使用**systemd服务管理机制**实现脚本程序的运行。脚本可单次手动执行，也可以作为开机自启服务。

1. 编写helloworld.sh

功能：

程序每隔1s输出hello word，输出格式为：[Ymd-H:M:S]: hello world，并将输出内容写入到/tmp/hello.log文件中。

文件绝对路径：

/home/lambda/Desktop/helloworld.sh

实现：

#!/bin/bash

while true; do

echo [$(date +%Y%m%d-%H:%M:%S)]: hello world

echo [$(date +%Y%m%d-%H:%M:%S)]: hello world >> /tmp/hello.log

sleep 1

done

注：chmod +x helloworld.sh # 给脚本添加执行权限

1. 编写hello.service

systemd service服务存放位置（将编写的service文件存于此目录下才能被systemd管理）：

* /etc/systemd/system：供系统管理员和用户使用。
* /usr/lib/systemd/system：供发行版打包者使用。

功能：

服务用来执行helloworld.sh脚本，期望在multi-user（有网络，多用户）系统状态下运行。

文件绝对路径：

/etc/systemd/system/hello.service

实现：

[Unit]

Description = hello daemon

[Service]

# 指定服务启动时执行的命令（拉起helloworld.sh脚本）

ExecStart = /home/lambda/Desktop//helloworld.sh

# 启动类型

Type = simple

[Install]

