基于Linux操作系统核心的汉字显示（4） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E5_9F_BA_E4_BA_8ELinu_c103_144360.htm 这个方案有一个困难，即xxxx_putc()函数不用缓冲区的地址，而是用一个整数作为参数，所以xxx_putc()无法直接利用相邻的字符来判别该字符是否是汉字。 解决方案是，利用xxxx_putc()的光标们置参数（yy,xx），可以逆推出该字符在缓冲区中的位置，但仍一些小麻烦，在Linux的虚拟终端下，用户可能会上卷该屏幕(Shift Pageup)，导致光标的y座标和相应字符在缓冲区的行数不一致，相应的解决方案是，在逆推的过程中，考虑在屏的参量。 这样一来，我们就又进了一步，得到了一个相对更好的版本。但仍有问题没有解决，敲入turbonetcfg，会发现菜单的边框字符也被当成汉字显示，这是因为，这种边框字符是扩展字符，也使用了字符的低8位，因而被当成汉字显示，这是因为，这种边框字符是扩展字符，也使用了字符的低8位，因而被当作汉字来赤示。例如，单线“ ”的制表符内码为0xC4，当连成一条长线时就是由一连串0xC4组成的，而0Xc4c4正是汉字“ 哪 ”，于是水平的制表符被一连串的“ 哪 ”字替代了，因为制表符的种类比较多，而且垂直制表符与其后面字符的组合形式又多种多样，因而很难判断出相应位置的字符是不是制表符，从理论上说，无论采取什么样的排除算法，都必然存在误判的情况，因为总存在二义性，没有充足的条件来推断出当前字符究竟是制表符还是汉字。 我们一方面寻找更好的排除组合算法，一方面试图寻找其他的解决方案，要想从根本上解决这个问题，必须利用其他的辅助信息，仅仅

利用缓冲区的字符来判断是不够的。经过一番努力，我们发现，在UNIX中使用扩展字符时，都要先输出字符转义序列（Escape sepuence）来切换当前字符集。字符转义序列是以控制字符Ecs为首的控制命令，在UNIX的虚拟终端中完成终端控制命令，这种命令包括移动光标座标、卷屏、删除、切换字符集等等。也就是说，在输出代表制表的字符串之前，通常是要先输出特定的字符转义序列，在console.c里，有根据字符转义序列命令来记录字符状态的变量，结合该变量提供的信息，就可以非常准确地把制表符与汉字区别开来。在如上思路的指引下，我们又产生了新的解决方案，经过改动得到了另一版本。在这个新的版本上，turbonetcfg在初次绘制的时候，制表符与汉字被清晰地区分开，但还有问题：turbonetcfg在重绘的时候（如切换虚拟终端或是移动鼠标光标的），制表符还是变成了汉字，因为重绘完全领带于缓冲区，而这时用来记录字符集状态的变量并不反映当前字符集状态。问题还是没有最终解决，我们又回到了起点。看来问题的最终解决手段必须是把字符集的状态伴随着每一个字符在每一个字符占用16位的缓冲区，低6、8位是ASCII值，完全被利用，高8位饮食前量颜色和背景颜色的属性，也没有多余的空间可以利用，因而只能另外开辟新的缓冲区。为了保持一致性，我们决定在原来的缓冲区后面添加相同大小的缓冲区，用来存放是否汉字的信息。