**云南大学软件学院**

**实 验 报 告**

课程： 操作系统实验 任课教师： 储　星 实验指导教师： 谢 诚

姓名： 陈俊宏 学号： 20211060245 专业： 人工智能 日期： 2023-05-05 成绩：

**实验三 进程调度**

1. **实验目的**

1、学习分析进程的描述、创建、调度和切换技术；

2、学习进程调度算法；

3、如何跟踪到由时钟中断引发的进程调度。

**二、实验内容**

1、分析Linux 0.11的进程调度算法，该操作系统采用了一种时间片与优先级相结合的调度算法，在当前进程的时间片用完后会进行调度，调度时会选择剩余时间片最大的就绪进程；

2、跟踪到由时钟中断引发的进程调度；

3、查看进程的剩余时间片；

4、查进程的信息；

5查看进程控制块信息。

**三、 实验要求**

1、本次实验不分组。

2、5月16日前提交纸质报告。

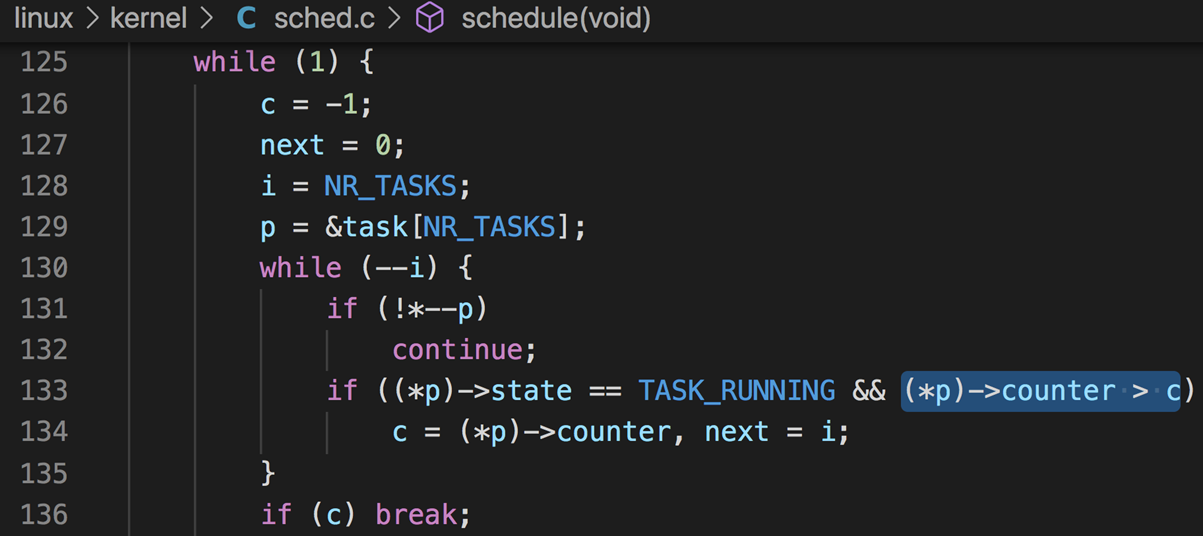
3、描述清楚实验过程，特别是第一次由时钟中断引发的进程调度和前三次进程调度分析，撰写实验总结。

