操作系统及Linux内核

西安邮电大学

进程控制块及Linux中的task_struct结构

什么是进程控制块？

什么是进程控制块?

进程控制块
Process Control Block ,简称PCB)

是一种数据结构，用于描述进程的当前情况，以及控制进程运行的全部信息。

- 存放进程的属性信息
- 与进程同生同灭
- 一个进程对应一个PCB
- OS通过PCB控制和管理进程

进程控制块中的信息

- 进程标识信息
- 处理器现场信息
- 进程调度信息
- 进程控制信息

唯一的标识进程，比如PID。

处理器现场信息

通用寄存器	8 到32个, RISC结构中超过100个
指令计数器	下一条指令的地址
状态寄存器	程序状态字 PSW,如:EFLAGS寄存器
用户栈指针	过程和系统调用参数及地址

进程调度信息

- 进程状态
  - 如: 运行，就绪，阻塞...
- 进程优先级
  - 优先级高的进程优先获得处理机
- 该进程在等待的事件
  - 阻塞的原因
- 调度所需其它信息
  - 如: 等待总时间, 执行总时间

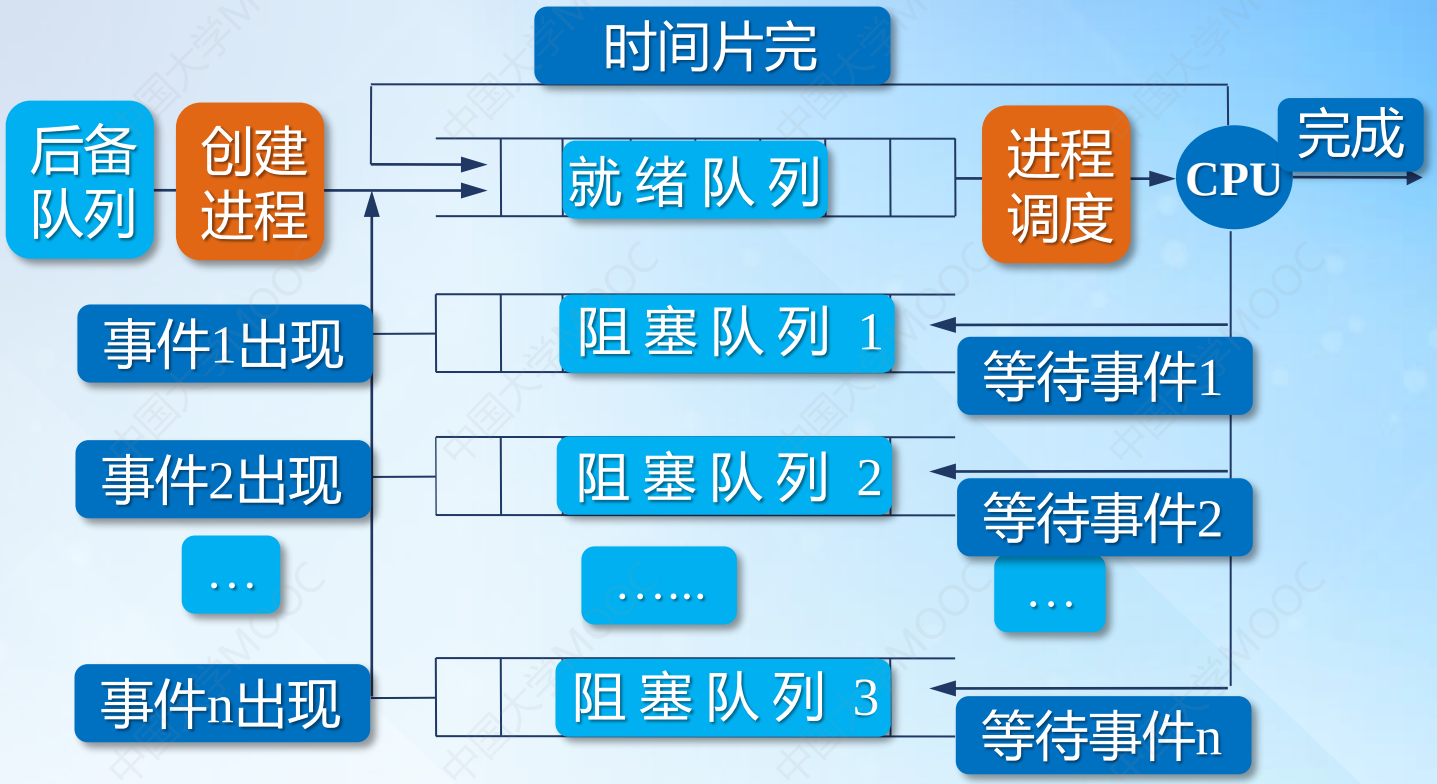
进程控制信息

程序和数据的地址	程序和数据所在的内存或外存地址
进程间同步和通信机制	需要的消息队列指针和信号量等
资源清单及使用情况	除CPU外的资源: 文件，I/O设备...
家族关系	父-子进程关系及其它结构

