linux系统下单独修改某个用户的语言环境-lurou-ITPUB博客

用户提出一个linux系统语言环境相关的需求：
需要从windows客户端上传到linux服务器，并且需要在linux环境下显示中文，但是该服务器上安装有英文字符集的oracle数据库，一时难倒我了，上网查到一个解决办法。总结如下：
新建一个用户，vi ~/.bashrc ,添加 export LC_CTYPE="zh_CN.gbk"，保存，退出。不放心的话退出之后再export一下
据说修改/etc/profile也可以，不过没试过。
用户的需求，说白了，就是他想上传包含中文内容的文件到linux英文操作系统上，并且有时候需要more一下文件的内容，需要显示中文字符串。
经测试，suse linux 11 sp1环境下，修改过程如下：
1、修改profile
cp /etc/profile /etc/profile.bak
vi /etc/profile
再最后的位置添加如下行：
export LC_CTYPE="zh_CN.GBK"
2、reboot
[@more@]关于locale的设定
locale是国际化与本土化过程中的一个非常重要的概念，个人认为，对于中文用户来说，通常会涉及到的国际化或者本土化，大致包含三个方面：看中文，写中文，与window中文系统的兼容和通信。从实际经验上看来，locale的设定与看中文关系不大，但是与写中文，及window分区的挂载方式有很密切的关系。本人认为就像一个纯英文的Windows能够浏览中文，日文或者意大利文网页一样，你不需要设定locale就可以看中文。那么，为什么要设定locale呢？什么时候会用到locale呢？
Tags: locale 设定 原因 解释
一、为什么要设定locale
正如前面我所讲的，设定locale与你能否浏览中文的网页没有直接的关系，即便你把locale设置成 en_US.ISO-8859-1这样一个标准的英文locale你照样可以浏览中文的网页，只要你的系统里面有相应的字符集（这个都不一定需要）和合适的字体（如simsun），浏览器就可以把网页翻译成中文给你看。具体的过程是网络把网页传送到你的机器上之后，浏览器会判断相应的编码的字符集，根据网页采用的字符集，去字体库里面找合适的字体，然后由文字渲染工具把相应的文字在屏幕上显示出来。
在下文本人会偶尔把字符集比喻成密码本，个人觉得对于一些东西比较容易理解，假如你不习惯的话，把全文copy到任何文本编辑器，用字符集替换密码本即可。
那有时候网页显示乱码或者都是方框是怎么回事呢？个人认为，显示乱码是因为设定的字符集不对(或者没有相应的字符集)，例如网页是用UTF-8 编码的，你非要用GB2312去看，而系统根据GB2312去找字体，然后在屏幕上显示，当然是一堆的乱码，也就是说你用一个错误的密码本去翻译发给你的电报，当然内容那叫一个乱；至于有些时候浏览的网页能显示一部分汉字，但有很多的地方是方框，能够显示汉字说明浏览器已经正确的判断出了网页的编码，并在字体库里面找到了相应的文字，但是并不是每个字体库都包含某个字符集全部的字体的缘故，有些时候会显示不完全，找一个比较全的支持较多字符集的字体就可以了。
既然我能够浏览中文网页，那为什么我还要设定locale呢？
其实你有没有想过这么一个问题，为什么gentoo官方论坛上中文论坛的网页是用UTF-8编码的（虽然大家一直强烈建议用GB2312编码），但是新浪网就是用GB2312编码的呢？而Xorg的官方网页竟然是ISO-8859-15编码的，我没有设定这个locale怎么一样的能浏览呢？这个问题就像是你有所有的密码本，不论某个网站是用什么字符集编码的，你都可以用你手里的密码本把他们翻译过来，但问题是虽然你能浏览中文网页，但是在整个操作系统里面流动的还是英文字符。所以，就像你能听懂英语，也能听懂中文。
最根本的问题是：你不可以写中文。
当你决定要写什么东西的时候，首先要决定的一件事情是用那种语言，对于计算机来说就是你要是用哪一种字符集，你就必须告诉你的linux系统，你想用那一本密码本去写你想要写的东西。知道为什么需要用GB2312字符集去浏览新浪了吧，因为新浪的网页是用GB2312写的。
为了让你的Linux能够输入中文，就需要把系统的locale设定成中文的(严格说来是locale中的语言类别LC_CTYPE)，例如zh_CN.GB2312、zh_CN.GB18030或者zh_CN.UTF-8。很多人都不明白这些古里古怪的表达方式。这个外星表达式规定了什么东西呢？这个问题稍后详述，现在只需要知道，这是locale的表达方式就可以了。
二、到底什么是locale?
locale这个单词中文翻译成地区或者地域，其实这个单词包含的意义要宽泛很多。Locale是根据计算机用户所使用的语言，所在国家或者地区，以及当地的文化传统所定义的一个软件运行时的语言环境。
这个用户环境可以按照所涉及到的文化传统的各个方面分成几个大类，通常包括用户所使用的语言符号及其分类(LC_CTYPE)，数字(LC_NUMERIC)，比较和排序习惯(LC_COLLATE)，时间显示格式(LC_TIME)，货币单位(LC_MONETARY)，信息主要是提示信息,错误信息, 状态信息, 标题, 标签, 按钮和菜单等(LC_MESSAGES)，姓名书写方式(LC_NAME)，地址书写方式(LC_ADDRESS)，电话号码书写方式(LC_TELEPHONE)，度量衡表达方式(LC_MEASUREMENT)，默认纸张尺寸大小(LC_PAPER)和locale对自身包含信息的概述(LC_IDENTIFICATION)。
所以说，locale就是某一个地域内的人们的语言习惯和文化传统和生活习惯。一个地区的locale就是根据这几大类的习惯定义的，这些

