VAX / VMS 培训教材

C 语言培训教材

2000 系列资料出版中心
沈阳计算所信息技术开发公司

前　言

本手册将有关 VAX C 编程语言的参考信息及在 VAX／VMS 操作系统下开发和调试 VAX C 程序的有关信息综合起来．也包括有关从 VAX／VMS 中将 C 程序移植到其它操作系统中或自其它操作系统中向 VAX／VMS 移植 C 程序的有关信息，同时还包括 VAX C 与其它语言工具的不同点．

读者对象

本手册的读者对象是需要学习 VAX C 的有经验的编程人员，要了解 VAX C 与其它语言不同点的用户及需要参考有关信息的有经验的 VAX C 用户．读者应熟悉某一种高级语言．VAX C 的新用户应先读第一章，VAX C 语言起步，读者应对 DIGITAL Command Languagc(DCL)有一定的了解，那些还不太了解或要参考有关 DCL 信息的读者，可参阅第 9 章，程序开发导论．

本手册的结构

本手册有三个部分:

- 第一部分．VAX C编程语言提供包括语言指导及VAX C词汇的VAX C结构的有关信息．
- 第二部分．使用VAX／VMS进行程序开发，提供有关DIGITAL Command Languagc 及其它语言解释器，建立及维护源程序，调试程序，使用VAX／VMS 记录管理服务子程序(RMS)，及在混合语言环境中编程的信息．
- 第三部分．VAX C运行时间库(RTL)提供有关在不同的操作系统间进行移植的信息，以及函数和宏的类描述．

本手册还有 6 个附录:

——附录A是VAX C运行时间库及其它C工具的运行时间库，它提供了VAX C RTL 函数及宏的构成以及其它 C 工具的相应函数．

——附录B是VAX C的定义模块，它提供了用于在源码中包含定义及VAX C RTL宏的.H 文件的描述．

——附录C是VAX C运行时间模块及入口点，它提供了用于此工具的VAX运行时间模块及 VAX C 模块的描述．

——附录D是VAX C编译器信息，它提供一个由出错信息及为避免错误所采取的可能措施的字母顺序目录．

——附录E是VAX C编译器的列表文件格式，它提供了通过使用在第十一章—编译，连接和运行 VAX C 程序—中介绍的命令行限定词而产生的列表文件的例子．

——附录F是VAX／VMS符号调试器的命令句法，它提供了所有调试命令及限定词的句法．

相关文献

在使用 VAX C 进行编程时，会发现下列文献会很有用：

- Installing VAX C ——是为安装 VAX C 软件的系统程序员准备的。
- VAX C Language Summary ——是为快速查找VAX C句法描述的程序员准备的。
- VAX／VMS Master Index ——是为使用VAX机器体系及VAX／VMS系统服务子程序的程序员准备的。这个索引开列有关访问 VAX／VMS 的单个主题的手册名单。
- The C Programming Language ——是为那些需要此第一章——开始使用VAX C——更多更详细的参考资料的读者准备的。如在 The C Programming Language 中说明的那样，VAX C 具有许多特点，而且对 C 语言有许多改进的地方，因而，Programming in VAX C 应作为 VAX C 的全部描述的参考书使用。

目　录

第一部分　C 编程语言

第二部分 使用 VAX / VMS 进行程序开发

第三部分　VAX C 语言运行时间库

第一部分　VAX C 编程语言

第一部论述了有关 VAX C 语言特点的参考资料.

第一章　VAX C 语言起步

本章是为希望了解一种不熟悉语言的基本特征的有经验的程序员设计的. 此点是通过详细例子与简短的注解实现的. 本章包括了整个手册所有各章要点的说明, 第一部分各章描述了 VAX C 程序语言要点. 第一部分还讨论了如下问题:

- 程序结构
- 语句
- 表达式与操作符
- 数据类型与说明
- 存贮类与分配
- 预处理器控制行
- VAX C 语言术语

熟悉 C 程序设计语言与 VAX / VMS 系统的程序员, 可从第二章, 程序结构开始学习.

