基于动态库的星载软件可重构设计与实现

白 亮[1,2],邱 源[1,2],韦 杰[1,2],孙逸帆[1,2],高 洁[1,2]
(1.上海航天智能计算技术重点实验室,上海 201109;2.上海航天电子技术研究所,上海 201109)

摘 要: 针对卫星在轨运行、长期处于无人值守状态,主要依靠星载软件的安全性和可靠性来保证整星任务的稳定工作。复杂多变的空间环境可能会引起星载器件的异常变化,从而导致星载软件异常,甚至发生软件"衰老"。本文在分析现有可重构方案基础上,提出了一种利用动态库的静态链接方式实现在轨可重构的方案,针对存在软件缺陷,或者需要功能升级和拓展的模块,利用遥控上注手段,采用MD5算法对数据文件完整性校验通过后,写入文件系统,并对原动态库文件作备份处理,以便版本回退。以具体实例对本文所述方案的可行性和有效性进行验证,结果表明:在嵌入式操作系统架构下,利用本方案实现星载软件可重构,能够有效提升星载软件在轨实施可重构的可靠性和安全性,进一步为星载软件的扩展和灵活应用提供支撑。

关键词: 嵌入式操作系统; 可重构; 动态库; 星载软件; 可靠性
中图分类号: TN 911.73; TP 391.9 **文献标志码:** A **DOI:** 10.19328/j.cnki.2096-8655.2021.04.012

Design and Implementation of On-board Software Reconfiguration Based on Dynamic Library

BAI Liang[1,2], QIU Yuan[1,2], WEI Jie[1,2], SUN Yifan[1,2], GAO Jie[1,2]
(1.Key Laboratory of Intelligent Computing Technology (SAST), Shanghai 201109, China;
2.Shanghai Aerospace Electronic Technology Institute, Shanghai 201109, China)

Abstract: It is a fact that satellite is operating in orbit and unattended for a long time, and the stable operation of the whole satellite mission mainly relies on the robustness and reliability of the on-board software. The complex and changeable space environment may cause abnormal changes of on-board devices, which may result in abnormality of on-board software and even software aging. In this paper, based on the analysis of the existing reconfiguration schemes, a scheme is proposed to realize on-board reconfiguration by using the static link mode of the dynamic library. For modules which have software bugs or need functional upgrades and extensions, the MD5 algorithm is used to verify the integrity of the data file by using the remote control method. Then, the data file is written into the file system, and the original dynamic library file is backed up for version fallback. The feasibility and effectiveness of the scheme are verified by a specific example. The results show that, under the embedded operating system architecture, using the proposed scheme to realize the reconfiguration of on-board software can effectively improve the reliability and security of the on-board software reconfiguration, and further provide support for the expansion and flexible application of on-board software.

Key words: embedded operating system; reconfiguration; dynamic library; on-board software; reliability

0 引言

星载嵌入式操作系统处于远程遥控的环境中,其软件的维护与修复相较于本地系统故障恢复要困难许多,无法进行手工维护,只能依靠操作系统的自我维护功能,以及接受地面上传的有限维护指令进行运行过程中的动态自我修复和更新。

针对星载嵌入式操作系统上运行的软件在长时间的运行过程中不可避免出现"软件衰老"现

收稿日期: 2021-03-23; **修回日期:** 2021-06-18
基金项目: 科技部国家重点研发计划(2016YFB0501004)
作者简介: 白 亮(1987—),男,硕士,高级工程师,主要研究方向为星载嵌入式软件设计。

象[1],即伴随着软件的运行,系统资源逐渐耗尽或运行错误逐渐积累所导致的系统性能持续下降甚至停机的现象,采取在轨可重构措施来恢复软件性能。

星载嵌入式操作系统工作于恶劣环境中,星上的设备暴露于外太空,易受到外部环境的影响,从而对系统产生影响。而操作系统负责管理平台所有外部设备,系统的可靠性和安全性对操作系统的管理具有依赖性[2]。而且,操作系统在运行过程中,也会暴露出一定的设计缺陷[3],或是测试过程中没有发现和排除潜在设计缺陷。但是可以通过在轨可重构技术对软件进行修正与完善,进一步提高星载系统的安全性和可靠性。同时,还可以针对在轨运行卫星,通过可重构的方式,基于现有功能基础,实现卫星功能的升级和扩展,实现卫星的“App”(应用程序)式的快速开发和快速集成。

