基于节点编号的通用树状菜单设计方法与实现*

■ 合肥工业大学 彭良清

摘 要 叙述嵌入式系统人机界面的各种形式和设计的基本原则，给出数据驱动的二叉树结构菜单的实现方法。该方法仅通过数据定义和特定菜单编号原则，无须修改执行代码就可生成不同的菜单界面，可移植性很好，实现代码很简洁。

关键词 人机界面 菜单 二叉树 编号方法

1 人机界面布局的一般原则

嵌入式系统最常用的人机接口部件仍然是行列键盘和显示器件，其它如打印输出部件、手写输入、CCD输入、语音交互接口等不常使用。从外观上看，键盘界面和显示界面可以独立分块，也可以设计成相关联的整体。后者在小型专用系统中是很常见的做法。

（1）屏幕布局方式

嵌入式系统的显示部件有LED、LCD和CRT三种，其中只有使用图形LCD模块（一般称为LCM）和CRT的系统才能够显示菜单，而图形液晶模块LCM有大规模（分辨率>480行×640列，和CRT分辨率相同）和中小规模LCM之分，由于显示的面积和应用系统的需求不同，对显示的要求也不同，一些系统不需要菜单，功能完全通过不同的按键来选择，对于需要菜单的应用系统来说，菜单布局大致有以下几种类型：

① 主菜单顶天式。在这种方式中，主菜单总是停留在屏幕上，如图1、图2所示，子菜单（如果有的话）一般在选择到时才显示，位置居左；右下方的屏幕区域用于显示进一步的对话窗或最终的信息内容。该区域的显示内容根据执行不同的菜单而变化（见图2）。

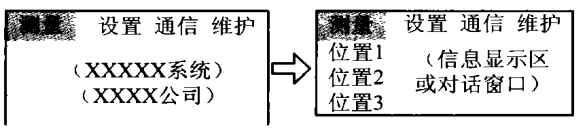


图 1 图 2

② 主菜单落地式。和第1种相反，这种菜单的主菜单在显示部件的下方，子菜单向上弹出。

③ 菜单居中式。在该方式中，菜单显示在屏幕中央，如图3所示。在下一级菜单显示时，上一级菜单显示总是消失，如图4所示。

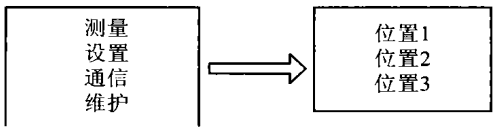


图 3 图 4

④ 图形菜单。一些产品的菜单采用图形方式显示，而不是采用文字方式；或者二者结合的方式。和文字方式相比，图形更加直观，但操作者记住图形和对应功能的对应关系需要经过训练，如果产品的使用者是经常轮换的（如工业测控仪器），则不适宜使用图形方式。对于文字理解阅读能力不足（如幼儿教育产品）的对象，图形是一个好的解决方案。

（2）菜单子项的排列顺序问题

在菜单中，各种功能的使用频率并不相同。例如，一些数据设置和维护功能很长时间才修改1次，因此，对于菜单的位置应考虑到其使用频率：常用功能子项置前（上）。另外，有时1次任务需要执行多个菜单，各个操作要按固定次序，排列的次序也必须考虑这个因素。

（3）辅助信息显示窗口

对于如日期、时间、操作指南和其它辅助信息，如果需要的话，也可安排一个位置专门显示这种信息。一般应放在屏幕下右方等不显眼的位置。

（4）布局的选择

以上介绍了几种菜单布局方式，那么，在实际系统中应根据什么选择呢？选择的因素有：屏幕大小、功能操作特点、操作习惯（例如仿照同类型装置的布局）。此外，在同一系统中应保持显示界面的一致性，避免将以上几种方式混合使用。

（5）键盘类型的选择

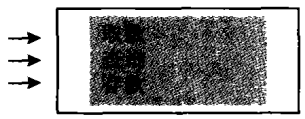
目前使用的键盘大致有：机械弹性按键、触摸键盘、导电橡胶键盘。其中触摸键盘对单件或小批量产品来讲，设计、加工均较为方便。此外，使用通用PC键盘也不失为一个好方法，只要设计简单的接口电路和解码软件即可[1]。

