小测验

列出进程

选择以下问题的正确答案：

1. 哪种状态表示已被停止或暂停的进程？

a. D
b. R
c. S
d. T
e. Z

2. 哪种状态表示已释放除 PID 之外所有资源的进程？

a. D
b. R
c. S
d. T
e. Z

3. 父进程使用哪一个进程来复制以创建新的子进程？

a. exec
b. fork
c. zombie
d. syscall
e. reap

4. 哪种状态表示在满足某个条件之前处于睡眠状态的进程？

a. D
b. R
c. S
d. T
e. Z

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

解决方案

列出进程

选择以下问题的正确答案：

1. 哪种状态表示已被停止或暂停的进程？

a. D
b. R
c. S
d. T
e. Z

2. 哪种状态表示已释放除 PID 之外所有资源的进程？

a. D
b. R
c. S
d. T
e. Z

3. 父进程使用哪一个进程来复制以创建新的子进程？

a. exec
b. fork
c. zombie
d. syscall
e. reap

4. 哪种状态表示在满足某个条件之前处于睡眠状态的进程？

a. D
b. R
c. S
d. T
e. Z

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

控制作业

培训目标

学完本节后，您应能够使用 Bash 作业控制来管理从同一终端会话启动的多个进程。

描述作业和会话

作业控制是 shell 的一种功能，允许单个 shell 实例运行和管理多个命令。

作业与在 shell 提示符中输入的每个管道相关联。该管道中的所有进程均是作业的一部分，并且是同一个进程组的成员。如果在 shell 提示符处仅输入了一条命令，则这条命令可视为命令的最小“管道”，创建仅含有一个成员的一个作业。

一次只能有一个作业从特定终端窗口中读取输入和键盘生成的信号。属于该作业的进程是该控制终端的前台进程。

该控制终端的后台进程是与该终端相关联的任何其他作业的成员。终端的后台进程无法从终端读取输入或接收键盘生成的中断，但可以写入终端。后台中的作业可能已停止（暂停），也可能正在运行。如果某个正在运行的后台作业尝试从终端读取内容，则该作业将自动暂停。

每个终端是其自身的会话，并且可以具有一个前台进程和任意数量的独立后台进程。一个作业只能属于一个会话，也就是属于其控制终端的会话。

ps 命令在 **TTY** 列中显示进程的控制终端的设备名称。某些进程（如系统守护进程）由系统启动，并不是从 shell 提示符启动。这些进程没有控制终端，也不是作业的成员，并且无法转至前台。**ps** 命令在 **TTY** 列中针对这些进程显示一个问号 (**?**)。

在后台运行作业

任何命令或管道都可以在后台启动，只需在命令行的结尾处附加符号 **&** 即可。Bash shell 显示作业编号（特定于会话的唯一编号）和新建子进程的 PID。shell 不等待子进程终止，而会显示 shell 提示符。

```
[user@host ~]$ sleep 10000 &
[1] 5947
[user@host ~]$
```

注意

如果利用 & 符号将包含管道的命令行发送到后台，管道中最后一个命令的 PID 将用作输出。管道中的所有进程仍是该作业的成员。

```
[user@host ~]$  example_command | sort | mail -s "Sort output" &
[1] 5998
```

您可以使用 **jobs** 命令显示 Bash 为特定会话跟踪的作业列表。

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

```
[user@host ~]$ jobs
[1]+  Running                 sleep 10000 &
[user@host ~]$
```

可以使用 **fg** 命令和后台作业的 ID（%作业编号）将该后台作业转至前台。

```
[user@host ~]$ fg %1
sleep 10000
```

在上例中，**sleep** 命令现在正在控制终端的前台中运行。shell 自身再次睡眠，等待这一子进程退出。

如要将前台进程发送到后台，请首先在终端中按键盘生成的暂停请求 (**Ctrl**+**z**)。

```
sleep 10000
^Z
[1]+  Stopped                 sleep 10000
[user@host ~]$
```

该作业将立即置于后台并暂停。

ps j 命令显示与作业相关的信息。PID 是进程的唯一进程 ID。PPID 是此进程的父进程（即启动（分叉）此进程的进程）的 PID。PGID 是进程组首进程的 PID，通常是作业管道中的第一个进程。SID 是会话首进程的 PID，对于作业而言，这通常是正在其控制终端上运行的交互 shell。由于示例 **sleep** 命令当前已暂停，因此其进程状态为 **T**。

```
[user@host ~]$ ps j
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 2764  2768  2768  2768 pts/0     6377 Ss    1000   0:00 /bin/bash
 2768  5947  5947  2768 pts/0     6377 T     1000   0:00 sleep 10000
 2768  6377  6377  2768 pts/0     6377 R+    1000   0:00 ps j
```

