> **注意**
> **monitor** 脚本会一直运行到被终止为止。它通过执行五万个加法题来生成人为 CPU 负载。然后，它休眠一秒钟，重置变量，再重复。

2.3. 使用 **chmod** 命令，使 **monitor** 文件变为可执行文件。

```
[student@servera ~]$ chmod a+x /home/student/bin/monitor
```

3. 在右终端 shell 中，运行 **top** 实用程序。尽可能将窗口调高。

```
[student@servera ~]$ top
top - 12:13:03 up 11 days, 58 min,  3 users,  load average: 0.00, 0.00, 0.00
Tasks: 113 total,   2 running, 111 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.0 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1829.4 total,   1377.3 free,    193.9 used,    258.2 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1476.1 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
5861 root      20   0       0      0      0 I   0.3   0.0   0:00.71 kworker/1:3-
events
6068 student   20   0  273564   4300   3688 R   0.3   0.2   0:00.01 top
   1 root      20   0  178680  13424   8924 S   0.0   0.7   0:04.03 systemd
   2 root      20   0       0      0      0 S   0.0   0.0   0:00.03 kthreadd
   3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
...output omitted...
```

4. 在左终端 shell 中，使用 **lscpu** 命令确定此虚拟机上的逻辑 CPU 数量。

```
[student@servera ~]$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              2
...output omitted...
```

5. 在左终端 shell 中，运行 **monitor** 可执行文件的一个实例。使用与符号 (&) 在后台运行进程。

```
[student@servera ~]$ monitor &
[1] 6071
```

6. 在右终端 shell 中，观察 **top** 显示。使用单一击键操作 **l**、**t** 和 **m**，切换负载、线程和内存标题行。在观察此行为后，确保所有标题都显示出来。

7. 记下 **monitor** 的进程 ID (PID)。查看该进程的 CPU 百分比，它应该在 15% 或 20% 左右徘徊。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
071 student   20   0  222448   2964   2716 S  18.7   0.2   0:27.35 monitor
...output omitted...
```

查看负载平均值。一分钟负载平均值的当前值应小于 1。观察到的值可能会受到另一虚拟机或虚拟主机资源争用的影响。

```
top - 12:23:45 up 11 days,  1:09,  3 users,  load average: 0.21, 0.14, 0.05
```

8. 在左终端 shell 中，运行 **monitor** 的第二个实例。使用与符号 (&) 在后台运行进程。

```
[student@servera ~]$ monitor &
[2] 6498
```

9. 在右终端 shell 中，记下第二个 monitor 进程的进程 ID (PID)。查看该进程的 CPU 百分比，它也应该在 15% 到 20% 之间徘徊。

```
[student@servera ~]$ top
 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
6071 student   20   0  222448   2964   2716 S  19.0   0.2   1:36.53 monitor
6498 student   20   0  222448   2996   2748 R  15.7   0.2   0:16.34 monitor
...output omitted...
```

再次查看一分钟负载平均值，它的值依然小于 1。务必至少等待一分钟，让计算调整到新的工作负载。

```
top - 12:27:39 up 11 days,  1:13,  3 users,  load average: 0.36, 0.25, 0.11
```

10. 在左终端 shell 中，运行 **monitor** 的第三个实例。使用与符号 (&) 在后台运行进程。

```
[student@servera ~]$ monitor &
[3] 6881
```

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

11. 在右终端 shell 中，记下第三个 **monitor** 进程的进程 ID (PID)。查看该进程的 CPU 百分比，它仍然应该在 15% 到 20% 之间徘徊。

```
[student@servera ~]$ top
 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
6881 student   20   0  222448   3032   2784 S  18.6   0.2   0:11.48 monitor
6498 student   20   0  222448   2996   2748 S  15.6   0.2   0:47.86 monitor
6071 student   20   0  222448   2964   2716 S  18.1   0.2   2:07.86 monitor
```

要将负载平均值推到 1 以上，您必须再启动一些 **monitor** 进程。课堂设置具有 2 个 CPU，因此只有 3 个进程不能提供足够的压力。再启动三个 **monitor** 进程。再次查看一分钟负载平均值，它的值现在应该会超过 1。务必至少等待一分钟，让计算调整到新的工作负载。

```
[student@servera ~]$ monitor &
[4] 10708
[student@servera ~]$ monitor &
[5] 11122
[student@servera ~]$ monitor &
[6] 11338
```

```
top - 12:42:32 up 11 days,  1:28,  3 users,  load average: 1.23, 2.50, 1.54
```

12. 观察完负载平均值时，从 **top** 内终止各个 **monitor** 进程。

12.1.在右终端 shell 中，按 **k** 键。观察标题下方和列上方的提示。

```
...output omitted...
PID to signal/kill [default pid = 11338]
```

12.2.提示已选择了列表顶部的 **monitor** 进程。按 **Enter** 键中断进程。

```
...output omitted...
Send pid 11338 signal [15/sigterm]
```

12.3.再次按 **Enter** 键，以确认 SIGTERM 信号 15。

确认所选的进程不再能够在 **top** 中观察到。如果 PID 依然还在，请重复终止步骤，并且在提示时换用 SIGKILL 信号 9。

