openEuler

系统服务的配置和管理

课程介绍

- 本章介绍了 openEuler 中的任务的基本概念及管理系统服务，介绍了任务的配置方式及配置示例。
- 学完本课程后，您将能够：管理系统服务；配置一次性或定期执行的任务。

目录

CONTENT

openEuler

系统服务的概念

systemd是在Linux下，与SysV和LSB初始化脚本兼容的系统和服务管理器。systemd使用socket和D-Bus来开启服务，提供基于守护进程的按需启动策略，支持快照和系统状态恢复，维护挂载和自挂载点，实现了各服务间基于从属关系的一个更为精细的逻辑控制，拥有更高的并行性能。

systemd命令

systemd开启和监督整个系统，是基于unit的概念。unit是由一个与配置文件对应的名字和类型组成的（例如：avahi.service unit有一个具有相同名字的配置文件，是守护进程Avahi的一个封装单元）。
unit有多重类型，如下表所示。

unit名称	后缀名	描述
Service unit	.service	系统服务。
Target unit	.target	一组systemd units。
Automount unit	.automount	文件系统挂载点。
Device unit	.device	内核识别的设备文件。
Mount unit	.mount	文件系统挂载点。
Path unit	.path	在一个文件系统中的文件或目录。
Scope unit	.scope	外部创建的进程。
Slice unit	.slice	一组用于管理系统进程分层组织的units。
Socket unit	.socket	一个进程间通信的Socket。
Swap unit	.swap	swap设备或者swap文件。
Timer unit	.timer	systemd计时器。

systemd units存储路径

所有的可用systemd unit类型，可在如下表所示的路径下查看

路径	描述
/usr/lib/systemd/system/	随安装的RPM产生的systemd units。
/run/systemd/system/	在运行时创建systemd units。
/etc/systemd/system/	由系统管理员创建和管理的systemd units。

systemd提供按需启动能力

当sysvinit系统初始化的时候，它会将所有可能用到的后台服务进程全部启动运行。并且系统必须等待所有的服务都启动就绪之后，才允许用户登录。这种做法有两个缺点：首先是启动时间过长；其次是系统资源浪费。

某些服务很可能在很长一段时间内，甚至整个服务器运行期间都没有被使用过。比如CUPS，打印服务在多数服务器上很少被真正使用到。您可能没有想到，在很多服务器上SSHD也是很少被真正访问到的。花费在启动这些服务上的时间是不必要的；同样，花费在这些服务上的系统资源也是一种浪费。

systemd可以提供按需启动的能力，只有在某个服务被真正请求的时候才启动它。当该服务结束，systemd可以关闭它，等待下次需要时再次启动它

systemd采用cgroup特性跟踪和管理进程的生命周期

init系统的一个重要职责就是负责跟踪和管理服务进程的生命周期。它不仅可以启动一个服务，也能够停止服务。这看上去没有什么特别的，然而在真正用代码实现的时候，您或许会发现停止服务比一开始想的要困难。

服务进程一般都会作为守护进程（daemon）在后台运行，为此服务程序有时候会派生（fork）两次。在UpStart中，需要在配置文件中正确地配置expect小节。这样UpStart通过对fork系统调用进行计数，从而获知真正的精灵进程的PID号。

cgroup已经出现了很久，它主要用来实现系统资源配额管理。cgroup提供了类似文件系统的接口，使用方便。当进程创建子进程时，子进程会继承父进程的cgroup。因此无论服务如何启动新的子进程，所有的这些相关进程都会属于同一个cgroup，systemd只需要简单地遍历指定的cgroup即可正确地找到所有的相关进程，将它们逐一停止即可

