# 11、进程管理

## 一、进程简介

### 1.1 灵魂三问

1. 进程是什么?
   - 程序在启动，运行时(执行命令，运行服务）都会产生进程，需要通过进程 来完成程序的工作
   - 包含：内存地址空间、安全属性、执行线程、进程状态
   - 程序 vs 进程：
     - 程序：二进制文件（静态）如 `/usr/bin/passwd`
     - 进程：程序运行过程（动态），有生命周期及状态
2. 进程来源：
   - 父进程通过 `fork` 创建子进程
   - 所有进程的祖先进程：
     - CentOS5/6: `init`
     - CentOS7/8: `systemd`
3. 进程状态：
   - 运行状态产生原因：CPU时间片轮转
   - 状态类型：
     - R (运行) | S (睡眠) | T (停止)
     - Z (僵尸) | X (死亡)

### 1.2 进程和pid(Process ID)

在Linux系统中，**PID（Process ID）** 是 **进程标识符（Process Identifier）** 的缩写。它是一个唯一且非负的整数，用于在操作系统中唯一标识一个正在运行的进程。每个进程（包括程序、服务、后台任务等）启动时都会被分配一个PID，系统通过PID管理和跟踪进程的生命周期。

```
[root@xnha ~]# ps
   PID TTY          TIME CMD
  3582 pts/1    00:00:00 bash
  6761 pts/1    00:00:00 ps           #输第一次ps命令，会产生pid为6761的进程
[root@xnha ~]# ps
   PID TTY          TIME CMD
  3582 pts/1    00:00:00 bash
  6768 pts/1    00:00:00 ps           #第二次，6768
[root@xnha ~]#
```

也有些服务类的会产生守护进程(也就是一直在运行中的进程，除非关闭了)

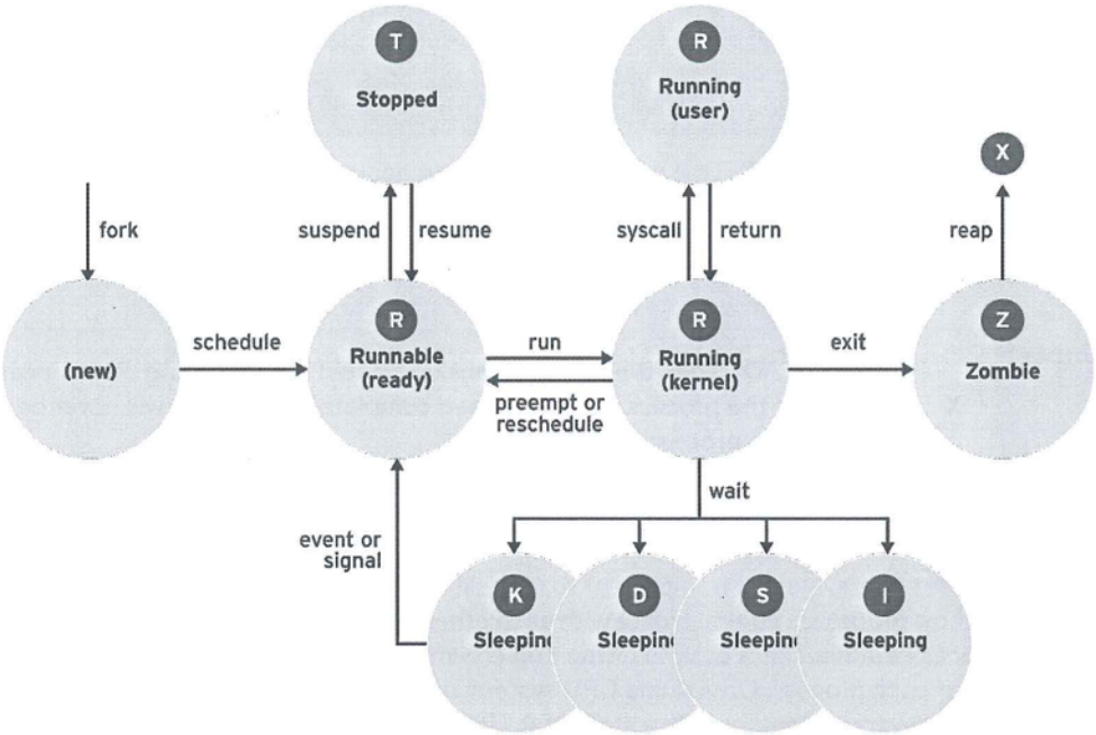
```
[root@xnha ~]# ps -e |grep sshd
  1078 ?        00:00:00 sshd
```