```
 6498 student   20   0  222448   2996   2748 R  22.9   0.2   5:31.47 monitor
 6881 student   20   0  222448   3032   2784 R  21.3   0.2   4:54.47 monitor
11122 student   20   0  222448   2984   2736 R  15.3   0.2   2:32.48 monitor
 6071 student   20   0  222448   2964   2716 S  15.0   0.2   6:50.90 monitor
10708 student   20   0  222448   3032   2784 S  14.6   0.2   2:53.46 monitor
```

13. 为每个剩余的 **monitor** 实例重复上述步骤。确认 **top** 中没有剩余的 **monitor** 进程。

14. 在右终端 shell 中，按 **q** 键以退出 **top**。在两个终端窗口中退出 **servera**。

海量视频题库 mvitpub.com QQ:5565462
www.52mvit.com

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

完成

在 `workstation` 上，运行 **`lab processes-monitor finish`** 脚本来完成本练习。

```
[student@workstation ~]$ lab processes-monitor finish
```

本引导式练习到此结束。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

开放研究实验

监控和管理 LINUX 进程

任务执行清单

在本实验中，您将查找和管理系统上使用最多资源的进程。

成果

您应能够使用 **top** 作为流程管理工具来管理流程。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab processes-review start** 命令。该命令将运行一个起始脚本，以确定主机 serverb 是否可从网络访问。

```
[student@workstation ~]$ lab processes-review start
```

1. 在 workstation 上，打开两个终端窗口并将它们并排显示。在本节中，这些终端指代为左和右。在每个终端窗口中，以 student 用户身份登录 serverb。

   创建一个名为 **process101** 的脚本，这将生成人为 CPU 负载。在 **/home/student/bin** 目录中创建该脚本。

```
#!/bin/bash
while true; do
  var=1
  while [[ var -lt 50000 ]]; do
    var=$(($var+1))
  done
  sleep 1
done
```

2. 在右侧终端中，运行 **top** 实用程序。
3. 在左侧终端 shell 中，确定虚拟机上的逻辑 CPU 数量。在后台运行 **process101** 脚本。
4. 在右终端 shell 中，观察 **top** 显示。在负载、线程和内存之间切换。记下 **process101** 的进程 ID (PID)。查看 CPU 百分比。它应该在 10% 到 15% 左右徘徊。在查看了负载、线程和内存后，确保 **top** 显示 CPU 使用率。
5. 关闭显示中的粗体使用。保存此配置，以便在重启 top 时重新利用。确认已保存更改。
6. 将 **process101** 脚本复制到名为 **process102** 的新文件。编辑脚本，以创建更多人为 CPU 负载。将负载从五万增加到十万。在后台启动 **process102** 进程。
7. 在右终端 shell 中，确认进程正在运行并在使用最多的 CPU 资源。负载应该在 25% 到 35% 之间徘徊。

海量视频题库 myitpub.com QQ:556462 www.52myit.com

8. 负载平均值仍低于 1。将 **process101** 复制到一个名为 **process103** 的新脚本。将加法数量增加到八十万。在后台启动 **process103**。确认负载平均值超过 1。可能需要等几分钟后，负载平均值才会变化。
9. 在左终端 shell 中，变为 root 用户。暂停 **process101** 进程。列出剩余的作业。观察 **process101** 的进程状态现在为 **T**。
10. 恢复 **process101** 进程。
11. 通过命令行终止 **process101**、**process102** 和 **process103**。确认 **top** 中不再显示这些进程。
12. 在左终端 shell 中，从 root 用户退出。在右终端 shell 中，停止 **top** 命令。在两个窗口中，退出 serverb。

评估

在 **workstation** 上，运行 lab processes-review grade 脚本来确认是否成功完成本练习。

```
[student@workstation ~]$ lab processes-review grade
```

完成

在 workstation 上，运行 **lab processes-review finish** 脚本来完成本实验。

```
[student@workstation ~]$ lab processes-review finish
```

本实验到此结束。

海量视频题库 myitpub.com QQ:556462
www.52myit.com

解决方案

监控和管理 LINUX 进程

任务执行清单

在本实验中，您将查找和管理系统上使用最多资源的进程。

成果

您应能够使用 **top** 作为流程管理工具来管理流程。

在你开始之前

以 student 用户身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab processes-review start** 命令。该命令将运行一个起始脚本，以确定主机 serverb 是否可从网络访问。

```
[student@workstation ~]$ lab processes-review start
```

1. 在 workstation 上，打开两个终端窗口并将它们并排显示。在本节中，这些终端指代为左和右。在每个终端窗口中，以 student 用户身份登录 serverb。

   创建一个名为 **process101** 的脚本，这将生成人为 CPU 负载。在 **/home/student/bin** 目录中创建该脚本。

```
#!/bin/bash
while true; do
  var=1
  while [[ var -lt 50000 ]]; do
    var=$(($var+1))
  done
  sleep 1
done
```

   1.1. 在 workstation 上，打开两个终端窗口并将它们并排显示。在每个终端中，使用 **ssh** 命令以 student 用户身份登录 serverb。

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

   1.2. 在左终端 shell 中，使用 **mkdir** 命令来创建 **/home/student/bin** 目录。

