- **1) Linux 中主要有哪几种内核锁?**

  Linux 的同步机制从 2.0 到 2.6 以来不断发展完善。从最初的原子操作,到后来的信号量,从大内核锁到今天的自旋锁。这些同步机制的发展伴随 Linux 从单处理器到对称多处理器的过渡;

  伴随着从非抢占内核到抢占内核的过度。Linux 的锁机制越来越有效,也越来越复杂。

  Linux 的内核锁主要是自旋锁和信号量。

  自旋锁最多只能被一个可执行线程持有,如果一个执行线程试图请求一个已被争用(已经被持有)的自旋锁,那么这个线程就会一直进行忙循环——旋转——等待锁重新可用。要是锁未被争用,请求它的执行线程便能立刻得到它并且继续进行。自旋锁可以在任何时刻防止多于一个的执行线程同时进入临界区。

  Linux 中的信号量是一种睡眠锁。如果有一个任务试图获得一个已被持有的信号量时,信号量会将其推入等待队列,然后让其睡眠。这时处理器获得自由去执行其它代码。当持有信号量的进程将信号量释放后,在等待队列中的一个任务将被唤醒,从而便可以获得这个信号量。

  信号量的睡眠特性,使得信号量适用于锁会被长时间持有的情况;只能在进程上下文中使用,因为中断上下文中是不能被调度的;另外当代码持有信号量时,不可以再持有自旋锁。

  Linux 内核中的同步机制:原子操作、信号量、读写信号量和自旋锁的 API,另外一些同步机制,包括大内核锁、读写锁、大读者锁、RCU (Read-Copy Update,顾名思义就是读-拷贝修改),和顺序锁。

  2) Linux 中的用户模式和内核模式是什么含意?

  MS-DOS 等操作系统在单一的 CPU 模式下运行,但是一些类 Unix 的操作系统则使用了双模式,可以有效地实现时间共享。在 Linux 机器上,CPU 要么处于受信任的内核模式,要么处于受限制的用户模式。除了内核本身处于内核模式以外,所有的用户进程都运行在用户模式之中。

  内核模式的代码可以无限制地访问所有处理器指令集以及全部内存和 I/O 空间。如果用户模式的进程要享有此特权,它必须通过系统调用向设备驱动程序或其他内核模式的代码发出请求。另外,用户模式的代码允许发生缺页,而内核模式的代码则不允许。

  在 2.4 和更早的内核中,仅仅用户模式的进程可以被上下文切换出局,由其他进程抢占。除非发生以下两种情况,否则内核模式代码可以一直独占 CPU:

  (1) 它自愿放弃 CPU;

  (2) 发生中断或异常。

  2.6 内核引入了内核抢占,大多数内核模式的代码也可以被抢占。

  3) 怎样申请大块内核内存?

在 Linux 内核环境下,申请大块内存的成功率随着系统运行时间的增加而减少,虽然可以通过 vmalloc 系列调用申请物理不连续但虚拟地址连续的内存,但毕竟其使用效率不高且在 32 位系统上 vmalloc 的内存地址空间有限。所以,一般的建议是在系统启动阶段申请大块内存,但是其成功的概率也只是比较高而已,而不是 100%。如果程序真的比较在意这个申请的成功与否,只能退用“启动内存”(Boot Memory)。下面就是申请并导出启动内存的一段示例代码:

```
void* x_bootmem = NULL;
EXPORT_SYMBOL(x_bootmem);
unsigned long x_bootmem_size = 0;
EXPORT_SYMBOL(x_bootmem_size);
static int __init x_bootmem_setup(char *str)
{
x_bootmem_size = memparse(str, &str);
x_bootmem = alloc_bootmem(x_bootmem_size);
printk(“Reserved %lu bytes from %p for x\n”, x_bootmem_size, x_bootmem);
return 1;
}
__setup(“x-bootmem=”, x_bootmem_setup);
```

可见其应用还是比较简单的,不过利弊总是共生的,它不可避免也有其自身的限制:

内存申请代码只能连接进内核,不能在模块中使用。

被申请的内存不会被页分配器和 slab 分配器所使用和统计,也就是说它处于系统的可见内存之外,即使在将来的某个地方你释放了它。

一般用户只会申请一大块内存,如果需要在其上实现复杂的内存管理则需要自己实现。

在不允许内存分配失败的场合,通过启动内存预留内存空间将是我们唯一的选择。

4) 用户进程间通信主要哪几种方式?

(1)管道(Pipe):管道可用于具有亲缘关系进程间的通信,允许一个进程和另一个与它有共同祖先的进程之间进行通信。

(2)命名管道(named pipe):命名管道克服了管道没有名字的限制,因此,除具有管道所具有的功能外,它还允许无亲缘关系进程间的通信。命名管道在文件系统中有对应的文件名。命名管道通过命令 mkfifo 或系统调用 mkfifo 来创建。

(3)信号(Signal):信号是比较复杂的通信方式,用于通知接受进程有某种事件发生,除了用于进程间通信外,进程还可以发送信号给进程本身;linux 除了支持 Unix 早期信号语义函数 sigal 外,还支持语义符合 Posix.1 标准的信号函数 sigaction(实际上,该函数是基于 BSD 的,BSD 为了实现可靠信号机制,又能够统一对外接口,用 sigaction 函数重新实现了 signal 函数)。

(4)消息(Message)队列:消息队列是消息的链接表,包括 Posix 消息队列 system V 消息队列。有足够权限的进程可以向队列中添加消息,被赋予读权限的进程则可以读走队列中的消息。消息队列克服了信号承载信息量少,管道只能承载无格式字节流以及缓冲区大小受限等缺

(5)共享内存:使得多个进程可以访问同一块内存空间,是最快的可用 IPC 形式。是针对其他通信机制运行效率较低而设计的。往往与其它通信机制,如信号量结合使用,来达到进程间的同步及互斥。

(6)信号量(semaphore):主要作为进程间以及同一进程不同线程之间的同步手段。

(7)套接字(Socket):更为一般的进程间通信机制,可用于不同机器之间的进程间通信。起初是由 Unix 系统的 BSD 分支开发出来的,但现在一般可以移植到其它类 Unix 系统上:Linux 和 System V 的变种都支持套接字。

5) 通过伙伴系统申请内核内存的函数有哪些?