进程还可能会产生子进程。一个父进程复制自己(fork)来创建一个新的子进程。每个进程分配一个唯一的标识(pid)，其父进程的标识为ppid. 在 centos8,centos7中第一个第1个进程为systemd,在centos6及以前版本第1 个进程为init.

```
[root@xnha ~]# pstree    #pstree命令可以查看所有进程的关系(树状结构显示)
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─VGAuthService
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─atd
        ...
```

## 进程的状态

进程从产生到关闭有很多种状态




## 状态说明

**Linux Process States**

| 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, does not respond to signals. Used only when process interruption may cause an unpredictable device state. |
| | **K** | TASK_KILLABLE: Identical to the uninterruptible **D** state, but modified to allow a waiting task to respond to the signal that it should be killed (exit completely). Utilities frequently display *Killable* processes as **D** state. |

| NAME | FLAG | KERNEL-DEFINED STATE NAME AND DESCRIPTION |
|---|---|---|
| | **I** | TASK_REPORT_IDLE: A subset of state **D**. The kernel does not count these processes when calculating load average. Used for kernel threads. Flags TASK_UNINTERRUPTABLE and TASK_NOLOAD are set. Similar to TASK_KILLABLE, also a subset of state **D**. It accepts fatal signals. |
| ***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. |

## 总结常见的状态

| 进程状态 | 说明 |
| --- | --- |
| R | 运行状态，cpu正在运算此进程或者进程正在等待CPU中。 |
| S | 睡眠状态，等待资源调用等条件满足才会转为运行状态 |
| T | 暂停状态 |
| Z | 僵尸进程是当子进程比父进程先结束，而父进程又没有回收子进程，释放子进程占用的资源，此时子进程将成为一个僵尸进程 |

## PID的核心作用

1. **唯一标识进程**
   每个进程的PID在系统范围内是唯一的（同一时刻不会重复），通过PID可以精准定位到具体的进程。
2. **进程管理**
   系统工具（如 `kill`、`top`、`ps`）和用户通过PID对进程进行控制，例如：
   - 终止进程：`kill -9 PID`
   - 查看进程资源占用：`top -p PID`
   - 调试进程：`gdb -p PID`
3. **进程间通信（IPC）**
   进程间通信（如信号、管道、共享内存）通常需要指定目标进程的PID。

## PID的分配规则

1. **取值范围**
   - PID从 `1` 开始分配（系统启动后的第一个进程是 `init` 或 `systemd`，PID=1）。
   - 最大PID值由内核参数 `/proc/sys/kernel/pid_max` 定义（默认通常为 `32768`）。
   - 当PID达到最大值时，会循环使用已释放的PID（但不会重复分配给仍在运行的进程）。
2. **回收机制**
   进程终止后，其PID会被释放，供后续新进程使用。

## PID的特殊角色

1. **PID=1（Init进程）**
   - 系统启动后的第一个进程，负责启动和管理其他所有用户进程。
   - 如果PID=1的进程终止，系统会触发内核恐慌（Kernel Panic）并崩溃。
2. **父进程PID（PPID）**
   每个进程（除Init）都有父进程（Parent Process），父进程的PID称为 **PPID**。
   - 通过 `ps -ef` 查看PPID：

```
ps -ef | grep <进程名>
# 输出示例：
# UID         PID  PPID  C STIME TTY          TIME CMD
# root       1234     1  0 Aug01 ?        00:00:00 nginx: master process
```

- 第三列（PPID）是父进程的PID（如 `1` 表示由Init进程启动）。

---

# 二、进程管理（ps、pstree、top）

## 2.1 静态查看进程 ps

`ps`（Process Status）是 Linux 中用于查看**当前系统进程状态**的工具。它可以显示进程的详细信息，如进程 ID（PID）、CPU 和内存占用、运行时间、命令行参数等