```
[student@serverb ~]$ mkdir /home/student/bin
```

   1.3. 在左终端 shell 中，使用 **vim** 命令来创建 **process101** 脚本。按 **i** 键，以进入交互模式。键入 **:wq** 以保存文件。

海量视频题库 myitpub.com QQ:5565462 www.52myit.com

```
[student@serverb ~]$ vim /home/student/bin/process101
#!/bin/bash
while true; do
  var=1
  while [[ var -lt 50000 ]]; do
    var=$(($var+1))
  done
  sleep 1
done
```

1.4. 使用 **chmod** 命令，使 **process101** 文件变为可执行文件。

```
[student@serverb ~]$ chmod +x /home/student/bin/process101
```

2. 在右侧终端中，运行 **top** 实用程序。

2.1. 在右侧终端中，运行 **top** 实用程序。尽可能将窗口调高。

```
[student@serverb ~]$ top
top - 13:47:06 up 19 min,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 110 total,   1 running, 109 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  3.1 sy,  0.0 ni, 96.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1829.4 total,   1439.1 free,    171.9 used,    218.4 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1499.6 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME  COMMAND
  1 root      20   0  178536  13488   8996 S   0.0   0.7   0:01.15 systemd
  2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 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_gp
  6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-
kblockd
...output omitted...
```

3. 在左侧终端 shell 中，确定虚拟机上的逻辑 CPU 数量。在后台运行 **process101** 脚本。

3.1. 使用 **grep** 命令确定逻辑 CPU 的数量。

```
[student@serverb ~]$ grep "model name" /proc/cpuinfo | wc -l
2
```

3.2. 使用 **cd** 命令更改到 **/home/student/bin** 目录。在后台运行 **process101** 脚本。

```
[student@serverb ~]$ cd /home/student/bin
[student@serverb bin]$ process101 &
[1] 1180
```

4. 在右终端 shell 中，观察 **top** 显示。在负载、线程和内存之间切换。记下 **process101** 的进程 ID (PID)。查看 CPU 百分比。它应该在 10% 到 15% 左右徘徊。在查看了负载、线程和内存后，确保 **top** 显示 CPU 使用率。

海量视频题库 qwitpub.com QQ:5565462
www.52qwit.com

4.1. 按 **Shift+m**。

```
top - 13:56:24 up 28 min,  2 users,  load average: 0.21, 0.08, 0.02
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.8 us,  1.3 sy,  0.0 ni, 92.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1829.4 total,   1438.1 free,    172.7 used,    218.6 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1498.7 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 705 root      20   0  409956  34880  33620 S   0.0   1.9   0:00.04 sssd_nss
 706 root      20   0  454304  34472  14304 S   0.0   1.8   0:00.62 firewalld
 725 root      20   0  611348  28244  14076 S   0.0   1.5   0:00.27 tuned
 663 polkitd   20   0 1907312  23876  16040 S   0.0   1.3   0:00.04 polkitd
 718 root      20   0  600316  17176  14832 S   0.0   0.9   0:00.06 NetworkManager
...output omitted...
```

注意

请注意，当 top 切换到内存模式时，**process101** 不再是第一个进程。您可以通过按 **Shift+p** 返回到 CPU 使用率。

4.2. 按 **m**。

```
top - 09:32:52 up 20:05,  2 users,  load average: 0.18, 0.10, 0.03
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
%Cpu(s):   7.8/1.5     9[|||||||||                                    ]
MiB Mem : 18.3/1829.4   [||||||||||||||||||                           ]
MiB Swap:  0.0/1024.0   [                                             ]
 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 705 root      20   0  409956  34880  33620 S   0.0   1.9   0:00.04 sssd_nss
 706 root      20   0  454304  34472  14304 S   0.0   1.8   0:00.62 firewalld
 725 root      20   0  611348  28244  14076 S   0.0   1.5   0:00.30 tuned
 663 polkitd   20   0 1907312  23876  16040 S   0.0   1.3   0:00.04 polkitd
 718 root      20   0  600316  17176  14832 S   0.0   0.9   0:00.07 NetworkManager
...output omitted...
```

4.3. 按 **t**。

```
Tasks: 113 total,   2 running, 111 sleeping,   0 stopped,   0 zombie
%Cpu(s):   7.8/1.5     9[|||||||||                                    ]
MiB Mem :   1829.4 total,   1436.7 free,    173.7 used,    219.0 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1497.7 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
1180 student   20   0  222448   3056   2808 S  12.0   0.2   1:59.94 process101
 705 root      20   0  409956  34880  33620 S   0.0   1.9   0:00.04 sssd_nss
 706 root      20   0  454304  34472  14304 S   0.0   1.8   0:00.62 firewalld
 725 root      20   0  611348  28244  14076 S   0.0   1.5   0:00.30 tuned
 663 polkitd   20   0 1907312  23876  16040 S   0.0   1.3   0:00.04 polkitd
 718 root      20   0  600316  17176  14832 S   0.0   0.9   0:00.07 NetworkManager
...output omitted...
```

海量视频题库 myitpub.com QQ:556462
www.52myit.com

4.4. 按 **Shift+p**。

