Linux 0.11进程调度

张延慈

输出进程的信息

- schedule is called by do_exit.
- Maybe 15 is the longest counter and 3 is the chosen one?
- The chosen task:
- 3: pid=3, state=0, counter = 15, father=1
- jiffies is 4

- all tasks as follows:
- 0: pid=0, state=1, counter = 14, father=-1
- jiffies is 4
- 1: pid=1, state=0, counter = 14, father=0
- jiffies is 4
- 2: pid=2, state=3, counter = 13, father=1
- jiffies is 4
- 3: pid=3, state=0, counter = 15, father=1
- jiffies is 4

```
struct task_struct {

    long state; //表示进程的状态，-1表示不可执行，0表示可执行，>0表示停止
    long counter;/* 运行时间片,以jiffs递减计数 */
    long priority; /* 运行优先数,开始时,counter = priority,值越大,表示优先数越高,等待时
    long signal;/* 信号.是一组位图,每一个bit代表一种信号. */
    struct sigaction sigaction[32]; /* 信号响应的数据结构，对应信号要执行的操作和标志信.
    long blocked;   /* 进程信号屏蔽码(对应信号位图) */
/* various fields */
    int exit_code; /* 任务执行停止的退出码,其父进程会取 */
    unsigned long start_code,end_code,end_data,brk,start_stack;/* start_code代码段地
end_data代码长度+数据长度(byte),brk总长度(byte),start_stack堆栈段地址 */
    long pid,father,pgrp,session,leader;/* 进程号,父进程号 ,父进程组号,会话号,会话头(发
    unsigned short uid,euid,suid;/* 用户id 号,有效用户 id 号,保存用户 id 号*/
    unsigned short gid,egid,sgid;/* 组标记号 (组id),有效组 id,保存的组id */
    long alarm;/* 报警定时值 (jiffs数) */
    long utime,stime,cutime,cstime,start_time;/* 用户态运行时间 (jiffs数),
系统态运行时间 (jiffs数),子进程用户态运行时间,子进程系统态运行时间,进程开始运行时刻 */
    unsigned short used_math;/* 是否使用了协处理器 */
/* file system info */
    int tty;        /* 进程使用tty的子设备号. -1表示设有使用 */
    unsigned short umask; /* 文件创建属性屏蔽位 */
    struct m_inode * pwd; /* 当前工作目录 i节点结构 */
    struct m_inode * root; /* 根目录i节点结构 */
    struct m_inode * executable;/* 执行文件i节点结构 */
    unsigned long close_on_exec; /* 执行时关闭文件句柄位图标志. */
    struct file * filp[NR_OPEN];
/* 文件结构指针表,最多32项. 表项号即是文件描述符的值 */
    struct desc_struct ldt[3];
/* 任务局部描述符表.0-空,1-cs段,2-Ds和Ss段 */
    struct tss_struct tss; /* 进程的任务状态段信息结构 */
```

