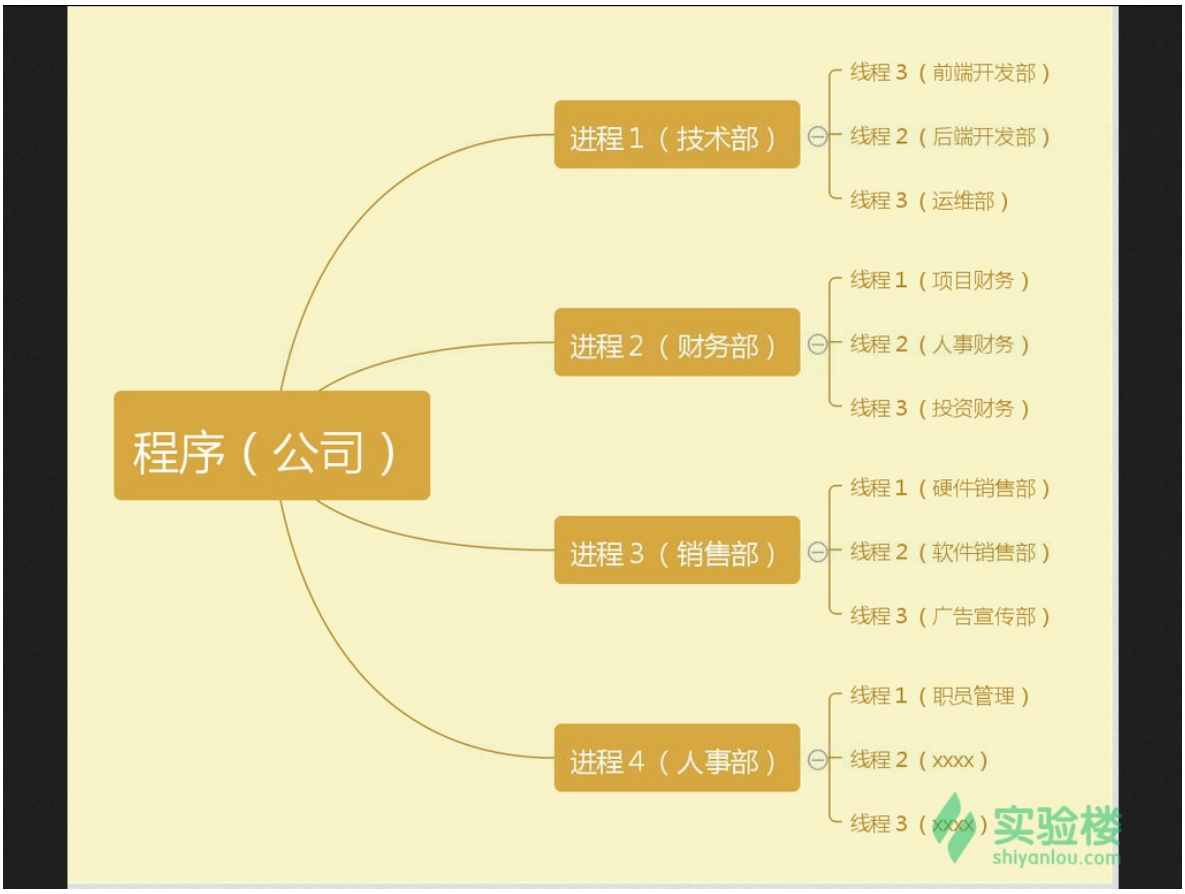
process 概念

1 特性

- 动态性：进程的实质是进程实体的一次执行的过程，有创建，撤销等状态的变化。而程序是一个静态的实体
- 并发性：进程可以做到在一个时间段内，有多个程序在运行中。程序只是静态的实体，所以不存在并发性
- 独立性：进程可以独立分配资源，独立接受调度，独立的运行。
- 异步性：进程以不可预知的速度向前推进。
- 结构性：进程拥有代码段、数据段、PCB（进程控制块，进程存在的唯一标志）。也这是进程的结构性才可以做到独立的运行

> **并发**在一个时间段内，宏观来看有多个程序都在活动，有条不紊的执行
>
> **并行**在每一个瞬间，都有多个程序都在同时执行，这个必须有多个 CPU 才行
>
> 
>