### 一、基本语法

bash

```
ps [选项]
```

---

### 二、常用选项

#### 1. 显示进程范围

| 选项 | 说明 |
| --- | --- |
| `-e` 或 `-A` | 显示 **所有进程**（包括其他用户进程） |
| `-a` | 显示 **所有终端进程**（除会话领导者） |
| `-x` | 显示 **不关联终端的进程**（如守护进程） |
| `-u <用户>` | 显示指定用户的进程（如 `-u root`） |

**示例**：

```
[root@xnha ~]# ps -e                      # 显示所有进程，按Pid升序排列

  PID TTY          TIME CMD
    1 ?        00:00:02 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 rcu_gp
    4 ?        00:00:00 rcu_par_gp
    6 ?        00:00:00 kworker/0:0H-kblockd
    7 ?        00:00:00 kworker/u256:0-events_unbound
    8 ?        00:00:00 mm_percpu_wq
    9 ?        00:00:00 ksoftirqd/0
   10 ?        00:00:00 rcu_sched
   11 ?        00:00:00 migration/0
   12 ?        00:00:00 watchdog/0
```

```
    13 ?        00:00:00 cpuhp/0
```

**`tty` 字段解析：**

**表示物理控制终端**


在 `ps` 命令的输出中，`TTY` 列表示进程关联的终端设备

- `tty`：进程运行在物理控制台终端。
- `pts/0`：进程运行在伪终端（如图形界面的终端或 SSH 会话）。
- `?`：进程不关联任何终端（如守护进程或内核线程）

```
[root@xnha ~]# ps -a                    #所有由终端创建的进程，不包含bash\shell，不包含守护进程（sshd,nginx等）
   PID TTY          TIME CMD
  1430 tty1     00:00:00 gnome-session-b
  1478 tty1     00:00:06 gnome-shell
  1671 tty1     00:00:00 Xwayland
  1692 tty1     00:00:00 ibus-daemon
  1695 tty1     00:00:00 ibus-dconf
  1698 tty1     00:00:00 ibus-x11
  1746 tty1     00:00:00 gsd-xsettings
  1749 tty1     00:00:00 gsd-a11y-settin
  1752 tty1     00:00:00 gsd-clipboard
  1755 tty1     00:00:00 gsd-color
  1757 tty1     00:00:00 gsd-datetime
  1759 tty1     00:00:00 gsd-housekeepin
  1764 tty1     00:00:00 gsd-keyboard
  1770 tty1     00:00:00 gsd-media-keys
  1776 tty1     00:00:00 gsd-mouse
  1780 tty1     00:00:00 gsd-power
[root@xnha ~]# ps -x                # 守护进程（sshd,nginx)内核线程、系统服务（systemd-journald)、计划任务（cron)、容器（dockerd)
   PID TTY      STAT   TIME COMMAND
     1 ?        Ss     0:02 /usr/lib/systemd/systemd --switched-root --system --
     2 ?        S      0:00 [kthreadd]
     3 ?        I<     0:00 [rcu_gp]
     4 ?        I<     0:00 [rcu_par_gp]
     6 ?        I<     0:00 [kworker/0:0H-kblockd]
     7 ?        I      0:00 [kworker/u256:0-events_unbound]
     8 ?        I<     0:00 [mm_percpu_wq]
     9 ?        S      0:00 [ksoftirqd/0]
    10 ?        I      0:00 [rcu_sched]
    11 ?        S      0:00 [migration/0]
    12 ?        S      0:00 [watchdog/0]
    13 ?        S      0:00 [cpuhp/0]
    14 ?        S      0:00 [cpuhp/1]
    15 ?        S      0:00 [watchdog/1]
    16 ?        S      0:00 [migration/1]
    17 ?        S      0:00 [ksoftirqd/1]
    19 ?        I<     0:00 [kworker/1:0H-kblockd]
```

```
    21 ?        S      0:00 [kdevtmpfs]
    22 ?        I<     0:00 [netns]
    23 ?        S      0:00 [kauditd]


ps -u root # 显示 root 用户的进程
```

