第25章 tty.c文件

在本章，交互式终端处理程序的神秘面纱将全部被揭开，交互式终端处理程序的主要内容包括：

1)“擦除”和“擦除行”字符的处理。

2) 在输入、输出中，对大写字符终端的字符变换。

3) 在各个特殊字符，例如“回车”符之后延迟时间的插入。

本章说明中提及的一些例程“ gtty”(8165)、“stty”(8183)、“sgtty”(8201)和“ttystty”(8577)已经在上一章中讨论过。

25.1 flushtty(8252)

本过程的目的是使与一终端相关连的各输入、输出队列处于正常的起始状态。其作用是立即终止对终端的传输，丢弃任何累积起来的输入字符。

8258：丢弃在非原始输入队列中的所有信息。

8259：对输出队列执行同样的操作。

8260：唤醒等待从“原始”输入队列中取一字符的所有进程。

8261：对输出队列执行同样的操作。

8263：提高处理机优先级以阻止在执行下面的 while语句期间来自终端的中断。

8264：刷清“原始”输入队列，并且……

8265：将“定界符计数”设置为0。

当满足下列两个条件之一时，“wflushtty”(见下面)和“ttyinput”(8346、8350)调用“flushtty”：

1) 终端不以“原始”方式运行，并从终端接到一“ quit”(退出)或“detete”(删除)字符。

2)“原始”输入队列不符合情理地变得很大 (可能是没有进程正在读来自该终端的输入)。

25.2 wflushtty(8217)

此过程等待终端字符队列为空 (因为所有字符都已被传送 !)，然后调用“flushtty”扫清输入队列。

“wflushtty”由“klclose”调用(8050)。这并不是经常发生的 —— 事实上，仅当引用该终端的所有文件皆被关闭，亦即，通常仅当用户注销 (log off)时，才进行此种操作。

“wflushtty”也由“ttystty”调用(8589)，其调用时机恰好在调整终端环境参数之前。

25.3 字符输入

对于要求从终端获得输入的程序，有一条延伸至“ ttread ”…的过程调用链。

25.3.1 ttread(8535)

8541：检查该终端在逻辑上是起作用的。
8543：如果在非原始输入队列中有字符，或者对“ canon ”(8274)的调用已获成功……
8544：从非原始输入队列传输字符，直至该队列空或者传输的字符已满足用户的要求。

25.3.2 canon(8274)

此过程由“ ttread ”调用(8543)，其功能是从原始输入队列向非原始输入队列传送字符 (在处理“ 擦除 ”和“ 擦除行 ”字符之后，而在大写终端情况下，则还在处理“ 转义 ” (escaped)字符之后。“ 转义 ”字符指的是紧接在“ \ ”符之后的字符)。若非原始输入队列不再是空，则“ canon ”返回非0值。

8284：如果在“ 原始 ”输入队列中的定界符数是 0，则……
8285：如果该终端在逻辑上并不存在，则立即返回。
8287：否则调用“ sleep ”。
注意，在此上下文中，定界符是全1字符(八进制值是377)并由“ ttyinput ”插入(8358)。
8291：设置“ bp ”，使其指向工作数组“ canonb ”中的第3个字符。
8292：开始一个循环(它延伸至8318行)，该循环每次从“ 原始 ”队列中取出一个字符。

则对空界符记数减1，然后退出此循环，亦即跳转至8319行。
方式运行……
bp〔－1〕标记法)不是一个“ \ ”，则执行从8299行至8307
的代码。

并且……
除，将指针“ bp ”退回一字符位置。
一轮循环。
则转移至8290行以丢弃为当前行累积起的所有字符。
4)(通常在终端上由control_D产生)，则忽略该字符(并且不
环。(若此字符位于1行的开始处，则后面的ttread(8544)将
亦即它将读得一 0长记录，这导致该程序接收到正常的

0，或者“ maptab〔c〕 ＝＝c ”，或者该终端是大写终端，

那么……

8310：如果最后第2个字符不是“ \ ”，则用“ maptab〔c〕”代换“ c ”，然后使“ bp ”后退一位置(使“ \ ”字符被擦除)。

25.3.5 字符准备

8315：将c送入“ canonb ”中的下一个字符位置，若此数组现在已满，则离开此循环。

25.3.6 已得到1行

8319：到此点时，在数组“ canonb ”中已装配好1行。

8322：将“ canonb ”中的内容移入非原始输入队列，然后返回“ 成功 ”结果。

25.3.7 注释

1) 为什么“ bp ”从“ canonb ”中的第3个字符开始?对此问题的回答可在8310行找到。

2) 很多处理“ \ ”字符的微妙之处(读者无疑在实际使用UNIX中已经遇到)到现在为止还不是非常明显的。因为对于 c== \ (八进制值134)“ maptab〔c〕”是0，所有“ \ ”字符都复制入“ canonb ”。如果后面一个字符是要判断的 (如#，@或eot(004))，或者是要为大写终端作更改的一个字母字符，那么一个“ \ ”字符最后会被覆盖重写。

