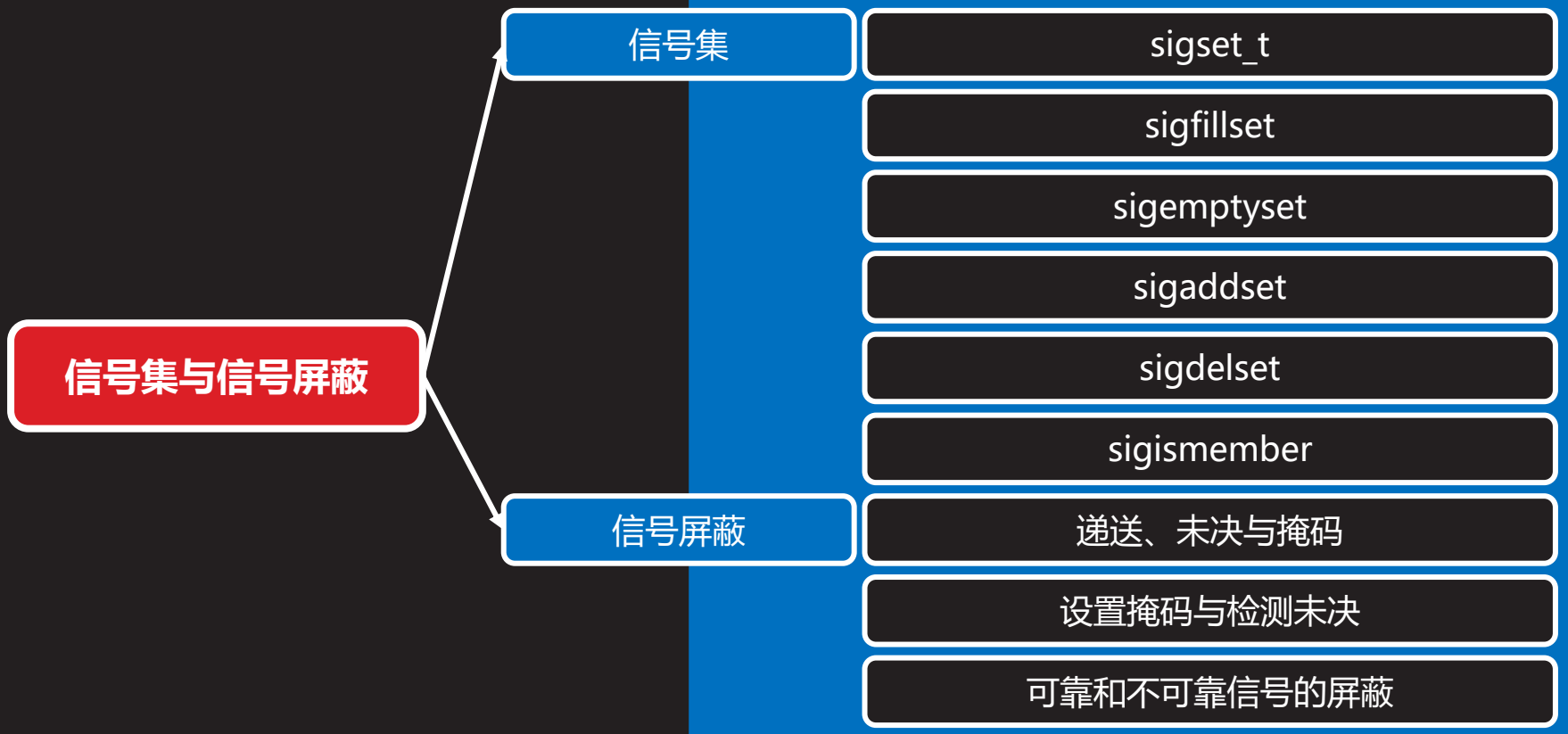
Technology Tarena 达内科技

Unix系统高级编程

信号屏蔽和定时器 Unit19

信号集与信号屏蔽


信号集与信号屏蔽
信号集
sigset_t
sigfillset
sigemptyset
sigaddset
sigdelset
sigismember
信号屏蔽
递送、未决与掩码
设置掩码与检测未决
可靠和不可靠信号的屏蔽

