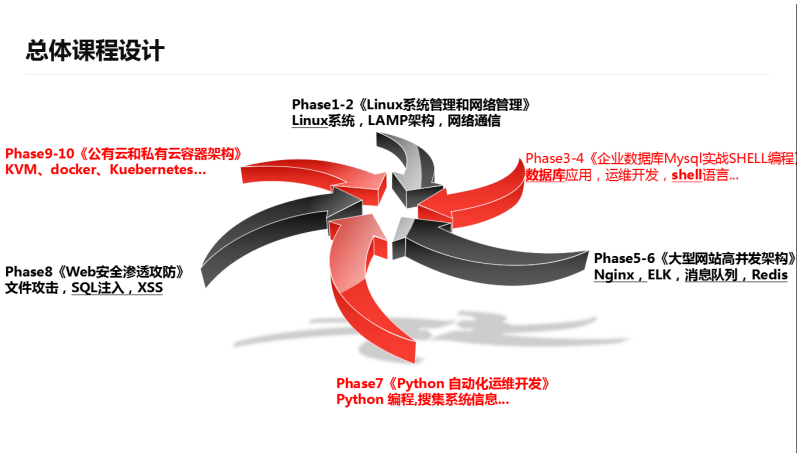
2023年 Linux云计算系统运维架构师全套实战课程（小白到就业）

课程总体介绍

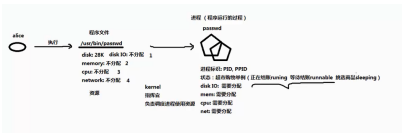


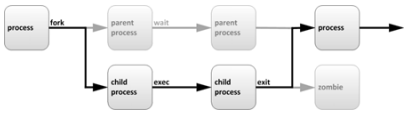
第一阶段总体介绍

第一阶段：Linux 系统及服务管理

第04章_Linux系统进程管理

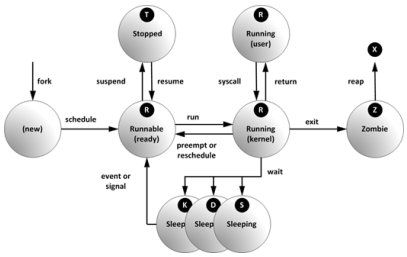
进程简介

- 灵魂三问
  - 我是谁?
    - 进程是什么?
  - 我从哪里来
    - 进程从哪里来?
  - 我要到哪里去?
    - 进程要到哪里去?
- 进程三问
  - 什么是进程？（了解）
    - 进程是已启动的可执行程序的运行实例，进程有以下组成部分：
      • 已分配内存的地址空间；
      • 安全属性，包括所有权凭据和特权；
      • 程序代码的一个或多个执行线程；
      • 进程状态。
      程序： 二进制文件，静态 /usr/bin/passwd ,/usr/sbin/useradd
      进程： 是程序运行的过程， 动态，有生命周期及运行状态。
    - 手稿
      -
      

    - 进程的生命周期（了解）
      -
      

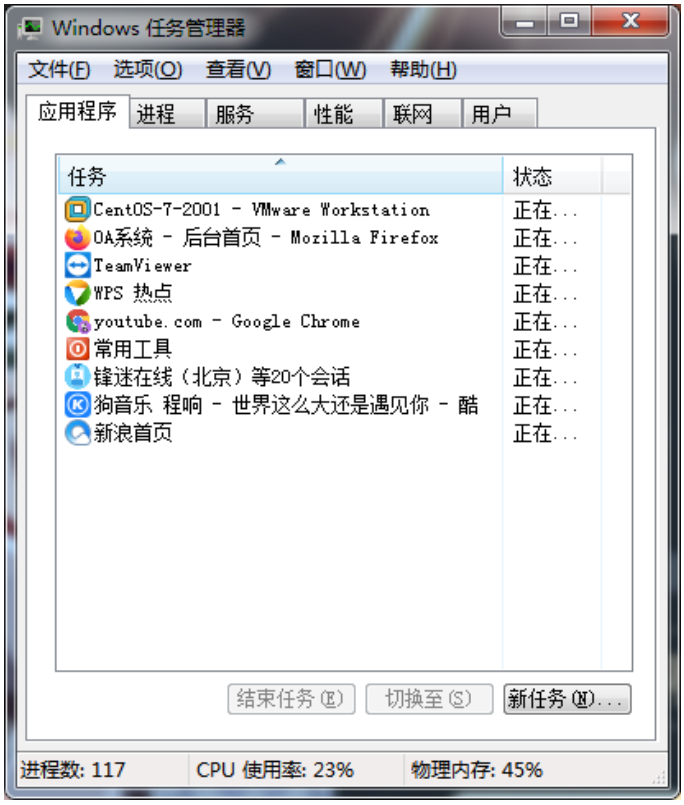
      - 父进程复制自己的地址空间（fork）创建一个新的（子）进程结构。
      每个新进程分配一个，唯一的进程 ID （PID），满足跟踪安全性之需。
      任何进程都可以创建子进程。
      所有进程都是第一个系统进程的后代：
      - Centos5/6系统进程: init
      Centos7系统进程: systemd
      Centos9系统进程：/usr/lib/systemd/systemd
    - 进程状态（了解）
      - 进程状态产生的原因

