# 用户使用指南

尚　琼　编写
晓　羲　审校

# 目 录


**第一章 引言**

1.1 C-scape 的特征 …… 1
1.2 硬件配置 …… 2
1.3 本手册的编排 …… 2
1.4 若干约定 …… 4

**第二章 启动**

2.1 安装 …… 6
2.2 编译和连接 …… 6
2.3 实践环节 …… 6
2.3.1 menu_Printf …… 6
2.3.2 定位 …… 9
2.3.3 颜色 …… 10
2.3.4 引用 …… 10

**第三章 进一步的研究**

3.1 C-scape 数据对象 …… 11
3.2 定制屏幕 …… 14
3.2.1 改变位置 …… 14
3.2.2 尺寸和卷滚 …… 15
3.2.3 边框和爆破 …… 15
3.2.4 加提示、确认和字段的命令 …… 16

**第四章 菜单对象**

4.1 打开菜单 …… 18
4.2 定义菜单:menu_Printf …… 18
4.2.1 定位 …… 19
4.2.2 颜色 …… 20
4.2.3 定义字段 …… 21
4.2.4 百分号替换 …… 23
4.2.5 引用 …… 24
4.2.6 制表键、换行符及字折行 …… 24
4.2.7 重复字串 …… 24
4.2.8 错误 …… 25
4.3 清菜单 …… 26
4.4 清除菜单 …… 26

**第五章 sed 对象**

5.1 打开 sed …… 28
5.2 位置、大小和卷滚 …… 28

5.3 激活 sed …… 30
5.3.1 绘制 …… 31
5.3.2 窗口 …… 31
5.3.3 激活 sed:sed_Go …… 33
5.3.4 sed 的专用函数 …… 35
5.4 颜色 …… 35
5.5 边框 …… 36
5.6 爆破 sed …… 36
5.7 阴影 …… 37
5.7.1 阴影属性 …… 37
5.8 sed 标号 …… 37
5.9 sed 光标类型 …… 38
5.10 便笺 …… 38
5.11 通用数据指针 …… 39
5.12 关闭 sed …… 39
5.13 辅助函数 …… 40
5.14 sed_Alloc …… 43
**第六章 字段**
6.1 在一个字段中移动 …… 44
6.2 检查和改变字段 …… 46
6.3 在字段之间移动 …… 47
6.4 字段坐标方格 …… 49
6.5 已命名的字段 …… 49
6.6 绘制字段 …… 50
6.7 颜色和标记字段 …… 50
6.8 保护字段 …… 51
6.9 字段数据指针 …… 52
6.10 字段 bob(基本对象) …… 52
6.11 字段变换 …… 53
**第七章 字段函数结构**
7.1 函数结构概述 …… 54
7.2 实例 …… 55
7.2.1 问题 …… 55
7.2.2 类型变换:senter 和 sexit …… 56
7.2.3 确认字段:fexit …… 60
7.2.4 接收用户输入:fkey …… 63
7.2.5 fenter 函数 …… 66
7.2.6 varsize(变量大小) …… 67
7.2.7 构成整体 …… 68

7.3 另一个示例 …… 68
7.4 程序员注意事项 …… 70
7.5 创建的可能性 …… 71
**第八章 标准字段函数**
8.1 专用键处理 …… 72
8.2 baton 的使用 …… 73
8.3 std 函数 …… 76
8.4 提示信息 …… 77
8.5 字串确认 …… 77
8.6 数值确认 …… 78
8.7 格式串 …… 79
8.8 ocountry_结构 …… 80
8.9 标准字段函数 …… 81
**第九章 高级函数**
9.1 弹射式函数 …… 86
9.1.1 pop_Menu …… 86
9.1.2 pop_Message …… 87
9.1.3 pop Text …… 88
9.1.4 pop_Prompt …… 88
9.1.5 pop_View …… 89
9.1.6 pop_Edit …… 89
9.2 字串函数 …… 90
9.3 日期和时刻函数 …… 90
**第十章 菜单系统**
10.1 嵌套菜单系统 …… 93
10.1.1 定义嵌套菜单 …… 93
10.1.2 slug_Open …… 95
10.1.3 slug_Go …… 95
10.1.4 在嵌套菜单中用户提供的函数 …… 96
10.1.5 slug_Close …… 97
10.2 框架菜单系统 …… 97
10.2.1 框架定义 …… 97
10.2.2 frame_Open …… 99
10.2.3 frame_Go …… 99
10.2.4 框架中用户提供的函数 …… 100
10.2.5 锁定一个菜单选择 …… 101
10.2.6 frame_Close …… 101