随着计算机技术的发展,基于动态库的可重构技术被广泛应用于商业等各个领域,主要由于动态链接库与主程序之间通过接口调用产生依赖关系,只要保证动态链接库输出接口不变,更换动态库不会对主程序造成任何影响,可以提高程序的可维护性和可扩展性;而且不同编程语言编写的程序,只要遵守相同的应用程序调用约定(Application Binary Interface, ABI),就可以调用同一个动态链接库;使用动态链接库,适用于大规模的软件开发,使得开发过程独立、耦合度小,便于多人团队间进行协同开发和测试;同时,多个应用程序使用相同的动态链接库时,在磁盘中可以只存在一份动态链接库文件,相比其他技术,可以节约磁盘空间。基于上述优点,在现代化大型软件开发中,大量应用动态链接库技术。而且利用动态链接库技术的一个最大优点在于,针对动态链接库更新升级后不用重新启动主程序,只需将更新的动态链接库重新载入程序的内存空间即可。基于该特点,将动态链接库技术应用于星载嵌入式软件的在轨重构实现中,既可以达到上述所有优势,又不需要星载系统软件重新启动,即在轨重构的实现不会对既定星载任务(如程控、数传、热控等任务)运行产生影响。基于动态链接库技术,在轨重构包是以文件的形式存在,即使卫星发生异常重启后,前一次在轨重构生成的补丁包仍然在,且重启运行后,直接使用重构后的程序,使得在轨重构技术对卫星可靠安全在轨运行提供支撑。

随着星载系统的发展,星载嵌入式软件在轨可重构技术也在持续不断发展。目前常用的在轨可重构技术有基于RAM型和微重启等。随着高性能处理器的发展,传统在轨可重构方式能否在高性能处理器平台上继续适用,需要作全面评估和验证测试。本文所提出的方案,以高性能处理器平台为出发点,进行设计、实现和验证。针对在轨卫星通过实际在轨可重构操作后,进一步验证本方案的有效性、安全性和可靠性。

1　基于星载嵌入式操作系统的可重构软件框架

本文中设计了一种基于星载嵌入式操作系统的可重构软件框架,嵌入式操作系统的层次如图1所示。本软件框架实现的硬件平台基于国产化处理器国微SM750,操作系统为风云翼辉(AIC-OS)嵌入式操作系统1.11.0,开发环境为RealEvo-IDE 3.9.10。整个系统可分为4个层次:处理器层、操作系统内核服务及BSP层、中间件层和应用层。其中,在本系统中中间件层由动态链接库的形式来实现。应用层由嵌入式软件框架和各个应用组件构成,在不同时刻加载不同应用组件时,应用层功能将可以随之发生变化而无需重新加电或复位操作系统,从而实现了嵌入式软件的功能可重构。而可重构软件框架作为一个中间层的形式运行在操作系统与应用组件之间。一方面通过对应用组件的动态加卸载、系统资源管理、多组件管理等功能实现了嵌入式软件功能的可重构;另一方面它为应用组件屏蔽了底层细节,使之与硬件和操作系统隔离,从而可以实现组件的二进制级复用。针对通用功能组件,无需重新修改和编译,直接通过多组件动态重构即可完成应用软件功能的重新定义,降低开发成本,同时缩短软件开发周期。

