第一章　ADS 简介

1.1　ADS 及开发环境

ADS(全称 AutoCAD Development System)是 AutoCAD 12.0 版提供的一种 C 语言编程环境,其用途是让应用程序开发者能用 C 语言灵活、方便地开发应用软件。ADS 应用程序对 AutoCAD 来说,和利用 AutoLISP 写成的函数一样,不能单独作为独立程序使用。

开发 ADS 应用程序所需的环境包括:

1. ADS 目标库和头文件。

2. 支持 ADS 和 AutoCAD 平台的操作系统、编译器和开发工具。

ADS 应用程序在源程序级可在各种 AutoCAD 平台之间移植。开发人员可以使用适合于自己具体环境的编译和链接工具编制应用程序。除了使应用程序具有可移植性,ADS 库函数将开发者与各种底层硬件隔离开。ADS 库函数还提供与 AutoLISP 和 AutoCAD 通信所必备的各种功能。

ADS 目标库和头文件:

ADS 自身环境由库函数和头文件所定义,目标库和头文件安装在同一目录下(\ACAD\ADS),它们适用于所有支持 AutoCAD 的平台(计算机和操作系统)。ADS 目标库由一个单独的文件(ads.lib)提供,在不同的平台上,此文件有不同的文件名。不论该目标库的名字如何,在建立 ADS 应用程序的可执行文件时,必须在链接时指定。

头文件共有四个,其文件名和内容如下:

adslib.h	包含 ADS 的一般定义(其中有些为 C 库调用重定义,目的是使其能在 ADS 环境下运行)。
adscodes.h	包含关于 ADS 库函数返回的代码值和传给 ADS 库函数的代码值的定义。
ads.h	包含 ADS 库类型定义和函数说明。

头文件 adslib.h 包含上面另外两个头文件的 #include 语句,因此每个应用程序源文件仅需包含一个关于 ADS 库的 #include 语句。

```
#include "adslib.h"
```

ol_erro.h	包含 AutoCAD 系统变量 ERRNO 使用的出错代码。

如果不使用符号代码来处理 ERRNO 的值,就不用在 ADS 应用程序中包含 ol_erro.h 文件。

1.2 ADS 应用程序的使用

1.2.1 在图形编译器中装入应用程序

使用 AutoLISP 函数(xload)装入已编译好的 ADS 应用程序,与用(load)函数装入 AutoLISP 应用程序类似。用户也可以配置 AutoCAD 使之启动时自动装入应用程序。

与(load)相同,AutoLISP 和(xload)函数要求用户提供一个文件名。

(xload filename [onfailture])

(xload)搜寻名为 filename 的文件,将其装入内存,并执行其初始化部分。与(load)不同,(xload)并不立即执行整个应用程序,例如,为装入一个名为 test.exp 的 ADS 应用程序,应执行:

Command:(xload "test")

(xload)自动为 ADS 应用程序加上扩展名(不同的平台上有不同的扩展名)。

如果(xload)找到应用程序且成功地调入内存,(xload)将返回应用程序的文件名(上例中将返回"test")。如果调用失败,返回一条错误信息。如果指定了参数 onfailture,则返回 onfailture 的值。参数 onfailture 为有效的字符串或表达式。如果是一个函数,(xload)返回时计算其值。

ADS 应用程序必须经过编译链接成可执行文件,才能被执行。

在图形编辑过程中,可多次通过调用(xload)装入多个 ADS 应用程序。一个大的 ADS 应用程序可以分成许多小程序分别装入与执行。

库搜索路径

如果不给定搜索路径,(xload)按 AutoCAD 库路径所定义的目录来搜索应用程序。AutoCAD 库路径由下述的目录组成,按查找顺序排列为:

(1) 当前目录。

(2) 包含当前图形文件的目录。

(3) 由 AutoCAD 环境变量指定的目录。

(4) 包含 AutoCAD 程序文件的目录。

这个路径同时被 AutoCAD 用来搜索菜单和其它支持文件,(load)函数也用这个路径来查找 AutoLISP 应用程序。两个或更多的库路径目录可以是相同的,这决定于当前的环境。

如果输入全路径名(例如:"d:\c5\test"),则(xload)并不搜索其它目录。

注意:不同的工作平台,路径名也有所不同。MS-DOS 使用反斜杠"\"来分隔目录名。AutoCAD 解释程序,允许用户用顺斜杠"/"来替代反斜杠。如果使用反斜杠作为路径名中的一个字符,则必须连续输入两次(例如"d:\\C5\\test")。AutoLISP 字符串变量和 C 字符串常量中需要双反斜杠。

