第27卷第6期 南京气象学院学报 Vol. 27 No. 6
2004年12月 Journal of Nanjing Institute of Meteorology Dec. 2004

文章编号: 1000-2022(2004)06-0806-08

插件技术在天气雷达数据处理软件中的应用

沃伟峰, 吴 蕾, 顾松山
(南京信息工程大学 电子工程系, 江苏 南京 210044)

摘 要: 介绍了在多普勒天气雷达二次产品软件的编写中使用动态链接库(Dynamic-link Library)技术设计的一个插件系统(Plug-in System)。该系统中所有的算法都设计成插件, 将产品算法部分和系统本身分离, 便于工程的管理和进一步开发, 使研究人员能够集中精力致力于算法的实现。该系统本身也可以作为一个新算法、新产品的开发平台。

关键词: 插件; 多普勒天气雷达二次产品; 雷达软件; 动态链接库
中图分类号: TP399 文献标识码: A

20世纪80年代以来, 世界各国纷纷把提高对强对流天气监测能力作为减灾防灾的主要措施之一, 许多发达国家和地区相继建立了多普勒天气雷达监测网。通过引进吸收、合作开发, 我国目前已具备生产多普勒天气雷达的能力, 计划到2010年前, 分期分批在全国范围内布设新一代天气雷达, 建立中国新一代天气雷达监测网。

新一代多普勒天气雷达的功能是常规天气雷达无法比拟的。它不仅能提供实时降水强度信息, 而且能够提供包括径向速度和频谱宽度等信息在内的几十种气象产品, 还具备一定的晴空探测能力。组成探测空间相互衔接覆盖的雷达监测网, 可对暴雨、热带气旋、龙卷、冰雹等灾害性天气进行有效的监测和预警。新一代天气雷达探测能力强、信息量大、智能化程度高, 能全天候工作, 时时刻刻监视天气系统演变状况。所以, 多普勒天气雷达是目前监测中小尺度灾害性天气最有效、最先进的手段之一。

目前国内的多普勒天气雷达资料的使用在方法上主要集中在各种算法所用适配参数的本地化, 处于引进消化阶段。国内14所、38所、784厂等单位提供的雷达资料处理程序, 数据格式及计算方法各不相同。中美合资敏视达公司生产的天气雷达, 其气象产品沿用美国的WSR-88D的数据处理算法[1], 产品较为丰富。该数据处理软件为国外20世纪80年代初基于封闭式小型计算机硬件平台设计而成, 将算法固定于处理程序中, 没有提供更多的技术细节和开发接口, 在程序结构上缺乏灵活性和扩充性, 用户无法进行数值产品的进一步开发。在软件组织上, 目前的雷达软件都是将功能封装在主程序里, 缺乏二次开发的接口, 随着雷达气象学研究的深入, 用户无法将最新的研究成果完全融合到天气雷达数据实时处理系统中, 这种状况无疑将限

收稿日期: 2003-03-24; 改回日期: 2003-05-20
基金项目: 国家自然科学基金资助项目(40275010)
作者简介: 沃伟峰(1977-), 男, 浙江宁波人, 硕士.

制天气雷达潜能的进一步挖掘。

随着雷达资料应用的进一步深入及雷达气象学的发展, 无论业务应用还是科学研究均迫切需要有一种通用型的雷达软件, 它能够独立于雷达本身, 作为一套单独的数据处理系统, 能够兼容多种数据格式。软件不仅要能在业务上稳定使用, 并且需要有良好的扩充性和定制性, 具有开放型结构, 可以随着业务人员素质的提高, 由业务人员根据当地天气的特点适当修改产品生成参数, 使二次产品能符合当地天气的特点, 同时要预留二次开发接口使其能够进行再度开发, 可以作为研究人员研究新的算法、开发新产品的平台。在设计上, 这样的雷达软件不仅要考虑到实用和业务化, 同时要考虑到软件通用性、兼容性, 代码的模块化、可复用性以及工程的可管理性和可持续开发性, 能够在基本平台不变的情况下根据软硬件的发展而升级及扩充其处理能力。