WantedBy = multi-user.target

1. 检索是否存在helloworld服务

systemctl list-unit-files --type=service |grep helloworld

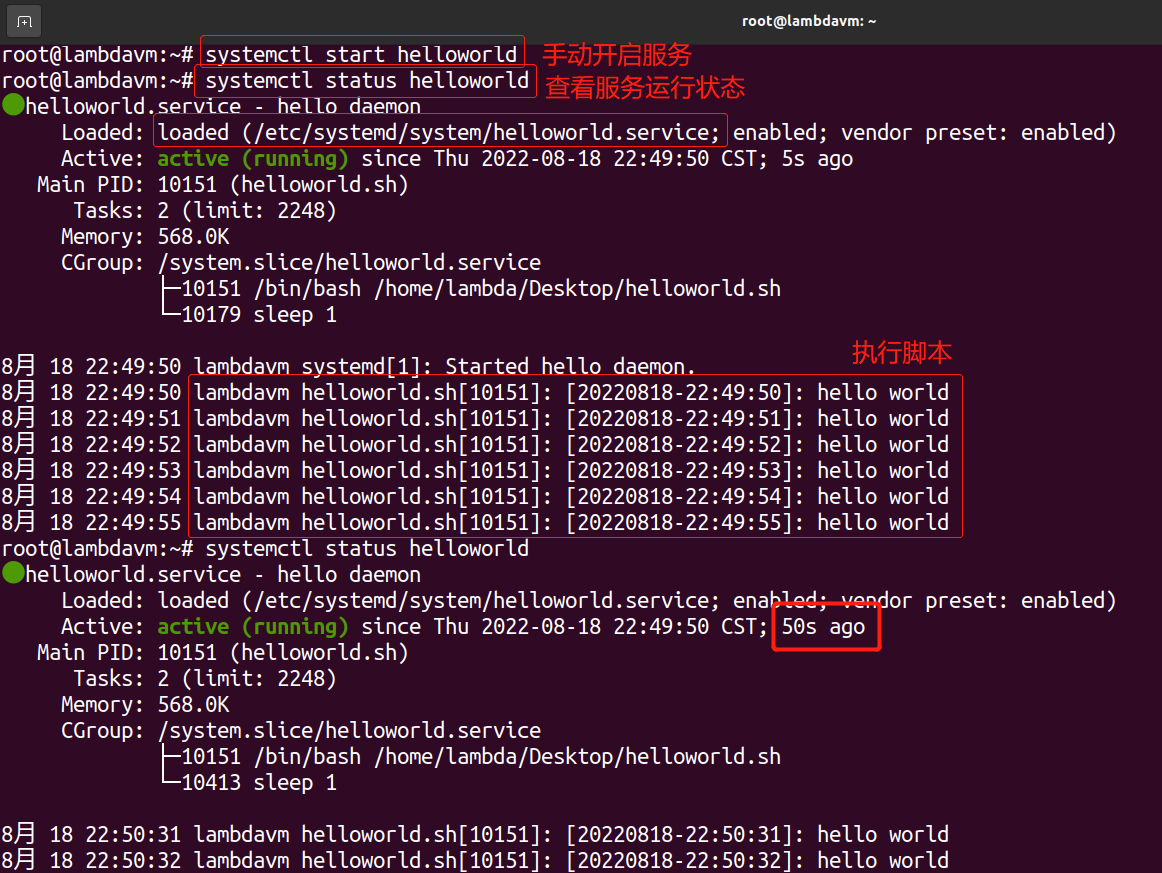


1. 运行服务

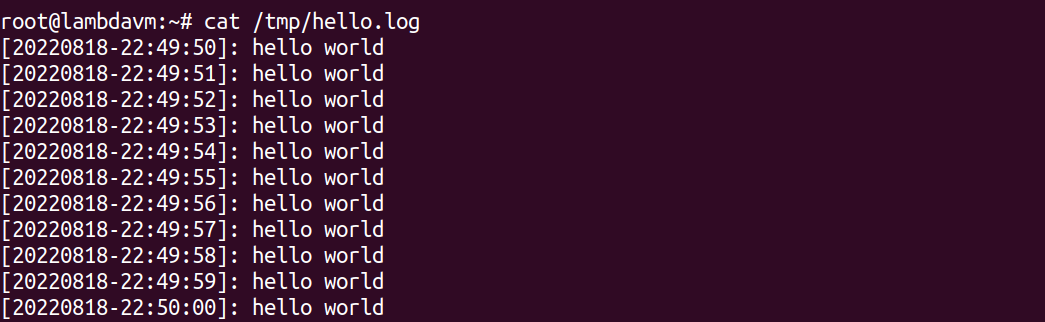
1）手动开启服务

# 通过systemd服务单次执行脚本

systemctl start hello



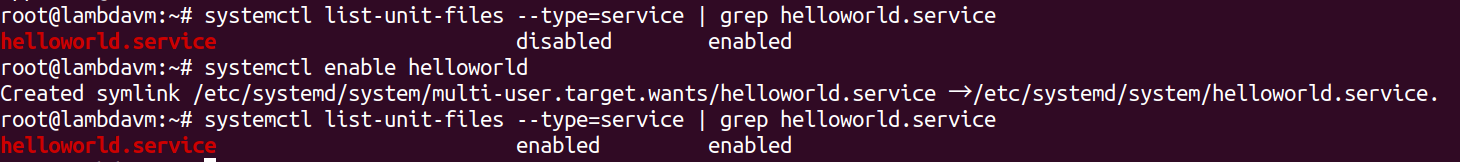
查看是否存储日志文件：



2）将服务设置为开机自启动

# 设置开机自动启动

systemctl enable hello

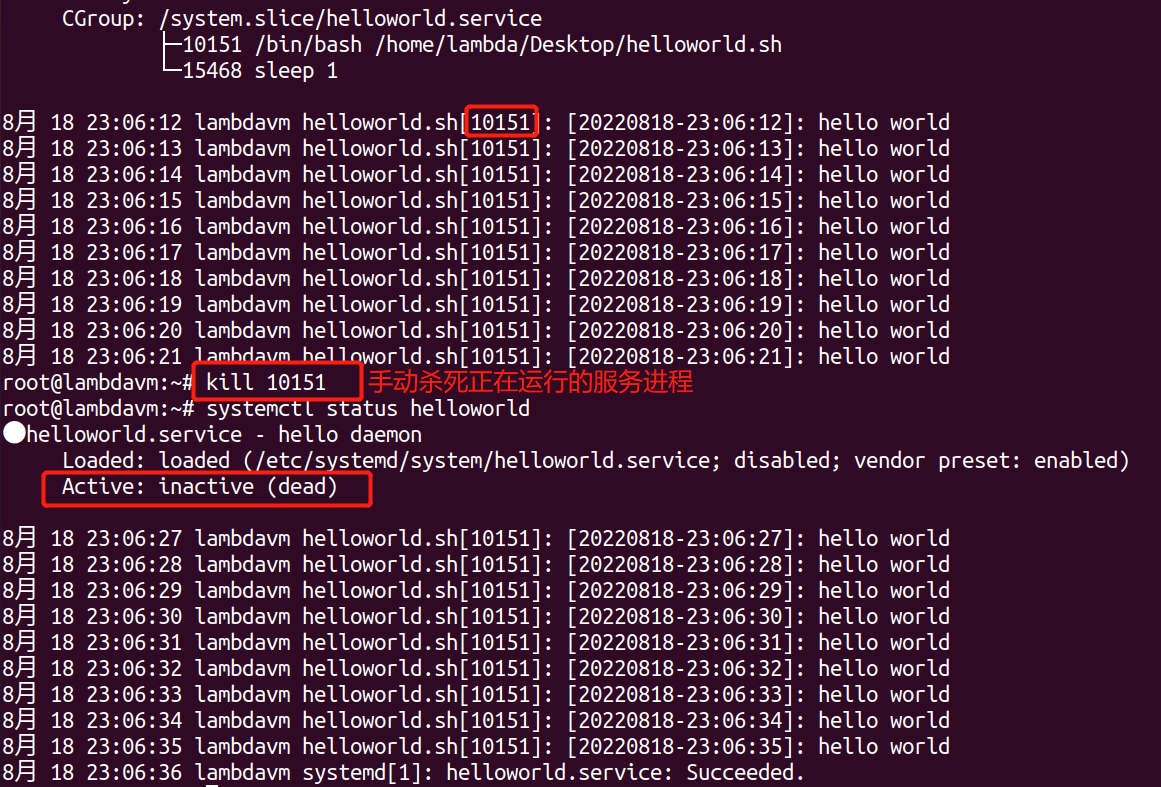


# 关闭服务开机自启

systemctl disable hello



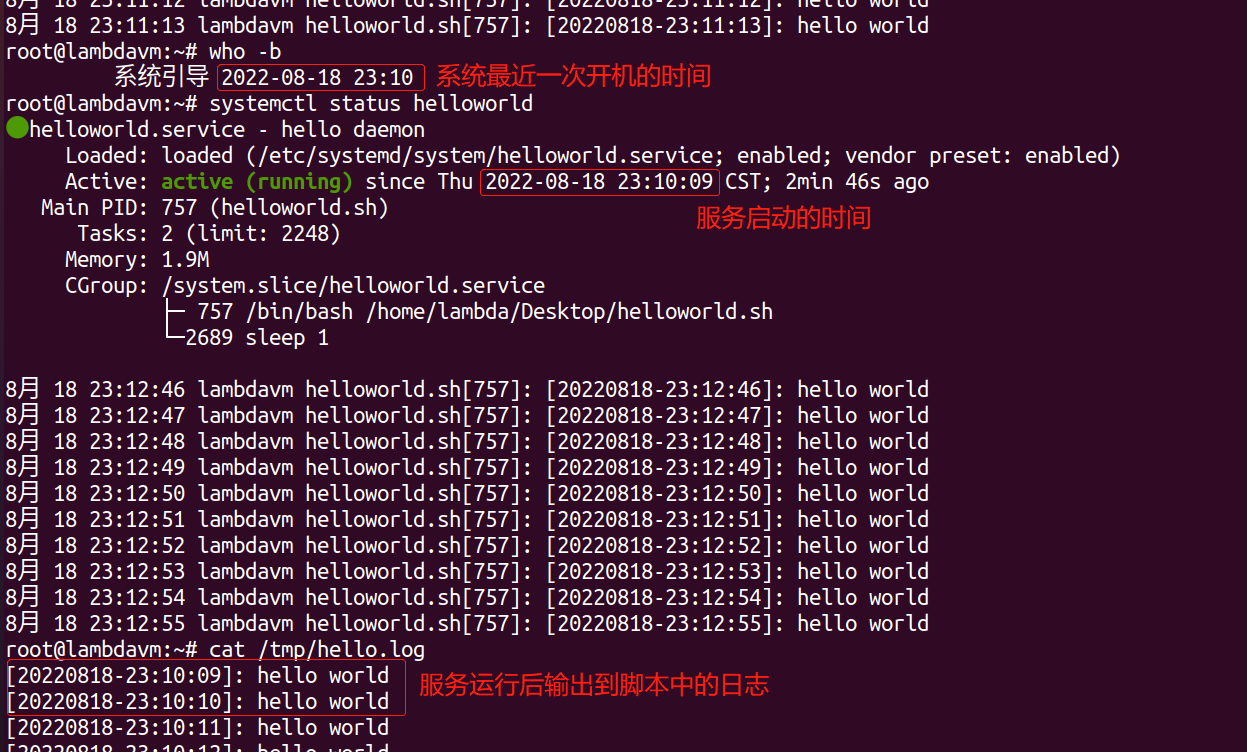
3）手动杀死服务进程



1. 重启，检查脚本是否开启执行

# 重启机器

reboot



service编写手册：

[Unit]区块：启动顺序与依赖关系

|  |  |
| --- | --- |
| Unit指令 | 含义 |
| Description | Unit的描述信息 |
| Documentation | 本Unit的man文档路径 |