2　星载软件可重构的实现方案

2.1　利用嵌入式操作系统的微重启实现可重构方案

嵌入式操作系统微重启的可重构方案采取可递归恢复框架实现,具体由故障检测代理、恢复管理器和恢复代理3个模块组成,如图2所示。故障

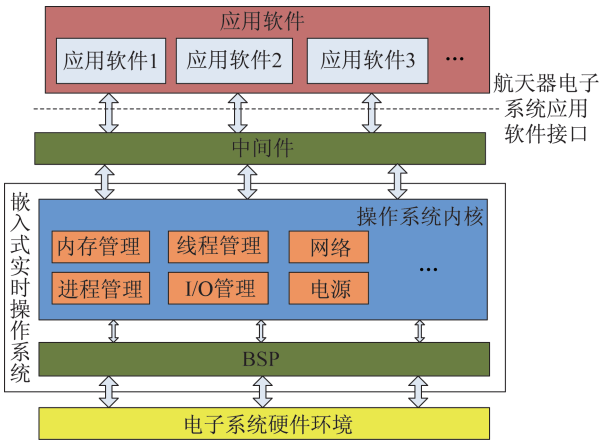

图 1　星载软件可重构框架层次图

Fig.1　Hierarchical diagram of reconfigurable framework for on-board software

监测代理负责实时监测系统的运行状态,查找系统故障并将相关信息传送给恢复管理器,故障监测代理将所有系统运行状态的变化信息递交给恢复管理器[4]。恢复管理器动态地保存系统的故障传播信息,当它接收到从检测代理发送过来的故障信息时,采用自动故障路径推断(Automatic Failure-Path Inference,AFPI)恢复策略,针对给定的递归重启树,计算最优的递归恢复路径[5],并触发恢复代理,实现组件或者嵌入式操作系统的微重启操作。通过裁剪嵌入式操作系统内核,将嵌入式操作系统原有功能模块上移至用户态组件层,同时确保裁剪后的嵌入式操作系统内核能够稳定运行。将微重启功能组件置于内核内,使微重启功能组件的权限高于功能性组件,从而有效地控制功能性组件的微重启动作。

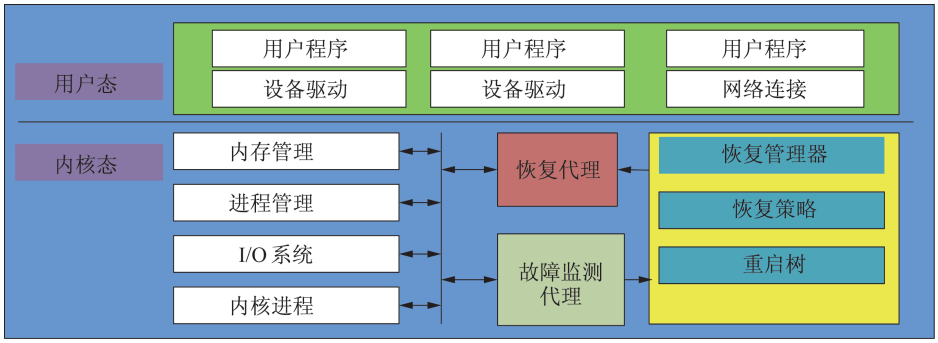

图 2　嵌入式操作系统微重启的可重构实现方案

Fig.2　Reconfigurable scheme for embedded operating system micro-restart

2.2　基于 RAM 型的软件动态升级实现可重构方案

基于 RAM 型的软件动态升级主要过程如下:在轨运行系统中发现软件故障或软件缺陷后,由开发人员在软件开发机中修改软件,将修改信息形成软件补丁,通过在地面目标机中进行模拟运行测试,验证本次形成的软件补丁的有效性和正确性;然后提交到地面监控系统,由地面监控系统发送至在轨设备上;接着在轨设备对在轨系统进行动态修改,通过反馈至地面监控系统在轨可重构的信息。

为了满足上述需求,设计了在轨软件维护系统,其功能分解如图 3 所示,3 个功能模块分别为:1) 补丁生成模块,包括补丁信息获取、补丁创建和补丁发送等;2) 在轨修改模块,包括补丁接收、变量修改、代码修改、任务处理等;3) 维护监控模块,包括维护信息反馈、反馈信息接收与处理等。

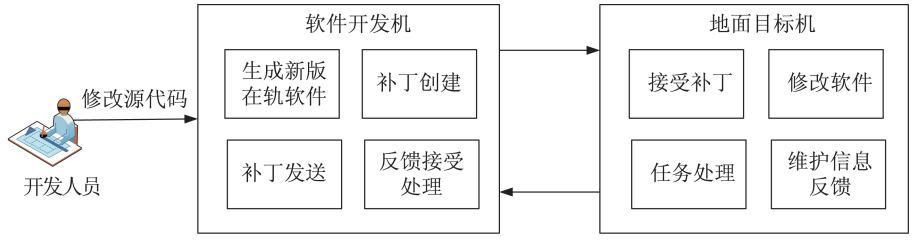

图 3　基于 RAM 型软件动态升级的可重构方案

Fig.3　Reconfigurable scheme based on the dynamic upgrade of RAM-type software

在图3中，基于RAM型的软件动态升级实现可重构方案由软件开发机和地面目标机两个子系统组成，整个系统各功能模块分布于各个子系统之上，开发机和地面目标机之间通过以太网或者串口进行通信。两个子系统协同工作，共同完成基于RAM型的软件动态升级可重构验证[6]。