本文介绍的插件系统(Plug-in System) 中, 各个产品的算法程序放在动态链接库中, 独立于主系统, 并以插件的方式和主系统挂接。实现了产品处理过程组件化, 使系统具有很强的定制特性。并且提供了插件的接口函数和开发模板, 有一定编程基础的用户能够在这一平台上进行新算法的研究和新产品的开发。

1　设计思路

过去的软件都是将数据的接收和处理包含在一个程序里, 没有给二次开发预留出处理函数接口。随着计算机技术的发展, 软件的开发过程走向开放式, 从源代码级的合作开发走向以提供数据接口为主的分布式开发, 也就是从设计时模块的合作方式走向运行时(run-time) 模块的合作方式。开发人员只要理解处理过程中模块之间的函数接口和数据结构, 不需要知道整个系统中的全部数据处理的流程就可以编制和修改数据处理模块, 这样便于基于该系统进行二次开发。

本系统设计时采用动态链接库技术, 每一个动态链接库是一个雷达二次产品生成的子系统, 由主系统将雷达数据读入, 然后将数据指针传送给每一个子系统, 子系统取得数据的指针后对数据进行处理, 最后将结果输出。系统总体结构如图1 所示。

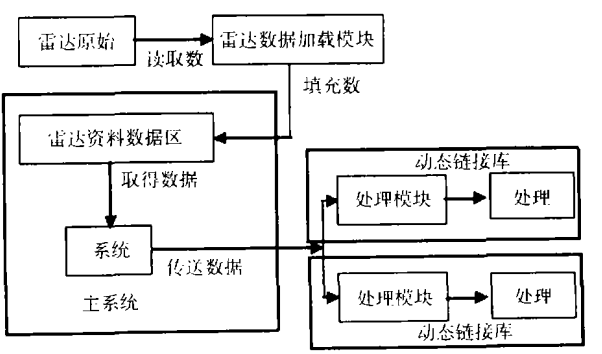

图1　系统结构示意

Fig. 1　System sketch map

在设计中, 插件的各个处理模块共享同一个数据区, 读取该数据区的数据进行处理, 然后各自输出结果。在主系统中, 对处理模块进行自动搜索, 通过动态链接库的动态加载方式, 使用插件中约定的接口函数, 使主系统能够正常调用各个模块的处理功能。

该设计使系统明确地模块化,运行时控制部分和各个处理模块分离,控制系统本身只做运行时的调度和运行状况的维护,任何数据进入数据区后即交给处理模块进行处理,输出结果。

2 动态链接库

实现插件系统的核心是运用动态链接库。动态链接库(DLL)是由全局函数、编辑函数及资源组成,作为程序运行的一部分。DLL有各种导出函数,客户程序(将DLL加载在第1位置)导出这些函数。Windows在加载DLL时将输入和导出匹配。使用动态链接库是模块化程序的一个重要手段。用动态链接库作为模块组织系统和用类作为模块组织系统不同,类是设计时模块,是代码,每次更改后都要重建和测试,DLL是运行时的模块,是二进制的执行文件,可以单独编译,测试,然后放到系统组织中去,这样就提供了一个可行性开放式结构,每一个处理模块可以单独设计成动态链接库,然后再挂接到主系统上去,让主系统来调用它,在不改变主系统的情况下增强或扩展系统的处理能力。

动态链接库有两种链接方式:隐式链接和显式链接[2]。在隐式链接中,需要指定随DLL建立的时候,链接程序产生的伴随输入的Lib文件。显式链接通过Win32的LoadLibrary函数来加载指定的动态链接库,用GetProcAdreess来取得函数地址,不必使用Lib输入文件。使用隐式链接,在主程序加载时所有DLL都被加载。使用显示链接时,可以决定哪些DLL加载,哪些不加载。因此,在需要程序具有动态扩展功能的时候,必须使用显式加载的方式。这样就可以通过一个DLL的列表来决定加载哪些DLL文件。通过对这个列表文件的编辑,实现模块的增加和删减,从而达到程序处理功能的增减。