| After | 本服务在哪些服务启动之后启动，仅定义启动顺序，不定义服务依赖关系，即使要求先启动的服务启动失败，本服务也依然会启动 |
| Before | 本服务在哪些服务启动之前启动，仅定义启动顺序，不定义服务依赖关系。通常用于定义在关机前要关闭的服务，如Before=shutdown.target |
| Wants | 本服务在哪些服务启动之后启动，定义服务依赖关系，不定义服务启动顺序。启动本服务时，如果被依赖服务未启动，则也会启动被依赖服务。如果被依赖服务启动失败，本服务不会受之影响，因此本服务会继续启动。如果未结合After使用，则本服务和被依赖服务同时启动。当配置在[Install]段落中时，systemctl enable操作将会将本服务安装到对应的.wants目录下(在该目录下创建一个软链接)，在开机自启动时，.wants目录中的服务会被隐式添加至目标Unit的Wants指令后。 |
| Requires | 本服务在哪些服务启动之后启动，定义服务强依赖关系，不定义服务启动顺序。启动本服务时，如果被依赖服务未启动，则也会启动被依赖服务。如果结合了After，当存在非active状态的被依赖服务时，本服务不会启动。且当被依赖服务被手动停止时，本服务也会被停止，但有例外。如果要保证两服务之间状态必须一致，使用BindsTo指令。当配置在[Install]段落中时，systemctl enable操作将会将本服务安装到对应的.requires目录下(在该目录下创建一个软链接)，在开机自启动时，.requires目录中的服务会被隐式添加至目标Unit的Requires指令后。 |
