

《Linux操作系统》实验2：进程及线程创建

学号：201904080103 区队：网安1901 姓名：叶俊康

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、实验内容

1. 使用编辑器sublime text新建一个hello.c源文件，并输入后面的范例代码。

保存退出gedit，使用gcc对源文件进行编译，然后运行，

```
h.c
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <unistd.h>
4
5  int main()
6  {
7      //pid_t是数据类型，实际上是一个整型，通过typedef重新定义了一个名字，用
8      pid_t pid,cid;
9      //getpid()函数返回当前进程的id号
10     printf("Before fork Process id :%d\n", getpid());
11
12     /*
13     fork()函数用于创建一个新的进程，该进程为当前进程的子进程，创建的方法是：
14     fork()的返回值：
15     如果成功创建子进程，对于父子进程fork会返回不同的值，对于父进程它的返回
16     如果创建失败，返回值为-1。
17     */
18     cid = fork();
19
20     printf("After fork, Process id :%d\n", getpid());
21     pause();
22     return 0;
23 }
```

使用gcc命令得到一个可执行文件，并执行此文件。

```
kang@kang-PC:~/Desktop$ gcc h.c -o 02
kang@kang-PC:~/Desktop$ ./02
Before fork Process id :5952
After fork, Process id :5952
After fork, Process id :5953
```

原因：产生了子进程，其id为父进程id加一

- 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。

```
int i;

scanf("%d",&i);
```

重新编译运行程序，开启一个新的终端窗口输入下面的命令并观察运行结果。

ps -al

```
kang@kang-PC:~/Desktop$ ps -al
F S  UID      PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000     6142    5922  0  80   0 -   569 wait_w pts/0      00:00:00 03
1 S  1000     6143    6142  0  80   0 -   569 n_tty_ pts/0      00:00:00 03
0 R  1000     6147    6144  0  80   0 -  2896 -      pts/1      00:00:00 ps
kang@kang-PC:~/Desktop$
```

3.通过判断fork的返回值让父子进程执行不同的语句。

```
h.c
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <unistd.h>
4
5  int main()
6  {
7      pid_t cid;
8      printf("Before fork process id :%d\n", getpid());
9
10     cid = fork();
11
12     if(cid == 0){ //该分支是子进程执行的代码
13         printf("Child process id (my parent pid is %d):%d\n", getppid(),getpid());
14         for(int i=0; i<3 ; i++)
15             printf("hello\n");
16     }else{ //该分支是父进程执行的代码
17         printf("Parent process id :%d\n", getpid());
18         for(int i=0; i<3 ; i++)
19             printf("world\n");
20     }
21     return 0;
22 }
```

```
kang@kang-PC:~/Desktop$ ./04
Before fork process id :6315
Parent process id :6315
world
world
world
Child process id (my parent pid is 6315):6316
hello
hello
hello
kang@kang-PC:~/Desktop$
```

由结果可得“父子进程其实是**并发**执行的”

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h2.c，输入下面的代码，然后编译运行。

```
kang@kang-PC:~/Desktop$ ./5
In parent: x=99
kang@kang-PC:~/Desktop$ In child: x=101
```

由结果得父子进程间的内存空间是相互独立，是复制体。

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

```
#include <sys/wait.h>
```

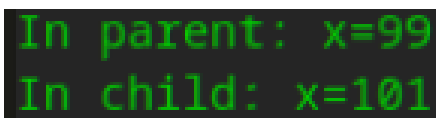
```
wait(NULL);
```

为了让效果更清楚，请将wait语句从20行移到18行，并在15行加上如下语句：

```
sleep(3);
```



```
h2.c
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <unistd.h>
4  #include <sys/wait.h>
5  int main()
6  {
7      pid_t cid;
8      int x = 100;
9
10     cid = fork();
11
12     if(cid == 0){ //该分支是子进程执行的代码
13         x++;
14         printf("In child: x=%d\n",x);
15         sleep(3);
16     }else{ //该分支是父进程执行的代码
17         x--;
18         wait(NULL);
19         printf("In parent: x=%d\n",x);
20     }
21
22     |
23     return 0;
24 }
```



```
In parent: x=99
In child: x=101
```

6.创建线程。先关闭先前的文件，gedit hThread.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。

```
hThread.c
1  #include <sys/types.h>
2  #include <unistd.h>
3  #include <stdio.h>
4  #include <pthread.h>
5
6  void* threadFunc(void* arg){ //线程函数
7
8      printf("In NEW thread\n");
9
10 }
11
12 int main()
13 {
14     pthread_t tid;
15
16     pthread_create(&tid, NULL, threadFunc, NULL);
17
18     //pthread_join(tid, NULL);
19
20     printf("In main thread\n");
21
22     return 0;
23 }
```

```
kang@kang-PC:~/Desktop$ ./ht
In main thread
kang@kang-PC:~/Desktop$
```

将上面第18行代码的注释去掉又观察到了什么现象？

```
kang@kang-PC:~/Desktop$ ./ht2
In NEW thread
In main thread
```

原因;主线程执行完毕，进程结束。新线程未开始执行

主线程和新线程里加入循环输出

```
hello (0)
hello (1)
hello (2)
hello (3)
hello (4)
hello (5)
hello (6)
hello (7)
hello (8)
hello (9)
hello (10)
world(0)
world(1)
world(2)
world(3)
world(4)
world(5)
world(6)
world(7)
world(8)
world(9)
world(10)
```

父子进程和父子线程的执行为并发执行。

五、总结

无。