



西安邮电大学
XI'AN UNIVERSITY OF POSTS & TELECOMMUNICATIONS

Linux 编程技术



第3章 进程管理

——创建进程





bash shell

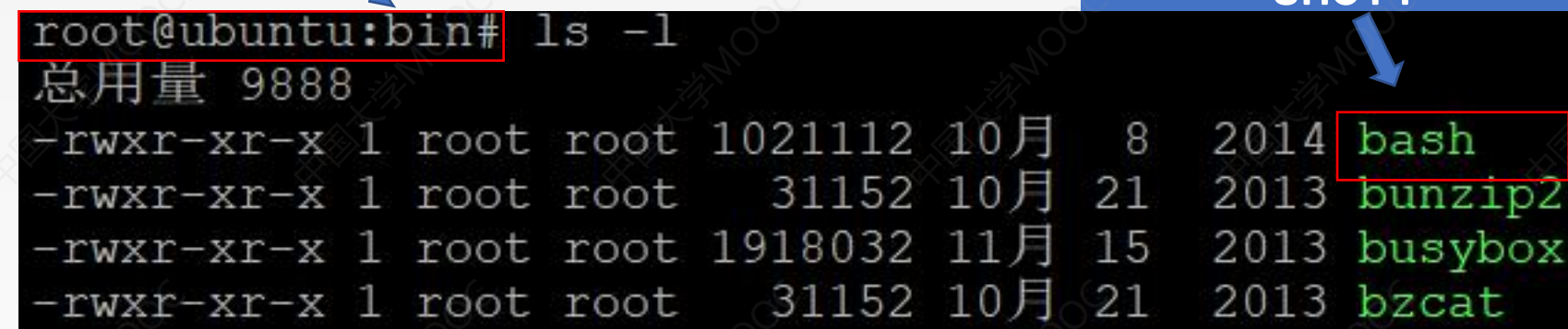


一个重要的进程-shell

Linux编程技术

shell已启动

/bin/bash就是
shell



```
root@ubuntu:bin# ls -l
总用量 9888
-rwxr-xr-x 1 root root 1021112 10月  8  2014 bash
-rwxr-xr-x 1 root root   31152 10月 21  2013 bunzip2
-rwxr-xr-x 1 root root 1918032 11月 15  2013 busybox
-rwxr-xr-x 1 root root   31152 10月 21  2013 bzipcat
```

shell是一个管理进程和运行程序的程序。

三个功能：

- 运行程序：如运行ls -l命令。
- 管理输入输出：提供提示符。
- 可编程：shell script脚本语言。

| getpid | | |
|--------|-----------------------|----|
| 功能 | 获取进程的进程号 | |
| 头文件 | /usr/include/unistd.h | |
| 函数原型 | pid_t getpid(void); | |
| 返回值 | 调用进程的进程号 | 成功 |
| | -1 | 失败 |

parent

getppid

| 功能 | 获取进程的父进程号 | |
|------|-----------------------|----|
| 头文件 | /usr/include/unistd.h | |
| 函数原型 | pid_t getppid(void); | |
| 返回值 | 调用进程的父进程号 | 成功 |
| | -1 | 失败 |

例1: getpid和getppid的使用

例1: 显示进程的进程号、父进程号

```
#include <stdio.h>
```

```
#include <unistd.h>
```

```
main() {
```

```
    pid_t pid, ppid;
```

```
    pid = getpid();
```

```
    ppid = getppid();
```

```
    printf ("PID is %d, its parent's PID is %d.\n", pid, ppid);}
```

```
root@ubuntu:4# ./getpid_demo
PID is 45636, its parent's PID is 45608
```

45608为shell
的pid

```
root@ubuntu:4# ps -aux |grep bash
root      45459  0.0  0.1  27032  5556 pts/12    Ss+   09:11   0:00 -bash
root      45549  0.0  0.1  27112  5604 pts/13    Ss+   09:20   0:00 -bash
root      45608  0.0  0.1  27032  5548 pts/14    Ss    10:07   0:00 -bash
```

