第33卷 第15期 计 算 机 工 程 2007年8月
Vol.33 No.15 Computer Engineering August 2007
·软件技术与数据库· 文章编号：1000—3428(2007)15—0061—03 文献标识码：A 中图分类号：TP311

嵌入式操作系统μC/OS-Ⅱ内核改进及其应用扩展

苏 娟，吴旭光，张 朝
(西北工业大学航海学院，西安 710072)

摘 要：实时嵌入式操作系统μC/OS-Ⅱ具有开放源码、研究免费的特点，并且通过了FAA安全认证。对μC/OS-Ⅱ内核的改进、外围文件系统、TCP/IP协议等模块的实现方法进行了讨论，对快速构建基于μC/OS-Ⅱ的嵌入式应用平台提供了方便。
关键词：嵌入式操作系统μC/OS-Ⅱ；内核；文件系统；TCP/IP协议；嵌入式应用平台

Improvement and Extensive Application of Embedded Operating System μC/OS-Ⅱ

SU Juan, WU Xuguang, ZHANG Zhao
(Marine College, Northwestern Polytechnical University, Xi'an 710072)

【Abstract】μc/os-Ⅱ(embedded real-time OS)passes FAA safety certification, and has character of open source codes and is free in study. Realization of system modules such as kernel improvement, file system, TCP/IP protocol stack is discussed. This platform provides utilities for application based on μC/OS-Ⅱ and makes the work easier and faster.
【Key words】embedded operating system μC/OS-Ⅱ; kernel; file system; TCP/IP protocol stack; embedded application platform

作为可抢占的实时内核，μC/OS-Ⅱ与商业嵌入式实时内核在性能上没有差异。但由于μC/OS-Ⅱ只是一个操作系统内核，只提供了资源管理的基本功能，不支持时间片轮转的调度算法，存在着优先级反转等不完善的问题，而且与商用实时操作系统相比也缺少必要的外围软件包。要实现一个相对完整、实用的嵌入式实时多任务操作系统，需要对内核进行改造并进行功能扩充。本文对μC/OS-Ⅱ内核的改造、及外围文件系统、TCP/IP协议等模块的扩展进行了研究，并提出详细的解决方案，对于快速构建基于μC/OS-Ⅱ的嵌入式应用平台提供了方便。

1 μC/OS-Ⅱ内核的改进

在实际应用中，往往有些任务需要同优先级进行调度，而且基于同优先级的调度机制，也能解决优先级反转问题。在此提出一种基于μC/OS-Ⅱ的同优先级调度方法。

当两个或两个以上的任务具有相同优先级，内核允许一个任务运行确定的一段时间后，切换给另一个任务，叫做时间片调度。μC/OS-Ⅱ支持不同优先级的任务调度，通过任务对应的优先级状态，快速地由OSTCBPrioTb1[OSPrioHighRdy]指针的指向切换到期望的TCB来完成调度。要想实现同优先级调度必须修改它的OSTCBList数据结构，如图1所示。

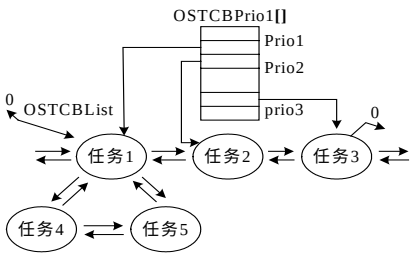

图1 数据结构修改

在图1中，任务1、任务2和任务3为不同优先级的任务，任务1、任务4和任务5为同优先级的任务。当没有比任务1优先级更高的任务时，每发生一次节拍中断，任务1、任务4和任务5就依次在运行态和就绪态切换。通过改变OSTCBPrioTb1[prio1]指向的任务TCB，也就是修改OSSched()来实现。很容易看出同优先级任务间的数据结构是一个循环链表结构，所以，TCB也相应地扩展两个指针域。新的数据结构将使就绪任务的TCB在一个循环链表和线性链表上不断地切换。而对于等待任务的TCB，也将在一个循环链表和线性链表上不断地切换。这里需要增加一个全局变量的数组OSTCBWaitTb1[OS_LOWEST_PRIO]指向相对应优先级的等待任务的TCB，以便控制就绪任务进入等待任务或等待任务进入就绪任务是链表上任务TCB的插入和删除。

扩展TCB的指针域。定义新的TCB为：

```
#ifOS_TASK_PrioEqu_EN
Struct OS_TCB*prioequNext;
Struct OS_TCB*prioequPrev;
#endif
……}OS_TCB
```

增加的全局变量的数组定义为：

```
#ifOS_TASK_PrioEqu_EN
OS_EXT OS_TCB OSTCBWaitTb1[OS_LOWEST_PRIO]
#endif
```