2.3　利用动态库实现星载软件的可重构方案

星载软件在轨可重构是通过遥控注数的方式，将软件的可执行代码注入到星载计算机中，替换原有模块实现。风云翼辉嵌入式操作系统在SM750平台实现了虚拟内存管理技术，支持应用软件以动态加载的方式设计实现和运行。

动态加载意味着编译生成的可执行目标文件与地址无关，则可以利用动态链接库实现对星载应用软件中特定功能的封装[7]。将具有相对独立功能的模块编写为动态链接库，应用软件在实现需求中定义的功能时，为满足特定的功能，可将具有该功能的动态库链接进来，从而星载软件最后的形态是以主程序加若干动态链接库呈现。

动态库中只实现特定功能，其规模可以做得较小，从而减轻每次在轨上注时信道的压力。而且在功能升级或者缺陷修复实现可重构时，可以只针对单一模块进行，从而提高星载软件在轨可重构方案的安全性和可靠性。

基于动态库实现星载软件可重构流程主要包括重构软件准备、重构软件分发、重构软件加载和重构结果反馈4个步骤。软件重构流程如图4所示。

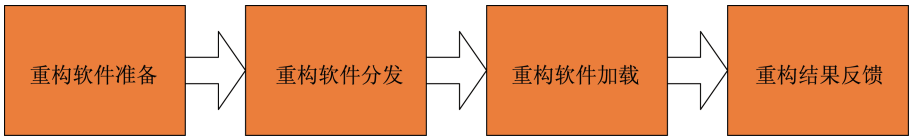

图4　基于动态库实现可重构方案

Fig.4　Reconfigurable scheme based on the dynamic library

2.4　可重构实现方案比较

相对于重启整个系统而言，微重启花费的代价较小。同时，重启单个服务的速度远比重启整个系统快得多，并且微重启仅对重启的服务和服务的进程造成影响，而不会影响到系统中其他服务的正常运行。但是该种方案仅适用于部分数据不正常，重启后修复原来错误的数据，而针对软件缺陷或者软件错误，该方案无法彻底修复。

基于RAM型的软件动态升级实现可重构过程的可靠性和安全性要求极高，一旦出现失误有可能导致整个系统软件的失效。

与地面系统不同，受天地通信速率、入站频率及过站时间等多方面因素的限制，星载软件的在轨升级很难达到与地面软件升级相同的实时性。而且，上注的星载软件可重构的内容存储于内存中，软件复位后，可重构的内容不复存在，还需要重新上传。

基于动态库实现的可重构方案，将已存在于在轨设备上的不同模块进行动态的装载与卸载，以提供不同功能，或提高系统运行速度。借助文件系统随时停止、加载、运行某个或某几个特定模块，而不会影响系统当前运行的其他功能，且单个模块编译生成的目标文件都比较小，状态控制和维护相对比较容易。

通过上述分析可知：使用动态库中间件的形式应用于目前星载系统中，具有较高的可靠性和安全性。而且采用此方法实现星载软件可重构，具有较强的软件可扩展性，易于实现在轨星载软件功能的全部替换和特定功能的可重构。

