Linux进程、线程和调度(1)

讲解时间：5月22-25日晚20点

宋宝华

麦当劳喜欢您来，喜欢您再来

扫描关注
Linuxer

第一次课大纲

* Linux进程生命周期(就绪、运行、睡眠、停止、僵死)
* 僵尸是个什么鬼？
* 停止状态与作业控制，cpulimit
* 内存泄漏的真实含义
* task_struct以及task_struct之间的关系
* 初见fork和僵尸

练习题

* fork的例子
* life-period例子，观察僵尸
* 用cpulimit控制CPU利用率

进程控制块PCB

task_struct
pid
...
*mm
*fs
*files
*signal

```
struct mm_struct {
    struct vm_area_struct * mmap;
    ...
    pgd_t * pgd;
};
```

```
struct fs_struct {
 * root,
 * pwd
};
```

```
struct files_struct {
    struct fdtable fdtab;
    struct file ___rcu *
fd_array[NR_OPEN_DEFAULT];
};
```
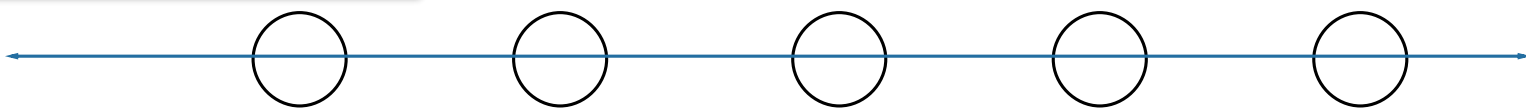
pid