```
top - 09:35:48 up 20:08,  2 users,  load average: 0.10, 0.10, 0.04
Tasks: 110 total,   4 running, 106 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6.8 us,  1.0 sy,  0.0 ni, 92.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 18.3/1829.4   [||||||||||||||||||                                  ]
MiB Swap:  0.0/1024.0   [                                                   ]

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
27179 student   20   0  222448   3060   2812 R  15.6   0.2   1:13.47 process101
...output omitted...
```

5. 关闭显示中的粗体使用。保存此配置，以便在重启 top 时重新利用。确认已保存更改。

5.1. 按 **Shift+b**，以关闭粗体使用。

```
top - 19:40:30 up  6:12,  2 users,  load average: 0.11, 0.12, 0.09
Tasks: 112 total,   1 running, 111 sleeping,   0 stopped,   0 zombie
%Cpu(s):   7.6/1.5     9[|||||||||                                      ]
MiB Mem : 18.2/1829.4   [||||||||||||||||||                              ]
MiB Swap:  0.0/1024.0   [                                                ]
```

5.2. 按 **Shift+w**，以保存此配置。默认配置存储在 **/home/student/.config/procps** 目录下的 **toprc** 中。在左终端 shell 中，确认 **toprc** 文件存在。

```
[student@serverb bin]$ ls -l /home/student/.config/procps/toprc
-rw-rw-r--. 1 student student 966 Feb 18 19:45 /home/student/.config/procps/toprc
```

5.3. 在右终端 shell 中，退出 **top**，然后重新启动它。确认新显示使用了保存的配置。

```
top - 00:58:21 up 43 min,  2 users,  load average: 0.29, 0.28, 0.20
Tasks: 105 total,   1 running, 104 sleeping,   0 stopped,   0 zombie
%Cpu(s):  11.0/1.8    13[||||||||||||                                   ]
MiB Mem : 18.7/1829.0   [||||||||||||||||||                              ]
MiB Swap:  0.0/0.0      [                                                ]
```

6. 将 **process101** 脚本复制到名为 **process102** 的新文件。编辑脚本，以创建更多人为 CPU 负载。将负载从五万增加到十万。在后台启动 **process102** 进程。

6.1. 在左终端 shell 中，使用 **cp** 命令将 **process101** 复制到 **process102**。

```
[student@serverb bin]$ cp process101 process102
```

6.2. 使用 **vim** 命令编辑 **process102** 脚本。将加法题数量从五万增加到十万。通过按 **i** 进入交互模式。键入 **:wq** 以保存文件。

```
[student@serverb bin]$ vim process102
#!/bin/bash
while true; do
  var=1
    while [[ var -lt 100000 ]]; do
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

```
        var=$(($var+1))
    done
    sleep 1
done
```

6.3. 在后台启动 **process102** 进程。

```
[student@serverb bin]$ process102 &
[2] 20723
```

6.4. 使用 **jobs** 命令，确认两个进程都在后台运行。

```
[student@serverb bin]$ jobs
[1]-  Running                 process101 &
[2]+  Running                 process102 &
```

7. 在右终端 shell 中，确认进程正在运行并在使用最多的 CPU 资源。负载应该在 25% 到 35% 之间徘徊。

7.1. 在右终端 shell 中，确认进程正在运行并在使用最多的 CPU 资源。负载应该在 25% 到 35% 之间徘徊。

```
top - 20:14:16 up  6:46,  2 users,  load average: 0.58, 0.34, 0.18
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
499 %Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
 st
500 MiB Mem :   1829.4 total,   1428.7 free,    179.0 used,    221.8 buff/cache
501 MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1492.1 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
20723 student   20   0  222448   3016   2764 S  24.7   0.2   0:53.28 process102
 1180 student   20   0  222448   3056   2808 S  12.0   0.2  58:01.56 process101
...output omitted...
```

> **注意**
> 如果您没有在进程列表的顶部看到 **process101** 和 **process102**，请按 **Shift+p** 以确保 top 按照 CPU 使用率排序。

8. 负载平均值仍低于 1。将 **process101** 复制到一个名为 **process103** 的新脚本。将加法数量增加到八十万。在后台启动 **process103**。确认负载平均值超过 1。可能需要等几分钟后，负载平均值才会变化。

8.1. 在右终端 shell 中，确认负载平均值低于 1。

```
top - 20:24:13 up  6:56,  2 users,  load average: 0.43, 0.41, 0.29
...output omitted...
```

8.2. 在左终端 shell 中，使用 **cp** 命令将 **process101** 复制到名为 **process103** 的新脚本。

```
[student@serverb bin]$ cp process101 process103
```

海量视频题库 mvitpub.com QQ:565462
www.52mvit.com

8.3. 在左终端 shell 中，使用 **vim** 命令来编辑 **process103** 脚本。将加法数量增加到八十万。通过按 **i** 键进入交互模式。键入 **:wq** 以保存文件。

```
[student@serverb bin]$ vim process103
#!/bin/bash
while true; do
  var=1
    while [[ var -lt 800000 ]]; do
      var=$(($var+1))
    done
    sleep 1
done
```

8.4. 在后台启动 **process103**。CPU 使用率在 60% 到 85% 之间徘徊。

```
[student@serverb bin]$ process103 &
[3] 22751
```