通过ps命令和例1，我们能够看到，shell本身是一个进程，即bash进程

exit命令可以退出shell

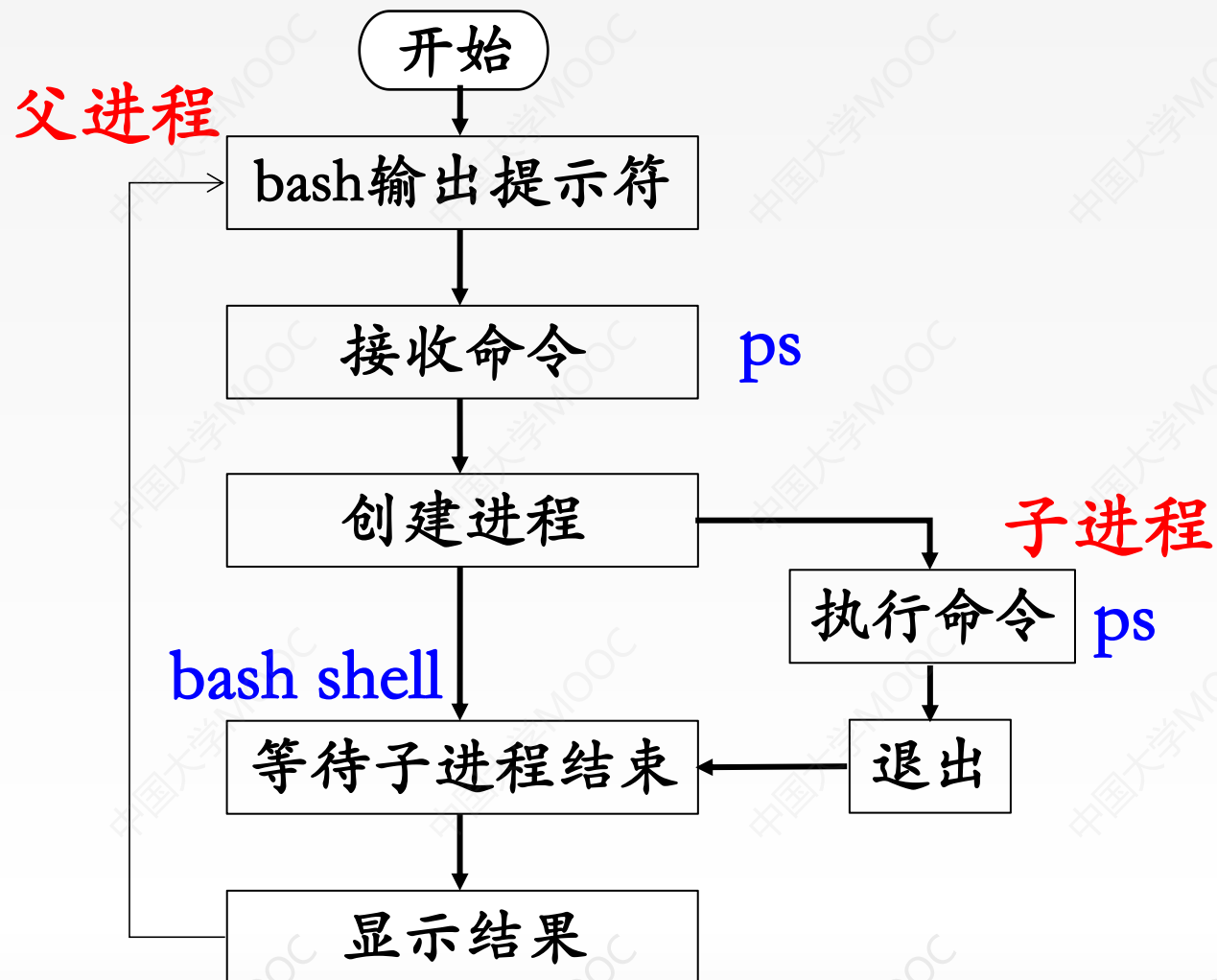
shell是一个无穷循环，等待执行输入的命令、程序或脚本

执行一条命令的过程：

- 接收输入的命令（如ps）
- 新建一个进程，用来运行该命令
- 将命令对应的程序从磁盘加载到新进程中执行（如/bin/ps）
- 等待新进程运行结束

shell程序的流程

- 前一条命令结束才能执行下一条命令



1. 怎样产生新进程?

