

Unix系统高级编程

进程和进程ID

Unit13

进程



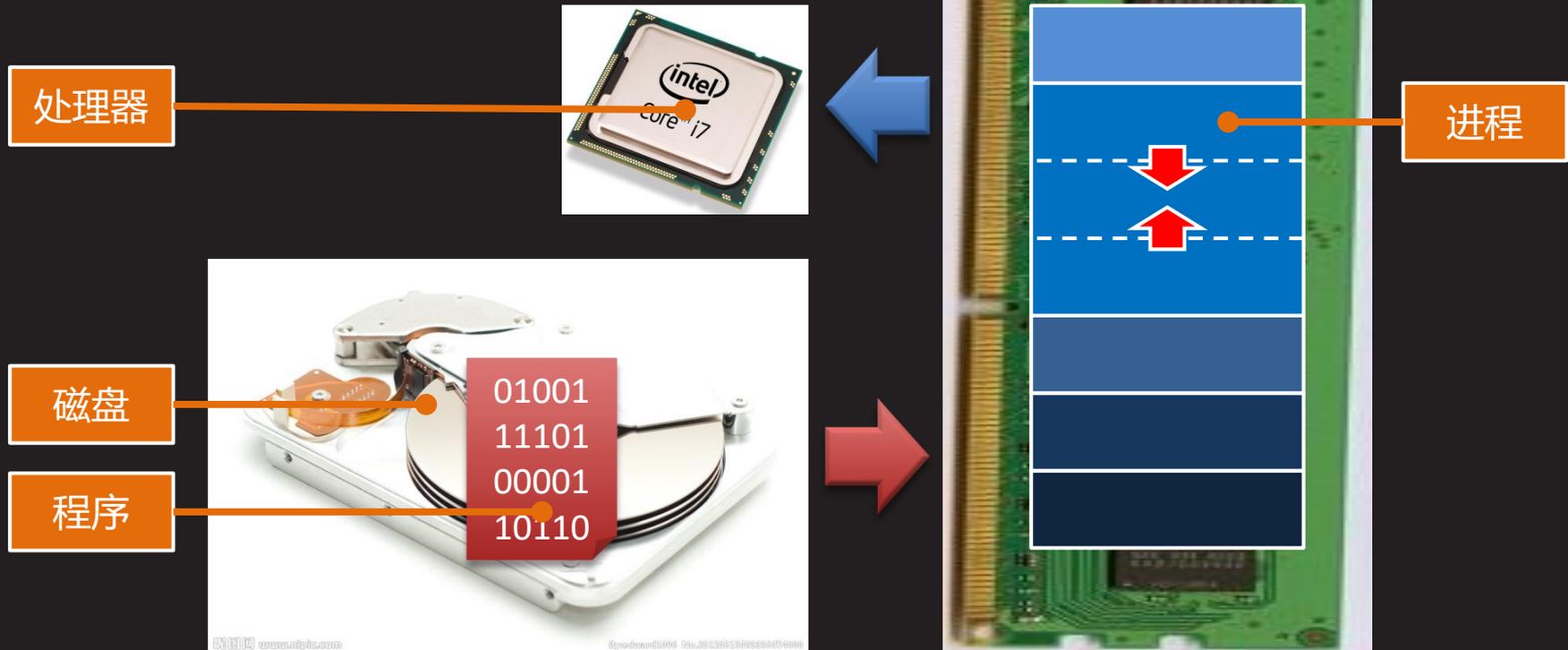
进程的基本概念



进程与程序

- 程序是被存储在磁盘上，包含机器指令和数据文件
- 进程是被装载到内存中，被处理器操作的代码和数据
- 一个程序可被同时运行为多个进程
- 进程在操作系统中执行特定的任务

知识讲解



进程的分类

- Unix/Linux系统中的进程一般被分为以下三类
 - 交互式进程
 - 由Shell启动，既可在前台运行，也可在后台运行，通过终端接收用户的输入，并为用户提供输出
 - 如：vi、ps等
 - 批处理进程
 - 与终端没有联系，以进程序列的方式，在无需人工干预的条件下完成一组批量任务
 - 如：各种Shell脚本程序
 - 守护进程
 - 又名精灵进程，是系统的后台服务进程
 - 独立于控制终端，周期性地执行某种任务或等待某些事件
 - 在系统引导时启动，在系统关闭时终止，生命周期很长
 - 如：crond、lpd等



