4.4 进程间通讯 - 信号 (二)_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.4 进程间通讯 – 信号（二）涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- 信号是由操作系统内核发送给指定进程, 进程收到信号后则需要进行处理
- 处理信号三种方式:
    - 忽略：不进行处理
    - 默认：按照信号的默认方式处理
    - 用户自定义：通过用户实现自定义处理函数来处理，由内核来进行调用
- 对于每种信号都有相应的默认处理方式
    - 进程退出:
    - SIGALRM,SIGHUP,SIGINT,SIGKILL,SIGPIPE,SIGPOLL,SIGPROF,SIGSYS,SIGTERM,SIGUSR1,SIGUSR2,SIGVTALRM
    - 进程忽略
    - SIGCHLD,SIGPWR,SIGURG,SIGWINCH
    - 进程暂停
    - SIGSTOP,SIGTSTP,SIGTTIN,SIGTTOU
- 用户自定义处理基本的流程
    - step 1 : 实现自定义处理函数
        - 用户实现自定义处理函数, 需要按照下面的形式定义
        - typedef void (*sighandler_t)(int);
        - typedef void (*)(int) sighandler_t
        -
    - step 2 : 设置信号处理处理方式
        - 通过 signal 函数设置信号处理方式
        - 函数头文件 #include <signal.h>
        - 函数原型 sighandler_t signal(int signum, sighandler_t handler);
        -
        - 函数功能 设置信号的处理方式, 如果是自定义处理方式，提供函数地址，注册到内核中
        - 函数参数 signum : 信号编号
        - handler : 信号处理方式
        - SIG_IGN : 忽略信号
        - SIG_DFL : 按照默认方式处理
        - 自定义处理函数的地址
        -
        -

```
typedef void __signalfn_t(int);
typedef __signalfn_t *__sighandler_t;

typedef void __restorefn_t(void);
typedef __restorefn_t *__sigrestore_t;

#define SIG_DFL ((__sighandler_t)0)
#define SIG_IGN ((__sighandler_t)1)
#define SIG_ERR ((__sighandler_t)-1)
```

- 函数返回值
- - 成功：返回信号处理函数地址
- - 失败：返回 SIG_ERR，并设置 errno
-
- - 要点
    - - 三种信号处理方式互斥，一般选择一种即可
-
- 示例
- 创建一个子进程，父进程给子进程发送 SIGUSR1 信号，并使用自定义的处理方式

- -

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>
#include <sys/wait.h>

void do_sig_usr(int sig)
{
    printf(" Receive %s \n",strsignal(sig));
}

int main(void)
{
    pid_t cpid;


    if(signal(SIGUSR1,do_sig_usr) == SIG_ERR){
        perror("[ERROR] signal(): ");
        exit(EXIT_FAILURE);
    }

    cpid = fork();

    if(cpid == -1){
        perror("fork(): ");
        exit(EXIT_FAILURE);
    }else if(cpid == 0){
        printf("Child Process < %d > start.\n",getpid());

        pause();

        exit(EXIT_SUCCESS);
    }else if(cpid > 0){

        sleep(1);

        kill(cpid,SIGUSR1);

        wait(NULL);
    }

    return 0;
}
```

练习

创建两个子进程 A 与 B，给子进程 A 发送 SIGUSR1 信号，子进程 B 发送 SIGUSR2 信号，子进程 A 的处理方式设置为默认，子进程 B 的处理方式为使用自定义处理函数，并打印接收的信号的字符串信息(使用 strsignal() 函数)