3 插件系统的设计

3.1 插件概念

插件(Plug-in)是当前很流行的一种模块化软件设计方式,比如广为流传的Mp3播放器winamp就使用了插件技术。插件技术基于动态链接库的显式链接方式,作为插件的DLL文件一般有一个同名的函数接口,主程序通过该接口把数据送入DLL中,然后DLL处理完数据,通过接口把数据结果返回给主程序。

对于插件系统,主系统并不需要知道插件具体工作的内容,只要与插件有个预先约定接口便可。将资料的读取放在主系统中,而资料的处理和二次产品的生成放在插件中,主系统读入资料后将数据区的指针传给各个插件,然后执行各个插件中的同名接口函数,在接口函数中调用插件内部的处理程序对数据进行处理,最后输出结果。如在PPI插件中,接口函数对体扫资料作PPI处理,最后输出PPI产品文件,在VAD插件中,接口函数对体扫资料作VAD处理,最后输出VAD产品文件。

3.2 插件设计

设计插件系统的时候,关键是定义好接口函数。接口函数的定义有两种组织方法,一种是定义一组导出函数,在主程序中分别调用这些导出函数;另一种是定义一个结构体,将导出函数封装在结构体里。前者比较直观,后者比较简洁。本文采用前一方案,定义了一组导出函数。出于模块化的目的,将所有的数据处理功能都封装在一个类里,而导出函数只是对这个类的一个实例的操作。在这个插件系统中,输入是雷达资料的数据指针,输出是产品文件,每一个插件有一个对应的配置文件来记录模块需要的各种参数。

插件的结构由4个部分组成的:(1)应用程序DLLSAMPLE类,这是由VC系统产生的

DLL 主框架。(2)产品处理类DLLCLASS,所有的数据处理都在这个类里面。(3)参数配置对话框类Configure,用来配置产品生成的参数。(4)产品的数据结构定义。插件的核心是数据处理类和全局的导出函数。

数据处理的过程都被封装在 DLLCLASS 类里面,这个类包括了一些基本的数据处理函数。这些函数列举如下:

(1)数据指针设置函数:

void SetDataPoint(void* buffer)

void SetHeadPoint(void* buffer)

这两个用来设置资料的数据区指针和文件头指针,用来从外部传入数据所在的位置。

(2)数据读取函数

RadarDataHead* GetHeadInfo()

int GetLayerNumber()

LineDataBlock* GetLineData(int Angle, int SweepNo)

char* GetCircleData(int BinNo, int SweepNo, int type)

char GetPointData(int Angle, int BinNo, int SweepNo, int type)

这些函数是用来读取指定位置的雷达数据值,方便计算,可以指定仰角、方位角和距离库等参数以取得需要的数据。其中RadarDataHead 是处理时使用的雷达数据格式中的数据头定义,LineDataBlock 是径向数据记录的定义,是在黄淮暴雨洪水监测项目的雷达数据标准上定义的。

(3)产品处理函数

int MakeProduct(LPARAM lParam, WPARAM wParam)

这个函数用来启动数据处理和产品生成的过程。

(1)与(3)所列函数是公有成员,(2)所列函数是私有成员。外部的导出函数通过(1)所列函数将数据指针传给处理类,通过(3)所列函数进行数据处理和产品生成,在每个处理模块中,都有一个全局的产品处理类的变量,由导出函数控制该变量的行为。

导出函数是主程序和插件之间的对话接口,也就是模块的"行动"指令。在设计中,定义了以下几种导出函数:

(1)初始化函数

void Init(void* head, void* data)

在执行其他指令前,要用初始化函数将数据传入到产品处理类中。

(2)模块配置和描述函数

BOOL Configure(HWND hWndParent)

这个函数调用模块内的配置对话框,对模块对应的配置文件进行修改。

void ModuleDescribe(char* result)

这个函数用来得到模块的描述信息:模块的名称、功能、作者等。

void GetConfigFile(char* result)

这个函数用来得到模块相对应的配置文件名称。

int ModuleType(void)

这个函数用来得到模块的类型。

(3)产品指令函数

int MakeProduction(LPARAM lParam, WPARAM wParam)