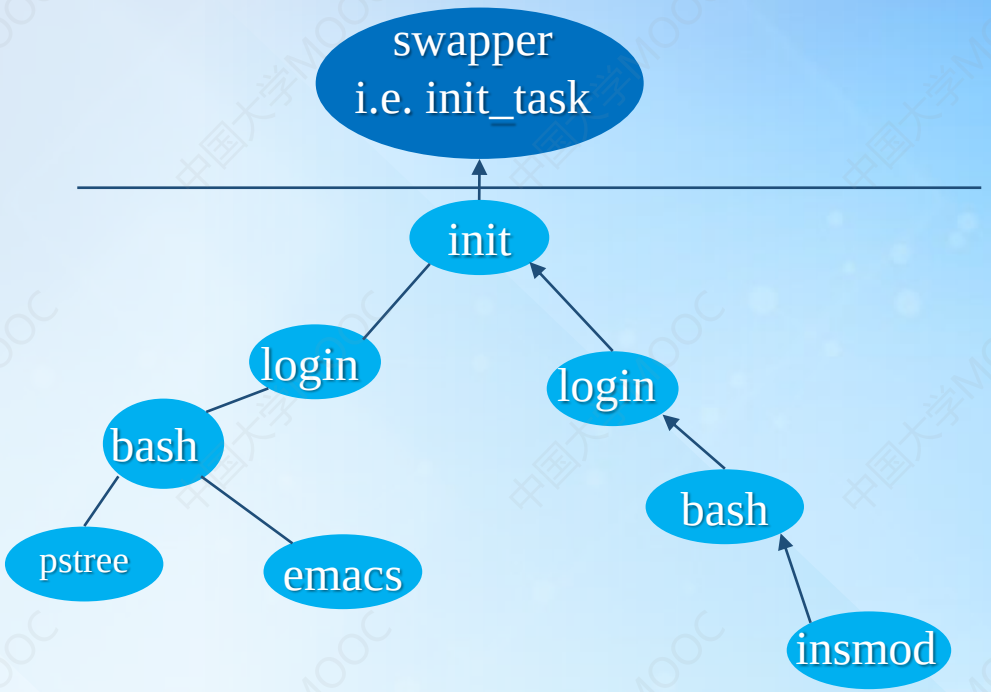
进程控制块组织方式

- 链表方式
- 索引方式
- 线性表
- 树型结构

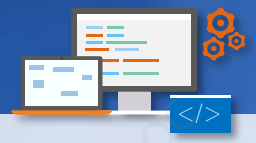
PCB组织方式-各种队列



Linux中进程的家族关系



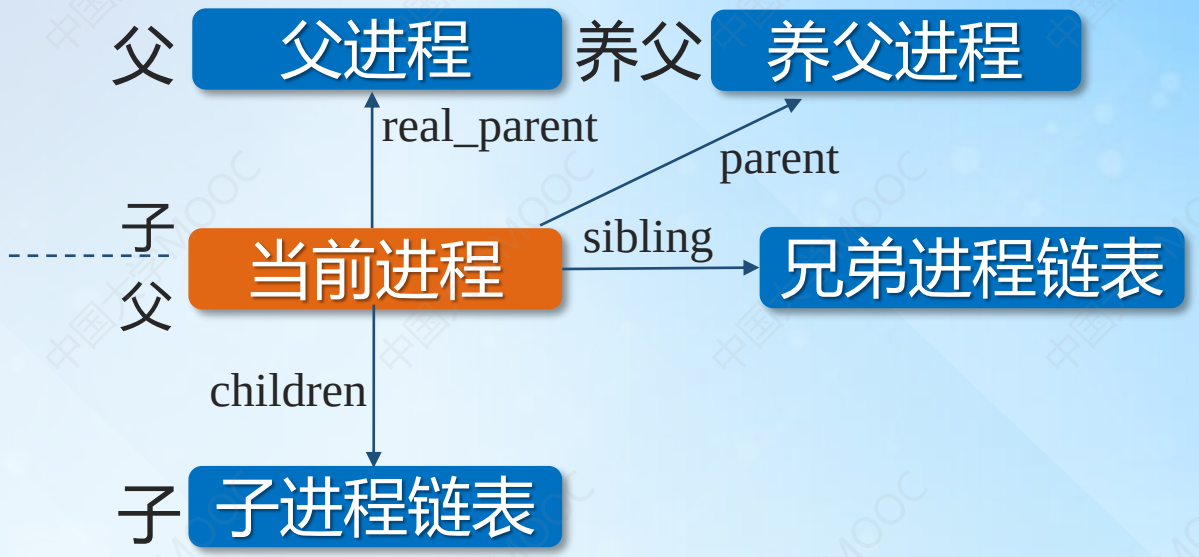
$pstree 命令查看进程树

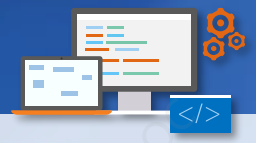


Linux中的task_struct结构

```
struct task_struct {
        volatile long state;                    /*进程状态*/
        int pid, uid, gid;                      /*一些标识符*/
        struct task_struct *real_parent;  /*真正创建当前进程
                                                的进程*/
        struct task_struct *parent;          /*相当于养父*/
        struct list_head children;           /*子进程链表*/
        struct list_head sibling;            /*兄弟进程链表*/
        struct task_struct *group_leader;/*线程组的头进程*/
      …
};
```

Linux中父子进程关系图

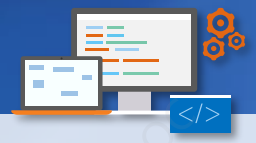




task_struct源代码片段

```
struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
        /*
         * For reasons of header soup (see current_thread_info()), this
         * must be the first element of task_struct.
         */
        struct thread_info              thread_info;
#endif
        /* -1 unrunnable, 0 runnable, >0 stopped: */
        volatile long                   state;

        pid_t                           pid;
        pid_t                           tgid;
```



task_struct源代码片段

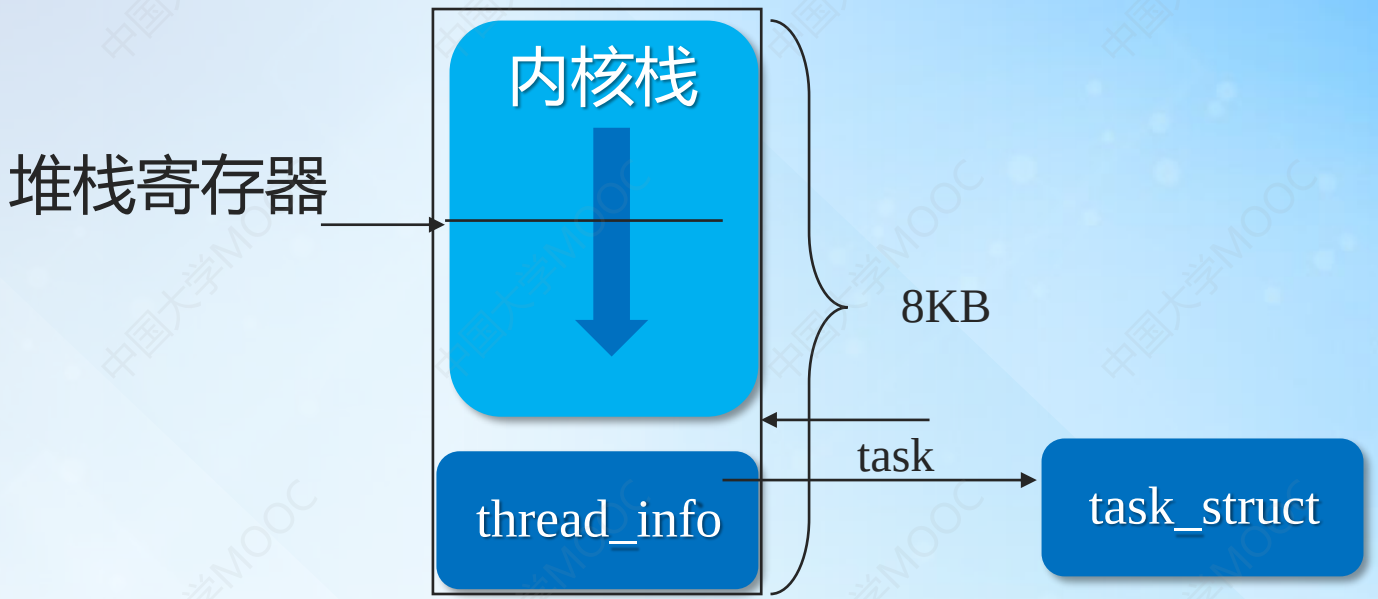
```
/* Real parent process: */
struct task_struct __rcu        *real_parent;

/* Recipient of SIGCHLD, wait4() reports: */
struct task_struct __rcu        *parent;

/*
 * Children/sibling form the list of natural children:
 */
struct list_head                children;
struct list_head                sibling;
struct task_struct              *group_leader;

/*
```