启动挂载点和自动挂载的管理

传统的Linux系统中，用户可以用/etc/fstab文件来维护固定的文件系统挂载点。这些挂载点在系统启动过程中被自动挂载，一旦启动过程结束，这些挂载点就会确保存在。这些挂载点都是对系统运行至关重要的文件系统，比如HOME目录。和sysvinit一样，systemd管理这些挂载点，以便能够在系统启动时自动挂载它们。systemd还兼容/etc/fstab文件，您可以继续使用该文件管理挂载点。
有时候用户还需要动态挂载点，比如打算访问DVD内容时，才临时执行挂载以便访问其中的内容，而不访问光盘时该挂载点被取消（umount)，以便节约资源。传统地，人们依赖autofs服务来实现这种功能。
systemd内建了自动挂载服务，无需另外安装autofs服务，可以直接使用systemd提供的自动挂载管理能力来实现autofs的功能

实现事务性依赖关系管理

系统启动过程是由很多的独立工作共同组成的，这些工作之间可能存在依赖关系，比如挂载一个NFS文件系统必须依赖网络能够正常工作。systemd虽然能够最大限度地并发执行很多有依赖关系的工作，但是类似“挂载NFS”和“启动网络”这样的工作还是存在天生的先后依赖关系，无法并发执行。对于这些任务，systemd维护一个“事务一致性”的概念，保证所有相关的服务都可以正常启动而不会出现互相依赖，以至于死锁的情况。

与SysV初始化脚本兼容

和UpStart一样，systemd引入了新的配置方式，对应用程序的开发也有一些新的要求。如果systemd想替代目前正在运行的初始化系统，就必须和现有程序兼容。任何一个Linux发行版都很难为了采用systemd而在短时间内将所有的服务代码都修改一遍。

systemd提供了和sysvinit以及LSB initscripts兼容的特性。系统中已经存在的服务和进程无需修改。这降低了系统向systemd迁移的成本，使得systemd替换现有初始化系统成为可能

能够对系统进行快照和恢复

systemd支持按需启动，因此系统的运行状态是动态变化的，人们无法准确地知道系统当前运行了哪些服务。systemd快照提供了一种将当前系统运行状态保存并恢复的能力。

比如系统当前正运行服务A和B，可以用systemd命令行对当前系统运行状况创建快照。然后将进程A停止，或者做其他的任意的对系统的改变，比如启动新的进程C。在这些改变之后，运行systemd的快照恢复命令，就可立即将系统恢复到快照时刻的状态，即只有服务A和B在运行。一个可能的应用场景是调试：比如服务器出现一些异常，为了调试用户将当前状态保存为快照，然后可以进行任意的操作，比如停止服务等等。等调试结束，恢复快照即可

随堂测

- 1.所有的可用systemd unit类型有以下哪些路径？（多选题）

  A. /usr/lib/systemd/system/

  B. /run/systemd/system/

  C. /opt/systemd/system/

  D. /etc/systemd/system/

- 2. systemd不兼容/etc/fstab文件。（判断题）

openEuler

目录

CONTENT

openEuler

systemctl命令与sysvinit命令的对应关系

systemd提供systemctl命令来运行、关闭、重启、显示、启用/禁用系统服务。
systemd提供systemctl命令与sysvinit命令的功能类似。当前版本中依然兼容service和chkconfig命令，相关说明如下表，但建议用systemctl进行系统服务管理。

sysvinit命令	systemd命令	备注
service network start	systemctl start network.service	用来启动一个服务 (并不会重启现有的)。
service network stop	systemctl stop network.service	用来停止一个服务 (并不会重启现有的)。
service network restart	systemctl restart network.service	用来停止并启动一个服务。
service network reload	systemctl reload network.service	当支持时，重新装载配置文件而不中断等待操作。
service network condrestart	systemctl condrestart network.service	如果服务正在运行那么重启它。
service network status	systemctl status network.service	检查服务的运行状态。
chkconfig network on	systemctl enable network.service	在下次启动时或满足其他触发条件时设置服务为启用。
chkconfig network off	systemctl disable network.service	在下次启动时或满足其他触发条件时设置服务为禁用。
chkconfig network	systemctl is-enabled network.service	用来检查一个服务在当前环境下被配置为启用还是禁用。
chkconfig \-\-list	systemctl list-unit-files \-\-type=service	输出在各个运行级别下服务的启用和禁用情况。
chkconfig network \-\-list	ls /etc/systemd/system/*.wants/network.service	用来列出该服务在哪些运行级别下启用和禁用。
chkconfig network \-\-add	systemctl daemon-reload	当您创建新服务文件或者变更设置时使用。