进程快照



进程快照

- 使用ps命令可以查看当前在系统中运行的进程快照
 - 简单形式

```
$ ps
```

以简略方式显示当前用户拥有控制终端的进程信息

```
PID TTY          TIME CMD
2668 pts/0        00:00:00 su
2697 pts/0        00:00:02 bash
16070 pts/0        00:00:00 ps
```

其中各列含义如下：

- **PID** - 进程标识
- **TTY** - 控制终端次设备号
- **TIME** - 进程运行时间
- **CMD** - 进程启动命令



进程快照 (续1)

- 使用ps命令可以查看当前在系统中运行的进程快照
 - BSD风格常用选项
 - **a** - 显示所有用户拥有控制终端的进程信息
 - **x** - 也包括没有控制终端的进程
 - **u** - 以详尽方式显示
 - **w** - 以更大列宽显示

\$ ps axu

```

USER      PID %CPU %MEM    USZ    RSS TTY      STAT START   TIME COMMAND
avahi     868  0.0  0.0   3464    348 ?        S      01:51   0:00 avahi-daemon: c
colord    917  0.0  1.0  53572 10428 ?        Sl     01:51   0:00 /usr/lib/i386-1
root      953  0.0  0.0   4636    860 tty2     Ss+    01:51   0:00 /sbin/getty -8
root      967  0.0  0.0   4636    856 tty6     Ss+    01:51   0:00 /sbin/getty -8
root     1001  0.0  0.0   2180    712 ?        Ss     01:51   0:00 acpid -c /etc/a
root     1002  0.0  0.0   2624    932 ?        Ss     01:51   0:00 cron
daemon   1003  0.0  0.0   2476    348 ?        Ss     01:51   0:00 atd
    
```



进程快照 (续2)

- 使用ps命令可以查看当前在系统中运行的进程快照
 - SVR4风格常用选项
 - -e - 显示所有用户的进程信息
 - -f - 按完整格式显示
 - -F - 按更完整格式显示
 - -l - 按长格式显示

\$ ps -efl

```
F S UID      PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  STIME TTY      TIME CMD
1 S avahi    868   857   0  80   0 -      866 unix_s 01:51 ?        00:00:00 avahi
4 S colord  917     1   0  80   0 -    13393 poll_s 01:51 ?        00:00:00 /usr/
4 S root    953     1   0  80   0 -    1159 n_tty_ 01:51 tty2    00:00:00 /sbin
4 S root    967     1   0  80   0 -    1159 n_tty_ 01:51 tty6    00:00:00 /sbin
1 S root   1001     1   0  80   0 -     545 poll_s 01:51 ?        00:00:00 acpid
1 S root   1002     1   0  80   0 -     656 hrtime 01:51 ?        00:00:00 cron
1 S daemon 1003     1   0  80   0 -     619 hrtime 01:51 ?        00:00:00 atd
```



进程快照 (续3)

- 进程信息列表
 - **USER/UID** : 进程的用户ID
 - **PID** : 进程ID
 - **%CPU/C** : CPU使用率
 - **%MEM** : 内存使用率
 - **VSZ** : 占用虚拟内存大小(KB)
 - **RSS** : 占用物理内存大小(KB)
 - **TTY** : 终端次设备号
 - **ttyn** - 实终端/物理终端
 - **pts/n** - 伪终端/虚拟终端
 - **?** - 无控制终端, 如后台进程



进程快照 (续4)

- 进程信息列表
 - **STAT/S** : 进程状态
 - **O** - 就绪, 等待被调度
 - **R** - 运行, Linux下没有O状态, 就绪状态也用R表示
 - **S** - 可唤醒睡眠, 系统中断, 获得资源, 收到信号, 都可被唤醒, 转入运行状态
 - **D** - 不可唤醒睡眠, 只能被wake_up系统调用唤醒
 - **T** - 暂停, 收到SIGSTOP信号转入暂停状态, 收到SIGCONT信号转入运行状态
 - **W** - 等待内存分页(2.6内核以后被废弃)