3　星载软件可重构方案的设计与验证

3.1　可重构设计

3.1.1　方案设计

本设计中，采用基于动态库的形式实现星载软件的可重构，利用嵌入式操作系统提供的动态加载功能，对在轨星载软件的特定功能实现可重构，修复已有的软件缺陷或者漏洞。

星载软件在设计初时，已经将部分功能相对独立的部分编写成可链接的动态库，后续在轨可重构既可以针对每个独立的动态库，也可以针对主模块进行重构，设计方案如图5所示。

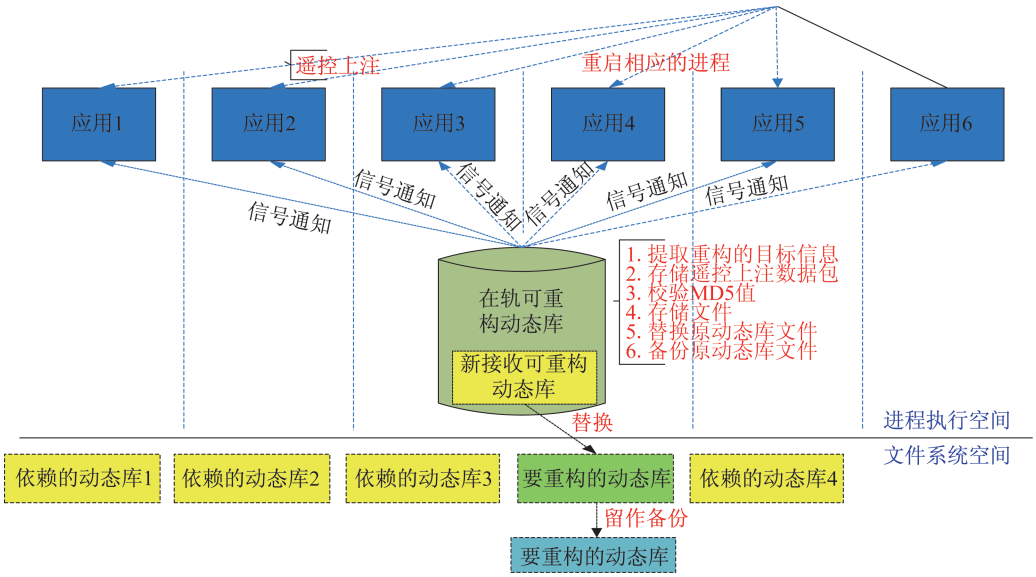

图 5　在轨可重构设计方案

Fig.5　On-orbit reconfigurable design scheme

3.1.2　数据包格式设计

根据 3.1.1 节方案所述，针对需要实施在轨可重构的模块，编译生成可执行目标文件后，需要将该文件中的二进制数据按着标准国际空间数据系统咨询委员会(Consultative Committee for Space Data Systems，CCSDS)协议封装为遥控包。为可重构设计了对应的上注重构数据包格式如图 6 所示。

含义	……	遥控包序号	……	所属进程索引	文件索引	MD5校验码	文件数据
字节数	……	2(首包)	……	4	4	16	223

含义	……	遥控包序号	……	文件数据
字节数	……	2(中间包)	……	247

含义	……	遥控包序号	……	文件数据
字节数	……	2(末包)	……	*N*

图 6　可重构上注数据包格式

Fig.6　Reconfigurable upload packet format

在图 6 中，首包格式同后续包差异较大，将地面生成的可执行目标文件所属的进程编号、文件索引，以及文件的消息摘要算法(Message Digest Algorithm MD5，MD5)值打入首包。将这些信息上注给星载计算机，作为后续文件完整性校验的依据。

引用该算法，是因为 MD5 算法可以将任意长度的输入串经过计算得到固定长度的输出，而且只有在明文相同的情况下，才能等到相同的密文，并且这个算法是不可逆的，即便得到了加密以后的密文，也不可能通过解密算法反算出明文。针对文件完整性和安全性时，具有突出的优点。

在上注的每一个数据包中都有包序号，作为后续文件完整性校验和接收结束的依据。当接收结束后，通过逐一判断接收到包序号，当发现有缺包的情况时，会将缺少的包序号打入遥测量下传，地面可以通过该遥测量确定补发动作。

