3.4 进程的等待_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.4 进程的等待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。

- 在子进程运行结束后，进入僵死状态, 并释放资源, 子进程在内核中的 数据结构 依然保留
- 父进程 调用 wait() 与 waitpid() 函数等待子进程退出后，释放子进程遗留的资源（task_struct）

函数头文件

#include <sys/types.h>

#include <sys/wait.h>

函数原型

pid_t wait(int *wstatus);

函数功能

让函数调用者进程进入到睡眠状态, 等待子进程进入僵死状态后，释放相关资源并返回

函数参数

- wstatus : 保存子进程退出状态值变量的指针
    - 获取具体值需要使用 WEXITSTATUS() 宏定义

函数返回值

- 成功 : 返回退出子进程的 pid
- 失败： 返回 –1
-

说明 :

- 会阻塞调用者进程（一般为父进程）
- 在子进程状态为僵死态时，回收资源，并释放资源后返回

示例:

创建一个子进程, 延时 3s 后退出, 父进程等待子进程退出

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

int main(void)
```

```
{
    pid_t cpid;

    cpid = fork();

    if (cpid == -1){
        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);
    }else if(cpid == 0){
        printf("The Child process < %d > running...\n",getpid());
        sleep(3);
        exit(88);
    }else if(cpid > 0){

        int rpid,status = 0;

        rpid = wait(&status);
        if (rpid == -1){
            perror("[ERROR] wait() : ");
            exit(EXIT_FAILURE);
        }

        printf("The Child Process < %d > has exited,exit code < %d >.\n",rpid,WEXITSTATUS(status))
    }

    return 0;
}
~
```

注意

- 在 wait 存储在 satus 变量的值, 存储了很多信息, 通过一系列 W 开头的宏来解析获取
- WIFEXITED(status) : 进程是否正常结束
- WEXITSTATUS(wstatus) : 获取进程退出状态值, exit 函数的参数
- WIFSIGNALED(wstatus) : 表示该子进程是否被信号结束的, 返回真，则表示被信号结束的
- WTERMSIG(wstatus) : 返回结束该子进程的那个信号的信号值
- WCOREDUMP(wstatus) : 表示该子进程被信号唤醒的
- WIFSTOPPED(wstatus) : 表示该子进程是否被信号中止 (stop) 的 , 返回真，则表示是被信号中止的
- waitpid 函数的功能与 wait 函数一样，但比 wait() 函数功能更强大, 可以理解成 wait() 底层调用 waitpid() 函数

函数头文件

#include <sys/types.h>

#include <sys/wait.h>

函数原型

pid_t waitpid(pid_t pid, int *wstatus, int options);

函数参数

- pid : 进程 id
    - –1 : 可以等待任意子进程
    - > 0 : 等待 id 为 pid 的进程
- wstatus : 保存子进程退出状态值变量的指针
- options : 选项
    - WNOHANG : 非阻塞选项

函数返回值

- 成功 :
    - > 0 : 退出进程的 pid
    - = 0 : 在非阻塞模式下，没有进程退出
- 失败:
    - –1 并设置 errno
- 示例 : 创建一个子进程, 子进程运行后 3s 退出, 父进程等待子进程退出

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

int main(void)
{
    pid_t cpid;

    cpid = fork();

    if (cpid == -1){
        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);
    }else if(cpid == 0){
        printf("The Child process < %d > running...\n",getpid());
        sleep(3);
        exit(88);
    }else if(cpid > 0){
```

```
        int rpid,status = 0;


        rpid = waitpid(-1,&status,0);
        if (rpid == -1){
            perror("[ERROR] wait() : ");
            exit(EXIT_FAILURE);
        }
#if 0
        while((rpid = waitpid(-1,&status,WNOHANG)) == 0){


        }
#endif
        printf("The Child Process < %d > has exited,exit code < %d >.\n",rpid,WEXITSTATUS(status))
    }



    return 0;
}
```

- 总结
    - waitpid 函数常见用法如下:
        - 使用阻塞的方式等待任意子进程退出

```
  waitpid(-1,&status,0);
```

```
while(waitpid(pid,&status,WNOHANG) == 0)
    {
        usleep(50000);
    }
```

练习:

创建两个子进程，子进程 A 与 子进程 B ,A 进程延时 2s 后退出, B 进程延时 5s 后退出，父进程分别等待两个子进程退出