- 在多任务处理操作系统中，每个CPU（或核心）
  在一个时间点上只能处理一个进程。
  在进程运行时，它对CPU 时间和资源分配的要求会不断变化，
  从而为进程分配一个状态，它随着环境要求而改变。



Name	Flag	Kernel-defined state name and description
Running	R	TASK_RUNNING: The process is either executing on a CPU or waiting to run. Process can be executing user routines or kernel routines (system calls), or be queued and ready when in the *Running* (or *Runnable*) state.
Sleeping	S	TASK_INTERRUPTIBLE: The process is waiting for some condition: a hardware request, system resource access, or signal. When an event or signal satisfies the condition, the process returns to *Running*.
	D	TASK_UNINTERRUPTIBLE: This process is also *Sleeping*, but unlike **S** state, will not respond to delivered signals. Used only under specific conditions in which process interruption may cause an unpredictable device state.
	K	TASK_KILLABLE: Identical to the uninterruptible **D** state, but modified to allow the waiting task to respond to a signal to be killed (exited completely). Utilities frequently display *Killable* processes as **D** state.
Stopped	T	TASK_STOPPED: The process has been *Stopped* (suspended), usually by being signaled by a user or another process. The process can be continued (resumed) by another signal to return to *Running*.
	t	TASK_TRACED: A process that is being debugged is also temporarily *Stopped* and shares the same **T** state flag.
Zombie	Z	EXIT_ZOMBIE: A child process signals its parent as it exits. All resources except for the process identity (PID) are released.
	X	EXIT_DEAD: When the parent cleans up (*reaps*) the remaining child process structure, the process is now released completely. This state will never be observed in process-listing utilities.

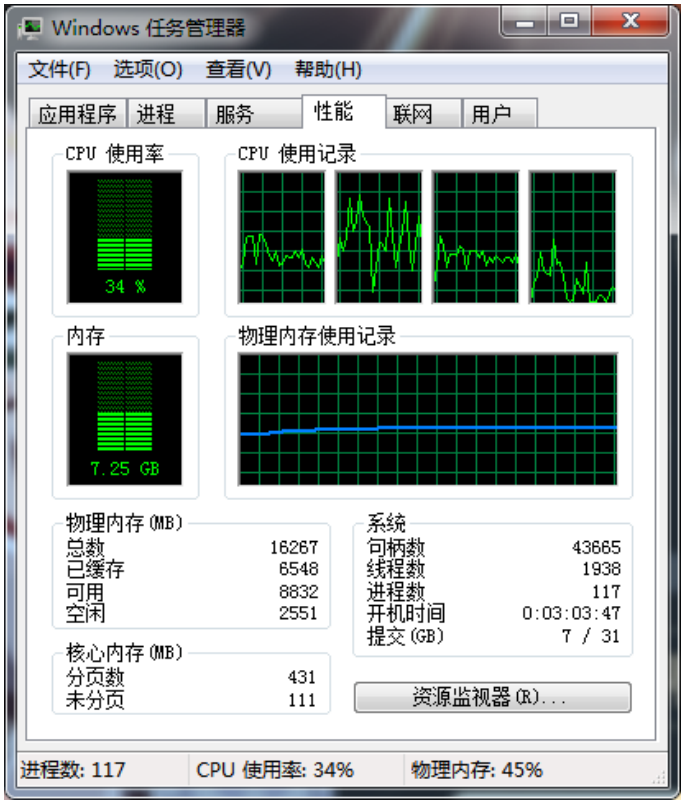
- 进程管理 process(掌握)
  - 目标
    - 了解进程的相关信息：
    - • PID,PPID
    - • 当前的进程状态
    - • 内存的分配情况
    - • CPU和已花费的实际时间
    - • 用户UID，它决定进程的特权
    - • 进程名称
  - 静态查看进程 ps
    - 静态查看进程 ps
      - PS
        - 不是photoshop
        - 是precess status
        - 叫进程 状态
      - 有人认识这个软件吗

