实验一：进程

例 1：这个例子示例了当前父进程 fork 了多个子进程，然后父子进程在屏幕上输出 pid、ppid 信息。

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

int main(void)
{
    int i;
    int n = 4;
    pid_t childpid;
    for (i=1; i<n; i++)
        if ((childpid = fork()) == 0)    //  子进程
            break;
    if (childpid > 0)
        sleep(1);                        //让父进程阻塞 1 秒
    fprintf(stderr, "This is process %ld with parent %ld\n",
        (long)getpid(), (long)getppid());
    return 0;
}
```

例 2：这个例子用到了 fork、execl、wait、exit 系统调用。

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

int main(int argc, char *argv[])
{
    pid_t childpid;
    int status;
    if ((childpid = fork()) == -1) {        //fork 出错
        perror("The fork failed");
        exit(-1);
    }
    else if (childpid == 0) {               //子进程
```

```
            if (execl("./hello","hello",NULL) < 0) {    //子进程执行另一个程序 hello
                perror("The exec of command failed");
                exit(-1);
            }
        }
        else if(childpid>0) {                   //父进程
                pid_t    apid = wait(&status);    //父进程调用 wait 等待子进程
                printf("parent process has wait the %ld child process exit\n",apid);
                exit(0);
        }
    }
```

上机练习：

Ex－1：运行例 1，观察结果，分析结果的成因。

Ex－2：考虑在例 1 中，sleep 函数的作用？如果将 sleep 函数去掉，程序运行结果是什么，并分析一下成因？

Ex－3：运行例 2,，观察结果，分析结果的成因。

Ex－4：如果将例 2 中“*if (execl("./hello","hello",NULL) < 0)*”代码改成“*if (execl("/bin/ls", "ls", "-l", NULL) < 0)*”，程序运行结果是什么。如果不用 exec 方式，而是自己编写一段代码，观察一下运行结果。考虑 wait 的作用。

Ex－5：自己编写一个程序，用循环语句实现父进程创建子进程 1，子进程 1 又创建另一个子进程 2，依次这样下去，一直创建到子进程 3。然后在屏幕上输出 pid、ppid 信息。