3.3 进程的退出_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.3 进程的退出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。

3. 进程的退出

- 在进程结束时，需要释放进程地址空间 以及内核中产生的各种数据结构
- 资源的释放需要通过调用 exit 函数或者 _exit 函数来完成
- 在程序结束时，会自动调用 exit 函数
- exit 函数让当前进程退出, 并刷新缓冲区
- exit 函数信息如下:

函数头文件

#include <stdlib.h>

函数原型

void exit(int status);

函数功能

结束进程，并刷新缓冲区

函数参数

status : 退出状态值

- 在系统中定义了两个状态值 : EXIT_SUCCESS : 正常退出 EXIT_FAILURE : 异常退出, 具体定义在 stdlib.h 中

```
#define EXIT_FAILURE    1
#define EXIT_SUCCESS    0
```

示例: 创建一个子进程，让子进程延时 3 s 后退出

```
int main(void)
{
    pid_t cpid;

    cpid = fork();

    if (cpid == -1){
        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);
    }else if(cpid == 0){
```

```
        printf("Child Process < %d > running...\n",getpid());
        sleep(3);
        printf("Child Process < %d >  has exited.\n",getpid());
        exit(EXIT_SUCCESS);
    }else if(cpid > 0){
        sleep(5);
    }

    return 0;
}
```

函数头文件

#include <unistd.h>

函数原型

void _exit(int status);

函数参数

status：进程退出的状态值

```
int main(void)
{
    pid_t cpid;

    cpid = fork();

    if (cpid == -1){
        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);
    }else if(cpid == 0){
        printf("Child Process < %d > running...\n",getpid());
        sleep(3);
        printf("Child Process < %d >  has exited.\n",getpid());
        _exit(EXIT_SUCCESS);
    }else if(cpid > 0){
        sleep(5);
    }

    return 0;
}
```

- exit 函数与 _exit 函数功能相似，但有很多不同，具体如下：
    - _exit() 属于**系统调用**，能够使进程停止运行，并释放空间以及销毁内核中的各种数据结构
    - exit() 基于_exit() 函数实现，属于库函数，**可以清理 I/O 缓冲区**

- 示例: 验证 exit 函数刷新缓冲区

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




int main(void)
{
    pid_t cpid;

    cpid = fork();

    if (cpid == -1){
        perror("[ERROR] fork(): ");
        exit(EXIT_FAILURE);
    }else if(cpid == 0){
        printf("I/O BUFFER.");
        sleep(3);

        exit(EXIT_SUCCESS);
    }else if(cpid > 0){
        sleep(5);
    }

    return 0;
}
```