# Yield&Nice调研作业

## 吕佳鸿 10235501436

## Yield 系统调用

```
// Give up the CPU for one scheduling round.
void
yield(void)
{
  struct proc *p = myproc();
  acquire(&p->lock);
  p->state = RUNNABLE;
  sched();
  release(&p->lock);
}
```

yield 操作的作用是让出目前正在执行的线程放弃当前的执行，让出CPU 权限，使得 CPU 去执行其他的线程。处于让步状态的线程仍然处于可执行状态，但是该线程所对应的操作系统层面的线程从状态上来说会从执行状态变成就绪状态。

yield函数做了几件事情，它首先获取了进程的锁。在锁释放之前，进程的状态会变得不一致，例如，yield将要将进程的状态改为RUNABLE，表明进程并没有在运行，但是实际上这个进程还在运行，代码正在当前进程的内核线程中运行。所以这里加锁的目的之一就是：即使我们将进程的状态改为了RUNABLE，其他的CPU核的调度器线程也不可能看到进程的状态为RUNABLE并尝试运行它。否则的话，进程就会在两个CPU核上运行了，而一个进程只有一个栈，这意味着两个CPU核在同一个栈上运行代码（注，因为XV6中一个用户进程只有一个用户线程）。

接下来yield函数中将进程的状态改为RUNABLE。这里的意思是，当前进程要出让CPU，并切换到调度器线程。当前进程的状态是RUNABLE意味着它还会再次运行，因为现在是一个定时器中断打断了当前正在运行的进程。

## Nice命令

`nice`命令用于在Linux系统中设置或查看进程的优先级。进程的优先级决定了操作系统调度CPU资源给该进程的顺序。默认情况下，每个进程都有一个优先级值，称为“nice值”。较低的nice值意味着较高的优先级，而较高的nice值则意味着较低的优先级。通过`nice`命令，我们可以修改已运行进程的nice值，或者为新启动的进程设置特定的nice值。

在数据处理和分析中，`nice`命令可以帮助我们平衡系统资源的使用。例如，如果我们有一个需要大量计算资源的分析任务，并且我们不想让它完全占据所有的CPU资源，我们可以使用`nice`命令将其优先级降低，以便其他进程也能获得足够的CPU时间。

当Linux 内核尝试决定哪些运行中的进程可以访问 CPU 时，其中一个需要考虑的因素就是进程优先级的值（也称为 nice 值）。每个进程都有一个介于 -20 到 19 之间的 nice 值。默认情况下，进程的 nice 值为 0。

进程的 nice 值，可以通过 nice 命令和 renice 命令修改，进而调整进程的运行顺序

nice 命令格式如下：

```
[root@localhost ~] # nice [-n NI值] 命令
```

-n NI值：给命令赋予 NI 值，该值的范围为 -20~19；

renice 命令格式如下：

```
[root@localhost ~] # renice [优先级] PID
```

注意，此命令中使用的是进程的 PID 号，因此常与 ps 等命令配合使用。同 nice 命令恰恰相反，renice 命令可以在进程运行时修改其 NI 值，从而调整优先级。