```
#!/bin/bash
#
USR='student'
OUT='/home/student/output'
#
for SRV in servera serverb
  do
ssh ${USR}@${SRV} "hostname -f" > ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
ssh ${USR}@${SRV} "lscpu | grep '^CPU'" >> ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
ssh ${USR}@${SRV} "grep -v '^$' /etc/selinux/config|grep -v '^#'" >> ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
ssh ${USR}@${SRV} "sudo grep 'Failed password' /var/log/secure" >> ${OUT}-${SRV}
echo "#####" >> ${OUT}-${SRV}
done
```

3. 执行 **/home/student/bin/bash-lab** 脚本，然后在 workstation 上查看输出内容。

   3.1. 在 workstation 上，执行 **/home/student/bin/bash-lab** 脚本。

```
[student@workstation ~]$ bash-lab
```

   3.2. 检查 **/home/student/output-servera** 和 **/home/student/output-serverb** 的内容。

```
[student@workstation ~]$ cat /home/student/output-servera
servera.lab.example.com
#####
CPU op-mode(s):       32-bit, 64-bit
CPU(s):               2
CPU family:           21
CPU MHz:              2294.670
#####
SELINUX=enforcing
SELINUXTYPE=targeted
#####
Mar 21 22:30:28 servera sshd[3939]: Failed password for invalid user operator1
 from 172.25.250.9 port 58382 ssh2
Mar 21 22:30:31 servera sshd[3951]: Failed password for invalid user sysadmin1
 from 172.25.250.9 port 58384 ssh2
Mar 21 22:30:34 servera sshd[3953]: Failed password for invalid user manager1 from
 172.25.250.9 port 58386 ssh2
#####


[student@workstation ~]$ cat /home/student/output-serverb
serverb.lab.example.com
#####
CPU op-mode(s):       32-bit, 64-bit
CPU(s):               2
CPU family:           6
```

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

```
CPU MHz:                2294.664
#####
SELINUX=enforcing
SELINUXTYPE=targeted
#####
Mar 21 22:30:37 serverb sshd[3883]: Failed password for invalid user operator1
 from 172.25.250.9 port 39008 ssh2
Mar 21 22:30:39 serverb sshd[3891]: Failed password for invalid user sysadmin1
 from 172.25.250.9 port 39010 ssh2
Mar 21 22:30:43 serverb sshd[3893]: Failed password for invalid user manager1 from
 172.25.250.9 port 39012 ssh2
#####
```

## 评估

在 workstation 上，运行 **lab console-review grade** 命令来确认是否成功完成本练习。

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

## 完成

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

```
[student@workstation ~]$ lab console-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：

- 如何创建和执行简单的 Bash 脚本。
- 如何使用循环来迭代命令行和 shell 脚本中的项目列表。
- 如何使用正则表达式和 **grep** 搜索日志文件和配置文件中的文本。

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

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

章 2

# 计划将来的任务

**目标**

规划任务以便在将来自动执行。

**培训目标**

- 设置一个在将来某个时刻运行一次的命令。
- 利用用户的 crontab 文件，安排命令按计划重复运行。
- 利用系统 crontab 文件和目录，安排命令按计划重复运行。
- 启用和禁用 systemd 定时器并配置管理临时文件的定时器。

**章节**

- 计划延迟的用户作业（及引导式练习）
- 计划周期性用户作业（及引导式练习）
- 计划周期性系统作业（及引导式练习）
- 管理临时文件（及引导式练习）

**实验**

计划将来的任务

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

# 计划延迟的用户作业

## 培训目标

学完本节后，您应能够设置一个在将来某个时刻运行一次的命令。

## 描述延迟的用户任务

有时，您可能需要在将来某个设定的点运行一个命令或一组命令。例如，员工计划要向老板发送电子邮件，或正在处理防火墙配置的系统管理员要落实在十分钟内重置防火墙设置的“安全”作业（除非他们提前停用该作业）。

这些计划好的命令通常被称为任务或作业，而延迟一词则表示这些任务或作业将在未来运行。

针对红帽企业 Linux 用户安排延迟任务，可用的解决方案之一为 at。at 软件包中提供了 (atd) 系统守护进程，以及一组与守护进程交互的命令行工具（**at**、**atq** 等等）。在默认的红帽企业 Linux 安装过程中，将自动安装并启用 atd 守护进程。

用户（包括 root）可以使用 **at** 命令为 atd 守护进程的作业排队。atd 守护进程提供了 **a** 到 **z** 共 26 个队列，作业按字母顺序排列，队列越后，系统优先级越低（后面章节将讨论更高的 nice 值）。

## 计划延迟的用户任务

使用 **at TIMESPEC** 命令可以计划新的作业。然后 **at** 命令从 stdin 通道中读取要执行的命令。手动输入命令时，您可以按 **Ctrl+D** 来完成输入。对于易出现印刷错误的复杂命令，通常借助脚本文件使用输入重定向会更为方便（例如：**at now +5min < myscript**），而非在终端窗口中手动输入所有命令。

**TIMESPEC** 参数与 **at** 命令可以接受许多强大组合，从而允许用户准确描述作业运行的时间。通常，这些组合以时间（例如 **02:00pm**、**15:59** 甚至 **teatime**）开头，后面接一个可选日期或将来的天数。下文列出了一些可以使用的组合示例。

- **now +5min**
- **teatime tomorrow**（下午茶时间为 **16:00**）
- **noon +4 days**
- **5pm august 3 2021**

如需有效时间规范的完整列表，请参阅参考资料中所列的 **timespec** 定义。

## 检查和管理延迟的用户作业

要获得当前用户的待处理作业的概述，请使用命令 **atq** 或 **at -l** 命令。

