

《Linux操作系统》实验2：进程及线程创建

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

```
#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;
}
```

```
ahua-PC:~$ gcc hellowprocess.c -o hellowprocess
ahua-PC:~$ ./hellowprocess
Before fork Process id :8967
After fork,Process id :8967
ahua-PC:~$ After fork,Process id :8968
```

fork()将当前进程完整拷贝了一份，子进程从fork()语句后开始执行

1.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);
```

```
ps -al
```

```
ahua@ahua-PC:~$ ps -al
  F S   UID     PID   PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
  0 S   1000    9272    4673   0  80   0 - 125364 poll_s pts/0    00:00:00 gedit
  0 R   1000    9333    9231   0  80   0 - 2896 -      pts/1    00:00:00 ps
ahua@ahua-PC:~$
```

2.通过判断fork的返回值让父子进程执行不同的语句。

```
#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){ //该分支是子进程执行的代码

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

    }else{ //该分支是父进程执行的代码

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

    return 0;
}
```

```
wbtchnxln@wbtchnxln-PC:~/Desktop/tempdir$ gcc a.c -o a
wbtchnxln@wbtchnxln-PC:~/Desktop/tempdir$ ./a
Before fork process id :6861
Parent process id :6861
world
world
world
Child process id (my parent pid is 6861):6862
hello
hello
hello
```

“hello”和“world”交替出现说明了他俩是并发进程

3.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，输入下面的代码，然后编译运行，解释其原因。

```

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
int main()
{
    pid_t cid;
    int x = 100;

    cid = fork();

    if(cid == 0){
        x++;

        printf("In child: x=%d\n",x);
        sleep(3);
    }else{
        x++;
        wait(NULL);
        printf("In parent: x=%d\n",x);

    }

    return 0;
}

```

4.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);

```

```

sleep(3);

```

```

-PC:~$ gcc hellowprocess2.c -o hellowprocess2
-PC:~$ ./hellowprocess2
: x=101
-PC:~$ In child: x=101

```

5. 创建线程。先关闭先前的文件，gedit helloThread.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

```

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

void* threadFunc(void* arg){ //xianxinghanshu

    printf("In NEW thread\n");

}

int main()
{
    pthread_t tid;

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

    |

    printf("In main thread\n");

    return 0;
}

```

将上面第18行代码的注释去掉又观察到了什么现象

```

i-PC:~/Desktop/tempdir$ gcc b.c -o b -pthread
i-PC:~/Desktop/tempdir$ gcc b.c -o b -pthread
i-PC:~/Desktop/tempdir$ ./b.c
够
i-PC:~/Desktop/tempdir$ ./b

i-PC:~/Desktop/tempdir$ gcc b.c -o b -pthread
i-PC:~/Desktop/tempdir$ ./b

i-PC:~/Desktop/tempdir$

```

线程中的语句背成功执行

总结

- 1.fork函数的使用。
- 2.进程是并发执行的
- 3.线程是CPU的基本单位
- 4.线程依赖于进程