shell 和进程

shell和进程的关系：

我们从login shell 说起，login shell用于表示登陆进程，是指用户刚登录系统时，由系统创建，用以运行shell 的进程。

这里先运行几个命令：

打印**登陆进程**（一直存在的，直到登陆退出）ID

george.guo@ls:~$ echo $PPID
3411
george.guo@ls:~$ ps -aux | grep 3411
george.+ 3411 0.0 0.0 99004 4520 ? S 11:00 0:00 sshd: george.guo@pts/46
打印登陆进程fork出的shell进程（一直存在的，直到登陆退出）
george.guo@ls:~$ echo $$
3412
george.guo@ls:~$ ps -aux | grep 3412
george.+ 3412 0.5 0.0 21380 5120 pts/46 Ss 11:00 0:00 -bash

从上面的几个命令可以看出：

登陆进程ID是3411，它创建了bash shell子进程3412。以后的脚本执行，

3412我们这里称为**主shell**，它会启动子shell进程处理脚本。

（注：在bash中，子shell进程的PID存储在一个特殊的变量‘$$’中，PPID存储子shell父进程的ID。）

我们写两个小程序验证下：

george.guo@ls:~$ cat yes.c

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

int main()
{
        pid_t pid;
        pid_t ppid;

        pid = getpid();
        ppid = getppid();
        system("./test");       //system will fork a process for exec ./test
        printf("yes pid = %d, yes ppid =  %d\n", pid, ppid);
}
```

george.guo@ls:~$ cat test

```
#!/bin/bash
echo "PID of this script: $$"
echo "test's PPID(system's fork id) = $PPID"
echo "tests's pid = $$"
```

运行结果如下：

george.guo@ls~$ ./yes

PID of this script: 6082
tests PPID(system's fork id)= 6081
echo tests self pid is 6082
yes PID = 6080, yes PPID = 3412

可见yes进程的父进程ID是3412，即登陆进程fork的bash shell子进程，**主shell**。这是因为

yes是由**主shell**执行的。yes进程ID是6080，调用system, fork出子shell ID为6081。

对于system调用：

使用system()运行命令需要创建至少两个进程。一个用于运行shell (这里其ID为6081)，

另外一个或多个则用于shell 所执行的命令(这里是一个子shell，就是脚本test本身).

脚本test本身进程ID为6082。