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int 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_OTHERunsigned 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_taskstruct 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 FPUchar 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;