1.1　C 程序设计语言的背景

C 语言由于它尺寸小而灵活, 由于它提供了丰实的操作符而方便了使用, 而且具有强功能的现代控制流与数据结构, 所以 C 语言是一种容易掌握的通用的程序设计语言. 最初 C 语言是在 PDP-11 机上 UNIX 操作系统下设计与实现的. C 语言的设计者就 C 语言功能指出:

"C 语言…不依赖于任何一种操作系统或计算机; 尽管由于它在编写操作系统程序方面的实用性而被称为"系统程序设计语言", 但它同样也成功的被用于编写大量的数字, 文本处理及数据库程序."

象汇编语言一样, C 语言不是为适应任何一种特定的应用需求而设计的. C 语言处理和存贮数据与现代机器结构相同. 然而 C 语言不象汇编语言那样复杂, 不依赖于机器(C 语言具有极强的可移植性). 一个程序可在几种不同的机器上使用几种不同的编译程序进行编译, 且能运行它的源程序, 则称这个程序是可移植的.

尽管 C 程序语言没有 ANSI 或其它的工业标准, 但在实现功能上保持一致. 如果 C 语言是跨系统可移植的, 就必须具备这种一致性. 这是语言最希望有的特征之一. 因此当用户编译且运行程序时, 不仅 C 源程序必须可移植, 而且由语言特征所决定在众多系统上必须产生同样效果.

因为 C 语言是在 UNIX 系统环境下研制的，最后被用于改写大部分 UNIX 系统程序，因此在 C 语言中许多"标准"操作方法与 UNIX 有关。例如，UNIX 系统通过一个数字文件描述符来访问文件，因此 C 语言工具通过文件描述符提供"访问"文件的功能。在 UNIX 系统环境下，用户通过一个简单的命令结构就可实现将一个程序或命令的输出做为另一个程序或命令的输入，并且能建立同步或异步子进程。无需许多限制和系统保护，就能使用操作系统的特征。

一些"标准"C 语言结构包括预处理器控制行和函数与宏指令的运行时间库。在 UNIX 系统环境下，一个预处理器在编译前对源代码中含有的预处理器控制行任务进行处理。

因为 C 语言无输入输出信息的手段，通常由运行时间库来提供这种服务。如果一个运行时间库函数产生了非 UNIX 系统环境下产生的副作用，则这一函数的可移植性就有问题了。

1.2 VAX C 程序设计语言

VAX C 程序设计语言合并了 C 语言基本特征和大多数 C 语言编译程序的特征。然而，VAX C 语言也提供了本身特有的特征，即只在 VAX／VMS 操作系统环境下，能直接而有效地工作。用户必须选择 VAX C 语言中对他编程最重要的特征：跨系统的可移植性或对 VAX／VMS 操作系统特性的高效使用。选择的特征集必有优点与缺点。

如果用户选择 VAX C 语言程序，则他的源程序具有极强的跨系统的可移植性，但在某种程度上会降低效率。为使 VAX C 语言编译程序模仿 UNIX 系统特征，且 VAX C 语言必须保持满意的可移植性，必然会在处理 VAX／VMS 系统特征时使效率降低。例如，鉴于 UNIX 系统使用称为文件描述符的结构存取文件，VAX 记录管理服务(RMS)用各种控制结构存取文件。在 VAX C 语言中，input 与 output 函数以 UNIX 系统同样方式存取文件，但编译程序实际上处理 RMS 结构，好象用户在 UNIX 系统环境下处理一样。但在某些场合中，除了程序执行慢以外，大多数 VAX／VMS 操作对用户是透明的。