显示所有当前服务

如果您需要显示当前正在运行的服务，使用命令如下：

```
systemctl list-units --type service
```

如果您需要显示所有的服务（包括未运行的服务），需要添加-all参数，使用命令如下：

```
systemctl list-units --type service --all
```

例如显示当前正在运行的服务，命令如下：

```
$ systemctl list-units --type service
UNIT                    LOAD   ACTIVE     SUB     JOB   DESCRIPTION
atd.service               loaded active     running       Deferred execution scheduler
auditd.service            loaded active     running       Security Auditing Service
avahi-daemon.service      loaded active     running       Avahi mDNS/DNS-SD Stack
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
dracut-shutdown.service   loaded active     exited        Restore /run/initramfs on shutdown
firewalld.service         loaded active     running       firewalld - dynamic firewall daemon
getty@tty1.service        loaded active     running       Getty on tty1
gssproxy.service          loaded active     running       GSSAPI Proxy Daemon
irqbalance.service        loaded active     running       irqbalance daemon
iscsid.service            loaded activating start   start Open-iSCSI
```

显示服务状态（1）

如果您需要显示某个服务的状态，可执行如下命令：

```
systemctl status name.service
```

相关状态显示参数说明如下表所示

参数	描述
Loaded	说明服务是否被加载，并显示服务对应的绝对路径以及是否启用。
Active	说明服务是否正在运行，并显示时间节点。
Main PID	相应的系统服务的PID值。
CGroup	相关控制组（CGroup）的其他信息。

显示服务状态（2）

如果您需要鉴别某个服务是否运行，可执行如下命令：

```
systemctl is-active name.service
```

is-active命令的返回结果如下：

状态	含义
active(running)	有一只或多只程序正在系统中执行
active(exited)	仅执行一次就正常结束的服务，目前并没有任何程序在系统中执行。 举例来说，开机或者 是挂载时才会进行一次的 quotaon 功能
active(waiting)	正在执行当中，不过要等待其他的事件才能继续处理。例如：打印的队列相关服务 就是这种状态，虽然正在启动中，不过也需要真的有队列进来 (打印作业) 这样他才会继续唤醒打印机 服务来进行下一步打印的功能
inactive	这个服务没有运行

显示服务状态（3）

如果您需要判断某个服务是否被启用，可执行如下命令：

```
systemctl is-enabled name.service
```

is-enabled命令的返回结果如下：

状态	含义
"enabled"	已经通过 /etc/systemd/system/ 目录下的 Alias= 别名、 .wants/ 或 .requires/ 软连接被永久启用。
"enabled-runtime"	已经通过 /run/systemd/system/ 目录下的 Alias= 别名、 .wants/ 或 .requires/ 软连接被临时启用。
"linked"	虽然单元文件本身不在标准单元目录中，但是指向此单元文件的一个或多个软连接已经存在于 /etc/systemd/system/ 永久目录中。
"linked-runtime"	虽然单元文件本身不在标准单元目录中，但是指向此单元文件的一个或多个软连接已经存在于 /run/systemd/system/ 临时目录中。
"masked"	已经被 /etc/systemd/system/ 目录永久屏蔽(软连接指向 /dev/null 文件)，因此 **start** 操作会失败。
"masked-runtime"	已经被 /run/systemd/systemd/ 目录临时屏蔽(软连接指向 /dev/null 文件)，因此 **start** 操作会失败。
"static"	尚未被启用，并且单元文件的 "[Install]" 小节中没有可用于 **enable** 命令的选项。
"indirect"	尚未被启用，但是单元文件的 "[Install]" 小节中 Also= 选项的值列表非空(也就是列表中的某些单元可能已被启用)、或者它拥有一个不在 Also= 列表中的其他名称的别名软连接。对于模版单元来说，表示已经启用了一个不同于 DefaultInstance= 的实例。
"disabled"	尚未被启用，但是单元文件的 "[Install]" 小节中存在可用于 **enable** 命令的选项
"generated"	单元文件是被单元生成器动态生成的。被生成的单元文件可能并未被直接启用，而是被单元生成器隐含的启用了。
"transient"	单元文件是被运行时API动态临时生成的。该临时单元可能并未被启用。
"bad"	单元文件不正确或者出现其他错误。 **is-enabled** 不会返回此状态，而是会显示一条出错信息。 **list-unit-files** 命令有可能会显示此单元。

