**内容概要**

* 程序和进程
* 内核中的进程结构
* C程序启动过程
* 进程终止方式
* 非局部跳转
* 进程资源限制
* 进程创建、执行和终止
* 进程类型
* 进程状态
* 进程组

**进程**

* 程序

1. 程序是存放在磁盘文件中的可执行文件

* 进程

1. 程序的执行实例被称为进程
2. 进程具有独立的权限与职责。如果系统中某个进程奔溃，它不会影响到其余进程。
3. 每个进程运行在其各自的虚拟地址空间中，进程之间可以通过由内核控制的机制相互通讯。

* 进程ID

1. 每个linux进程都一定有一个唯一的数字标识符，成为进程ID进程ID总是一非负数。

进程描述符（进程表项）

struct task\_struct {

volatile long state; /\* -1 unrunnable, 0 runnable, >0 stopped \*/

void \*stack;

atomic\_t usage;

unsigned int flags; /\* per process flags, defined below \*/

unsigned int ptrace;

int lock\_depth; /\* BKL lock depth \*/

#ifdef CONFIG\_SMP

#ifdef \_\_ARCH\_WANT\_UNLOCKED\_CTXSW

int oncpu;

#endif

#endif

int load\_weight; /\* for niceness load balancing purposes \*/

int prio, static\_prio, normal\_prio;

struct list\_head run\_list;

struct prio\_array \*array;

unsigned short ioprio;

#ifdef CONFIG\_BLK\_DEV\_IO\_TRACE

unsigned int btrace\_seq;

#endif

unsigned long sleep\_avg;

unsigned long long timestamp, last\_ran;

unsigned long long sched\_time; /\* sched\_clock time spent running \*/

enum sleep\_type sleep\_type;

unsigned int policy;

cpumask\_t cpus\_allowed;

unsigned int time\_slice, first\_time\_slice;

#if defined(CONFIG\_SCHEDSTATS) || defined(CONFIG\_TASK\_DELAY\_ACCT)

struct sched\_info sched\_info;

#endif

struct list\_head tasks;

/\*

\* ptrace\_list/ptrace\_children forms the list of my children

\* that were stolen by a ptracer.

\*/

struct list\_head ptrace\_children;

struct list\_head ptrace\_list;

struct mm\_struct \*mm, \*active\_mm;

/\* task state \*/

struct linux\_binfmt \*binfmt;

int exit\_state;

int exit\_code, exit\_signal;

int pdeath\_signal; /\* The signal sent when the parent dies \*/

/\* ??? \*/

unsigned int personality;

unsigned did\_exec:1;

pid\_t pid;

pid\_t tgid;

#ifdef CONFIG\_CC\_STACKPROTECTOR

/\* Canary value for the -fstack-protector gcc feature \*/

unsigned long stack\_canary;

#endif

/\*

\* pointers to (original) parent process, youngest child, younger sibling,

\* older sibling, respectively. (p->father can be replaced with

\* p->parent->pid)

\*/

struct task\_struct \*real\_parent; /\* real parent process (when being debugged) \*/

struct task\_struct \*parent; /\* parent process \*/

/\*

\* children/sibling forms the list of my children plus the

\* tasks I'm ptracing.

\*/

struct list\_head children; /\* list of my children \*/

struct list\_head sibling; /\* linkage in my parent's children list \*/

struct task\_struct \*group\_leader; /\* threadgroup leader \*/

/\* PID/PID hash table linkage. \*/

struct pid\_link pids[PIDTYPE\_MAX];

struct list\_head thread\_group;

struct completion \*vfork\_done; /\* for vfork() \*/

int \_\_user \*set\_child\_tid; /\* CLONE\_CHILD\_SETTID \*/

int \_\_user \*clear\_child\_tid; /\* CLONE\_CHILD\_CLEARTID \*/

unsigned int rt\_priority;

cputime\_t utime, stime;

unsigned long nvcsw, nivcsw; /\* context switch counts \*/

struct timespec start\_time;

/\* mm fault and swap info: this can arguably be seen as either mm-specific or thread-specific \*/

unsigned long min\_flt, maj\_flt;

cputime\_t it\_prof\_expires, it\_virt\_expires;

unsigned long long it\_sched\_expires;

struct list\_head cpu\_timers[3];

/\* process credentials \*/

uid\_t uid,euid,suid,fsuid;

gid\_t gid,egid,sgid,fsgid;

struct group\_info \*group\_info;

kernel\_cap\_t cap\_effective, cap\_inheritable, cap\_permitted;

unsigned keep\_capabilities:1;

struct user\_struct \*user;

#ifdef CONFIG\_KEYS

struct key \*request\_key\_auth; /\* assumed request\_key authority \*/

struct key \*thread\_keyring; /\* keyring private to this thread \*/

unsigned char jit\_keyring; /\* default keyring to attach requested keys to \*/

#endif

/\*

\* fpu\_counter contains the number of consecutive context switches

\* that the FPU is used. If this is over a threshold, the lazy fpu

\* saving becomes unlazy to save the trap. This is an unsigned char

\* so that after 256 times the counter wraps and the behavior turns

\* lazy again; this to deal with bursty apps that only use FPU for

\* a short time

\*/

unsigned char fpu\_counter;

int oomkilladj; /\* OOM kill score adjustment (bit shift). \*/

char comm[TASK\_COMM\_LEN]; /\* executable name excluding path

- access with [gs]et\_task\_comm (which lock

it with task\_lock())