要启动在后台运行的已暂停进程，可使用 **bg** 命令及相同的作业 ID。

```
[user@host ~]$ bg %1
[1]+ sleep 10000 &
```

如果用户尝试退出带有暂停作业的终端窗口（会话），那么 shell 将发出警告。如果用户再次尝试立即退出，暂停的作业将被中断。

注意

注意以上示例中 **[1]** 后面的 **+** 符号。**+** 符号表示此作业是当前的默认作业。也就是说，如果使用的某个命令需要参数 **%**作业编号，但并没有提供作业编号，那么将对具有 **+** 指示符的作业执行该操作。

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

参考文献

Bash info 页面（GNU Bash 参考手册）
https://www.gnu.org/software/bash/manual

· 第 7 节：作业控制

`bash`(1)、`builtins`(1)、`ps`(1)、`sleep`(1) man page

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

指导练习

控制作业

在本练习中，您将使用作业控制来启动和暂停多个进程，并将它们转到前台或后台。

成果

您应该能够使用作业控制来暂停和重启用户进程。

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-control start** 命令。此脚本确保 servera 可用。

```
[student@workstation ~]$ lab processes-control 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，这两个终端指代为左和右。在每个终端中，使用 **ssh** 命令以 student 用户身份登录 servera。

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

2. 在左窗口中，创建一个名为 **/home/student/bin** 的新目录。在新目录中，创建一个名为 **control** 的 shell 脚本。使脚本成为可执行文件。

   2.1. 使用 **mkdir** 命令，创建一个名为 **/home/student/bin** 的新目录。

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

   2.2. 使用 **vim** 命令，在 **/home/student/bin** 目录中创建一个名为 **control** 的脚本。要进入 Vim 交互模式，请按 **i** 键。使用 **:wq** 命令保存文件。

```
[student@servera ~]$ vim /home/student/bin/control
#!/bin/bash
while true; do
  echo -n "$@ " >> ~/control_outfile
  sleep 1
done
```

> **注意**
> 此 control 脚本会一直运行到被终止为止。它将每隔一秒向文件 ~/**control_outfile** 附加一次命令行参数。

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

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

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

3. 执行 **control** 脚本。该脚本以一秒为间隔，不断将词语“technical”和一个空格附加到文件 **~/control_outfile** 中。

> **注意**
> 您可以执行自己的 **control** 脚本，因为它已在 PATH 中，并且已变为可执行文件。

```
[student@servera ~]$ control technical
```

4. 在右终端 shell 中，使用 **tail** 命令及 **-f** 选项确认新进程正在写入 **/home/student/control_outfile** 文件。

```
[student@servera ~]$ tail -f ~/control_outfile
technical technical technical technical
...output omitted...
```

5. 在左终端 shell 中，按 **Ctrl+z** 来暂停正在运行的进程。shell 将返回作业 ID 并用方括号括起。在右窗口中，确认进程输出已经停止。

```
^Z
[1]+  Stopped                 control technical
[student@servera ~]$
```

```
technical technical technical technical
...no further output...
```

6. 在左终端 shell 中，查看 **jobs** 列表。记住 **+** 符号表示默认作业。在后台重新启动该作业。在右终端 shell 中，确认进程输出再次处于活动状态。

6.1. 使用 **jobs** 命令，查看作业列表。

```
[student@servera ~]$ jobs
[1]+  Stopped                 control technical
```

6.2. 使用 **bg** 命令，在后台重新启动 **control** 作业。

```
[student@servera ~]$ bg
[1]+ control technical &
```

6.3. 使用 **jobs** 命令，确认 **control** 作业再次处于运行状态。

```
[student@servera ~]$ jobs
[1]+  Running                 control technical &
```

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

6.4. 在右终端 shell 中，确认 **tail** 命令正在生成输出。

```
...output omitted...
technical technical technical technical technical technical technical technical
```

7. 在左终端 shell 中，再启动两个 **control** 进程，以便向 **~/output** 文件附加内容。使用与符号 (&) 在后台启动进程。将 **technical** 替换为 **documents**，然后替换为 **database**。替换参数有助于区分这三个进程。

```
[student@servera ~]$ control documents &
[2] 6579
[student@servera ~]$
[student@servera ~]$ control database &
[3] 6654
```

> **注意**
> 每个新进程的作业编号显示在方括号中。第二个数字是进程的系统范围唯一进程 ID 编号 (PID)。

8. 在左终端 shell 中，使用 **jobs** 命令查看正在运行的三个进程。在右终端 shell 中，确认所有三个进程都附加内容到文件中。