显示服务状态（4）

例如查看gdm.service服务状态，命令如下：

```
# systemctl status gdm.service
gdm.service - GNOME Display Manager   Loaded: loaded (/usr/lib/systemd/system/gdm.service;
enabled)   Active: active (running) since Thu 2013-10-17 17:31:23 CEST; 5min ago
 Main PID: 1029 (gdm)
   CGroup: /system.slice/gdm.service
           ├─1029 /usr/sbin/gdm
           ├─1037 /usr/libexec/gdm-simple-slave --display-id /org/gno...
           └─1047 /usr/bin/Xorg :0 -background none -verbose -auth /r...Oct 17 17:31:23 localhost
systemd[1]: Started GNOME Display Manager.
```

服务的运行与关闭

如果您需要运行某个服务，请在root权限下执行如下命令：

```
systemctl start name.service
```

例如运行httpd服务，命令如下：

```
# systemctl start httpd.service
```

如果您需要关闭某个服务，请在root权限下执行如下命令：

```
systemctl stop name.service
```

例如关闭蓝牙服务，命令如下：

```
# systemctl stop bluetooth.service
```

重启服务

如果您需要重启某个服务，请在root权限下执行如下命令：

```
systemctl restart name.service
```

执行命令后，当前服务会被关闭，但马上重新启动。如果您指定的服务，当前处于关闭状态，执行命令后，服务也会被启动。

例如重启蓝牙服务，命令如下：

```
# systemctl restart bluetooth.service
```

启用与禁用服务

如果您需要在开机时启用某个服务，请在root权限下执行如下命令：

```
systemctl enable name.service
```

例如设置httpd服务开机时启动，命令如下：

```
# systemctl enable httpd.service
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

如果您需要在开机时禁用某个服务，请在root权限下执行如下命令：

```
systemctl disable name.service
```

例如在开机时禁用蓝牙服务启动，命令如下：

```
# systemctl disable bluetooth.service
Removed /etc/systemd/system/bluetooth.target.wants/bluetooth.service.
Removed /etc/systemd/system/dbus-org.bluez.service.
```

改变运行级别

systemd用目标（target）替代了运行级别的概念，提供了更大的灵活性，如您可以继承一个已有的目标，并添加其他服务，来创建自己的目标。下表列举了systemd下的目标和常见runlevel的对应关系。

运行级别	systemd目标（target）	描述
0	runlevel0.target，poweroff.target	关闭系统。
1, s, single	runlevel1.target，rescue.target	单用户模式。
2, 4	runlevel2.target，runlevel4.target，multi-user.target	用户定义/域特定运行级别。默认等同于3。
3	runlevel3.target，multi-user.target	多用户，非图形化。用户可以通过多个控制台或网络登录。
5	runlevel5.target，graphical.target	多用户，图形化。通常为所有运行级别3的服务外加图形化登录。
6	runlevel6.target，reboot.target	重启系统。
emergency	emergency.target	紧急Shell。

系统启动目标管理

查看当前系统默认的启动目标，命令如下：

```
systemctl get-default
```

查看当前系统所有的启动目标，命令如下：

```
systemctl list-units --type=target
```

改变系统默认的目标，在root权限下执行如下命令：

```
systemctl set-default name.target
```

改变当前系统的目标，在root权限下执行如下命令：

```
systemctl isolate name.target
```

切换救援与紧急模式

改变当前系统为救援模式，在root权限下执行如下命令：

```
systemctl rescue
```

这条命令和“systemctl isolate rescue.target”类似。命令执行后会在串口有如下打印信息：