Technology Tarena 达内科技

信号集

sigset_t

- 多个信号组成的信号集合谓之信号集
- 系统内核用sigset_t类型表示信号集
- sigset_t类型是一个结构体，但该结构体中只有一个成员，是一个包含32个元素的整数数组
  - 在<sigset.h>中有如下类型定义

```
#define _SIGSET_NWORDS (1024 /
    (8 * sizeof (unsigned long int)))
typedef struct {
    unsigned long int __val[_SIGSET_NWORDS];
}   __sigset_t;
```

  - 在<signal.h>中又被定义为

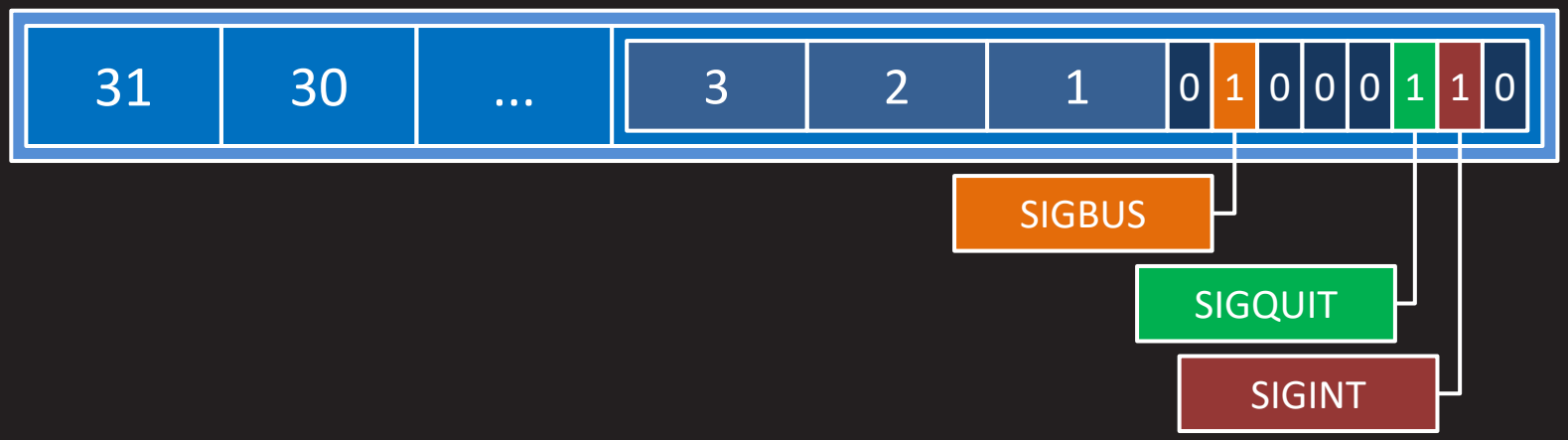
```
typedef __sigset_t sigset_t;
```

知识讲解

sigset_t（续1）

- 可以把sigset_t类型看成一个由1024个二进制位组成的大整数
  - 其中的每一位对应一个信号，其实目前远没有那么多信号
  - 某位为1就表示信号集中有此信号，反之为0就是无此信号
  - 当需要同时操作多个信号时，常以sigset_t作为函数的参数或返回值的类型

知识讲解



sigfillset

- 填满信号集，即将信号集的全部信号位置1

```
#include <signal.h>

int sigfillset (sigset_t* sigset);
```

成功返回0，失败返回-1

  - *sigset*：信号集
- 例如
  - sigset_t sigset;
    if (sigfillset (&sigset) == -1) {
        perror ("sigfillset");
        exit (EXIT_FAILURE);
    }

知识讲解

sigemptyset

- 清空信号集，即将信号集的全部信号位清0

```
#include <signal.h>

int sigemptyset (sigset_t* sigset);
```

成功返回0，失败返回-1

  - *sigset*：信号集

- 例如
  - sigset_t sigset;

```
if (sigemptyset (&sigset) == -1) {
    perror ("sigemptyset");
    exit (EXIT_FAILURE);
}
```

知识讲解

sigaddset

- 加入信号，即将信号集中与指定信号编号对应的信号位置1

```
#include <signal.h>

int sigaddset (sigset_t* sigset, int signum);
```

成功返回0，失败返回-1

  - *sigset*：信号集
  - *signum*：信号编号

- 例如
  -
```
if (sigaddset (&sigset, SIGINT) {
    perror ("sigaddset");
    exit (EXIT_FAILURE);
}
```

知识讲解

sigdelset

- 删除信号，即将信号集中与指定信号编号对应的信号位清0

```
#include <signal.h>

int sigdelset (sigset_t* sigset, int signum);
```

