Technology Tarena 达内科技

Unix系统高级编程

信号的发送与定时 Unit18

发送信号

键盘、错误与命令

键盘、错误与命令

- 由键盘触发的信号
  - SIGINT (2)：Ctrl+C，中断符
  - SIGQUIT (3)：Ctrl+\，退出符
  - SIGTSTP (20)：Ctrl+Z，停止符

知识讲解

键盘、错误与命令（续1）

- 由错误和异常引发的信号
  - SIGILL (4)：进程试图执行非法指令
  - SIGBUS (7)：硬件或对齐错误
  - SIGFPE (8)：算术异常
  - SIGSEGV (11)：无效内存访问
  - SIGPIPE (13)：向无读取进程的管道写入
  - SIGSTKFLT (16)：协处理器栈错误
  - SIGXFSZ (25)：文件资源超限
  - SIGPWR (30)：断电
  - SIGSYS (31)：进程试图执行无效系统调用

键盘、错误与命令（续2）

Technology Tarena 达内科技

- 用专门的系统命令发送信号

$ kill [-信号] PIDs

  - 若不指明具体信号，缺省发送SIGTERM(15)信号
    - 该信号允许用户优雅地终止进程。进程可以选择捕获该信号，并在临终之前完成必要的清理和善后工作。但如果捕获了该信号，却死赖着不走，则有流氓进程之嫌
  - 若要指明具体信号，可以使用信号编号，也可以使用信号名称，而且信号名称中的“SIG”前缀可以省略不写。例如
    - kill -9 1234
    - kill -SIGKILL 1234 5678
    - kill -KILL -1
  - 接收信号的进程可以是一个、多个或所有的(PIDs取-1)
  - 超级用户可以发给任何进程，而普通用户只能发给自己的进程

知识讲解

Technology Tarena 达内科技

调用函数发送信号

kill

- 向指定进程(组)发送信号

```
#include <signal.h>

int kill (pid_t pid, int signum);

                成功(至少发出去一个信号)返回0，失败返回-1
```

  - *pid*：可取以下值
    - <-1 - 向特定进程组(由-*pid*标识)的所有进程发送信号
    - -1 - 向系统中的所有进程发送信号
    - 0 - 向与调用进程同进程组的所有进程发送信号
    - > 0 - 向特定进程(由*pid*标识)发送信号
  - *signum*：信号编号，取0可用于检查*pid*进程是否存在，如不存在kill函数会返回-1，且errno为ESRCH

杀死进程

【参见：kill.c】

课堂练习

- 杀死进程

raise

- 向调用进程自己发送信号

```
#include <signal.h>

int raise (int signum);
```

成功返回0，失败返回非0

  - *signum*：信号编号
- raise函数实际上是给调用进程或者线程发送信号
  - 对于单线程应用来说，它相当于
    kill (getpid (), signum);
  - 对于多线程应用来说，它相当于
    pthread_kill (pthread_self (), signum);
- 若所发信号被捕获，raise函数会在信号处理函数返回后返回

