《Linux操作系统》实验2：进程及线程创建

班级：网安1901 学号：201904080139 姓名：赵昕然

一、实验目的

理解创建子进程函数的fork()的用法，通过观察运行结果理解进程的基本特征；通过代码及运行结果理解线程的概念，能够理解进程与线程之间的关联。

二、实验方法

本次实验属于验证型实验，按照实验内容的指导完成所有实验步骤，并记录下实验结果，遇到不懂的问题或是在某一步骤上卡壳，先尝试在搜索引擎上寻找解决方法，积极与老师、同学沟通，务必亲自将实验完成。

三、实验内容

1. 使用编辑器gedit新建一个1.c源文件，并输入后面的范例代码。

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

int main()
{
    pid_t pid,cid;
    printf("Before fork Process id :%d\n",getpid() );
    cid = fork();
    printf("After fork,Process id :%d\n",getpid() );
    return 0;
}
```

```
chendacongming@chendacongming-PC:~$ cd Desktop
chendacongming@chendacongming-PC:~/Desktop$ gcc 1.c -o 1
chendacongming@chendacongming-PC:~/Desktop$ ./1
Before fork Process id :15456
After fork,Process id :15456
After fork,Process id :15457
chendacongming@chendacongming-PC:~/Desktop$
```

原因：成功创建父子进程，并输出了两个pid值，原因是在fork（）函数后面的代码会执行两遍（父进程、子进程各执行一遍）

2.练习ps命令，该命令可以列出系统中当前运行的进程状态，我们在上面代码的21行处加入下面两行语句，目的是让父子进程暂停下　　来，否则我们无法观测到他们运行时的状态。

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

int main()
{
    pid_t pid,cid;
    printf("Before fork Process id :%d\n",getpid() );
    cid = fork();
    printf("After fork,Process id :%d\n",getpid() );
    int i;
    scanf("%d",&i);
    return 0;
}
```

```
chendacongming@chendacongming-PC:~/Desktop$ ps -al
F S   UID    PID   PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 R  1000  15880  15873  0  80   0 -  2890 -      pts/1    00:00:00 ps
chendacongming@chendacongming-PC:~/Desktop$
```

3.通过判断fork的返回值让父子进程执行不同的语句。

1.c 2.c

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

int main()
{
    pid_t cid;
    printf("Before fork process id :%d\n", getpid());

    cid = fork();

    if(cid == 0){

        printf("Child process id (my parent pid is %d):%d\n", getppid(),getpid());
        for(int i=0; i<3 ; i++)
            printf("hello\n");

    }else{

        printf("Parent process id :%d\n", getpid());
        for(int i=0; i<3 ; i++)
            printf("world\n");
    }

    return 0;
}
```

```
chendacongming@chendacongming-PC:~/Desktop$ gcc 2.c -o 2
chendacongming@chendacongming-PC:~/Desktop$ ./2
Before fork process id :16292
Parent process id :16292
world
world
world
Child process id (my parent pid is 16292):16293
hello
hello
hello
chendacongming@chendacongming-PC:~/Desktop$
```

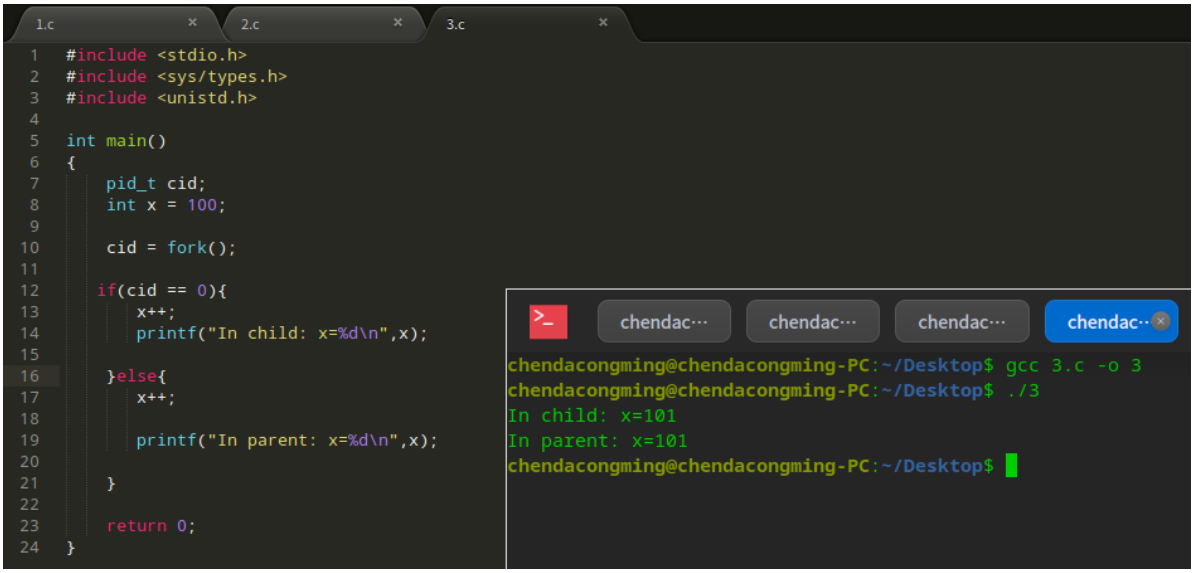
重新编译观察结果，重点观察父子进程是否判断正确（通过比较进程id）。父子进程其实是**并发**执行的，但实验结果好像是顺序执行的，多执行几遍看看有无变化，如果没有变化试着将两个循环的次数调整高一些，比如30、300，然后再观察运行结果并解释原因。

1.c 2.c 3.c

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

int main()
{
    pid_t cid;
    printf("Before fork process id :%d\n", getpid());

    cid = fork();

    if(cid == 0){

        printf("Child process id (my parent pid is %d):%d\n", getppid(),getpid());
        for(int i=0; i<5 ; i++)
            printf("hello\n");

    }else{

        printf("Parent process id :%d\n", getpid());
        for(int i=0; i<8; i++)
            printf("world\n");
    }

    return 0;
}
```

```
chendacongming@chendacongming-PC:~/Desktop$ gcc 2.c -o 2
chendacongming@chendacongming-PC:~/Desktop$ ./2
Before fork process id :17214
Parent process id :17214
world
world
world
world
world
Child process id (my parent pid is 17214):17215
world
world
world
hello
hello
hello
hello
hello
chendacongming@chendacongming-PC:~/Desktop$
```