**第十一章 Bob(基本对象)**

11.1 嵌入 sed …… 102
11.2 Bob:进一步的研究…… 106
11.3 bob 家谱 …… 109
11.4 嵌入编辑器…… 110
11.5 由其它资源建立 bob(基本对象) …… 111

**第十二章 sled(卷滚列表编辑器)**

12.1 sled:一个实例…… 112
12.2 sled:进一步的研究…… 115

**第十三章 正文编辑**

13.1 建立编辑器…… 118
13.2 把正文放入正文缓冲区…… 121
13.3 光标移动…… 123
13.4 插入和删除…… 126
13.5 制表、换行和字折行 …… 128
13.6 块操作…… 130
13.7 搜索…… 133
13.8 绘制正文和刷新状态…… 137
13.9 把正文存到文件中…… 137

**第十四章 屏幕文件**

14.1 Sed 在屏幕文件中的存放 …… 139
14.2 从屏幕文件中调入 Sed …… 145

**第十五章 鼠标**

15.1 鼠标入门…… 150
15.2 字段间的移动…… 151
15.3 sed 之间的鼠标移动 …… 154
15.4 反馈式鼠标处理程序…… 157
15.5 鼠标和菜单…… 158

**第十六章 边框**

16.1 使用边框…… 163
16.2 标准边框…… 164
16.2.1 标题…… 165
16.2.2 提示…… 166
16.2.3 卷滚线和卷滚条…… 167
16.2.4 边框特性…… 167

**第十七章 求助系统**

17.1 求助文件…… 169
17.2 启动…… 170
17.3 标号…… 172

17.4 标准显示函数…… 174
17.4.1 help_View …… 174
17.4.2 help_Xref …… 175
17.5 建立新的显示函数…… 177
**第十八章 设备接口**
18.1 设备接口组…… 179
18.1.1 启动设备接口…… 179
18.2 颜色和属性图…… 181
18.3 键盘…… 183
18.3.1 kb_Idle …… 185
18.3.2 kb_Record …… 186
18.4 发声…… 186
**第十九章 高级论题**
19.1 与旧版本兼容…… 188
19.2 内存分配…… 191
19.3 C-scape 限制 …… 191
19.4 重新编译源代码…… 191
19.5 编写可移植的代码…… 191
19.6 向其它系统移植…… 192
**第二十章 错误处理**
20.1 截取错误信息…… 193
**第二十一章 一般问题**
21.1 常见问题…… 195
21.2 解决方法…… 196
21.3 问题解答…… 196
**第二十二章 词汇解释**

# 第一章 引 言

C scape 是一种用于设计 C 程序用户界面的工具，其特点是简单易学、高效灵活。更确切地说，C scape 是一个 C 程序库，用户使用这些程序可以建立和修改任何类型的正文和数据输入屏幕。用户可按原样使用 C- scape 的例行程序，也可随意地修改这些程序。我们设计 C scape 的宗旨是，对于新的和熟练的程序员来说，他们都会觉得它十分有用。本手册讲述如何使用 C scape，并详细描述了它的组成。

