第30卷　第14期　　计　算　机　工　程　　2004年7月
Vol.30　№ 14　　Computer Engineering　　July 2004

·软件技术与数据库·　　文章编号：1000—3428(2004)14—0050—03　　文献标识码：A　　中图分类号：TP316

LAMOST总控系统中的进程优先级调度算法的设计与实现

郭　健，徐欣圻
(中国科学院国家天文台南京天文光学技术研究所，南京 210042)

摘　要：结合国家“九五”重大科学工程项目LAMOST望远镜总控系统研制开发的实践，着重描述了在QNX实时操作系统平台上如何通过设计优先级调度算法来协调完成多个分布式进程的机制。并以此机制为基础，对LAMOST望远镜的环境监控系统和GPS时标系统进行了设计完善。

关键词：LAMOST望远镜；实时操作系统；QNX；进程优先级；调度算法

The Design and Realization of Scheduling Algorithm for Process Priority in LAMOST Control System

GUO Jian，XU Xinqi
(National Astronomical Observatories /Nanjing Institute of Astronomical Optics & Technology，Chinese Academy of Sciences，Nanjing 210042)

【**Abstract**】In association with the research and development of the control system for LAMOST, which is one of the national large scientific projects, this paper focuses on the mechanism of scheduling algorithm for distributed multi-process priority on the QNX real-time operating system platform. Furthermore, based on the mechanism the design of the GPS timescale control system and environment parameter control system are optimized.

【**Key words**】LAMOST telescope；Real-time opertion system(RTOS)；QNX；Process priority；Scheduling algorithm

近年来，实时操作系统中的进程优先级调度问题日益得到广泛的关注，并且在许多国家重大科学工程和基金项目中得到了应用并受到广泛好评。在这些应用中，安全性至关重要，因为一旦无法满足时序约束要求，产生的后果将非常严重。但是介绍在大系统中如何实现具体的多进程优先级调度的文章并不多见，本文以国家重大科学工程项目LAMOST为实践基础，着重描述了在大系统中的多进程之间的优先级调度。

1 LAMOST介绍

LAMOST(Large sky Area Multi-Objects fiber Spectroscopic Telescope)是我国“九五”重大科学工程，项目投资2.35亿元。建成后它将成为世界上4米级口径以上光学望远镜中视场最大和光谱观测效率最高的望远镜。这样大的工程不仅给光学和机械的设计提出了很高的要求，也给望远镜控制系统的设计提出了挑战。总控系统的基本功能在于给LAMOST望远镜的各种信息，如命令信息、数据信息、状态信息等，提供通道。根据总体设计的要求以一定的规律和精度通过各电控子系统驱动望远镜的电控运动部件，与光机协调完成对目标天体的观测任务。LAMOST总控系统是在QNX4.25实时操作系统下开发的。

2 QNX实时操作系统

QNX是一个分布式、嵌入式、可规模扩展的实时操作系统。它遵循POSIX.1、(程序接口)和POSIX.2(Shell和工具)、部分遵循POSIX.1b(实时扩展)。它最早开发于1980年，到现在已相当成熟。

QNX是一个微内核操作系统，其核心仅提供4种服务：进程调度、进程间通信、底层网络通信和中断处理，其进程在独立的地址空间运行。因此QNX核心非常小巧(QNX4.25大约为12kb)而且运行速度极快。

QNX系统上的控制软件开发用Watcom C/C++，用户界面的编程使用QNX软件包中Photon microGUI所提供的工具。

3 QNX中的进程优先级调度策略

QNX4.25提供POSIX.1b标准进程调度：32个进程优先级；抢占式的、基于优先级的正文切换；可选3种进程调度算法：SCHED_FIFO，SCHED_RR，SCHED_OTHER。

(1) SCHED_FIFO：这是一种进程先入先出(FIFO)调度算法。一个被选择运行的进程会一直运行，直到它因I/O阻塞，或者主动释放CPU，或者是CPU被另一个具有更高优先级的进程抢先。

(2) SCHED_RR：这是一种进程循环(round-robin)调度算法。当SCHED_RR进程的时间片用完后，就被放到SCHED_FIFO和SCHED_RR队列的末尾。一个时间片的长度为50ms。除了时间片有些不同外，这种策略与SCHED_FIFO类似。