4.验证父子进程间的内存空间是相互独立的。在终端中进入自己的主目录，使用gedit命令新建一文件3.c，输入下面的代码，然后编译运行，解释其原因。



5.在上一步的代码的20行添加如下语句，同时代码最顶端要包含一个新的头文件

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

void* threadFunc(void* arg){

        printf("In NEW thread\n");

}

int main()
{
    pthread_t tid;

    pthread_create(&tid, NULL, threadFunc, NULL);



    printf("In main thread\n");

    return 0;
}
```

```
chendacongming@chendacongming-PC:~/Desktop$ gcc 3.c -o 3
chendacongming@chendacongming-PC:~/Desktop$ ./3
In child: x=101
In parent: x=101
chendacongming@chendacongming-PC:~/Desktop$
```

6.创建线程。先关闭先前的文件，gedit 4.c以创建一个新的C语言源文件，将下面的代码拷贝进编辑器。

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

void* threadFunc(void* arg){

        printf("In NEW thread\n");

}

int main()
{
    pthread_t tid;

    pthread_create(&tid, NULL, threadFunc, NULL);



    printf("In main thread\n");

    return 0;
}
```

运行一下观察到什么现象了？

```
chendacongming@chendacongming-PC:~/Desktop$ gcc 4.c -o 4 -pthread
chendacongming@chendacongming-PC:~/Desktop$ ./4
In main thread
chendacongming@chendacongming-PC:~/Desktop$
```

将上面第18行代码的注释去掉又观察到了什么现象?

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

void* threadFunc(void* arg){

        printf("In NEW thread\n");

}

int main()
{
    pthread_t tid;

    pthread_create(&tid, NULL, threadFunc, NULL);

    pthread_join(tid, NULL);

    printf("In main thread\n");

    return 0;
}
```

```
chendacongming@chendacongming-PC:~/Desktop$
chendacongming@chendacongming-PC:~/Desktop$ gcc 4.c -o 4 -pthread
chendacongming@chendacongming-PC:~/Desktop$ ./4
In NEW thread
In main thread
chendacongming@chendacongming-PC:~/Desktop$
```

为什么?

该函数的作用是等待线程结束才能执行下一条指令，去掉注释后等指定的线程执行完成后，主线程才能执行。

试着在主线程和新线程里加入循环输出，观察一下输出的效果和并发父子进程的执行效果是否相似。

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

void* threadFunc(void* arg){
    for(int i=0;i<10;i++)
    {
        printf("In NEW thread\n");

    }

}

int main()
{
    pthread_t tid;

    pthread_create(&tid, NULL, threadFunc, NULL);

    for(int i=0;i<10;i++)
    {
        printf("In main thread\n");
    }

    return 0;
}
```

```
chendacongming@chendacongming-PC:~/Desktop$ gcc 4.c -o 4 -pthread
chendacongming@chendacongming-PC:~/Desktop$ ./4
In main thread
In main thread
In main thread
In main thread
In main thread
In main thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In NEW thread
In main thread
In main thread
In main thread
In main thread
chendacongming@chendacongming-PC:~/Desktop$
```

父子线程的执行和父子进程的执行相同均为并发执行。

四、总结

无