UNIX 下数据录入编辑的实现

诸中杨　（绍兴人民银行）

数据录入、编辑是应用软件必不可少的。在 UNIX 系统

下，利用系统提供的 ETI 表格生成开发系统，可以实现数据的录入、编辑功能，具有全屏幕编辑的功能。

我在开发 TCP/IP 的网络通讯软件时，利用该开发系统方便地实现了数据录入、编辑功能，如图 1，下面简单介绍如何实现。

```
┌──────────────────┐
│接收单位：        │
│发送文件：        │
│接收文件：        │
│开始发送[Y]：     │
│退出系统[ESC]：   │
│提示信息：        │
└──────────────────┘
```

图　1

ETI 的表格可以分成两个域，标题域和数据录入域。表格的实现过程包括建立、驱动和释成三部分。

1.表格建立的基本函数

new-field( )，set-field-buffer( )，set-field-opts( )，new-form( )。

new-field( )的调用格式：

FIELD * new-field (nrow, nocl, firstrow, firstcol, erow, nbuf)

该函数建立一个域，int nrow，ncol 为行数、例数，int firstrow，firstcol 为域的起始位置，int nbuf 为可以存放域内容的缓冲区数量，域的内容可以存放在多个缓冲区内，int erow 为分配给域的屏幕之外的行数，函数返回建立的域指针。

set-field-buffer( )的调用格式为：

set-fieled-buffer(f, n, label)

该函数设置域的内容存放在几号缓冲区内，FIELD * f 为域，int n 为缓冲区号，char * label 为域的内容。

set-field-opts()的调用格式为：

set-field-opts(f, option)

该函数设置域的一些特性，FIELD * f 为建立的域，OPTIONSoption 为域的一些任意选项，如标题项为非激活域(~O-ACTIVE)，表明该域的内容不能修改，数据录入域为激活域(O-ACTIVE)，表明该域的内容可以被修改。

new-form()的调用格式为：

FORM * new-form(FIELD * f)，该函数最后生成所需的表格。

2.表格驱动函数：driver-form()

调用格式为：

form-driver(form, c)

表格驱动函数，自定义了一系列键映射，FORM * form 为建立的表格，int c 为键盘输入的字符，若 c 为命令如删除、光标上、下、左、右移动，则执行相应的动作，若是可显示的字符，则加以显示。函数返回值 E-OK，表示 c 是可接受的命令或可显示的字符，返回值 E-UNKNOWM-COMMAND，表示不认识的命令或字符。

3.表格显示和释放函数：post-form()，unpost-form()，free-form()，free-field()。

它们的调用格式为：

post-form(FORM * form)，unpost-form(FORM * form)，free-form(FORM * form)，free-field(FIELD * f)。

```
#include"tinfo.h"
#include"stdio.h"
#include"string.h"
#include"form.h"
#define QUIT (MAX-COMMAND+1)
#define DEL 0177
int get-request();
int my-driver(FORM * form, int c);
FIELD * make-label(int frow, int fcol, char * label);
FIELD * make-field(int frow, int fcol, int cols);
void my-initscr();
void domy-menu();
void my-exit();
char * menu[]={
"接 收 单 位",
"发 送 文 件",
"接 受 文 件",
"开始发送[Y]:",
"退出系统[ESC]:",
"提 示 信 息:"};
FORM * form;
FIELD * f[11];
char * yesno[]={"y","n",(char * )0};
main()
{
my-initscr();
domy-menu();
my-exit();
}
void my-initscr()
{
int i;
initscr();
keypad(stdscr, TRUE);
raw();
noecho();
cbreak();
for(i=0;i<=5;i++)
f[i]=make-label(6+i,25,menu[i]);
for(i=0;i<=2;i++)
f[6+i]=make-field(6+i,37,19);
f[9]=make-field(9,7,19);
/* 确定域的类型，录入的字符只能是 y 或 n */
set-field-type(f[9], TYPE-ENUM, yesno, FALSE, FALSE);
f[10]=(FIELD * )0;
form=new-form(f);
post-form(form);
refresh();
}
void domy-menu()
{
int c, done=FALSE;
while(! done){
c=get-request();
switch(form-driver(form, c)){
case E-OK:
break;
case E-UNKNOWN-COMMAND:
```

```
done = TRUE;
break;
default:
beep();
break;
}
}
}
int get-request()
{
int c;
c = getch();
switch(c){
case KEY-DOWN: return REQ-NEXT-FIELD;
case KEY-UP: return REQ-PREV-FIELD;
case KEY-LEFT: return REQ-LEFT-CHAR;
case KEY-RIGHT: return REQ-RIGHT-CHAR;
case DEL: return REQ-DEL-CHAR;
case 0xa: return REQ-NEXT-FIELD;
case 0x1b: return QUIT;
}
return c;
}
FIELD * make-label(int frow, int fcol, char * label)
{
FIELD * f;
f = new-field(1, strlen(label), frow, fcol, 0, 0);
if(f)
{ set-field-buffer(f, 0, label);
set-field-opts(f, field-opts(f)&O-ACTIVE);
}
return f;
}
FIELD * make-field(int frow, int fcol, int cols)
{
FIELD * f = new-field(1, cols, frow, fcol, 0, 0);
return f;
}
void my-exit()
{
int i = 0;
unpost-form(form); /* 撕表格 */
refresh();
free-form(form);
while(f[i])
free-field(f[i++]); /* 释放域 */
endwin();
exit(0);
}
```