对OSTCBList操作的每个相关函数都要修改，主要修改以下方面的内容。

(1)对于链表数据结构

1)链表的初始化及头结点。与OSTaskCreatExt()函数中的

作者简介：苏 娟(1982－)，女，硕士研究生，主研方向：嵌入式操作系统；吴旭光，教授；张 朝，硕士研究生
收稿日期：2006-07-23 **E-mail：**ssjnpu20022@sina.com

OSTCBInit()相关，需要完成同任务 TCB 构成的循环链表及不同任务 TCB 构成的线性链表。

2)链表结点的插入和删除，与任务的状态变化有关。当任务由就绪态进入等待状态或者由等待状态进入就绪状态时，任务 TCB 需要添加到相应的链表中或从对应的链表中删除。这些函数与任务由就绪态进入等待状态或者由等待状态进入就绪态相关，有：OSEventTaskWait()，OSEventTo()，OSTaskRdy()，OSTaskDel()以及OSTaskChangePrio()。

(2)对同优先级任务调度时，由 OSTCBPrioTb1[prio]指针的移动实现 TCB 的切换，这里是对循环链表的操作，可以修改 OSSched()实现。

(3)由时间延时满引起的超时，使等待的任务进入就绪态，这一方面是对链表结点的插入和删除，另外还是对任务延时的控制，需要修改 OSTimeDly()和 OSTimeTick()。

对这几个函数修改需要注意的是，当要对某个优先级的循环链表插入结点前或删除结点后对应的 TCB 为空时，需要对就绪列表或等待列表相应的位置置位或清零。

解决优先级反转的方法主要思路是提升优先级低但占有资源的任务至申请该资源的高优先级任务的优先级，直到低优先级的任务释放该资源，恢复低优先级任务的优先级，高优先级的任务才占有该资源。这里需要改变任务优先级，改变的优先级分别是提升优先级到高优先级和恢复任务到低优先级。需要存储将要改变结果的优先级数，通过扩展 OS_Event 来实现。因任务对共享资源的占有和释放是通过对信号量的操作来实现的，所以要分别修改 OSSemPend()和 OSSemPost()。关键点在于当资源恰好分配完时，记下当前任务的优先级。当任务等待分配资源时提升占有资源任务的优先级。当任务释放临界资源时恢复一个任务的优先级。具体实现代码从略。

2 μC/OS-Ⅱ的扩展

随着应用范围和深度的加大，μC/OS-Ⅱ实时内核往往不能满足要求。所以可以有选择地向其中加入文件系统、TCP/IP 协议、图形界面等模块，使 μC/OS-Ⅱ成为一个面向问题实用的嵌入式操作系统。

2.1 文件系统

嵌入式系统一般都用大容量的电子盘(flash)作为永久存储介质，这种设备的特性就是数据只能被整块(Block)地改写(擦除)，所以，数据需要按照整块存储。本文仿照 FAT16 文件系统为 μC/OS-Ⅱ建立 FAT8 文件系统。在此系统中，把文件作为一种无结构的字节序列，用户任务可以在文件中加入任何内容，并且以任何方式来处理它们。文件也是以“簇”为单位，分块存储。每个簇的大小根据实际系统的电子盘特性固定为整块的大小。

整个文件系统由文件分配表和文件目录表组成。文件分配表的表项号就是簇号，每个表项号 1 个字节，所以是 8 位的文件分配表(FAT8)，这样就用表项号的形式登记上大容量电子盘的所有块，表项号的内容表示该文件的下一个簇的簇号。文件目录表登记文件的相关信息(如创建时间、文件属性等)，同时登记上文件的首簇号。读取文件时候，在文件目录表中找到相应的文件首簇号，读取完一个簇以后，如果一个文件大于一个簇(电子盘的一个块)，就从文件分配表中找到下一个簇的号码，然后继续读取，直到文件分配表中下一个簇的号码为 OFFH 时为止。整个文件系统结构如图 2、图 3 所示。其中：File1_Name 文件的起始簇号为 03H，文件包括电子盘块号：03H，07H，05H，09H。File2_Name 文件的起始簇号为 06H，文件包括电子盘块号：06H，01H，0AH，OBH。

00H	
01H	0AH
02H	
03H	07H
04H	
05H	09H
06H	01H
07H	05H
08H	
09H	0FFH
0AH	0BH
0BH	0FFH
0CH	
0DH	
…	
0FFH	

图 2 文件系统结构(1)

00H	File1_Name	03H	type	…
01H	File2_Name	06H	type	…
02H	…	…	…	…

图 3 文件系统结构(2)

系统对文件的操作是围绕着某些数据结构进行的，如目录项、文件分配表、文件结构体、用户任务控制块中的文件请求登记项等。所以要对它们进行定义并在内核中增加一些相关的数据结构，还要对原来相关数据结构进行必要的修改。