（6）通用键盘的界面设计

一般系统的按键有：数字键（0～9和小数点"."）、功能键、控制（Enter、ESC等）与光标键。有一些系统需要字符输入键，以便能输入中文或英文文字。对于用菜单方式选择功能的系统，往往使用光标键控制，可以不需要功能键；而对一些需要操作便捷的系统，功能键方式比菜单操作方式的速度快。应根据实际的情况选择。菜单方式的好处是一个硬件界面可以用于不同的系统，只需要修改软件（菜单的显示内容），系统修改和升级容易，无须更改键盘布局和内容。

（7）相关联的键盘和显示输出设计

现在一些嵌入式系统将菜单显示和按键作为一个整体设计。典型的系统有ATM机，一些PDA产品也是如此，如图5所示。



注：深色区域表示显示部件，箭头表示按键。

图 5

在这种设计中，菜单的选择不需要通过一个高亮度或其它形式的光标来进行，通过和显示内容相对应的按键就可以选择当前要执行的功能。

（8）树型菜单的层次深度

超过3层的菜单选择会造成操作不便，选择型菜单一般以2层为限。如果功能太多，可以通过合理的功能分类，增加菜单窗的选项数来减少层次深度。

2 嵌入式系统菜单的设计限制

和PC平台上的界面设计不同，对于很多嵌入式系统来说，由于开发平台的限制，一般只能自己编写人机界面代码。虽然菜单代码实现技术上并不困难，但如果做到代码尽量短（因为内存有限），并且可移值性又好，仍然需要仔细考虑。以下通过一种菜单树的编号方法来达到这个目的。

3 树形结构菜单的二叉树数据表示与节点编号

（1）什么是菜单

在对菜单的特征信息进行描述前，还要对菜单进行明确和严格的定义。如果仔细观察现在的各种人机界面的话，可以给菜单下定义如下：

一个菜单是包含多个固定条目内容，并同时在屏幕上显示或消失的矩形窗口；一个软件往往有多个菜单窗口，相互之间可以是相互独立的（如图5），或者呈现树型的关系，可以通过光标键、回车键、退出键或其它按键来控制菜单的显示、消失和执行。图6所示为多叉菜单树中的选择型菜单节点（白色背景菜单）和功能形菜单节点（灰色背景菜单）。