8.5. 确认所有三个作业都在后台运行。

```
[student@serverb bin]$ jobs
[1]   Running                 process101 &
[2]-  Running                 process102 &
[3]+  Running                 process103 &
```

8.6. 在右终端窗口中，确认负载平均值超过 1。

```
top - 20:45:34 up  7:17,  2 users,  load average: 1.10, 0.90, 0.64
```

9. 在左终端 shell 中，变为 root 用户。暂停 **process101** 进程。列出剩余的作业。观察 **process101** 的进程状态现在为 **T**。

9.1. 使用 **su -** 命令变为 root 用户。密码是 redhat。

```
[student@serverb bin]$ su -
Password: redhat
```

9.2. 使用 **pkill** 命令及 **-SIGSTOP** 选项来暂停 **process101** 进程。

```
[root@serverb ~]# pkill -SIGSTOP process101
```

9.3. 在右终端 shell 中，确认 **process101** 不再运行。

```
top - 20:52:01 up  7:24,  2 users,  load average: 1.19, 1.19, 0.89
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
499 %Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
 st
500 MiB Mem :   1829.4 total,   1428.7 free,    179.0 used,    221.8 buff/cache
501 MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1492.1 avail Mem
```

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
24043 student   20   0  222448   2992   2744 R  66.1   0.2   6:59.50 process103
20723 student   20   0  222448   3016   2764 R  29.9   0.2  11:04.84 process102
...output omitted...
```

9.4. 在左终端 shell 中，运行 **ps jT** 命令以查看剩余的作业。

```
[root@serverb ~]# ps jT
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 ...output omitted...
 27138  1180  1180 27138 pts/0    28558 T     1000   3:06 /bin/bash /home/student/
bin/process101
 27138 20723 20723 27138 pts/0    28558 R     1000   1:23 /bin/bash /home/student/
bin/process102
 27138 24043 24043 27138 pts/0    28558 R     1000   2:35 /bin/bash /home/student/
bin/process103
 ...output omitted...
```

注意 `process101` 状态为 **T**。这表示该进程目前已被暂停。

10. 恢复 **process101** 进程。

10.1.在左终端 shell 中，使用 **pkill** 命令及 **-SIGCONT** 选项来恢复 **process101** 进程。

```
[root@serverb ~]# pkill -SIGCONT process101
```

10.2.在右终端 shell 中，确认进程重新处于运行状态。

```
top - 20:57:02 up  7:29,  2 users,  load average: 1.14, 1.20, 0.99
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1829.4 total,   1428.7 free,    179.0 used,    221.8 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1492.1 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
24043 student   20   0  222448   2992   2744 R  66.8   0.2  10:40.61 process103
20723 student   20   0  222448   3016   2764 S  24.9   0.2  12:25.10 process102
 1180 student   20   0  222448   3056   2808 S  17.9   0.2  64:07.99 process101
```

11. 通过命令行终止 **process101**、**process102** 和 **process103**。确认 **top** 中不再显示这些进程。

11.1.在左终端 shell 中，使用 **pkill** 命令终止 **process101**、**process102** 和 **process103**。

```
[root@serverb ~]# pkill process101
[root@serverb ~]# pkill process102
[root@serverb ~]# pkill process103
```

11.2.在右终端 shell 中，确认 **top** 中不再显示这些进程。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

```
top - 21:05:06 up  7:37,  2 users,  load average: 1.26, 1.29, 1.12
Tasks: 112 total,   2 running, 110 sleeping,   0 stopped,   0 zombie
499 %Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
 st
500 MiB Mem :   1829.4 total,   1428.7 free,    179.0 used,    221.8 buff/cache
501 MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   1492.1 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  1 root      20   0  178536  13488   8996 S   0.0   0.7   0:01.21 systemd
  2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
  3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
...output omitted...
```

12. 在左终端 shell 中，从 `root` 用户退出。在右终端 shell 中，停止 **`top`** 命令。在两个窗口中，退出 `serverb`。

12.1.使用 **`exit`** 命令，从 `root` 用户注销。

```
[root@serverb ~]# exit
logout
[1]   Terminated              process101
[2]   Terminated              process102
[3]-  Terminated              process103
```

12.2.从所有终端窗口退出。

```
[student@serverb bin]$ exit
[student@workstation ~]$
```

12.3.在右终端 shell 中，按 **`q`** 键以退出 **`top`**。使用 **`exit`** 命令注销。

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

评估

在 **`workstation`** 上，运行 lab processes-review grade 脚本来确认是否成功完成本练习。

```
[student@workstation ~]$ lab processes-review grade
```

完成

在 workstation 上，运行 **`lab processes-review finish`** 脚本来完成本实验。