C scape 是易于使用的。用户可以使用类似于标准 printf 函数的程序来快速地设计屏幕。另外，它还提供了一个高级函数集合来建立弹射式信息与菜单系统。

C-scape 是高效的。用户可将几个复杂的功能加到用户程序中，这些功能包括开窗口、图形支持、上下文相关求助、可自动折行的正文编辑、滚卷列表、及数据确认等。

C-scape 是灵活的。用户可以修改它的任一部分，以适应其特定应用程序的需求。用户可以建立自己的字段类型、有效性验证程序、正文编辑器、求助屏幕及菜单系统。

## 1.1 C-scape 的特征

C-scape 的主要特征包括：

- 用于屏幕设计的、高效的定义语言，该语言是基于 printf 的；
- 窗口管理程序，该程序自动地控制弹射式窗口的外观与定位，它决定输出应发送到哪个窗口，可以正文和图形两种方式工作。
- 定义字段，并可将函数连接到这些字段。这些字段函数确认字段内容或规定字段内容的范围，并给字段赋个性特征。
- 字段函数具有下面三种功能，确定一个字段可显示的数据类型，确定如何输入该数据，并确定如何确认该数据。C-scape 具有许多不同的字段函数，并为用户提供了建立自己的字段函数所需的所有例行程序。
- 用于正文编辑的例行程序系列。这些例行程序可以建立从弹射式注释填充到完整的正文编辑器的任何部件。
- 复杂的屏幕边框，它们的范围从简单的方框到具有卷动条和信息提示的有标题的边框，信息提示是一种边框类型，它差不多具有所有的布局与操作。
- 一种高效的上下文相关的求助系统，需要时它让用户显示求助信息。每个屏幕或字段可以有它本身的求助信息。用户可以很容易地建立显示求助信息的功能函数，以便为任一程序提供一个概述。
- 高级例行程序，它们是由低级 C-scape 例行程序建立起来的。这些例行程序包含多种菜单系统，例如下拉菜单、上托菜单、嵌套菜单及诸如弹射式提示与信息的弹射框例行程序。

作为软件开发的一种更先进的辅助工具，C-scape 可与 Look&Feel 的 Screen Designer 一起使用。使用 Look&Feel，用户可建立屏幕与交互菜单，并且可将它们转为与 C 兼容的代码。这样便可免除字段定义，字符定位及颜色设置的屏幕特性输入代码了。

Look&Feel 也可以建立和编辑屏幕图象文件。运行时，一个程序可以从一个屏幕文件加载或执行它的屏幕与菜单。这便减小了可执行文件的大小，不必重新编译一个程序，便可改

变其外观。

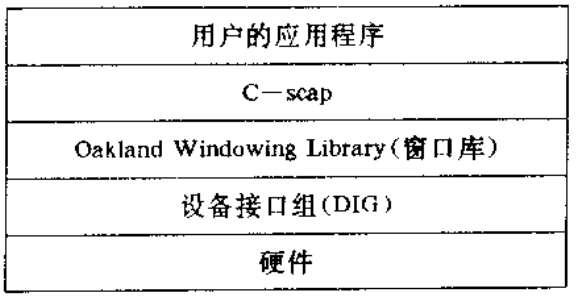
Look&Feel 可以读入用 Dan Bricklin 的 Demo Program 定义的 ASCII 正文文件及图片。

若要进一步了解 Look&Feel，参见《Look&Feel 使用手册》。

C-scape 可用于多种不同的应用程序之中。目前，已用它建立了多种应用程序，如数据库管理程序、通信程序、屏幕设计辅助程序，棋盘式分析表，嵌入过程应用程序、格式布局工具、安装辅助程序、表演程序、地址列表处理程序、数据输入工具、源代码调试程序及其它一些应用程序。

## 1.2 硬件配置

让我们看一下，在用户应用程序与物理硬件之间，C scape 位于哪个层次上。用户使用 C-scape 建立其应用程序，而 C-scape 被建于 Oakland Windowing Library(OWL)之上：




OWL 处理 C-scape 与硬件之间的通讯(通过 DIG)。它也管理窗口的建立与使用。由于所有的 Oakland C 工具都被建于 OWL 上，所以，它们可以使用以高效方式共享显示器的通用代码。

OWL 把依赖于硬件的函数都放在 DIG 中。这有助于保证 Oakland C 工具建立起来的程序的移植性。DOS 的 C-scape 具有两个标准的设备接口结构：一个与显示 RAM 直接通信，一个使用 BIOS 调用。

使所有的 C-scape 组成部分一起工作。C-scape 是面向对象的，并且在其实现中是具有一致性的。这便使得它易于学习、修改与维护。由于字段、边框、及硬件接口都使用可替换的函数指针，因而对 C-scape 任一部分操作进行修改都很方便。