列出已装入的所有 ADS 应用程序名

为查看已装入的所有 ADS 应用程序名,可调用 AutoLISP 函数(ads)(无参数),该函数返回当前已装入的 ADS 应用程序名。

1.2.2 调用 ADS 函数

ADS 应用程序所定义的一组函数对 AutoLISP 来说是外部函数。每次用(xload)装入后，就可以象使用一个内部建立的或用户定义的 AutoLISP 函数一样，在一个 AutoLISP 表达式中使用一个外部函数，AutoLISP 变量作为参数传给外部函数，外部函数也象 AutoLISP 函数一样返回一个值。

外部函数也可以提示用户从键盘上或用定标设备拾取点或实体的方法输入数据。

外部函数可以被 AutoLISP 函数调用，也可以交互方式调用，但 ADS 应用程序不能调用 AutoLISP 函数。

ADS 应用程序也可以像 AutoLISP 一样用 C:XXXX 的方法定义一个新的 AutoCAD 命令，在"Command:"命令提示符后直接键入命令名来使用，而不必加括号。

使用同名的外部函数可以替换原来的函数，如果两个外部函数同名，先装入的函数被丢掉。此时，即使用(xunload)函数卸载了第二个函数，先前的同名函数也不能使用了。

1.2.3 卸载 ADS 应用程序

用(xload)装入的 ADS 应用程序可以用(xunload)卸掉。(xunload)也需要指定应用程序名。

(xunload filename)

filename 文件名必须与(xload)装入时使用的文件名一致，否则无法卸载已装入的应用程序。

在下面两种情况下 AutoLISP 自动卸载 ADS 应用程序：

(1) 应用程序本身通过 ads_abort()报告一个致命错误。

(2) 退出 AutoCAD。

当退出作图或开始作新图(用 END 或 NEW 命令)时，应用程序保持装入状态。

当在内存有限的硬件平台上调用较大的 ADS 应用程序时，应将其分成多个小块，利用覆盖技术并且在需要时用(xload)和(xunload)显式操作。

注意：如果在装入一个应用程序时，其执行文件被删除或修改，则执行结果不确定。(xunload)和(xload)将不再起作用。

1.2.4 在 AutoCAD 初始化时装入应用程序

用户可以配置 AutoCAD，使之在初始化时装入应用程序。具体做法是：在初始化时，AutoCAD 在其目录下查找名为 acad.ads 的文件，且装入其中命名的每个文件。若加载成功，AutoCAD 在进入图形编辑器时提示已装入了应用程序。

acad.ads 是一个简单的文本文件，它包含一个或多个 ADS 应用程序的文件名；每行一个文件名，可以带也可以不带扩展名。如果有扩展名，该文件名不变。否则 AutoCAD 将为之附加一个与系统有关的缺省扩展名。如果文件名前有路径，则按指定路径搜索。如果没有指定路径，则按前面所述的库搜索路径查找。不过在初始化时，没有图形文件目录。

另一种在系统初始化时装入应用程序的方法是定义一个 AutoLISP 自动装入函数。显式调用(xload)，例如：

```
(defun S::STARTUP()
    (if (not (ads))              ;没有应用程序被装入
        (xload "application")
        ...
    )
)
```

1.3 ADS 函数调用与 AutoLISP 函数调用的异同

一些 ADS 库函数对于 ADS 编程环境是唯一的。但大多数 ADS 库函数的功能与 AutoLISP 函数相同。这些函数名是在 AutoLISP 同名函数前加了"ads_"。这样命名用户比较容易将 AutoLISP 程序改写成 C 程序(ADS 函数)。但解释型的 AutoLISP 和编译型的 C 环境还是有很大的不同。

1.3.1 LISP 和 C 的参数表

许多嵌入式的 AutoLISP 函数接受任意数量的参数,这对 LISP 环境是自然的,但如果 ADS 库中的 C 函数也要求有变长的参数,则会增加不必要的复杂程度与空间开销。为避免这一问题,建库时采用一条原则:与 AutoLISP 函数对应的 ADS 函数的参数表中包含前者的所有表项,当一个参数在 AutoLISP 中是可选的,则在 ADS 库中就有一个特殊的值被传递过来指定,该参数选项未使用,通常为一个 NULL 指针,有时也用一个整数,比如 0 或-1,这使得编程时比较容易,有少数 ADS 库函数例外。

