相关概念

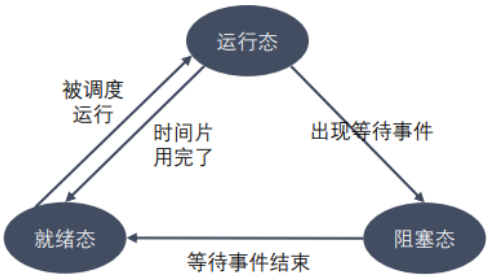
1. 程序：是指编译过的，可执行的二进制代码，保存在存储介质上，不运行。规模很大的二进制程序集可以称为应用。
2. 进程：是指正在运行的程序。进程包括二进制镜像，加载到内存中，还涉及到其他各个方面：虚拟内存实例、内核资源如打开的文件，以及一个或者多个线程。
3. 线程：线程是进程内的活动单元。每个线程有包含自己的虚拟存储器，包扩栈、近程状态等。
4. 线程和进程：在单线程的进程中，进程即线程。一个近程只有一个虚拟内存实例，一个虚拟处理器。在多线程的进程中，一个进程有多个线程。由于虚拟内存是和进程关联的，所以所有进程会共享相同的内存地址空间。但是进程与进程之间确实有单独的虚拟内存空间。
5. 并行：并行是指在同一个时刻，有多条指令再多个处理器上同时执行。要求计算机是有多个cpu。
6. 并发：是指在同一个时刻只有一条指令在执行，但是多个进程指令被快速的轮换执行，使得宏观上具有多个进程在同时执行的效果，但是在微观上并不是同时执行的，只是根据时间片策略快速轮换执行。
7. 进程控制块PCB
   在虚拟地址空间中，内核区有一块用于进程管理的区域。为了管理进程，内核必须对每个进程所做的事情进行清楚的描述。内核为每个进程分配一个 PCB(Processing Control Block)进程控制块，维护进程相关的信息，Linux 内核的进程控制块是 task_struct 结构体。主要有一下几部分：

> 进程id：系统中每个进程有唯一的 id，用 pid_t 类型表示，其实就是一个非负整数
> 进程的状态：有就绪、运行、挂起、停止等状态
> 进程切换时需要保存和恢复的一些CPU寄存器
> 描述虚拟地址空间的信息
> 描述控制终端的信息
> 当前工作目录（Current Working Directory）
> umask 掩码
> 文件描述符表，包含很多指向 file 结构体的指针
> 和信号相关的信息
> 用户 id 和组 id
> 会话（Session）和进程组
> 进程可以使用的资源上限（Resource Limit）

进程状态及其转换

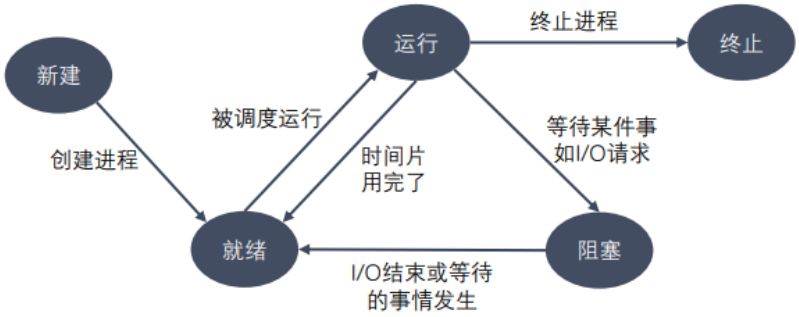
进程状态反映进程执行过程的变化。这些状态随着进程的执行和外界条件的变化而转换。
在三态模型中，进程状态分为三个基本状态，即**就绪态，运行态，阻塞态。**在五态模型中，进程分为**新建态、就绪态，运行态，阻塞态，终止态。**

三态进程



- 运行态：进程占有处理器正在运行
- 就绪态：进程具备运行条件，等待系统分配处理器以便运行。当进程已分配到除CPU以外的所有必要资源后，只要再获得CPU，便可立即执行。在一个系统中处于就绪状态的进程可能有多个，通常将它们排成一个队列，称为就绪队列
- 阻塞态：又称为等待(wait)态或睡眠(sleep)态，指进程不具备运行条件，正在等待某个事件的完成