```
[student@servera ~]$ jobs
[1]   Running                 control technical &
[2]-  Running                 control documents &
[3]+  Running                 control database &


...output omitted...
technical documents database technical documents database technical documents
 database technical documents database
...output omitted...
```

9. 暂停 **control technical** 进程。确认它已被暂停。终止 **control documents** 进程，并确认它已被终止。

9.1. 在左终端 shell 中，使用 **fg** 命令及作业 ID 将 **control technical** 进程转到前台。按 **Ctrl+z** 暂停进程。使用 **jobs** 命令确认进程已被暂停。

```
[student@servera ~]$ fg %1
control technical
^Z
[1]+  Stopped                 control technical
[student@servera ~]$ jobs
[1]+  Stopped                 control technical
[2]   Running                 control documents &
[3]-  Running                 control database &
```

9.2. 在右终端 shell 中，确认 **control technical** 进程不再发送输出。

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

```
database documents  database documents  database
...no further output...
```

9.3. 在左终端 shell 中，使用 **fg** 命令及作业 ID 将 **control documents** 进程转到前台。按 **Ctrl+c** 终止进程。使用 **jobs** 命令确认进程已被终止。

```
[student@servera ~]$ fg %2
control documents
^C
[student@servera ~]$ jobs
[1]+  Stopped                 control technical
[3]-  Running                 control database &
```

9.4. 在右终端 shell 中，确认 **control documents** 进程不再发送输出。

```
...output omitted...
database database database database database database database database
...no further output...
```

10. 在左窗口中，使用 **ps** 命令及 **jT** 选项查看剩余的作业。已暂停作业的状态为 **T**。其他后台作业为睡眠状态 (**S**)。

```
[student@servera ~]$ ps jT
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 27277 27278 27278 27278 pts/1    28702 Ss    1000   0:00 -bash
 27278 28234 28234 27278 pts/1    28702 T     1000   0:00 /bin/bash /home/student/
bin/control technical
 27278 28251 28251 27278 pts/1    28702 S     1000   0:00 /bin/bash /home/student/
bin/control database
 28234 28316 28234 27278 pts/1    28702 T     1000   0:00 sleep 1
 28251 28701 28251 27278 pts/1    28702 S     1000   0:00 sleep 1
 27278 28702 28702 27278 pts/1    28702 R+    1000   0:00 ps jT
```

11. 在左窗口中，使用 **jobs** 命令查看当前作业。终止 **control database** 进程，并确认它已被终止。

```
[student@servera ~]$ jobs
[1]+  Stopped                 control technical
[3]-  Running                 control database &
```

使用 **fg** 命令及作业 ID 将 **control database** 进程转到前台。按 **Ctrl+c** 终止进程。使用 jobs 命令确认进程已被终止。

```
[student@servera ~]$ fg %3
control database
^C
[student@servera ~]$ jobs
[1]+  Stopped                 control technical
```

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

▶ **12.** 在右终端 shell 中，使用 **Ctrl+c** 命令来停止 **tail** 命令。使用 **rm** 命令来删除 **~/control_outfile** 文件。