如用户对文件进行读写操作时候，系统必须维护当前被用户所打开的所有文件，用文件控制块来登记这些被打开的文件，其定义如下：

```
Typedef struct//占用 6 字节大小
{     OS_FCB *pNext；//指向下一个控制块
      OS_DIRITEM *Dirltem； //文件目录项的内容
      INT8U LinkCount； //打开此文件的用户数量
      INT8U *PathName； //文件的绝对路径
} OS_FCB;
```

一个用户任务可能同时打开几个文件，对文件读写是要记录下文件的当前位置，以便在挂起后重新调度运行时能从这个位置继续进行。用打开文件数据结构来描述：

```
Typedef struct//占用 8 字节
{   OS_OPENFILE * pNext；//下一个打开的文件
    OS_FCB * pFCB；//被打开的文件的文件控制块
    INT32U Position；//文件的当前读写位置
} OS_OPENFILE；
```

修改原有的任务控制块 OS_TCB，增加一个指向打开文件的指针 OS_FILE * pFile。修改系统初始化 OSInit()过程，增加初始化文件系统 OSFileInit()的调用。

用户可以通过调用 void OSFileInit()、void OSFileClose (OS_FILE * pFile)、void OSFileBlckErase(INT8U blck)等 API 函数实现对文件系统的使用。

FAT8 结构简单，系统资源开销小，运行时 RAM 占总量的 0.07%；额外的 Flash 存储器消耗合计占 0.65%。系统有全局的文件分配表数组，利用数组的随机访问特点，大大提高了文件簇的寻址速度。可靠性好，每次初始化文件系统时，都要对文件分配表和目录表这两个极重要的数据结构做新备份以防不测，还提供 OSFileCheck()系统调用，检查 FLASH 盘文件系统并在必要时修改错误。可移植性好，数据结构及函数的定义都采用 μC/OS-Ⅱ的与平台无关的数据类型，所以只要能运行 μC/OS-Ⅱ的平台，就能很方便地实现 FAT8。

2.2 TCP/IP 协议栈

图 4 系统示意图

LWIP 是一个小型的 TCP/IP 协议栈，占用空间小，对操作系统依赖少，而且实现了较为完备的 IP、ICMP、UDP、TCP 协议。把开放源代码的 TCP/IP 协议栈 LWIP 移植到 μC/OS-

Ⅱ上来，可得到可免费研究、学习的嵌入式网络软件平台。系统示意图如图 4 所示。

LWIP 协议栈中所有与硬件、OS、编译器相关的部分均放在/src/arch 目录下。因此只需修改这个目录下的文件。主要修改以下几部分：

(1)与 CPU 或编译器相关的 include 文件。在/src/arch/include/arch 目录下 cc.h、cpu.h、perf.h 中有一些与 CPU 或编译器相关的定义。如数据长度，字的高低位顺序等。这应该与用户实现 μC/OS-Ⅱ时定义的数据长度等参数一致。

(2)与 OS 相关的结构和函数：1)sys_sem_t 数据结构及信号量函数；2)sys_mbox_t 消息函数；3)系统超时函数；4)创建新线程函数。

在 UC/OS-Ⅱ中，没有线程的概念，只有任务。它已经提供了创建新任务的系统 API 调用 OSTaskCreate，只要把 OSTaskCreate 封装一下，就可以实现 sys_thread new。而且用户要事先为 LWIP 中创建的线程分配好优先级。

(3)Lib_arch 中库函数的实现。LwIP 协议栈中用到了 8 个外部函数，这些函数通常与用户使用的系统或编译器有关，因此留给用户自己实现。如 u16_t htons(u16_t n) //16 位数据高低字节交换；int strlen(const char *str) //返回字符串长度等。

(4)网络芯片驱动程序。用户为自己的网络设备实现驱动时应参照 LwIP 提供的驱动模板，即/src/netif/ethernetif.c 文件。在 LwIP 中可以有多个网络接口，每个网络接口都对应了一个 struct netif，netif 包含相应网络接口的属性、收发函数。LwIP 调用 netif 的方法 netif->input()及 netif->output()进行以太网 packet 的收、发等操作。在驱动中主要是实现网络接口的收、发、初始化以及中断处理函数。

做完移植修改工作以后，在 μC/OSII 中初始化 LwIP，创建 TCP 或 UDP 任务进行测试。初始化关键部分代码如下：

```
main(){
OSInit();
OSTaskCreate(lwip_init_task,&LineNo11,
&lwip_init_stk[TASK_STK_SIZE-1], 0);
OSTaskCreate(usr_task,&LineNo12,&usr_stk[TASK_STK_SIZE-
1],1);
OSStart();  }
```