```
You are in rescue mode. After logging in, type "journalctl -xb" to viewsystem logs, "systemctl reboot" to
reboot, "systemctl default" or "exit"to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):
```

改变当前系统为紧急模式，在root权限下执行如下命令：

```
systemctl emergency
```

这条命令和“systemctl isolate emergency.target”类似。命令执行后会在串口有如下打印信息：

```
You are in emergency mode. After logging in, type "journalctl -xb" to viewsystem logs, "systemctl
reboot" to reboot, "systemctl default" or "exit"to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):
```

用户需要重启系统，才能从救援模式或者紧急模式进入正常模式。

关闭、暂停和休眠系统

systemd通过systemctl命令可以对系统进行关机、重启、休眠等一系列操作。当前仍兼容部分Linux常用管理命令，对应关系如下表。建议用户使用systemctl命令进行操作。

Linux常用管理命令	systemctl命令	描述
halt	systemctl halt	关闭系统
poweroff	systemctl poweroff	关闭电源
reboot	systemctl reboot	重启

关闭系统

关闭系统并下电，在root权限下执行如下命令：

```
systemctl poweroff
```

关闭系统但不下电机器，在root权限下执行如下命令：

```
systemctl halt
```

执行上述命令会给当前所有的登录用户发送一条提示消息。如果不想让systemd发送该消息，您可以添加“--no-wall”参数。具体命令如下：

```
systemctl --no-wall poweroff
```

重启系统

重启系统，在root权限下执行如下命令：

```
systemctl reboot
```

执行上述命令会给当前所有的登录用户发送一条提示消息。如果不想让systemd发送该消息，您可以添加“--no-wall”参数。具体命令如下：

```
systemctl --no-wall reboot
```

使系统待机和休眠

使系统待机，在root权限下执行如下命令：

```
systemctl suspend
```

使系统休眠，在root权限下执行如下命令：

```
systemctl hibernate
```

使系统待机且处于休眠状态，在root权限下执行如下命令：

```
systemctl hybrid-sleep
```

随堂测

- 1.设置httpd服务开机时启动用下列哪个命令？单选题)

  A. systemctl enable httpd.service

  B. systemctl enabled httpd.service

  C. systemctl start httpd.service

  D. systemctl disenable httpd.service

- 2.执行systemctl status gdm.service可以查看Main PID。 (判断题)

openEuler

目录

CONTENT

openEuler

任务的概念和相关术语

当你在终端或控制台工作时，可能不希望由于运行一个作业而占住了屏幕，因为可能还有更重要的事情要做，比如阅读电子邮件。对于密集访问磁盘的进程，你可能希望它能够在每天的非负荷高峰时间段运行。为了使这些进程能够在后台运行，也就是说不在终端屏幕上运行，有几种选择方法可供使用。
在本章中我们将讨论：
- 设置c r o n t a b文件，并用它来提交作业。
- 使用a t命令来提交作业。
- 在后台提交作业。
- 使用n o h u p命令提交作业。

名词解释：
cron 系统调度进程。可以使用它在每天的非高峰负荷时间段运行作业，或在一周或一月中的不同时段运行。
at命令。使用它在一个特定的时间运行一些特殊的作业，或在晚一些的非负荷高峰时间段或高峰负荷时间段运行。
& 使用它在后台运行一个占用时间不长的进程。
nohup 使用它在后台运行一个命令，即使在用户退出时也不受影响。

cron和crontab

cron是系统主要的调度进程，可以在无需人工干预的情况下运行作业。有一个叫做c r o n t a b的命令允许用户提交、编辑或删除相应的作业。每一个用户都可以有一crontab文件来保存调度信息。可以使用它运行任意一个 shell脚本或某个命令，每小时运行一次，或一周三次，这完全取决于你。每一个用户都可以有自己的 crontab文件。

corn 运行机制

cron 会搜索 /var/spool/cron 目录，寻找以 /etc/passwd 文件中的用户名命名的crontab 文件，然后将 crontab 文件装入内存

cron 还将搜索 /etc/crontab 文件，这个文件（cron 时间表）是以特定格式写成的cron 启动以后，先检查是否有用户设置了crontab文件，如果没有就转入睡眠状态，释放系统资源；所以该后台进程占用资源极少，之后每分钟被换醒一次，查看当前是否有需要运行的命令

定时任务分类

Linux下的任务调度分为两类，系统任务调度和用户任务调度。

系统任务调度：系统周期性所要执行的工作，比如写缓存数据到硬盘、日志清理等。

在/etc/crontab文件，这个就是系统任务调度的配置文件。

用户任务调度：用户定期要执行的工作，比如用户数据备份、定时邮件提醒等。

用户可以使用 crontab 工具来定制自己的计划任务。

在crontab 文件都被保存在/var/spool/cron目录中。其文件名与用户名一致