输出调用进程调度函数的位置

- 在exit.c和schedule.c中，schedule.c被调用

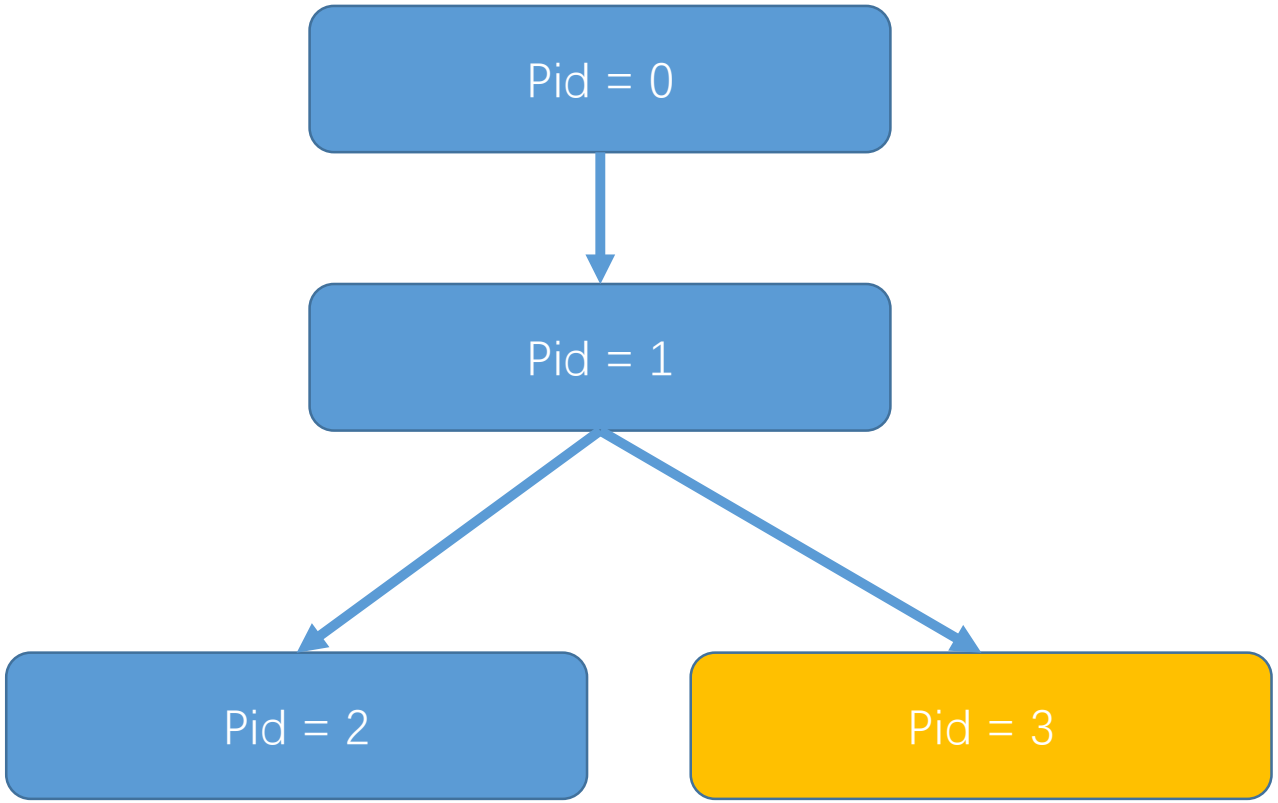
```
int sys_pause(void)
{
        current->state = TASK_INTERRUPTIBLE;
        log("Schedule is called by sys_pause.\n");
        schedule();
        log("\n");
        return 0;
}
void sleep_on(struct task_struct **p)
{
        struct task_struct *tmp;

        if (!p)
                return;
        if (current == &(init_task.task))
                panic("task[0] trying to sleep");
        tmp = *p;
        *p = current;
        current->state = TASK_UNINTERRUPTIBLE;
        log("Schedule is called by sleep_on.\n");
        schedule();
        log("\n");
        *p = tmp;
        if (tmp)
                tmp->state=TASK_RUNNING;
}
```

```
void show_task(int nr,struct task_struct * p)
{
        //int i,j = 4096-sizeof(struct task_struct);
        //printk("%d: pid=%d, state=%d, ",nr,p->pid,p->state);
        log("%d: pid=%d, state=%d, counter = %d, father=%d \njiffies is %d\n\n",nr,p->pid,p->state, p->counter, p->father, jiffies);
        //i=0;
        //while (i<j && !((char *)(p+1))[i])
        //      i++;
        //printk("%d (of %d) chars free in kernel stack\n\r",i,j);
        //log("%d (of %d) chars free in kernel stack\n",i,j);
}

void show_stat(void)
{
        int i;

        for (i=0;i<NR_TASKS;i++)
                if (task[i])
                        show_task(i,task[i]);
        log("\n");
}
```



- 3: pid=3, state=0, counter = 15, father=1
- jiffies is 4

- all tasks as follows:
- 0: pid=0, state=1, counter = 14, father=-1
- jiffies is 4
- 1: pid=1, state=0, counter = 14, father=0
- jiffies is 4
- 2: pid=2, state=3, counter = 13, father=1
- jiffies is 4
- 3: pid=3, state=0, counter = 15, father=1
- jiffies is 4

```
Schedule is called by sys_pause.
Maybe 15 is the longest counter and 1 is the chosen one?
The chosen task:
1: pid=1, state=0, counter = 15, father=0
jiffies is 1

all tasks as follows:
0: pid=0, state=1, counter = 14, father=-1
jiffies is 1

1: pid=1, state=0, counter = 15, father=0
jiffies is 1
```

```
Maybe 15 is the longest counter and 2 is the chosen one?
The chosen task:
2: pid=2, state=0, counter = 15, father=1
jiffies is 1

all tasks as follows:
0: pid=0, state=1, counter = 14, father=-1
jiffies is 1

1: pid=1, state=1, counter = 15, father=0
jiffies is 1

2: pid=2, state=0, counter = 15, father=1
jiffies is 1
```

```
schedule is called by do_exit.
Maybe 15 is the longest counter and 3 is the chosen one?
The chosen task:
3: pid=3, state=0, counter = 15, father=1
jiffies is 4

all tasks as follows:
0: pid=0, state=1, counter = 14, father=-1
jiffies is 4

1: pid=1, state=0, counter = 14, father=0
jiffies is 4

2: pid=2, state=3, counter = 13, father=1
jiffies is 4

3: pid=3, state=0, counter = 15, father=1
jiffies is 4
```

```
Schedule is called by sys_pause.
The chosen task:
0: pid=0, state=1, counter = 0, father=-1
jiffies is 25
```