`stat`**字段解析**：在 Linux 的 `ps` 命令中，**STAT**（**Process State**）字段表示进程的当前状态。每个状态由 **1个主字母** 和 **可选的修饰符** 组成（如 `R+`、`Ss`）

**一、主状态（单字母）**

| 状态符 | 名称 | 说明 |
| --- | --- | --- |
| `R` | Running / Runnable | 进程正在运行或在运行队列中等待 CPU 时间片（可被调度执行）。 |
| `S` | Interruptible Sleep | 进程处于**可中断睡眠**，等待事件（如用户输入、信号或资源可用）。 |
| `D` | Uninterruptible Sleep | 进程处于**不可中断睡眠**，通常因等待 I/O 操作（如磁盘读写）阻塞，无法被信号唤醒。 |
| `Z` | Zombie | **僵尸进程**，进程已终止，但未被父进程回收资源（需手动清理）。 |
| `T` | Stopped | 进程被**暂停**（如通过 `Ctrl+Z` 或 `kill -STOP`）。 |
| `t` | Tracing Stop | 进程正在被调试器跟踪（如使用 `gdb` 时的断点暂停）。 |
| `X` | Dead (已废弃) | 进程完全终止（非常罕见，通常瞬间消失，`ps` 几乎不显示）。 |
| `I` | Idle (内核线程) | 内核线程的空闲状态（仅在某些系统中显示）。 |

**1.** `R` **（Running / Runnable）运行/可运行状态**

- **通俗解释**：
  就像在银行排队等叫号的人。
  - **正在运行**：当前正在窗口办理业务（实际在用 CPU）。
  - **可运行**：已经取号，坐在椅子上等待叫号（在 CPU 的等待队列中）。
- **实际场景**：
  你用 `top` 命令看到的 CPU 使用率高的进程，通常处于 `R` 状态。

  bash

  ```
  $ ps aux | grep ' R '  # 查看所有运行中的进程
  ```

---

**2.** `S` **（Interruptible Sleep）可中断睡眠**

- **通俗解释**：
  你网购后等快递，但可以边等快递边刷手机。如果快递到了（事件发生），你就去取；如果不想等了（收到信号），可以取消订单。

- **实际场景**：
  - 进程等待用户输入（如 `vim` 未操作时）。
  - 等待网络响应（如浏览器加载网页时）。
  - **唤醒方式**：按回车键、收到信号（如 `Ctrl+C` 终止进程）。

  bash

  ```
  $ sleep 100  # 输入后按 Ctrl+Z，进程进入 S 状态
  ```

---

### 3. `D`（Uninterruptible Sleep）不可中断睡眠

- **通俗解释**：
  你在手术室做手术，医生让你别动。此时即使有人叫你（发信号），你也无法回应，必须等手术完成。
- **实际场景**：
  - 进程等待磁盘 I/O（如写入大文件到 U 盘，突然拔掉 U 盘）。
  - **风险**：进程卡在 `D` 状态无法终止（即使 `kill -9` 也无效）。
  - **解决方案**：重启系统或修复硬件/驱动。

  bash

  ```
  $ ps aux | grep ' D '  # 查找 D 状态进程
  ```

---

### 4. `Z`（Zombie）僵尸进程

- **通俗解释**：
  餐厅吃完饭，服务员（父进程）没来收拾桌子（回收资源），桌子一直占着。
- **实际场景**：
  - 父进程未正确处理子进程的终止状态。
  - **无害性**：僵尸进程不占用 CPU/内存，只占一个 PID。
  - 清理方法：

    bash

    ```
    $ kill -9 父进程PID  # 终止父进程，僵尸进程由 init 回收
    ```

---

### 5. `T`（Stopped）暂停状态

- **通俗解释**：
  你正在看视频，突然按下暂停键，画面定格。
- **实际场景**：
  - 用 `Ctrl+Z` 暂停前台进程（如 `ping` 命令）。
  - 恢复方法：

    bash

```
$ bg  # 让进程后台继续运行（状态变回 S）
$ fg  # 让进程回到前台运行（状态变回 R/S）
```

---

**6. `t`（Tracing Stop）调试暂停**

- **通俗解释**：
  程序员用调试器（如 GDB）一步步执行代码，遇到断点暂停。
- **实际场景**：
  - 用 `gdb` 调试程序时，进程会在断点处暂停。
  - **恢复方法**：在调试器中输入 `continue`。