VAX C 语言与其它 C 语言实现的最大区别——防碍源码可移植性区别——是由 VAX／VMS 与 UNIX 系统之间不同产生的。例如，当 VAX／VMS 系统不象在 UNIX 系统环境下那样将大量的控制权分配给用户，很难使 VAXC 建立一种环境允许程序员有很大的控制权。I／O 重定向不是 VAX／VMS 系统命令行语法部分；在 VAX／VMS 系统中建立子进程不如在 UNIX 系统那样有效；VAX／VMS 高级语言不用象在 UNIX 系统中的语言那样的方法来实现预处理。在手册内，正文部分与索引部分，标明了 VAX C 语言与其它 C 语言实现之间的差别，因此用户可广泛使用不可移植的 VAX C 语言结构。

如果用户选择用 VAX C 语言编程序使其在 VAX／VMS 系统下以有效的方式运行，在某种程度上，降低了程序移植到其它系统或从其它系统移过来的可能性。例如，在 VAX C 语言程序内，调用 VAX／VMS 系统运行时间库例程，即有此效果。

然而，也具有可移植性与有效地存取功能强的 VAX／VMS 系统环境!

一种特殊选件是用 VAX C 语言特殊结构和 DIGITAL 命令语言(DCL)(象 VAX C 语言预处理器代换与 DCL 命令行限定/STANDARD=PORTABLE)编写具有 VAX/VMS 系统环境优点的可移植代码。这些结构仅允许在 VAX/VMS 系统下运行时，执行一些代码段，而在非 VAX/VMS 系统下运行其它代码段。可移植性的完整讨论，参见第 15 章 VAX C 语言运行时间库信息。有关预处理器控制行的更多信息，参见第 7 章，预处理器控制行。有关 DCL 命令行信息，参见第 11 章编译，连接与执行 VAX C 语言程序。

1.3 编写第一个程序

尽管概念简单，但用一种不熟悉的语言编写第 1 个程序可能有失败的经历。由于计算机以有效的处理数字而著称，因此在这里第 1 个程序是将二数相加且将总和存在一个变量中。例 1-1 显示了此程序是如何实现的。

例 1-1：一个简单的 VAX C 语言加法程序

```
① /*  This program adds two numbers and places the sum in   *
   *  the variable total.                                     */

② main( )                                  /* The function name "main" */
   {                                       /* Begins function body     */
③     int total;                           /* Variable of type "int"   */
                                           /* Blank lines are allowed  */
④     total = 2 + 2;                       /* Answer placed in "total" */
   }                                       /* Ends the function body   */
```

注解：

① 字符(/*)与(*/)之间的正文是注释。注释内不能再放注释(即注释不能"嵌套")，但注释可放在任何空白处。空白是在源代码内分隔代码的空格与空白行之间的区域。在下面的章节中，定义了 VAX C 语言结构的许可空白。

② VAX C语言程序包括了用户定义的不可嵌套的外部函数。本例中，定义的函数名为"main"。在 VAX C 语言中，程序可从名为 main 函数开始执行，也可从使用"main-program"选择项的函数开始执行，或两者均可；如果用户指明的 main 函数不存在，在外部引用函数时，程序流中的第一个函数是缺省 main 函数。main 选择项是 VAX C 语言特有的，因而是不可移植的。有关 main-program 选择项的语法及使用的进一步信息，参见第 2 章程序结构。

VAX C语言函数有用参数与变元交换信息的方法。由于在主函数定义中，空括号内无参数；因此，在上例中，主函数不能通过使用参数来接收信息。

为在函数定义中指明参数，在括号内列出了参数标识符且用逗号(,)分隔开。必须在函数体开始以前说明参数。如果在函数 main 内调用函数(通常不从程序其它部分调用主函数)，函数名放在括号内的变元表中且用逗号分开，函数说明中变元的个数必须与参数相同。在上例中不存在函数调用。

函数完成由其函数体内语句所决定的任务，也许将值返回调用的表达式。主函数体用花括号({})定界。这与 PL/I 中的 DO-END 或 Pascal 中的 BEGIN-END 类似。通常，函数体含有一个或多个返回语句。一个返回语句规定了(如果有返回值的话)把什么返回给调用函数的表达式。依赖于函数的编写，可省略返回语句，省略返回语句的函数返回值无定义。如果函数无返回值，用户可将函数说明为具有 void 数据类型。有关函数的进一步信息，参见第 2 章，程序结构。