在物理页面管理上实现了基于区的伙伴系统(zone based buddy system)。对不同区的内存使用单独的伙伴系统(buddy system)管理,而且独立地监控空闲页。相应接口 alloc_pages(gfp_mask, order),_ _get_free_pages(gfp_mask, order)等。

6) Linux 虚拟文件系统的关键数据结构有哪些?(至少写出四个)

struct super_block,struct inode,struct file,struct dentry;

7) 对文件或设备的操作函数保存在那个数据结构中?

struct file_operations

8) Linux 中的文件包括哪些?

执行文件,普通文件,目录文件,链接文件和设备文件,管道文件。

9) 创建进程的系统调用有那些?

clone(),fork(),vfork();系统调用服务例程:sys_clone,sys_fork,sys_vfork;

10) 调用 schedule()进行进程切换的方式有几种?

1.系统调用 do_fork();

2.定时中断 do_timer();

3.唤醒进程 wake_up_process

4.改变进程的调度策略 setscheduler();

5.系统调用礼让 sys_sched_yield();

11) Linux 调度程序是根据进程的动态优先级还是静态优先级来调度进程的?

Liunx 调度程序是根据根据进程的动态优先级来调度进程的,但是动态优先级又是根据静态优先级根据算法计算出来的,两者是两个相关联的值。因为高优先级的进程总是比低优先级

的进程先被调度,为防止多个高优先级的进程占用 CPU 资源,导致其他进程不能占有 CPU,所以引用动态优先级概念

12) 进程调度的核心数据结构是哪个?

struct runqueue

13) 如何加载、卸载一个模块?

insmod 加载,rmmod 卸载

14) 模块和应用程序分别运行在什么空间?

模块运行在内核空间,应用程序运行在用户空间

15) Linux 中的浮点运算由应用程序实现还是内核实现?

应用程序实现,Linux 中的浮点运算是利用数学库函数实现的,库函数能够被应用程序链接后调用,不能被内核链接调用。这些运算是在应用程序中运行的,然后再把结果反馈给系统。Linux 内核如果一定要进行浮点运算,需要在建立内核时选上 math-emu,使用软件模拟计算浮点运算,据说这样做的代价有两个:用户在安装驱动时需要重建内核,可能会影响到其他的应用程序,使得这些应用程序在做浮点运算的时候也使用 math-emu,大大的降低了效率。

16) 模块程序能否使用可链接的库函数?

模块程序运行在内核空间,不能链接库函数。

17) TLB 中缓存的是什么内容?

TLB,页表缓存,当线性地址被第一次转换成物理地址的时候,将线性地址和物理地址的对应放到 TLB 中,用于下次访问这个线性地址时,加快转换速度。

18) Linux 中有哪几种设备?

字符设备和块设备。网卡是例外,他不直接与设备文件对应,mknod 系统调用用来创建设备文件。

19) 字符设备驱动程序的关键数据结构是哪个?

字符设备描述符 struct cdev,cdev_alloc()用于动态的分配 cdev 描述符,cdev_add()用于注册一个 cdev 描述符,cdev 包含一个 struct kobject 类型的数据结构它是核心的数据结构

20) 设备驱动程序包括哪些功能函数?

open(),read(),write(),llseek(),realse();

21) 如何唯一标识一个设备?

Linux 使用一个设备编号来唯一的标示一个设备,设备编号分为:主设备号和次设备号,一般主设备号标示设备对应的驱动程序,次设备号对应设备文件指向的设备,在内核中使用 dev_t 来表示设备编号,一般它是 32 位长度,其中 12 位用于表示主设备号,20 位用于表示次设备号,利用 MKDEV(int major,int minor);用于生成一个 dev_t 类型的对象。

22) Linux 通过什么方式实现系统调用?

靠软件中断实现的,首先,用户程序为系统调用设置参数,其中一个编号是系统调用编号,参数设置完成后,程序执行系统调用指令,x86 上的软中断是有 int 产生的,这个指令会导致一个异常,产生一个事件,这个事件会导致处理器跳转到内核态并跳转到一个新的地址。并开始处理那里的异常处理程序,此时的异常处理就是系统调用程序。

23) Linux 软中断和工作队列的作用是什么?

Linux 中的软中断和工作队列是中断处理。

1.软中断一般是“可延迟函数”的总称,它不能睡眠,不能阻塞,它处于中断上下文,不能进城切换,软中断不能被自己打断,只能被硬件中断打断(上半部),可以并发的运行在多个 CPU 上。所以软中断必须设计成可重入的函数,因此也需要自旋锁来保护其数据结构。

2.工作队列中的函数处在进程上下文中,它可以睡眠,也能被阻塞,能够在不同的进程间切换。已完成不同的工作。

可延迟函数和工作队列都不能访问用户的进程空间,可延时函数在执行时不可能有任何正在运行的进程,工作队列的函数有内核进程执行,他不能访问用户空间地址。