---

**7. `X`（Dead）死亡状态**

- **通俗解释**：
  人去世后火化，骨灰盒已下葬（进程资源完全释放）。
- **实际场景**：
  - 进程终止后，内核将其标记为 `X`，但 `ps` 几乎不会显示（瞬间消失）。
  - **无需处理**：系统自动回收资源。

---

**8. `I`（Idle）空闲状态（内核线程）**

- **通俗解释**：
  工厂的机器人在没有生产任务时进入待机状态。
- **实际场景**：
  - 内核线程（如 `kworker`）在没有任务时的状态。
  - **无需关注**：属于系统正常行为。

**二、修饰符（附加字母）

| **修饰符** | **说明** |
| --- | --- |
| `<` | 高优先级（**低 nice 值**，CPU 优先级高）。 |
| `N` | 低优先级（**高 nice 值**，CPU 优先级低）。 |
| `L` | 进程有**锁定的内存页**（如用于实时任务）。 |
| `s` | 进程是**会话领导者**（Session Leader），通常是 Shell 进程。 |
| `l` | 进程是**多线程**的（有多个线程）。 |
| `+` | 进程位于**前台进程组**（与终端关联，可接收输入）。 |
| ``（空格） | 进程无特殊修饰符。 |

**三、常见状态组合示例**

| **STAT** | **解释** |
| --- | --- |
| `R+` | 进程正在运行，且位于前台进程组。 |
| `Ss` | 会话领导者（如 Shell），处于可中断睡眠。 |
| `D<` | 进程处于不可中断睡眠，且具有高优先级。 |
| `Z` | 僵尸进程，需父进程回收资源。 |

```
| `Sl` | 进程处于可中断睡眠，且是多线程的。 |
| `Tl` | 进程被暂停，且是多线程的。 |

##### 2. **显示完整信息**

| 选项 | 说明 |
| :--: | :-------------------------------------: |
| `-f` | 显示完整格式（包括 UID、PID、PPID 等） |
| `-F` | 更详细的完整格式（添加 SZ、RSS 等字段） |
| `-l` | 长格式显示（包括 F、S、PRI 等） |
| `-j` | 显示作业信息（会话 ID、进程组 ID） |

**示例**:

bash

```bash
ps -ef     # 显示所有进程的完整信息
ps -l      # 长格式显示当前终端进程
```
```

---

**3. 过滤进程**

| 选项 | 说明 |
| --- | --- |
| `-p <PID>` | 显示指定 PID 的进程 |
| `-C <命令>` | 按进程名过滤（如 `-C sshd`） |
| `-t <终端>` | 显示指定终端的进程（如 `-t pts/0`） |

**示例**:

bash

```bash
ps -p 1234          # 显示 PID=1234 的进程
ps -C nginx         # 显示所有名为 "nginx" 的进程
```

---

**4. 动态排序**

| 选项 | 说明 |
| --- | --- |
| `--sort=<字段>` | 按指定字段排序（如 `--sort=-%cpu` 按 CPU 降序） |

**示例**:

bash

```bash
ps -e --sort=-%mem  # 按内存使用降序显示所有进程
```

---

**5. 树状显示进程关系**

| 选项 | 说明 |
| --- | --- |
| `--forest` | 以树状结构显示进程父子关系 |

**示例**：

bash

```
ps -ef --forest  # 显示进程树
```

---

## 三、输出字段详解

常用输出列及其含义：

| 字段 | 说明 |
| --- | --- |
| `PID` | 进程 ID |
| `PPID` | 父进程 ID |
| `USER` | 进程所有者 |
| `%CPU` | CPU 占用率 |
| `%MEM` | 内存占用率 |
| `VSZ` | 虚拟内存大小（KB） |
| `RSS` | 物理内存大小（KB） |
| `TTY` | 关联的终端 |
| `STAT` | 进程状态（如 `S`=睡眠，`R`=运行） |
| `START` | 进程启动时间 |
| `TIME` | 累计 CPU 时间 |
| `COMMAND` | 启动命令（`[]` 表示内核线程） |

---

## 四、常用组合选项

**1. 经典组合：`ps aux`**

bash

```
ps aux
```

- **说明**：BSD 风格，显示所有进程的详细信息。
- **输出字段**：USER、PID、%CPU、%MEM、VSZ、RSS、TTY、STAT、START、TIME、COMMAND.