3.1.3　实施流程

可重构数据包生成后，将这些数据包当作正常的遥控注数包，当卫星在测控弧段内时，上注给需要重构的在轨卫星，具体的实施流程如图 7 所示。

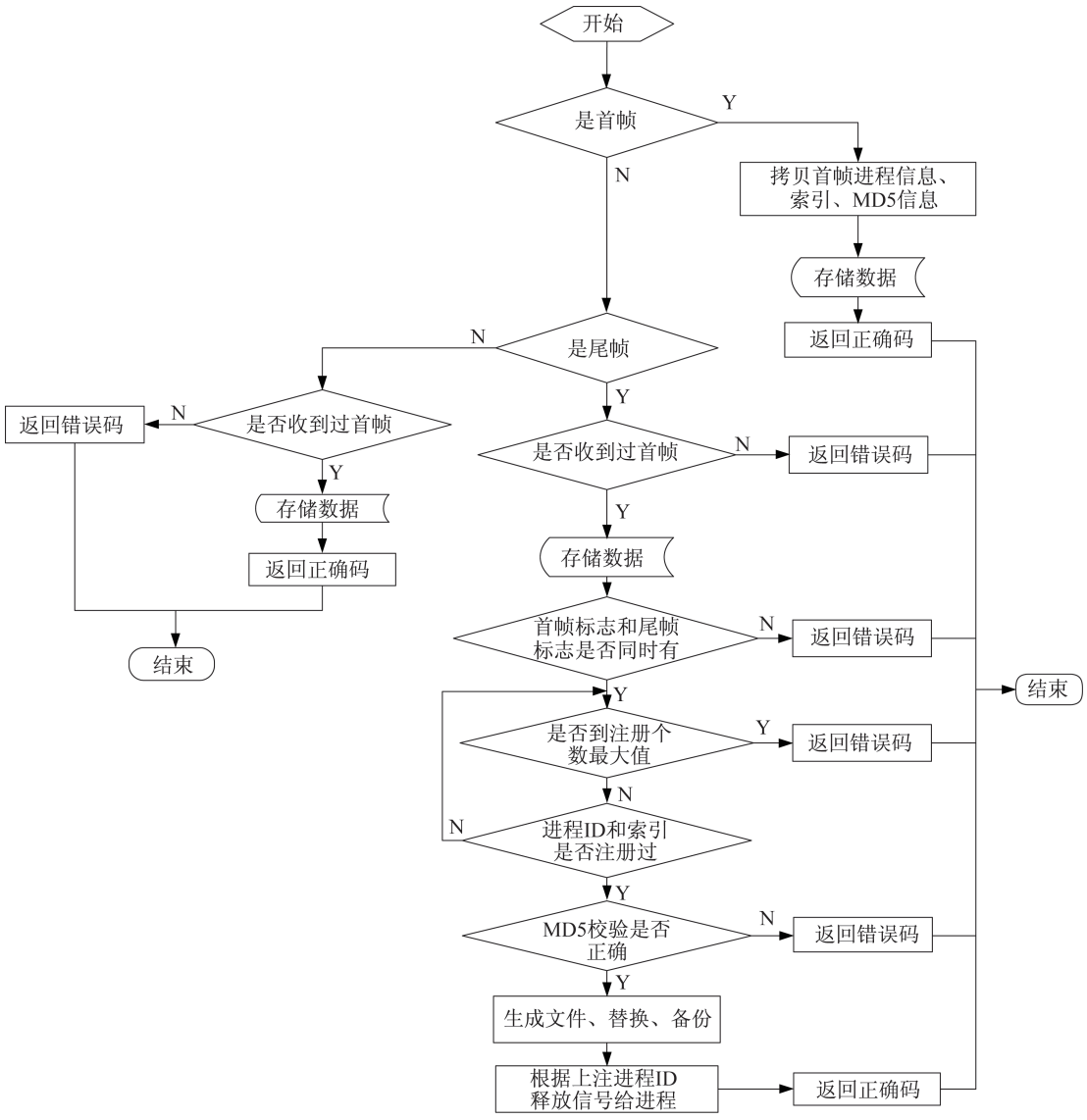

图 7　可重构实施流程

Fig.7　Reconfigurable implementation flow chart

3.1.4　可靠性和安全性保证

卫星在轨自主运行时,无法像地面测试阶段实时监视其运行状态,只能通过有限的下行遥测量监控其运行状态[8]。本方案设计中,设计了相应的可重构模块的遥测量,包括正确包计数、错误包计数、错误包编号、可重构的模块编号等遥测量。在遥控上注过程中,可以通过实时遥测查看当前在轨可重构情况。当有数据包校验错误,会将错误包的包号下传,此时,地面只需要补发错误的单个数据包,即可修复上注错误。