成功返回0，失败返回-1

  - *sigset*：信号集
  - *signum*：信号编号
- 例如
  - if (sigdelset (&sigset, SIGINT) {
    perror ("sigdelset");
    exit (EXIT_FAILURE);
    }

知识讲解

sigismember

- 判断信号集中是否有某信号，即检查信号集中与指定信号编号对应的信号位是否为1

```
#include <signal.h>

int sigismember (const sigset_t* sigset, int signum);
```

有则返回1，没有返回0，失败返回-1

  - *sigset*：信号集
  - *signum*：信号编号
- 例如
  - if (sigismember (&sigset, SIGINT) == 1)
    printf ("信号集中有SIGINT信号\n");

知识讲解

Technology Tarena 达内科技

信号集

【参见：sigset.c】

课堂练习

- 信号集

Technology Tarena 达内科技

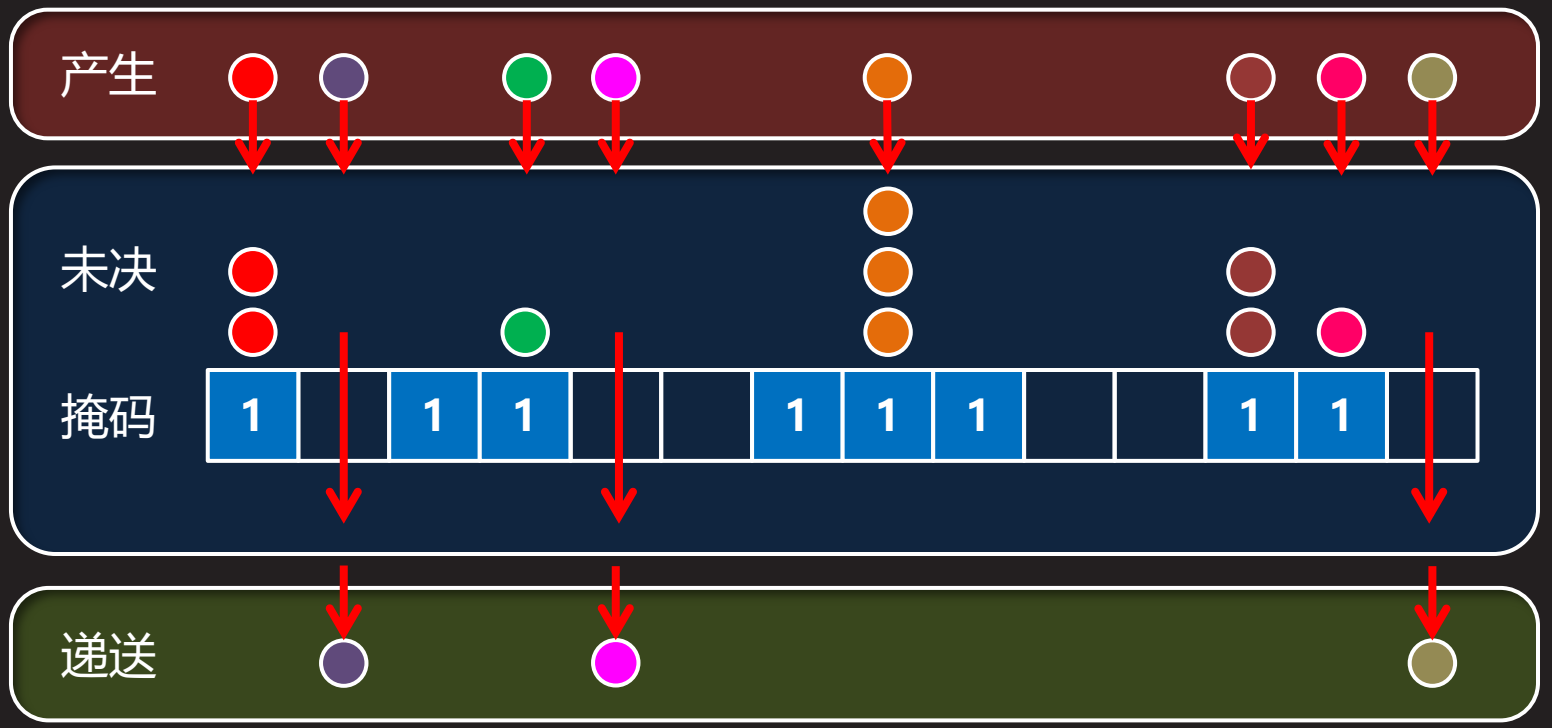
信号屏蔽

递送、未决与掩码

- 当信号产生时，系统内核会在其所维护的进程表中，为特定的进程设置一个与该信号相对应的标志位，这个过程就叫做递送(delivery)
- 信号从产生到完成递送之间存在一定的时间间隔，处于这段时间间隔中的信号状态称为未决(pending)
- 每个进程都有一个信号掩码(signal mask)，它实际上是一个信号集，位于该信号集中的信号一旦产生，并不会被递送给相应的进程，而是会被阻塞(block)在未决状态
- 在信号处理函数执行期间，这个正在被处理的信号总是处于信号掩码中，如果又有该信号产生，则会被阻塞，直到上一个针对该信号的处理过程结束以后才会被递送

递送、未决与掩码（续1）