```
[user@host ~]$ atq
❶28  ❷Mon Feb  2 05:13:00 2015 ❸a ❹user
29  Mon Feb  3 16:00:00 2014 h user
27  Tue Feb  4 12:00:00 2014 a user
```

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

在上述输出中，每一行都代表一个计划在将来运行的不同作业。

1. 该作业的唯一作业编号。
2. 计划作业的执行日期和时间。
3. 表示作业已排入默认队列 a。不同的作业可以排入不同的队列中。
4. 作业所有者（以及将运行作业的用户身份）。

**重要**
非特权用户只能查看和控制自己的作业。root 用户可以查看和管理所有作业。

要检查在执行作业时将运行的实际命令，请使用 **at -c JOBNUMBER** 命令。该命令会显示所设置作业的环境，以便在用户创建作业时反映用户的环境，后跟将运行的实际命令。

## 删除作业

**atrm JOBNUMBER** 命令将会删除计划的作业。当不再需要计划的作业时（例如，远程防火墙配置成功且不需要重置时），可将其删除。

**参考文献**
**at**(1) 和 **atd**(8) man page

**/usr/share/doc/at/timespec**

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

## 指导练习

# 计划延迟的用户作业

在本练习中，您将使用 **at** 命令来计划在未来指定时间运行多个命令。

### 成果

您应能够：

- 计划在未来的指定时间运行作业。
- 检查计划作业所运行的命令。
- 删除计划的作业。

### 在你开始之前

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

在 workstation上，运行 **lab scheduling-at start** 开始本练习。此脚本会确保环境清理干净且设置正确无误。

```
[student@workstation ~]$ lab scheduling-at start
```

**1.** 从 workstation，以 student 用户身份打开连接 servera 的 SSH 会话。

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

**2.** 使用 **at** 命令计划从现在起三分钟后运行一项作业。该作业必须将 **date** 命令的输出保存至 **/home/student/myjob.txt**。

2.1. 使用 **echo** 命令将字符串 **date >> /home/student/myjob.txt** 作为输入传给 **at** 命令，以便从现在起三分钟后运行作业。

```
[student@servera ~]$ echo "date >> /home/student/myjob.txt" | at now +3min
warning: commands will be executed using /bin/sh
job 1 at Thu Mar 21 12:30:00 2019
```

2.2. 使用 **atq** 命令列出计划的作业。

```
[student@servera ~]$ atq
1 Thu Mar 21 12:30:00 2019 a student
```

2.3. 使用 **watch atq** 命令实时监控延迟作业的队列。执行完后，该作业将从队列中删除。

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

```
[student@servera ~]$ watch atq
Every 2.0s: atq          servera.lab.example.com: Thu Mar 21 12:30:00 2019

1 Thu Mar 21 12:30:00 2019 a student
```

默认情况下，上述 **watch** 命令将每隔两秒钟更新一次 **atq** 的输出。从队列中删除延迟作业后，按 **Ctrl+c** 退出 **watch** 并返回到 shell 提示符。

2.4. 使用 **cat** 命令验证 **/home/student/myjob.txt** 的内容是否与 **date** 命令的输出相匹配。

```
[student@servera ~]$ cat myjob.txt
Thu Mar 21 12:30:00 IST 2019
```

上述输出与 **date** 命令的输出相匹配，从而确认计划作业已成功执行。

**3.** 使用 **at** 命令以交互方式将作业排入将在 **teatime** (16:00) 运行的队列 **g**。该作业将执行向 **/home/student/tea.txt** 中显示消息 **It's teatime** 的命令。新消息应附加至文件 **/home/student/tea.txt**。