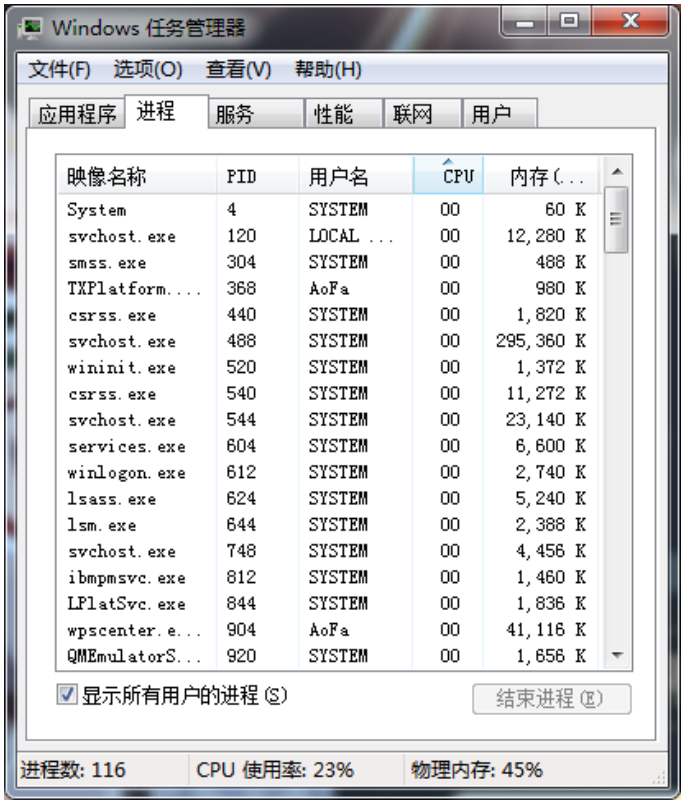


- 一个进程为例
  - [root@localhost ~]# ps aux | head -2
    USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
    root 1 0.0 0.6 128096 6708 ? Ss 16:20 0:01 /usr/lib/systemd/systemd
  - 命令参数说明
    - ps a 显示现行终端机下的所有程序
    - ps u 以用户为主的格式来显示程序状况。
    - ps x 不以终端机来区分。
- ps aux 输出的字段含义
  - USER: 运行进程的用户
  - PID： 进程ID
    - 我们云工程师靠PID，杀死他
  - %CPU: CPU占用率
  - %MEM: 内存占用率
  - VSZ： 占用虚拟内存
  - RSS: 占用实际内存

- TTY： 进程运行的终端
- STAT： 进程状态
  - [常见]
    - R 运行
    - S 睡眠 Sleep
    - T 停止的进程
    - Z 僵尸进程
    - X 死掉的进程
  - START: 进程的启动时间
  - TIME： 进程占用CPU的总时间
    - 分钟：秒
  - COMMAND： 进程文件，进程名
- 进程排序
  - 语法
    - ps aux --sort %cpu
  - 示例
    - 以CPU占比降序排列（减号是降序）
      [root@localhost ~]# ps aux --sort -%cpu
      [root@localhost ~]# ps aux --sort %cpu
- 进程的父子关系
  - 语法
    - ps -ef
  - 示例
    - 查看进程的父子关系。 请观察PID和PPID
      [root@localhost ~]# ps -ef
      UID PID PPID C STIME TTY TIME CMD
      root 1 0 0 1月22 ? 00:00:07 /usr/lib/systemd/systemd
      root 2 0 0 1月22 ? 00:00:00 [kthreadd]
      root 3 2 0 1月22 ? 00:00:06 [ksoftirqd/0]
- 自定义显示字段(了解)
  - 语法
    - ps axo 自定义字段

