程序 14-30 linux/include/sys/resource.h

```
/*
 * Resource control/accounting header file for linux
 */
/*
 * Linux 资源控制/审计头文件。
 */

#ifndef _SYS_RESOURCE_H
#define _SYS_RESOURCE_H

// 以下符号常数和结构用于 getrusage()。参见 kernel/sys.c 文件第 412 行开始。
/*
 * Definition of struct rusage taken from BSD 4.3 Reno
 *
 * We don't support all of these yet, but we might as well have them....
 * Otherwise, each time we add new items, programs which depend on this
 * structure will lose.  This reduces the chances of that happening.
 */
/*
 * rusage 结构的定义取自 BSD 4.3 Reno 系统。
 *
 * 我们现在还没有支持该结构中的所有这些字段，但我们可能会支持它们的....
 * 否则的话，每当我们增加新的字段，那些依赖于这个结构的程序就会出问题。
 * 现在把所有字段都包括进来就可以避免这种事情发生。
 */
// 下面是 getrusage()的参数 who 所使用的符号常数。
#define RUSAGE_SELF     0       // 返回当前进程的资源利用信息。
#define RUSAGE_CHILDREN -1      // 返回当前进程已终止和等待着的子进程的资源利用信息。

// rusage 是进程的资源利用统计结构，用于 getrusage()返回指定进程对资源利用的统计值。
// Linux 0.12 内核仅使用了前两个字段，它们都是 timeval 结构（include/sys/time.h）。
// ru_utime - 进程在用户态运行时间统计值；ru_stime - 进程在内核态运行时间统计值。
struct  rusage {
        struct timeval ru_utime;        /* user time used */
        struct timeval ru_stime;        /* system time used */
        long    ru_maxrss;              /* maximum resident set size */
        long    ru_ixrss;               /* integral shared memory size */
        long    ru_idrss;               /* integral unshared data size */
        long    ru_isrss;               /* integral unshared stack size */
        long    ru_minflt;              /* page reclaims */
        long    ru_majflt;              /* page faults */
        long    ru_nswap;               /* swaps */
        long    ru_inblock;             /* block input operations */
        long    ru_oublock;             /* block output operations */
        long    ru_msgsnd;              /* messages sent */
        long    ru_msgrcv;              /* messages received */
        long    ru_nsignals;            /* signals received */
        long    ru_nvcsw;               /* voluntary context switches */
        long    ru_nivcsw;              /* involuntary " */
};

// 下面是 getrlimit()和 setrlimit()使用的符号常数和结构。
/*
```

```
38  * Resource limits
39  */
    /*
     * 资源限制。
     */
    // 以下是 Linux 0.12 内核中所定义的资源种类，是 getrlimit()和 setrlimit()中第 1 个参数
    // resource 的取值范围。 其实这些符号常数就是进程任务结构中 rlim[] 数组的项索引值。
    // rlim[]数组的每一项都是一个 rlimit 结构，该结构见下面第 58 行。
40
41  #define RLIMIT_CPU      0           /* CPU time in ms */        /* 使用的 CPU 时间 */
42  #define RLIMIT_FSIZE    1           /* Maximum filesize */      /* 最大文件长度 */
43  #define RLIMIT_DATA     2           /* max data size */         /* 最大数据长度 */
44  #define RLIMIT_STACK    3           /* max stack size */        /* 最大栈长度 */
45  #define RLIMIT_CORE     4           /* max core file size */    /* 最大 core 文件长度 */
46  #define RLIMIT_RSS      5           /* max resident set size */ /* 最大驻留集大小 */
47
    // 如果定义了符号 notdef，则也包括以下符号常数定义。
48  #ifdef notdef
49  #define RLIMIT_MEMLOCK  6           /* max locked-in-memory address space*/ /* 锁定区 */
50  #define RLIMIT_NPROC    7           /* max number of processes */   /* 最大子进程数 */
51  #define RLIMIT_OFILE    8           /* max number of open files */  /* 最大打开文件数 */
52  #endif
53
    // 这个符号常数定义了 Linux 中限制的资源种类。RLIM_NLIMITS=6，因此仅前面 6 项有效。
54  #define RLIM_NLIMITS    6
55
    // 表示资源无限，或不能修改。
56  #define RLIM_INFINITY   0x7fffffff
57
    // 资源界限结构。
58  struct rlimit {
59          int     rlim_cur;           // 当前资源限制，或称软限制（soft limit）。
60          int     rlim_max;           // 硬限制（hard limit）。
61  };
62
63  #endif /* _SYS_RESOURCE_H */
64
```