菜单是一种单向的显示输出方式。屏幕上的显

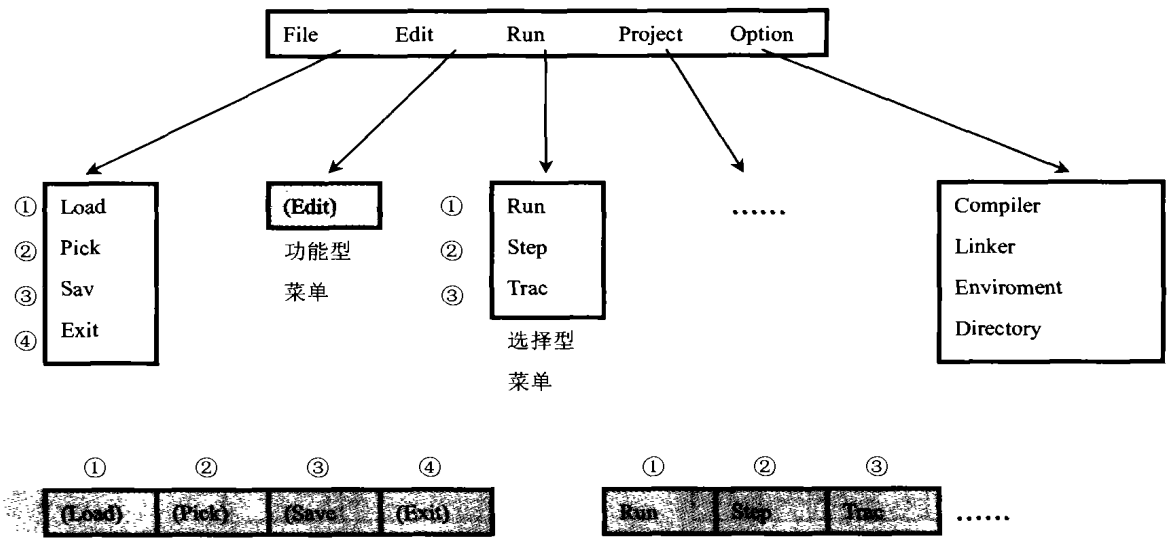


图 6

示内容除了这种方式外，还有“对话窗”（图6允许用户输入某些信息）以及单向不受控的显示窗口（如帮助显示窗口）。

（2）选择型菜单和功能型菜单

一般菜单指用于选择执行某项功能的显示窗口，其本身并不执行具体的任务，只有执行到最下层的菜单条目时，程序才会完成一个实际的应用功能。因此，对于最下级（树的叶节点）的菜单条目可称为功能型菜单。从屏幕显示情况区分，功能型菜单执行过程一般有3种：一是显示1个对话窗口，用于输入某种数据；第2种是显示某种特定的应用信息；第3种是没有任何信息显示，只完成某个应用。

（3）菜单信息的表示方法

菜单的特征信息包含两个方面，一个是菜单窗窗口的显示信息，一个是各个菜单间相互关系的信息。前者包括显示位置、字符和背景的显示颜色、菜单内容等，后者指本菜单的上级菜单和下级菜单是哪个菜单。

（4）菜单关系信息的二叉树表示方法

对于多叉树结构的菜单（见图6），为了表示菜单之间的相互关系，必须表示其所有的下级菜单节点。由于每个菜单的下级菜单个数不同，难以用统一的数据结构来描述这种相互关系，为此，将多叉树表示成二叉树形式，对菜单的下级菜单只表示其“长子”菜单，同时增加其“兄弟”菜单表示，图7所示为二叉树描述的菜单关系和菜单编号（ID）。

（5）菜单的数据结构定义

由以上描述，可以定义以下菜单的数据结构：

```
typedef struct menu {
    int ID;                          /* 菜单唯一的标识码 */
    char name[10];                   /* 菜单内容 */
    int  x0, y0, width,  high;       /* 显示位置 */
    char total,                      /* 菜单项个数 */
        bcolor, wcolor,              /* 背景颜色，字符颜色 */
        dir;                         /* 显示方向，横向或纵向 */
    struct menu *up,*down,*left,*right; /* 和其它菜单
                                           的关系信息：
                                       up: 上级菜单
                                       down: 第1下级菜单
                                       left: 同级左邻菜单
                                       rightt: 同级右邻菜单 */
    int cur; /* 当前选中的菜单条目号，在代码执行中动态
                变化（0,1,2,…,） */
} MENU;
```

（6）菜单ID号的编号方法和基于ID号的菜单树初始化实现

图7表示菜单的ID号编号算法。第1个下级菜单是上级菜单的ID号乘10，再加下级菜单在上级菜单中对应的子项编号（1，2，3…）；将主菜单的ID号定义为0，则其下级菜单编号就为11，12，13…；而对于编号11的下级编号则为111，112，113…。依次类推（见图7）。这种编号体系的好处是，只要知道当前菜单的ID号，就可以计算出其up、down、left、right等关系菜单的ID号。因此，如果先定义以下菜单变量数组：

```
MENE menus[MAX_NUM ] = {
    ..... /* 各个菜单初始化代码，略 */
};
```