```
[root@localhost cron.d]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# 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,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

crontab 文件参数

在 crontab 文件中输入需要执行的命令和时间

该文件中每行都包括6个域，其中前5个域是指定命令被执行的时间，最后一个域是要被执行的命令

每个域之间使用空格或者制表符分隔，格式如下：

```
minute hour day-of-month month-of-year day-of-week  commands
```

minute	分钟（0 – 59）
hour	小时（0 – 23）
day-of-month	一个月的第几天（1 – 31）
month-of-year	一年的第几个月（1 – 12）
day-of-week	一周的星期几（0 – 6），0 代表周日
commands	需要执行的命令

例：每周一上午 8:00 执行命令

```
00 08 * * 1 commands
```

crontab 文件参数补充说明

上述参数项都不能为空，必须指定值

对于要执行的命令，调用的时候需要写出命令的绝对路径

除了数字还有几个特殊的符号“*”、“/”和“-”、“，”

参数	描述
*	代表所有可能的值，如month字段为星号，则表示在满足其它字段的制约条件后每月都执行该命令操作
/	可以用正斜线指定时间的间隔频率，例如“0-23/2”表示每两小时执行一次。
-	可以用整数之间的中杠表示一个整数范围，例如“2-6”表示“2,3,4,5,6”
,	可以用逗号隔开的值指定一个列表范围，例如，“1,2,5,7,8,9”

例：晚上11点到早上8点之间每两个小时，在/tmp/test.txt文件中加入sleepy文本，
crontab 文件中对应的行如下：

```
* 23-8/2 * * * echo"sleepy" >> /tmp/test.txt
```

crontab命令选项

crontab命令的一般形式为：

```
crontab [-u user] -e -l -r
```

其中：
-u 用户名。
-e 编辑crontab文件。
-l 列出crontab文件中的内容。
-r 删除crontab文件。
如果使用自己的名字登录，就不用使用 - u选项，因为在执行 crontab命令时，该命令能够知道当前的用户。

root查看自己的cron设置，命令为：

```
crontab -u root -l
```

将程序放入后台执行

直接将程序放入后台处理，使用&号，如sleep 100 &

Jobs命令可查看当前shell的后台的任务

```
[root@localhost ~]# ping 9.xx.xx.195 >/dev/null &
[1] 2079581
[root@localhost ~]# sleep 100 &
[2] 2079597
[root@localhost ~]# jobs
[1]-  Running                 ping 9.xx.xx.195 > /dev/null &
[2]+  Running                 sleep 100 &
```

将正在运行的任务放入后台暂停，使用ctrl+z，比如在使用vim时，需要查找一个文件，可以在vim的一般模式下，使用ctrl+z，把vim放入后台暂停

```
[root@localhost ~]# jobs
[root@localhost ~]# vim test--------按CTRL+Z
[1]+  Stopped                 vim test
[root@localhost ~]# jobs
[1]+  Stopped                 vim test
```

fg和bg命令

fg %job ID：将任务放入前台执行

bg %job ID：将任务放入后台执行

如果fg和bg后不加job ID，表示对当前任务进行操作

```
[root@localhost devkit2.5RC1]# tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner  //将Hyper_tuner目录打包并压缩。
^Z    //使用ctrl+z，将任务放入后台并且暂停。
[1]+  Stopped                 tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner
[root@localhost devkit2.5RC1]# fg   //使用fg命令，将后台任务再次放到前台执行。
tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner
^Z //使用ctrl+z，将任务放入后台并且暂停。
[1]+  Stopped                 tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner
[root@localhost devkit2.5RC1]# bg   //使用bg命令，将任务放入后台执行。
[1]+ tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner &
[root@localhost devkit2.5RC1]# jobs  //使用jobs命令查看在后台执行的任务。
[1]+  Running                 tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner &
[root@localhost devkit2.5RC1]# jobs
[1]+  Done                    tar -zcf Hyper_tuner_test.tar.gz Hyper_tuner
```