**2. 系统进程查看：`ps -ef`**

bash

```
ps -ef
```

- **说明**：Unix 风格，显示所有进程的完整信息。
- **输出字段**：UID、PID、PPID、C、STIME、TTY、TIME、CMD.

---

## 五、自定义输出格式

使用 `-o` 选项指定显示的字段：

bash

```
ps -eo pid,user,%cpu,%mem,cmd --sort=-%cpu | head
```

- **说明**：显示 PID、用户、CPU 占用、内存占用和命令，按 CPU 降序排序。

---

## 六、实际应用场景

**1. 查找进程的 PID**

bash

```
ps aux | grep nginx      # 查找 nginx 进程的 PID
```

**2. 监控资源占用**

bash

```
ps -eo pid,%cpu,%mem,cmd --sort=-%mem | head  # 查看最耗内存的进程
```

**3. 杀死进程前的确认**

bash

```
ps -p 5678 -o pid,cmd    # 确认 PID=5678 的进程是否为目标进程
```

**4. 查看进程树**

bash

```
ps -ef --forest          # 显示父子进程关系
```

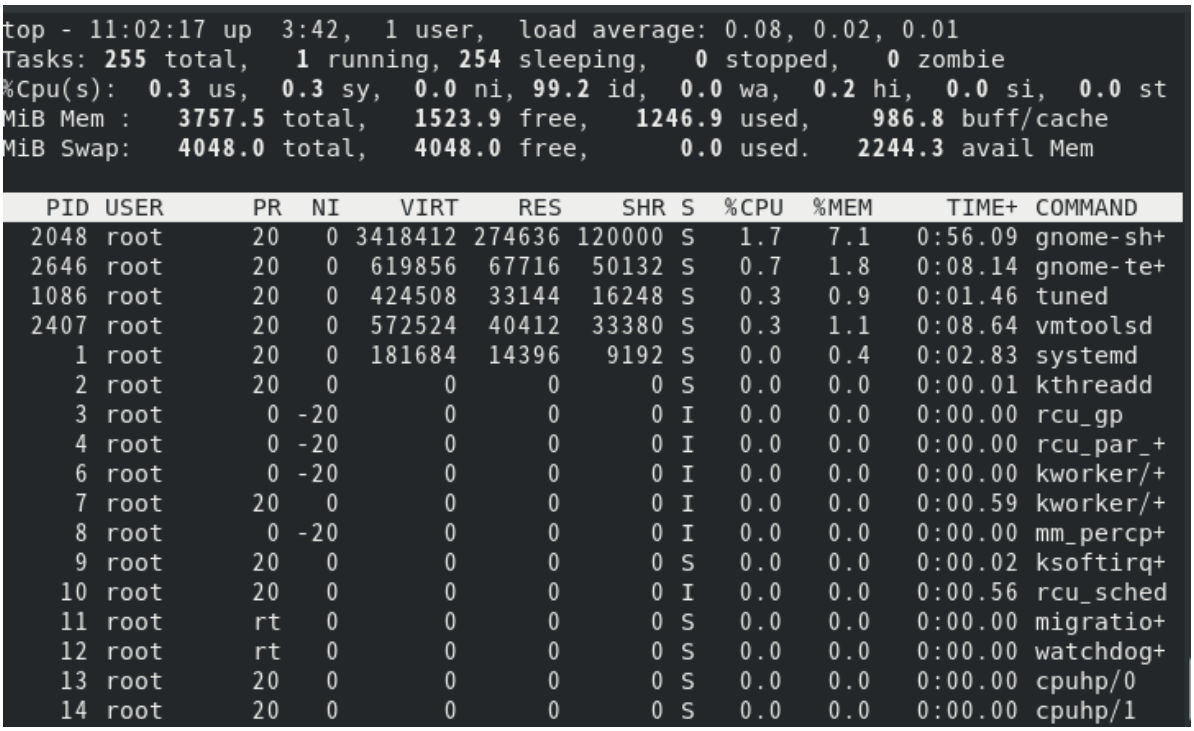
---

## 2.3 动态查看进程 top

`top` 是 Linux 中最常用的实时系统监控工具，能够动态显示 **进程资源占用** 和 **系统整体状态。**它提供了交互式操作，方便用户快速分析性能瓶颈