ads_printf()与标准 C 库的 printf()类似,是一个变长参数函数。

ads_command()和 ads_cmd()更为复杂:AutoLISP 的(command)函数不仅接受不同类型的变长参数,也接受专门为 AutoCAD 定义的类型参数,如点、选择集等。为在 ADS 中达到同样的目的,ads_command()除了使用变长参数,还使用某些参数表示所传递的 AutoCAD 数据类型。ads_cmd()不仅需要类似的数值集合,还传入一个链表,因此,ads_command()和 ads_cmd()函数参数并不严格对应 AutoLISP 的(command)的函数参数。

ads_entget()函数没有对应的函数。AutoLISP 的(entget)函数具有一个用来查找扩展实体数据的任选参数。而 ads_entget()没有此值,因此补充了一个 ads_entgetx()函数专门用来查寻扩展实体数据。

1.3.2 关于内存的考虑

ADS 应用程序所要的内存量与 AutoLISP 不同。一方面 C 程序使用的数据结构比 AutoLISP 的表结构紧凑;另一方面,运行 ADS 应用程序的开销很大。这不仅由于程序本身的代码占一部分内存,ADS 目标库也占较大的空间。不同的平台该目标库大小也不同,对 AutoCAD 386 来说,大的在 56K—87K 之间,它取决于所支持的编译程序。

1.3.3 有关应用程序文件的组织

因为所有 ADS 程序都必须有与 AutoLISP 接口的代码以及链接的库函数,所以,如果

相关的外部函数放在一个应用程序中会减少所占内存和装入时间。如果让 AutoLISP 装入许多很小的模块，会影响 ADS 应用程序的效率。

1.3.4 内存管理

一些 ADS 函数自动申请内存，但 ADS 没有 AutoLISP 的内存自动收集功能。在大多数情况下，应用程序必须显式地释放它自己所申请的内存，否则会导致系统性能下降甚至死机。

第二章　ADS 应用程序的编写

2.1　典型的接口调用顺序

每个 ADS 应用程序都必须支持以 ADS 环境所定义的与 AutoLISP 之间的接口。这一接口要求每个应用程序按一定的顺序调用 ADS 库函数，并按特定顺序来使用某些值。

举例：

下面是一个 C 语言程序，其中的 main()函数代表了一个 ADS 应用程序的典型结构。

```
/* —ADS 应用程序的原型结构 */
#include <stdio.h>
#include "adslib.h"

static int loadfuncs();
/* main ()——主函数 */
void main (argc,argv)
int argc;
char *argc[];
{
int stat;
short scode=RSRSLT;      /* 缺省的结果码 */

ads_init (argc,argv);    /* 初始化通信接口 */
for (;;){                /* 无限循环 */
    if((stat=ads_link(scode))<0){
      printf("TEMPLATE:bad status from ads_link()=%d\n",stat);
      /* 因为通信链失败，故不能使用 ads_printf() */
      fflush (stdout);
      exit(1);
      /* 非正常结束时使用 exit() */
}
scode=RSRSLT;            /* 缺省的返回值 */

/* 用 switch_case  语句检查 AutoLISP 的请求码 */
switch (stat){
    case RQXLOAD;     /* 定义，登录 ADS 外部函数 */
        scode=loadfuncs()==GOOD? RSRSLT:RSERR;
        break;
    case RQSUBR;      /* 选择该应用程序定义的某个外部函数 */
```

```
            break;
        case RQXUNLD:     /* 对这些情况正常处理 */
        case RQSAVE:      /* 可仅返回 RSRSLT,如不需 */
        case RQEND:       /* 显式地处置它们,则毋须写 */
        case RQQUIT:      /* 输出这四个语句 */
        default:          /* 对不认识的请求,应返回 */
            break;
        }       /* switch (stat) */
    }/* for (..) */
    }/* end of main */


    /* loadfuncs()  定义外部函数 */
    static int loadfuncs()
    {
        if (ads_defun("ADSFUNC",0)==RINORM){
          ads_regfunc(adsfunc,0);/* 用函数名登录 */
          return 1;          /* 一般对每个外部函数调用一次 */
                             /* ads_defun()/*
        }
        else
            return 0;
    }
        /* adsfunc()—调用已定义的外部函数 ADSFUNC */
        /*            也可在 RQSOBR 调用此函数 */
    int adsfunc()
        {
          if (ads_getfuncodes()==0)  /* 0 是外部函数 ADSFUNC 的函数请求码 */
            return 0;
                /* 如果不是 0,则认为是对其它外部函数的请求 */
        else
            return 1;
        }
```