```
[student@workstation ~]$ lab processes-review finish
```

本实验到此结束。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

总结

在本章中，您学到了：

- 进程是可执行程序的运行中实例。进程被分配有一个状态，可以是运行、睡眠、停止或僵停。**ps** 命令用于列出进程。
- 每个终端是其自身的会话，并且可以具有一个前台进程和多个独立的后台进程。**jobs** 命令显示终端会话中的进程。
- 信号是向执行中的程序报告事件的软件中断。**kill**、**pkill** 和 **killall** 命令使用信号来控制进程。
- 负载平均值是一种对系统繁忙程度的估算。要显示负载平均值，可使用 **top**、**uptime** 或 **w** 命令。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

章 9

控制服务和守护进程

目标	使用 Systemd 控制和监控网络服务与系统守护进程。
培训目标	· 列出由 `systemd` 服务启动的系统守护进程和网络服务，以及套接字单元。 · 使用 **`systemctl`** 控制系统守护进程和网络服务。
章节	· 识别自动启动的系统进程（及引导式练习） · 控制系统服务（及引导式练习）
实验	控制服务和守护进程

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

识别自动启动的系统进程

培训目标

学完本节后，您应能够列出 systemd 服务和套接字单元启动的系统守护进程和网络服务。

systemd 简介

systemd 守护进程管理 Linux 的启动，一般包括服务启动和服务管理。它可在系统引导时以及运行中的系统上激活系统资源、服务器守护进程和其他进程。

守护进程是在执行各种任务的后台等待或运行的进程。一般情况下，守护进程在系统引导时自动启动并持续运行至关机或被手动停止。按照惯例，许多守护进程的名称以字母 **d** 结尾。

systemd 意义上的服务通常指的是一个或多个守护进程，但启动或停止一项服务可能会对系统的状态进行一次性更改，不会留下守护进程之后继续运行（称为 oneshot）。

在红帽企业 Linux 中，第一个启动的进程 (PID 1) 是 systemd。如下是 systemd 提供的几项功能：

- 并行化功能（同时启动多个服务），它可提高系统的启动速度。
- 按需启动守护进程，而不需要单独的服务。
- 自动服务依赖关系管理，可以防止长时间超时。例如，只有在网络可用时，依赖网络的服务才会尝试启动。
- 利用 Linux 控制组一起追踪相关进程的方式。

描述服务单元

systemd 使用单元来管理不同类型的对象。下方列出了一些常用单元类型：

- 服务单元具有 **.service** 扩展名，代表系统服务。这种单元用于启动经常访问的守护进程，如 web 服务器。
- 套接字单元具有 **.socket** 扩展名，代表 systemd 应监控的进程间通信 (IPC) 套接字。如果客户端连接套接字，systemd 将启动一个守护进程并将连接传递给它。套接字单元用于延迟系统启动时的服务启动，或者按需启动不常使用的服务。
- 路径单元具有 **.path** 扩展名，用于将服务的激活推迟到特定文件系统更改发生之后。这通常用于使用假脱机目录的服务，如打印系统。

systemctl 命令用于管理单元。例如，可通过 **systemctl -t help** 命令来显示可用的单元类型。

> **重要**
> 使用 **systemctl** 时，您可以缩写单元名称、进程树条目和单元描述。

海量视频题库 hyitpub.com QQ:556462
www.52myit.com

列出服务单元

您可以使用 **systemctl** 命令来探索系统的当前状态。例如，以下命令会列出所有当前加载的服务单元，使用 **less** 可对输出进行分页。

```
[root@host ~]# systemctl list-units --type=service
UNIT                                LOAD   ACTIVE SUB     DESCRIPTION
atd.service                         loaded active running Job spooling tools
auditd.service                      loaded active running Security Auditing Service
chronyd.service                     loaded active running NTP client/server
crond.service                       loaded active running Command Scheduler
dbus.service                        loaded active running D-Bus System Message Bus
...output omitted...
```

以上输出中，通过 **--type=service** 选项将列出的单元类型限制为服务单元。输出包含以下几列：

systemctl list-units 命令输出中的列

UNIT
服务单元名称。

LOAD
systemd 是否正确解析了单元的配置并将该单元加载到内存中。

ACTIVE
单元的高级别激活状态。此信息表明单元是否已成功启动。

SUB
单元的低级别激活状态。此信息指示有关该单元的更多详细信息。信息视单元类型、状态以及单元的执行方式而异。

DESCRIPTION
单元的简短描述。

默认情况下，**systemctl list-units --type=service** 命令仅列出激活状态为 **active** 的服务单元。**--all** 选项可列出所有服务单位，不论激活状态如何。使用 **--state=** 选项可按照 **LOAD**、**ACTIVE** 或 **SUB** 字段中的值进行筛选。

```
[root@host ~]# systemctl list-units --type=service --all
UNIT                          LOAD      ACTIVE   SUB     DESCRIPTION
  atd.service                 loaded    active   running Job spooling tools
  auditd.service              loaded    active   running Security Auditing ...
  auth-rpcgss-module.service  loaded    inactive dead    Kernel Module ...
  chronyd.service             loaded    active   running NTP client/server
  cpupower.service            loaded    inactive dead    Configure CPU power ...
  crond.service               loaded    active   running Command Scheduler
  dbus.service                loaded    active   running D-Bus System Message Bus
● display-manager.service     not-found inactive dead    display-manager.service
...output omitted...
```

不带任何参数运行 **systemctl** 命令可列出已加载和活动的单元。

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
[root@host ~]# systemctl
UNIT                                    LOAD   ACTIVE SUB       DESCRIPTION
proc-sys-fs-binfmt_misc.automount       loaded active waiting   Arbitrary...
sys-devices-....device                  loaded active plugged   Virtio network...
sys-subsystem-net-devices-ens3.deviceloaded active plugged   Virtio network...
...
-.mount                                 loaded active mounted   Root Mount
boot.mount                              loaded active mounted   /boot
...
systemd-ask-password-plymouth.path      loaded active waiting   Forward Password...
systemd-ask-password-wall.path          loaded active waiting   Forward Password...
init.scope                              loaded active running   System and Servi...
session-1.scope                         loaded active running   Session 1 of...
atd.service                             loaded active running   Job spooling tools
auditd.service                          loaded active running   Security Auditing...
chronyd.service                         loaded active running   NTP client/server
crond.service                           loaded active running   Command Scheduler
...output omitted...
```

