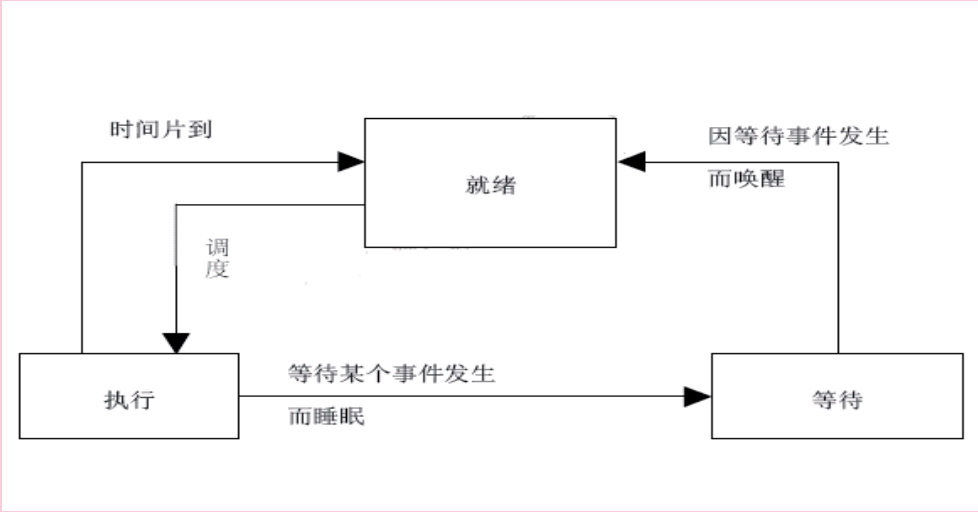
第10章　嵌入式Linux多任务编程简介

10.1 进程的基本概念

通常一个任务是一个程序的一次运行，一个任务包含一个或多个完成独立功能的子任务，这个独立的子任务就是进程或线程。对系统而言，当用户在系统中键入命令执行一个程序的时候，它将启动一个进程。

一、进程的状态



二、进程的标示符

在Linux 中最主要的进程标识有进程号（PID，Process Idenity Number）和它的父进程号（PPID，parent process ID）。其中PID 惟一地标识一个进程。PID 和PPID 都是非零的正整数。

三、进程的创建、执行和终止

1、创建

在进程中通过fork()函数通过复制当前进程创建子进程，子进程和父进程只有进程号不同，在发生写入数据时复制资源。

2、执行

由exec函数族负责读取可执行文件将其载入地址空间运行。

3、终止

释放资源、进入僵尸状态，通知父进程，由父进程终止。

四、进程的内存结构

Linux操作系统采用虚拟内存技术，每个进程都有各自互不干涉的内存空间，每个进程具有4GB的虚拟内存空间。

虚拟内存空间分为用户空间(0~3GB),内核空间（3~4GB).

10.2 进程控制编程

一、创建新进程

通过调用**fork()**函数在父进程中创建子进程，子进程继承父进程的地址空间，复制了父进程中的代码段、数据段和堆栈段的大部分内容。所独有的只是进程号、资源使用和计数器等。两个进程同时运行，都获得**fork()**的返回值。

1、fork()函数

所需头文件	#include <sys/types.h> #include <unistd.h>
函数原型	pid_t fork(void)
函数返回值	0：子进程
	子进程ID（大于0的整数）：父进程
	-1：出错

```
/*fork.c*/
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
pid_t result;
/*调用fork函数，其返回值为result*/
result = fork();
/*通过result的值来判断fork函数的返回情况，首先进行出错处理*/
if(result == -1){
perror("fork");
exit;
}
```

```
/*返回值为0代表子进程*/
else if(result == 0){
printf("The return value is %d\nIn child process!!\nMy PID
is%d\n",result,getpid());
}
/*返回值大于0代表父进程*/
else
{
printf("The return value is %d\nIn father process!!\nMy PID is
%d\n",result,getpid());
}
}
```

现在大部分嵌入式Linux系统采用vfork()函数创建子进程，vfork()创建的子进程与父进程可以访问相同的物理内存，只有子进程需要改变内存数据是才复制父进程。

2、exec函数族

exec 函数族就提供了一个在进程中启动另一个程序执行的方法。它可以根据指定的文件名或[illegible]名找到可执行文件，并用它来取代原调用进程的数据段、代码段和堆栈段，在执行完之后，原调用进程的内容除了进程号外，其他全部被新的进程替换了。另外，这里的可执行文件既可以是二进制文件，也可以是Linux 下任何可执行的脚本文件。