2.4 Unix进程实现

- **Unix进程定义**

❖UNIX的进程由三部分组成：

❖ proc结构(常驻内存的PCB)

❖ 数据段(执行时用到的数据)

❖ 正文段(程序代码)

合称进程映像，UNIX中把进程定义为映像的执行。

3.3 Unix进程控制实现

创建fork

- 父进程还应执行以下操作之一，以完成进程指派：
- 继续待在父进程中。把进程控制切换到父进程的用户模式，**在fork()点继续运行**，而子进程进入“就绪”状态。
- 把进程控制传递到子进程，**子进程在fork()点继续运行**，父进程进入“就绪”态。
- 把进程控制传递到其他进程，父进程和子进程进入“就绪”状态。

3.3 Unix进程控制实现

- **fork应用1**

分析程序的执行结果

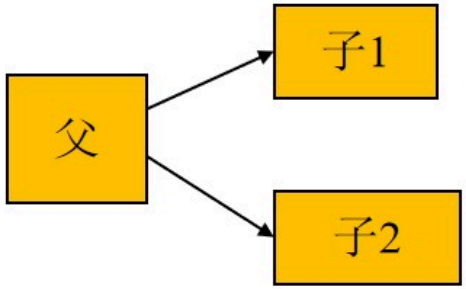
```
#include <stdio.h>
#include <unistd.h>
int main()
{ int p,x;
  x=2;
  if((p=fork())>0)
     printf("this is parent:  %d\n",++x);
  else
     printf("this is child:  %d\n",++x);
  return 0;
}
```

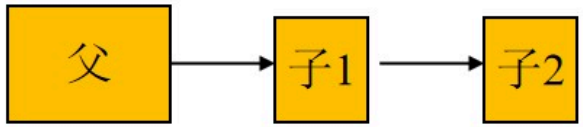
3

3

3.3 Unix进程控制实现

- **fork应用2**






```
main( )
{
 if(fork()==0)
   { 子1的代码段}
 else
   {if(fork()==0)
     {子2的代码段}
    else
     {父代码段}
   }
}
```

```
main( )
{
 if(fork()==0)
   { 子1的代码段;
     if(fork()==0)
       {子2的代码段}
     else
    { 子1的代码段}
   }
 else
   {父代码段}
}
```

3.3 Unix进程控制实现

- **fork应用3**

分析程序的执行结果

```
main()
{   int pid;
    printf("Before fork\n");
   while((pid=fork())==-1);
   if(pid){
       printf("It is parent process:PID=%d\n",  getpid());
       printf("Produce child  'sPID=%d\n" ,pid);}
      else
          printf("It is child process:PID=%d\n",getpid());
    printf("It is parent or child process:PID==%d\n",getpid());
}
```

3.3 Unix进程控制实现

- **fork应用4**

写出程序生成的进程树

```
#include<unistd.h>
#include<stdlib.h>
int main()
{  fork();
   fork();
   fork();
   sleep(20);
   return 0;
}
```