第33卷 第21期 计 算 机 工 程 2007年11月
Vol.33 No.21 Computer Engineering November 2007
·软件技术与数据库· 文章编号：1000—3428(2007)21—0039—03 文献标识码：A 中图分类号：TP391

基于 uClinux 多类 IPC 的可靠嵌入式多任务软件

张凯龙，周兴社
（西北工业大学计算机学院，西安 710072）

摘 要：分析了 uClinux 中进程管理和多类 IPC 的特点，研究了基于 uClinux 进程机制的可靠嵌入式软件的设计方法，并提出了一种基于策略的多级故障自检测与自恢复机制。实际应用证明，该机制是有效的。
关键词：嵌入式；uClinux；IPC；多任务；自恢复

Reliable Embedded Multi-tasks Software with Multiple IPC Based on uClinux

ZHANG Kai-long, ZHOU Xing-she
(School of Computer, Northwestern Polytechnical University, Xi'an 710072)

【Abstract】 This paper analyses characteristics of the management of processes and characteristic of IPCs in uClinux, studies design method of reliable embedded software based on uClinux process mechanism, brings forward a multi-level mechanism of self fault-check and fault-recovery based on special policy. Practice shows that the mechanism is effective.
【Key words】 embedded; uClinux; IPC; multi-tasks; self-recovery

嵌入式操作系统 uClinux 主要针对无内存管理单元(MMU)，Flash 等资源受限的嵌入式系统而设计，目前已在很多嵌入式产品中得到了应用。uClinux 是在标准 Linux 基础上进行定制和优化所形成的嵌入式操作系统，具有标准 Linux 可靠、移植性好、网络功能强大等优点。由于 uClinux 中没有 MMU，其进程管理机制与标准 Linux 存在一定差异，因此基于 uClinux 的多任务嵌入式软件设计将更为复杂。

1 uCLinux 多进程管理及其特点

1.1 进程管理技术

uClinux 的进程管理沿用了标准 Linux 中的大部分机制，允许设计人员使用多数标准 Linux API 来控制特定进程的状态。Linux 内核为用户提供 fork( ),vfork( ),clone( )等接口函数来创建子进程，分别对应 sys_fork( ),sys_vfork( ),sys_clone( )等系统调用。其中，sys_fork( )调用创建的子进程是父进程的拷贝，除代码段外不共享任何内容，其独立性强、可靠性好；sys_vfork( )创建与父进程共享内存的子进程，子进程可能修改父进程的地址空间，导致父进程异常；sys_clone( )通过修改参数来指定父子进程间是否共享内存，进而实现 sys_fork( )或 sys_vfork( )的功能。

uClinux的多进程管理机制是标准Linux进程管理的子集和扩展，实现方式又与内存管理密切相关。没有MMU，意味着当所有程序运行时都将直接访问内存的物理地址且共享同一运行空间，这可能引起进程空间的非法占用，从而导致系统异常[1]。在加载可执行文件时，uClinux必须对其进行reloc等预处理。另外，uClinux进程创建方法的实现特点为：fork( )函数本质上就是vfork( )，系统中除init进程外的其他进程均通过vfork( )调用创建。如前所述，由于基于vfork( )创建的子进程共享父进程内存，因此在软件设计时需要采用信号量等方法对多进程间的数据进行保护，以保证系统的可靠性。

对于进程终止处理机制，uClinux与标准Linux也存在差异。在uClinux中，__exit_mm( )在释放子进程占据的内存空间之前，判断进程所占据存储空间的内容与初始状态是否相同，若不同则允许系统恢复到初始状态，将使系统更加安全，这是uClinux进程管理的另一特点[2]。该函数还对内存共享计数器 mm->count 进行判断，当还有其他进程共享使用 mm_struct时，不释放mm指针，节省了空间也保证了系统中其他进程的正常运行。

1.2 面向事件型多任务的多种 IPC 分析

uCLinux 同样可提供信号、管道、信号量、消息队列和共享内存等 IPC 机制。而对于事件型多任务系统而言，信号机制、消息机制、信号量机制可以很好地满足多任务协同设计的需求，3 种 IPC 机制如下：