### 1.基础语法

```
top                 # 启动 top
q 或 Ctrl+C          # 退出 top
```



```
top - 11:02:17 up  3:42,  1 user,  load average: 0.08, 0.02, 0.01
Tasks: 255 total,   1 running, 254 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.2 id,  0.0 wa,  0.2 hi,  0.0 si,  0.0 st
MiB Mem :   3757.5 total,   1523.9 free,   1246.9 used,    986.8 buff/cache
MiB Swap:   4048.0 total,   4048.0 free,      0.0 used.   2244.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2048 root      20   0 3418412 274636 120000 S   1.7   7.1   0:56.09 gnome-sh+
 2646 root      20   0  619856  67716  50132 S   0.7   1.8   0:08.14 gnome-te+
 1086 root      20   0  424508  33144  16248 S   0.3   0.9   0:01.46 tuned
 2407 root      20   0  572524  40412  33380 S   0.3   1.1   0:08.64 vmtoolsd
    1 root      20   0  181684  14396   9192 S   0.0   0.4   0:02.83 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_+
    6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/+
    7 root      20   0       0      0      0 I   0.0   0.0   0:00.59 kworker/+
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percp+
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.02 ksoftirq+
   10 root      20   0       0      0      0 I   0.0   0.0   0:00.56 rcu_sched
   11 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migratio+
   12 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 watchdog+
   13 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
```

### 2. 界面解析

`top` 界面分为 **摘要区**（系统概览）和 **进程列表区**（实时进程信息）

**上半部分（系统概览）**

1. 系统时间/负载
2. 任务统计（176 total）
3. CPU使用分布（99.7% idle）
4. 内存使用（3.8GB total）
5. Swap使用（4.0GB free）

**下半部分（进程列表）**

| 字段 | 说明 |
| --- | --- |
| VIRT | 虚拟内存（包含共享库） |
| RES | 常驻内存 |
| SHR | 共享内存 |
| NI | Nice值（-20~19） |

### 3.交互式命令（运行中按快捷键）

| 命令 | 功能 |
| --- | --- |
| `k` | 终止进程（输入 PID，再输入信号，默认 `SIGTERM`）。 |
| `r` | 调整进程优先级（输入 PID，再输入 Nice 值）。 |
| `P` | 按 CPU 使用率排序。 |
| `M` | 按内存使用率排序。 |
| `T` | 按运行时间排序。 |
| `N` | 按 PID 排序。 |
| `z` | 切换颜色高亮显示。 |
| `c` | 显示完整命令路径。 |
| `1` | 展开显示每个 CPU 核心的负载。 |
| `h` | 查看帮助信息。 |
| `空格` | 立即刷新界面。 |

### 4.常用操作

```
快捷键：
M - 按内存排序    P - 按CPU排序
k - 终止进程      1 - 显示CPU核心
z - 彩色显示      q - 退出
```

---

## 2.4 信号控制

### 1.信号列表

bash

kill -l # 查看所有信号

```
[root@xnha ~]# 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
```

## 2.常用信号

| 信号 | 说明 |
| --- | --- |
| 1 | SIGHUP (重载配置)/重刷新一个进程，主要用户服务的刷新 |
| 9 | SIGKILL (强制终止)/杀掉一个进程，但不杀掉其子进程（centos7） |
| 15 | SIGTERM (正常终止)/杀掉进程及其子进程 |
| 18 | SIGCONT (恢复执行) |
| 19 | SIGSTOP (暂停)/暂停一个进程 |

## 实验

1.启动一个httpd服务（会产生多个进程）

```
[root@xnha ~]# systemctl restart httpd
```

2.查找所有httpd进程

```
[root@xnha ~]# ps -ef | grep httpd
root       8267      1  0 11:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
 #8267是httpd服务的主进程，其他都是子进程
apache     8283   8267  0 11:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8284   8267  0 11:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8285   8267  0 11:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8286   8267  0 11:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
root       8508   3582  0 11:17 pts/1    00:00:00 grep --color=auto httpd
```