③ 本例中，变量total在主函数中说明和定义。在程序内引用变量前必须先对其说明。说明以分号(;)结束。在说明一个变量时，要指定它的数据类型。数据类型指出了所需要的存贮量与如何解释存贮对象。例如，变量 total 具有 int(整型)数据类型，其对象需要 32 位存贮空间(4 字节或 1 个长字)，VAX C 语言把 int 类型变量解释为具有正或负号(或 0)的整数。

当定义一个变量时，需指明它的存贮类，它的存贮类影响它的位置，生命期与作用域．在函数内被说明为 int 类型的变量有一个 auto(自动)的缺省存贮类．具有这种存贮类型的变量，当运行函数时获得存贮区．且当控制返回到调用函数处时释放存贮区．并非所有的存贮类均都用缺省值说明．用户可指明 VAX C 语言的存贮类并且在变量说明里将存贮类关键字放在数据类型关键字前面或后面．

当确定一个变量的作用域时，数据类型与存贮类是很重要的．有关数据类型的进一步信息，参见第 5 章，数据类型与说明．有关存贮类的进一步信息，参见第 6 章，存贮类与分配．

用于说明数据类型(如int，double)，存贮类(如auto，globalvalue)，语句(如if，goto)，运算符(如sizeof)的保留字称为关键字．关键字是预定义的标识符且不能再作说明．在用户程序中不能使用这些字来标识变量与函数．关键字必须用小写字母表示．有关 VAX C 语言关键字表，参见第 2 章，程序结构．

VAX C语言是一种区分大小写字符的语言，说明变量(如total)可用大写或小写字母混写．但如果在程序中引用变量 total，这种引用必须用小写字母．例如，如果企图以"Total"形式引用变量，则产生错误；编译程序由于该字开始使用大写字母而不识别这种变量．当引用 VAX C 语言所有标识符时，不必象上面 total 那样；语言区分大小写的所有例子在本手册中被标出来了．

④ 最后，2+2的和被存贮在变量total中．这是通过使用一个有效的VAX C语句来实现的．任何以分号(；)结束的有效的表达式都可作为一个语句．标识符 total 是一种说明了的变量；等号(=)与加号(+)是有效的 VAX C 语言运算符；加法中的数字是有效的常数．有关各种 VAX C 语言语句的进一步信息，参见第 3 章，语句．有关 VAX C 运算符的进一步信息，参见第 4 章，表达式与运算符．

1.4 输入／输出

C 语言不包括处理输入与输出(I／O)的功能．然而，所有的语言工具必须具有程序与用户通信的方法．上例中缺少通信手段是很不方便的；使用户无法知道程序是否将正确的值 4 赋给变量 total．为了查看 total 变量中是否为 4，用户可使用 VAX C 语言运行时间库(RTL)函数将变量 total 值输出到终端．

所有 C 语言编译程序用时间库函数及宏指令完成输入，输出和与特定的操作环境有关的各种任务．VAX C 语言运行时间库提供了其他 C 语言包括的许多函数与 C 语言能实现的宏指令．另外，还存在在 VAX／VMS 系统环境下直接而有效地工作的函数．

VAX C 语言 RTL 函数当解决用户程序中的外部引用时，要"访问"目标码的程序段(运行时，为可共享的映象)．所有的 RTL 目标码模块装在 SYS$LIBRARY:VAXCCURSE.OLB，SYS$LIBRARY:VAXCRTLG.OLB 与 SYS$LIBRARY:VAXCRTL.OLB 库中．在用户执行本手册任何程序实例前，用户必须以恰当的顺序定义库，以便连接程序可查寻库，从而解决对这些 VAX C 语言 RTL 函数的引用．为了确定以哪种顺序定义这些库，参见第 15 章，VAX C 语言运行时间库信息．有关库的一般信息，参见第 9 章，对程序开发的介绍．