**四、 实验过程**

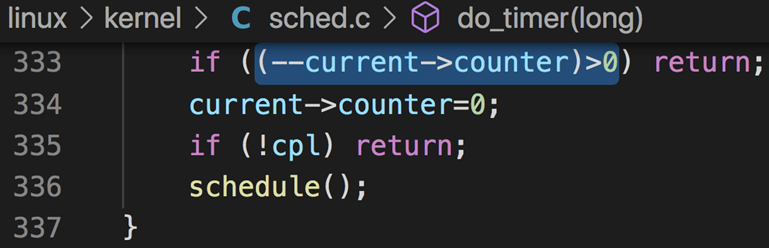
1、分析Linux 0.11的进程调度算法：

#### Linux 0.11 的进程调度算法是怎样的

采用了一种时间片与优先级相结合的调度算法。每次调度时，会选择剩余时间片最大的就绪进程来运行：

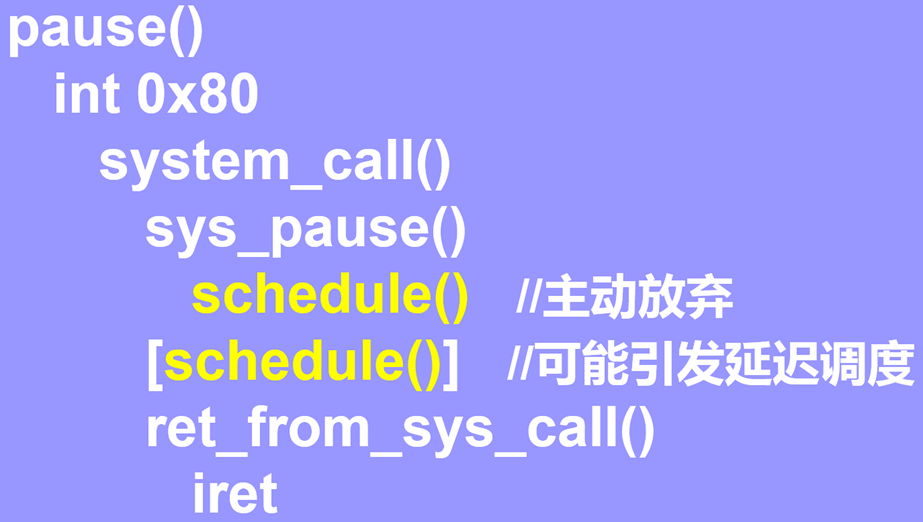


执行到上面的第 136 行时，next 和 c 分别记录了被选中进程在 task 数组中的下标和它的剩余时间片。同时，如果当前进程的时间片已经用完且时钟中断发生时系统处于用户态，则会启动进程调度：