locale定义文件放在/usr/share/i18n/locales目录下面，例如en_US, zh_CN and de_DE@euro都是locale的定义文件，这些文件都是用文本格式书写的，你可以用写字板打开，看看里边的内容，当然出了有限的注释以外，大部分东西可能你都看不懂，因为是用的Unicode的字符索引方式。
对于de_DE@euro的一点说明，@后边是修正项，也就是说你可以看到两个德国的locale：
/usr/share/i18n/locales/de_DE@euro
/usr/share/i18n/locales/de_DE
打开这两个locale定义，你就会知道它们的差别在于de_DE@euro使用的是欧洲的排序、比较和缩进习惯，而de_DE用的是德国的标准习惯。
上面我们说到了zh_CN.GB18030的前半部分，后半部分是什么呢？大部分Linux用户都知道是系统采用的字符集。
三、什么是字符集？
字符集就是字符，尤其是非英语字符在系统内的编码方式，也就是通常所说的内码，所有的字符集都放在 /usr/share/i18n/charmaps，所有的字符集也都是用Unicode编号索引的。Unicode用统一的编号来索引目前已知的全部的符号。而字符集则是这些符号的编码方式，或者说是在网络传输，计算机内部通信的时候，对于不同字符的表达方式，Unicode是一个静态的概念，字符集是一个动态的概念，是每一个字符传递或传输的具体形式。就像Unicode编号U59D0是代表姐姐的“姐”字，但是具体的这个字是用两个字节表示，三个字节，还是四个字节表示，是字符集的问题。例如：UTF-8字符集就是目前流行的对字符的编码方式，UTF-8用一个字节表示常用的拉丁字母，用两个字节表示常用的符号，包括常用的中文字符，用三个表示不常用的字符，用四个字节表示其他的古灵精怪的字符。而GB2312字符集就是用两个字节表示所有的字符。需要提到一点的是Unicode除了用编号索引全部字符以外，本身是用四个字节存储全部字符，这一点在谈到挂载windows分区的时候是非常重要的一个概念。所以说你也可以把Unicode看作是一种字符集（我不知道它和UTF-32的关系，反正UTF-32就是用四个字节表示所有的字符的），但是这样表述符号是非常浪费资源的，因为在计算机世界绝大部分时候用到的是一个字节就可以搞定的26个字母而已。所以才会有UTF-8，UTF-16等等，要不然大同世界多好，省了这许多麻烦。
四、zh_CN.GB2312到底是在说什么？
Locale 是软件在运行时的语言环境, 它包括语言(Language), 地域 (Territory) 和字符集(Codeset)。一个locale的书写格式为: 语言[_地域[.字符集]]. 所以说呢，locale总是和一定的字符集相联系的。下面举几个例子：