这个函数用来读取处理参数及执行产品处理类的产品生成函数,输出产品文件或者返回处理出错的错误值。

通过以上这些函数,主系统可以调用模块传送数据,生成产品,这些函数就是约定好的接口函数,任何一个模块必须拥有这些接口函数才能保证该模块能够顺利加入到系统中并被主模块调用。

4 数据的输入和输出及处理流程

每一个插件可以看作一个独立的数据处理器,插件处理数据的过程为:接收数据,处理数据,输出数据。接收数据是从主系统获得数据,处理数据是插件内部对数据进行处理,输出数据是将处理完的数据输出或者返回给主程序。具体到这个系统中,接收数据是从主程序中得到数据指针,处理数据是进行产品处理,输出数据是将产品数据输出到指定的目录。

图2表示了插件系统中数据流的方向:数据进入主程序后被送往插件加工,如果数据要送回主程序显示或者其他处理,则通过回传主程序送回调用系统;如果数据不需要回送,则直接通过插件输出到文件。

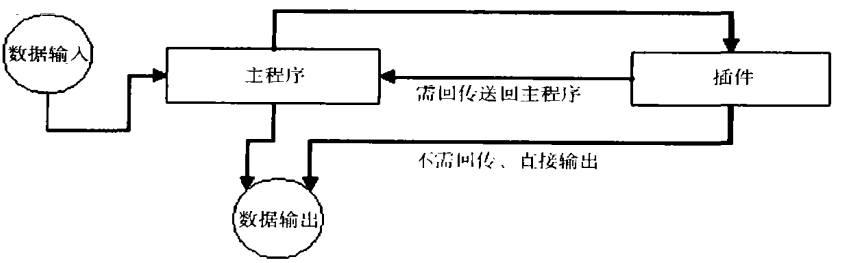

图2 插件数据流

Fig.2 Data stream in plug-in

在这个系统设计中,主系统和插件之间有着数据指针的交换、指令发送和接受。插件和主系统之间的数据交换和处理的流程如图3所示。

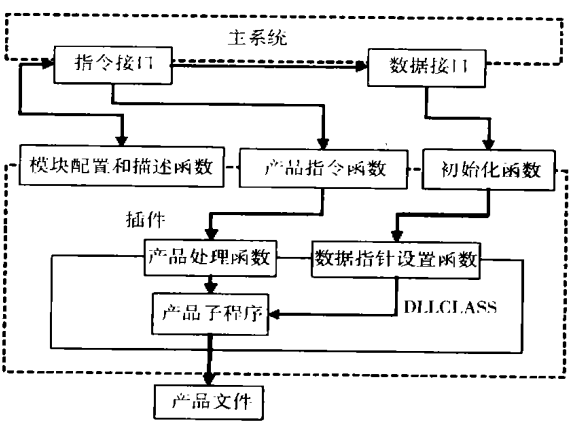

图3 模块数据流

Fig.3 Data stream in module

主系统调用模块的时候, 在配置和查询分支, 指令接口可以通过插件的模块配置函数和描述函数发出配置命令, 调用模块内部的配置界面, 以及查询模块的相应信息。在产品生成分支, 指令接口通过数据接口将数据指针送入插件内部, 然后调用产品指令函数, 产品指令函数调用内部DLLCLASS 的产品处理函数, 在类的产品处理函数中使用产品子程序生成产品。

在数据交换过程中, 主系统通过数据接口, 把数据区的指针送到插件的初始化函数, 插件的初始化函数再传给内部DLLCLASS 的数据指针设置函数, 这样内部处理的时候, 产品子程序就可以通过该指针取得数据进行计算, 输出产品文件。

5　系统对接方式

5.1　系统和插件的组织

实际应用中, 作为插件系统, 应该具有插件自动搜寻和插件的管理功能。设计中, 系统的目录树中包含了一个插件目录, 并有两个列表: 插件总列表和使用中的插件列表。每次系统启动后自动搜寻该目录下的插件文件(动态链接库) , 然后生成一个总列表, 用户可以维护插件使用列表, 以决定使用插件或者不使用哪些插件。在处理数据时, 系统读取使用中的插件列表并调用列表中的动态链接库处理数据, 对于未在使用列表中的插件, 系统不调用。

