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　参考答案　　2023-11-20
学号　　姓名

一、判断题

1、进程不执行系统调用就不会入睡。
错。例外：Unix V6++，堆栈扩展。0#进程睡眠等待执行换入换出操作。
基于虚拟存储器的计算机系统，缺页中断。
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。入睡或终止也会让出 CPU。
（但入睡或终止需要进程执行系统调用，强调系统调用是广义中断，那这题也是对的）
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。
错。优先级最高的有可能是睡眠进程或者在盘交换区上，没资格运行的。
5、Unix V6++系统使用的调度算法是时间片轮转调度。
错。可剥夺的动态优先级调度算法。
6、没有中断就没有调度（现运行进程就不会让出 CPU）。
对。
7、用户态进程，系统中至多只有一个。
对。现运行进程执行应用程序时是系统唯一的用户态进程。
就绪，阻塞进程全在核心态。立即返回用户态的就绪进程也要先在核心态完成恢复用户态现场的操作。
8、Unix V6++，核心态不调度。所以，如果不是入睡或终止，现运行核心态进程不会让出 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？　用 EAX 寄存器
2、传递系统调用的参数？　用 EBX，ECX，EDX，ESI，EDI 寄存器
3、将系统调用的返回结果传给应用程序？　系统调用的返回值用 EAX 寄存器。其它数据，可以存入其它通用寄存器 或是 钩子函数传入的指针变量所指的用户空间内存区域。

三、言简意赅

（一）描述 20#系统调用的执行过程。
答：
1. 应用程序调用系统调用的钩子函数
2. 钩子函数将系统调用号 20 传入 EAX，执行 int 80h 陷入内核

3. 系统调用入口函数 SystemCallEntrance()保存用户态寄存器，将系统调用号 20 存入核心栈。
4. 系统调用处理程序 Trap()，从核心栈取出系统调用号，查系统调用表 m_SystemEntranceTable，间接调用系统调用子程序 Sys_Getpid( )。
5. Sys_Getpid( )将系统调用的返回值（现运行进程的 p_pid）存入核心栈、u_ar0 指向的单元。系统调用返回用户态后该单元的值将回传 EAX 寄存器。钩子函数将其传回应用程序。

（二）描述为 Unix V6++系统添加一个新的系统调用的过程。
答：
1、新建一个系统调用子程序 Sys_***；修改.h，加入该子程序的声明。
2、系统调用表 m_SystemEntranceTable 中寻找为 null 的表项 i。填入新系统调用子程序的入口地址和参数的数量。下标 i 是分配给这个新系统调用的系统调用号。
3、lib/sys.c 新建该系统调用的钩子函数；修改 lib/include/sys.h，加入钩子函数的声明。
4、重新编译、生成 内核和 Unix V6++的静态库。

补充：
（三）描述 Unix V6++中断响应流程。
答:
1、 硬件保存现场：启用现运行进程核心栈，将 CS，EIP，SS、ESP、EFLAGS 压入核心栈。
2、 硬件：用中断号查中断向量表，将中断入口函数的入口地址赋给 CS 和 EIP。
3、 执行中断入口函数，压栈、保护 其余的段寄存器和通用寄存器。赋值其余段寄存器让 CPU 全面转入核心态运行。
4、 调用中断处理程序，对中断请求进行具体的处理。
5、 中断处理程序返回后，中断入口函数会调度：
如果中断改变了系统的调度状态，被中断的现运行进程会放弃 CPU 给优先级最高的进程。（Unix V6，返回用户态时才会调度。现代 Unix 不是这样）。
6、 被选中的最高优先级进程恢复现场，IRET。

四、请回答以下问题，言简意赅补齐系统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。
答：
- PA 执行系统调用，陷入核心态，保护用户态现场。
- 执行 sleep 系统调用处理函数 Sys_Sslep。设置 tout 值之后，入睡，放弃 CPU。
- PB 进程被选中，上台运行。

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？简述系统中断响应，调度过程和 PB 唤醒后上台运行
答:
系统调用完成时，PB 进程上台运行。

系统中断响应和调度过程如下:
中断，唤醒睡眠进程 PB。先前态是核心态，中断返回时不调度，现运行进程继续执行

被打断的系统调用。系统调用的先前态是用户态，所以系统调用完成后，现运行进程放弃 CPU 给被中断处理程序唤醒的进程 PB。

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？

答：

开中断后系统响应时钟中断。

时钟 time 的调整，这次时钟中断不会做。会延迟，至先前态是用户态的下次时钟中断。

4、T3 时刻整数秒，现运行进程 PA 正在执行应用程序。内存中的 PC 进程（SLOAD）闹钟到期。（注，仅考虑 PA、PC 进程）。

答：

- PA 响应时钟中断，执行时钟中断处理程序。这次时钟中断的先前态是用户态。
- 时钟中断处理程序执行整数秒时刻需要执行的定时器（闹钟）服务：唤醒 PC 进程，RunRun++。
- 中断返回用户态前调度，PA 让出 CPU 给 PC 执行 Sleep 系统调用。
- PC 执行完系统调用后，计算、重置自己的用户态优先数 p_pri，runrun++，重新调度。
- PA 或 PC，随便谁，先返回用户态执行应用程序。随后，2 个进程时间片轮转。

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
    ……
}
```

系统 idle 的时候，整数秒要调整 time。此时，CPU KERNEL_MODE，现运行的 0#进程 SSLEEP。删掉红色的判断，idle 时走 if 分支，无法调整 lbolt 和 time。

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）

Unix 的理念，所有行为基于现运行进程，也就是，只有现运行进程优先级下降时才考虑让出 CPU。

现运行进程，Setpri 重算优先数，RunRun++。

对非现运行进程，Setpri 仅重算优先数。RunRun 会错置，激活调度，但 select 选中的一定是优先级最高的进程。

七、你自己的任何设计或想法　　（这个题写着玩）