- 我说中文，身处中华人民共和国，使用国标2312字符集来表达字符。

zh_CN.GB2312＝中文_中华人民共和国＋国标2312字符集。

- 我说中文，身处中华人民共和国，使用国标18030字符集来表达字符。

zh_CN.GB18030＝中文_中华人民共和国＋国标18030字符集。

- 我说中文，身处台湾省，使用国标Big5字符集来表达字符。

zh_TW.BIG5=中文_台湾.大五码字符集

- 我说英文，身处大不列颠，使用ISO-8859-1字符集来表达字符。

en_GB.ISO-8859-1=英文_大不列颠.ISO-8859-1字符集

- 我说德语，身处德国，使用UTF-8字符集，习惯了欧洲风格。

de_DE.UTF-8@euro＝德语_德国.UTF-8字符集@按照欧洲习惯加以修正
注意不是de_DE@euro.UTF-8，所以完全的locale表达方式是 [语言[_地域][.字符集] [@修正值]
生成的locale放在/usr/lib/locale/目录中，并且每个locale都对应一个文件夹，也就是说创建了de_DE@euro.UTF-8 locale之后，就生成/usr/lib/locale/de_DE@euro.UTF-8/目录，里面是具体的每个locale的内容。
五、怎样去自定义locale
在gentoo生成locale还是很容易的，首先要在USE里面加入userlocales支持，然后编辑locales.build文件，这个文件用来指示glibc生成

locale文件。很多人不明白每一个条目是什么意思。 其实根据上面的说明现在应该很明确了。
File: /etc/locales.build
en_US/ISO-8859-1
en_US.UTF-8/UTF-8
zh_CN/GB18030
zh_CN.GBK/GBK
zh_CN.GB2312/GB2312
zh_CN.UTF-8/UTF-8
上面是我的locales.build文件，依次的说明是这样的：

- en_US/ISO-8859-1：生成名为en_US的locale，采用ISO-8859-1字符集，并且把这个locale作为英文_美国locale类的默认值，其实它和en_US.ISO-8859-1/ISO-8859-1没有任何区别。
- en_US.UTF-8/UTF-8：生成名为en_US.UTF-8的locale，采用UTF-8字符集。
- zh_CN/GB18030：生成名为zh_CN的locale，采用GB18030字符集，并且把这个locale作为中文_中国locale类的默认值，其实它和zh_CN.GB18030/GB18030没有任何区别。
- zh_CN.GBK/GBK：生成名为zh_CN.GBK的locale，采用GBK字符集。
- zh_CN.GB2312/GB2312：生成名为zh_CN.GB2312的locale，采用GB2312字符集。
- zh_CN.UTF-8/UTF-8：生成名为zh_CN.UTF-8的locale，采用UTF-8字符集。

关于默认locale，默认locale可以简写成en_US或者zh_CN的形式，只是为了表达简单而已没有特别的意义。
Gentoo在locale定义的时候掩盖了一些东西，也就是locale的生成工具：localedef。在编译完glibc之后你可以用这个localedef 再补充一些locale，就会更加理解locale了。具体的可以看 localedef 的manpage。

$localedef -f 字符集 -i locale定义文件 生成的locale的名称
例如
$localedef -f UTF-8 -i zh_CN zh_CN.UTF-8
上面的定义方法和在locales.build中设定zh_CN.UTF-8/UTF-8的结果是一样一样的。
六、locale的五脏六腑
刚刚生成了几个locale，但是为了让它们生效，必须告诉Linux系统使用那(几)个locale。这就需要对locale的内部机制有一点点的了解。在前面我已经提到过，locale把按照所涉及到的文化传统的各个方面分成12个大类，这12个大类分别是：

- 语言符号及其分类(LC_CTYPE)
- 数字(LC_NUMERIC)
- 比较和排序习惯(LC_COLLATE)
- 时间显示格式(LC_TIME)
- 货币单位(LC_MONETARY)
- 信息主要是提示信息,错误信息,状态信息,标题,标签,按钮和菜单等(LC_MESSAGES)
- 姓名书写方式(LC_NAME)
- 地址书写方式(LC_ADDRESS)
- 电话号码书写方式(LC_TELEPHONE)
- 度量衡表达方式(LC_MEASUREMENT)
- 默认纸张尺寸大小(LC_PAPER)
- 对locale自身包含信息的概述(LC_IDENTIFICATION)。

