个访问我博客首页返回的结果示例(text/html; charset=UTF-😎(The Content-Type of the requested document, if there was any.)

num_connects Number of new connects made in the recent transfer. (Added in 7.12.3)

num_redirects Number of redirects that were followed in the request. (Added in 7.12.3)

redirect_url When a HTTP request was made without -L to follow redirects, this variable will show the actual URL a redirect would take you to. (Added in 7.18.2)

ftp*entry*path The initial path libcurl ended up in when logging on to the remote FTP server. (Added in 7.15.4)

ssl*verify*result ssl认证结果，返回0表示认证成功。(The result of the SSL peer certificate verification that was requested. 0 means the verification was successful. (Added in 7.19.0))

注意：

1、若多次使用-w参数，按最后一个的格式输出。

2、在使用上面变量的时候，注意看后面小括号中的 Added in XXX，这个表示支持该变量curl所需的最低版本，查看curl版本使用curl -V。如果版本不够，curl会提示类似下面的错误。

curl: unknown –write-out variable: ‘redirect_url’

curl -w用法一例

检查一批URL的HTTP状态：cat url.txt|while read line; do curl -I $line -m 5 –connect-timeout 5 -o /dev/null -s -w “$line “%{http_code}”\n”; done>ok.txt

12月25日

curl

http://blog.csdn.net/five3/article/details/7181521
http://blog.csdn.net/xifeijian/article/details/9367339

标题

Printed on: 2022/10/05 18:57