(1)信号机制

作为一种异步的IPC机制，信号机制对事件触发型软件设计很有用，其主要用于通知接收进程某些特定事件发生。进程、内核、中断和异常等均可作为信号源，而所发送信号仅在接收进程被调度时才有机会被处理。接收进程接收到信号后可采取忽略、阻塞、内核缺省处理和自主处理等不同方式。传统信号集中的信号类型按位掩模表示，不能表示同类信号的数量，会造成信号丢失，进而可能导致任务进程的不可靠。实际中，系统设计人员常常使用基于队列结构的新型可靠信号机制。系统调用sigqueue( )可支持信号带参数发送附加信息，并与sigaction( )配合使用。发送非实时信号时，该函数对同类信号进行合并，从而兼容传统信号类型[3]。

在事件型系统设计中，信号机制的不足之处为：信号集

作者简介：张凯龙(1977－)，男，博士研究生、讲师，主研方向：网络化嵌入式计算技术，嵌入式软件测试技术；周兴社，教授、博士生导师
收稿日期：2006-12-11 **E-mail：**kl.zhang@nwpu.edu.cn

宽度有限，复杂交互操作受到制约；没有优先级，紧急信号不能被及时地处理；大量信号嵌套响应可能导致系统异常。

(2)消息机制

消息机制是多进程间的另一种异步通信机制。消息队列以更为复杂的数据包在多个进程间传送异步消息，与管道的服务相比，其功能更为强大。消息队列采用不连续消息的方式传递数据，能够灵活地处理数据，小数据块的传输效率较高。在消息通信过程中，内核会对消息源发出的消息及目标消息队列的合法性进行检查，以保证消息传递的正确性。若检查通过，内核将为消息分配存储空间并将其添加到消息队列末尾，唤醒等待该消息的进程进行处理。当然，结合信号机制及共享内存等机制也可实现这一功能，但软件实现将更复杂。

消息队列存在之处为：1)不支持广播机制，即一个单独消息不能为多个进程所接收；2)传送消息时需要进行 2 次数据复制，传送数据消耗较长时间。

(3)信号量机制

信号量机制是互操作进程间的一种更为复杂的同步机制，也是多个进程间更为可靠的同步方法。如前所述，由于 uClinux 中子进程与父进程共享内存，因此在父子进程中对特定内存区域进行访问时要加以有效的控制。采用信号量机制，可以在软件中方便地实现多进程对共享数据区域的互斥访问及多个进程间的同步与协同。在没有 MMU 的 uClinux 上开发多进程协同软件时，信号量机制将非常有用。信号量机制提高了多进程同步的可靠性，但可能会导致进程的死锁等异常现象。

在基于进程的事件型多任务系统中，灵活应用以上 3 种机制将是非常有效的。由于机制本身，其不可能严格保证软件乃至整个系统的可靠性，因此还需要进一步研究嵌入式多任务软件中的故障自检测与自恢复机制。

2 故障自检测与基于策略的自恢复机制

具有自恢复能力的软件系统大多都基于系统运行时监测、系统可变状态规划、配置可变状态描述以及激活配置等过程来实现[4~5]。在上述对基于uCLinux进程机制的研究基础上，本文进一步研究多任务系统中的故障自检测技术并将提出一种基于策略的故障自恢复机制。

(1)故障自监测技术

常用的故障自监测技术有看门狗技术、心跳任务等技术，其基本思想是通过设计相应的状态监测硬件或软件来监测关键任务的运行状态。本研究中的关键任务进程一方面负责重要功能的进程，另一方面包括可能导致系统不可靠的进程。广义上，关键任务进程包括系统创建的所有进程，而狭义上则指其自身故障将导致其他进程故障或系统部分/全部功能失效的进程。在此基础上，本文引入了一个特定的系统任务监管进程，主要作用之一就是周期性地监测关键任务进程簇中各进程。为了实现对关键任务进程状态的灵活监管，本文采用系统任务集和任务进程状态描述表来表示系统任务与任务进程的状态信息。在特定系统中，任务数量和类型是固定的，系统任务集可由多个元组组成，即

$\{T_{id}, T_{fid}, T_{hd}, T_p\}$

其中，T_{id}表示任务标识号，同时也代表任务的类别；T_{fid}是创建该任务的任务标识号；T_{hd}表示了该任务所占用的硬件资源集合；T_p表示任务的权重，任务权重越大对整个系统就越为关键。