pid的数量是有限的
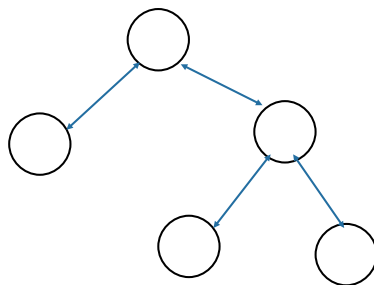
$ cat /proc/sys/kernel/pid_max
32768
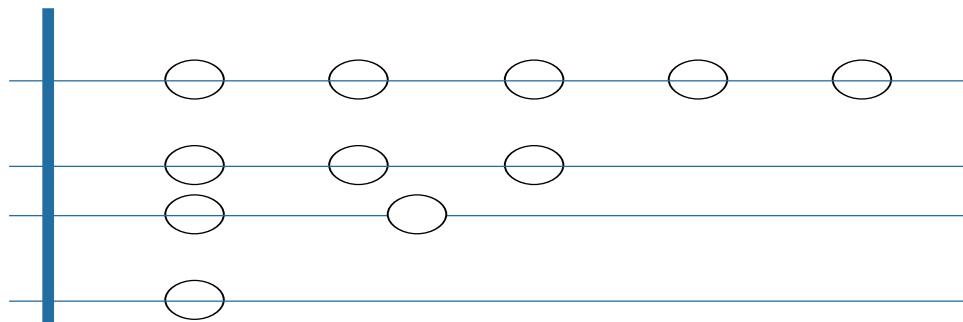
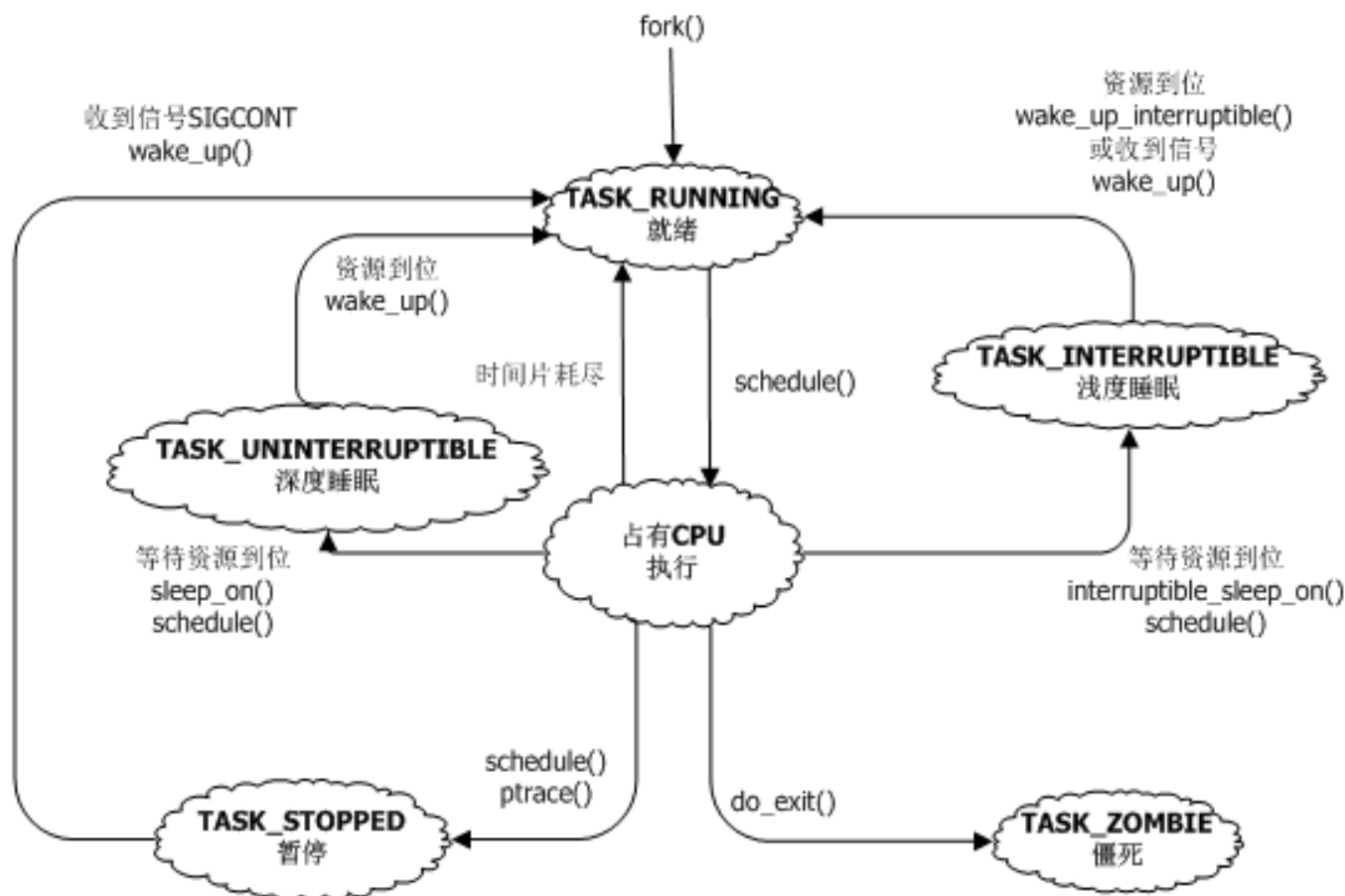**Fork炸弹**

:(){ :|:& };:

task_struct被管理

形成链表

形成树

形成哈希：pid-> task_struct

进 程 生 命 周 期

僵尸是什么

资源已经释放：无内存泄漏等
task_struct还在：父进程可以查到子进程死因

```c
static int wait_task_zombie(struct wait_opts *wo, struct task_struct *p)
{
        int state, retval, status;
        pid_t pid = task_pid_vnr(p);
        uid_t uid = from_kuid_munged(current_user_ns(), task_uid(p));
        struct siginfo __user *infop;

        if (!likely(wo->wo_flags & WEXITED))
                return 0;

        if (unlikely(wo->wo_flags & WNOWAIT)) {
                int exit_code = p->exit_code;
                int why;

                get_task_struct(p);
                read_unlock(&tasklist_lock);
                sched_annotate_sleep();

                if ((exit_code & 0x7f) == 0) {
                        why = CLD_EXITED;
                        status = exit_code >> 8;
                } else {
                        why = (exit_code & 0x80) ? CLD_DUMPED : CLD_KILLED;
                        status = exit_code & 0x7f;
                }
```
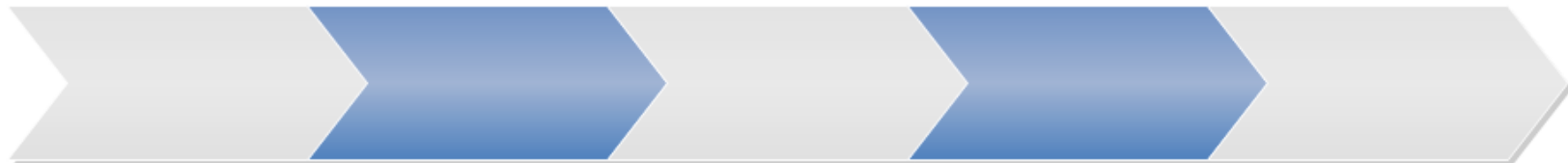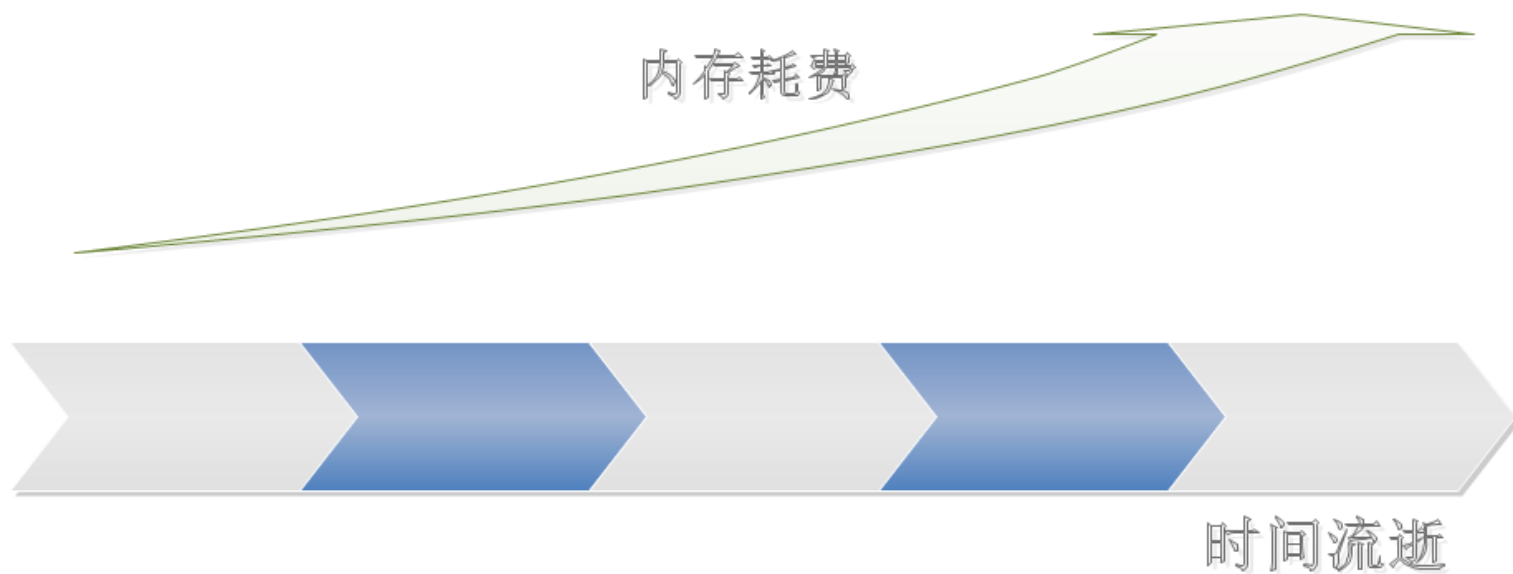
内存泄漏到底是什么？

不是：进程死了，内存没释放

内存消耗

死亡

自由落体为0

时间流逝

内存泄漏到底是什么 (cont.)?