25.3.8 ttyinput(8333)

“ canon ”从“ 原始 ”输入队列移出字符。而这些字符首先是由“ [illegible] ”存放在“ 原始 ”输入队列中的，则只要从硬件控制器接收到 1个输入字符，
给“ ttyinput ”的参数是1个字符以及一个指向“ tty ”结构的
8342：如果此字符是“ 回车 ”，并且此终端对此只执行
行 ”对)，则将此字符代换为“ 新行 ”。
8344：如果此终端并不以“ 原始 ”方式运行，并且此
(删除)(7958)，那么调用“ signal ”(3949)将一软件中断发送
个进程，刷清与该终端相关连的所有字符队列，然后返回。
8349：如果“ 原始 ”输入队列已成长得过分大，则刷
后返回。(初看起来这种处理似乎有点粗糙，但这通常就是
8353：如果此终端有一有限字符集，字符是大写字母，
8355：将此字符插入“ 原始 ”输入队列。
8356：如果此终端以“ 原始 ”方式运行，或者此字符是
8357：“ 唤醒 ”等待从该终端输入的每一个进程，也将
列，然后将定界符计数增 1。注意，这里是“ putc ”的可能
别出的一个位置。发生在此处的一个失误可以解释在 8316行
8361：最后，若该输入字符是要回打的，亦即该终端

复制至输出队列，然后安排使其发送(ttstart)回终端。

25.4 字符输出

25.4.1 ttwrite(8550)

当要将输出发送至终端时，经由“ klwrite ”(8067)调用此过程。

8556：若此终端在逻辑上并不起作用，则不执行任何操作。

8558：对每一个要发送的字符执行一循环……

8560：在队列中仍有合适数量的字符可以发送给终端，在此期间……

8561：恰好是可向终端发送另一个字符的时间，调用“ ttstart ”。

8562：在这里设置“ ASLEEP ”标志(同样在wflushtty(8224)中)是没有任何作用的，在关闭该文件之前决不会查询该标志，也不会清该标志。

8563：进入睡眠状态。同时，中断处理程序将从输出队列取字符，并将它们发送至终端。

8566：调用“ ttyoutput ”，将该字符插入输出队列，然后安排使其被发送至终端。

8568：再次调用“ ttstart ”。

25.4.2 ttstart(8505)

只要有理由试探并将下一个字符发送给终端，在此种时刻就可调用此过程。但是并不是每次调用都会执行有益的操作，经常是无功而返。

8514：请阅读8499行的注释 此代码与我们在此处的说明无关。

绪(亦即，发送器状态寄存器的第7位尚未设置)，或者跟
束，则不执行任何操作。
。如果“c”为正，则该队列原先非空(如所期望的那样)……
则这是一个要发送的字符……
元素设置奇偶位之后，将“ c ”写入发送器数据缓存寄存

符插入输出队列，表示此处需要延迟。调用“ timeout ”
造一项。这种处理造成的后果是：经过“ c&0177 ”时钟
)。我们将会观察到，“ ttrstrt ”调用“ ttstart ”，然后对
理将保证：如果“ ttstart ”的另一次执行是以同一终端的
ttstart ”的调用将立即返回。

。

多注释，因此与其他部分相比，对此过程的解释就可以

少一些。注意，用递归调用 (8392)为制表符产生一空格字符串。在 8403和8413行也使用递归调用。

25.4.5 具有受限字符集的终端

8400：“ colp ”指向一对字符构成的字符串。如果要输出的字符与这些字符对中任一个中的第2个字符相匹配，那么该字符就被代换成一个“ \ ”，后面跟随该字符对的第2个字符。

8407：小写字母加一个常数，变换成相应的大写字母。

注意，此处的变换应与对输入中出现的相反问题的处理相比较。此处我们有一种以时间换空间的算法(不使用类似于maptab的表格，但需要8400行中对整个字符串的串行搜索)。在“ canon ”中也可采用节省空间的方法，但问题会变得相当复杂。