在程序源码中 VAX C 语言的 RTL 宏引用，看来就象函数引用一样．然而，编译程序在执行过程的早期阶段用 VAX C 语言源码来取代宏引用．编译程序将 VAX C 语言 RTL 宏指令源码放在 VAX C 语言提供的.H 的定义文件内．如果系统管理员在安装时对.H 文件进行摘选．可在 SYS$LIBRARY 目录下访问所需文件，例如，在终端上可用如下命令显示 STDIO.H 文件:

```
$TYPE SYS$LIBRARY.STDIO.H RETURN
```

如果此命令有错，向系统管理员了解在安装期间对.H 文件是如何摘选的．显示或打印所有的.H 文件，看看 VAX C 语言提供的宏与定义是个好办法．

用户可将.H 定义文件放在文本库，VAX C DEF.TLB 中，此文本库装在 SYS $ LIBRARY 目录下，本手册把.H 文件看成定义模块，因此它们可象这个文本库中的模块一样访问．

有关宏指令的进一步信息，参见第 7 章，预处理器控制行．有关访问 VAX C 语言 RTL 函数的各种方法的更多信息，参见第 15 章，VAX C 语言运行时间库信息．

例 1-2 表明一个 VAX C 语言程序，通过使用 VAX C 语言 RTL 的 printf 函数可在终端上显示信息．

例 1-2: 信息的输出

```
/*  This program adds two numbers, assigns the value 4 to *
 *  variable total, and then prints the answer on the     *
 *  terminal screen.                                      */

main( )
{
   int total;

   total = 2 + 2;
                                /* Print intro string       */
❶  printf("Here is the answer: ");
   printf("%-d.", total);       /* Print the answer         */
}
```

注解:

① VAX C语言的RTL函数printf将信息写到标准输出设备上(终端显示屏幕)．第一次对RTL printf函数调用将一个字符串作为变元传递输出．第二次对 printf 调用将有特定格式的字符串与一个变量传递输出．在格式串内，百分号(%)由 total 值取代，减号(-)调整左对齐输出，字母 d 迫使变元值以十进制数表示．句号(.)立即在 total 值后输出．

此程序输出形式如下:

```
Here is the answer:4
```

如果用户予将 total 值在另一行上显示，那么串中必须有换行符．例 1-3 说明如何将信息输出到两行上．

例 1-3:使用换行符输出的程序

```
/*  This program adds two numbers, stores the sum in the  *
 *  variable total, and then prints the answer on two     *
 *  separate lines on the terminal screen.                */

main( )
{
   int total;

   total = 2 + 2;
                                /* Print intro string       */
   printf("Here is the answer...\n");
                                /* Print the answer         */
   printf("%-d.", total);
}
```

此程序输出如下:

```
Here is the answer...
4.
```

现在一个带有输出的程序已编完，然后它必须用 DIGITAL 命令语言(DCL)进行编

译，连接和运行，最后看结果．编译一个程序就是将源码转换成目标码；连接一个程序就是组织存贮区且解决外部引用(例如，对 VAX C 语言 RTL 函数的引用)；运行一个程序就是执行映象．

在 VAX／VMS 系统环境下，一个文件由文件名与文件扩展名来识别．用户可选自己比较容易识别的文件名．文件扩展的选择会影响文件的功能．例如，文件名为 ADDITION.C 是 VAX C 语言源程序的一个文件名．文件扩展名.C 是 VAX C 语言编译程序缺省的文件扩展名．如果在编译 C 语言时写文件名 ADDITION，则编译程序将寻找文件 ADDITON.C．

一但用户程序已建立了且具有恰当的名字，则程序可按如下方式编译，连接且运行：

```
$ DEFINE  LNK$LIBRARY  SYS$LIBRARY:VAXCRTL.OLB RETURN
$ CC  ADDITION.C RETURN
$ LINK  ADDITION.OBJ RETURN
$ RUN  ADDITION.EXE RETURN
Here is the answer...
4.
```