由上面例子可以看出 AutoLISP 访问 ADS 应用程序的顺序：

1. 当调用(xload)或 ads_xload()时，AutoLISP 在初始化时装入应用程序。

2. 调用 ads_init()初始化 ADS 应用程序与 AutoLISP 之间的通信。

3. ADS 应用程序调用 ads_link()并使用一个应用程序的返回码 RSRSLT 表明已做好接受 AutoLISP 请求的准备。

4. ads_link()从 AutoLISP 中返回请求码 RQXLOAD。

5. 通过调用 ads_defun()应用程序定义外部函数。

6. 应用程序用返回码 RSRSLT 再次调用 ads_link()(除非检查出一个错误并返回 RSERR)。

7. 当用户或一个 AutoLISP 函数要调用应用程序中的外部函数时,ads_link()从 AutoLISP 中返回请求码 RQSUBR。

8. 调用完该外部函数时,应用程序用返回码 RSRSLT 调用 ads_link()(若外部函数调用失败,则用 RSEER 调用 ads_link())。

一个 ADS 应用程序在每次调用 ads_link()之后处于非激活态,直到 AutoLISP 请求它执行一个外部函数时为止。当 ADS 应用程序在对 AutoLISP 请求作出响应过程中,AutoCAD 和 AutoLISP 处于等待状态,等待从 ADS 库函数返回的结果,并且不能对用户的输入作出任何响应。

因为要反复调用 ads_link(),所以采用将函数放在一个"无限的"循环中。用一个 switch 语句处理各种 AutoLISP 的请求。

当 ADS 应用程序做长时间计算时,AutoCAD 和 AutoLISP 处于等待状态,应允许用 Ctrl+C 来中断程序执行,可以调用 ads_usrbrk()来实现。

2.2 ADS 应用程序的初始化及有关代码

2.2.1 与 AutoLISP 通信的初始化

为了正确建立与 AutoLISP 的通信,ADS 应用程序的 main()函数必须首先调用 ads_init()。该调用建立起所有 ADS 库函数使用的通信链接,在调用 ads_init()建立与 AutoLISP 通信之前,不得调用任何其它的 ADS 库函数,否则结果不确定,很可能是灾难性的。

传给 ads_init()的参数是由 C 语言系统接口定义的 argc 和 argv,它们必须与 AutoLISP 调用该应用程序时传给其 main()函数的参数值相同,如果 ads_init()调用失败,将不会从该调用返回,所以不必去检查其返回值。

2.2.2 AutoLISP 的请求码

ADS 应用程序通过 ads_link()返回 AutoLISP 的请求码。在 ADS 应用程序装入后,通常处于等待 AutoLISP 返回请求码的状态。

ADS 共有六个请求码,分支循环必须识别不同的请求码,并作出不同的响应。通常用 switch 语句,对未使用的或不能识别的代码作出缺省处理。

当 ADS 应用程序接到 AutoLISP 的请求后,可以做自己的工作或调用 ADS 库函数,但最终必须通过再次调用 ads_link()来完成对请求的反应,并通过返回一个结果码来表明成功与失败。

在文件 adscodes.h 中,请求码定义为整型值。

RQXLOAD 请求定义函数。

此码表明应用程序已装入内存,要求应用程序定义外部函数。

应用程序必须为每个外部函数调用一次 ads_defun()。

注意:当应用程序第一次被装入内存或 AutoLISP 重新初始化

时，AutoLISP 都发出 RQXLOAD 的请求，因此应用程序每次接到 RQXLOAD 时，都必须重新定义外部函数。如果应用程序忽略了这一请求，AutoLISP 就不能识别该应用程序的外部函数名。

RQSUBR 请求调用外部函数。

此码表明 AutoLISP 要调用一个外部函数。应用程序必须调用 ads_getfuncode()来得到该函数的(非负整数)码值，然后用这一代码值来选择和调用该函数。这个代码值必须是该应用程序在此之前用 ads_defun()定义的码值之一。

注意：如果一个外部函数用 ads_regfunc()登记过，AutoLISP 可直接调用它。如果所有的外部函数都被登记过，则该应用程序不再接受 RQSUBR 请求，否则将发生内部错误。

RQXUNLD 请求卸载应用程序。

通知 ADS 应用程序、用户或 AutoLISP 函数已调用了(xunload)，请求 AutoLISP 卸去该应用程序。通常，应用程序应仅返回一个正常的状态码(RSRSLT)，做一些善后工作，如释放分配空间、关闭文件、释放选择集等。

