Apêndice A

O descritor de tarefa do Linux

A estrutura em linguagem C apresentada neste anexo constitui o descritor de tarefas (*Task Control Block*) do Linux (estudado na Seção 5.1). Ela foi extraída do arquivo `include/linux/sched.h` do código fonte do núcleo Linux versão 1.0. Essa versão do núcleo foi escolhida por sua simplicidade; na versão mais recente do código-fonte do Linux (4.18 nesta data), a mesma estrutura possui cerca de 600 linhas de código C.

```
1  // part of the Linux kernel source code, file include/linux/sched.h, version 1.0
2
3  struct task_struct {
4
5  /* these are hardcoded - don't touch */
6         volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
7         long counter;
8         long priority;
9         unsigned long signal;
10        unsigned long blocked; /* bitmap of masked signals */
11        unsigned long flags; /* per process flags, defined below */
12        int errno;
13        int debugreg[8]; /* Hardware debugging registers */
14
15 /* various fields */
16        struct task_struct *next_task, *prev_task;
17
18        struct sigaction sigaction[32];
19        unsigned long saved_kernel_stack;
20        unsigned long kernel_stack_page;
21        int exit_code, exit_signal;
22        int elf_executable:1;
23        int dumpable:1;
24        int swappable:1;
25        int did_exec:1;
26        unsigned long start_code, end_code, end_data, start_brk, brk,
27                start_stack, start_mmap;
28        unsigned long arg_start, arg_end, env_start, env_end;
29        int pid,pgrp,session,leader;
30        int    groups[NGROUPS];
31
32        /* pointers to (original) parent process, youngest child, younger sibling,
33         * older sibling, respectively. (p->father can be replaced with
34         * p->p_pptr->pid */
35        struct task_struct *p_opptr,*p_pptr, *p_cptr, *p_ysptr, *p_osptr;
```

```
36
37          struct wait_queue *wait_chldexit; /* for wait4() */
38          /* For ease of programming... Normal sleeps don't need to
39           * keep track of a wait-queue: every task has an entry of its own */
40
41          /* user/group IDs */
42          unsigned short uid, euid, suid;
43          unsigned short gid, egid, sgid;
44
45          unsigned long timeout;
46          unsigned long it_real_value, it_prof_value, it_virt_value;
47          unsigned long it_real_incr, it_prof_incr, it_virt_incr;
48          long utime,stime,cutime,cstime,start_time;
49          unsigned long min_flt, maj_flt;
50          unsigned long cmin_flt, cmaj_flt;
51          struct rlimit rlim[RLIM_NLIMITS];
52          unsigned short used_math;
53          unsigned short rss;  /* number of resident pages */
54          char comm[16];
55          struct vm86_struct * vm86_info; /* virtual memory info */
56          unsigned long screen_bitmap;
57
58  /* file system info */
59          int link_count;
60          int tty;            /* -1 if no tty, so it must be signed */
61          unsigned short umask;
62          struct inode * pwd;
63          struct inode * root;
64          struct inode * executable;
65          struct vm_area_struct * mmap;
66          struct shm_desc *shm;
67          struct sem_undo *semun;
68          struct file * filp[NR_OPEN];
69          fd_set close_on_exec;
70
71  /* ldt for this task - used by Wine. If NULL, default_ldt is used */
72          struct desc_struct *ldt;
73
74  /* tss for this task */
75          struct tss_struct tss;
76
77  #ifdef NEW_SWAP
78          unsigned long old_maj_flt;  /* old value of maj_flt */
79          unsigned long dec_flt;      /* page fault count of the last time */
80          unsigned long swap_cnt;     /* number of pages to swap on next pass */
81          short swap_table;           /* current page table */
82          short swap_page;            /* current page */
83  #endif NEW_SWAP
84
85          struct vm_area_struct *stk_vma;
86  };
```