8414：将该字符插入输出队列。如果“ putc ”因缺少缓存空间而失败，则无需为插入的任何以后发生的延迟，或对当前打印列的更新而担心。

8423：设置“ colp ”，使其指向“ tty ”结构的“ t_col ”字符，亦即，“ *colp ”的值是刚打印列的顺序数。

8424：将对应于输出字符“ C ”的“ partab ”元素值赋予“ ctype ”。

8425：清“ C ”。

8426：取出“ ctype ”的有效位(最后6位)，用其作为“ switch ”的索引。

8428：(case 0)：是一般情况!使“ t_col ”增1。

8431：(case 1):非打印字符。此组字符包含ASCⅡ字符集中第1、3和4部分，还加上“ so ”(016)、“ si ”(017)和“ del ”(0177)。不对“ t_col ”加1。

8434：(case 2):退格。除非“ t_col ”值为0，否则使其

8439：(case 3):新行。显然，应将“ t_col ”设置为0。
个字符之前应当有的延迟时间。

对于Model 37电传打字机，这取决于打印机构移动一
值至少是0.1s(6个时钟滴答)，也可以多至((132/16)+3)/60＝

对于VT05，该延迟时间是0.1s。对于DECwriter，因为
倍速打印方式，所以此值为0。

8451：(case 4):水平制表。将“ t_flags ”的第10、11位

8453：对于“ c ”==1(或Model 37电传打字机)，计算
这种计算方式不很清晰)。如果计算结果是第4列或更小，则

8458：将“ *colp ”(亦即由colp所指向单元的值)取为下

8459：使“ *colp ”增加到8的整倍数。

8462：(case 5):垂直移动。若“ t_flags ”中的第14位
0177(或127)时钟滴答，大约2s多。

8467：(case 6)：回车。将“ t_flags ”的第12、13位值

8469：对于第1类(“ ctype==1 ”)，设置延迟时间为5个

8472：对于第2类，设置延迟时间为10个时钟滴答。

8475：设置“ *colp ”(最近一次打印的列)为0。

在结束对文件“ tty.c ”的分析之前，有两个问题值得作进一步讨论。

1. 对 ' TTLOWAT ' 的测试(8074行)

在“ klxint ”中的8074行，对是否需要唤醒因等待向终端发送输出而睡眠的进程进行测试。若字符数为0或者等于TTLOWAT，则该测试成功。

如果字符数在两者之间，那么在输出队列全空前不会执行任何唤醒操作，于是在对终端的字符输出流中非常可能会出现停顿，也就是在一段时间内没有字符传输至终端。因为实际上经常会观察到输出流中的暂时中断，这会激发人们提出一个合乎情理的问题：是否有任何方法使字符计数从大于“ TTLOWAT ”变为小于它，而不被检测到它恰好为“ TTLOWAT ”的情况(在8074行)?

非常清楚有这种可能，其原因是：每次调用“ ttstart ”都能使队列长度减 1，而只有一次这种调用后随着测试。于是，来自“ ttrstrt ”(8492)或“ ttwrite ”(8568)的对“ ttstart ”的调用可能越过此界线，于是造成一种延迟。发生这种情况的可能性很小而且有限，所以只有当输出序列相当长时，才可能观察到这种情况。

还有两种调用“ ttstart ”的情况似乎能满足上述要求。在“ ttwrite ”的(8561)行调用“ ttstart ”，此时输出队列达到其最大长度；在“ ttyinput ”的8363行调用“ ttstart ”，而在此之前调用了“ ttyoutput ”，它通常(但并不都如此)将会把一个字符加到输出队列中。

2. 不起作用的终端

当终端的最后一个特殊文件被关闭时，调用“ klclose ”(8055)，其中清“ ISOPEN ”和“ CARR_ON ”标志(8059)。但是，接收器控制状态寄存器的“ 读允许 ”位没有被清除，因此的“ ttyinput ”(8333)存放在该终端的“ 原始 ”输入队列中”标志以判断该终端是否在逻辑上接入系统。

以至塞满字符缓冲存储。仅当“ 原始 ”输入队列达到 256的内容才被丢弃。所以在 8058行之前似乎应当包括一条禁

E困难地自学最后一个余留的文件，“ mem.c ”了。作为注讨论就到此为止了。

设备驱动程序，读者可以耐心地阅读它们。确实，我们对完整，也不全面。读者在此方面还有很多事情可做。