2.3 用户 Shell

构造一个易扩展的用户 shell，来完成命令行解析和输入输出功能。其实现主要分如下几部分：

(1)底层的串口中断处理。设置发送和接收缓冲区，串口中断处理程序从发送缓冲区取得数据进行发送，把接收到的数据放入接收缓冲区。当接收到回车键，中断处理程序向 Shell 进程发送消息，提交输入命令行。中断处理程序要区分命令行输入和数据传送两种不同的模式。对于命令行输入，必须具有字符回显、特殊字符处理等功能。对于数据传送，要根据 Modem 协议进行处理，传送完后向 Shell 进程提交。

(2)基本的输入输出函数。输出函数如 printf、putchar,由函数直接写发送缓冲。对于输入函数如 scanf、getchar，应提交给 Shell 进程统一处理。Shell 进程在收到数据输入后判断，再向应用程序提交。

(3)Shell 进程。作为 μC/OS-Ⅱ下的一个 task 独立运行，处于最低优先级，主要用于管理输入输出。Shell 启动后，进入睡眠状态，等待用户输入或数据输入。它根据命令行输入和数据输入，对命令行进行解析，并找到对应的处理程序执行。

3 总结

μC/OS-Ⅱ是一个良好的实时操作系统，随着更多的使用和不断的改进，其功能将日趋成熟，应用会更加广范，必将会有更好的应用前景。

参考文献

1 Labrosse J J. 嵌入式实时操作系统 μC/OS-II[M]. 邵贝贝, 译. 2 版. 北京: 北京航空航天大学出版社, 2003.

2 王田苗. 嵌入式系统设计与实例开发——基于 ARM 微处理器与 μC/OS-Ⅱ实时操作系统[M]. 北京: 清华大学出版社, 2002.

3 田　泽. 嵌入式系统开发与应用[M]. 北京: 北京航空航天大学出版社, 2005.

（上接第 42 页）

较大，从而导致算法的时间开销较大。而在有参数取值范围信息的情况下，可以将遗传算法的种群数和进化代数设置得比较小，从而使参数优化所需的时间大大减少。虽然两种算法中，SVR 参数初始取值范围不同，但最终获得的参数值偏差不大，且性能相当，都具满意的逼近效果。混合算法在逼近程度和时间效率上的提高，对于应用 SVR 解决具有高维度、大样本的实际问题尤为重要。

5 结语

针对 SVR 参数选择问题，本文提出一种混合选择算法，利用学习样本信息，确定 SVR 参数的一个较小范围，再用随机优化算法搜索最优参数值。研究表明，混合选择算法是确定 SVR 参数取值的一种有效方法，在非线性函数逼近的仿真实验中得到了验证。这为各种应用环境下选择 SVR 参数提供了一条有效途径。

参考文献

1 Vapnik V. 统计学习理论的本质[M]. 北京: 清华大学出版社, 2000.

2 Smola A, Schölkopf B. A Tutorial on Support Vector Regression[J]. Statistics and Computing, 2004, 14(3): 199-222.

3 Chapelle O, Vapnik V, Bousquet O, et al. Choosing Multiple Parameters for Support Vector Machines[J]. Machine Learning, 2002, 46(1): 131-159.

4 Chalimourda A, Schölkopf B, Smola A. Experimentally Optimal *v* in Support Vector Regression for Different Noise Models and Parameter Setting[J]. Neural Networks, 2004, 17(1): 127-141.

5 Jordaan E M, Smith G F. Estimation of the Regularization Parameter for Support Vector Regression[C]//Proceedings of the IEEE International Joint Conference on Neural Network. 2002, 3: 2192-2197.

6 Kwok J T, Tsang I W. Linear Dependency Betweenε and the Input Noise inε-Support Vector Regression[J]. IEEE Transactions on Neural Networks, 2003, 14(3): 544-553.

7 朱嘉钢, 王士同, 杨静宇. 鲁棒 *r*-支持向量回归机中参数 *r* 的选择研究[J]. 控制与决策, 2004, 19(12): 1383-1386.

8 Cherkassky V, Ma Y. Practical Selection of SVM Parameters and Noise Estimation for SVM Regression[J]. Neural Networks, 2004, 17(1): 113-126.

9 Üstün B, Melssena W J, Oudenhuijzenb M, et al. Determination of Optimal Support Vector Regression Parameters by Genetic Algorithms and Simplex Optimization[J]. Analytica Chimica Acta, 2005, 544(1/2): 292-305.

10 袁小芳, 王耀南. 基于混沌优化算法的支持向量机参数选取方法[J]. 控制与决策, 2006, 21(1): 111-113.

11 张礼兵, 金菊良, 刘　丽. 基于实数编码的免疫遗传算法研究[J]. 运筹与管理, 2004, 13(4): 17-20.