## 1.3 本手册的编排

C-scape 的文档分为两卷。本书是第一卷：《C-scape 用户手册》，它从概念上对这个库进行概括性地论述。

读者在开始工作之前，不必读完整个手册。读者可以基本理解 C—scape 时，就可以用 C-scape写程序。需要进一步了解某个具体特征或某个具体程序时，再查看本手册。

C-scape 磁盘有几个实例程序，这些程序有助于用户理解 C—scape 的功能。参见 1 号盘上的 read. me 文件，该文件给出了实例程序清单。

在第 2 和第 3 章，讨论有关屏幕建立的基本问题。

如果读者需要进一步了解 C-scape 的内部工作机制，阅读第 4 到第 8 章。

本手册的其余部分对 C-scape 高级函数和其它功能进行讨论。

下面较为详细地介绍一下本手册的内容：

第二章　启动

这一章介绍如何安装、连接和编译 C-scape。同时，它也包含一个建立 C-scape 屏幕的实践环节。

第三章　进一步的研究

本章更加详细地讨论实践环节，它首先研究一下 C-scape 的数据对象。同时，本章也对定制屏幕和使用高级函数进行简单地讨论。

第四章　菜单对象

本章论述菜单对象，并且介绍如何建立菜单对象。建立菜单对象是程序员打开并且定义菜单的过程。本章解释 menu_Printf 命令。

第五章　Sed 屏幕编辑器

本章介绍 Sed(屏幕编辑器)的使用和属性。更确切地说，本章介绍如何打开 Sed，如何定制 Sed 和使用 Sed 的基本功能。

第六章　字段

本章介绍如何在字段间进行移动，同时也介绍如何使用和修改字段

第七章　字段函数结构

本章考察每一字段函数的结构。并结合一个实例，介绍字段函数的特定用途。

第八章　标准字段函数

本章讨论字段函数的多种特征。这些特征包括专用关键字函数，类数据指针(及它们的使用)，和棒(baton)。对棒的讨论不仅限于字段函数。

第九章　高级函数

本章讨论几个 C-scape 高级函数，这些函数建于其它的 C-scape 例行程序之上。高级函数包括弹射式函数、字串函数、和数据与时间函数。本章也对这些函数的一些使用方法进行讨论。

第十章　菜单系统

本章讨论几个 C-scape 的菜单系统和对这些菜单系统的使用。

第十一章　Bob(基本对象)

本章介绍基本对象。在这一章中，描述如何在其它的 sed 中使用 bob 来嵌入对象(例如 Sed)。同时，本章也讨论如何将多个对象嵌入 Sed 中，和嵌入功能的多种使用方法。

第十二章　Sled(卷滚列表编辑器)