进程快照 (续5)

- 进程信息列表
 - **STAT/S** : 进程状态
 - **x** - 死亡, 不可见
 - **z** - 僵尸, 已停止运行, 但其父进程尚未获取其状态
 - **<** - 高优先级
 - **N** - 低优先级
 - **L** - 有被锁到内存中的分页, 实时进程和定制IO
 - **s** - 会话首进程
 - **l** - 多线程化的进程
 - **+** - 在前台进程组中



进程快照 (续6)

- 进程信息列表

- **START/STIME** : 进程开始时间
- **TIME** : 进程运行时间
- **COMMAND/CMD** : 进程启动命令
- **F** : 进程标志, 由下列值位或
 - 1 - 通过fork产生但是没有exec
 - 4 - 拥有超级用户特权
- **PPID** : 父进程ID
- **NI** : 进程nice值, -20到19
- **PRI** : 进程动态优先级
 - 静态优先级=80+nice, 60到99, 值越小优先级越高, 内核在静态优先级的基础上, 根据进程的交互性计算出它的动态优先级, 体现对IO消耗型进程的奖励和对处理机消耗型进程的惩罚



进程快照 (续7)

- 进程信息列表
 - ADDR : 内核进程的内存地址, 普通进程显示 “-”
 - SZ : 占用虚拟内存页数
 - WCHAN : 进程正在等待的内核函数或事件
 - PSR : 进程当前被指派给哪个处理器运行