任务进程状态描述表包括了每个任务进程运行时的信息，用元组表示为

$\{P_{id}, P_{fid}, P_t, P_p, P_{mec}, P_{ec}, P_s\}$

其中，P_{id}为该任务对应进程的进程号；P_{fid}为其父进程号；P_t为对应的任务类型，与系统任务集中的T_{id}相对应；P_p为任务权重，有对应的T_p值；P_{mec}是该类任务的最大异常次数；P_{ec}是该类任务的异常次数计数值，P_t相同的任务异常一次，该值加 1，且该值越大就表明对应资源、功能越不可靠；P_s为进程的运行状态。其中，P_{id},P_{fid},P_t,P_p,P_{mec}等内容由创建进程在创建任务时依据系统任务集中的信息等进行管理，P_{ec},P_s则由监管进程在动态运行过程中依据监测信息进行管理。

具体实现上，结合uClinux中多进程机制以及信号、消息等IPC机制，任务进程簇中的关键任务进程向监管进程周期性地发送实时信号或消息。监管进程依据信号或消息复位相应关键任务的异常标识，并依据信息刷新任务进程状态描述表中的相关表项。当某进程出现异常时，监管进程将设置任务进程状态描述表中P_{ec},P_s的值，标识异常产生。

(2)基于策略的故障自恢复机制

可靠嵌入式系统通常都需要具有一定的自管理、自恢复等能力。硬件电路复位是比较常用的方法，在硬件资源正常的情况下可保证软件功能的彻底恢复。但是，硬件复位方法需要消耗较长的时间，这对于具有时间约束的嵌入式系统而言显然不能完全适用。结合上述故障自检测技术机制，本文提出了基于策略的故障自恢复机制。

在该机制中，系统任务监管进程周期性地监测各任务状态。正常时，监控进程周期性复位硬件复位电路，系统正常运行。当有任务失效时，监控进程将尝试清除该任务、管理任务进程状态描述表，并依据该任务特征信息进行重建。创建成功后，监控任务仍将复位硬件复位电路。但如果系统中任务难以恢复或单位时间内某类任务的故障率过高，那么监管进程必须依据恢复策略来决定是否触发硬件复位。自恢复策略主要是依据任务进程状态描述表中的P_p,P_{mec},P_{ec}来决定，用$V=(P_p+ (P_{ec}-P_{mec}))$描述。该策略约定：当出现任务异常时，系统计算$V$值并判断是否大于用户设定的复位门限值$V_{初}$，若大于则必须进行硬件恢复，否则尝试进行软件恢复；权重越大的任务允许的故障次数越少；$V_{初}$的具体取值依具体系统和任务而定。

采用基于策略的自恢复方式可提高嵌入式软件自身的故障处理能力，保证故障时尽最大努力恢复，避免了频繁硬件复位对系统整体功能及性能的影响，分级自恢复机制见图 1。该机制在 uClinux 平台多进程环境下的具体应用方式中，系统创建主进程，主进程依据系统进程集的信息创建任务进程树和监管进程，监管进程对任务进程进行监测和管理。

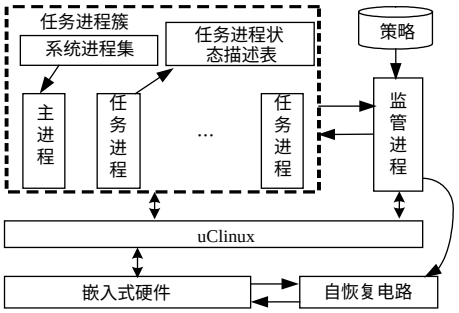

图 1 基于策略的分级自恢复机制

3 应用实例分析

上述方法在基于uClinux进程机制的某机载数据采集系统嵌入式多任务软件设计中得到应用和验证。系统任务进程树包括界面主进程T_{Maindisp}、界面子进程T_{subdisp}、数据采集主进程T_{Tmana}及负责GPS,Arinc429,RS232 等接口数据采集的多个子进程T_{N}。图 2 中实线箭头表示了启动采集任务时，系统自T_{Maindisp}开始依层向下创建T_{subdisp}，T_{Tmana}，T_N等进程的过程和采集数据传输过程。终止采集任务时，T_{subdisp},T_{Tmana}依次向下一层进程发送结束信号并等待应答，且只有m层进程全部退出时(m-1)层进程才退出，最后返回主界面，如图 2 中的虚线箭头所示。系统故障自检测与自恢复逻辑见图 3。