实际应用中设计了一个插件管理对话框, 如图4 所示。

图4　插件管理对话框

Fig. 4　Plug-in manage dialog

左边的列表表示未被加入使用列表的插件, 右边是在使用列表中的插件, 通过校验模块按钮来检查插件内部函数接口是否齐全, 通过配置模块按钮来调用插件内部的参数配置对话框。

5.2　插件的调用方式

程序运行的时候, 主程序通过系统函数LoadLibrary 来加载插件, 该函数的参数是动态链接库文件名, 返回一个动态链接库的句柄。在加载完一个动态链接库后, 通过系统函数GetProcAdreess 得到接口函数地址, 这个函数的参数是动态链接库句柄和函数名, 动态链接库的句柄是由上述的LoadLibrary 得到, 函数名就是定义了的一系列的接口函数。在用完一个动态链接库后, 需要用系统函数FreeLibrary 释放掉动态链接库, 该函数的参数是加载了的动态链接库句柄。

从上面可以看到, 只要更改LoadLibrary 函数的参数, 也就是传入不同的动态链接库文件

名,只要后面各个库拥有相同的接口函数,就可以使用不同的动态链接库产生不同的处理结果,而接口函数是插件系统中约定好的函数名称,所以在这种情况下,只需要做一个循环进行批处理,将使用中的模块列表各个文件名依次送入,然后依次执行其中的初始化函数、产品指令函数,就可以完成一系列的产品生成。每次有符合规格的新的产品制作插件时,只要放到插件的目录下,并加入插件使用列表中,就可以使系统具有这种新产品的生成功能。因此从这个角度来看,这个系统是开放的,主系统并不涉及产品的生成,而只是一个数据的读取和预处理程序,同时调度插件,所有产品生成都是靠外挂的插件来完成的。通过对插件的管理,可以使系统具有不同的产品生成能力。这个过程的流程见图5。

从图5可以看出,系统首先加载需要处理的资料,然后加载使用模块列表,接着以使用中的模块总数为循环数,进入一个循环,循环中首先加载模块,然后检查初始化函数和产品指令函数是否正常,如果不正常则进入下一个循环,否则就将数据传入模块并执行产品生成指令,完成后释放掉模块,继续加载新的模块处理一直到最后一个模块。

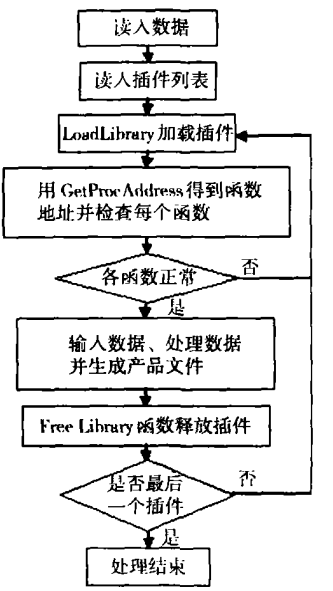

图5 模块处理流程

Fig. 5 Module processing flow

6 实际使用情况

在1427工程C波段多普勒天气雷达项目中,使用了该项技术。这是一个多普勒天气雷达的二次产品处理软件,从多普勒雷达的资料可以计算得到丰富的雷达二次产品,从软件的组织和维护角度出发,将各个二次产品的算法分类和模块化是必要的。该软件是多人合作开发的,模块的划分和集成就显得很重要,该系统在划分的时候各个算法的模块独立性很强,都是有自己的数据流,不相互干涉,因此使用插件的方法来组织算法模块。最后在集成的时候,开发出来的组件以插件的方式嵌入到系统中。调试过程对于某一产品的修正和进一步研究只需更改生成该产品的插件,不会影响到系统其他产品功能。

实际运行中主框架代码较为简洁,主框架建立一个不断检查新数据和资料处理请求的时钟,当有新资料到达或有处理请求的时候,启动一个线程,该线程读取资料并按要求调用相应的处理模块,生成产品,放入数据库,最后通知终端处理完成。如果某一插件出错,主框架将略过该模块,并且将出错信息记录在日志文件里,以备调试使用,系统仍可正常运行。程序经过多次测试,表明插件式的系统开发有利于分布式开发,模块化程度高,集成方便,维护和管理较为简洁,确定错误点较为方便。该系统已应用于1427工程。图6是系统中一个PPI产品样本。