```
...output omitted...
Ctrl+c
[student@servera ~]$ rm ~/control_outfile
```

▶ **13.** 在两个终端中从 servera 注销。

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

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

完成

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

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

本引导式练习到此结束。

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

中断进程

培训目标

学完本节后，您应能够：

· 使用命令中断进程并与进程通信。

· 确定守护进程的特征。

· 结束用户会话和进程。

使用信号控制进程

信号是传递至进程的软件中断。信号向执行中的程序报告事件。生成信号的事件可以是错误或外部事件（I/O 请求或定时器过期），或者来自于显式使用信号发送命令或键盘序列。

下表列出了系统管理员用于日常进程管理的基本信号。请通过短名称 (HUP) 或正确名称 (SIGHUP) 指代信号。

基本进程管理信号

信号编号	短名称	定义	用途
1	HUP	挂起	用于报告终端控制进程的终止。也用于请求进程重新初始化（重新加载配置）而不终止。
2	INT	键盘中断	导致程序终止。可以被拦截或处理。通过按 INTR 键序列 (**Ctrl**+**c**) 发送。
3	QUIT	键盘退出	与 SIGINT 相似；在终止时添加进程转储。通过按 QUIT 键序列 (**Ctrl**+****) 发送。
9	KILL	中断，无法拦截	导致立即终止程序。无法被拦截、忽略或处理；总是致命的。
15 默认	TERM	终止	导致程序终止。和 SIGKILL 不同，可以被拦截、忽略或处理。要求程序终止的“友好”方式；允许自我清理。
18	CONT	继续	发送至进程使其恢复（若已停止）。无法被拦截。即使被处理，也始终恢复进程。
19	STOP	停止，无法拦截	暂停进程。无法被拦截或处理。
20	TSTP	键盘停止	和 SIGSTOP 不同，可以被拦截、忽略或处理。通过按 SUSP 键序列 (**Ctrl**+**z**) 发送。

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

> **注意**
> 信号编号视不同的 Linux 硬件平台而异，但信号名称和含义都是标准化的。作为命令使用时，建议您使用信号名称而不是编号。本节中讨论的编号适用于 x86_64 系统。

每个信号都有一个默认操作，通常是如下之一：

- 终止 — 导致程序立即终止（退出）。
- 核心转储 — 导致程序保存内存镜像（核心转储），然后终止。
- 停止 — 导致程序停止执行（暂停），再等待继续（恢复）。

程序可以通过实施处理程序例程来为响应预期的事件信号做准备，以忽略、替换或扩展信号的默认操作。

通过明确请求发送信号的命令

您可以向当前的前台进程发送信号，具体操作为按下键盘控制序列以暂停 (**Ctrl**+**z**)、中止 (**Ctrl**+**c**) 或核心转储 (**Ctrl**+****) 该进程。不过，您将使用信号发送命令向后台进程或另一会话中的进程发送信号。

可以通过名称（例如 **-HUP** 或 **-SIGHUP**）或编号（相关的 **-1**）将信号指定为选项。用户可以中断自己的进程，但需要 root 特权才能终止由其他人拥有的进程。

kill 命令根据 PID 编号向进程发送信号。虽然其名称为 kill，但该命令可用于发送任何信号，而不仅仅是终止程序的信号。您可以使用 **kill -l** 命令列出所有可用信号的名称和编号。

```
[user@host ~]$ 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
...output omitted...
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user   5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job2
user   5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
user   5783  0.0  0.0 221860   964 pts/1    S+   16:43   0:00 grep --color=auto
 job
[1]   Terminated              control job1
[user@host ~]$ kill -9 5199
[user@host ~]$ ps aux | grep job
user   5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
```

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

```
user   5930  0.0  0.0 221860  1048 pts/1    S+   16:44   0:00 grep --color=auto
 job
[2]-  Killed                  control job2
[user@host ~]$ kill -SIGTERM 5205
user   5986  0.0  0.0 221860  1048 pts/1    S+   16:45   0:00 grep --color=auto
 job
[3]+  Terminated              control job3
```

killall 命令可以根据命令名称向多个进程发送信号。

```
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ killall control
[1]   Terminated              control job1
[2]-  Terminated              control job2
[3]+  Terminated              control job3
[user@host ~]$
```

使用 **pkill** 向一个或多个符合选择条件的进程发送信号。选择条件可以是命令名称、特定用户拥有的进程，或所有系统范围进程。**pkill** 命令包括高级选择条件：

- 命令 — 具有模式匹配的命令名称的进程。
- UID — 由某一 Linux 用户帐户拥有的进程，无论是有效的还是真实的。
- GID — 由某一 Linux 组帐户拥有的进程，无论是有效的还是真实的。
- 父级 — 特定父进程的子进程。
- 终端 — 运行于特定控制终端的进程。

```
[user@host ~]$ ps aux | grep pkill
user   5992  0.0  0.1 222448  3040 pts/1    S    16:59   0:00 /bin/bash /home/
user/bin/control pkill1
user   5996  0.0  0.1 222448  3048 pts/1    S    16:59   0:00 /bin/bash /home/
user/bin/control pkill2
user   6004  0.0  0.1 222448  3048 pts/1    S    16:59   0:00 /bin/bash /home/
user/bin/control pkill3
[user@host ~]$ pkill control
[1]   Terminated              control pkill1
[2]-  Terminated              control pkill2
[user@host ~]$ ps aux | grep pkill
user   6219  0.0  0.0 221860  1052 pts/1    S+   17:00   0:00 grep --color=auto
 pkill
[3]+  Terminated              control pkill3
[user@host ~]$ ps aux | grep test
user   6281  0.0  0.1 222448  3012 pts/1    S    17:04   0:00 /bin/bash /home/
user/bin/control test1
user   6285  0.0  0.1 222448  3128 pts/1    S    17:04   0:00 /bin/bash /home/
user/bin/control test2
user   6292  0.0  0.1 222448  3064 pts/1    S    17:04   0:00 /bin/bash /home/
user/bin/control test3
```

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