在上注的每包遥控包末尾都有相应的CRC值校验,可以完成本包数据的正确性校验。上注的数据并不是立即写入文件系统,而是暂存在内存空间,当接收完全部的数据包后,通过对整个文件的完整性校验通过后,再将全部数据写入文件系统,形成对应的文件,保证生成的文件的完整性。

每次实施在轨重构时,都会将之前的文件存储为备份版本,并不会自主删除。当启动重构后的模块运行后,通过下行遥测发现其功能不符合预期时,可以进行版本回退,保证星载系统软件的安全。

在文件系统中,针对每一份可执行目标文件,都有相应的出厂版本,当某次可重构实施过程中出现失误,造成星载系统软件损坏[9]。此时,将整星进行瞬时断电一次,全部可执行目标文件回到出厂版本,保证整星的安全。运行正常后,可以通过遥测

上注,指定单个的应用或者某个动态库使用最后可重构后的版本,启动最新功能。

3.2 验证实施

初始化时,针对需要在重构的模块函数进行注册,形成函数表[8]。在某一时刻,需要针对某一函数进行重构时,通过遥控命令的方式,完成新的模块函数编成动态链接库上载,星上系统根据相关信息打开动态链接库,映射对应函数,替换已经创建函数表中对应函数指针,完成可重构。

为实现在AIC-OS操作系统下的在轨可重构,需要把即将重构的模块编写到动态链接库中[10],生成一个动态链接库文件,通过上行通道,将生成的动态链接库文件上传到星载计算机的文件系统中。当计算机收到重构遥控命令时,将接收文件存储到文件系统中,并根据遥控命令类型,分别通知相应进程,同时分别打开动态链接库,实现对应函数的映射。

为实现将目前定义的确定函数调用同重构后对应的函数调用间的无缝衔接,特定义如图8所示的结构体。其中,name成员为该确定函数的名称,名称是实现正确映射的关键,所以名称一定要保证正确;ready标志只是用来表示是否映射成功,可以忽略不用;com_ptr为函数指针,用于表示在没有可重构之前的函数指针,类型为void*(*)(void*),即输入参数为void*类型,返回类型为void*类型的函数指针,将输入参数定义为void*类型的原因是为兼容不同函数对输入参数类型的限定。当有多个参数时,可以定义为结构体,调用函数时,只需要将该结构体传入即可。new_ptr类型同com_ptr,用于表示当发生重构时新映射的函数指针。

```
#ifndef DM_ONBOARD_H_
#define DM_ONBOARD_H_

#include "common.h"

typedef struct _onboard_func_obj {
    int8_t        name[30];
    int8_t        ready;
    func_ptr      com_ptr;
    func_ptr      new_ptr;
} onboard_func_obj_t, *ponboard_func_obj_t;

void  do_dm_register_onboard_func(ponboard_func_obj_t objp);
void *do_dm_common_func(void *argv);

#endif /* DM_ONBOARD_H_ */
```

图8 实现可重构的结构体

Fig.8 Reconfigurable structure

这样的结构定义可以实现对多个模块进行重构[11],将多个可重构对象组成一个数组,根据索引进行相应的替换,如onboard_func_obj_t onboard_func_obj[12]。当然,这个索引和对应的函数之前的关系需要自己定好。可重构模块调用示例如图9所示。

```
static void *DMHotCtrlTask(void *argv)
{
    sys_time_t st;

    for ( ; ; ) {
        st = get_sys_time();
        fprintf(stdout, "\033[%d;%dH \033[36m [%5d.%10d] DM Hot-Ctrl Task Running\r\n",
                curr_info_y_pos(), dm_info_x_start(), st.week, st.msec);
        inc_info_y_pos();
        up_symbol_obj.new_ptr(NULL);
        up_change_obj.new_ptr(NULL);
        sleep(3);
    }

    return (NULL);
}
```

