翡青的博客
C++, Linux系统与编程, TCP/IP, 数据结构与算法, Hadoop与云计算



个人资料

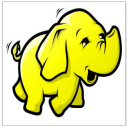
翡青

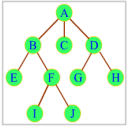
访问： 545222次
积分： 11051
等级： BLOG 7
排名： 第556名

原创： 309篇 转载： 30篇
译文： 2篇 评论： 695条

博客专栏


Programming In The Linux Environment
文章：57篇
阅读：48041


Hadoop与云计算
文章：8篇
阅读：8168


数据结构与STL
文章：31篇
阅读：26711


Linux实践与提高
文章：53篇
阅读：50032


C++ 入门与提高
文章：110篇
阅读：139604

# 操作系统学习笔记_5_进程管理 -- 同步与互斥

分类： 计算机系统与Linux内核 2014-07-18 20:17 943人阅读 评论(0) 收藏 举报

操作系统 进程管理 同步与互斥 信号量 管程



## 进程管理

## --同步与互斥

### 一、进程同步与互斥的基本概念

1.基本概念

在多道程序系统中,由于进程,各进程之间有两种形式的制约关系:

(1)间接相互制约- 源于资源共享 -互斥。

(2)直接相互制约- 源于进程合作 -同步。

进程同步:主要源于进程合作,为进程之间的直接制约关系。

进程互斥:主要源于资源共享,是进程之间的间接制约关系。

临界资源:一次只允许一个进程使用的资源称为临界资源,如打印机、公共变量等。

临界区:在每个进程中,访问临界资源的那段程序称为临界区。

2.同步机制应遵循的准则

(1)空闲则进:当临界区空闲时,进程可以立即进入,以便有效地利用临界资源。

(2)遇忙等待:当已有进程进入其临界区时,其它进程必须等待,以保证互斥。

(3)有限等待:对要求进入的进程,应在有限的时间内使之进入,以免陷入"死等"。

(4)让权等待:对于等待的进程,它必须立即释放处理机,以避免进程忙等。

### 二、实现临界区互斥的基本方法

1.用软件实现的同步互斥机制

(1)算法一:单标志法

(2)算法二:双标志法先检查

(3)算法三:双标志法后检查

(4)算法四:Peterson'sAlgorithm

2.进程互斥的硬件方法

(1)检测和设置(TS)指令

(2)swap指令(或exchange指令)该指令的作用是交换两个字(字节)的内容

### 三、 信号量

1.对信号量S进行 P操作,记为P 如下

```
[cpp]
```

文章分类

C++ (115)
Linux (74)
算法与数据结构 (33)
Hadoop与云计算 (8)
Linux环境高级编程 (51)
计算机网络与TCP/IP (4)
计算机系统与Linux内核 (14)
MySQL数据库编程 (15)
程序人生 (32)

文章存档

2015年05月 (1)
2015年04月 (4)
2015年03月 (13)
2015年02月 (40)
2015年01月 (24)

展开

阅读排行

2.5年, 从0->阿里 (17574)
Windows下远程登录到Li (5950)
C++ Primer 学习笔记_1_ (4751)
UNIX网络编程 --环境搭建 (4039)
我的2013 --岁月划过生命 (3884)
C++ Primer 学习笔记_27 (3804)
C++ Primer 学习笔记_45 (3652)
C++ Primer 学习笔记_2_ (3106)
为了那永不坠落的梦想... (2990)
Linux 学习笔记_2_Linux (2770)

最新评论

2.5年, 从0->阿里
pirDOL: 楼主你好，拜读了《2.5年，从0->阿里》，收获良多。我也在今年拿到了yunos的实习offer，但...

2.5年, 从0->阿里
adminabcd: 哎，自从上了csdn才发现牛人那么多，而自己却是个井底之蛙，大学完全荒废了

2.5年, 从0->阿里
wodeai1625: 哥们看了你的博客后感慨很深很深，但是有些内容我还不是很了解。想和你qq交流下。不知道学长方便吗

我的2013 --岁月划过生命线(大二
wodeai1625: 哥们，看你的博客感慨很深很深。可不可以加你的qq想和你好好聊聊

2.5年, 从0->阿里
MonroeD: 请问一下学长，有没有问你mySQL等数据库的问题吗?

C++ Primer 学习笔记_108(大结局
cyfcsd: 楼主高人

TCP/IP入门(3) --传输层
荒漠之弦: 编辑的非常好！我也转走了，向你学习！

为了那永不坠落的梦想...
kkwsj: 向LZ学习

2.5年, 从0->阿里
翡青: @Suprman:恩恩, 对, 还是感觉自己会的东西太浅太少了... 正加倍努力中....奋斗

2.5年, 从0->阿里

```
01.  --S.Q;          //表示
02.  if (S.Q < 0)    //表示
03.  {
04.      调用进程进入等待队列
05.      阻塞调用进程;
06.  }
```

2.对信号量S进行 V操作,记为V(S),处理过程如下

```cpp
01.  ++S.Q;          //表示释放一个资源
02.  if (S.Q <= 0)   //表示有进程处于阻塞状态
03.  {
04.      从等待队列 S.Q 中取出一个进程 P;
05.      进程 P 进入就绪队列;
06.  }
```

## 四、 管程

一个管程定义了一个数据结构和能为并发进程所运行的一组操作,这组操作能同步进程和改变管程中的数据。

管程由三部分组成:

局部于管程的共享数据说明;

对该数据结构进行操作的一组过程;