- initialized normally by flush\_old\_exec \*/

/\* file system info \*/

int link\_count, total\_link\_count;

#ifdef CONFIG\_SYSVIPC

/\* ipc stuff \*/

struct sysv\_sem sysvsem;

#endif

/\* CPU-specific state of this task \*/

struct thread\_struct thread;

/\* filesystem information \*/

struct fs\_struct \*fs;

/\* open file information \*/

struct files\_struct \*files;

/\* namespaces \*/

struct nsproxy \*nsproxy;

/\* signal handlers \*/

struct signal\_struct \*signal;

struct sighand\_struct \*sighand;

sigset\_t blocked, real\_blocked;

sigset\_t saved\_sigmask; /\* To be restored with TIF\_RESTORE\_SIGMASK \*/

struct sigpending pending;

unsigned long sas\_ss\_sp;

size\_t sas\_ss\_size;

int (\*notifier)(void \*priv);

void \*notifier\_data;

sigset\_t \*notifier\_mask;

void \*security;

struct audit\_context \*audit\_context;

seccomp\_t seccomp;

/\* Thread group tracking \*/

u32 parent\_exec\_id;

u32 self\_exec\_id;

/\* Protection of (de-)allocation: mm, files, fs, tty, keyrings \*/

spinlock\_t alloc\_lock;

/\* Protection of the PI data structures: \*/

spinlock\_t pi\_lock;

#ifdef CONFIG\_RT\_MUTEXES

/\* PI waiters blocked on a rt\_mutex held by this task \*/

struct plist\_head pi\_waiters;

/\* Deadlock detection and priority inheritance handling \*/

struct rt\_mutex\_waiter \*pi\_blocked\_on;

#endif

#ifdef CONFIG\_DEBUG\_MUTEXES

/\* mutex deadlock detection \*/

struct mutex\_waiter \*blocked\_on;

#endif

#ifdef CONFIG\_TRACE\_IRQFLAGS

unsigned int irq\_events;

int hardirqs\_enabled;

unsigned long hardirq\_enable\_ip;

unsigned int hardirq\_enable\_event;

unsigned long hardirq\_disable\_ip;

unsigned int hardirq\_disable\_event;

int softirqs\_enabled;

unsigned long softirq\_disable\_ip;

unsigned int softirq\_disable\_event;

unsigned long softirq\_enable\_ip;

unsigned int softirq\_enable\_event;

int hardirq\_context;

int softirq\_context;

#endif

#ifdef CONFIG\_LOCKDEP

# define MAX\_LOCK\_DEPTH 30UL

u64 curr\_chain\_key;

int lockdep\_depth;

struct held\_lock held\_locks[MAX\_LOCK\_DEPTH];

unsigned int lockdep\_recursion;

#endif

/\* journalling filesystem info \*/

void \*journal\_info;

/\* stacked block device info \*/

struct bio \*bio\_list, \*\*bio\_tail;

/\* VM state \*/

struct reclaim\_state \*reclaim\_state;

struct backing\_dev\_info \*backing\_dev\_info;

struct io\_context \*io\_context;

unsigned long ptrace\_message;

siginfo\_t \*last\_siginfo; /\* For ptrace use. \*/

/\*

\* current io wait handle: wait queue entry to use for io waits

\* If this thread is processing aio, this points at the waitqueue

\* inside the currently handled kiocb. It may be NULL (i.e. default

\* to a stack based synchronous wait) if its doing sync IO.

\*/

wait\_queue\_t \*io\_wait;

#ifdef CONFIG\_TASK\_XACCT

/\* i/o counters(bytes read/written, #syscalls \*/

u64 rchar, wchar, syscr, syscw;

#endif

struct task\_io\_accounting ioac;

#if defined(CONFIG\_TASK\_XACCT)

u64 acct\_rss\_mem1; /\* accumulated rss usage \*/

u64 acct\_vm\_mem1; /\* accumulated virtual memory usage \*/

cputime\_t acct\_stimexpd;/\* stime since last update \*/

#endif

#ifdef CONFIG\_NUMA

struct mempolicy \*mempolicy;

short il\_next;

#endif

#ifdef CONFIG\_CPUSETS

struct cpuset \*cpuset;

nodemask\_t mems\_allowed;

int cpuset\_mems\_generation;

int cpuset\_mem\_spread\_rotor;

#endif

struct robust\_list\_head \_\_user \*robust\_list;

#ifdef CONFIG\_COMPAT

struct compat\_robust\_list\_head \_\_user \*compat\_robust\_list;

#endif

struct list\_head pi\_state\_list;

struct futex\_pi\_state \*pi\_state\_cache;

atomic\_t fs\_excl; /\* holding fs exclusive resources \*/

struct rcu\_head rcu;

/\*

\* cache last used pipe for splice

\*/

struct pipe\_inode\_info \*splice\_pipe;

#ifdef CONFIG\_TASK\_DELAY\_ACCT

struct task\_delay\_info \*delays;

#endif

#ifdef CONFIG\_FAULT\_INJECTION

int make\_it\_fail;

#endif

};

