优先权定义：

PRI ：进程优先权，代表这个进程可被执行的优先级，其值越小，优先级就越高，越早被执行

PRI即进程的优先级，或者通俗点说就是程序被CPU执行的先后顺序，此值越小进程的优先级别越高。那NI呢？就是我们所要说的nice值了，其表示进程可被执行的优先级的修正数值。如前面所说，PRI值越小越快被执行，那么加入nice值后，将会使得PRI变为：PRI(new)=PRI(old)+nice。由此看出，PR是根据NICE排序的，规则是NICE越小PR越前（小，优先权更大），即其优先级会变高，则其越快被执行。如果NICE相同则进程uid是root的优先权更大。

nice值定义：

在LINUX系统中，Nice值的范围从-20到+19（不同系统的值范围是不一样的），较大的值表示低优先级，较小的值表示高优先级，值为零则表示不会调整该进程的优先级。具有最高优先级的程序，其nice值最低，所以在LINUX系统中，值-20使得一项任务变得非常重要；与之相反，如果任务的nice为+19，则表示它是一个高尚的、无私的任务，允许所有其他任务比自己享有宝贵的CPU时间的更大使用份额，这也就是nice的名称的来意。

修正nice值的必要性：

进程在创建时被赋予不同的优先级值，而如前面所说，nice的值是表示进程优先级值可被修正数据值，因此，每个进程都在其计划执行时被赋予一个nice值，这样系统就可以根据系统的资源以及具体进程的各类资源消耗情况，主动干预进程的优先级值。在通常情况下，子进程会继承父进程的nice值，比如在系统启动的过程中，init进程会被赋予0，其他所有进程继承了这个nice值（因为其他进程都是init的子进程）。

实例：

对nice值一个形象比喻，假设在一个CPU轮转中，有2个runnable的进程A和B，如果他们的nice值都为0，假设内核会给他们每人分配1k个cpu时间片。但是假设进程A的为0，但是B的值为-10，那么此时CPU可能分别给A和B分配1k和1.5k的时间片。故可以形象的理解为，nice的值影响了内核分配给进程的cpu时间片的多少，时间片越多的进程，其优先级越高，其优先级值（PRI）越低。%nice，就是改变过优先级的进程的占用CPU的百分比，如上例中就是0.5k/2.5k=1/5=20%。

附注\*进程nice值和进程优先级不是一个概念，但是进程nice值会影响到进程的优先级变化。

对nice的修改命令分别是nice和renice：

1、nice命令就是设置一个要执行command进程的nice值，其命令格式是 nice –n adjustment command command\_option，如果这里不指定adjustment，则默认为10。

2、renice命令就是设置一个】

已经在运行的进程的nice值，假设一运行进程本来nice值为0，renice为3后，则这个运行进程的nice值就为3了。

说明：如果用户设置的nice值超过了nice的边界值（LINUX为-20到+19），系统就取nice的边界值作为进程的nice值。

权限。

文章原址：<http://blog.csdn.net/longdel/article/details/7317511>

黄色字体为修改内容

具体实现：

简单的实现读取进程的nice值和修改进程nice值的服务

参数flag为1时修改nice

参数为0时读取nice

SYSCALL\_DEFINE3(mysetnice, pid\_t, pid, int, flag, int, nicevalue){

int error = 0;

struct task\_struct \*p;

for(p = &init\_task;(p = next\_task(p)) != &init\_task;){

if(p->pid == pid){

if(flag == 0){

printk("the process's nice = %d",task\_nice(p));

} else if(flag == 1){

set\_user\_nice(p,nicevalue);

printk("the process changed to %d",task\_nice(p));

} else {

error = -EFAULT;

}

}

}

return error;

}

SYSCALL\_DEFINE3的3即为参数个数

上面程序的主要思路是遍历进程的，简单粗暴，但是效率不高，下面一种稍微麻烦点，但是更直接高效的方式。

有很多函数涉及到了linux内核的api。可以在网上找一份，下载地址如下：<http://download.csdn.net/detail/zhanglu231123/4584636>

SYSCALL\_DEFINE3(mysetnice, pid\_t, pid, int, flag, int, niceval)

{

struct pid \* kpid;

struct task\_struct \* task;

int nicebef;

kpid = find\_get\_pid(pid);/\* 返回pid \*/

task = pid\_task(kpid, PIDTYPE\_PID);/\* 返回task\_struct \*/

nicebef = task\_nice(task);/\* 返回进程当前nice值 \*/

if(flag == 1)

{

set\_user\_nice(task, niceval);/\* 修改进程nice值 \*/

printk("修改前nice值：%d\t修改后nice值：%d\n", nicebef, niceval);

return 0;

}

else if(flag == 0)

{

printk("该进程的nice值为%d\n", nicebef);

return 0;

}

return EFAULT;

}

最后重新编译内核

文章原址：<http://blog.tiaozaoj.com/index.php/archives/188/>