

Linux 进程 与作业

Processes

Jobs

- 进程管理
- 前台与后台进程
- 进程与作业控制
- 守护进程
- PS
- 监视系统进程



进程管理方式

- 内核维护进程表 (**process table**)
 - 每个进程均有一个条目，依据 **PID** 进行追踪管理
 - **\$\$** : 表示当前**shell**的**PID**
 - 由内核提供，基于时间片的进程调度服务
 - 内存管理
 - 文件管理
 - 进程管理
 - 安全与访问控制
 - 进程通信
 - 网络访问
 - 输入输出

#0 进程

- 系统通电引导后，手动（而非分叉）创建一个PID为0的特殊进程，用于初始化内核所需的数据结构等
- 之后，#0 进程分叉建立 #1 进程
- 之后，#0 进入无限循环，若没有其他进程执行，则调用它。但它本身并不做任何事情
- 因此，#0 进程也被称为空闲进程 **idle process**

1 进程

- #1 进程被 #0 分叉创建
- 打开系统控制台 (一个特殊的`shell`)，挂载文件系统 (`/etc/fstab`)，执行初始化脚本 (`/etc/inittab`)
- 其间，#1 多次分叉，建立系统所需要基本进程，成为系统中所有其他进程的祖先
- 因此，#1 又被称为初始化进程 `init process`

进程生命周期

1. 父进程使用 `fork` 系统调用来创建
2. 子进程使用 `exec` 系统调用来替换进程程序
3. 父进程可以使用 `wait` 系统调用来暂停自身并等待子进程结束
4. 子进程使用 `exit` 系统调用来停止自身

进程生命周期

- 5. 进程死亡后，它使用的所有资源：内存、文件等都被释放，以便其他进程使用
- 6. 仍保留进程表条目，称为僵进程 (**zombie**)
- 7. 父进程随子进程的消亡而被内核唤醒
- 8. 父进程查看**zombie**，并将其从进程表中删除

进程生命周期

- 若父进程在子进程结束前，意外终止
 - 子进程会继续执行，成为孤儿进程
 - 子进程结束后，成为孤儿僵进程
 - 孤儿僵进程会被 #1 进程，即`init`进程收养，并终结
 - 用户可以通过 `kill` 来终结

前台进程与后台进程

- 前台进程：
 - `vi` 等进程启动后，就占据 `shell`，无法输入其它命令
 - `$ sort < BIGFILE > results #` 启动后，也会占据 `shell`，只有等命令结束后，才会将控制权返回 `shell`
- 后台进程：
 - 进程启动后，`shell`不必等程序结束，而直接获取控制权

前台进程与后台进程

- 启动一个后台进程，只需在命令结尾加入 **&** 符
- `$ sort < BIGFILE > results &`
- 作为后台进程：
 - 无法从键盘获取STDIN
 - 仍将STDOUT, STDERR输出到屏幕上

前台进程与后台进程

- 程序进入后台后，会显示进程的PID
- 后台进程不再响应 `intr(^C)` , `quit(^\\)` 信号
- 要强行终止后台进程，只能使用 `kill` 命令

进程与作业

- 进程：正在执行或者准备执行的程序
- 作业：解释整个命令行（完成某项自动化任务）所需要的全部进程
 - `$ who | cut -c 1-8 | sort | uniq -c &`
 - `$ (date; who; uptime; cal 6 2016) &`
 - 上述命令会生成4个PID，但在作业表中则只有一项

后台作业

- `$ (date; who; uptime; cal 6 2016) &`

`[1] 2183`

`作业ID 最后进程的PID`

- 当作业结束后，则提示

`[1]+ Done (date; who; uptime; cal 6 2016)`

后台作业

- 作业的3种状态
 - 前台运行
 - 后台运行
 - 暂停 `suspend`

后台作业

- 通过 `^z` 键, 发送`susp`信号, 可以将前台作业暂停

[1]+ Stopped

vi

- 通过 `fg` 命令可以恢复挂起的作业
 - `$ man fg`
 - `$ type fg`
 - `fg` 是 `shell builtin`

显示作业列表

- `$ jobs [-1]`
- 显示所有作业列表，包括挂起的和后台的作业
- `$ sleep 20 &`
- `-1` : 显示作业的进程ID
- `+` : 当前作业
- `-` : 前一个作业

移动作业到前台

- `$ fg %[job]`
- 若没有指定作业, `fg` 只能恢复当前作业

- 指定`job`的方式:

作业号	含义
<code>%%</code>	当前作业
<code>%+</code>	当前作业
<code>%-</code>	前一个作业
<code>%n</code>	作业#n
<code>%name</code>	含有指定命令名的作业
<code>%?name</code>	命令任意位置含有 <code>name</code> 的作业

移动作业到后台

- `$ bg %[job]`
 - `fg`: foreground
 - `bg`: background
- 可以先通过`^z`暂停作业，然后再通过 `bg` 激活
- `$sleep 1m (^Z)`
- `$bg`

守护进程(daemon)

- 守护进程：在后台运行的，并非由用户运行的程序
- **Disk and Executing Monitor**
- 完全不与任何终端连接，仅提供服务
- 由特定动作触发：事件、请求、中断、时间片等

守护进程(daemon)

- 多数名称以d结尾
- `init`, `apache`, `crond`, `cupsd`, `ftpd`, `httpd`,
`lpd`, `mysql`, `nfsd`, `ntpd`, `routed`, `sshd`,
`sendmail`, `syslogd` ...