其中，与中文输入关系最密切的就是 LC_CTYPE， LC_CTYPE 规定了系统内有效的字符以及这些字符的分类，诸如什么是大写字母，小写字母，大小写转换，标点符号、可打印字符和其他的字符属性等方面。而locale定义zh_CN中最最重要的一项就是定义了汉字(Class “hanzi”)这一个大类，当然也是用Unicode描述的，这就让中文字符在Linux系统中成为合法的有效字符，而且不论它们是用什么字符集编码的。

```
LC_CTYPE
% This is a copy of the "i18n" LC_CTYPE with the following modifications: - Additional classes: hanzi
copy "i18n"
class "hanzi"; /
% ..;/
..;/
;;;;;;;;/
;;;;;;;;/
;;;;
END LC_CTYPE
```

在en_US的locale定义中，并没有定义汉字，所以汉字不是有效字符。所以如果要输入中文必须使用支持中文的locale，也就是zh_XX，如zh_CN，zh_TW，zh_HK等等。

另外非常重要的一点就是这些分类是彼此独立的，也就是说LC_CTYPE，LC_COLLATE和 LC_MESSAGES等等分类彼此之间是独立的，可以根据用户的需要设定成不同的值。这一点对很多用户是有利的，甚至是必须的。例如，我就需要一个能够输入中文的英文环境，所以我可以把LC_CTYPE设定成zh_CN.GB18030，而其他所有的项都是en_US.UTF-8。

七、怎样设定locale呢？

设定locale就是设定12大类的locale分类属性，即 12个LC_*。除了这12个变量可以设定以外，为了简便起见，还有两个变量：LC_ALL和LANG。它们之间有一个优先级的关系：

LC_ALL>LC_*>LANG

可以这么说，LC_ALL是最上级设定或者强制设定，而LANG是默认设定值。

-
  - 如果你设定了LC_ALL＝zh_CN.UTF-8，那么不管LC_*和LANG设定成什么值，它们都会被强制服从LC_ALL的设定，成为 zh_CN.UTF-8。
  - 假如你设定了LANG＝zh_CN.UTF-8，而其他的LC_*=en_US.UTF-8，并且没有设定LC_ALL的话，那么系统的locale设定以LC_*=en_US.UTF-8。
  - 假如你设定了LANG＝zh_CN.UTF-8，而其他的LC_*，和LC_ALL均未设定的话，系统会将LC_*设定成默认值，也就是LANG的值 zh_CN.UTF-8 。
  - 假如你设定了LANG＝zh_CN.UTF-8，而其他的LC_CTYPE=en_US.UTF-8，其他的LC_*，和LC_ALL均未设定的话，那么系统的locale设定将是：LC_CTYPE=en_US.UTF-8，其余的 LC_COLLATE，LC_MESSAGES等等均会采用默认值，也就是LANG的值，也就是LC_COLLATE＝LC_MESSAGES＝……＝ LC_PAPER＝LANG＝zh_CN.UTF-8。

所以，locale是这样设定的：

-
  - 如果你需要一个纯中文的系统的话，设定LC_ALL= zh_CN.XXXX，或者LANG= zh_CN.XXXX都可以，当然你可以两个都设定，但正如上面所讲，LC_ALL的值将复盖所有其他的locale设定，不要作无用功。
  - 如果你只想要一个可以输入中文的环境，而保持菜单、标题，系统信息等等为英文界面，那么只需要设定LC_CTYPE＝zh_CN.XXXX，LANG= en_US.XXXX就可以了。这样LC_CTYPE＝zh_CN.XXXX，而

LC_COLLATE＝LC_MESSAGES＝……＝ LC_PAPER＝LANG＝en_US.XXXX。

- 假如你高兴的话，可以把12个LC_*一一设定成你需要的值，打造一个古灵精怪的系统：

LC_CTYPE＝zh_CN.GBK/GBK(使用中文编码内码GBK字符集)；
LC_NUMERIC=en_GB.ISO-8859-1(使用大不列颠的数字系统)
LC_MEASUREMEN=de_DE@euro.ISO-8859-15(德国的度量衡使用ISO-8859-15字符集)
罗马的地址书写方式，美国的纸张设定……。估计没人这么干吧。

-

- 假如你什么也不做的话，也就是LC_ALL，LANG和LC_*均不指定特定值的话，系统将采用POSIX作为lcoale，也就是C locale。