systemctl list-units 命令显示 systemd 服务尝试解析并加载到内存中的单元；它不显示已安装但未启用的服务。要查看所有已安装的单元文件的状态，请使用 **systemctl list-unit-files** 命令。例如：

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                   STATE
arp-ethers.service                          disabled
atd.service                                 enabled
auditd.service                              enabled
auth-rpcgss-module.service                  static
autovt@.service                             enabled
blk-availability.service                    disabled
...output omitted...
```

在 **systemctl list-units-files** 命令的输出中，**STATE** 字段的有效条目有 **enabled**、**disabled**、**static** 和 **masked**。

查看服务状态

使用 **systemctl status name.type** 来查看特定单元的状态。如果未提供单元类型，则 **systemctl** 将显示服务单元的状态（如果存在）。

```
[root@host ~]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset:
 enabled)
   Active: active (running) since Thu 2019-02-14 12:07:45 IST; 7h ago
 Main PID: 1073 (sshd)
   CGroup: /system.slice/sshd.service
           └─1073 /usr/sbin/sshd -D ...

Feb 14 11:51:39 host.example.com systemd[1]: Started OpenSSH server daemon.
Feb 14 11:51:39 host.example.com sshd[1073]: Could not load host key: /et...y
```

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
Feb 14 11:51:39 host.example.com sshd[1073]: Server listening on 0.0.0.0 ....
Feb 14 11:51:39 host.example.com sshd[1073]: Server listening on :: port 22.
Feb 14 11:53:21 host.example.com sshd[1270]: error: Could not load host k...y
Feb 14 11:53:22 host.example.com sshd[1270]: Accepted password for root f...2
...output omitted...
```

此命令将显示服务的当前状态。各个字段的含义如下：

服务单元信息

字段	描述
Loaded	服务单元是否已加载到内存中。
Active	服务单元是否正在运行，若是，它已经运行了多久。
Main PID	服务的主进程 ID，包括命令名称。
Status	有关该服务的其他信息。

状态输出中可以找到表示服务状态的几个关键字：

`systemctl` 输出中的服务状态

关键字	描述
loaded	单元配置文件已处理。
active (running)	正在通过一个或多个持续进程运行。
active (exited)	已成功完成一次性配置。
active (waiting)	运行中，但正在等待事件。
inactive	不在运行。
enabled	在系统引导时启动。
disabled	未设为在系统引导时启动。
static	无法启用，但可以由某一启用的单元自动启动。

注意

`systemctl status NAME` 命令取代了红帽企业 Linux 6 及早期版本中使用的 **`service NAME status`** 命令。

验证服务的状态

`systemctl` 命令提供了一些方法来验证服务的具体状态。例如，可使用以下命令验证服务单元当前是否处于活动状态（正在运行）：

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
[root@host ~]# systemctl is-active sshd.service
active
```

该命令将返回服务单元的状态，通常是 **active** 或 **inactive**。

运行以下命令可验证服务单元是否已启用在系统引导期间自动启动：

```
[root@host ~]# systemctl is-enabled sshd.service
enabled
```

该命令将返回服务单元是否已启用在系统引导期间启动，通常是 **enabled** 或 **disabled**。

要验证单元是否在启动过程中失败，请运行以下命令：

```
[root@host ~]# systemctl is-failed sshd.service
active
```

如果单元运行正常，该命令将返回 **active**，如果启动过程中发生了错误，则返回 **failed**。如果单元已被停止，它将返回 **unknown** 或 **inactive**。

要列出所有失败的单元，请运行 **systemctl --failed --type=service** 命令。

参考文献

systemd(1)、systemd.unit(5)、systemd.service(5)、systemd.socket(5) 和 systemctl(1) man page

如需更多信息，请参阅《红帽企业 Linux 8.0 基本系统设置配置指南》中的使用 systemd 管理服务章节，网址为：
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-services-with-systemd_configuring-basic-system-settings

海量视频题库 myitpub.com QQ:556546 2
www.52myit.com

指导练习

识别自动启动的系统进程

在本练习中，您将列出已安装的服务单元，并识别服务器上当前启用和活动的服务。

成果

您应能够列出已安装的服务单元，并在系统上识别活动和已启用的服务。

在你开始之前

在 workstation 上，以 student 用户身份并使用密码 student 进行登录。

从 workstation，运行 **lab services-identify start** 命令。该命令将运行一个起始脚本，它将确定主机 servera 是否可从网络访问。