```
[student@servera ~]$ at -q g teatime
warning: commands will be executed using /bin/sh
at> echo "It's teatime" >> /home/student/tea.txt
at> Ctrl+d
job 2 at Thu Mar 21 16:00:00 2019
```

**4.** 使用 **at** 命令以交互方式将另一作业排入将在 **16:05** 运行的队列 **b**。该作业将执行向 **/home/student/cookies.txt** 中显示消息 **The cookies are good** 的命令。新消息应附加至文件 **/home/student/cookies.txt**。

```
[student@servera ~]$ at -q b 16:05
warning: commands will be executed using /bin/sh
at> echo "The cookies are good" >> /home/student/cookies.txt
at> Ctrl+d
job 3 at Thu Mar 21 16:05:00 2019
```

**5.** 检查待处理作业中的命令。

5.1. 使用 **atq** 命令查看待处理作业的作业编号。

```
[student@servera ~]$ atq
2 Thu Mar 21 16:00:00 2019 g student
3 Thu Mar 21 16:05:00 2019 b student
```

记下上述输出中的作业编号。这些作业编号可能会因系统而异。

5.2. 使用 **at** 命令查看待处理作业编号 2 中的命令。

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

```
[student@servera ~]$ at -c 2
...output omitted...
echo "It's teatime" >> /home/student/tea.txt
marcinDELIMITER28d54caa
```

请注意，上述计划作业会执行 **echo** 命令，它将把消息 **It's teatime** 附加至 **/home/student/tea.txt**。

5.3. 使用 **at** 命令查看待处理作业编号 3 中的命令。

```
[student@servera ~]$ at -c 3
...output omitted...
echo "The cookies are good" >> /home/student/cookies.txt
marcinDELIMITER1d2b47e9
```

请注意，上述计划作业会执行 **echo** 命令，它将把消息 **The cookies are good** 附加至 **/home/student/cookies.txt**。

**6.** 使用 **atq** 命令查看将在 **teatime** (16:00) 运行的作业的作业编号，并使用 **atrm** 命令将该作业删除。

```
[student@servera ~]$ atq
2 Thu Mar 21 16:00:00 2019 g student
3 Thu Mar 21 16:05:00 2019 b student
[student@servera ~]$ atrm 2
```

**7.** 验证计划在 **teatime** (16:00) 运行的作业已不复存在。

7.1. 使用 **atq** 命令查看待处理作业的列表，确认计划在 **teatime** (16:00) 运行的作业已不复存在。

```
[student@servera ~]$ atq
3 Thu Mar 21 16:05:00 2019 b student
```

7.2. 从 servera 注销。

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

## 完成

在 workstation 上，运行 **lab scheduling-at finish** 完成本练习。此脚本将删除在实验过程中创建的文件，并确保环境清理干净。

```
[student@workstation ~]$ lab scheduling-at finish
```

本引导式练习到此结束。

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

# 计划周期性用户作业

## 培训目标

学完本节后，您应能够利用用户的 crontab 文件来安排命令按计划重复运行。

## 描述周期性用户作业

按计划重复运行的作业被称为周期性作业。红帽企业 Linux 系统附带了特别针对周期性作业的 crond 守护进程，由 cronie 软件包提供，且默认启用并启动。crond 守护进程会读取多个配置文件（每个用户对应一个配置文件，使用 **crontab**(1) 命令进行编辑）以及一组系统范围内的文件。这些配置文件使用户和管理员拥有细微的控制权，可以控制应执行周期性作业的时间。

如果计划的命令生成了任何未被重定向的输出或错误，则 crond 守护进程将尝试使用系统中配置的邮件服务器将该输出或错误通过电子邮件发送给拥有该作业的用户（除非被覆盖）。根据环境，这可能需要进行其他配置。可以将计划命令的输出或错误重定向到其他文件。

## 计划周期性用户作业

普通用户可以使用 **crontab** 命令来管理作业。可以通过四种方式调用该命令：

### Crontab 示例

| 命令 | 预期用途 |
|---|---|
| **crontab -l** | 列出当前用户的计划作业。 |
| **crontab -r** | 删除当前用户的所有作业。 |
| **crontab -e** | 编辑当前用户的作业。 |
| **crontab filename** | 删除所有作业，并替换为从 filename 读取的作业。如果没有指定文件，则使用 **stdin**。 |

**注意**
超级用户可以对 **crontab** 命令使用 **-u** 选项，以此来管理其他用户的作业。不应使用 **crontab** 命令来管理系统作业；而应使用下一节中介绍的方法。

## 描述用户作业格式

除非 EDITOR 环境变量采用了不同的设置，否则 **crontab -e** 命令默认会调用 Vim。每行只输入一个作业。其他有效条目包括：空行，通常为了便于阅读；注释，由以数字符号 (#) 开头的行标识；采用格式 **NAME=value** 的环境变量，它会影响相应声明行下面的所有行。常见变量设置包括 SHELL 变量和 MAILTO 变量，前者声明用于解释 crontab 文件其余行的 shell，后者确定由谁来接收通过电子邮件发送的输出。

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

> **重要**
> 发送电子邮件可能需要另外配置本地邮件服务器或系统中的 SMTP 转发。

**crontab** 文件中的字段按以下顺序显示：

- 分钟
- 小时
- 几号
- 月
- 星期几
- 命令

> **重要**
> 如果几号和星期几字段都不是 *****，则该命令将在其中任一字段满足时执行。例如，要在每月 15 号或每周五 **12:15** 运行命令，请使用以下作业格式：
>
> ```
> 15 12 15 * Fri command
> ```

前五个字段全部使用相同的语法规则：

- ***** 表示“无关紧要”/始终。
- 数字可用于指定分钟数或小时数、日期或者工作日。对于工作日，**0** 表示星期日，**1** 表示星期一，**2** 表示星期二，依此类推。**7** 也表示星期日。
- **x-y** 表示范围，**x** 到 **y**（含）。
- **x,y** 表示列表。列表也可以包含范围，例如分钟列中的 **5,10-13,17** 表示作业应当在每小时过去 5 分钟、10 分钟、11 分钟、12 分钟、13 分钟和 17 分钟后运行。
- ***/x** 表示 **x** 的时间间隔，例如分钟列中的 ***/7** 表示每七分钟运行一次作业。

此外，可以使用三个字母的英文缩写来表示月份和工作日，例如 Jan、Feb 以及 Mon、Tue。

最后一个字段包含使用默认 shell 执行的命令。**SHELL** 环境变量可用于更改计划命令的 shell。如果命令包含未转义的百分比符号 (%)，则该百分比符号将被当作换行字符，且百分比符号之后的所有内容将传给 **stdin** 中的命令。

## 周期性用户作业示例

本节介绍周期性作业的一些示例。

- 以下作业将在每年 2 月 2 日上午 9 点准点执行命令 **/usr/local/bin/yearly_backup**。

