volatile long state; //说明了该进程是否可以执行,还是可中断等信息\_unsigned long flags; //Flage 是进程号,在调用fork()时给出\_int sigpending; //进程上是否有待处理的信号\_mm\_segment\_t addr\_limit; //进程地址空间,区分内核进程与普通进程在内存存放的位置不同\_i volatile long need\_resched;//调度标志,表示该进程是否需要重新调度,若非0,则当从内核态返回到用户态,会发生调度\_int lock\_depth; //锁深度\_long nice; //进程的基本时间片\_//进程的调度策略,有三种,实时进程:SCHED\_FIFO,SCHED\_RR, 分时进程:SCHED\_OTHER\_unsigned long policy; \_struct mm\_struct \*mm; //进程内存管理信息\_int processor;//若进程不在任何CPU上运行, cpus\_runnable 的值是0，否则是1 这个值在运行队列被锁时更新\_unsigned long cpus\_runnable, cpus\_allowed; \_struct list\_head run\_list; //指向运行队列的指针\_unsigned long sleep\_time; //进程的睡眠时间\_//用于将系统中所有的进程连成一个双向循环链表, 其根是init\_task\_struct task\_struct \*next\_task, \*prev\_task; \_struct mm\_struct \*active\_mm; \_struct list\_head local\_pages; //指向本地页面\_unsigned int allocation\_order, nr\_local\_pages; \_struct linux\_binfmt \*binfmt; //进程所运行的可执行文件的格式\_int exit\_code, exit\_signal; \_int pdeath\_signal; //父进程终止时向子进程发送的信号\_unsigned long personality;//Linux可以运行由其他UNIX操作系统生成的符合iBCS2标准的程序\_int did\_exec:1; \_pid\_t pid; //进程标识符,用来代表一个进程\_pid\_t pgrp; //进程组标识,表示进程所属的进程组\_pid\_t tty\_old\_pgrp; //进程控制终端所在的组标识\_pid\_t session; //进程的会话标识\_pid\_t tgid; \_int leader; //表示进程是否为会话主管\_struct task\_struct \*p\_opptr,\*p\_pptr,\*p\_cptr,\*p\_ysptr,\*p\_osptr; \_struct list\_head thread\_group; //线程链表\_struct task\_struct \*pidhash\_next; //用于将进程链入HASH表\_struct task\_struct \*\*pidhash\_pprev; \_wait\_queue\_head\_t wait\_chldexit; //供wait4()使用\_struct completion \*vfork\_done; //供vfork() 使用\_unsigned long rt\_priority; //实时优先级，用它计算实时进程调度时的weight值\_unsigned long it\_real\_value, it\_prof\_value, it\_virt\_value; \_unsigned long it\_real\_incr, it\_prof\_incr, it\_virt\_value; \_struct timer\_list real\_timer; //指向实时定时器的指针\_struct tms times; //记录进程消耗的时间\_unsigned long start\_time; //进程创建的时间\_long per\_cpu\_utime[NR\_CPUS], per\_cpu\_stime[NR\_CPUS]; /记录进程在每个CPU上所消耗的用户态时间和核心态时间\_//内存缺页和交换信息: \_//min\_flt, maj\_flt累计进程的次缺页数（Copy on　Write页和匿名页）和主缺页数（从映射文件或交换\_//设备读入的页面数）； nswap记录进程累计换出的页面数，即写到交换设备上的页面数。//cmin\_flt, cmaj\_flt, cnswap记录本进程为祖先的所有子孙进程的累计次缺页数，主缺页数和换出页面数。\_//在父进程回收终止的子进程时，父进程会将子进程的这些信息累计到自己结构的这些域中\_unsigned long min\_flt, maj\_flt, nswap, cmin\_flt, cmaj\_flt, cnswap; \_int swappable:1; //表示进程的虚拟地址空间是否允许换出\_//uid,gid为运行该进程的用户的用户标识符和组标识符，通常是进程创建者的uid，gid\_//euid，egid为有效uid,gid\_//fsuid，fsgid为文件系统uid,gid，这两个ID号通常与有效uid,gid相等，在检查对于文件

//系统的访问权限时使用他们。\_//suid，sgid为备份uid,gid\_uid\_t uid,euid,suid,fsuid; \_gid\_t gid,egid,sgid,fsgid; \_int ngroups; //记录进程在多少个用户组中\_gid\_t groups[NGROUPS]; //记录进程所在的组\_//进程的权能，分别是有效位集合，继承位集合，允许位集合\_kernel\_cap\_t cap\_effective, cap\_inheritable, cap\_permitted; \_int keep\_capabilities:1; \_struct user\_struct \*user; \_struct rlimit rlim[RLIM\_NLIMITS]; //与进程相关的资源限制信息\_unsigned short used\_math; //是否使用FPU\_char comm[16]; //进程正在运行的可执行文件名 //文件系统信息\_int link\_count, total\_link\_count; \_//NULL if no tty 进程所在的控制终端，如果不需要控制终端，则该指针为空\_struct tty\_struct \*tty; \_unsigned int locks; \_//进程间通信信息\_struct sem\_undo \*semundo; //进程在信号灯上的所有undo操作\_struct sem\_queue \*semsleeping; //当进程因为信号灯操作而挂起时，他在该队列中记录等待的操作\_//进程的CPU状态，切换时，要保存到停止进程的task\_struct中\_struct thread\_struct thread;//文件系统信息struct fs\_struct \*fs; \_//打开文件信息struct files\_struct \*files; \_//信号处理函数spinlock\_t sigmask\_lock; \_struct signal\_struct \*sig; //信号处理函数\_sigset\_t blocked; //进程当前要阻塞的信号，每个信号对应一位\_struct sigpending pending; //进程上是否有待处理的信号\_unsigned long sas\_ss\_sp;