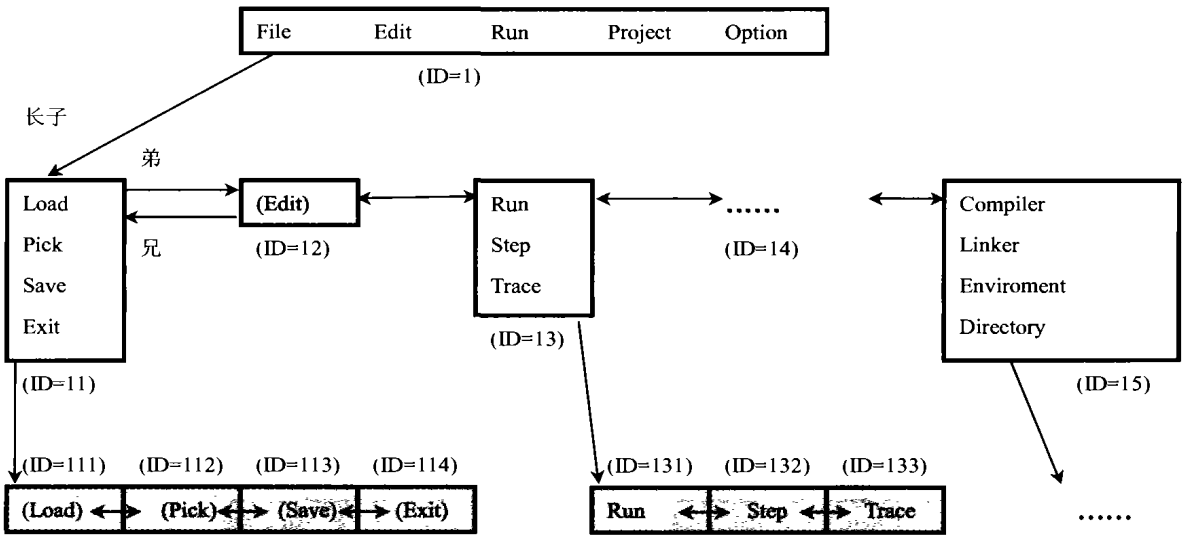


图 7

就可以按ID号对菜单树上的每个菜单节点的关系菜单信息进行赋值：

```
void initmenutree(void){
    int i;
    for( i = 0; i < MAX_NUM; i++ ) {
        menus[i].up = getpmenumfromID( menus[i].ID/10 );
        menus[i].down= getpmenumfromID( menus[i].ID*10+1 );
        menus[i].left = getpmenumfromID( menus[i].ID-1 );
        menus[i].right = getpmenumfromID( menus[i].ID+1 );
    }
}
```

而其中的函数getpmenufromID()的作用是根据ID号查找菜单结构变量：

```
MENU *getpmenumfromID( int id ){
int i;
    for( i=0; i < MAX_NUM; i++ )
        if ( menus[i].ID == id )
            return( &menus[i] );
    return(NULL);
}
```

可见，代码很简单。

4 菜单节点的选择控制实现

对于使用特定功能键方式来选择菜单的方法无须多述，这里只说明如何通过光标键和控制键来选择树型的菜单。假设当前菜单为“pmenu”，则应对按键响应如下：

ESC —— 回到其上级（up）菜单；

ENTER —— 根据当前菜单的光标位置i，进入其第i个下级菜单；

UP —— 本菜单窗中的光标条上移1个；

DOWN —— 本菜单窗中的光标条下移1个；

LEFT —— 本菜单窗中的光标条左移1个（对横向显示菜单）；

RIGHT —— 本菜单窗中的光标条右移1个（对横向显示菜单）。

对于UP、DOWN、LEFT、RIGHT等光标键，无须转移当前菜单，只需要修改结构“MENU”的“cur”数值，同时改变激活的光标条即可；对于ESC键，应将当前菜单改为其上级菜单，即：pmenu = pmenu-up；对于ENTER键，应根据当前菜单的“cur”数值达到其第“cur+1”个下级菜单，如果cur = 0，使pmenu=pmenu->down即可，其它情况应先找到其第1个下级菜单（pmenu=pmenu->down），然后再重复取pmenu=pmenu->right，这样可以编写一个函数来完成所有选择型菜单的显示：

```
/*  二叉树结构菜单的显示控制代码 */
MENU *menuselect(MENU *pmenu, int key){
    int cur,i;
    switch(key) {
        case RIGHT_KEY:
            if (pmenu->dir==YDIR)break;
            delmenucur(pmenu, pmenu->cur++);
            if ( pmenu->cur == pmenu->total ) pmenu->cur=0;
            menucurdisplay(pmenu, pmenu->cur);
            break;
        case LEFT_KEY:
            if (pmenu->dir==YDIR)break;
            delmenucur(pmenu, pmenu->cur--);
            if ( pmenu->cur < 0 ) pmenu->cur=pmenu->total-1;
            menucurdisplay(pmenu, pmenu->cur);
            break;
        case UP_KEY:
            if (pmenu->dir==XDIR) break;
            delmenucur(pmenu, pmenu->cur--);
            if ( pmenu->cur < 0 ) pmenu->cur=pmenu->total-1;
            menucurdisplay(pmenu, pmenu->cur);
            break;
        case DOWN_KEY:
            if (pmenu->dir==XDIR)break;
            delmenucur(pmenu, pmenu->cur++);
            if ( pmenu->cur == pmenu->total ) pmenu->cur=0;
            menucurdisplay(pmenu, pmenu->cur);
            break;
        case ENTER_KEY:
            cur=pmenu->cur; pmenu=pmenu->down;
            for(i=0; i<cur; i++) pmenu=pmenu->right;
            if( pmenu->total>1 ) {
                pmenu->cur = 0; menuwindisplay(pmenu);
                menucurdisplay(pmenu, pmenu->cur);
            };
            break;
        case ESC_KEY:
            if( pmenu->up!=NULL) {
                delmenuwin(pmenu);  pmenu=pmenu->up;
            };
        break;
        default: ;
    };
    return(pmenu);
}
```