五态进程



- 新建态：进程刚被创建时的状态，尚未进入就绪队列
- 终止态：进程完成任务到达正常结束点，或出现无法克服的错误而异常终止，或被操作系统及有终止权的进程所终止时所处的状态。进入终止态的进程以后不再执行，但依然保留在操作系统中等待善后。一旦其他进程完成了对终止态进程的信息抽取之后，操作系统将删除该进程。

进程相关命令

查看进程ps

命令是ps，是progress status的缩写，添加其他参数可以实现更精细的功能。用来列出系统中 当前正在运行 的那些进程，就是执行 ps 命令的那个时刻的那些进程的快照。

```
ps -aux  或者 ps -ajx
```

```
ps -f  #仅显示当前终端的相关进程，且按照一定格式输出
```

参数:
a --> 显示终端上的所有进程，包括其他用户的进程。 (all)
u --> 显示进程的详细信息
x --> 显示没有控制终端的进程
j --> 列出与作业控制相关的信息
f --> 按照格式输出
可以通过`man ps`查看这个命令的详细信息。

1、查到的进程有多个列，其中一列是STAT，具体值得意义如下：

```
STAT参数意义：
D 不可中断 Uninterruptible（usually IO）
R 正在运行，或在队列中的进程
S(大写) 处于休眠状态
T 停止或被追踪
Z 僵尸进程
W 进入内存交换（从内核2.6开始无效）
X 死掉的进程
< 高优先级
N 低优先级
s 包含子进程
+ 位于前台的进程组
```

2、USER表示进程所属的用户，PID是进程的ID（都是大于0的整数），TTY是线程的开启位置，比如在一个终端上通过./test执行一个程序，那么这个进程的TTY就是这个终端，pts/0 。

实时显示进程状态top

命令top，这个命令可以实时刷新显示的进程状态信息，而ps只是当时的一种进程状态的快照。
可以在使用 top 命令时加上 -d 来指定显示信息更新的时间间隔(默认也有一定的刷新间隔时间)；在 top 命令执行后，可以按以下按键对显示的结果进行排序：
- M 根据内存使用量排序
- P 根据 CPU 占有率排序
- T 根据进程运行时间长短排序
- U 根据用户名来筛选进程
- K 输入指定的 PID 杀死进程

```
top -d6  #6秒刷新
```

杀死进程kill

kill命令的本意是发送一个信号到进程。默认情况下，该信号是用来杀死进程的。

```
kill [-signal] pid
```

通过`kill -l`可以查看有哪些信号。其中的9号信号就是杀死进程的信号。

```
kiko@Akiko: $ kiko@Akiko: $ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```


例子：
1、通过指定9号信号杀死进程

```
kill -9 68960    #指定9号杀死进程的信号，杀死进程号为68960的进程
kill -SIGKILL 69155  #直接指定9号信号的名字也可以
kill 68960       #可选参数不写，默认就是杀死进程
```

2、杀死所有可以杀死的进程

```
kill -9 -1
```

3、一个类似的命令killall
killall命令用来杀死指定的进程程名的进程

```
killall xxx
```

进程的认知

- 进程pid
  每个进程都有一个唯一的标识符表示，也就是进程的ID，简称pid。系统保证在任意时刻进程pid都是唯一的，但是可以重用。从本质上来讲，大多数程序会假定内核不会重用已用过的pid值，这的确是真的。比如杀死一个最近创建的进程，然后再创建一个进程，可以看到新创建的pid并不是刚才杀死的进程pid，而是比那个pid还要大的数值。
- 空闲进程
  当没有其他进程在运行时，内核所运行的进程是pid = 0的进程。
- init进程
  系统启动后，内核运行的进程是pid = 1的进程，代表了系统中运行的第一个进程。

```
kiko@Hkiko: $ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 168952 13068 ?        Ss   6月09   0:34 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    6月09   0:00 [kthreadd]
```


- pid进程号
  可以使用的进程pid是一个大于0的数。从编程的角度看，pid的类型是pid_t，本质是一个int类型，因此是一个有符号的整数。但是进程本身的pid不可能小于0。

进程体系

- 父进程

创建进程的那个进程是父进程。除了init进程，每个进程都是有其他进程创建的，因此每个进程都有一个父进程。父进程号为ppid。
比如在一个终端上运行一个可执行程序，此时这个可执行程序就是新的进程，而他的父进程就是当前这个终端进程。