



## Linux 进程介绍

1

## 主要内容和学习要求

- q 理解进程与多进程的概念
- q 掌握如何运行后台进程
- q 掌握如何进行进程控制
- q 掌握相关命令的使用

2

## 进程

### q 进程概念

- u 正在运行的程序叫做进程 (**process**)  
程序只有被系统载入内存并运行后才能称为进程。
- u **Linux** 允许同时运行多个程序, 为了区分每一个运行的程序, **Linux** 给每个进程都做了标号, 称为进程号 (**process ID**), 每个进程的进程号是唯一的。
- u 查看当前运行的程序及其进程号: **ps**

3

## 多进程

### q 多进程

- u **Linux** 是一个多用户的操作系统, 当多个用户同时在一个系统上工作时, **Linux** 要能够同时满足用户们的要求, 而且还要使用户感觉不到系统在同时为多个用户服务, 就好像每一个用户都单独拥有整个系统一样。
- u **Linux** 不可能在一个 **CPU** 上同时处理多个任务 (作业) 请求, 而是采用 “分时” 技术来处理这些任务请求。

4

## 多进程

### u 分时技术

所有的任务请求被排除一个队列, 系统按顺序每次从这个队列中抽取一个任务来执行, 这个任务执行很短的时间 (几毫秒) 后, 系统就将它排到任务队列的末尾, 然后读入队列中的下一个任务, 以同样的方式执行。这样经过一段时间后, 任务队列中的所有任务都被执行一次, 然后又开始新一轮循环。

### u 任务/作业

就是一个被用户指定运行的程序。如用户发出一个打印命令, 就产生一个打印任务/作业, 若打印成功, 表示任务完成, 没有成功表示任务没完成。

5

## 多进程

- u **Linux** 是多用户系统, 它必须协调各个用户。

**Linux** 给每个进程都打上了运行者的标志, **用户可以控制自己的进程**: 给自己的进程分配不同的优先级, 也可以随时终止自己的进程。

6

## 前台与后台

### u 前台进程

指一个程序控制着标准输入/输出, 在程序运行时, **shell** 被暂时挂起, 直到该程序运行结束后, 才退回到 **shell**。在这个过程中, 用户不能再执行其它程序。

### u 后台进程

用户不必等待程序运行结束就可以执行其它程序。

- u 在一个终端里只能同时存在一个前台任务, 但可以有多 个后台任务。

7

## 运行后台进程

### q 运行后台进程

- l 在命令最后加上 “&”

```
例: sleep 60 &
```

- l 如果程序已经在前台运行, 需要将其改为后台运行, 这时可以先按组合键 **Ctrl+z**, 将任务挂起, 然后利用 **bg** 命令将该程序转为后台运行
- l 若要将一个后台进程转到前台运行, 可以使用 **fg** 命令
- l 相关命令: **jobs, bg, fg**

8

## jobs 命令

q jobs: 查看后台运行或被挂起的进程

例:

```

[jypan@server236 ~]$ jobs
[1]  Stopped                  sleep 111
[2]- Stopped                  sleep 112
[3]+ Stopped                  sleep 113

```

| 第一列显示的是作业号

| “+”表示当前作业，“-”表示当前作业之后的作业

| 若加上选项 -l，则显示进程号

```

[jypan@server236 ~]$ jobs -l
[1] 16368 Stopped                  sleep 111
[2]- 16369 Stopped                  sleep 112
[3]+ 16371 Stopped                  sleep 113

```

9

## bg / fg 命令

q bg: 将被挂起的进程转化到后台运行

```
bg jobnumber
```

| jobnumber 是通过 jobs 查出来的作业号

例:

```
bg 2
```

```
bg 1 2
```

q fg: 将后台进程转化到前台运行

| 用法与 bg 类似

10

## 进程控制: ps

q 查看正在运行的程序: ps

```
ps [选项]
```

例:

```
ps
```

```
ps -u jypan
```

```
ps u -u jypan
```

```
ps u -u jypan --sort=cmd
```

```
ps -u jypan -o "%U %p %c %x %t"
```

11

## ps 常用选项

- A, -e 显示所有进程
- u 查看指定用户的进程 (用户名或用户ID)
- l 长格式显示, 可查看各个进程的优先权值
- f 完全显示 (显示完整的命令)
- C 列出指定命令名称的进程

- u 增加用户名, 起始时间, CPU和内存使用等信息
- a 显示终端机下用户执行的进程, 包含其它用户
- f 显示进程树, 等价于 --forest
- r 显示正在运行的进程

- o 按指定的格式输出
- sort 按指定内容进行排序

12

## ps 举例

```
ps -A
```

```
ps -u jypan
```

```
ps -u jypan -l 或 ps -l -u jypan
```

```
ps -u jypan -f
```

```
ps -C sleep -f
```

```
ps -C sleep u
```

```
ps af
```

```
ps r
```

13

## 常见列标志的含义

```
例: ps -u jypan u
```

PID	进程 ID	CMD	命令名 (COMMAND)
UID	用户 ID	START	进程启动时间
USER	用户名	%CPU	进程所用CPU时间百分比
TIME	执行时间	%MEM	进程所有MEM百分比
STAT	进程状态	NI	优先权值 / nice 值

TTY	启动进程的终端	RSS	进程所用内存块数
PGID	进程组 ID	VSZ	所用虚拟内存块数

| 更多列标志见 man ps

14

## 进程状态

R	正在运行或处在运行队列中
S	休眠 (等待)
T	停止或被追踪
D	不可中断的睡眠, 通常指 I/O
Z	僵尸进程 (已结束但未被告进程收回)
X	已死进程 (这个状态不会出现)