  ```
  0 9 2 2 * /usr/local/bin/yearly_backup
  ```

- 以下作业将在七月每周五的上午 9 点和下午 5 点间，每五分钟向该作业的所有者发送包含单词 **Chime** 的电子邮件。

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

```
*/5 9-16 * Jul 5 echo "Chime"
```

上文中 **9-16** 的小时范围表示作业定时器将在 9 点 (09:00) 启动并一直持续到 16 点结束 (16:59)。作业将从 **09:00** 开始执行，在 **16:55** 结束执行，因为 **16:55** 再过五分钟是 **17:00**，而这超出了给定的小时范围。

- 以下作业将在每个工作日午夜前的两分钟运行命令 **/usr/local/bin/daily_report**。

```
58 23 * * 1-5 /usr/local/bin/daily_report
```

- 以下作业将在每个工作日（周一到周五）上午 9 点执行 **mutt** 命令，从而将邮件消息 **Checking in** 发送给收件人 **boss@example.com**。

```
0 9 * * 1-5 mutt -s "Checking in" boss@example.com % Hi there boss, just checking
 in.
```

**参考文献**

**crond**(8)、**crontab**(1) 和 **crontab**(5) man page

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

## 指导练习

# 计划周期性用户作业

在本练习中，您将以无特权用户的身份，使用 **crontab** 命令来安排按计划重复运行的命令。

### 成果

您应能够：

- 以无特权用户的身份来计划周期性作业。
- 检查计划的周期性作业所运行的命令。
- 删除计划的周期性作业。

### 在你开始之前

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

在 workstation 上，运行 **lab scheduling-cron start** 开始本练习。此脚本会确保环境清理干净且设置正确无误。

```
[student@workstation ~]$ lab scheduling-cron start
```

**1.** 从 workstation，以 student 用户身份打开连接 servera 的 SSH 会话。

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

**2.** 以 student 身份计划周期性作业，在上午 8 点到晚上 9 点之间每两分钟一次将当前日期和时间附加到 **/home/student/my_first_cron_job.txt**。该作业只能在周一到周五运行，周六或周日不能运行。

> **重要**
> 如果打算在上述说明中所提及的日期和时间之外进行本实验，则应相应地调整系统时间和/或日期，以便在您工作时运行本作业。

2.1. 使用 **crontab -e** 命令，以默认文本编辑器打开 crontab。

```
[student@servera ~]$ crontab -e
```

2.2. 插入以下行。

```
*/2 08-20 * * Mon-Fri /usr/bin/date >> /home/student/my_first_cron_job.txt
```

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

2.3. 在文本编辑器中，按 **Esc** 并输入 **:wq** 以保存更改并退出编辑器。退出编辑器时，您应看到以下输出：