知识讲解

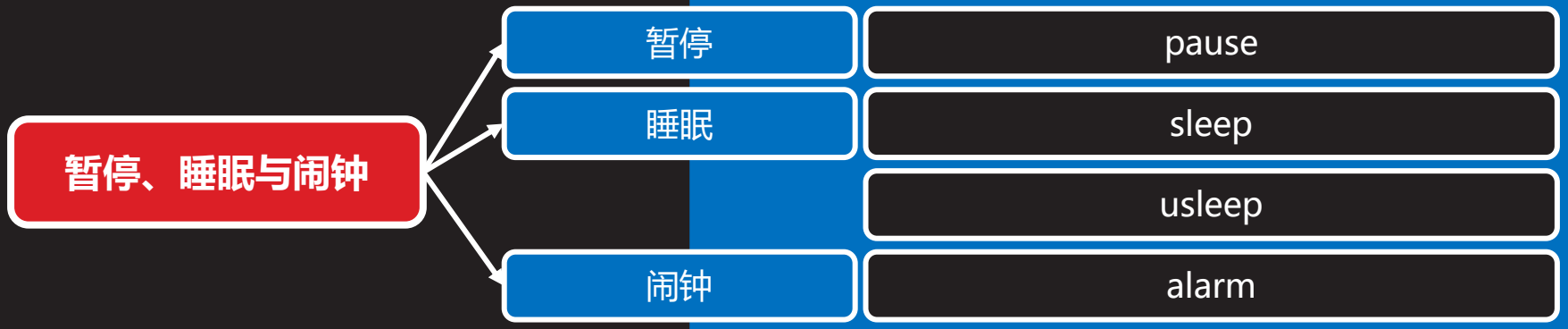
进程自杀

【参见：raise.c】

课堂练习

- 进程自杀

暂停、睡眠与闹钟



Technology Tarena 达内科技

暂停

pause

- 无限睡眠

```
#include <unistd.h>

int pause (void);
```

成功阻塞，失败返回-1

知识讲解

- 该函数使调用进(线)程进入无时限的睡眠状态，直到有信号终止了调用进程或被其捕获
- 如果有信号被调用进程捕获，在信号处理函数返回以后，pause函数才会返回，且返回值为-1，同时置errno为EINTR，表示阻塞的系统调用被信号中断
- pause函数要么不返回，要么返回失败，不会返回成功

暂停

【参见：pause.c】

课堂练习

- 暂停

Technology Tarena 达内科技

睡眠

sleep

- 有限睡眠

```
#include <unistd.h>

unsigned int sleep (unsigned int seconds);

                                    返回0或剩余秒数
```

  - ***seconds***：以秒为单位的睡眠时限
- 该函数使调用进(线)程睡眠***seconds***秒，除非有信号终止了调用进程或被其捕获
- 如果有信号被调用进程捕获，在信号处理函数返回以后，sleep函数才会返回，且返回值为剩余的秒数，否则该函数将返回0，表示睡眠充足

睡眠

【参见：sleep.c】

课堂练习

- 睡眠

usleep

- 更精确的有限睡眠

```
#include <unistd.h>

int usleep (useconds_t usec);

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

  - ***usec***：以微秒(1微秒=10^{-6}秒)为单位的睡眠时限，必须小于1000000(即1秒)
- 该函数使调用进(线)程睡眠***usec***微秒，除非有信号终止了调用进程或被其捕获
- 如果有信号被调用进程捕获，在信号处理函数返回以后，usleep函数才会返回，且返回值为-1，同时置errno为EINTR，表示阻塞的系统调用被信号中断

知识讲解

Technology Tarena 达内科技

闹钟

alarm

- 设置闹钟

```
#include <unistd.h>

unsigned int alarm (unsigned int seconds);

                                返回0或先前所设闹钟的剩余秒数
```

  - ***seconds***：以秒为单位的闹钟时间
- alarm函数使系统内核在该函数被调用以后***seconds***秒的时候，向调用进程发送SIGALRM(14)信号
- 若在调用该函数前已设过闹钟且尚未到期，则该函数会重设闹钟，并返回先前所设闹钟的剩余秒数，否则返回0
- 若***seconds***取0，则表示取消先前设过且尚未到期的闹钟

知识讲解

闹钟

【参见：alarm.c】

课堂练习

- 闹钟

alarm（续1）

- 例如
  - void sigalrm (int signum) {
    time_t t = time (NULL);
    struct tm* lt = localtime (&t);
    printf ("\r%02d:%02d:%02d",
    lt->tm_hour, lt->tm_min, lt->tm_sec);
    alarm (1); }
  - if (signal (SIGALRM, sigalrm) == SIG_ERR) {
    perror ("signal"); return -1; }
    sigalrm (SIGALRM);
- 注意，通过alarm函数所设置的闹钟只是一次性的，若要获得周期性的效果，可在SIGALRM信号处理函数中再次设定

电子时钟

【参见：clock.c】

课堂练习

- 电子时钟

Technology Tarena 达内科技

总结和答疑