<	具有最高优先权
N	具有较低的优先权
s	is a session leader
l	is multi-threaded
+	is in the foreground process group

15

## 指定输出格式

```
例: ps -u jypan -o "%U %p %c %x %t"
```

| 输出格式中的常用字段

%U	用户名	%c	命令名
%u	用户名	%a	命令名 (含选项与参数)
%G	用户组	%p	进程号
%g	用户组	%x	运行时间
%C	CPU	%t	Elapsed time
%P	父进程	%n	nice 值 (代表优先权)

```
%r PGID -- ID of the process group (leader)
```

```
例: ps -u jypan -o %c%p%r%n
```

16

## 指定输出格式

另一种使用方式

```
ps -u jypan -o user,pid,pcpu,time,etime
```

字段对应表

%U	用户名	user	%c	命令名	comm
%u	用户名	ruser	%a	命令名	args
%G	用户组	group	%p	进程号	pid
%g	用户组	rgroup	%x	运行时间	time
%C	CPU	pcpu	%t	Elapsed time	etime
%P	父进程	ppid	%n	nice 值	nice
%r	进程组	pgid		用户ID	uid

17

## 进程排序

例: `ps au --sort=uid,-pid`

```
uid  用户名
user 用户名
cmd  命令名
pid  进程ID
ppid 父进程ID
pgrp 用户组ID
size 内存大小
pcpu  CPU
utime 用户时间
start_time 起始时间
```

更多选项见 `man ps`

18

## nohup 命令

用户退出系统后能继续运行的进程

通常当用户退出系统后，所有属于该用户的进程将全部被终止。如果希望程序在退出系统后仍然能够继续运行，需使用 `nohup` 命令后台启动该进程

```
nohup 命令 [选项] [参数] &
```

若有输出，则通常输出到指定的文件中

19

## 进程的优先权

进程的 nice 值和 优先权

在任务队列中的进程并不享有同等的优先权，每个进程都有一个指定的 nice 值（优先权值），从 -20 到 19  
nice 值为 -20 的进程具有最大优先权

进程的缺省 nice 值为 0

```
例: 查看缺省的 nice 值: nice
```

查看进程的 nice 值

```
ps -l 进程号 % NI 的值
```

20

## 进程的优先权

调整进程的 nice 值

可以在进程启动时指定，也可以在启动后修改

在启动进程时就指定优先级: `nice`

```
nice -n 命令 &
```

n 是指优先级的增量

- 若为正，表示增加 nice 值，即降低进程优先权
- 若为负，表示减小 nice 值，即提高优先权
- 若缺省，则默认为 10，即 nice 值增加 10

21

## 进程的优先级

例: `nice -5 sleep 60 &`

普通用户只能增加 nice 值  
只有系统管理员才能降低一个进程的 nice 值

```
例: sudo nice --5 sleep 60 &
```

使用 nice 同样可以增加前台任务的 nice 值

```
例: nice -5 sleep 60
```

22

## 进程的优先级

进程运行后调整 nice 值: `renice`

进程已经运行，此时又有许多用户登录，他们使得各个进程分得的 CPU 时间下降。此时，root 可以提高进程的优先权，但普通用户没这个权限，在系统资源紧张时，只能通过降低其它不着急的进程的优先权，从而使得急用的进程能分得更多的 CPU 时间。

```
renice n [-p pid] [-u user] [-g pgid]
```

- 增加指定进程的 nice 值
- n 可以是正的，也可以是负的；
- 注意与 nice 命令的区别：没有减号
- pgid 是进程组的 ID

23

## 进程的优先级

```
例: renice 5 2673 % 增加进程2673的nice值
      % -p 可以省略
```

增加进程2673的 nice 值  
-p 可以省略

```
例: renice 5 -u jypan
```

增加指定用户的所有进程的 nice 值

注：普通用户一旦增加某个进程的 nice 值（即降低优先权）后，就无法再回复到原来的 nice 值

24

## 终止进程

### q 终止进程

- | 终止前台进程使用: `Ctrl+c`
- | 终止后台进程使用: `kill`
- | `kill` 有两种方法: 正常结束和强制结束

`kill pid`

`kill -9 pid`

注: (1) 使用 `kill` 前需要先用 `ps` 查看需要终止的进程的pid;  
(2) `kill -9` 很霸道, 它在杀死一个进程的同时, 将杀死其所有子进程, 使用时要谨慎。如错杀 `login` 进程或 `shell` 进程等。

25

## 常用 bash 内部命令

### q 一些常用的 bash 内部命令

- | `alias/unalias`: 设置和取消 `bash` 别名。
- | `bg`: 使一个被挂起的进程在后台继续执行。
- | `cd`: 切换当前工作目录。
- | `exit`: 退出 shell。
- | `export`: 使变量的值对当前 shell 的所有子进程都可见。
- | `fc`: 用来显示和编辑历史命令列表里的命令。
- | `fg`: 使一个被挂起的进程在前台继续执行。
- | `help`: 显示帮助信息。
- | `kill`: 终止某个进程。
- | `pwd`: 显示当前工作目录。

26

## 相关命令

- | `id`: print real and effective UIDs and GIDs
- | `who`: show who is logged on
- | `whoami`: `id -un`
- | `hostname`: show or set the system's host name
- | `w`: show who is logged on and what they are doing
- | `last`: show listing of last logged in users
- | `finger`: displays information about the system users
- | `top`: display Linux tasks (很有用的系统监控工具)

- | 更多 `bash` 内部命令见: `man bash --> 3370`  
或任一 `bash` 内部命令的 manual, 例: `man bg`

27