UNIX／XENIX 系统下 C 语言程序的调试法

罗　升　（中国工商银行江苏省洪泽县支行）

摘要:C 语言程序的调试、检验工作量很大、且具有较强的技巧性。本文从实践中总结出一些常用的方法，介绍给同仁以供参考。

C 语言自产生以来，以其简洁的表达式，高级语言的编程形式，低级语言的操作方式，具有预处理，函数及宏调用等功能，俞来愈成为程序开发人员爱不释手的编程工具。由于 C 语言在程序开发过程中调试所占的比重很大，对较大的系统开发更是如此。如果没有一套较好的程序调试方法，则不但不能够处理编程中出现的各种问题，反而会使编程者如坠烟云，无所适从。笔者在工作实践中摸索到一些较好的 C 语言程序调试，检验方法。

任何一个程序都有这样或那样的错误，包括一些已在广泛使用的操作系统，应用程序。这些错误从概念上分为语法及语义错误两种。而语法错误一般而言都有章法可循较容易解决，而语义错误则很难发现，只有靠经验及机遇。一个 C 语言程序编写完后都必须经过调试，测验才能知道是否有错误，其一般的调试，测验方法总括起来无非是以下几种。

1.用 LINT 工具软件检查一次源程序，以求检查出 C 语言程序结构中可能存在的错误及问题，不可移值或根本没有用到的语句及变量等错误，这基本可发现 C 语言编译程序所不能检测出来的问题及错误。

2. 利用 C 语言编译处理程序 CC 编译源程序，以发现语法上的错误。

3. 用预处理程序#IFDEF....#ENDIF 语句在程序中加在怀疑的需要排除的程序代码之间。

4. 对较小的程序或程序中的多个函数使用 CTRACE 工具软件。

5.以 SDB 工具软件为主进行调试及测验。

6. 对没有 SDB 工具的 UNIX／XENIX 系统或在没有源程序的情况下使用 ADB 工具软件。

以上这几种方法编程调试者可根据程序的大小，视具体情况选择使用。对 LINT，SDB，ADB 等工具软件可参考有关 UNIX／XENIX 资料。本文就利用 C 语言本身的预处理程序＃IFDEF....＃ENDIF 语句及 Ctrace 命令工具软件谈一谈在 C 语言程序中的调试，检验技巧及有关的方法。

一、利用 C 语言的预处理程序 ＃ifdef.....＃endif 语句进行调试

预处理程序属于 C 语言编译的一部分，它能识别程序中嵌入的一些特殊命令，语句，并在分析源程序前先处理这些命令，语句，进行宏代换，文件包含以及条件编译等操作。利用 C 语言预处理程序的以上功能可以用来在程序中加入待调试的源程序部分。具体使用预处理程序的一般形式为:

＃IFDEF[需调试的源程序]＃ENDIF 只要能正确使用＃IFDEF....＃ENDIF 预处理语言功能，则可以视具体情况加入或消除怀疑有问题的程序段部分以达到调试，检验的目的。下面以一个短程序说明其具体使用方法。

程序 1:

```
￥cat sumber.c
#include＜stdio.h
＞#define SUMBER
main(argc, argv)
int argc;
char #argv[]
FILE * stderr;
{
int i, j, k, l, m, n;
lF = scsnf("%d%d%d%d%d", &i, &j, &k, &l, &m);
#ifdefSUMBER
```

```
    fprintf{stdett,"Number of integers read = %d\n",n);
    fprintf(stderr,"i = %d, j = %d, i = %d, m = %d\n"i, j,
k,l,m,);
    #endif
    printf(total = %d\n".sum(i,j,k,l,m))
    }
    int sum(i,j,k,l,m)
    int i,j,k,l,m;
    {
    return(i+j+k+l+m);
    }
```

程序 SUMBER1.C 是输入五个数,然后显示它们的和,见源程序。经过 C 语言编译程序 CC 处理之后,生成 A.OUT 可执行程序运行该程序,当输入 1,2,3,4,5 时则终端显示。

Number of integers read = 5

i = 1, j = 2, k = 3, l = 4, m = 5

Total = 15 当预处理程序变量 SUMBER 有定义时,(即有#define SUMBER);则被调试程序段(即#IFDEF 与#ENDIF 之间的语句)在 stderr 文件中显示有关信息,同其它程序语句一起编译。如去掉预处理定义变量#define SUMBER,则被调试的#ifdef 与#endif 之间的语句则不被编译。