长期联机运行结果表明,该系统稳定可靠,维护方便,以插件作为数据处理的组织方式可以很好地使程序模块化,并具有很大的独立性,可以单独进行维护。各个插件的功能明确,流程清晰,不同的插件搭配方式使系统的处理能力具有很大的伸缩性和灵活性。

在后续开发工作中,如果取消配置对话框,使用另外的配置工具维护配置文件,动态链接库的代码以ANSI C工程的方式重组织代码,可以减少平台依赖性,提高模块的移植性,便于放到Linux操作系统下重新编译,使用Linux的动态链接库系统同样可以实现插件功能。后续

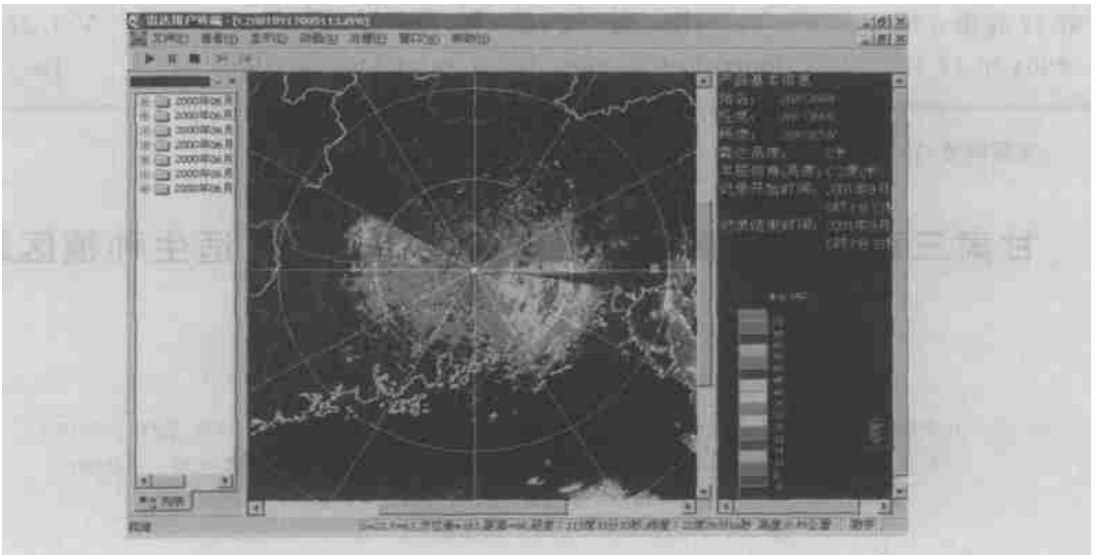

图6　产品样本

Fig. 6　Product sample

开发中可能扩展接口函数, 接口函数扩展时原来已有的接口应该保持不变, 在此基础上增加新的函数, 便于向下兼容。

参考文献:

[1]　胡　胜. 多普勒天气雷达产品软件的开发与设计[D]. 南京: 南京气象学院大气物理系, 2000.

[2]　Kruglinski David, Wingo Scot, Shepherd George. Visual C++ 6.0 技术内幕(第5版)[M]. 北京: 北京希望电子出版社, 1999: 509-524.

Application of Plug-in Technique in Radar Software Design

WO Wei-feng,　WU Lei,　GU Song-shan

(Department of Electronic Engineering, NUIST, Nanjing　210044, China)

Abstract: This paper introduces a plug-in system based on dynamic-link library technology in the software design of Second Product processing Doppler Weather Radars. Each product is designed as a plug-in separated from the main system, so researchers can concentrate on how to implement product algorithm. It also makes this software well organized and modularized. This project is easy to manage and further develop. This software also provides a platform for new product research and development.

Key words: plug-in; Doppler Weather Radar second product; radar software; dynamic-link library