.OBJ 与.EXE 扩展分别指目标文件与映象文件的缺省文件扩展名。

用户为了在程序中使用 VAX C 语言 RTL 函数，可为连接程序定义更多的库。上面中的定义满足了执行本章内的所有例程．一旦用户定义了库，在其余的终端对话期间不必再定义它们(直到退出为止)．有关对连接程序库规范的进一步信息，参见第 9 章，对程序开发的介绍．有关源码的建立的进一步信息，参见第 10 章，建立源程序．有关编译过程的进一步信息，参见第 11 章，编译，连接与运行 VAX C 语言程序．

1.5 控制程序流

有些场合，用户在已知某种条件下必须执行 VAX C 语言的一条或多条语句，而在另一些场合，用户必须重复地在循环体内执行一条或多条 VAX C 语言语句，直到遇到某种条件为止．在 VAX C 语言中，有几种语句用于完成这些任务．这些语句有 if 语句，switch 语句，do 语句与 for 语句．有关 while 语句(直到满足条件才停止的另一种循环语句)的信息，参见第 3 章，语句．

1.5.1 if 语句

在给定的条件下执行一条或多条 VAX C 语言语句时，用户可使用 if 语句．例 1–4 表示了使用 if 语句的一个程序．

例 1–4: 使用 if 条件语句的程序例

```
/*  This program asks the user to guess a letter.  The     *
 *  program tells whether the answer's correct or          *
 *  incorrect.  The program's hard coded to accept 'a' or  *
 *  'A' as the correct letter.                             */

main( )
{
```

```
    char ch;                        /* Declare a character      */
                                    /* Ask the user to guess    */
    printf("Guess which letter I'm thinking of!\n");

❶   ch = getchar( );                 /* Get the character        */

                                    /* Correct = "a" or "A"     */
❷   if (ch == 'a' || ch == 'A')
                                    /* If correct guess         */
      printf("You're right!");
    else                            /* If incorrect guess       */
      {
        printf("You're wrong.\n");
        printf("You'll have to try again!");
      }
}
```

注解:

① VAX C语言RTL的getchar函数，从标准输出设备(终端)上取一个字符；程序暂停，等候用户键入一个字符后再按 RETURN 键．函数 getchar 只取第一个字符，而不理采其它键人的字符．

② 如果用户键人的字符是'a'或'A'，那么输出信息指出选择是正确的．如果键入其它字符，那么输出信息指出选择是不正确的．等号(==)将变量 ch 与常量'a'与'A'比较．逻辑或运算符(||)表示测试条件．如果在测试条件下执行多个语句．那么必须用花括号({})将语句括起来．用花括号括起来的语句称为一个块或复合语句．块的概念对确定变量作用域很重要的．有关块的进一步信息，参见第 2 章，程序结构．

本程序的一个输出例如下:

```
$ RUN  EXAMPLE4 RETURN
Guess which letter I'm thinking of!
B RETURN
You're wrong.
You'll have to try again!
```

1.5.2 开关(switch)语句

if 语句不是唯一的使指定的语句在给定的某种条件下执行的唯一方法．switch 语句可完成上例中 if 语句所完成的同样任务，但当需要测试许多条件时 switch 语句特别有用．switch 语句例如下:

例 1-5:使用 switch 语句有条件执行的程序例

```
/*  This program plays the same guessing game as the      *
 *  previous example except that it uses a switch         *
 *  statement.                                            */

❶ #include ctype               /* Include proper module     */

main( )
{
   char  ch;

   printf("Guess what letter I'm thinking of!\n");
   ch = getchar( );
❷  _tolower(ch);                /* Convert "ch": lowercase   */
   switch(ch)                   /* Examine "ch"              */
     {                          /* Body of switch statement  */

       case 'a' :
         printf("You're right!");
         return;

       default  :               /* Any other answer          */
         printf("You're wrong.\n");
```