```
...output omitted...
crontab: installing new crontab
[student@servera ~]$
```

上述输出确认已成功计划作业。

**3.** 使用 **crontab -l** 命令列出计划的周期性作业。检查上一步中计划作为周期性作业运行的命令。

```
[student@servera ~]$ crontab -l
*/2 08-20 * * * /usr/bin/date >> /home/student/my_first_cron_job.txt
```

请注意，上述计划作业将运行 **/usr/bin/date** 命令并将其输出附加至 **/home/student/my_first_cron_job.txt**。

**4.** 使用 **while** 命令，以便在因成功执行计划的周期性作业而创建文件 **/home/student/my_first_cron_job.txt** 前，您的 shell 提示符处于睡眠状态。等待您的 shell 提示符返回。

```
[student@servera ~]$ while ! test -f my_first_cron_job.txt; do sleep 1s; done
```

上述 **while** 命令会使用 **! test -f** 继续循环运行 **sleep 1s** 命令，直到在 **/home/student** 目录中创建 **my_first_cron_job.txt** 文件。

**5.** 使用 **cat** 命令验证 **/home/student/my_first_cron_job.txt** 的内容是否与 **date** 命令的输出相匹配。

```
[student@servera ~]$ cat my_first_cron_job.txt
Fri Mar 22 13:56:01 IST 2019
```

上述输出可能在您的系统上有所不同。

**6.** 删除计划以 student 用户身份运行的所有周期性作业。

6.1. 使用 **crontab -r** 命令删除为 student 计划的所有周期性作业。

```
[student@servera ~]$ crontab -r
```

6.2. 使用 **crontab -l** 命令验证已不存在 student 的周期性作业。

```
[student@servera ~]$ crontab -l
no crontab for student
```

6.3. 从 servera 注销。

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 scheduling-cron finish** 来完成本练习。此脚本将删除在实验过程中创建的文件，并确保环境清理干净。

```
[student@workstation ~]$ lab scheduling-cron finish
```

本引导式练习到此结束。

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

# 计划周期性系统作业

## 培训目标

学完本节后，您应能够利用系统 crontab 文件和目录来安排命令按计划重复运行。

## 描述周期性系统作业

系统管理员经常需要运行周期性作业。最佳做法是从系统帐户而不是从用户帐户运行这些作业。也就是说，不要使用 **crontab** 命令来安排运行这些作业，而是要使用系统范围的 crontab 文件。系统范围的 crontab 文件中的作业条目与用户的 crontab 条目类似，唯一的不同是系统范围的 crontab 文件的命令字段前面有一个额外用户字段；此命令应在该用户的授权下运行。

**/etc/crontab** 文件的随附注释中包含实用的语法图。

```
 # For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue ...
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

周期性系统作业在两个位置定义：**/etc/crontab** 文件和 **/etc/cron.d/** 目录中的文件。要计划周期性系统作业，应始终在 **/etc/cron.d** 目录下创建自定义 crontab 文件。通过将自定义 crontab 文件放入 **/etc/cron.d**，可以防止在 **/etc/crontab** 的供应商进行软件包更新时将它覆盖，因为更新可能会覆盖 **/etc/crontab** 中的现有内容。需要周期性系统作业的软件包会将其 crontab 文件放入包含作业条目的 **/etc/cron.d/** 中。管理员也使用此位置将相关作业组合到单个文件中。

crontab 系统中还包含需要每小时、每天、每周和每月运行的脚本的存储库。这些存储库分别对应于名为 **/etc/cron.hourly/**、**/etc/cron.daily/**、**/etc/cron.weekly/** 和 **/etc/cron.monthly/** 的目录。再次申明，这些目录中包含可执行的 shell 脚本，而不是 crontab 文件。

> **重要**
> 记得使置于这些目录中的所有脚本可执行。如果某个脚本不可执行，则它不会运行。要使脚本可执行，请使用 **chmod +x script_name**。

一个从 **/etc/cron.d/0hourly** 文件调用的名为 **run-parts** 的命令可以运行 **/etc/cron.hourly/*** 脚本。**run-parts** 命令也运行每日、每周和每月的作业，但它是从另一个名为 **/etc/anacrontab** 的配置文件调用的。

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

> **注意**
> 过去，曾使用一个名为 anacron 的独立服务来处理 **/etc/anacrontab** 文件，但在红帽企业 Linux 7 及更高版本中，使用常规 crond 服务来解析此文件。

**/etc/anacrontab** 旨在确保重要的作业始终运行，且不会因为系统在应执行作业时关闭或休眠而意外跳过。例如，如果由于系统重启而导致某个每日运行的系统作业上次未按时执行，则在系统就绪后就会执行此作业。但是，启动作业时可能会有几分钟的延迟，具体要视 **/etc/anacrontab** 中为该作业指定的 **Delay in minutes** 参数值而定。