如果释放空间或选择集后影响到共享资源中的其它 ADS 应用程序，该应用程序拒绝这样做，并返回出错状态码(RSERR)。同时 AutoLISP 照常终止和卸载应用程序，并把错误信息报告给用户。

下面的请求码通知应用程序：AutoCAD 将存盘、退出或存盘退出。通常，应用程序只返回正常状态(RSRSLT)，存取数据库文件的应用程序应做一些处理工作。

RQSAVE 请求保存图形，通知 ADS 应用程序已调用 AutoCAD 的 SAVE 命令，在保存图形之前，AutoCAD 等待应用程序作出反应。

RQEND 请求保存并退出。

通知 ADS 应用程序，已调用 AutoCAD 的 END 命令，并在实际存储图形并保存图形编辑器当前状态之前，AutoCAD 会等候应用程序对此作出反应。

RQQUIT 请求放弃当前图形并退出。

通知 ADS 应用程序，已调用 AutoCAD 的 QUIT 命令，并已确认放弃当前图形，在退出图形编辑器之前，AutoCAD 等待应用程序对此请求作出反应。

2.2.3 ADS 应用程序结果码

ads_link()有一个整型参数，ADS 应用程序可以用此参数表示自身状态，该参数必须等于 adscodes.h 中定义的某个结果码。

RSRSLT AutoLISP、应用程序已完成任务并等待进一步请求。如 2.1 节的实例所示，主分支循环得到控制后，必须使用缺省结果码。

注意:为保持向上的兼容性,应用程序接到不能识别的代码时,必须返回 RSRSLT。将来的 AutoCAD 版本会定义其它请求码。

RSERR

通知 AutoLISP 在 ADS 应用程序中发生错误,如果这个结果是在 RQXLOAD 请求之后,AutoLISP 将终止该应用程序,并卸掉。如果错误在 RQSUBR 请求时发生,则 AutoLISP 将取消对外部函数的请求。

当 AutoLISP 接收到 RQSUBR 请求返回的 RSERR 结果码时,将显示信息:

error:ADS error in evaluation

注意:应用程序结果码可正也可负,为保持向上兼容性,ads_link()函数可接收正的或负的结果码。在 12 版本中,正负结果码等价。

2.3 外部函数的定义和引用

应用程序通过 ads_defun()定义的外部函数,可以像内部函数或用户定义的 AutoLISP 函数一样被 AutoLISP 用户的程序调用,可以将 AutoLISP 数据及变量传入一个外部函数,该函数在调用它的表达式中返回一个值。

2.3.1 定义外部函数

举例:

```
ads_defun("doit",0);
```

上面的例子在 AutoLISP 中定义一个叫(doit)的外部函数,当 AutoLISP 调用(doit)时,函数码 0 就传给了应用程序。

当 AutoLISP 发出 RQXLOAD 请求后,ADS 应用程序必须用 ads_defun()来定义所有的外部函数,调用 ads_defun()只是赋给外部函数名一个唯一的非负的、不大于 32,767 的整数码。

外部函数名可以是任何有效的 AutoLISP 符号名,AutoLISP 将其自动变成大写并插入到一个 EXSUBR 的原子表中。

举例:

```
ads_defun("C:DOIT",0)
```

此例中,把外部函数定义成 AutoCAD 命令。与 AutoLISP 类似,ADS 在其外部函数名前加上前缀"C:",即定义了一个新的 AutoCAD 命令,可在"Command:"提示行中直接调用,不必加括号。

上面定义的函数也可以被 AutoCAD 用户象调用函数一样调用,并带有参数。

Command:(C:doit x y)　　与

Command:doit　　相同。

注意:如果应用程序用"C:xxx"方式定义的外部函数与已有命令(系统原有或 acad.pgp 文件中定义的)名相冲突,AutoCAD 将不把该外部函数识别为一条命令(但在 AutoLISP 中