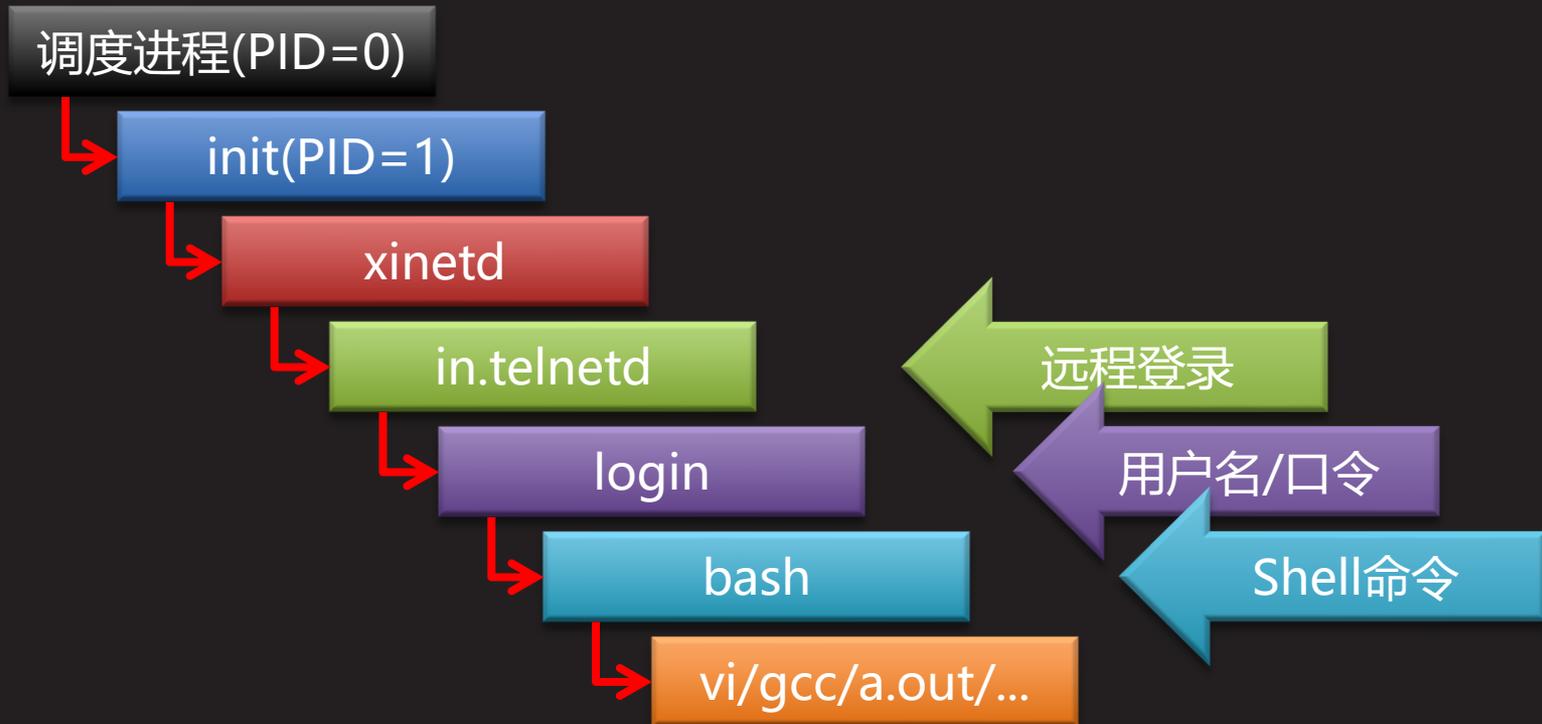
父子孤尸



父子进程

- Unix/Linux系统的中进程存在父子关系。一个父进程可以创建多个子进程，但每个子进程最多只能有一个父进程。整个系统中只有一个根进程，即PID为0的调度进程。系统中的所有进程构成了一棵以调度进程为根的进程树

知识讲解



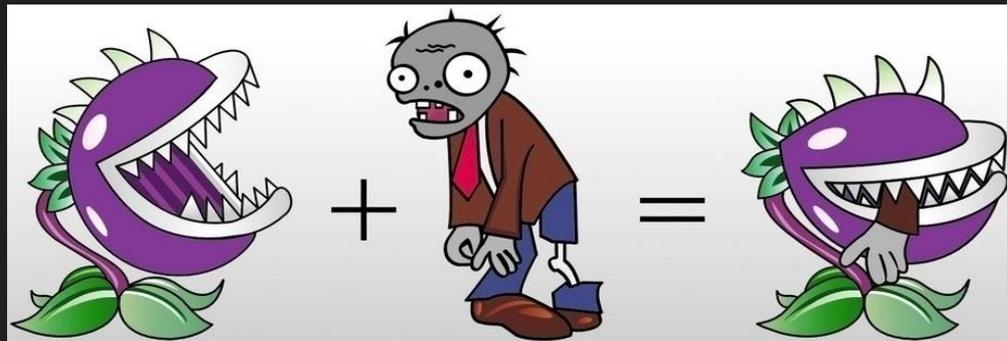
孤儿进程

- 父进程创建子进程以后，子进程在操作系统的调度下与其父进程同时运行
- 如果父进程先于子进程终止，子进程即成为孤儿进程，同时被init进程收养，即成为init进程的子进程，因此init进程又被称为孤儿院进程
- 一个进程成为孤儿进程是正常的，系统中的很多守护进程都是孤儿进程



僵尸进程

- 如果子进程先于父进程终止，但父进程由于某种原因，没有回收子进程的退出状态，子进程即成为僵尸进程
- 僵尸进程虽然已经不再活动，但其终止状态仍然保留，也会占用系统资源，直到被其父进程回收才得以释放
- 如果父进程直到终止都未回收它的已成僵尸的子进程，init进程会立即收养并回收这些处于僵尸状态的子进程，因此一个进程不可能既是孤儿进程同时又是僵尸进程
- 一个进程成为僵尸进程需要引起注意，如果它的父进程长期运行而不终止，僵尸进程所占用的资源将长期得不到释放



进程的各种ID



进程ID



PID和PPID

- 每个进程都有一个非负整数形式的唯一编号，即PID (Process Identification, 进程标识)
- PID在任何时刻都是唯一的，但是可以重用，当进程终止并被回收以后，其PID就可以为其它进程所用
- 进程的PID由系统内核根据延迟重用算法生成，以确保新进程的PID不同于最近终止进程的PID



PID和PPID (续1)