fork、vfork

2. 怎样在进程中运行一个程序?

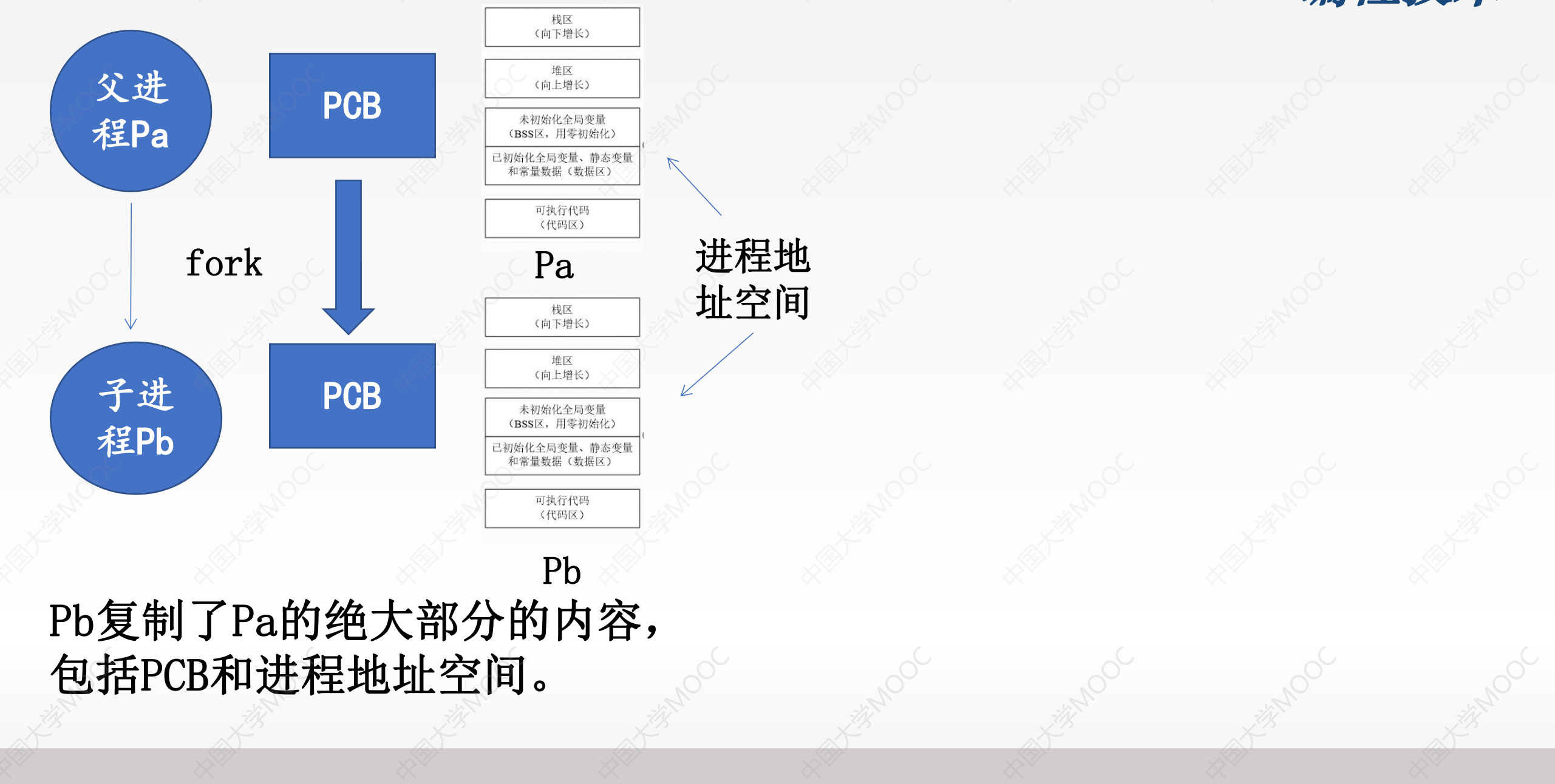
eXec

3. 怎样让shell等待子进程结束?

wait、waitpid

| fork | |
|------|-----------------------|
| 功能 | 产生一个新的进程 |
| 头文件 | /usr/include/unistd.h |
| 函数原型 | pid_t fork(void); |
| 返回值 | -1 创建失败（父进程） |
| | >0 子进程号（父进程） |
| | 0 创建成功（子进程） |

执行成功时，会有两个返回值，一个返回给父进程（子进程PID），一个返回给子进程（为0）。



fork时，内核并不复制整个进程的地址空间，而是让父子进程共享同一个地址空间。只有在需要写入的时候才会复制地址空间，从而使各个进程拥有各自的地址空间。

资源的复制是在需要写入的时候才会进行，在此之前，以只读方式共享。

例2: fork的使用

```
//fork_hello.c
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

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

```
main() {
```

```
    pid_t pid;
```

```
    if((pid=fork())== -1) {
```

```
        perror("fork"); exit(EXIT_FAILURE);
```

```
    }
```

```
    printf("hello\n");    return 0;
```

```
}
```

例2: fork执行分析

哪个hello是父进程输出的?

父进程

```
.....  
if((pid=fork())== -1) {  
    perror("fork");  
    exit(EXIT_FAILURE);  
}  
printf("hello\n");  
.....
```

父进程

```
.....  
if((pid=fork())== -1) {  
    perror("fork");  
    exit(EXIT_FAILURE);  
}  
printf("hello\n");  
.....
```

继续执行

子进程

```
.....  
if((pid=fork())== -1) {  
    perror("fork");  
    exit(EXIT_FAILURE);  
}  
printf("hello\n");  