- 当进程正在执行类似更新数据库这样的敏感任务时，可能不希望被某些信号中断。这时可以通过信号掩码暂时屏蔽而非忽略掉这些信号，使其一旦产生即被阻塞于未决状态，待特定任务完成后，再回过头来处理这些信号



设置掩码与检测未决

- 设置调用进程的信号掩码

```
#include <signal.h>

int sigprocmask (int how, const sigset_t* sigset,
    sigset_t* oldset);
```

成功返回0，失败返回-1

– *how*：修改信号掩码的方式，可取以下值

`SIG_BLOCK` - 将*sigset*中的信号加入当前信号掩码

`SIG_UNBLOCK` - 从当前信号掩码中删除*sigset*中的信号

`SIG_SETMASK` - 把*sigset*设置成当前信号掩码

设置掩码与检测未决（续1）

- 设置调用进程的信号掩码
  - ***sigset***：信号集，取NULL则忽略此参数
  - ***oldset***：输出原信号掩码，取NULL则忽略此参数
- 例如
  - sigset_t sigset;
    sigemptyset (&sigset);
    sigaddset (&sigset, SIGINT);
    sigaddset (&sigset, SIGQUIT);
    sigset_t oldset;
    if (sigprocmask (SIG_SETMASK, &sigset,
    &oldset) == -1) {
    perror ("sigprocmask");
    exit (EXIT_FAILURE); }

知识讲解

设置掩码与检测未决（续2）

- 获取调用进程的未决信号集

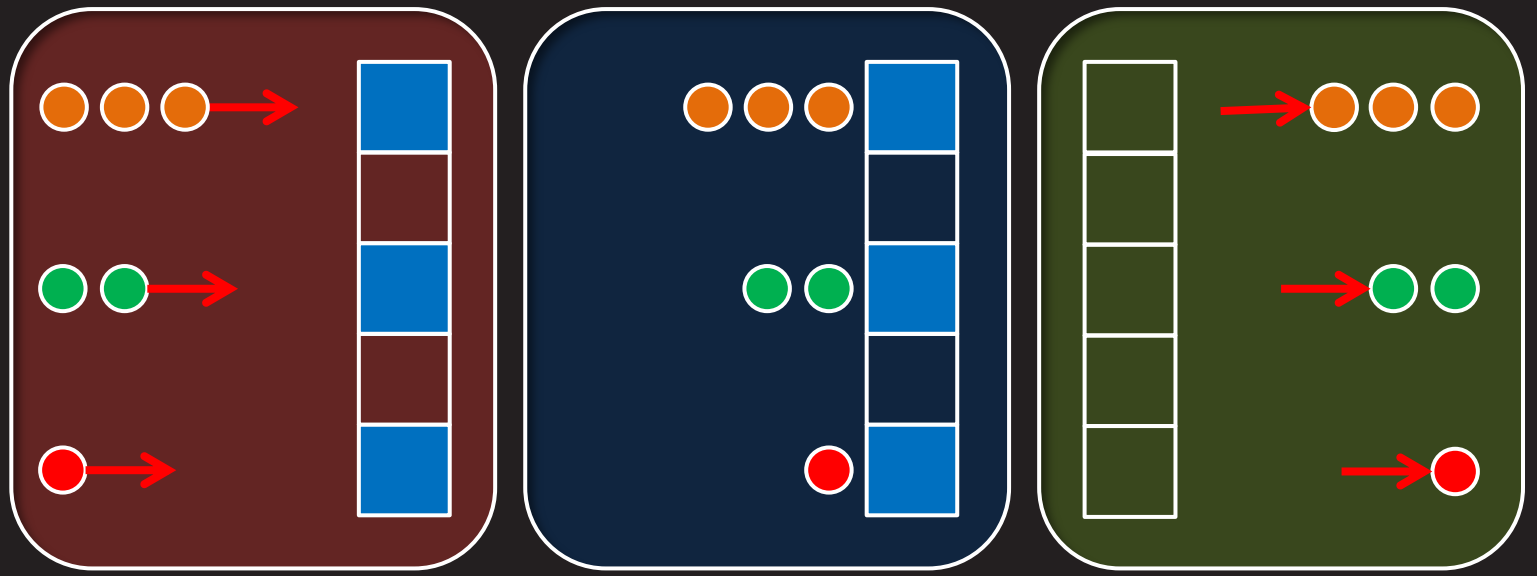
```
#include <signal.h>

int sigpending (sigset_t* sigset);

                                成功返回0，失败返回-1
```