```
user    6318  0.0  0.0 221860  1080 pts/1    S+   17:04   0:00 grep --color=auto
 test
[user@host ~]$ pkill -U user
[user@host ~]$ ps aux | grep test
user    6870  0.0  0.0 221860  1048 pts/0    S+   17:07   0:00 grep --color=auto
 test
[user@host ~]$
```

以管理员身份注销用户

出于各种各样的原因，您可能需要注销其他用户。以下是许多可能性中的一些示例：用户做出了安全违规行为；用户可能过度使用了资源；用户的系统可能不响应；或者，用户不当访问了资料。在这些情况下，您可能需要以管理员身份使用信号来终止其会话。

要注销某个用户，首先确定要终止的登录会话。使用 **w** 命令列出用户登录和当前运行的进程。记录 **TTY** 和 **FROM** 列，以确定要关闭的会话。

所有用户登录会话都与某个终端设备 (TTY) 相关联。如果设备名称的格式为 **pts/N**，说明这是一个与图形终端窗口或远程登录会话相关联的伪终端。如果格式为 **ttyN**，则说明用户位于一个系统控制台、替代控制台或其他直接连接的终端设备上。

```
[user@host ~]$ w
 12:43:06 up 27 min,  5 users,  load average: 0.03, 0.17, 0.66
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty2                      12:26   14:58   0.04s  0.04s -bash
bob      tty3                      12:28   14:42   0.02s  0.02s -bash
user     pts/1    desk.example.com 12:41    2.00s  0.03s  0.03s w
[user@host ~]$
```

查看会话登录时间，了解用户已登录该系统的时长。对于每个会话，当前作业耗用的 CPU 资源（包括后台任务和子进程）位于 **JCPU** 列中。当前的前台进程 CPU 占用情况列在 **PCPU** 列中。

可以逐一向进程和会话发送信号，也可以集体发送。要终止一个用户的所有进程，可使用 **pkill** 命令。由于登录会话中的初始进程（会话首进程）设计为可以处理会话终止请求并忽略不想要的键盘信号，中断某一用户的所有进程和登录 shell 需要使用 SIGKILL 信号。

> **重要**
> 管理员通常太快使用 SIGKILL。
>
> 由于 SIGKILL 信号无法被处理或忽略，它总是致命的。然而，它会强行终止进程，而不允许被中断的进程运行自我清理例程。建议先发送 SIGTERM，然后尝试 SIGINT；只有在这两个进程都失败时通过 SIGKILL 重试。

首先使用 **pgrep** 确定要中断的 PID 编号，此操作与 **pkill** 非常相似，包括使用相同的选项，但 **pgrep** 是列出进程而非中断进程。

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

```
[root@host ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@host ~]# pkill -SIGKILL -u bob
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

当需要注意的进程在同一登录会话中时，可能不需要中断用户的所有进程。通过 **w** 命令确定会话的控制终端，然后仅中断引用同一终端 ID 的进程。除非指定了 **SIGKILL**，否则会话首进程（此处为 Bash 登录 shell）可以成功处理终止请求并保留，但所有其他会话进程将被终止。

```
[root@host ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@host ~]# w -h -u bob
bob      tty3      18:37    5:04   0.03s  0.03s -bash
[root@host ~]# pkill -t tty3
[root@host ~]# pgrep -l -u bob
7391 bash
[root@host ~]# pkill -SIGKILL -t tty3
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

也可利用父进程和子进程关系来应用相同的选择性进程终止。使用 **pstree** 命令查看系统或单个用户的进程树。使用父进程的 PID 中断其创建的所有子进程。此时，父进程 Bash 登录 shell 可以保留，因为信号仅定向至它的子进程。

```
[root@host ~]# pstree -p bob
bash(8391)─┬─sleep(8425)
           ├─sleep(8426)
           └─sleep(8427)
[root@host ~]# pkill -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]# pkill -SIGKILL -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]#
```

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

参考文献

`info libc signal` （GNU C 库参考手册）

· 第 24 节：信号处理

`info libc processes` （GNU C 库参考手册）

· 第 26 节：进程

`kill`(1)、`killall`(1)、`pgrep`(1)、`pkill`(1)、`pstree`(1)、`signal`(7) 和 `w`(1) man page

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

指导练习

中断进程

在本练习中，您将使用信号来管理和停止进程。

成果

您应能够启动和停止多个 shell 进程。

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-kill 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 processes-kill 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。在每个终端中，使用 **`ssh`** 命令以 `student` 用户身份登录 `servera`。

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

2. 在左窗口中，创建一个名为 **`/home/student/bin`** 的新目录。在新目录中，创建一个名为 **`killing`** 的 shell 脚本。使脚本成为可执行文件。

2.1. 使用 **`mkdir`** 命令，创建一个名为 **`/home/student/bin`** 的新目录。

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

2.2. 使用 **`vim`** 命令，在 **`/home/student/bin`** 目录中创建一个名为 **`killing`** 的脚本。按 **`i`** 键，以进入 Vim 交互模式。使用 **`:wq`** 命令保存文件。