而是：进程活着，运行越久，耗费内存越多

内存耗费

时间流逝

作 业 控 制

ctrl+ z, fg/bg
cpulimit

cpulimit -l 20 -p 10111
限制pid 为10111程序 的 cpu使用率不超过 10%

stop		cont	stop		cont					

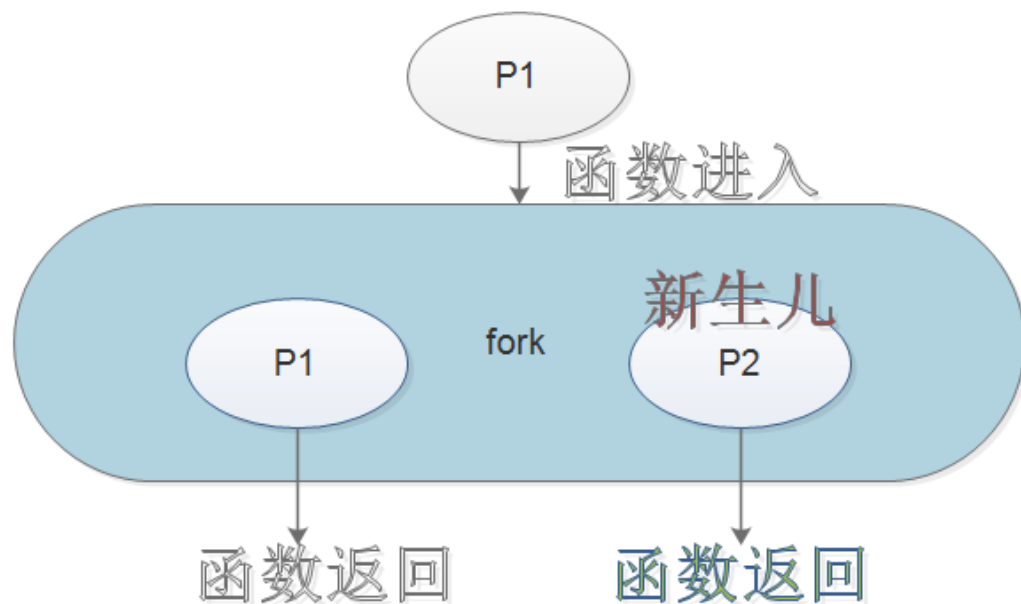| 运行 | 暂停 | 运行 | 暂停 | 运行 | 暂停 | | | | |
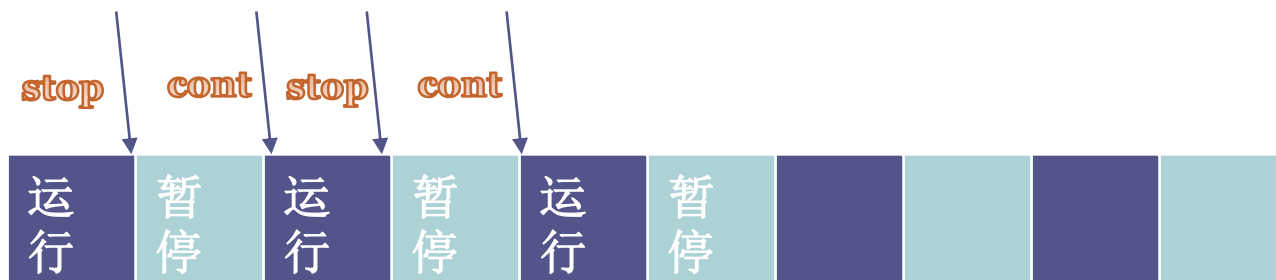
fork

打印几个hello？

```
1 main()
2 {
3         fork();
4         printf("hello\n");
5         fork();
6         printf("hello\n");
7         while(1);
8 }
```

P1

函数进入

新生儿

P1   fork   P2

函数返回   函数返回

fork(cont.)

怎么打印？

```c
6  int main(void)
7  {
8          pid_t pid,wait_pid;
9          int status;
10
11         pid = fork();
12
13         if (pid==-1)     {
14                 perror("Cannot create new process");
15                 exit(1);
16         } else  if (pid==0) {
17                 printf("a\n");
18         } else {
19                 printf("b\n");
20         }
21
22         printf("c\n");
23         while(1);
24 }
```

子死父清场

```c
pid = fork();

if (pid==-1)    {
        perror("Cannot create new process");
        exit(1);
} else  if (pid==0) {
        printf("child process id: %ld\n", (long) getpid());
        pause();
        _exit(0);
} else {
        wait_pid=waitpid(pid, &status, WUNTRACED | WCONTINUED);

        if (wait_pid == -1) {
                perror("cannot using waitpid function");
                exit(1);
        }

        if(WIFSIGNALED(status))
                printf("child process is killed by signal %d\n", WTERMSIG(status));
```

课程练习源码

https://github.com/21cnbao/process-courses

谢 谢！