(3) SCHED_OTHER：这是QNX默认的进程调度算法类型,它采用动态优先调度策略,选择进程的依据主要是根据进程优先级值的大小。这种进程在运行时,可被高优先级值的进程抢先。如果一个进程用完了自己的时间片仍未被阻塞，它的优先级将被减1，并被放置到下一优先级队列的末尾。而一旦该进程被阻塞后，它的优先级将被恢复。值得注意的是，在一个进程的运行过程中它的优先级只会被降低一次。

基金项目：国家“九五”重大科学工程基金资助项目

作者简介：郭　健(1974－)，男，硕士生，主要研究方向：计算机应用；徐欣圻，研究员、博导

收稿日期：2003-06-26　　E-mail：marineisland@sina.com

4 LAMOST总控系统中进程优先级调度算法的设计

LAMOST总控系统中包括TCS(望远镜控制系统)及其下层的8个子系统。其中TCS与下层子系统之间的进程间通信大多数是采用C/S(客户/服务器)的模式来设计运行。这样，在整个系统中的TCS网络运行起来以后，将有数十个进程在同时运行。在这种情况下，如何设计好多进程之间的优先级调度就显得尤为关键。

进程调度策略是直接影响实时性能的因素。尽管调度算法多种多样，但大多由速率单调调度(Rate Monotonic Scheduling)和最早期限优先算法(Earliest Deadline First)变化而来。前者主要用于静态周期任务的调度，后者主要用于动态调度，在不同的系统状态下两种算法各有优劣。

把LAMOST总控系统中的所有进程分为两类：周期性进程和突发进程。

周期性进程由反复执行的运算组成，每个固定的时间周期执行一次运算。在LAMOST总控系统中一个典型的周期性进程是每隔五分钟一次读取现场温度传感器数据并更新内部变量和输出的当前状态。

突发进程由响应内部或外部事件的运算组成，突发进程在LAMOST总控系统中一个典型应用是外部风力环境参数突然变化，请求系统紧急关闭。

4.1 周期性进程的优先级算法

对于系统中的周期性进程首先采用速率单调调度算法，该算法是最具代表性的优先级调度方法。在运行之前已经把进程的主要特性确定，即预先了解进程在最坏情形下的执行时间和周期。

根据进程的周期设定固定的优先级：周期越短，优先级越高。任何时候，只要所有进程中具有最高优先级的进程就绪，即可分配到处理器。

但是，速率单调调度算法假定所有的任务都是独立的，即任务之间不存在优先关系。但是在系统中的进程是有关联的，它们之间存在着应用约束的问题。这样就需要对所有的进程在运行之前进行可调度性分析，来考虑那些通常在实时应用中存在的应用约束。显然，重要周期性进程与普通周期性进程的优先级不能单纯依靠进程的周期来划分。这样，我们就采用了速率单调调度算法和优先约束相结合的方法来确定周期性进程的优先级。

那么根据进程的优先约束来划分优先级又会出现一个问题，如果重要进程的执行周期较长，而一般进程的执行周期较短，这个时候他们的优先级该如何确定呢？采用了切割进程的办法，即把一个执行时间较长的进程划分为几个小进程，以缩短每个进程的执行时间，这样就可以避免一个执行时间很长的优先级高的重要进程长期占用CPU，从而极大地降低系统的运行效率的问题。

在QNX中设定所有周期性进程采用如下方法：对于不改变调度算法的进程，就用系统函数int setprio(pid_t pid, int prio)直接给进程设定优先级。对于需要改变调度算法的进程，采用如下方法：

```
struct sched_param{
  int sched_priority;//对应调度算法的优先级
  int sched_reserved;//系统保留参数
};//定义了一个参数结构
param.sched_priority = 15;//进程优先级数值
sched_setscheduler( 0, SCHED_FIFO, &param );
 //设定进程的调度算法和优先级
```

如果需要设定的进程不在当前节点的话就调用QNX专用函数:

```
int qnx_scheduler(pid_t proc_pid, pid_t pid, int alg, int prio, int ret);
```

通过配置proc_pid就可以在设置不同节点进程的优先级。

4.2 突发进程的优先级确定

突发进程a可由3个方面ca, da, mina描述。ca 表示进程a在最坏情形下的运算时间；da 表示时限，即从向进程a发出请求到进程a完成执行之间的时间间隔。无法预知执行一个突发进程的精确请求时间，但通常可以预先得到两个连续请求间的最短时间mina。这样就可以把突发进程等价于相应的周期性进程，通过设定优先级而使得它能够在它的时限内完成进程。而它的优先级确定方法就应用以上所述的周期性进程调度算法。