可被引用)。

用 ads_defun()定义的外部函数可用 ads_undef()来解除定义,因为解除后该函数将从 AutoLISP 的原子表中被删除,再调用会发生错误。

2.3.2 引用外部函数

举例:

```
/* 定义一个函数调用表 */
#define ELEMENTS (array) (sizeof (array)/sizeof ((array)[0]))
struct func_entry {char * func_name;int (*func) ();};
static struct func_entry func_table[]={{"fact",fact},
                                       {"sqr",squareroot};
                                       };
/* 声明外部函数的原型 */
int fact (struct resbuf *rb);
int squareroot (struct resbuf *rb);

int dofun (void);
int funcload (void);
/--定义外部函数 */
static int funcload()
{
    int i;
    for (i=0;i<ELEMENTS(func_table);i++){
      if(! ads_defun(func_table[i].func_name,i))
          return RTERROR;
    }
    return RTNORM
}
/* 调用外部函数 */
static int dofun()
{
    struct resbuf * rb;
    int val;
    if ((val=ads_getfuncode())<0||val>=ELEMENTS(func_table)){
        ads_fail ("Received nonexistent function code.");
        return RTERROR;
    }
    rb=ads_getargs();
    return (*func_table[val].func)(rb);
}
```

上面是一个定义和引用外部函数的例子。本例中,先定义了一个函数调用表,用来选取和引用外部函数,然后用 funcload()函数定义外部函数,通过 ads defun()把 0 函数码定义给外部函数(fact),把函数码 1 定义给(sqr)。

一个外部函数被正确定义后,AutoLISP 即可以用 RQSUBR 请求来引用它。当相应的 ADS 应用程序接收到这个请求后,通过调用 ads getfuncode()得到该外部函数的整数码,然后使用 switch 语句、if 语句或一个事先由该应用程序定义好用于选取和调用给定处理函数的表。

注意:该函数名可以与由 ads_defun()定义的名不完全相同。上例中由 ads_defun()定义的 sqr(用于由 AutoLISP 识别)和引用时的 squareroot 不完全相同。

如果该处理函数要求有参数输入,可以通过调用 ads getargs()得到相应的参数值。(ads_getargs()返回一个指向"结果缓存"链表的指针,该缓存中存有由 AutoLISP 返回的值)。如果处理函数不要求输入参数(通常用来定义 AutoCAD 命令)就不必调用 ads_getargs()。因为处理函数从链表取得它的输入参数,所以一个处理函数可以有可变长度参数或不同的参数类型。

注意:处理函数必须自己确定传给它的参数个数和数据类型,没有办法让 AutoLISP 知道这些要求。

对于定义为 AutoCAD 命令的外部函数,当 ads_getargs()返回空表时,应要求该处理函数提示用户输入数值。

有一部分 ADS 库函数可以返回值,如 ads_retint(),ads_retreal(),ads_retpoint(),这些函数使外部函数返回一个值给调用它的 AutoLISP 表达式。

外部函数和 AutoLISP 之间传递的参数必须是以下数据类型中的一种:整数、实数(浮点型)、串、坐标点(在 AutoLISP 中被表示成一个用两个或三个实型数组成的表)、实体名称、选择集名、AutoLISP 中的符号 t 和 nil 或是包含以上元素的表。总之,AutoLISP 中的符号(除了 t 和 nil)均不能传递到外部函数,或由外部函数返回。但通过调用 ads_getsmg()和 ads putsym(),ADS 应用程序可以取得和传递 AutoLISP 符号。

假设 ADS 应用程序中有一外部函数,其参数为一个串,一个整数和一个实数,用 AutoLISP 可以写为:

```
(doitagain pstr iarg rarg)
```

如果此函数已用 ads_defun()定义,则 AutoCAD 中可以这样调用它:

```
Command: (doitagain "starting width is" 37.12)
```

这个调用提供了函数所需的输入参数,doitagain 处理函数可通过调用 ads_getargs()来得到这些参数。

2.4 出错处理

在复杂的交互环境下运行的 ADS 应用程序必须很健壮。ADS 环境提供了一些检查和处理出错情况的方法。

(1)检查 ADS 应用程序传递给 AutoLISP 的结果码;

(2)检查 ADS 库函数返回的结果码;

(3)提示和接收用户输入的库函数；

(4)ADS 库中提供 ads_fail()、ads_abort()和 ads_alert()来处理出错；

ads_fail()函数在 AutoCAD 的提示符 Command：后显示一个作为单串传递的错误信息，该函数可用于恢复出错。

在 test.exp 程序中调用 ads_fail()：

ads_fail("Invalid osnap point \n")；

该 ads_fail()打印下列信息：

Application test.exp ERROR：Invalid osnap point.

ads_abort()函数像 ads_fail()一样显示一条错误信息。但是等信息显示完后，AutoLISP 还将卸载整个应用程序。ads_abort()函数处理的是 ADS 应用程序中致命性错误，不到万不得已时不用。

在 test.exp 程序中调用 ads_abort()：

ads_abort("invalid entity list detected \n")；

ads_abort()函数将在程序中终止前显示：

Application test.exp FATAL ERROR：invalid entity list detected.