本章讨论 Sled 对象。Sled 是特殊类型的 Sed，它拥有一个尺寸可变的表。

第十三章　正文编辑

本章介绍 C-scape 的正文编辑能力，同时解释用户应该怎样建立自己的正文编辑程序。

第十四章　屏幕文件

本章讨论 C-scape 怎样才能使用屏幕文件，这些屏幕文件是用户用 Look&Feel 屏幕设计程序建立的。

第十五章　鼠标

本章讨论 C-scape 鼠标处理程序，如何讨论在用户的应用程序中使用这些处理程序，并且对其中的理论进行介绍。

第十六章　边框

本章具体地讨论 C-scape 边框函数。它对 5.5 节介绍的内容进行扩充。

第十七章　Help(求助)系统

本章详细地讨论 C-scape 求助系统。解释如何建立一个文件，以保存用户的求助信息，同时也解释如何将求助信息与用户屏幕联系起来。

第十八章　设备接口

本章介绍 C-scape 提供的一些例行程序，使用这些程序来初启化显示器、颜色、键盘和扬声器。

第十九章　高级论题

本章考虑几个问题：从 C-scape 2.0 版升级、C-scape 的限制；C—scape 对外部世界的论述(存储器分配、代码的移植、源代码的重新编译)。

第二十章　错误处理

本章介绍 C-scape 如何处理错误和打印错误信息。

第二十一章　一般问题

本章解答 C-scape 用户常见的问题，帮助用户尽快地排除使用 C-scape 过程中遇到的困难。

第二十二章　术语

本章定义了 C-scape 的常用术语。

本套丛书的第二卷是 C-scape 函数调用。该卷包含所有 C-scape 例行程序调用和所有标准的字段函数调用，同时还包含一个 C—scape 错误信息表。

附录 A　函数调用

本附录，对每个 C-scape 函数进行解释，并介绍如何使用函数。

附录 B　字段函数调用

本附录对每个标准字段函数进行了完整的描述。

附录 C　错误信息

本章给出一个错误信息表，并且对每个错误信息进行解释。

如果需要更详细地了解这部分内容，请参阅几本 oakland 出版的书籍：

| | |
|---|---|
| 《Lock&Feel 用户手册》 | 这本书描述了 Look&Feel 屏幕设计程序的操作。 |
| 《Oakland 高级程序设计指南》 | 本书详细地讨论了 Oakland Windowing Libary(Oakland 窗口库) |
| 《C scape 程序设计实例》 | 本书提供了几个使用 C-scape 的实例，并且对每个实例进行了详细的解释。 |

## 1.4　若干约定

本手册使用下面几个约定：

术语"显示器"描述真正的硬件显示设备(终端、监视器、CRT)。

术语“屏幕”描述用于某一特定用途的显示器的特性集(“求助”屏幕,“数据输入”屏幕)。

“菜单”一词系指一个 C-scape 数据对象(菜单对象),同时也意谓着向用户提供选择(“123 菜单”)。

在函数调用和 C-scape 源代码中,使用下面两种数据类型:

VOID * 是用于 ANSI C 编译程序的 #define 定义为 vaid *,对应于旧编译程序,则 #defined 为 char*。每当需要通用数据指针时,就使用这种数据类型。要正常使用 VOID,C-scape 就使用这种数据类型。参见 19.5 节,这一节更详细地介绍了 VOID *。

SIZE_T 是用于 ANSI C 编译程序的 #defined 为 size_t,对应于旧编译程序,则定义为 unsigned int。

在源代码实例中,/*...*/表示未列出的代码段。

# 第二章 启 动

本章介绍如何安装 C-scape，同时还描述了 C-scape 盘上的文件。本章讲述了一个实践环节，它让用户熟悉 C-scape 的基本操作，以便让用户开始设计自己的屏幕与菜单。在使用该产品之前，用户一定要阅读 1 号磁盘上的 read. me 文件，同时也要阅读本手册前面的 C—scape License Agreement(许可证)。

## 2.1 安 装

存放 C-scape 需要许多张磁盘。1 号磁盘中的 read. me 文件将给出有关磁盘内容的列表，同时也给出本手册没有的其它信息。

在开始使用 C-scape 之前，用户得把磁盘中的包含文件(. h)和库文件(. lib)拷备到一个地方，在这个地方编译程序和连接程序可以找到这些文件。这些文件，连同代码实例通常以归档格式放在分布磁盘上。阅读 1 号盘上的 read. me 文件，该文件介绍如何取出和安装这些 C—scape 文件。

通常约定，将库文件和包含文件放在"\cscape"目录中。

## 2.2 编译和连接

要编译调用 C-scape 函数的文件，用户得采用如下步骤：

(1)INCLUDE FILES：包含文件 cscape. h。

(2)NAME LENGTH：如果可以应用长名，用户在其编译程序中必须使用长名选择。

(3)OTHER OPTIONS：以标准的，"vanilla"方式对 C-scape 库进行编译。使用外来编译选择，例如，组装式结构，会有困难。

对于用户给定的程序来说，当然可能需要其它的选择和其它的包含文件。1 号盘上的 read. me 文件给出了编译与连接命令完整的清单。

根据编译程序和存储器的型号来为 DOS 的 C-scape 库命名，防止错误连接。

## 2.3 实践环节

本节向用户介绍在开始设计有用的屏幕时，需要了解的内容。这些内容将有助于用户尽快地开始设计屏幕和菜单。正因如此，并不是对每个概念进行详细的介绍。如果读者遇到问题，或者希望更详细地了解某部分的内容，请看目录，找到相应的章节。下面的示例程序和相应的解释描述了 menu_Printf 的语法，该语句是 C-scape 的中心语句。