.....
```

继续执行

例3：区分父子进程

```
#include <stdio.h>
```

```
#include <unistd.h>
```

```
int main() {
```

```
    pid_t pid;
```

```
    printf("Let's distinguish parent and child!\n");
```

```
    pid=fork(); 创建进程
```

继续执行

```
    switch(pid) {
```

```
        case -1: perror("fork");
```

```
            break;
```

```
        case 0: printf("This is child process, pid=%d, ppid=%d!\n", getpid(), getppid());
```

```
            break;
```

```
        default: sleep(1);
```

```
            printf("This is parent process, pid=%d!\n",getpid());
```

```
            break;
```

```
    }
```

```
    return (0);
```

结论：fork时，通过fork返回值区分父子进程。



父进程，接收>0的
返回值，即子进程PID



子进程，接收=0的返回值。

例3：区分父子进程

```
[huangru@xiyoulinux chap3]$ gcc fork_fmt.c -o  
fork_fmt  
[huangru@xiyoulinux chap3]$ ./fork_fmt  
Let's distinguish parent and child!  
This is child process, pid=29172, ppid=29171!  
This is parent process, pid=29171!
```

shell是一个程序：接收用户命令，执行命令，显示命令结果

shell的实现过程：fork、exec函数族、wait、waitpid

fork系统调用：复制方式创建子进程

fork的具体实现：写时复制技术

fork具体使用：特殊现象，两个返回值

区分父子进程方法：fork返回值

获取进程ID号的方法：getpid和getppid



西安邮电大学
XI'AN UNIVERSITY OF POSTS & TELECOMMUNICATIONS

Linux 编程技术



谢谢大家！