```
[student@servera ~]$ vim /home/student/bin/killing
#!/bin/bash
while true; do
  echo -n "$@ " >> ~/killing_outfile
  sleep 5
done
```

> **注意**
> **`killing`** 脚本会一直运行到被终止为止。它将每隔 5 秒钟向 **`~/killing_outfile`** 附加一次命令行参数。

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

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

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

3. 在左终端 shell 中，使用 **cd** 命令更改到 **/home/student/bin/** 目录。分别使用 **network**、**interface** 和 **connection** 参数，启动三个 **killing** 进程。启动三个进程，分别名为 network、interface 和 connection。使用与符号 (**&**) 在后台启动进程。

```
[student@servera ~]$ cd /home/student/bin
[student@servera bin]$ killing network &
[1] 3460
[student@servera bin]$ killing interface &
[2] 3482
[student@servera bin]$ killing connection &
[3] 3516
```

您的进程具有不同的 PID 编号。

4. 在右终端 shell 中，使用 **tail** 命令及 **-f** 选项确认所有三个进程都附加内容到 **/home/student/killing_outfile** 文件中。

```
[student@servera ~]$ tail -f ~/killing_outfile
network interface network connection interface network connection interface
 network
...output omitted...
```

5. 在左终端 shell 中，使用 **jobs** 命令列出作业。

```
[student@servera bin]$ jobs
[1]   Running                 killing network &
[2]-  Running                 killing interface &
[3]+  Running                 killing connection &
```

6. 使用信号暂停 network 进程。确认 network 进程状态为 **Stopped**。在右终端 shell 中，确认 network 进程不再附加输出到 **~/killing_output**。

6.1. 使用 **kill** 及 **-SIGSTOP** 选项来停止 network 进程。运行 **jobs** 以确认它已被停止。

```
[student@servera bin]$ kill -SIGSTOP %1
[1]+  Stopped                 killing network
[student@servera bin]$ jobs
[1]+  Stopped                 killing network
[2]   Running                 killing interface &
[3]-  Running                 killing connection &
```

6.2. 在右终端 shell 中，查看来自 **tail** 命令的输出。确认词语 **network** 不再附加到 **~/killing_outfile** 文件。

```
...output omitted...
interface connection interface connection interface connection interface
```

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

7. 在左终端 shell 中，使用信号终止 `interface` 进程。确认 `interface` 进程已经消失。在右终端 shell 中，确认 `interface` 进程的输出不再附加到 **~/killing_outfile** 文件。

7.1. 使用 **kill** 命令及 **-SIGSTERM** 选项来终止 `interface` 进程。运行 **jobs** 命令确认它已被终止。

```
[student@servera bin]$ kill -SIGTERM %2
[student@servera bin]$ jobs
[1]+  Stopped                 killing network
[2]   Terminated              killing interface
[3]-  Running                 killing connection &
```

7.2. 在右终端 shell 中，查看来自 **tail** 命令的输出。确认词语 **interface** 不再附加到 **~/killing_outfile** 文件。

```
...output omitted...
connection connection connection connection connection connection connection
 connection
```

8. 在左终端 shell 中，使用信号恢复 `network` 进程。确认 `network` 进程状态为 **Running**。在右侧窗口中，确认 `network` 进程输出正在附加到 **~/killing_outfile** 文件。

8.1. 使用 **kill** 命令及 **-SIGCONT** 选项来恢复 `network` 进程。使用 **jobs** 命令确认该进程的状态为 **Running**。

```
[student@servera bin]$ kill -SIGCONT %1
[student@servera bin]$ jobs
[1]+  Running                 killing network &
[3]-  Running                 killing connection &
```

8.2. 在右终端 shell 中，查看来自 **tail** 命令的输出。确认词语 **network** 正在附加到 **~/killing_outfile** 文件。

```
...output omitted...
network connection network connection network connection network connection
 network connection
```

9. 在左终端 shell 中，终止剩余的两个作业。确认没有剩余的作业，输出也已停止。

9.1. 使用 **kill** 命令及 **-SIGTERM** 选项来终止 `network` 进程。使用同样的命令终止 `connection` 进程。