定时任务命令-at

用户使用 at 命令在指定时刻执行指定的命令序列

该命令至少需要指定一个命令和一个执行时间

at 命令可以只指定时间，也可以时间和日期一起指定

at命令的基本形式为：

```
at [-f script] [-m -l -r] [time] [date]
```

其中，
-f script 是所要提交的脚本或命令。
-l 列出当前所有等待运行的作业。 a t q命令具有相同的作用。
-r 清除作业。为了清除某个作业，还要提供相应的作业标识（ l D）； 有些U N l X变体只接受a t r m作为清除命令。
-m 作业完成后给用户发邮件。
time at命令的时间格式非常灵活；可以是 H、H H . H H M M、H H : M M或H : M，其中H和M分别是小时和分钟。还可以使用 a . m .或p . m .。
date 日期格式可以是月份数或日期数，而且 a t命令还能够识别诸如t o d a y、t o m o r r o w这样的词。

使用at命令提交命令

使用at命令提交作业有几种不同的形式，可以通过命令行方式，也可以使用 at命令提示符。

一般来说在提交若干行的系统命令时，我使用 at命令提示符方式，而在提交 shell脚本时，使用

命令行方式。如果你想提交若干行的命令，可以在 at命令后面跟上日期/时间并回车。然后就进入了 at命

令提示符，这时只需逐条输入相应的命令，然后按 ‘< CTRL - D >’ 退出。下面给出一个例子:

```
[root@localhost ansible]# at 19:30
warning: commands will be executed using /bin/sh
at> find / -name "passwd" -print
at> <EOT>
job 1 at Mon May 23 19:30:00 2022
```

其中，< EOT >就是< CTRL - D>。在19: 30系统将执行一个简单的 f i n d命令。你应当已经注意到，我所提交的作业被分配了一个唯一标识 job 1。

使用at命令提交脚本

如果希望向a t命令提交一个s h e l l脚本，使用其命令行方式即可。在提交脚本时使用 - f选项。

```
at 19:30pm tomorrow –f /apps/bin/db.table.sh
warning: commands will be executed using /bin/sh
job 1 at Mon May 24 19:30:00 2022
```

在上面的例子中，一个叫做 d b _ t a b l e . s h的脚本将在明天下午19: 30运行。

还可以使用e c h o命令向a t命令提交作业，示例如下：

```
echo find /etc -name "passwd" -print | at now +1 minute
```

执行权限

对于 at 命令来说，需要定时执行的命令是从标准输入或者使用 –f 选项指定的文件中读取并执行的

如果 at 命令是从一个使用 su 命令切换到用户 shell 中执行的，那么当前用户被认为是执行用户，所有的错误和输出结果都会送给这个用户

对于其他用户来说，是否执行某个命令或脚本取决于 /etc/at.allow

nohup命令

如果你正在运行一个进程，而且你觉得在退出帐户时该进程还不会结束，那么可以使用nohup命令。该命令可以在你退出帐户之后继续运行相应的进程。 nohup就是不挂起的意思(n o hang up)。
该命令的一般形式为：

```
nohup command &
```

如果使用nohup命令提交作业，那么在缺省情况下该作业的所有输出都被重定向到一个名为nohup. out的文件中，除非另外指定了输出文件：
nohup command > myout.file 2>&1

随堂测

- 1. 什么命令可查看当前shell的后台的任务？（单选题）

  A. cat

  B. vim

  C. jobs

  D. bg

- 2. at 命令可以只指定时间，也可以时间和日期一起指定。（判断题）

openEuler

Thank you

openEuler