4.3 服务器进程和客户进程的优先级关系

由于在系统中客户进程的数量大大超过了服务器进程的数量，为了对客户及时提供服务，服务器进程的运行需要具有与客户进程相同或是更高的优先级。其结果，服务器进程所使用的优先级可能会高于其所有的客户进程优先级。如果一个低优先级的客户进程向服务器进程发送了一个消息，那么该请求将由服务器进程以较高的优先级进行处理。如果服务器进程以较长的时间来响应客户进程的服务请求，那么低优先级的客户进程就可能影响比其优先级高但又比服务器进程低的进程的运行。

为了解决这个问题，利用了QNX中提供的改变当前进程的系统标志字的方法。具体方法如下：

```
long bits, old_bits，new_bits; //定义系统标志字
int qnx_pflags(long bits,long mask,long *old_bits,long new_bits);   //系统函数调用
```

当在系统中的服务器进程内添加了qnx_pflags( )函数以后，只要设定对应的mask变量的值就可以使得服务器进程的优先级由给它发送消息的客户进程来设定。为了实现这个方法，把服务器进程的mask值设定为_PPF_PRIORITY_REC | _PPF_PRIORITY_FLOAT。_PPF_PRIORITY_REC的含义为以发送消息进程的优先级大小来接受消息队列。_PPF_PRIORITY_ FLOAT的含义为调整自身优先级为发送消息进程的优先级。这样设置以后，当服务器进程收到客户进程发来的一个消息时，便将自己设置为具有与客户进程同样的优先级。这时，如果有一个具有较高优先级的客户进程给该服务器进程发送消息的话，该服务器进程将进一步提升自己的优先级。这样就不会出现低优先级的服务器进程被较高优先级的第三方进程所阻塞，从而间接导致该客户端进程被阻塞的情况。

5 LAMOST中的进程优先级调度算法实现

在EPCS(环境参数控制系统)中，有一个温度监测服务器进程：temperature.c，还有与之相关的两个客户进程：shutter.c、exhaustfan.c。在TCS节点有一个gpstime.c服务器进程,其它各个子系统都有与之对应的客户进程timeclient.c。那么如何协调各个进程之间的优先级关系就成为一个问题。依照上述的由发送消息的客户进程来决定服务器进程优先级的原则采用如下方法确定了各个进程的优先级关系。

首先在temperature.c进程中调用sched_setscheduler(0, SCHED_RR, &param)函数，其中param的参数值为18，这样就把temperature.c的调度方法设为了RR，优先级设为了18。然后调用qnx_pflags(0，_PPF_PRIORITY_REC_|PPF_PRIORITY_FLOAT,0,0)这样在客户进程访问它时，它的优先级就变为了客户进程的优先级了。在客户进程shutter.c、 exhaustfan.c中调用函数setprio (0，12)，setprio (0，13)分别设定优先级为12，13。在TCS节点的服务器进程gpstime.c中调用函数sched_setscheduler(0, SCHED_RR, &param)，把进程调度方法设定为RR，优先级设定为26，各个客户进程timeclient.c中调用setprio(0，16)，优先级设定为16，调用方法为默认方法other。

6 结果分析

由于在temperature.c进程中调用了qnx_pflags函数，这样当它自身运行时它的优先级为18，当客户访问它时它的优先级为客户进程的优先级，从而就不会阻塞比它优先级低而又比客户进程优先级高的进程(例如timeclient.c)。下面以图1和图2来说明运用进程调度算法后的进程执行前后的差异。

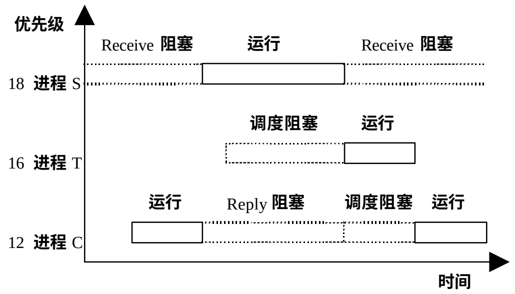

图1 未运用调度算法

图1运用调度算法前，客户进程C利用服务器进程S阻碍了比它优先级高的进程T运行。

图2 运用调度算法以后进程T不再被优先级低的进程C阻塞。图例说明：进程S－temperature.c；进程C－shutter.c；进程T－timeclient.c。