- 系统中有些PID是专用的，比如
 - 0号进程，调度进程
 - 亦称交换进程(swapper)，系统内核的一部分，所有进程的根进程，磁盘上没有它的可执行程序文件
 - 1号进程，init进程
 - 在系统自举过程结束时由调度进程创建
 - 读写与系统有关的初始化文件，引导系统至一个特定状态，收养系统中的孤儿进程
 - 以超级用户特权运行的普通进程，永不终止
 - 2号进程，页守护进程
 - 负责虚拟内存系统的分页操作
- 除调度进程以外，系统中的每个进程都有唯一的父进程，对任何一个子进程而言，其父进程的PID即是它的PPID



PID和PPID (续2)

- 获取调用进程的PID

```
#include <unistd.h>  
  
pid_t getpid (void);
```

返回调用进程的PID

- 获取调用进程的PPID

```
#include <unistd.h>  
  
pid_t getppid (void);
```

返回调用进程的父进程的PID



用户ID和组ID

实际用户ID和实际组ID

- 当一个用户通过合法的用户名和口令登录系统以后，系统就会为他启动一个Shell进程，Shell进程的实际用户ID和实际组ID就是该登录用户的用户ID和组ID。该用户在Shell下启动的任何进程都是Shell进程的子进程，自然也就继承了Shell进程的实际用户ID和实际组ID
- 获取调用进程的实际用户ID和实际组ID

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

```
uid_t getuid (void);  
uid_t getgid (void);
```

分别返回调用进程的实际用户ID和实际组ID



有效用户ID和有效组ID

- 一个进程的用户和组身份决定了它可以访问哪些资源，比如读、写或者执行某个文件。但真正被用于权限验证的并不是进程的实际用户ID和实际组ID，而是其有效用户ID和有效组ID。一般情况下，进程的有效用户ID和有效组ID就取自其实际用户ID和实际组ID，二者是等价的
- 获取调用进程的有效用户ID和有效组ID

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

```
uid_t geteuid (void);
```

```
uid_t getegid (void);
```

分别返回调用进程的有效用户ID和有效组ID



设置用户ID和设置组ID

- 如果用于启动进程的可执行文件带有设置用户ID位和(或)设置组ID位, 那么该进程的有效用户ID和(或)有效组ID就不再取自其实际用户ID和(或)实际组ID, 而是取自可执行文件的拥有者用户ID和(或)组ID

```
mode_t st_mode; // unsigned int
```



1			S_ISUID	设置用户ID
	1		S_ISGID	设置组ID
		1	S_ISVTX	粘滞

4 2 1



设置用户ID和设置组ID (续1)

- 例如
 - 假设ids文件的拥有者用户和组都是root, 且其它用户对该文件有可执行权限

```
-rwxr-xr-x 1 root root ... ids
```

- 隶属于tarena组(GID=1000)的tarena用户(UID=1000)登录系统, 运行ids程序

```
进程ID: ...  
父进程ID: ...  
实际用户ID: 1000  
实际组ID: 1000  
有效用户ID: 1000  
有效组ID: 1000
```



设置用户ID和设置组ID (续2)

- 例如
 - 可以看到进程的有效用户ID和实际用户ID一样都是1000, 而有效组ID也和实际组ID一样都是1000
 - 现在root用户为ids文件添加设置用户ID和设置组ID权限位

```
# chmod u+s ids  
# chmod g+s ids
```

```
-rwsr-sr-x 1 root root ... ids
```



设置用户ID和设置组ID (续3)

- 例如
 - 隶属于tarena组的tarena用户再次运行ids程序

```
进程ID: ...  
父进程ID: ...  
实际用户ID: 1000  
实际组ID: 1000  
有效用户ID: 0  
有效组ID: 0
```

- 不难发现，进程的实际用户ID和实际组ID并没有发生变化，仍然是1000，但它的有效用户ID和有效组ID却变成了0，显然这是ids文件的拥有者root用户的用户ID的组ID，而参与权限判断，决定该进程能做什么不能做什么的恰恰是它的有效用户ID和有效组ID，tarena用户扮演root用户行权



进程的各种ID

【参见：pid.c】

- 进程的各种ID



总结和答疑