当然这个程序相当短,似乎无需这种调试方法,但是当调试一个成百上千条语句时,只需在该调试语句前后,各插入预处理语句#ifdef 及#endif,再通过预处理变量设置#define 及可实现加入或取消被调试程序段,极为方便。

二、使用 Ctrace 命令工具软件实现对程序的调试

UNIX / XENIX 操作系统命令 Ctrace 是用以观察 C 语言程序执行情况的调试工具软件。它可以在程序中加入一些输出语句以显示或打印出对每个可执行语句所做的追踪信息。对调试者而言,可将 Ctrace 视同预处理程序来加以使用。其具体使用规则为:

$ Ctrace [参数][待调试 C 源程序]> 中间文件(扩展名为.C)对中间文件用 C 语言编译程序进行编译,即

$ CC 中间文件(扩展名为。C)可产生可执行文件 a.out,并执行之则产生有关输出信息,即执行到的每一条语句都会显示出来,而且每一条前都列出原程序及所对应的行号。据此可以调试,检验程序中存在的问题及不足。

下面以一个小程序来说明 Ctrace 具体使用方法及有关技巧。程序 expll.c 是由用户输入三个数,程序执行后输出第一个数除以第二个数及第三个数的和,见源程序 2。$ cc expll.c 经编译后产生 a.out 执行文件;执行该程序,输入 20 4 35 则结果为 40;当输入 7 0 6 时程序执行出错显示。

程序 2:

```
¥cat expll.c
1 msinargc, argv
2 int srgc;
3 char argv[]
4{
5 int a.b.c.;
6 scanf("%d¥n", procd(a, b, c, );
7 printf("%d¥n", procd(a, b, c))
8 }
9 int procd(d, e, f, )
10 int d, e, f,
11 {
12 d = die+{
13 return(d)
14 }
```

Floating exception-core dump

根据以上的程序调试例子,说明程序的缺陷,用 Ctrace 命令找出非正常结束的原因,现给出具体调试的方法。

1. $ Ctrace expll.c > exl.c 产生跟踪调试的中间文件;

2. $ cc exl.c 编译中间文件产生可执行文件 a.out;

3. a.out 执行忠文件编译后的可执行文件。显示:

```
    1   main(argc, argv)
6   scanf("%d%d%d", &a, &b, &c); 输入 701
7   printf("%d\n", procd(a, b, c));
          / * a = = 7 * /
          / * b = = 0 * /
          / * c = = 1 * /
9   procd(d, e, f)
12      d = d / e+f;
          / * d = = 7 * /
```

/ * e= =0 * /

/ * f= =1 * /

Floating exception-core dump,

可以看出出错在语句第 12 行,即 d=d / e+f 由于 e=0 从而使除数为零的原因;所以可通过对原程序进行优化,给出当除数是否为零的判断选择程序段,以排除这一类的错误。

在使用 Ctrace 时需注意以下几点:

1.在正常情况下,所有 Ctrace 命令处理的输出都导向到标准输出设备,即终端或显示器上。也可以利用-P 可选参数项改变输出的产生方式。譬如:

$ Ctrace-p expl2.c > ex1.c

2.Ctrace 也有缺点,即其直接运用时产生的中间文件比较冗长,经 CC 编译后的输出也比较冗长;所以使用 Ctrace 的-s 参数选择项,可除去一些简单的赋值语句等产生的输出信息。譬如:

$ Ctrace-s expl2.c > ex2.c

3.当需要对指定的一个或多个函数进行跟踪时可使用-f 参数选择项,则使 Ctrace 仅对参数所列出的函数进行追踪。譬如:

$ Ctrace-f ex001 ex002 ex003 expl2.c > ex3.c

即对程序 expl2.c 中的 ex001, ex002, ex003 函数进行跟踪。

4.当对一个或多个函数不跟踪时可采用-v 参数选择项。譬如:

$ Ctrace-v ex001 ex003 expl2.c > ex4.c

即告诉 ctrace 对 expl2.c 程序中的函数 ex001, ex003 不进行跟踪。需要提醒注意的是-v 及-f 参数选择项不能同时并用。

以上介绍的 C 语言程序调试,检验方法只是笔者的实践与经验,各位同仁的办法一定更多更好。