在GPS时标系统中，gpstime.c进程是一个重要而又占用系统资源较长时间的进程，在未运用进程优先级调度算法之前，它运行一次需要742ms(平均值)，这样它就较长时间的阻碍了同优先级的其它进程。我们对其调度算法设为RR后，它运行时间变为了776ms(平均值)，这样虽然它的运行时间略有增加，但是由于有了时间片的限制，它就不会阻碍同优先级的其它进程的运行，使得系统整体的运行效率得到提高。例如：与它同优先级的windpower.c进程，在与gpstime.c同时运行时需要634ms才能完成一次执行，在运用调度算法以后，只需要72ms就可以完成同样的一次执行。

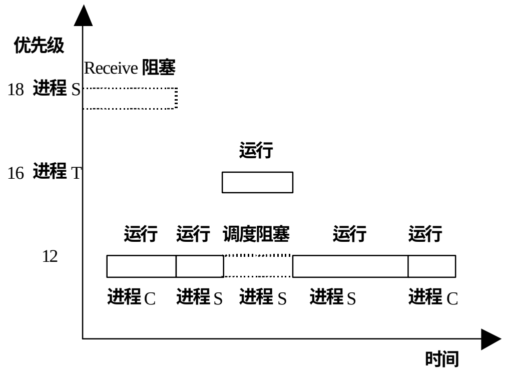

图2 运用调度算法

7 总结

通过以上在EPCS和GPS时标子系统中进程优先级算法的应用，多个进程的执行时间满足设计要求，实现了预期的效果。当然，整个LAMOST总控系统尚未最终完成，随着系统的进一步增大，还需要进一步努力优化我们的设计，争取取得更好的效果。

参考文献

1 徐欣圻. LAMOST望远镜总控模拟系统.LAMOST工程部内部技术报告,2001

2 QNX OS System Architecture (Second Edition).QNX Software Systems Ltd. ,1999

3 侯业勤,张 菁.分布式嵌入式实时操作系统QNX.北京:宇航出版社,1999

4 邹志强. LAMOST软件跟踪模拟系统[硕士学位论文].南京: 中科院南京天文光学研究所，2003

5 徐灵哲. LAMOST分布式数据库设计.中科院南京天文光学研究所, 2002

(上接第31页)

2.4.2 讨论

实验虽然采用不同类别的文档为实验对象，但是由于实验的查询条件相同，因此获得的信息在很大的程度上具有相似性，有一些信息本来就是同一领域的信息在不同领域的专业刊物上发表的。另外有研究表明文档标题对文档分类正确的贡献率一般只有70%左右，而我们只是提取了文档的标题进行实验，所以实验结果出现了一定的检索偏差。此外本算法没有考虑用户的进一步参与，即对关键词的取舍及权重的修改。实验的结果只是最基本的个性化信息的匹配，事实上如果算法考虑与用户的交互，增强个性化的比重；或是把特征提取的范围扩大到摘要或是全文，实验的结果会更好。

3 结论

本文提出的基于奇异值分解结合神经网络的个性化信息模式识别的方法只是实现个性化信息管理的第一步，即信息的个性化识别。只有利用用户个性特征信息去识别信息、管理信息，才能有效地为用户提供个性化服务。在进行实际的信息识别中，利用现在的信息服务系统，如搜索引擎或是元搜索引擎进行信息的初步搜索识别，得到一个初步相关的信息集合，然后应用本文中的算法在这些相关信息中进行个性化信息的模式匹配和个性化信息挖掘服务。以后研究的重点方向应该是个人信息获取特征的建模、个性化特征关键词权重的设定、个性化信息识别神经网络的构造等方面的问题。

参考文献

1 Deerwester S.Indexing by Latent Sematric Analysis.Journal of the American Society for Information Science,1990,41:391-407

2 Foley D H,Sammon J J W.An Optimal Set of Discriminant Vectors. IEEE Trans. on Computer, 1975,c-24(3):281-289

3 Woo S M.User-centered Filtering and Document Ranking. TENCON 99. Proceedings of the IEEE Region 10 Conference,1999,2:1059-1062

4 Lee J H.Ranking Documents in Thesaurus-based Boolean Retrieval Systems.Information Processing & Management,1994,30:79-91

5 丛 爽.面向MATLAB工具箱的神经网络理论与应用.合肥:中国科学技术大学出版社,1998