```
[student@workstation ~]$ lab services-identify start
```

1. 使用 **ssh** 命令，以 student 用户身份登录 servera。系统已配置为使用 SSH 密钥来进行身份验证，因此不需要提供密码就能登录 servera。

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. 列出 servera 上安装的所有服务单元。

```
[student@servera ~]$ systemctl list-units --type=service
UNIT                 LOAD   ACTIVE SUB     DESCRIPTION
atd.service          loaded active running Job spooling tools
auditd.service       loaded active running Security Auditing Service
chronyd.service      loaded active running NTP client/server
crond.service        loaded active running Command Scheduler
dbus.service         loaded active running D-Bus System Message Bus
...output omitted...
```

按 **q** 键退出该命令。

3. 列出 servera 上所有活动和不活动的套接字单元。

```
[student@servera ~]$ systemctl list-units --type=socket --all
UNIT                 LOAD   ACTIVE   SUB       DESCRIPTION
dbus.socket          loaded active   running   D-Bus System Message Bus Socket
dm-event.socket      loaded active   listening Device-mapper event daemon FIFOs
lvm2-lvmpolld.socket loaded active   listening LVM2 poll daemon socket
...output omitted...
systemd-udevd-control.socket    loaded active   running   udev Control Socket
```

海量视频题库 myrtpub.com QQ:556462
www.52myit.com

```
systemd-udevd-kernel.socket     loaded active   running   udev Kernel Socket

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

12 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
```

4. 浏览 chronyd 服务的状态。此服务用于网络时间同步 (NTP)。

4.1. 显示 chronyd 服务的状态。记下任何活动守护进程的进程 ID。

```
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
 preset: enabled)
   Active: active (running) since Wed 2019-02-06 12:46:57 IST; 4h 7min ago
     Docs: man:chronyd(8)
           man:chrony.conf(5)
  Process: 684 ExecStartPost=/usr/libexec/chrony-helper update-daemon
 (code=exited, status=0/SUCCESS)
  Process: 673 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/
SUCCESS)
  Main PID: 680 (chronyd)
    Tasks: 1 (limit: 11406)
   Memory: 1.5M
   CGroup: /system.slice/chronyd.service
           └─680 /usr/sbin/chronyd

... jegui.ilt.example.com systemd[1]: Starting NTP client/server...
...output omitted...
... jegui.ilt.example.com systemd[1]: Started NTP client/server.
... servera.lab.example.com chronyd[680]: Source 172.25.254.254 offline
... servera.lab.example.com chronyd[680]: Source 172.25.254.254 online
... servera.lab.example.com chronyd[680]: Selected source 172.25.254.254
```

按 **q** 键退出该命令。

4.2. 确认列出的守护进程正在运行。在以上命令输出中，与 chronyd 服务关联的进程 ID 是 680。您系统上的进程 ID 可能有所不同。

```
[student@servera ~]$ ps -p 680
  PID TTY          TIME CMD
  680 ?        00:00:00 chronyd
```

5. 浏览 sshd 服务的状态。此服务用于系统间的安全加密通信。

5.1. 确定 sshd 服务是否已启用在系统引导时启动。

```
[student@servera ~]$ systemctl is-enabled sshd
enabled
```

海量视频题库 myitpub.com QQ:5565462
www.52myit.com

5.2. 确定 sshd 服务是否活动，但不显示所有状态信息。

```
[student@servera ~]$ systemctl is-active sshd
active
```

5.3. 显示 sshd 服务的状态。

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset:
 enabled)
   Active: active (running) since Wed 2019-02-06 12:46:58 IST; 4h 21min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 720 (sshd)
    Tasks: 1 (limit: 11406)
   Memory: 5.8M
   CGroup: /system.slice/sshd.service
   └─720 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,
   chacha20-poly1305@openssh.com,aes256-ctr,
   aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,
   aes128-cbc -oMACs=hmac-sha2-256-etm@openssh.com,hmac-sha>

... jegui.ilt.example.com systemd[1]: Starting OpenSSH server daemon...
... servera.lab.example.com sshd[720]: Server listening on 0.0.0.0 port 22.
... servera.lab.example.com systemd[1]: Started OpenSSH server daemon.
... servera.lab.example.com sshd[720]: Server listening on :: port 22.
...output omitted...
... servera.lab.example.com sshd[1380]: pam_unix(sshd:session): session opened for
 user student by (uid=0)
```

按 **q** 键退出该命令。

6. 列出所有服务单元的已启用或已禁用状态。

```
[student@servera ~]$ systemctl list-unit-files --type=service
UNIT FILE                    STATE
arp-ethers.service           disabled
atd.service                  enabled
auditd.service               enabled
auth-rpcgss-module.service   static
autovt@.service              enabled
blk-availability.service     disabled
chrony-dnssrv@.service       static
chrony-wait.service          disabled
chronyd.service              enabled
...output omitted...
```

按 **q** 键退出该命令。

7. 从 servera 退出。

海量视频题库 myitpub.com QQ:556462
www.52myit.com

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

完成

在 workstation 上，运行 **lab services-identify finish** 脚本来完成本练习。

```
[student@workstation ~]$ lab services-identify finish
```

本引导式练习到此结束。

海量视频题库 myitpub.com QQ:5565462
www.52myit.com