  - sigset：输出未决信号集
- 例如
  - sigset_t sigset;
    if (sigpending (&sigset) == -1) {
        perror ("sigpending"); exit (EXIT_FAILURE); }
    if (sigismember (&sigset, SIGINT) == 1)
        printf ("SIGINT信号未决\n");

知识讲解

可靠和不可靠信号的屏蔽

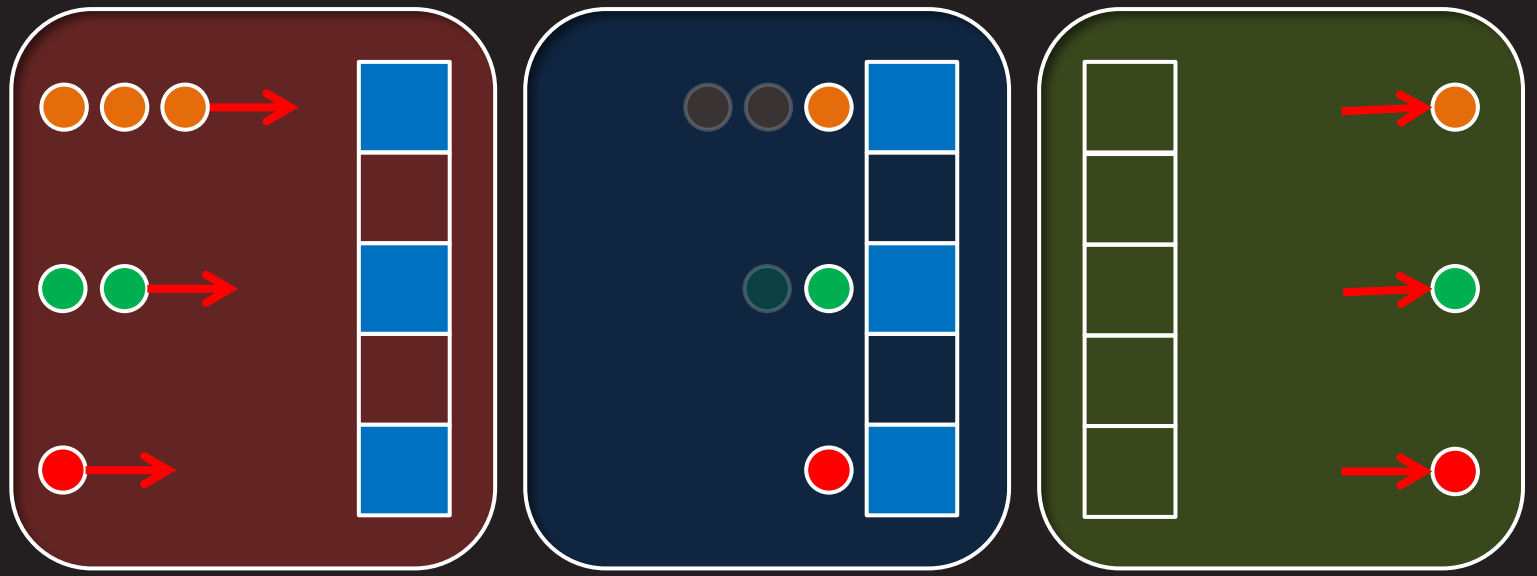
- 对于可靠信号，通过sigprocmask函数设置信号掩码以后，每种被屏蔽信号中的每个信号都会被阻塞，并按先后顺序排队，一旦解除屏蔽，这些信号会被依次递送