```
[student@servera bin]$ kill -SIGTERM %1
[student@servera bin]$ kill -SIGTERM %3
[1]+  Terminated              killing network
[student@servera bin]$ jobs
[3]+  Terminated              killing connection
```

10. 在左终端 shell 中，列出在所有打开的终端 shell 中运行的 `tail` 进程。终止正在运行的 tail 命令。确认该进程不再运行。

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

10.1.使用 **ps** 命令及 **-ef** 选项来列出所有运行中的 tail 进程。使用 **grep** 命令来优化搜索。

```
[student@servera bin]$ ps -ef | grep tail
student   4581 31358  0 10:02 pts/0    00:00:00 tail -f killing_outfile
student   4869  2252  0 10:33 pts/1    00:00:00 grep --color=auto tail
```

10.2.使用 **pkill** 命令及 **-SIGTERM** 选项来中断 tail 进程。使用 **ps** 确认它已不再存在。

```
[student@servera bin]$ pkill -SIGTERM tail
[student@servera bin]$ ps -ef | grep tail
student   4874  2252  0 10:36 pts/1    00:00:00 grep --color=auto tail
```

10.3.在右终端 shell 中，确认 **tail** 命令不再运行。

```
...output omitted...
network connection network connection network connection Terminated
[student@servera ~]$
```

11. 从两个终端窗口退出。不从所有会话退出将导致完成脚本运行失败。

```
[student@servera bin]$ 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-kill finish** 脚本来完成本练习。

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

本引导式练习到此结束。

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

监控进程活动

培训目标

学完本节后，您应能够描述负载平均值的定义，并确定对服务器上高资源使用量负责的进程。

描述负载平均值

负载平均值是 Linux 内核提供的一种度量方式，它可以简单地表示一段时间内感知的系统负载。您可以用它来粗略衡量待处理的系统资源请求数量，并确定系统负载是随时间增加还是减少。

根据处于可运行和不可中断状态的进程数，内核会每五秒钟收集一次当前的负载数。通过汇总这些数值，可以得到最近 1 分钟、5 分钟和 15 分钟内的指数移动平均值。

了解 Linux 负载平均值计算

负载平均值代表一段时间内感知的系统负载。通过报告 CPU 上准备运行的进程数以及等待磁盘或网络 I/O 完成的进程数，Linux 可以确定负载平均值。

- 负载数基本上是根据准备运行的进程数（进程状态为 **R**）和等待 I/O 完成的进程数（进程状态为 **D**）而得到的。
- 一些 UNIX 系统仅考虑 CPU 使用率或运行队列长度来指示系统负载。Linux 还包含磁盘或网络利用率，因为它们与 CPU 负载一样会对系统性能产生重大影响。遇到负载平均值很高但 CPU 活动很低时，请检查磁盘和网络活动。

负载平均值可粗略衡量在可以执行其他任何作业之前，有多少进程当前在等待请求完成。请求可能是用于运行进程的 CPU 时间。或者，请求可能是让关键磁盘 I/O 操作完成；在请求完成之前，即使 CPU 空闲，也不能在 CPU 上运行该进程。无论是哪种方式，都会影响系统负载；系统的运行看起来会变慢，因为有进程正在等待运行。

解读显示的负载平均值

uptime 命令是显示当前负载平均值的一种方法。它可显示当前时间、计算机启动时长、运行的用户会话数以及当前的负载平均值。

```
[user@host ~]$ uptime
 15:29:03 up 14 min,  2 users,  load average: 2.92, 4.48, 5.20
```

这三个负载平均值代表了最近 1、5 和 15 分钟的负载情况。一眼看去就可指出系统负载似乎在增高还是降低。

如果等待 CPU 处理的进程是负载平均值的主要贡献因素，则可以计算近似的每 CPU 负载值以判断系统是否在遭遇显著的等待。

lscpu 命令可以帮助您确定系统有多少 CPU。

在以下示例中，系统是一个双核单套接字系统，每个内核有两个超线程。粗略地说，Linux 会出于调度目的将此视为有四个 CPU 的系统。

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

```
[user@host ~]$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  2
Core(s) per socket:  2
Socket(s):           1
NUMA node(s):        1
...output omitted...
```

暂时不妨设想，需要 CPU 时间的进程是负载数的唯一贡献因素。然后您可以用显示的负载平均值除以系统中的逻辑 CPU 数。值低于 1 表示资源利用率较好，等待时间很短。值高于 1 表示资源饱和，而且有一定的处理延迟。

```
# From lscpu, the system has four logical CPUs, so divide by 4:
#                              load average: 2.92, 4.48, 5.20
#           divide by number of logical CPUs:    4     4     4
#                                              ----  ----  ----
#                       per-CPU load average: 0.73  1.12  1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.
# During the last 5 minutes, the system was overloaded by ~12%.
# During the last 15 minutes, the system was overloaded by ~30%.
```