针对各个每日、每周和每月作业，**/var/spool/anacron/** 中会有不同的对应文件，以确定特定的作业是否已运行。当 crond 从 **/etc/anacrontab** 中启动作业时，它会更新这些文件的时间戳。同一时间戳也用于确定作业上次运行的时间。**/etc/anacrontab** 的语法与其他常规 **crontab** 配置文件有所不同。它每行正好包含四个字段，如下所示。

- 以天为单位

  按重复计划运行的作业的间隔天数。该字段接受整数或宏作为值。例如，宏 **@daily** 相当于整数 **1**，它表示每天执行该作业。同样，宏 **@weekly** 相当于整数 **7**，它表示每周执行该作业。

- 延迟（分钟）

  在启动该作业前，crond 守护进程应等待的时间。

- 工作号

  在日志消息中用于标识作业的唯一名称。

- 命令

  要执行的命令。

**/etc/anacrontab** 文件中还包含使用语法 **NAME=value** 的环境变量声明。特别要注意其中的变量 START_HOURS_RANGE，它指定了作业运行的时间间隔。超出此范围时，作业将不会启动。如果某一天作业在此时间间隔内未运行，则必须等到第二天才能执行此作业。

# SYSTEMD 定时器简介

随着红帽企业 Linux 7 中 systemd 的出现，一种新的调度功能现已推出：systemd 定时器单元。systemd 定时器单元可以激活另一个不同类型的单元（例如某项服务），其单元名称与该定时器单元名称相匹配。定时器单元允许基于定时器来激活其他单元。为了便于调试，systemd 将定时器事件记录在系统日志中。

## 定时器单元示例

sysstat 软件包提供了一个名为 sysstat-collect.timer 的 systemd 定时器单元，它将每 10 分钟收集一次系统统计信息。以下输出显示了 **/usr/lib/systemd/system/sysstat-collect.timer** 的配置行。

```
...output omitted...
[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
```

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

```
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

参数 **OnCalendar=*:00/10** 表示该定时器单元将每 10 分钟激活一次相应的单元 (sysstat-collect.service)。不过，您可以指定更复杂的时间间隔。例如，针对 **OnCalendar** 参数，值 **2019-03-*12:35,37,39:16** 将使定时器单元在 2019 年整个 3 月的每一天分别于 **12:35:16**、**12:37:16** 和 **12:39:16** 激活相应的服务单元。您也可以使用参数（如 **OnUnitActiveSec**）来指定相对定时器。例如，**OnUnitActiveSec=15min** 选项将使定时器单元在该定时器单元最后一次激活其相应单元 15 分钟后触发相应的单元。

**重要**

请勿修改 **/usr/lib/systemd/system** 目录下的任何单元配置文件，因为对配置文件的供应商软件包的任何更新都可能会覆盖您在该文件中所做的配置更改。因此，请在 **/etc/systemd/system** 目录下制作一份要更改的单元配置文件的副本，然后修改此副本，以使您对单元所做的配置更改不会被供应商软件包的任何更新所覆盖。如果 **/usr/lib/systemd/system** 和 **/etc/systemd/system** 目录下存在两个同名的文件，systemd 将解析 **/etc/systemd/system** 目录下的文件。

更改定时器单元配置文件后，使用 **systemctl daemon-reload** 命令以确保 systemd 知道这些更改。 该命令将重新加载 systemd 管理器配置。

```
[root@host ~]# systemctl daemon-reload
```

重新加载 systemd 管理器配置后，使用以下 **systemctl** 命令激活定时器单元。

```
[root@host ~]# systemctl enable --now <unitname>.timer
```

**参考文献**

**crontab**(5)、**anacron**(8)、**anacrontab**(5)、systemd.time(7)、systemd.timer(5) 和 crond(8) man page

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

## 指导练习

# 计划周期性系统作业

在本练习中，您将通过向系统 crontab 目录中添加配置文件，以此安排命令按各种计划运行。

### 成果

您应能够：

- 计划周期性系统作业，以统计活动用户数量。
- 更新用于收集系统活动数据的 systemd 定时器单元。

### 在你开始之前

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

在 workstation 上，运行 **lab scheduling-system start** 开始本练习。此脚本会确保环境清理干净且设置正确无误。