可靠和不可靠信号的屏蔽（续1）

- 对于不可靠信号，通过sigprocmask函数设置信号掩码以后，每种被屏蔽信号中只有第一个会被阻塞，并在解除屏蔽后被递送，其余的则全部丢失

知识讲解



信号屏蔽

【参见：sigmask.c】

课堂练习

- 信号屏蔽

定时器

Technology Tarena 达内科技

系统计时器

系统计时器

- 运行一个进程所消耗的时间包括三个部分
  - 用户时间：进程消耗在用户态的时间
  - 内核时间：进程消耗在内核态的时间
  - 睡眠时间： 进程消耗在等待I/O、睡眠等不被调度的时间

- 系统内核为系统中的每个进程维护三个计时器
  - 真实计时器：统计进程的执行时间
  - 虚拟计时器：统计进程的用户时间
  - 实用计时器：统计进程的用户时间和内核时间之和

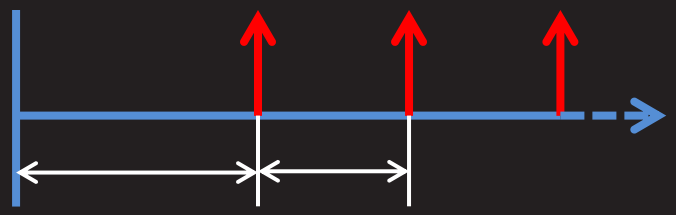
知识讲解

Technology Tarena 达内科技

设置定时器

设置定时器

- 三个系统计时器除了统计进程的各种时间以外，还可以按照各自的计时规则，以定时器的方式工作，向进程周期性地发送不同的信号
  - SIGALRM (14)：真实定时器到期
  - SIGVTALRM (26)：虚拟定时器到期
  - SIGPROF (27)：实用定时器到期
- 定时器在可以发送信号之前，必须先行设置。每个定时器均包括两个属性，需要在设置时初始化好
  - 初始间隔：从设置定时器到它首次发出信号的时间间隔
  - 重复间隔：定时器发出的两个相邻信号之间的时间间隔



设置定时器（续1）

- 设置、启动、关闭定时器

```
#include <sys/time.h>

int setitimer (int which,
    const struct itimerval* new_value,
    struct itimerval* old_value);
```

成功返回0，失败返回-1

– *which*：指定哪个定时器，可取以下值

`ITIMER_REAL` - 真实定时器

`ITIMER_VIRTUAL` - 虚拟定时器

`ITIMER_PROF` - 实用定时器

– *new_value*：新设置值

知识讲解

设置定时器（续2）

- 设置、启动、关闭定时器
  - ***old_value***：输出原设置值，可置NULL
- 该函数有关定时器设置的参数都选用了itimerval类型
  - struct itimerval {

```
// 重复间隔，取0将使定时器在发送第一个信号后停止
struct timeval it_interval;
// 初始间隔，取0将立即停止定时器
struct timeval it_value;
};
```

  - struct timeval {

```
long tv_sec;  // 秒数
long tv_usec; // 微秒数
};
```

知识讲解

设置定时器（续3）

- 例如
  - 5秒以后开始计时，每3毫秒计时一次

```
struct itimerval it;
it.it_value.tv_sec = 5;
it.it_value.tv_usec = 0;
it.it_interval.tv_sec = 0;
it.it_interval. tv_usec = 3000;
if (setitimer (ITIMER_REAL, &it, NULL) == -1) {
    perror ("setitimer");
    exit (EXIT_FAILURE); }
```

  - 关闭定时器

```
it.it_value.tv_sec = 0;
setitimer (ITIMER_REAL, &it, NULL);
```


电子秒表

【参见：timer.c】

课堂练习

- 电子秒表

获取定时器设置

- 获取定时器设置

```
#include <sys/time.h>

int getitimer (int which, struct itimerval* curr_value);
```

成功返回0，失败返回-1

  - *which*：指定哪个定时器，可取以下值

    `ITIMER_REAL` - 真实定时器

    `ITIMER_VIRTUAL` - 虚拟定时器

    `ITIMER_PROF` - 实用定时器

  - *curr_value*：输出当前设置值

知识讲解

Technology Tarena 达内科技

总结和答疑