注意：应用程序终止的另一种方法是调用 ads_exit()。一旦调用 ads_init()，ADS 应用程序就不能调用标准 C 库函数 exit()。(与 11 版 ADS 库不同，调用 exit()并不导致调用 ads_exit()，因为不是所有的编译器都允许重新定义)。

ads_alert()函数可以显示警告框来通知用户出错情况。

举例：

下面对 ads_alert()的调用将显示下图所示的警告框，

ads_alert("File not found")；

图 2-1 警告框

若想了解更多调用某个 ADS 库函数失败的原因，可检查 AutoCAD 系统变量 ERRNO。

ADS 库函数调用(或 AutoLISP 库函数调用)失败时，可以通过调用 ads_getvar()来查看 ERRNO 所设置的错误码信息。这些码的符号名定义在 ol_errno.h 中。

2.5 ADS 应用程序之间的通信

ADS 库提供一个 ads_invoke()函数，使得一个 ADS 应用程序可以调用在其他 ADS 应用程序中已定义的外部函数。和别的外部函数一样，被 ads_invoke()调用的函数必须由当前装入的 ADS 应用程序定义。

ads_invoke()使用其应用程序在 ads_defun()中定义的名字来调用外部函数，如果外部函数被定义成 AutoLISP 命令，名字前加上前缀 C：，用 ads_invoke()调用函数时，函数名前

必须带前缀。

注意:每次编辑期间所装入的应用程序不能有相同的函数名。当开发多个 ADS 程序组成的应用程序时,应采用适当的方式和命令习惯以保证每个外部函数名的唯一性。

外部函数名及其所要求的参数值是通过 ads_invoke()返回的结果缓存表给出。

举例:

下面的函数用 ads_invoke()来调用样例程序 fact.c 定义的阶乘函数 fact():

```
static void test()
{
int stat,x=10;
struct resbuf *result=NULL,*list;
list=ads_buildlist (RTSTR,"fact",
                    RTSHORT,x,
                    RTNONE);
if (list != NULL){
    stat=ads_invoke(list,&result);
    ads_relrb(list);
}
if (result != NULL){
    ads_printf("\n Test succeeded:factoria, of %d is %d \n",
                              x,result -> resval.rint);
    ads_relrb(result);
}else
    ads_printf("Test failed \n");
} /* 程序结束 */
```

如果使用 ads_invoke()来调用函数,应用程序必须用 ads_regfunc()来注册。ads_regfunc()的调用使 ADS 库直接调用该函数。

如果要使用 ads_regfunc(),则应在 ads_defun()定义了函数后调用。

举例:

下面的例子是前面例子的扩充:

```
static int funcload()
{
    int i;
    for (i=0; i <ELEMMENTS(func_table);i++){
        if (! ads_defun(func_table [i],func_name;i))
          return RTERROR;
        if(! ads_regfunc(func_table [i],func,i))
          retrun RTERROR;
        }
    return RTNORM;
```

}

ads_regfunc()的第一个参数是函数指针，而不是由 ads_defun()定义并由 AutoLISP 或 ads invoke()使用的外部函数名。但 ads_defun()和 ads_regfunc()必须用同一整型函数码。

注意：由 ads_regfunc()登录的函数必须是整型的，并返回某个应用结果码。

如果一个函数内嵌套了一系列函数，其中一个函数位于同一应用程序内且由 ads_invoke()调用，那么这个函数必须用 ads_regfunc()来注册，否则 ads_invoke()返回错误信息。

举例：

A_tan()调用 B_sin()，

A_tan()调用 C_cos()，

B_sin()调用 A_pi()，

C_cos()调用 A_pi()，

其中：

A_tan()和 A_pi 定义在应用程序 A 中，

B_sin()定义在应用程序 B 中，

C_cos()定义在应用程序 C 中，

则函数 A_pi()必须通过 ads_regfunc()登记。

为避免担心 ads_invoke()返回错误信息，一个安全的办法是将所有的 ads_invoke()调用的函数都登记。

ads_regfunc()也可以用来解除对外部函数的定义。必须是同一应用程序负责登录和解除，ADS 不允许一个应用程序管理另一个应用程序。

当 ads_invoke()返回 RTNORM 时，只表明外部函数的调用和返回是成功的，并不说明外部函数得到了结果，须对 result 参数进行检查。

2.6 装入和卸载外部应用程序

ADS 12.0 新增加了函数 ads_xload()、ads_xunload()和 ads_loaded()，用来装入、卸载其它 ADS 应用程序，并得到一个当前装入的应用程序表。

ads_xload()装入一个外部应用程序，其寻找指定文件方法与(xload)一样。

举例：

下面的调用装载一个 myapp 的应用程序：

```
if (ads_xload("myapp")!=RTERROR){
    /* 用 ads_invoke()来调用 myapp 中的外部函数 */
}
```