3.对httpd 的主进程发送信号1

```
[root@xnha ~]# kill -1 8267              #当服务配置变更以后，可以通过-1来刷新主进程
[root@xnha ~]# ps -ef | grep httpd        #主进程pid没变，子进程Pid全变了

root       8267      1  0 11:16 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8549   8267  0 11:20 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8550   8267  0 11:20 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8551   8267  0 11:20 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8552   8267  0 11:20 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
root       8766   3582  0 11:20 pts/1    00:00:00 grep --color=auto httpd
```

4.对httpd的主进程发送信号15，查看服务状态

```
[xnha ~]# ps -ef | grep httpd
root       8810   3582  0 11:22 pts/1    00:00:00 grep --color=auto httpd
[root@xnha ~]# ps -ef | grep httpd |grep -v grep
[root@xnha ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor pres>
   Active: inactive (dead)
     Docs: man:httpd.service(8)
```

5.重新启动httpd服务，对主进程发送信号9

```
[root@xnha ~]# systemctl restart httpd
[root@xnha ~]# ps -ef | grep httpd |grep -v grep
root       8865      1  0 11:25 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8872   8865  0 11:25 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8873   8865  0 11:25 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8874   8865  0 11:25 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     8875   8865  0 11:25 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
[root@xnha ~]# kill -9 8865
[root@xnha ~]# ps -ef | grep httpd |grep -v grep
[root@xnha ~]#   #所有进程都没了（centos7中子进程依然存在，可以提供服务）
```

## 2.5 进程优先级

### Nice值范围

- -20（最高）到 19（最低）
- 查看优先级：

  bash

  ```
  ps axo pid,command,nice --sort=-nice
  ```

---

## 2.6 作业控制

linux可以在一个终端中管理多个任务进程(这些进程称之为job)。 有些进程会占用终端运行很久或是一直占用终端，那么想要运行第二进程 就无法操作了, 所以进程可以放到后台执行，这样就不影响在同一个终端运 行第二个进程了

### 1. 前后台管理

```
[root@shanxun ~]# jobs -l                       #查看当前终端运行的job，目前为空
[root@shanxun ~]# sleep 100000                  #此命令为等待100000秒，也就是说会占用终端
100000秒
                                                #使用ctrl+c取消，此时sleep 100000还没有
完成工作
[root@shanxun ~]# sleep 100000&                  # &为后台符号，这样sleep 100000可以在终
端后台继续运行
[1] 9262
[root@shanxun ~]# ps -ef | grep sleep
```

```
root       9261   1059  0 11:36 ?        00:00:00 sleep 60
root       9262   3582  0 11:36 pts/1    00:00:00 sleep 100000
root       9270   3582  0 11:36 pts/1    00:00:00 grep --color=auto sleep

[root@shanxun ~]# jobs -l                    #jobs -l可以查看到了
[1]+ 12615 Running

[root@shanxun ~]# fg %1                      # fg命令将job由后台转到终端前台运行，%1代表第1个job
sleep 100000                                 #按ctrl+z可以将占用前台的进程转到后台暂停执行
[1]+  Stopped
[root@shanxun ~]# jobs -l
[1]+ 12615 Stopped  sleep 100000 &           #确认为stop状态

[root@shanxun ~]# bg %1                      # bg命令将后台stopped状态的进程转为后台running运行状态
[1]+ sleep 100000 &
[root@shanxun ~]# jobs -l
[1]+ 12615 Running   sleep 100000 &          #确认为运行状态
```

### 2. 注意！！！

- `kill 1`：终止PID为1的进程（系统进程）
- `kill %1`：终止作业号为1的后台进程

---

## 2.7 虚拟文件系统 proc

### 关键文件

bash

```
/proc/cpuinfo    # CPU信息
/proc/meminfo    # 内存信息
/proc/cmdline    # 内核启动参数
```

### 查看命令

bash

```
cat /proc/cpuinfo       # 查看CPU详情
less /proc/meminfo      # 查看内存使用
```

---