5 功能型菜单的执行

对于最低层的功能型菜单（实际上不是菜单，本文中为了叙述统一，也将其称为菜单），由于作用各不相同，必须编写各不相同的函数。我们将功能型菜单变量的项数“total”定义为“1”来和选择型菜单相区分。因此执行功能菜单的过程可写如下程序：

```
while(1)   {
    key=getch();
    pmenu=menuselect(pmenu,key);
    if (pmenu->total> 1 )
        continue;/* 当前菜单为选择菜单,继续等待键按下*/
    switch(pmenu->ID){
        case 111: fmenu111(); break; /* 以下为功能型菜单
                                        代码调用 */
        case 112: fmenu112(); break;
        ⋮
    };
};
```

但这种方式，如果功能型菜单很多的话，switch-case 语句会很多，代码较长，为此，定义一个包含菜单 ID 号和菜单函数指针的结构变量：

```
struct function_menu_table {
    int MenuID; void ( *pfmenu)(void);
}   FunTab[]=
{
    111,fmenu111,112,menu112, /* fmenu111,fmenu112..为
                                 功能菜单的函数名称 */
    ⋮
};
```

这样，在知道了 ID 号后，就可以通过函数指针来调用功能型菜单：

```
i=0;
do {
if(FunTab[i].MenuID==pmenu->ID)
    break;                  /* 根据ID号查找函数指针 */
    i++;
}while(1);
( FunTab[i].pfmenu) ( );   /* 执行功能型菜单 */
```

可见，代码大大缩短了，并且增加和修改功能型菜单时只需要修改变量“FunTab[ ]”，无须修改执行代码。

6 代码的移值

以上实现了 1 个菜单代码，现在，只要修改以下数据就可以构造任意内容的树型菜单，步骤如下：

① 画出菜单树，按照上述原则给每个菜单 1 个 ID 号；

② 编写 MENU menus[]结构数组，定义每个结构变量（显示位置和内容）；

③ 编写不同功能菜单函数；

④ 编写 struct function_menu_table FunTab[ ]结构数组，定义每个结构变量。

完成后编译执行，1 个菜单界面的应用软件就完成了。

附件中给出了使用这种方法来实现 Turbo C2.0 软件界面的代码，读者可以从中对此方法和实现代码的长度有个直观的了解。只要修改代码中的输出显示函数，就可以将这段代码用在不同的硬件平台上。附件见本刊网络补充版（http://www.dpj.com.cn）。ME

参考文献

1 严蔚敏. 数据结构. 北京：清华大学出版社，1992

研祥北京公司成功召开“EIP 安防智能平台应用研讨会”

近日，由中国安全防范产品行业协会和研祥智能科技股份有限公司北京分公司共同举办的“EIP 安防智能平台应用研讨会”在广州召开。这是一场旨在探讨新兴产业 EIP 针对于安全防范行业的产品以及方案应用的专题会议。会议在中国安防协会副秘书长王俊义的嘉宾致辞中开始。随后由研祥公司合作伙伴部总经理徐英建先生，向与会的人员做了关于研祥公司和合作伙伴部工作的介绍，其间徐总的发言不但深入浅出给研祥公司的定位、业务核心和秉承的价值观进行了剖析与精辟的总结，而且更以 ATM 机作为典型的 EIP(智能嵌入式平台）系统应用，给在场的部分非专业的听众以通俗的讲解。会议安排的自由提问时间，则大部分留给了与会的专业听众；所有的提问均得到了研祥公司资深产品工程师张昆的认真解答，双方还对有关的专业技术问题进行了探讨。

此次会议虽然定位于小规模的 EIP 安防平台的研讨会，但是通过各种渠道到会的人数亦有 70 多人。