当用户运行到 myapp 结束时，可通过 ads_xunload()来卸载：

```
ads_xunload ("myapp");
```

ads_loaded()函数与(ads)等价。

下面的程序获得当前装载的应用程序名，显示它们，然后卸载所有的应用程序。

```
struct resbuf *rb1, *rb2;
for (rb2=rb1=ads_loaded();rb2!=NULL;rb2=rb2->rbnext){
```

```
if (rb2→restype==RTSTR)
    ads_printf("%s\n",rb2→resval.rstring);
if (strcmp (ads_appname,rb2→resval.rstring)!=0)
    ads_xunload (rb2→resval.rstring);
}
ads_relrb(rb);
```

2.7 函数返回值与函数返回结果

少数 ADS 库函数不返回值(void 类型),有些直接返回结果,但多数返回一个 int 类型的整数状态码来表示其调用是否成功。acadcodes.h 文件中定义了这些状态码。

代码 RTNORM 表明函数调用成功;其它的可能值表示调用失败或遇到特殊情况。返回状态码的库函数将通过传递地址的参数回送其结果。

注意:不要把库函数的返回值与库函数的返回结果参数混为一谈。返回值一般为整数状态码,返回结果是放在一个传递指针的参数中。

举例:

考虑下面典型的 ADS 库函数的函数原型。

```
int ads_entnext (ads_name ent,ads_name result);
int ads_osnap (ads_point pt,char *mode,ads_point result);
int ads_getint (char *prompt,int *result);
```

在应用程序中用下面方式来调用

```
stat = ads_entnext (ent,entres);
stat = ads_osnap (pt,mode,ptres);
stat = ads_getint (prompt,&intres);
```

当函数调用完成后,变量 stat 的值表示该次调用是成功(stat==RTNORM)或失败(stat==RTERROR)或另一个错误代码(如 RTCAN 表明取消)。每个参数表中的最后一项是结果参数,必须以传递指针方式给出。

如果调用成功,ads_entnext()将用参数 entres 返回一个实体名,ads_osnap()将用 ptres 返回一个点,ads_getint()将用 intres 返回一整数。因此 ads_name 和 ads_point 是数组类型,所以参数 entres 和 ptres 无须写成传递指针形式。

注意:在 ADS 库函数声明中,结果参数总是位于输入参数之后,这不是语法上的要求,而是一种广泛的习惯做法。

第三章　ADS 中定义的变量、类型和值

ADS 在 C 语言定义的类型基础上，扩充定义了一些适用于 AutoCAD 环境的特殊的类型，一些函数传送的符号值以及若干全局变量、定义和说明都放在 ADS 头文件中。

3.1　一般类型及其意义

此处介绍的类型和定义是用来使应用程序适应 AutoCAD 的要求，并保持一致性。

3.1.1　实数

AutoCAD 中实数的值总是双精度浮点值。ADS 定义了其特殊类型以保持这一标准。

```
typedef double ads_real;
```

在向 AutoCAD 传递和接收实数或计算时，都应使用 ads_real 类型变量。

3.1.2　坐标点

AutoCAD 的坐标点(简称点)在 ADS 中(头文件 ads.h 中)定义成数组类型。

```
typedef ads_real ads_point [3];
```

一个点总是包含几个值，如果点是二维的，其第三个元素被忽略，为保险起见应将其置为 0。

为处理点值时清晰起见，ads.h 中包含下述定义：

```
#define X 0
#define Y 1
#define Z 2
```

与简单数据类型(或 AutoLISP 中的点表示)不同，ADS 中不能用单一的语句给一个点所有分量赋值。

下面的表达式是错的：

```
newpt=oldpt;
```

正确的应该是：

```
newpt[X]=oldpt[X];
newpt[Y]=oldpt[Y];
newpt[Z]=oldpt[Z];
```

利用 ads.h 中的宏定义 ads_point_set()可以拷贝一个点。

举例：

下面的代码将点 to 的值赋给点 from：

```
ads_point to, from;
ads_point_set (from,to);
```