2 进程分类

- 第一个角度来看，我们可以分为用户进程与系统进程
- 第二角度来看，我们可以将进程分为交互进程、批处理进程、守护进程

> **交互进程**：由一个 shell 终端启动的进程，在执行过程中，需要与用户进行交互操作，可以运行于前台，也可以运行在后台。
>
> ---
>
> **批处理进程**：该进程是一个进程集合，负责按顺序启动其他的进程
>
> ---
>
> **守护进程**：守护进程是一直运行的一种进程，经常在 Linux 系统启动时启动，在系统关闭时终止。它们独立于控制终端并且周期性的执行某种任务或等待处理某些发生的事件。例如httpd进程，一直处于运行状态，等待用户的访问。还有经常用的 cron（在 centOS 系列为 crond）进程，这个进程为 crontab 的守护进程，可以周期性的执行用户设定的某些任务。

2.1 进程的衍生

比如我们启动了终端，就是启动了一个 bash 进程，我们可以在 bash 中再输入 bash 则会再启动一个 bash 的进程。一般称呼第一个 bash 进程是第二 bash 进程的父进程，第二 bash 进程是第一个 bash 进程的子进程

> 父进程与子进程便会提及这两个**系统调用 fork() 与 exec()**
>
> ---
>
> fork() 是一个系统调用（system call），它的主要作用就是**为当前的进程创建一个新的进程**，这个新的进程就是它的子进程，这个子进程除了返回值和 PID 父进程以外其他的都一模一样，如进程的执行代码段，内存信息，文件描述，寄存器状态等等
>
> ---
>
> exec() 也是系统调用，作用是**切换子进程中的执行程序**

2.2 child process exit–>僵尸进程（Zombie）

> 进程结束时它的主函数 main() 会执行 exit(n); 或者 return n，这里的返回值 n 是一个信号，系统会把这个 SIGCHLD 信号传给其父进程
>
> ---
>
> 这个时候的子进程代码执行部分其实已经结束执行了，系统的资源也基本归还给系统了，但是其进程的**进程控制块（PCB）仍驻留在内存中**，而它的 PCB 还在，代表这个进程还存在（因为 PCB 就是进程存在的唯一标志，里面有 PID 等消息），并没有消亡，这样的进程称之为**僵尸进程（Zombie）**
>
> ---
>
> 正常情况下，**父进程会收到两个返回值一个是 exit code** 也是 SIGCHLD 信号与 **reason for termination** 之后，父进程会**使用 wait(&status) 系统调用以获取子进程的退出状态**，然后内核就可以从内存释放已结束的子进程的 PCB

Linux 系统中能使用的 PID 是有限的，如果系统中存在有大量的僵尸进程，系统将会因为没有可用的 PID 从而导致不能产生新的进程。

2.3 father process exit–>孤儿进程

另外如果**父进程非正常的结束**，**未能及时收回子进程**，子进程仍在运行，这样的子进程**称之为孤儿进程。**
在 Linux 系统中，孤儿进程一般会被 init 进程所“收养”，成为 init 的子进程。由 init 来做善后处理，所以它并不至于像僵尸进程那样无人问津，不管不顾，**大量存在会有危害。**

2.4 进程 0 和进程1(init process)

是系统引导时创建的一个特殊进程，也称之为**内核初始化**，其**最后一个动作就是**调用 fork() 创建出一个子进程运行 /sbin/init 可执行文件,而该进程就是 **PID=1 的进程1，也就是 init 进程**
而**进程 0 就转为交换进程**（也被称为空闲进程），而**进程 1**（init 进程）是第一个普通用户态的进程,再由它 **不断调用 fork() 来创建系统里其他的进程**，所以它**是所有进程的父进程**或者祖先进程。**同时是一个守护程序**，直到计算机关机才会停止。

2.5 进程组与 Sessions

- process是process group中的一员
- 每个process group依靠PGID(process group ID)区分
- 进程组的 PGID = 进程组的第一个成员的 PID(leader process)
- process 通过 getpgrp() 系统调用来寻找其所在组的 PGID
- leader process可以先终结，不影响process group
- process group 又是session的一员

> **session** 主要是针对一个 tty 建立，Session 中的**每个进程都称为一个工作(job)。Session 意义在于将多个jobs囊括在一个终端，并取其中的一个 job 作为前台，来直接接收该终端的输入输出以及终端信号。其他jobs在后台运行。**