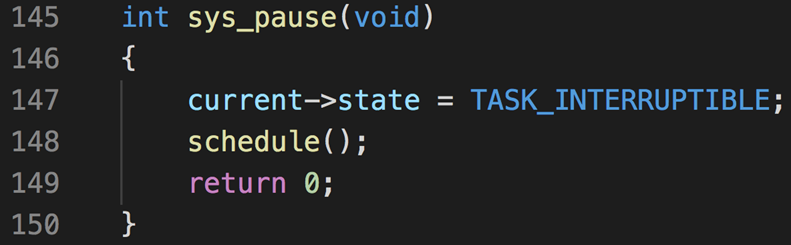


#### 在执行一次 pause 系统调用时，可能引发几次进程调度

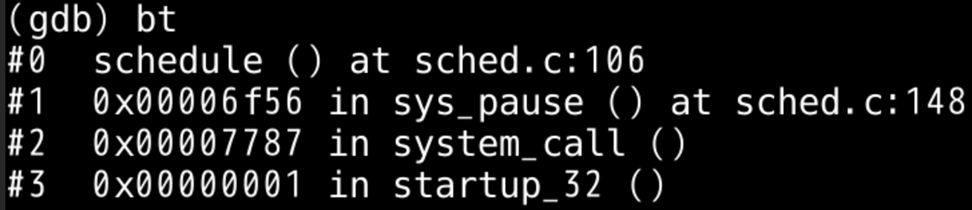
最多可能引发两次进程调度，调用关系如下：



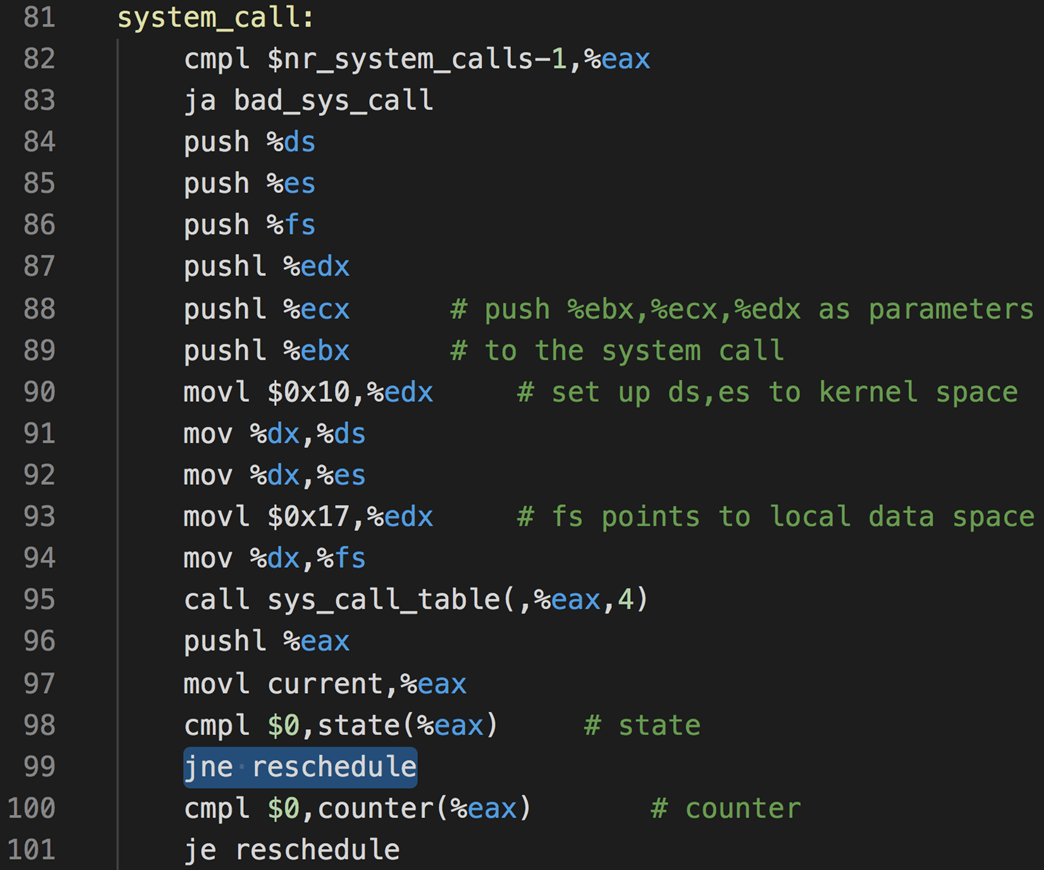
其中第一调度是由 pause 的实现函数 sys\_pause 直接调用的：

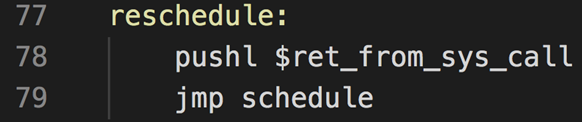


这次调度在函数调用栈中看来是这样的：

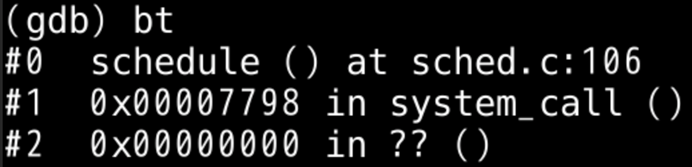


而第二次调度是由系统调用的总控函数 system\_call 在执行完系统调用之后、返回之前进行的，其激发条件是：当前进程不处于就绪态或者当前进程的时间片已经用完，如下图所示：





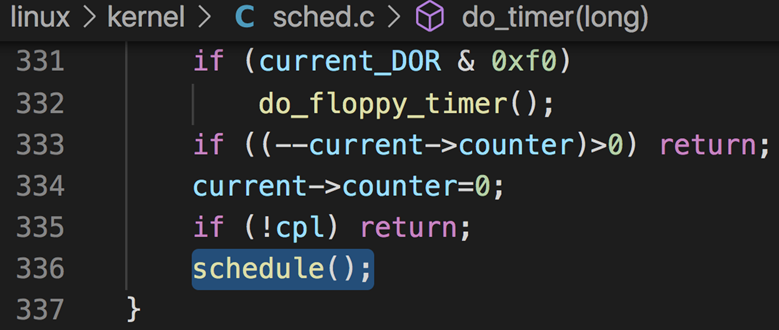
这次调度在函数调用栈中看来是这样的：



2、跟踪到由时钟中断引发的进程调度：

#### 如何跟踪到由时钟中断引发的进程调度

时钟中断处理程序 timer\_interrupt 会调用 C 函数 do\_timer ，后者有可能进行进程调度，具体条件是：当前进程的时间片已经用完且中断发生时系统处于用户态，如下所示：



