

中断与调度部分的复习题

同济大学计算机系 操作系统作业
学号 2151769

2023-11-20

姓名 吕博文

一、判断题

- 1、进程不执行系统调用就不会入睡。 (错)
- 2、现运行进程不响应中断就不会被剥夺。 (对)
- 3、现运行进程不响应中断就不会让出 CPU。 (错)
- 4、现运行进程让出 CPU 后，一定是优先级最高的进程上台运行。 (对)
- 5、Unix V6++系统使用的调度算法是时间片轮转调度。 (对)
- 6、没有中断就没有调度（现运行进程就不会让出 CPU）。 (错)
- 7、用户态进程，系统中至多只有一个。 (错)
- 8、Unix V6++，核心态不调度。所以，如果不是入睡或终止，现运行进程不会让出 CPU。 (对)

二、系统调用不同于一般的子程序调用。请问：UNIX V6++和 Linux 的系统调用如何

- 1、传递应用程序想要执行的系统调用号？ UNIX V6++使用特定的寄存器传递系统调用号，通常使用 R0 寄存器，Linux 中一般使用 EAX 寄存器传递系统调用号
- 2、传递系统调用的参数？ UNIX V6++中，通常使用寄存器来传递系统调用参数吗，Linux 中系统调用参数通常通过 EBX, ECX, EDX, EBP 等寄存器传递
- 3、将系统调用的返回结果传给应用程序？ UNIX V6++系统调用的返回值通常存储在 EAX 寄存器中，Linux 系统调用的返回值存储在 rax 寄存器中。

三、言简意赅

- 1、描述 20#系统调用的执行过程。
 - (1) 系统调用号传递，应用程序通过某种方式将系统调用号传递给操作系统。
 - (2) 用户态切换到内核态，
 - (3) 系统调用服务例程，内核根据传递的系统调用号找到相应的系统调用服务例程
 - (4) 参数传递，应用程序传递的系统调用参数从用户空间拷贝到内核空间
 - (5) 系统调用执行，系统调用 20#号进程的执行代码

- (6) 返回值传递, 返回值从内核空间通过寄存器传给用户空间
- (7) 内核态到用户态切换
- 2、描述为 Unix V6++ 系统添加一个新的系统调用的过程。
 - (1) 在系统调用子程序表中添加新的入口
 - (2) 在 SystemCall 类中添加系统调用子程序的定义。

四、请回答以下问题, 言简意赅补齐系统中中断响应和调度过程。

1、T0 时刻整数秒, 系统中 SRUN 进程 PA 和 PB。现运行进程 PA 执行 sleep (10) 系统调用。

PA 执行 sleep(10) 的系统调用之后进入阻塞状态, 在整数秒的边界, 系统中断发生, 进入中断处理程序, 发现 PA 进程处于睡眠状态, 中断处理程序减少 PA 的睡眠计时器, 之后执行调度程序, 因为 PA 处于阻塞状态, 所以调度程序选择 PB 上台执行, 10s 过后, PA 从睡眠状态唤醒, 重新加入可运行队列, 等待调度系统选择它运行。

2、现运行进程 PA SRUN, 正在执行系统调用。T1 时刻, 响应中断, 唤醒一个睡眠进程 PB。问, PB 进程何时上台运行? 简述系统中中断响应, 调度过程和 PB 唤醒后上台运行

PA 响应中断后, 进入中断处理程序进行相关工作, 之后由于 PA 正在系统调用, 所以不会进行调度, 而是首先等待 PA 系统调用完成之后再行调度, PA 系统调用完成之后, 进入调度程序, 此时 PB 可能会上台。

3、T2 时刻整数秒, CPU 关中断执行硬盘中断处理程序, 硬盘中断处理程序的先前态是用户态。时钟中断何时响应? 时钟 time 的调整是否会延迟, 延迟到什么时候?

CPU 处于关中断时, 时钟中断无法做出相应, 只有等待硬盘中断处理程序完成以后, 开中断之后, 时钟中断才会响应, 如果在开中断之前错过了一个时钟中断, 那么系统时间的维护可能就会出现偏差, 时钟 time 的调整就会延迟, 延迟的具体时刻取决于中断的发生事件, 中断处理程序的执行时间以及开中断的时间, 指导开中断后被重新打开。

五、擦掉红色的判断, Unix V6++ 系统的钟就不走了。为什么? (这个题写着玩)

```
void Time::Clock( struct pt_regs* regs, struct pt_context* context )
{
    .....
    if( current->p_stat == Process::SRUN &&
        (context->xcs & USER_MODE) == KERNEL_MODE )
    {
        发 EOI 命令;
        return;
    }

    Time::lbolt -= HZ;
    Time::time++; //修改 wall clock time
```

```
.....  
}
```

红色代码是时钟中断处理程序中判断当前运行进程状态是否为系统运行态以及上下文是否为内核模式，如果是，则发出 EOI 命令，并返回，不再执行下面的时钟更新代码。如果红色判断被去掉，那么时钟中断处理程序会无条件的执行更新代码，而不管当前进程状态和上下文的模式如何，这可能导致时钟的正常计数和更新被打断，影响系统时间的正确维护。

六、Setpri()有没有一点儿怪。说出你的疑惑，尝试解释原系统设计的合理性，或对它进行质疑。
(这个题写着玩)

质疑：缺乏参数： setpri() 函数看起来并没有参数传递，这可能让人觉得奇怪，因为通常设置进程优先级时，你会期望传递一个新的优先级值。

可能的解释：基于当前进程： 如果 setpri() 函数是基于当前进程的当前状态来设置优先级，而不是依赖于显式的参数传递，这样的设计可能是为了简化接口和调用。

七、你自己的任何设计或想法
(这个题写着玩)

安全增强： 引入更强大的安全性特性，如硬件隔离、安全启动等，以应对现代的网络威胁。也可以考虑加强用户和进程的身份验证机制。