```
[student@workstation ~]$ lab scheduling-system start
```

**1.** 从 workstation，以 student 用户身份打开连接 servera 的 SSH 会话。

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

**2.** 使用 **sudo -i** 命令，切换为 root 用户帐户。

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

**3.** 计划周期性系统作业，该作业将生成一条日志消息，指示系统中当前活动用户的数量。该作业必须每天运行。可以使用 **w -h | wc -l** 命令来检索系统中当前活动用户的数量。此外，使用 **logger** 命令还可以生成日志消息。

3.1. 创建名为 **/etc/cron.daily/usercount** 并包含以下内容的脚本文件。可以使用 **vi /etc/cron.daily/usercount** 命令来创建脚本文件。

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
logger "There are currently ${USERCOUNT} active users"
```

3.2. 使用 **chmod** 命令，在 **/etc/cron.daily/usercount** 上启用执行 (**x**) 权限。

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

```
[root@servera ~]# chmod +x /etc/cron.daily/usercount
```

**4.** sysstat 软件包中提供了名为 sysstat-collect. timer 和 sysstat-collect.service 的 systemd 单元。该定时器单元将每 10 分钟触发一次服务单元，以使用名为 **/usr/lib64/sa/sa1** 的 shell 脚本收集系统活动数据。确保已安装 sysstat 软件包并更改定时器单元的配置文件，从而每两分钟收集一次系统活动数据。

4.1. 使用 **yum** 命令安装 sysstat 软件包。

```
[root@servera ~]# yum install sysstat
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  sysstat-11.7.3-2.el8.x86_64              lm_sensors-
libs-3.4.0-17.20180522git70f7e08.el8.x86_64

Complete!
```

4.2. 将 **/usr/lib/systemd/system/sysstat-collect.timer** 复制到 **/etc/systemd/system/sysstat-collect.timer**。

```
[root@servera ~]# cp /usr/lib/systemd/system/sysstat-collect.timer \
/etc/systemd/system/sysstat-collect.timer
```

> **重要**
> 不要编辑 **/usr/lib/systemd** 目录下的文件。对于 systemd 而言，您可以将单元文件复制到 **/etc/systemd/system** 目录并编辑该副本。systemd 进程会解析您的自定义副本，而非 **/usr/lib/systemd** 目录下的文件。

4.3. 编辑 **/etc/systemd/system/sysstat-collect.timer**，使定时器单元每两分钟运行一次。同时，将整个单元配置文件中出现的所有 **10 minutes** 字符串替换为 **2 minutes**，包括注释行中的这些字符串。您可以使用 **vi /etc/systemd/system/sysstat-collect.timer** 命令来编辑配置文件。

```
...
#        Activates activity collector every 2 minutes

[Unit]
Description=Run system activity accounting tool every 2 minutes

[Timer]
OnCalendar=*:00/02

[Install]
WantedBy=sysstat.service
```

上述更改将使 sysstat-collect. timer 单元每两分钟触发一次 sysstat-collect.service 单元，后者将运行 **/usr/lib64/sa/sa1 1 1**。运行 **/usr/**

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

**lib64/sa/sa1 1 1** 时将收集系统活动数据并将其放在 **/var/log/sa** 目录下的二进制文件中。

4.4. 使用 **systemctl daemon-reload** 命令，确保 systemd 知道这些更改。

```
[root@servera ~]# systemctl daemon-reload
```

4.5. 使用 **systemctl** 命令，激活 sysstat-collect.timer 定时器单元。

```
[root@servera ~]# systemctl enable --now sysstat-collect.timer
```

4.6. 使用 **while** 命令，等待在 **/var/log/sa** 目录下创建好二进制文件。等待您的 shell 提示符返回。

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ]; do sleep 1s; done
```

在上述 **while** 命令中，如果文件不存在，**ls /var/log/sa | wc -l** 将返回 **0**；如果文件存在，则返回 **1**。**while** 将确定返回值是否为 **0**，如果是，则进入循环，从而暂停一秒钟。当文件存在时，**while** 循环退出。

4.7. 使用 **ls -l** 命令，验证 **/var/log/sa** 目录下的二进制文件在最后两分钟内是否被修改。

```
[root@servera ~]# ls -l /var/log/sa
total 8
-rw-r--r--. 1 root root 5156 Mar 25 12:34 sa25
[root@servera ~]# date
Mon Mar 25 12:35:32 +07 2019
```

上述命令的输出可能在您的系统上有所不同。

4.8. 退出 root 用户的 shell。

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

4.9. 从 servera 注销。

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

## 完成

在 workstation 上，运行 **lab scheduling-system finish** 来完成本练习。此脚本将删除在实验过程中创建的文件，并确保环境清理干净。

```
[student@workstation ~]$ lab scheduling-system finish
```

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

本引导式练习到此结束。

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

# 管理临时文件

## 培训目标

学完本节后，您应能够启用和禁用 systemd 定时器并配置管理临时文件的定时器。

## 管理临时文件

现代系统需要大量的临时文件和目录。有些应用（和用户）会使用 **/tmp** 目录来保存临时数据，还有一些应用（和用户）则使用更特定于任务的位置，如守护进程以及 **/run** 下特定于用户的易失性目录。就此而言，易失性意味着存储这些文件的文件系统只存在于内存中。在系统重新启动或断电时，易失性存储器的所有内容都会丢失。

为保持系统充分运行，有必要创建那些不存在的目录和文件，因为守护进程和脚本可能会依靠它们的存在，而清除旧文件后就不会填满磁盘空间或提供错误信息。

红帽企业 Linux 7 及更高版本中包含了一个名为 **systemd-tmpfiles** 的新工具，它提供了一种结构化和可配置的方法来管理临时目录和文件。

在 systemd 启动系统后，其中一个最先启动的服务单元是 systemd-tmpfiles-setup。该服务运行命令 **systemd-tmpfiles --create --remove**。此命令从 **/usr/lib/tmpfiles.d/*.conf**、**/run/tmpfiles.d/*.conf** 和 **/etc/tmpfiles.d/*.conf** 读取配置文件。系统会删除这些配置文件中标记要删除的任何文件和目录，并且会创建标记要创建（或修复权限）的任何文件和目录，并使其拥有正确的权限（如有必要）。

## 使用 Systemd 定时器清理临时文件

为确保长期运行的系统不会用陈旧数据填满磁盘，一个名为 systemd-tmpfiles-clean.timer 的 systemd 定时器单元会定期触发 systemd-tmpfiles-clean.service 来执行 **systemd-tmpfiles --clean** 命令。

**systemd** 定时器单元的配置文件中有一个 **[Timer]** 部分，它指示了启动同名服务的频率。

使用以下 **systemctl** 命令可以查看 **systemd-tmpfiles-clean.timer** 单元配置文件的内容。

```
[user@host ~]$ systemctl cat systemd-tmpfiles-clean.timer
# /usr/lib/systemd/system/systemd-tmpfiles-clean.timer
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published
#  by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Daily Cleanup of Temporary Directories
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)
```

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

```
[Timer]
OnBootSec=15min
OnUnitActiveSec=1d
```

在上述配置中，参数 **OnBootSec=15min** 表示名为 systemd-tmpfiles-clean.service 的服务单元将在系统启动 15 分钟后被触发。参数 **OnUnitActiveSec=1d** 表示在上一次激活服务单元 24 小时后再次触发 systemd-tmpfiles-clean.service 服务单元的时间。

根据您的要求，您可以更改 **systemd-tmpfiles-clean.timer** 定时器单元配置文件中的参数。例如，参数 **OnUnitActiveSec** 的值为 **30min** 时，将在上一次激活服务单元 30 分钟后触发 systemd-tmpfiles-clean.service 服务单元。因此，在更改生效后将每 30 分钟触发一次 systemd-tmpfiles-clean.service。

更改定时器单元配置文件后，使用 **systemctl daemon-reload** 命令以确保 systemd 知道这些更改。该命令将重新加载 systemd 管理器配置。

```
[root@host ~]# systemctl daemon-reload
```

重新加载 systemd 管理器配置后，使用以下 **systemctl** 命令激活 systemd-tmpfiles-clean.timer 单元。

```
[root@host ~]# systemctl enable --now systemd-tmpfiles-clean.timer
```

## 手动清理临时文件

命令 **systemd-tmpfiles --clean** 解析的配置文件与 **systemd-tmpfiles --create** 命令相同，但前者不会创建文件和目录，而是会清除在比配置文件中定义的最长期限更近的时间尚未访问、更改或修改的所有文件。

**tmpfiles.d**(5) man page 中详述了 **systemd-tmpfiles** 的配置文件格式。基本语法由七列构成：“类型”、“路径”、“模式”、“UID”、“GID”、“期限”和“参数”。类型指的是 **systemd-tmpfiles** 应执行的操作；例如 **d** 表示创建还不存在的目录，或者 **Z** 表示以递归方式恢复 SELinux 上下文以及文件权限和所有权。

以下是一些示例及相关解释。

```
d /run/systemd/seats 0755 root root -
```

在创建文件和目录时，如果目录 **/run/systemd/seats** 还不存在，则创建该目录，所有者为用户 root 和组 root，权限设置为 **rwxr-xr-x**。系统不会自动清除该目录。

```
D /home/student 0700 student student 1d
```

如果目录 **/home/student** 还不存在，请创建该目录。如果存在，则清空其所有内容。运行 **systemd-tmpfiles --clean** 时，删除在超过一天时间内尚未被访问、更改或修改的所有文件。

```
L /run/fstablink - root root - /etc/fstab
```

创建指向 **/run/fstablink** 的符号链接 **/etc/fstab**。绝对不要自动清除这一行。

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