- 示例
  - [root@localhost ~]# ps axo user,pid,ppid,%mem,command |head -3
    root 8310 1 0.1 /usr/sbin/httpd
    apache 8311 8310 0.0 /usr/sbin/httpd
    apache 8312 8310 0.0 /usr/sbin/httpd
- 动态查看进程 top
  - 有人认识这个软件吗
    -





- 上半部分
  - 示例
    - top - 11:45:08 up 18:54, 4 users, load average: 0.05, 0.05, 0.05
      Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie
      %Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
      KiB Mem : 3865520 total, 1100000 free, 580268 used, 2185252 buff/cache
      KiB Swap: 4063228 total, 4063228 free, 0 used. 2917828 avail Mem
  - 说明
    - 第1行
      - top - 11:45:08 up 18:54, 4 users, load average: 0.05, 0.05, 0.05
        程序名-系统时间 运行时间 登录用户数 CPU 负载 1 5 15 分钟
    - 第2行
      - Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie
        总进程数 运行数 1 睡眠数 175 停止数 0 僵死数 0

- 第3行
  - %Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si 0.0 st
    CPU 使用占比 us 用户 sy 系统 ni 优先级 id 空闲 wa 等待 hi 硬件 si 软件 st 虚拟机
- 第4行
  - KiB Mem : 3865520 total, 1100000 free, 580268 used, 2185252 buff/cache
    物理内存 K total 总共 4G free 空闲 1G userd 使用 500M cache 缓存硬盘内容 2G
- 第5行
  - KiB Swap: 4063228 total, 4063228 free, 0 used. 2917828 avail Mem
    交换分区 total 总共 4G free 空闲 4G used 使用 0 avail 下次可用的空间 3G
- 下半部分
  - 字段介绍（了解）
    - PID,USER,%CPU,%MEM略
    - VIRT：virtual memory usage 虚拟内存
      - 需要这些内存，但并没有占满。
    - RES：resident memory usage 常驻内存
      - 用了多少内存
    - SHR：shared memory 共享内存
      - 1、除了自身进程的共享内存，也包括其他进程的共享内存
      - 2、共享内存大小公式：RES – SHR
  - top常用内部指令
    - h|?帮助
      M 按内存的使用排序
      P 按CPU使用排序
      N 以PID的大小排序

      < 向前
      > 向后
      z 彩色，Z设置彩色，使用数字调整
- top技巧

- 动态查看进程 top，像windows的任务管理器
  [root@localhost ~]# top　　//回车，立刻刷新。按z彩色显示，按F，通过光标设置列的顺序。
  [root@localhost ~]# top -d 1　//每1秒刷新。
  [root@localhost ~]# top -d 1 -p 10126 查看指定进程的动态信息
  [root@localhost ~]# top -d 1 -p 10126,1　查看10126和1号进程

- 使用信号控制进程kill
  - 信号种类
    - 给进程发送信号(kill -l列出所有支持的信号)
      [root@localhost ~]# kill -l
      编号 信号名
      1) SIGHUP 重新加载配置
      2) SIGINT 键盘中断Ctrl+C
      3) SIGQUIT 键盘退出Ctrl+\，类似SIGINT
      9) SIGKILL 强制终止，无条件
      15) SIGTERM 终止（正常结束），缺省信号
      18) SIGCONT 继续
      19) SIGSTOP 暂停
      20)SIGTSTP 键盘暂停Ctrl+Z
  - 信号9,15
    - 1 创建2个文件，查看终端号。
      [root@localhost ~]# touch file1 file2

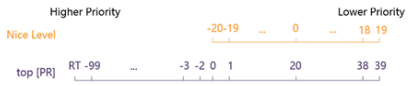
      2 通过一个终端，打开一个vim
      [root@localhost ~]# vim file1