### 2.3.1 menu_Printf

下面是等同于"hello, world"程序的 C-scape 代码：

```
#include (stdio. h)
#include (cscape. h) / * must be in all C-scape applications * /

void main(
```

```
    menu_type menu;
    sed_type sed;
    char phone[11];

    phone[0]='\0';

    /* Initialize the hardware interface */
    disp_Init(def_ModeText,NULL);

    menu=menu_Open();
    menu_Printf(menu,"phone Number:@f[(###)###_####]",
                phone,&string_funcs);
    menu_Flush(menu);

    sed=sed_Open(menu);
    sed_Repaint(sed);
    sed_Go(sed);

      /* shut down the hardware interface */
    disp_Close();

    printf("\nphone=is\n",phone);
}
```

上面的代码是构成一个基本输入编辑屏幕所需的所有代码。上面的程序提示用户输入数据、从用户那儿获取输入(让用户用光标键进行编辑)、并将输入结果存于一个变量。所有的屏幕都是由这个基核派生而得的,用户可扩展这个基核,以包含更多的字段。

这段代码完成一系列操作。首先,通过调用 disp_Init,来启动 C-scape 设备接口和显示硬件。用户只有完成了这一步骤,才能使用 C-scape 例行程序。接下来,用一个"菜单"对象来定义屏幕的轮廓。菜单对象如同一幅兰图,勾画屏幕的结构,但并不真正地将它放到操作中。使用菜单,建立一个屏幕或一个"sed"(屏幕编辑器)对象(以后将更详细地讨论菜单和屏幕编辑器)。将 sed 显示于显示器上,并且激活它。用户输入电话号码之后,便消除菜单和 sed 对象,最后,disp-Close 关闭设备接口。在一个程序结束之前,用户必须调用这个例行程序。

用户可以输入上面的程序,也可以使用 sample.c 提供的文件。类似地,用户可以编译和连接本程序,也可以使用 sample.exe,sample.exe,它们位于实例盘上(仅有 C-scape 的 DOS 版本包含 sample.exe)。

sample.exe 可能要比用户期望的大些。原因是它包含几个例行程序,这几个程序来自于 C-scape 库和标准 C 运行库。当用户将其它的屏幕加到一个程序中时,它的可执行程序的长

度将增加，这样才能定义这些新的屏幕。

sample.exe 将下面的菜单放在该屏幕上：

Phone Number:(　　)

该菜单有两种数据类型：正文缓冲器和字段数组。正文缓冲器含有简单的正文，它是该屏幕的一部分。字段数组包含屏幕的字段。

一个字段有两种不同的位置：可写的和不可写的实例。在上面的电话实例中，括号和破折号是不可写的位置；空格（在源代码中，用"#"标记）是可写的位置。如果用户在菜单内移动光标，将会看到光标能够跳到拥有数字的位置，而忽略括号和破折号。

在这个实例中，正文缓冲区含有"Phone Number"；定义菜单的这部分是不可写的。它通常包含帮助用户输入数据的信息。

现在回顾一下，已介绍过的术语：

| | |
|---|---|
| 字段 | 整个的数据输入区域 |
| 可写位置 | 用户可在这样的位置上写字符。在实例中，可写位置是用户输入电话号码数字的位置。 |
| 不可写位置 | 在字段中不可以写字符的位置。在实例中，括号、破折号和前缀与电话号码间的空格是不可写位置。 |
| 正文 | 位于字段之外的字符。在实例中，字串"Phone Number"是正文。 |

在 menu_Printf 行中，几乎完成了格式化屏幕所需的所有工作：

```
menu_Printf(menu,"Phone Number:@f[(###)###_#####]",
        phone,&string_funcs);
```

menu_Printf 定义了正文，并设置了字段，一般正文，例如"Phone Number"，按原样输出，"@"是一种专用格式字符，它与 C 的 Printf 中的"%"很相似。这里，"@f"开始一个字段定义，这与在 C 的 Printf 中，"%"定义一个十进制串相同。