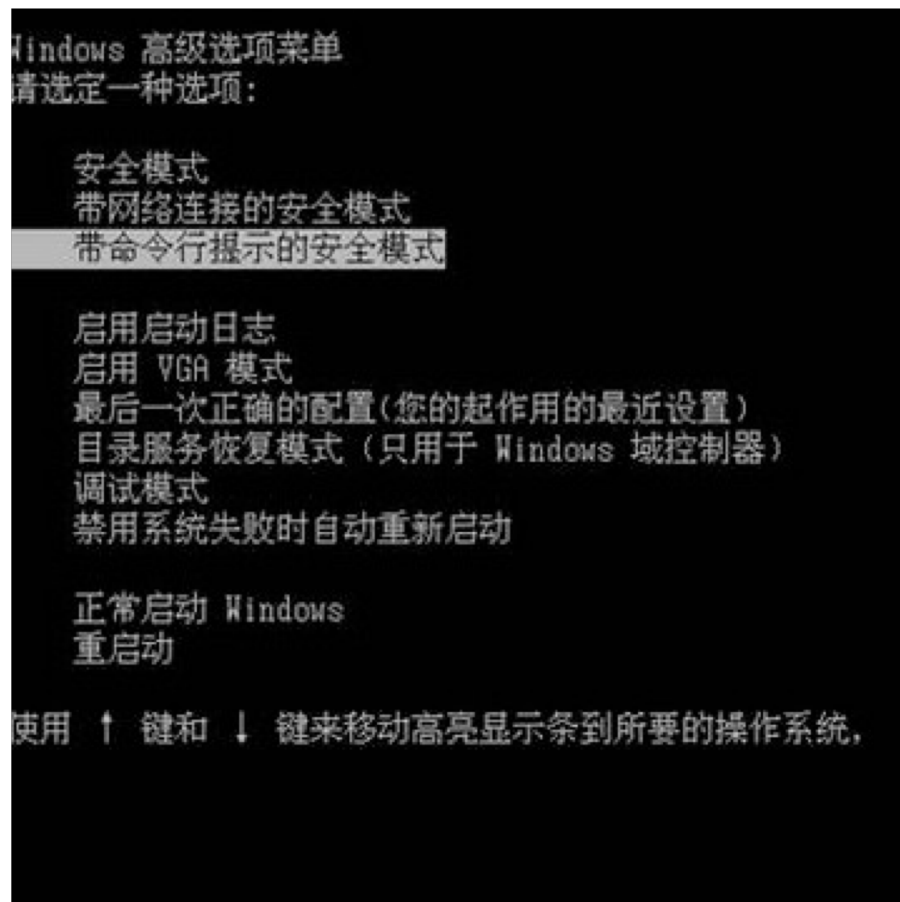
守护进程(daemon)

- (全部) 守护进程的程序保存在 `/etc/init.d`
- 查看进程状态可以
 - `$ ps -e | grep ***`
 - `$ /etc/init.d/** status` (不是全部都有此选项)
 - `$ service *** status` (不是所有的Linux中有此命令)

守护进程(daemon)


- 可用的守护进程选项
 - `start`
 - `stop`
 - `reload`
 - `restart`
 - `force-reload`
 - `status`

运行级别 (Linux & System V, not BSD)



运行级别	描述
0	关机
1	单用户模式：命令行
2	非标准化
3	多用户模式：命令行
4	非标准化
5	多用户模式：GUI
6	重新启动

由 /etc/inittab 中的initdefault值来设置

 \$ runlevel
\$ sudo init 6

PS

- `$ ps (process status)`
- 显示进程信息
- **AT&T** 和 **BSD Unix**都有自己的`ps`版本，两者均被广为传播，但拥有不同的选项，分别被称为**Unix**选项和**BSD**选项。某些**Linux**同时支持两种选项，也有一些只支持一类。**GNU/Linux**则又支持一些只适用于**GNU**的选项。

PS 选项

- UNIX 选项以 - 开头
- `$ ps [-aefFly] [-p pid] [-u userid]`
- BSD 选项没有
- `$ ps [ajluvx] [p pid] [U userid]`
- 两类选项不可以混用

PS 常见选项

- 默认：与当前用户和终端相关的进程
 - **-a**：与任何用户和终端相关的进程，不包括守护进程
 - **-e**：所有进程，包括守护进程
 - **-p**：指定进程
 - **-u**：指定用户
 - **-f**：显示更加全面的内容
- 显示当前全部进程
 - `$ ps -e`
- 显示进程树
 - `$ ps -ejH`
- 显示线程信息
 - `$ ps -eLf`

top

- **ps** 显示系统进程状态的快照
- **top** 可以对当前进程进行追踪监控
- **-d** : **delay**, 更新频率
- **-n** : 刷新次数
- **h**: 显示帮助; **1**: 显示**core**使用情况; **q**: 退出

pstree

- 显示进程树
- **-p**: 显示进程PID
- **-n**: 依据PID排序, 默认按字母排序
- **-u**: 显示进程用户标识
- 找找当前进程的位置?

screen 终端管理器

- `$ for i in {1..500}; do echo $i >> out; sleep 1; done &`
- 如果我们退出当前终端，会发生什么事情？
- 进程在后台运行，但是无法再恢复到前台
- `ps -e | grep xxxx` # 该进程不属于任何终端
- `pstree -p | grep xxxx`
 - # 该进程的父bash进程交给了init

screen 终端管理器

- `$ sudo apt-get install screen`
- `$ screen -S foo` # 创建一个新终端窗口
 - `^A-D` : 退出当前窗口
- `$ screen -ls` # 查看已创建的窗口
- `$ screen -r foo` # 回到已创建窗口

Kill : 杀死进程

- `$ kill [-signal] pid... | jobid...`
- 杀死指定进程，作业
- 向进程发送信号，默认为15 (SIGTERM)
- 确定杀死信号： -9
- `$ kill -l # 显示信号列表`

KILL BILL GATES