      3 通过另一个终端，打开一个vim
      [root@localhost ~]# vim file2

      3 通过另一个终端，查询两个进程。
      [root@localhost ~]# ps aux |grep vim
      root 4362 0.0 0.2 11104 2888 pts/1 S+ 23:02 0:00 vim file1
      root 4363 0.1 0.2 11068 2948 pts/2 S+ 23:02 0:00 vim file2

      4 发送信号15 和信号9，观察两个终端程序状态。
      [root@localhost ~]# kill -15 4362
      [root@localhost ~]# kill -9 4363
      观察两个终端，一个正常终止，一个非法杀死。
- 进程优先级nice
  - 简介

- Linux 进程调度及多任务
  每个CPU在一个时间点上只能处理一个进程，通过时间片技术，来同时运行多个程序。

- 优先级范围和特性
  - 优先级图示
    - 

    - 系统中的两种优先级
      - 在top中显示的优先级有两个，PR值和nice值

        NI: 实际nice值
        PR（+20）: 将nice级别显示为映射到更大优先级队列，-20映射到0，+19映射到39
  - 优先级特性
    - nice 值越大： 表示优先级越低，例如+19
      nice 值越小： 表示优先级越高，例如-20

- 查看进程的nice级别
  - [root@localhost ~]# ps axo pid,command,nice --sort=-nice

- 启动具有不同nice级别的进程
  - 示例
    - 默认情况
      - 启动进程时，通常会继承父进程的 nice级别，默认为0。
    - 手动启动不同nice
      - ```
        [root@localhost ~]# nice -n -5 sleep 6000 &
        [1] 2220
        [root@localhost ~]# nice -n -10 sleep 7000 &
        [2] 2229
        [root@localhost ~]# ps axo command,pid,nice | grep sleep
        sleep 6000                  2220  -5
        sleep 7000                  2229 -10
        grep --color=auto sleep       2233   0
        ```

- 更改现有进程的nice级别
  - 示例
        ```

- 使用shell更改nice级别
  1 创建一个睡眠示例程序。
  [root@localhost ~]# sleep 7000 &
  [2] 2669

  2 修改他的nice值。
  [root@localhost ~]# renice -20 2669
  2669 (进程 ID) 旧优先级为 0，新优先级为 -20，观察修旧的nice值。

- 作业控制 jobs(了解)
  - 简介
    - 作业控制是一个命令行功能，也叫后台运行。
    - 关键词介绍
      - foreground
        - fg
          - 前台进程：是在终端中运行的命令，占领终端。
      - background
        - bg
          - 后台进程：没有控制终端，它不需要终端的交互。看不见，但是在运行。
    - 后台程序控制示例
      - 1.观察占领前台的现象
        - [root@localhost ~]# sleep 2000
        - 运行一个程序，当前终端无法输入。观察占领前台的现象。
          大部分命令行输入已经无效。
        - ctrl + c 终止进程
      - 2.运行后台程序
        - [root@localhost ~]# sleep 3000 &
      - 3.ps查询所有程序。
        - [root@localhost ~]# ps aux |grep sleep
        - root 8895 0.0 0.0 100900 556 pts/0 S 12:13 0:00 sleep 3000
      - 4.jobs查看后台进程。
        - [root@localhost ~]# jobs

- [1]+ Running sleep 3000 &
  - +,-代表，使用fg时，默认调动至前台的进程。先是+，后是-
- 5.调动后台程序至前台。
  - [root@localhost ~]# fg 1 //将作业1调回到前台
- 6.消灭后台进程
  - [root@localhost ~]# kill %1
  - 注意
    - 注意，“kill 1” 和 “kill %1”不同，
      前者终止PID为1的进程，
      后者杀死作业序号为1的后台程序。
- 总结
  - & 后台运行程序
  - jobs 查询后台
  - kill %1 停止后台进程

- 虚拟文件系统 proc （了解）
  - 简介
    - 虚拟文件系统：采集服务器自身 内核、进程运行的状态信息
  - CPU
    - /proc/cpuinfo
      - [root@localhost ~]# cat /proc/cpuinfo
  - 内存
    - /proc/meminfo
      - [root@localhost ~]# less /proc/meminfo
  - 内核
    - /proc/cmdline
      - [root@localhost ~]# cat /proc/cmdline