在方括号中的正文被称为"字段说明"；它定义了该字段的外观。"#"字符标明可写位置，其它字符标明是不可写的字符。

每个字段具有两个参数。第一个用来存储用户输入的变量，第二个是指向"函数结构"的指针，它确定该字段的一般行为。

在这个实例中，字段变量是 phone，一个字串。函数结构是 string－funcs，它表明该字段是字串类型。

变量 phone 必须足够大，以容纳记录中的所有字符（加上结束符'\0'）。在本例中，phone 具有 11 个字符（11 代表字段说明中有 10 个'#'，再加上一个结束符'\0'）。如同多数 C 函数一样，C－scape 不能判定是否有足够的空间来容纳一个变量。如果用户不能为字串变量提供足够的空间，那么，每当输入的数据长于提供的空间，字符就会被写出界地址。

与 Printf 相同，用户可按任一顺序把普通正文和字段定义混合起来。在一行中，可有任意个字段定义。例如，下面给出一个 menu_Printf 语句，它检索一个名字和一个电话号码：

```
menu_Printf(menu,
    "Name:@f[##########] Phone:
```

```
                @f[(###)###_#####]",
    name,&string_funcs,phone,&string_funcs);
```

注意现在跟着控制串有四个变元——两个变元代表字段。如同 Printf menu_Printf 可有任意个变元。用户也可以将一个 menu_Printf 分散为几个。下面给出两个 menu_Printf，这两个 menu_Printf 合起来与上面单个 menu_Printf 的效果相同：

```
menu_Printf(menu,"Name:@f[##############]",
            name,&string_funcs);
menu_Printf(menu,"Phone:@f[(###)###_#####]",
            phone,&string_funcs);
```

换行符("\n")将在下一行的开始继续输出。如果用户想在不同行中进行输入，就可写出如下的语句：

```
menu_Printf(menu,"Name :@f[##############]:\n",name,
  &string_funcs);
menu_Printf(menu,"Phone :@f[(###)###_#####]:",phone,
  &string_funcs);
```

如同在 Printf 中那样，用户可以在 menu_Printf 格式串中的任意位置使用百分号替换符。可以把前面的实例重新写为：

```
menu_Printf(menu,"Name :@f[%s]:\n",name,&string_funcs,
  "##############");
menu_Printf(menu,"%s@f[(###)###_#####]:","phone:",phone,
  &string_funcs);
```

注意用户传递变元的顺序依赖于 menu_Printf 读专用格式字符的顺序：每当发现一个"%"时，便从表中读入一个变元，每当发现一个"@f"时，便从表中读入两个变元，这两个变元一个变量，一个是字段函数。

如果用户希望字段使用变量，而不使用字串，传入一个不同的变量类型和函数结构便可。例如，要输入某人的年龄，使用下面的 menu_Printf 语句：

```
menu_Printf(menu,"Age:@f[###]",&sge,&int_funcs);
```

同使用 C 的扫描程序一样，用户必须传入一个指针，该指针指向存储这些数据的区域。对于 char、int、long 和 double 这几种 C 类型，有为这几种类型定义的字段函数结构。它们被称为 char_funcs、int_funcs、long_funcs 和 double_funcs(见 8.9 节，这一节给出了可应用的字段函数的完整清单)。

### 2.3.2 定位

用户可以使用"@P[row,col]"，直接确定字符在屏幕中的位置，这里 row 是行号，col 是列号。位置[0,0]是显示器的左上角。下面给出一个屏幕，在这个屏幕中，字段名位于字段之上：

```
menu_Printf(menu,"@p[0,0]Name @p[0,40]Phone");
```

```
menu_Printf(menu,"@p[1,0]@f[##########]",name,
  &string_funcs);

menu_Printf(menu,"@p[1,40]@f[(###)###_####]",phone,
  &string_funcs);
```

### 2.3.3 颜色

"@a"命令改变书写菜单的颜色特性。它仅有一个参数，该参数代表新的打印颜色。这个参数是新颜色的特性值。要转换打印颜色以反相显示，使用下面的 menu_Prntf 语句：

```
menu_Printf(menu,"Before:Normal @a[0x70] After:Reverse");
```

特性参数是代表特性值的串。以 0x 打头的数字被解释为十六进制数。其它的数字都被解释为十进制数：

```
menu_Printf(menu,"Before:Normal @a[112] After:Reverse");
```