只要在其中的第 336 行设置断点即可。

3、查看进程的剩余时间片：

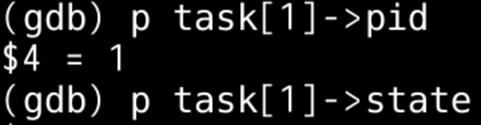
#### 如何查看进程的剩余时间片



4、查进程的信息：

#### 如何查看 1 号进程的信息

可直接通过 task 数组找到该进程的进程控制块并查看：

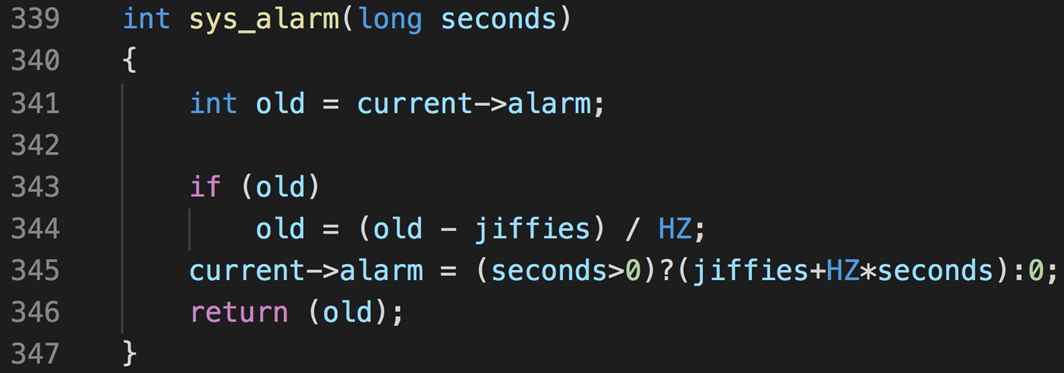


5、查看进程控制块信息：

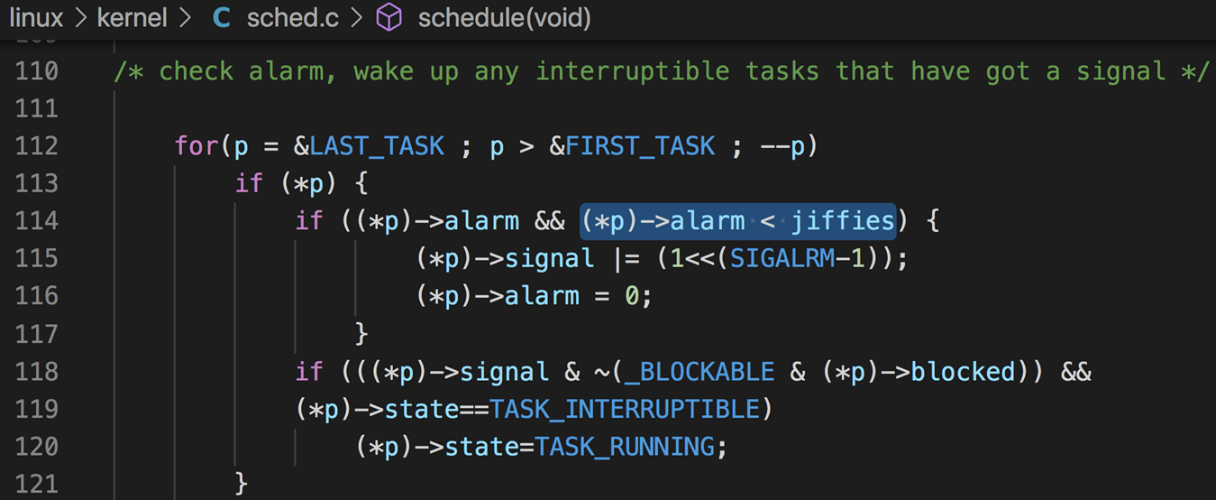
#### 进程控制块中的 alarm 成员的作用是什么

该成员记录了一个定时器截止时间，单位为时钟中断周期（在 0.11 中为 10ms ，类似于全局变量 jiffies ），用户程序执行 alarm 系统调用时会设置该变量：





在进程调度时会检查每个进程的该定时器字段，如果已超时，就向该进程发 SIGALRM 信号并唤醒它：



第一次由时钟中断引发的进程调度：



前三次进程调度：



**五、实验总结**

答：

在本次实验中，我们学习了进程的描述、创建、调度和切换技术，以及进程调度算法和如何跟踪由时钟中断引发的进程调度。具体来说，我们分析了Linux 0.11的进程调度算法，该算法采用了时间片和优先级相结合的调度策略，选择剩余时间片最大的就绪进程进行调度。

通过跟踪时钟中断，我们了解了进程调度是如何在操作系统中实现的。我们还学习了如何查看进程的剩余时间片和进程控制块信息，这些信息对于调试和优化进程调度算法非常有用。

总的来说，本次实验让我们更深入地了解了进程调度和操作系统的工作原理，为我们今后的学习和研究提供了基础和参考。