Linux中如何存放进程控制块？



Linux中如何存放进程控制块？

- 内核中使用下列的联合结构表示这样一个混合结构：

```
union thread_union {
        struct thread_info thread_info;
        unsigned long stack[THREAD_SIZE/sizeof(long)];
        /*大小一般是8KB，但也可以配置为4KB*/
};
```

- x86上，thread_info的定义如下：

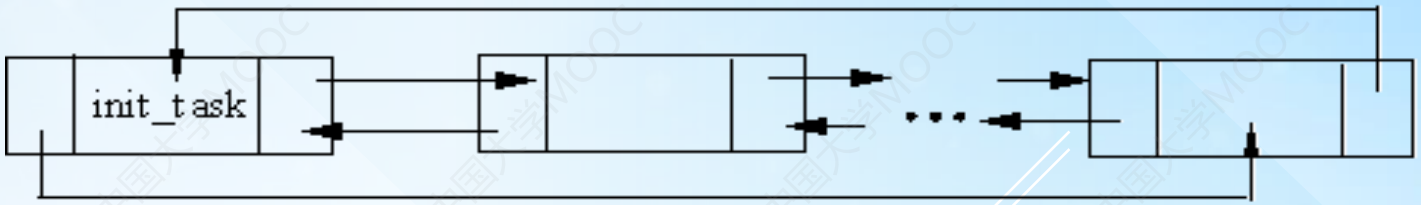
```
union thread_info {
        struct task_struct *task;
        struct exec_domain *exec_domain;
        ……
};
```

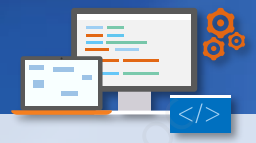
thread_info存放进程PCB中频繁访问和需要快速访问的字段。

进程的组织方式-进程链表

在task_struct中定义如下:

```
struct task_struct {
        ...
    struct list_head tasks;
   char comm[TASK_COMM_LEN];/*可执行程序的名字
        ...
};
```





动手实践-打印进程控制块中的字段

```
static int print_pid( void)
{
    struct task_struct *task,*p;
     struct list_head *pos;
     int count=0;
     printk("Hello World enter begin:\n");
     task=&init_task;
     list_for_each(pos,&task->tasks)   /*关键*/
              {
              p=list_entry(pos, struct task_struct, tasks);
              count++;
              printk("%d--->%s\n",p->pid,p->comm);
              }
     printk(the number of process is:%d\n",count);
     return 0;
}
```

task_struct小结

处理器环境信息

状态信息

链接信息

时间和定时器

各种标识符

PCB

调度信息

进程间通信

文件系统信息

虚拟内存信息