用户也可以使用"%"替换符来代表一个颜色属性参数，因此用户可以使用自己的#define或逻辑属性来代替颜色。下面看一看"%d"的使用：

```
menu_Printf(menu,"Before:Normal @a[%d] After:Reverse,0x07");
```

原来的"@c"命令依然存在，它设置菜单颜色，以便原来的程序仍会正常工作。新的应用程序应该使用@a 语法。

### 2.3.4 引用

怎样才能使一个屏幕含有标号"@"，或者怎样才能使一个字段含有方括号呢?要打印菜单中的专用控制字符，使用标记"@"来引用这些字符。例如，两上"@"连在一起——"@@"将打印出一个单个的"@"。因此语句：

```
menu_Printf(menu,"50 pcs @@ $2.00");
```

将生成这样的一个字串：

50 pcs @ $2.00

并且下面的字段定义，

```
menu_Printf(menu,"Date:@f[[##/##/##@]]",date,&data-funcs);
```

将生成如下的字段：

Date:[/ /]

注意，需要括起字段中的字符"]"，以便使它与结束符"]"相分离。然而，不需要括起字符"["。

现在用户可以设计 C-scape 屏幕了，这些屏幕可以获取来自于用户的任何类型的输入数据。需要更详细地了解 menu_Printf，参见 4.2 节。

# 第三章　进一步的研究

C-scape 是一种面向对象的工具。不同的数据对象包含并且控制它的每一个组成成分。对于每一个数据对象来说，都有一组函数，用户使用这些函数可以建立该对象、观察该对象的内容，修改该对象的内容、和消除该对象。本章简要地介绍数据对象，同时通过一个实例初步了解它们的用法，该实例介绍如何使用 C-scape 来建立一个数据输入屏幕。后面几章将更加详细地描述 C-scape 的不同组成成份。

## 3.1　C-scape 数据对象

最重要的 C-scape 数据对象是菜单、字段和 sed(屏幕编辑器)对象。

菜单对象控制屏幕的结构和格式。它的作用如同一幅蓝图，让一般的正文和被称为字段的专用数据输入区域显示在屏幕上。

下面给出一个实例：

PHONE NUMBER:(＃＃＃)＃＃＃_＃＃＃＃

在这个例子中，"PHONE NUMBER"是一般正文，除此之外是一个字段的一部分。字段由两种不同类型的位置组成——可写位置和不可写位置。在上面的例子中，括号和破折号是不可写的字段位置，面用字符"＃"来标识，将要保存输入数据的位置是可写的字段位置。

一个"拼接"是包含该字段整个内容的字符串，它包含可写位置和不可写位置。在上面的例子中，如果一个用户输入了一个区域代码和一个数字，那么拼接是：

"(617)491—7311"

然面，一个记录仅引用可写位置；即，输入真正的数据。因此，对于上面的例子来说，记录应是：

"6174917311"

每个字段有两个以它为变元的数据对象。一个是变量，该变量将要保存与字段相关的数据。另一个是指针，该指针指向字段函数。字段函数管理字段的操作。该函数必须处理击键、在字段内编辑数据、并且完成从约束于该字段的变量到显示所需的字符串之间的转换。"正文缓冲区"包含不在字段中的所有正文。在上面的实例中，正文缓冲器将包含：

"PHONE NUMBER:"

菜单对象中的数据包括关于正文和字段位置的信息。它也记录已定义的字段类型。这一信息决定屏幕的结构或格式。

sed 对象(屏幕编辑器)接受一个菜单模板，并给出它的特征。sed 的作用如同一个框架，通过它可以看到及操纵菜单对象；同时 sed 也将菜单放到显示器上，并且从显示器获取信息。任一时刻，屏幕可有多个 sed，并且可以移动 sed，也可以改变它的大小。在 sed 的尺寸小于菜单尺寸的情况下，sed 将自动卷滚。

在 sed 对象中，大多数数据由它在显示器上的位置、它的颜色、当前字段号、和字段中当前位置这样的信息构成，这一信息是有关显示一个屏幕的信息。

现在来考虑另一个实例，一个简单的数据输入屏幕。从用户程序的另一部分调用这个数据输入屏幕，并且返回用户输入的数据。本实例要求输入的数据包括顾客的名字、城市名，及