信号初步认识

Selection_311

1. 特点: ○ 简单 ○ 携带的信息量少 ○ 使用在某个特定的场景中
2. 信号的状态 ○ 产生 ○ 未决状态 - 没有被处理 ○ 递达 - 信号被处理了
3. 处理方式

   
4. 信号的四要素

   
5. 通过man文档查看信号 a. man 7 siganl b. The signals SIGKILL and SIGSTOP cannot be caught, blocked, or ignored.
6. 概念: 阻塞信号集, 未决信号集 ○ pcb ○ 不能直接操作 ○ 阻塞信号集: 👇 要屏蔽的信号 ○ 未决信号集: 👇 没有被处理的信号的集合

2 - 信号相关函数

1. kill -- 发送信号给指定进程 ○ 函数原型: int kill(pid_t pid, int sig);
   - pid>0 将信号发给指定进程
   - =0 发送给目前进程相同进程组的所有进程

```

```

include

include

include

// // Created by bruce on 18-5-19. // int main() {

```
    pid_t pid = fork();
    if(pid >0)
    {
        while(1)

        {
```

```
            printf("parent process,%d\n",getpid());
            sleep(1);
        }
    }
    else if(pid ==0)
    {
        kill(getppid(),SIGKILL);
        sleep(2);
    }
    return 0;
```

}

-

2. raise -- 自己给自己发信号
   a. kill(getpid(), SIGINT);
   ○ 函数原型: int raise(int sig);
   👇 返回值:

   ```c
   //
   // Created by bruce on 18-5-19.
   //

   #include <unistd.h>
   #include <cstdio>
   #include <signal.h>
   #include <sys/wait.h>
   #include <cstdlib>

   int main ()
   {
       pid_t pid = fork();
       if(pid >0)
       {
           int s;
           pid_t wpid = wait(&s);
           printf("child died pid = %d\n,wpid");
           if(WIFSIGNALED(s)){
               printf("died by signal:%d\n",WTERMSIG(s));
           }
       }
       else if(pid ==0)
       {
   //        raise(SIGINT);
           while(1)
           {
               abort();
           }
       }
       return 0;
```

```
}
```

1. abort -- 给自己发送异常终止信号 ○ 函数原型: void abort(void);

   没有参数没有返回值, 永远不会调用失败

   终止信号6

2. 

   例子:

```
//
// Created by bruce on 18-5-19.
//
```

include

include

include

include

include

int main() {

```
    int ret =alarm(5);
    printf("ret = %d\n",ret);

    sleep(2);
    ret = alarm(6); //  闹钟重置了
    printf("ret = %d\n",ret);

    while(1)
    {
        printf("hello,world\n");
        sleep(2);
    }
    return 0;
```

}

```

```

```c
//
// Created by bruce on 18-5-19.
//
#include <unistd.h>
#include <cstdio>
#include <signal.h>
#include <sys/wait.h>
#include <cstdlib>
int main()
{
    int num=0;
    alarm(1);
    while(1)
        printf("%d\n",num++);
    return 0;
}
```

- setitimer -- 定时器, 并实现周期性定时

函数原型: int setitimer(int which, const struct itimerval *new_value, struct itimerval *old_value // NULL); 分区 07 - 信号 的第 3 页); struct itimerval { struct timeval it_interval; // 定时器循环周期 struct timeval it_value; // 第一次触发定时器的时间 }; struct timeval { time_t tv_sec; /* seconds */ suseconds_t tv_usec; /* microseconds */ };

REAL	绝对时间	**SIGALRM**
VIRTUAL	设定执行时间,只有在用户模式下才可跟踪	SIGVTALRM
PROF	从用户进程开始及时	SIGPROF

3 - 信号集 信号集操作相关函数

1. 概念:
    - 未决信号集:
    - 没有被当前进程处理的信号
    - 阻塞信号集:
    - 将某个信号放到阻塞信号集,这个信号就不会被进程处理
    - 阻塞解除之后,信号被处理
2. 自定义信号集
    - int sigemptyset(sigset_t *set); 将set集合置空
    - int sigfillset(sigset_t *set); 将所有信号加入set集合
    - int sigaddset(sigset_t *set,int signo);
    - 将signo信号加入到set集合
    - int sigdelset(sigset_t *set,int signo);
    - 从set集合中移除signo信号
    - int sigismember(const sigset_t *set,int signo);
    - 判断信号是否存在
3. sigprocmask函数

- 屏蔽and接触信号屏蔽, 将自定义信号集设置给阻塞信号集
- 函数原型:

int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);

4. sigpending -- 读取当前进程的未决信号集
    - 函数原型: int sigpending(sigset_t *set);
    - 参数: set -- 内核将未决信号集写入set

练习:

- 编写程序,设置阻塞信号集并把所有常规信号的未决状态打印至屏幕。