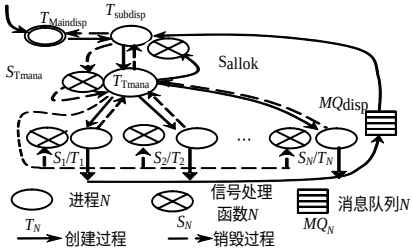

图 2　基于实时信号及消息机制的多进程管理与协同

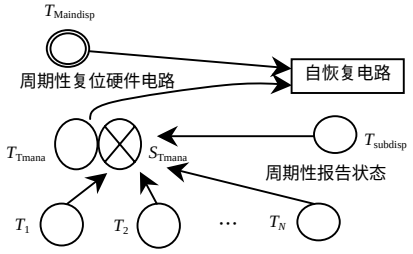

图 3　基于实时信号机制的故障自检测与自恢复

采集主进程T_{Tmana}同时也被作为监管进程，各采集子进程周期性向T_{Tmana}发送带有标识的信号以表明自己的状态。系统启动后T_{Maindisp}周期性复位自恢复电路直到启动数据采集任务后，该工作交由T_{Tmana}负责。T_{subdisp}及各采集子进程向T_{Tmana}周期性发送状态信息。如有异常，T_{Tmana}将强制清除异常任务并依据系统任务表重建。当故障率过高或者遇到关键任务严重失效时，T_{Tmana}将依据任务权重等策略决定是否复位硬件电路。

4　结束语

现在，uClinux 已在多类嵌入式系统中得到了广泛的应用。然而 uClinux 与标准 Linux 之间存在的差异要求设计人员必须优化传统的开发方法。本文在研究 uClinux 多进程管理及 IPC 机制的基础上，提出了一种具有故障自检测及策略化多级故障自恢复能力的可靠多任务嵌入式软件的设计方法。该方法可以从不同层面较好地提高嵌入式软件及整个嵌入式系统的可靠性，具有一定的可扩展性和通用性。

参考文献

1 朱显新, 黄　涛, 卢珞先. uC/OS和uClinux的比较[J]. 单片机与嵌入式系统应用, 2004, (10): 5-7.

2 李　莉, 张振宇. uClinux 的多进程管理[J]. 微计算机信息, 2005, 26(5).

3 王文义, 武华北. Linux 中进程间信号通信机制的分析及其应用[J]. 计算机工程与应用, 2005, 41(3): 108-110.

4 王纪文, 游　静, 许满武. 自恢复软件系统的建模与分析[J]. 系统仿真学报, 2005, 17(12): 2912-2921.

5 Schmerl B, Garlan D. Exploiting Architectural Design Knowledge to Support Self Repairing Systems[C]//Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering. 2002: 241-248.

（上接第 38 页）

图 5　OLAP 展现的分析结果界面

4　结束语

本文显示了项目组多年来的研究成果，基于一种快速原型的思想，设计并实现了一个一体化多维数据建模平台——MDMP，该平台集成了模型的构建、匹配与结果展现等功能，为 OLAP 应用系统设计者提供了一个强有力的辅助分析手段。本文结合具体实例说明了该平台应用方法，主要成果体现在以下 3 个方面：(1)探讨了一体化建模平台的总体流程，给出了一种 3 层体系结构(模型构建层、模型匹配与映射层、分析结果展现层)描述，并对各层的功能进行了简要说明。(2)设计并实现了该平台的主要功能模块：模型构建子系统，元数据管理子系统，综合指标管理子系统，模型匹配与映射子系统，数据综合与装载子系统，OLAP 分析子系统与报表展现子系统。通过这些子系统的协同工作，OLAP 应用系统设计者在用图示化的方法描述用户分析需求后，进行简单的操作，就可以展示对应的分析结果，在用户与 OLAP 应用系统设计者之间搭建了一座高效的沟通桥梁。(3)结合具体的项目背景——股票交易 CRM 系统的建设情况，描述了该平台的应用方法。

参考文献

1 Kimball R, Ross M. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling[M]. New York: John Wiley & Sons Inc., 2002.

2 LU Changhui, Deng Su, Zhang Weiming. Applying XML for Designing and Interchanging Information for Multidimensional model[J]. Systems Engineering and Electronics, 2005, 16(4).

3 陆昌辉, 邓　苏. 基于 UML 的多维数据概念建模方法的研究[J]. 系统工程与电子技术, 2004, 26(4): 522-525.

4 戴超凡. 数据仓库中数据志跟踪的理论与方法研究[D]. 长沙: 国防科技大学, 2002.

5 国家质量监督检查检疫总局. 信息分类和编码的基本原则和方法[M]. 北京: 中国标准出版社, 2002.

6 林　鹏. 战场信息 OLAP 支撑工具[D]. 长沙: 国防科技大学, 2005.