对局部于管程的数据设置初始值的语句。

## 五、 经典的同步与互斥问题

1.生产者-消费者问题

用C语言和信号量机制描述生产者-消费者问题的程序如下:有界缓冲区的大小为100。

```cpp
01.  #define N 100            //有界缓冲区大小
02.  typedef int semaphore;   //定义信号量
03.  semaphore mutex = 1;     //临界区互斥信号量
04.  semaphore empty = N;     //空闲缓冲区
05.  semaphore full = 0;      //缓冲区初始化为空
06.
07.  void producer(void)
08.  {
09.      int item;            //局部变量
10.      while(1)
11.      {
12.          item = produce_item();  //生产数据
13.          P(empty);           //获取空数据槽
14.          P(mutex);           //获得进入临界区的信号量
15.          insert_item(item);  //将数据放入缓冲池
16.          V(mutex);           //释放互斥量
17.          V(full);            //数据量加一
18.      }
19.  }
20.
21.  void consumer(void)
22.  {
23.      int item;            //局部变量
24.      while(1)
25.      {
26.          P(full);        //获取数据槽
27.          P(mutex);       //获得进入临界区的信号量
28.          item = remove_item();   //将数据从缓冲池读出
29.          V(mutex);       //释放互斥量
30.          V(empty);       //数据量减一,即空槽加一
31.          consume_item(item); //消费数据
32.      }
33.  }
```

翡青: @sunqunsunqun:对, 山科大, 计算机系

2.读者-写者问题

设置互斥信号量wmutex表示写者间、读者和写者间互斥,用readcount变量来记录读者数,由于readcount是读者间共享变量,属于临界资源,它也需互斥,为此,又增设互斥信号量rmutex。程序如下:

```cpp
typedef int semaphore;  //定义信号量
semaphore rmutex = 1;   //读者计数器的互斥量
semaphore wmutex = 1;   //写-写,读-写互斥量
int readcount = 0;      //读者计数器

void reader(void)       //读者进程
{
    while (1)           //进程被调度
    {
        P(rmutex);      //取得读者计数器的互斥量
        readcount = readcount + 1;  //进来一个读者,读者数量加一
        if (readcount == 1)         //如果是第一个读者,取得读-写互斥量
            P(wmutex);
        V(rmutex);      //释放读者计数器的互斥量
        read_data_base();           //读数据
        P(rmutex);      //读者读完数据要离开,先取得读者计数器的互斥量
        readcount = readcount - 1;  //读者数量减一
        if(readcount == 0)          //如果是最后一个离开的读者,释放读-写互斥量
            V(wmutex);
        V(rmutex);      //释放读者计数器的互斥量
        use_dataread(); //读者使用数据
    }
}

void writer(void)       //写者进程
{
    while (1)           //进程得到调度
    {
        think_up_data();//写者产生数据
        P(wmutex);      //获得写-写,读-写操作互斥量
        write_data_base();          //写入数据库
        V(wmutex);      //释放写-写,读-写操作互斥量
    }
}
```

3.哲学家进餐问题-解决办法

(1)至多只允许四个哲学家同时进餐,以保证至少有一个哲学家可以获得二只筷子:

```cpp
typedef int semaphore;                  //定义信号量
semaphore chopstick[5]= {1,1,1,1,1};    //初始化信号量
semaphore eating = 4;                   //仅允许四个哲学家可以进餐

void philosopher(int i)                 //第 i 个哲学家的程序
{
    while(1)
    {
        thinking();                     //工作之一
        P(eating);                      //请求进餐,若是第五个则先挨饿
        P(chopstick[i]);                //请求左手边的筷子
        P(chopstick[(i+1)%5]);          //请求右手边的筷子
        eating();                       //进餐
        V(chopstick[(i+1)%5]);          //释放右手边的筷子
        V(chopstick[i]);                //释放左手边的筷子
        V(eating);                      //释放信号量给其他挨饿的哲学家
    }
}
```

(2)另一种解决方法,仅当哲学家的左、右两支筷子均可用时,才允许他拿起筷子进餐。

```cpp
typedef int semaphore;                  //定义信号量
semaphore chopstick[5]= {1,1,1,1,1};    //初始化信号量
```

```
03. semaphore mutex = 1;                    //设置取筷子的信号量
04.
05. void philosopher(int i)                 //第 i 个哲学家的程序
06. {
07.     while(1)
08.     {
09.         thinking();
10.         P(mutex);                       //在取筷子前获得互斥量
11.         P(chopstick[i]);
12.         P(chopstick[(i+1)]%5);
13.         V(mutex);                       //释放互斥量
14.         eating();
15.         V(chopstick[(i+1)]%5);
16.         V(chopstick[i]);
17.     }
18. }
```

(3)规定奇数号哲学家先拿起其左边筷子,然后再去拿右边筷子;而偶数号哲学家则相反。程序代码如下:

```
[cpp]
01. typedef int semaphore;                  //定义信号量
02. semaphore chopstick[5]= {1,1,1,1,1};    //初始化信号量
03.
04. void philosopher(int i)                 //第 i 个哲学家的程序
05. {
06.     while(1)
07.     {
08.         thinking();
09.         if(i%2 == 0)                    //偶数哲学家,先右后左
10.         {
11.             P(chopstick[i+1]%5);
12.             P(chopstick[i]);
13.             eating();
14.             V(chopstick[i+1]%5);
15.             V(chopstick[i]);
16.         }
17.         else                            //奇数哲学家,先左后右
18.         {
19.             P(chopstick[i]);
20.             P(chopstick[i+1]%5) ;
21.             eating();
22.             V(chopstick[i]);
23.             V(chopstick[i+1]%5);
24.         }
25.     }
26. }
```



上一篇 操作系统学习笔记_4_进程管理 --处理机调度

下一篇 C++ Primer 学习笔记_101_特殊工具与技术 --运行时类型识别

反编译Android应用

操作系统总结之文件系统实现



查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap