基于编译技术的协议自动化解析程序的设计

邓 伟[1,2]，石 炜[2]
（1. 通信信息控制和安全技术重点实验室 浙江 嘉兴 314033；
2. 中国电子科技集团公司第三十六研究所 浙江 嘉兴 314033）

摘要： 复杂系统中多个单元之间存在着不同的通信协议，开发人员通常需要花费大量的时间编写相应的协议解析代码。文中通过对常见的通信协议进行分析，提出了采用形式化描述方法来描述这些协议，并利用编译技术开发了相应的协议自动化解析程序。文中提出的方法避免了大量重复性代码的开发，减少了协议代码的开发工作，在通信协议软件开发方面具有很强的实用性。

关键词： 协议分析；编译技术；flex；bison
中图分类号： TN332 **文献标识码：** A **文章编号：** 1674-6236(2012)13-0033-04

Protocol automated parser design based on compiler technology

DENG Wei[1,2], SHI Wei[2]
(1. *Science and Technology on Communication Information Security Control Laboratory*, *Jiaxing* 314033, *China*;
2. *No.36 Research Institute of CETC*, *Jiaxing* 314033, *China*)

Abstract: Among multiple units in a complex system, there exist a lot of different communication protocols, engineer often need to spend a lot of time to develop the corresponding protocol parsing code. This paper analyzes the common communication protocols, proposes using the formalize-description to describe these protocols and implements a protocol automated parser using compiler technology. The proposed method avoids the development of a large number of repetitive codes, reducing the development work, and has a highly practical application in communication protocol software development.

Key words: protocol analysis; compiler technology; flex; bison

在复杂的通信系统设计过程中，设计者通常会将系统划分为独立的功能单元，单元之间通过接口相互连接，实现系统功能。一般来说，单元之间的物理连接会尽量采用同一种物理接口，但建立在物理接口之上的通信协议却因单元提供功能的差别而各不相同。开发人员必须为每个单元开发相应的协议解析程序，从协议数据中提取出所需要的信息加以处理，完成单元的控制。

这些通信协议的构成通常很类似，其主要构成为帧数据，并且帧数据的格式内容大都包含地址、命令、参数、数据等信息。传统的手工开发方式需要针对每条协议编写相应的解析程序，使得系统开发人员的主要精力无法集中在系统功能本身。针对这种情况，研究人员提出了远程过程调用(RPC)机制试图解决这一问题，但 RPC 的实现通常需要系统提供操作系统和 TCP/IP 协议栈[1-2]，这无疑增加了系统开发的难度；此外，一些简单的单元的接口控制逻辑通常是由低端的单片机构成，无法提供 RPC 机制所需的运行环境。这极大地限制了 RPC 机制的应用范围。相关文献[3]指出，可以通过采用编译技术来完成协议的解析和识别，但该文献提出的方案的需要宿主机提供解释性语言的支持，如 VB、Python 等语言，这些解释性语言通常在大多数单元中并不存在。

笔者针对上述解决方案的不足，提出了一种接近于帧协议文本的帧协议描述语言，并利用编译技术实现了一种自动化解析器，该解析器将帧协议描述语言自动转换为标准 C 语言的数据类型，并生成相应的解析函数和释放函数。利用该方案，开发人员只需要针对协议文本中的帧结构编写相应的帧协议描述说明，自动化解析器可根据帧协议描述说明自动生成相应的解析程序。开发人员调用生成的解析函数，即可提取出帧数据中包含的信息。该方案相对于 RPC 机制，减少了对系统环境的需求，极大地扩展了该方案的适用性。

1 总体设计方案

文中设计的协议自动化解析器主要包括帧协议语言、词法分析器、语法分析器、语义分析等，协议自动化解析器的工作流程如图 1 所示。

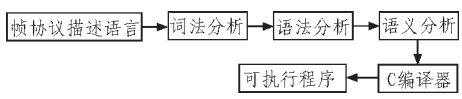

图 1 协议自动化解析器工作流程
Fig. 1 Working flow of automatic protocol analyzer

收稿日期：2012-04-10 稿件编号：201204073
作者简介：邓 伟(1978—)，男，甘肃兰州人，硕士，工程师。研究方向：嵌入式系统设计、高速电路设计。

© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

协议自动解析器首先对帧格式描述语言进行词法分析和语法分析,根据分析结果生成标准 C 语言的数据结构定义和帧数据的解析函数。由于在帧数据的解析的过程中,需要动态的分配系统内存,因此自动化解析器还生成了对应的内存释放函数。这样避免了开发人员手工编写内存释放代码可能存在的内存泄露问题。将自动化解析器生成的 C 代码通过 C 编译器编译变为可执行代码。当某帧数据到达时,利用可执行代码即可完成帧协议的解析,并提取相应的信息,复制到帧数据 C 语言结构体相关域,应用程序通过访问该结构体即可完成信息的访问。

2 设计方案概述

2.1 帧协议描述语言

文中设计的帧协议描述语言是为了描述常用的通信协议而提出的一种专用描述语言,通过该语言可使设计人员方便的描述通信帧结构。帧协议描述语言主要包括词法部分和语法部分。词法主要包括对帧协议语言所使用的关键字和保留符号的描述。为了使设计的描述语言符合开发人员惯用的方式,文中定义了 6 类关键字和 4 类保留符号,如表 1 所示。

表1 帧描述语言中的关键字和保留符号

Tab.1 Keywords and reserved symbols in frame description language

关键字	定义	字节数	保留符号	定义
int	整型	4	:	变量/数组/域开始标志符
uint	无符号整型	4	;	变量/数组结束标志符
short	短整型	2		数组定义标志符
ushort	无符号短整型	2	{}	域定义标志符
char	字符型	1	“标志符”由字符和数字构成的字符串	
uchar	无符号字符型	1		

帧协议描述语言语法部分采用了常见的上下文无关文法来定义。文中设计的帧协议描述语言的主要语法元素包括帧、域、变量和数组,具体的语法结构定义如表 2 所示。

表 2 帧描述语言的语法元素

Tab.2 Syntax element in frame description language

语法元素	描述
帧	由帧名称和一个或多个域构成
域	由一个或多个变量/数组/子域构成
变量	由关键字构成
数组	由一组变量/子域构成,可以是定长数组或是变长数组

2.2 基于 Flex 工具的词法分析

Flex 是 lex 工具的开源版本[4],它的主要功能是根据用户定义的词法规则,生成面向字符流的自动扫描程序,将字符流分解为独立的 token 和 token 所对应的词法属性。采用 flex 编写的词法分析程序主要包括 3 部分, 每一部分用%%号分开。其中第一部分为引用的头文件,变量声明等;第二部分为词法规则定义的部分;第三部分为用户定义的函数。每个词法规则都由模式和动作两部分构成。其中模式部分采用正则表达式来定义,当字符流中出现了符合该正则表达式的部分时,词法扫描程序将执行该模式对应的动作。

帧协议描述语言词法分析主要包括对描述语言所定义的关键字、保留符号和合法标志符的识别,其部分 flex 程序如下:

```
“int”              { return INT_DECL; }
“short”        { return SHORT_DECL; }
“char”         { return CHAR_DECL;  }
“uint”         { return UINT_DECL; }
“ushort”       { return USHORT_DECL; }
“uchar”        { return UCHAR_DECL;  }
[a-z][0-9a-zA-Z_]*    {  strcpy(yylval.idName,yytext);
return VAR;
}
[0-9]+      {
yylval.iVal = atoi(yytext);
return INT;
}
[\{\}:;]     { return *yytext; }
[ \t\n]+       ;   /* skip whitespace */
.           yyerror("Unknown character");
```

上述程序通过 flex 工具编译并生成词法分析程序后,即可对帧协议语言进行词法分析,该程序遇到 int 等关键字,会返回该关键字属性;在遇到合法标志符时,会将 yytext 中该标志字符串复制到 yylval.idName 域中,返回变量属性;在遇到整型字符串时, 会将 yytext 中的字符串转为整型, 赋值到 yylval.iVal 域中,返回整型属性。

2.3 基于 Bison 工具的语法分析

Bison 工具作为 yacc 的开源版本[4],主要功能是根据用户定义的上下文无关文法来描述语言规则,从而生成语法分析的程序。Bison 程序的结构与 flex 类似,同样采用%%将文件分为 3 部分,第一部分包含头文件声明、函数声明等;第二部分包括所有的语法规则表达式, 也是语法分析的主要部分;第三部分是用户自定义函数,主要包含帧协议描述文件的打开、关闭,输出文件的创建等工作。

帧协议描述语言的 bison 文件的规则部分是主要构成部分。bison 文件的规则部分将帧协议语法描述为一套产生规则, 每条规则都由

```
<符号>: < 展开式 1 > { 动作 1 }
        | < 展开式 2 > { 动作 2 }
        | …
        | < 展开式 N > { 动作 N }
```

构成。例如,帧的 bison 语言定义如下:

```
frame: frame_name ‘:’ ‘{‘fields’}’ { print_def($4);
print_all($4); }
```

它表示一个通信帧由帧名称和一个或多个域构成,对应的动作是生成这条通信帧的 C 语言结构体的定义。帧描述语

© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

言完整的规则部分如下：

```
frame: frame_name ':' '{' fields '}' { print_def($4); print_all($4);};
frame_name: VAR ;
fields: field { $$=$1; }
| field fields { $1->pnegb = $2; $$=$1; };
field: VAR ':' type ';'{ $$ = field_const_ins($1,1,$3,norm_node,NULL);}
| VAR':''{'fields'}'{$$=field_const_ins($1,1,typeNULL,stru_node,$4);}
| var_arr { $$ = $1; };
var_arr:VAR'[' INT ']':' type ';'{ $$ = field_const_ins($1,$3,$6,norm_node,NULL);}
| VAR '['INT ']' ':' '{'fields '}' { $$ = field_const_ins($1,$3,typeNULL,stru_node,$7); }
| VAR '[' VAR ']' ':' type ';' { $$ = field_var_ins($1,$3,$6,norm_node,NULL);}
| VAR '[' VAR ']' ':' '{' fields '}' { $$ = field_var_ins($1,$3,typeNULL,stru_node,$7);}
type: INT_DECL { $$ = typeInt; }
| TRIB_DECL { $$ = typeTrib; }
| SHORT_DECL { $$ = typeShort; }
| CHAR_DECL { $$ = typeChar; }
| UINT_DECL { $$ = typeuInt; }
| UTRIB_DECL { $$ = typeuTrib; }
| USHORT_DECL { $$ = typeuShort; }
| UCHAR_DECL { $$ = typeuChar; } ;
```

Bison 语法分析程序采用 LALR 分析方法[5-6]将词法分析程序返回的 token 序列与给定的语法规则匹配，并依据匹配的语法部分构建成相应的语法树。在帧语言文件分析结束后，相对应的语法树也已建立，此时语法分析阶段结束，由主程序调用语义分析函数完成帧协议分析函数的生成。

帧协议描述语言的语义分析部分主要是从根节点遍历语法部分构成的语法树，并根据树中每个节点的不同类型，产生相应的标准 C 代码。语义部分通过 3 次遍历语法树，分别产生帧结构定义，帧数据解析函数和帧数据释放函数。

3　试验验证

将文中设计的协议自动化解析器应用到某通信系统单元中，该单元的授时帧格式如表 3 所示。

表3　授时协议

Tab.3　Time sync protocol

域名称	类型	字节数	描述
信息长度	unsigned int	4	帧长度
时间	Char[8]	8	年月日时分秒
精确时间	int	4	毫秒

根据表 3，可采用协议描述语言将授时帧描述如下：

```
time_sync_frame:  // 授时帧
{
    frame_len : int;  // 信息长度
    time[8] : char;  // 时间
    time_in_ms : int;  // 精确时间
}
```

将上述文件作为协议自动化解析器的输入，输出结果如下：

```
typedef struct _time_sync_frame
{
    int frame_len;
    char time[8];
    int time_in_ms;
}_time_sync_frame_t;

_time_sync_frame_t*time_sync_frame_anaylze_fn(char*buf)
{
    _time_sync_frame_t* ptr=NULL;
    int pos;
    int i0;
    if ( buf == NULL )
    {
        printf("para is wrong.\n");
        return NULL;
    }
    if ( (ptr = (_time_sync_frame_t*)malloc (sizeof (_time_sync_frame_t))) == NULL )
    {
        printf("alloc memory error.\n");
        return NULL;
    }
    pos = 0 ;
    // assign value to frame_len
    ptr->frame_len = *((int*)(&(buf[pos])));
    pos += sizeof(int);
    // assign value to time
    for(i0 = 0; i0 < 8 ; i0++ )
    {
        ptr->time[i0] = *((char*)(&(buf[pos])));
        pos += sizeof(char);
    }
    // assign value to time_in_ms
    ptr->time_in_ms = *((int*)(&(buf[pos])));
    pos += sizeof(int);
    return ptr;
}
void time_sync_frame_destory_fn (_time_sync_frame_t
```


© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

```
*xnode)
    {
        _time_sync_frame_t* ptr=xnode;
        int i0;
          if ( xnode == NULL )
        {
            printf("input para is NULL! . \n");
            return;
        }
        // free whole allocated structure.
        free(ptr);
        return;
    }
```

将输出的文件通过 C 编译器编译，编译顺利通过。该试验结果显示，协议自动化解析器生成的 C 代码可以取代手工代码开发，应用程序通过调用 time_sync_frame_anaylze_fn 函数即可完成授时帧的解析，并在函数返回后可直接访问返回变量中的域，提取相关信息。如果授时帧协议有修改，开发人员只需要修改帧协议描述文件，自动化解析器将自动完成 C 代码的生成，这极大地减少了开发人员的负担。

4 结束语

文中开发自动化解析器可减少大量的代码编写工作，并且编码的准确性有很大提高。针对通信帧协议的更新，开发人员只需要修改帧协议描述文件，这简化了帧协议代码的维护工作，降低了帧协议开发的难度，在通信协议软件开发方面具有很强的实用性。

参考文献：

[1] Iseminger D. 网络连接服务开发人员参考库RPC与WNet[M]. 3卷.北京：机械工业出版社，2001.

[2] 许琼，蔡文皓. 一种嵌入式RPC的设计与实现[J]. 电子设计工程，2011(5): 127–129.
XU Qiong，CAI Wen-hao. Design and implementation of embedded RPC[J]. Electronic Design Engineering，2011(5): 127–129.

[3] 董立，赵恒永. 基于编译技术的协议解析方法[J]. 计算机工程，2007(11):66–68.
DONG Li，ZHAO Heng-yong. Protocol parsing method based on compiling technology [J]. Computer Engineering，2007(11): 66–68.

[4] 利文. flex与bison[M]. 中文版.南京：东南大学出版社，2011.

[5] 丁文魁，杜淑敏. 编译原理和技术[M]. 北京：电子工业出版社，2008.

[6] 刘坚. 编译原理基础[M]. 西安：西安电子科技大学出版社，2008.

（上接第 32 页）

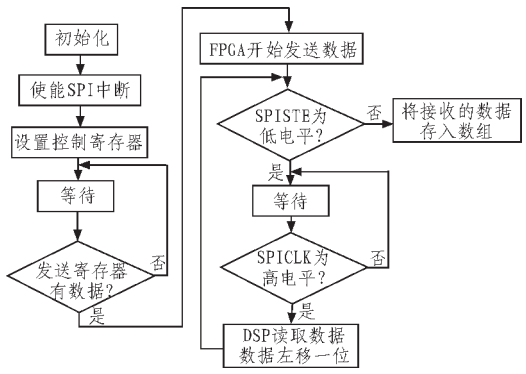

图 7 DSP 与 FPGA 之间的通信流程

Fig. 7 Flowchart of SPI communication between DSP and FPGA

满足磁铁电源输出极低纹波和受精确控制的电流。

参考文献：

[1] 马红霞，燕伟康，赵升. 一种高稳定度软开关加速器磁铁稳流电源[J]. 电气传动自动化，2009，31(3):32–34.
MA Hong-xia，YAN Wei-kang，ZHAO Sheng. A magnet stabilized-current supply of high-stability soft switch accelerator[J]. Electric Drive Automation，2009，31(3):32–34.

[2] 赵久籍，尹兆升. 粒子加速器技术[M]. 北京:高等教育出版社，2006.

[3] 李夕红. 基于DSP和FPGA的数字化开关电源的实用化研究[D]. 成都:成都理工大学，2008

[4] 刘勇，祝忠明，罗文渊，等. 基于FPGA+DSP的高精度数字电源数据采集系统设计[J]. 电子元器件应用，2009，11(1): 25–27.
LIU Yong，ZHU Zhong-ming，LUO Wen-yuan，et al. Design of high–precision digital power data collecting system based on FPGA +DSP [J]. Electronic Components Applications，2009，11(3):25–27.

[5] 刘鹏鹏，王晶，尹小杰，等. 基于DSP和FPGA的通用控制器的设计[J]. 电子设计工程，2011，19(21):170–172.
LIU Peng-peng，WANG Jing，YIN Xiao-jie，et al. Design of a genrral motion controller based on DSP and FPGA [J]. Electronic Design Engineering，2011，19(21):170–172.

[6] 顾卫钢. 手把手教你学DSP—基于TMS320X281X[M]. 北京:北京航空航天大学出版社，2010.

© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net