空闲 CPU 队列的负载数为 0。每个等待 CPU 处理的进程都会使负载数加 1。如果有一个进程在 CPU 上运行，则负载数为 1；尽管资源 (CPU) 处于使用状态，但没有等待的请求。如果该进程运行了整整一分钟，那么它对这一分钟负载平均值的贡献就是 1。

不过，因为磁盘或网络资源忙碌而等待关键 I/O 并处于不可中断睡眠状态的进程也包含在该计数内，而且使负载平均值增大。虽然不能表示 CPU 使用率，但这些进程也被添加到队列数中，因为它们正在等待资源，并且在获取资源之前无法在 CPU 上运行。由于资源限制导致进程无法运行，因此这仍然是系统负载。

在资源饱和前，负载平均值将保持在 1 以下，因为几乎不会在队列中发现等待的任务。只有资源饱和导致请求留在排队状态并且被负载计算例程计数时，负载平均值才会增大。当资源使用率接近 100% 时，每个增加的请求将开始遭遇服务等待时间。

还有许多工具也可报告负载平均值，包括 **w** 和 **top**。

实时进程监控

top 程序是系统进程的动态视图，显示一个摘要标题，以及与 **ps** 信息类似的进程或线程列表。与静态的 **ps** 输出不同，**top** 以可配置的间隔持续刷新，而且也提供列重新排列、排序和突出显示功能。用户配置可以保存，变为永久。

默认输出列可以从其他资源工具识别：

- 进程 ID ((**PID**)。
- 用户名称 (**USER**) 是进程所有者。

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

- 虚拟内存 (**VIRT**) 是进程正在使用的所有内存，包括常驻集合、共享库，以及任何映射或交换的内存页。（**ps** 命令中标为 **VSZ**。）
- 常驻内存 (**RES**) 是进程所用的物理内存，包括任何驻留的共享对象。（**ps** 命令中标为 **RSS**。）
- 进程状态 (**S**) 显示为：
  - **D** = 不可中断睡眠
  - **R** = 运行中或可运行
  - **S** = 睡眠中
  - **T** = 已停止或已跟踪
  - **Z** = 僵停
- CPU 时间 (**TIME**) 是进程启动以来总的处理时间。可以切换为包含所有过去子进程的累计时间。
- 进程命令名称 (**COMMAND**)。

top 中的基本击键操作

按键	用途
? 或 h	交互式击键操作的帮助。
l、t、m	切换到负载、线程和内存标题行。
1	标题中切换显示单独 CPU 信息或所有 CPU 的汇总。
s (1)	更改刷新（屏幕）率，以带小数的秒数表示（如 0.5、1、5）。
b	切换反向突出显示运行中的进程；默认为仅粗体。
转移+ **b**	在显示中使用粗体，用于标题以及运行中的进程。
Shift+h	切换线程；显示进程摘要或单独线程。
u、Shift+u	过滤任何用户名称（有效、真实）
Shift+m	按照内存使用率，以降序方式对进程列表排序。
Shift+p	按照处理器使用率，以降序方式对进程列表排序。
k (1)	中断进程。若有提示，输入 **PID**，再输入 **signal**。
r (1)	调整进程的 nice 值。若有提示，输入 **PID**，再输入 **nice_value**。
Shift+w	写入（保存）当前的显示配置，以便在下一次重新启动 **top** 时使用。
q	退出。
f	通过启用或禁用字段的方式来管理列。同时还允许您为 **top** 设置排序字段。
注：	(1) 如果 top 在安全模式中启动，则不可用。请参见 **top** 帮助。

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

参考文献

`ps`(1)、`top`(1)、`uptime`(1) 和 `w`(1) man page

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

指导练习

监控进程活动

在本练习中，您将使用 **top** 命令来动态检查正在运行的进程并控制它们。

成果

您应能够实时管理进程。

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-monitor 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 processes-monitor start
```

1. 在 workstation 上，打开两个终端窗口并将它们并排显示。这些终端指代为左和右。在每个终端中，使用 **ssh** 命令以 student 用户身份登录 servera。

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

2. 在左终端 shell 中，创建一个名为 **/home/student/bin** 的新目录。在新目录中，创建一个名为 **monitor** 的 shell 脚本。确保脚本可以执行。

2.1. 使用 **mkdir** 命令，创建一个名为 **/home/student/bin** 的新目录。

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

2.2. 使用 **vim** 命令，在 **/home/student/bin** 目录中创建一个名为 **monitor** 的脚本。按 **i** 键，以进入 Vim 交互模式。使用 **:wq** 命令保存文件。

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

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