图9 实现可重构的函数调用示例

Fig.9 Example of implementing a reconfigurable function call

可重构验证过程中,分别给对应进程使用了不同的动态链接库,即各自使用一个动态链接库文件。其中,一个进程中将要实现可重构的函数定义如图10所示。

```
static void *up_symbol_func(void *argv)
{
    uint32_t   left, right;
    uint32_t   result;
    sys_time_t st;

    srand(time(NULL));

    st = get_sys_time();
    fprintf(stdout, "\033[%d;%dH \033[36m SO-DM-SYMBOL-MUL\r\n",
        curr_onboard_y_pos(),
        dm_onboard_x_start(),
        st.week, st.msec);
    inc_onboard_y_pos();

    left   = rand() % 100;
    right  = rand() % 120;
    result = left * right;

    return (NULL);
}

static void *up_change_func(void *argv)
{
    sys_time_t st;

    st = get_sys_time();
    fprintf(stdout, "\033[%d;%dH \033[36m SO-DM-CHANGE-A\r\n",
        curr_onboard_y_pos(),
        dm_onboard_x_start(),
        st.week, st.msec);
    inc_onboard_y_pos();
    st = get_sys_time();
    fprintf(stdout, "\033[%d;%dH \033[36m SO-DM-CHANGE-B\r\n",
        curr_onboard_y_pos(),
        dm_onboard_x_start(),
        st.week, st.msec);
    inc_onboard_y_pos();

    return (NULL);
}
```

图10 需要重构进程的原函数

Fig.10 Original function of the reconstruction process

当上位机发送完遥控数据包后,嵌入式操作系统负责监视遥控信息的线程将动态链接库文件写入文件系统中,主动通知需要重构的进程,当需要重构的进程收到该信号后,调用dlopen()打开动态链接库文件[12],对相应的函数进行映射,运行结果如图11所示。

图 11　实现可重构后的运行结果

Fig.11　Results after reconfiguration

运行结果显示,利用动态库形式,能够较便捷地实现星载软件的可重构。

4　结束语

星载软件的可重构机制对提高整星系统安全性和可靠性有着举足轻重的作用,并为卫星长期在轨稳定运行和在轨任务的扩展提供保障。本文对3种实现星载软件可重构方案的进行优缺点比较,选择利用动态库形式实现。详细阐述了基于动态库形式的星载软件可重构实现方案、数据包格式、实施流程及实施过程中可靠性和安全性的保证。通过具体的实例,完成基于动态库的星载软件可重构验证。结果表明:基于动态库形式的可重构机制的运用,可以有效丰富和扩展星载软件开发模式和架构设计,为未来实现“App”式的星载软件开发奠定基础。本重构实现方案可以有效提升在轨卫星实现星载软件可重构过程中的安全性和可靠性。

参考文献

[1] 赵云山.基于符号分析的静态缺陷检测技术研究[D].北京:北京邮电大学,2012.

[2] 钱方亮,林荣锋,周宇,等.一种基于微小卫星系统软件在轨编程功能的设计方法[J].计算机应用与软件,2018,35(12):16-20.

[3] 李兴伟,白博,周军.基于FPGA的立方星可重构星载处理系统研究[J].计算机测量与控制,2018,26(8):172-176.

[4] 陆峥,金光,杨天社,等.基于可重构度的在轨卫星多级健康评估方法[J].系统工程与电子技术,2018,40(8):1769-1776.

[5] 李丹丹,马金全,杨平平.信号处理平台中可重构组件的设计与实现[J].计算机工程,2017,44(5):33-39.

[6] 汪俊锋,叶函函,易维宁,等.在轨超高分辨率傅里叶光谱仪仪器线型函数更新方法研究[J].红外与毫米波学报,2018,37(5):613-620.

[7] 刘基余.四大系统的卫星导航电文概论:GNSS导航信号的收发问题之五[J].数字通信世界,2014(S1):1-10.

[8] 汪宏浩,王慧泉,金仲和.基于增量链接的可回滚星载软件在轨更新方法[J].浙江大学学报(工学版),2015,49(4):724-731.

[9] 温连强.卫星光通信系统软件在轨更新策略[D].哈尔滨:哈尔滨工业大学,2012.

[10] 韦涌泉,董振辉,张红军.一种基于文件的嵌入式星载软件在轨升级方法[J].单片机与嵌入式系统应用,2018,18(5):32-35.

[11] 史毅龙,薛长斌.基于“龙芯”的VxWorks系统函数在轨更新研究[J].电子设计工程,2015,23(21):106-109.

[12] 王战强,翟盛华.星载处理设备软件在轨重构技术研究[J].空间电子技术,2013,10(1):7-13,43.