C 语言中的多线程编程

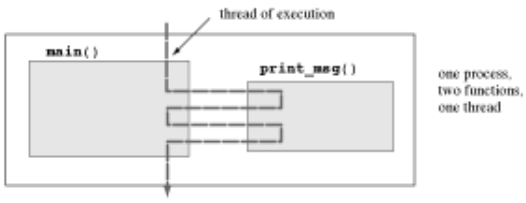
Linux　C 语言环境下的多线程编程
下面先看一个简单的单线程程序：

```
/*
    Sghello.c
    Hello,world -- Single Thread
*/
#include<stdio.h>
#define NUM 6
int main()
{
      void print_msg(char*);
      print_msg("hello,");
      print_msg("world!");
}
void print_msg(char* m)
{
      int i;
      for(i=0;i<NUM;i++)
      {
            printf("%s",m);
            fflush(stdout);
            sleep(1);
      }
}
```

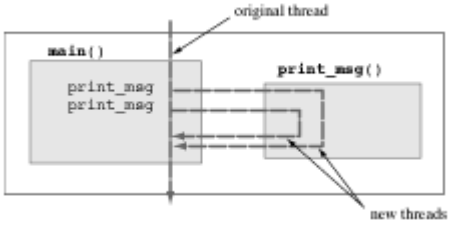
下图反映了程序的执行流程：

执行结果：

```
$ ./sghello.exe
hello,hello,hello,hello,hello,hello,world!world!world!world!world!world!
```



那么如果想同时执行两个对于 print_msg 函数的调用，就像使用 fork 建立两个新的进程一样，那该怎么办？这种思想清楚的体现在下图：



那么怎么才能达到这种效果呢？

我们可以使用函数 pthread_create 创建一个新的线程。

函数原型：

```
int pthread_create(pthread_t              *thread,
pthread_attr_t *attr,
void                  *(*func)(void*),
void                  *arg);
```

参数：thread 指向 pthread_t 类型变量的指针

attr 指向 pthread_attr_t 类型变量的指针，或者为 NULL

func 指向新线程所运行函数的指针

arg 传递给 func 的参数

返回值 0 成功返回

errcode 错误

我们可以使用函数 pthread_join 等待某进程结束。

函数原型：int pthread_join(pthread_t thread,void ** retval);

参数：thread 所等待的进程

retval 指向某存储线程返回值的变量

返回值： 0 成功返回

errorcode 错误

以上两个函数都包含在头文件 pthread.h 中。

下面请看多线程版的 Hello,world!

```
/*
    Mhello1.c
    Hello,world -- Multile Thread
*/
#include<stdio.h>
#include<pthread.h>
#define NUM 6
int main()
{
      void print_msg(void*);

      pthread_t t1,t2;

      pthread_create(&t1,NULL,(void*)print_msg,(void *)"hello,");

      pthread_create(&t2,NULL, (void*)print_msg,(void *)"world!\n");
```

```
        pthread_join(t1,NULL);
        pthread_join(t2,NULL);
}
void print_msg(void* m)
{
        char *cp=(char*)m;
        int i;
        for(i=0;i<NUM;i++)
        {
                printf("%s",m);
                fflush(stdout);
                sleep(1);
        }
}
```

运行结果：

```
$ gcc mhello1.c -o mhello1.exe -lpthread
$ ./mhello1.exe
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
hello,world!
```

C 语言又一次拓展了我的视野，多线程的问题还有很多，像线程间的分工合作、使用互斥机制保证线程间数据的安全共享、使用条件变量同步线程间的数据传输、传递多个参数给线程等，若读者有兴趣，可自行深入。

推荐《Understanding Unix/Linux Programming》