| Requisite | 本服务在哪些服务启动之后启动，定义服务依赖关系，不定义服务启动顺序。启动本服务时，如果被依赖服务处于尚未启动状态，则不会主动去启动这些服务，所以本服务直接启动失败。该指令一般结合After一起使用，以便保证启动顺序。 |
| BindsTo | 绑定两个服务，两服务的状态保证一致。如服务1为active，则本服务也一定为active。 |
| PartOf | 本服务是其它服务的一部分，定义了单向的依赖关系，且只对stop和restart操作有效。当被依赖服务执行stop或restart操作时，本服务也会执行操作，但本服务执行这些操作，不会影响被依赖服务。一般用于组合target使用，比如a.service和b.service都配置PartOf=c.target，那么stop c的时候，也会同时stop a和b。 |
| Conflicts | 定义冲突的服务，本服务和被冲突服务的状态必须相反。当本服务要启动时，将会停止目标服务，当启动目标服务时，将会停止本服务。启动和停止的操作同时进行，所以，如果想要让本服务在目标服务启动之前就已经处于停止状态，则必须定义After/Before。 |
| OnFailure | 当本服务处于failed时，将启动目标服务。如果本服务配置了Restart重启指令，则在耗尽重启次数之后，本服务才会进入failed。有时候这是非常有用的，一个典型用法是本服务失败时调用定义了邮件发送功能的service来发送邮件，特别地，可以结合systemd.timer定时任务实现cron的MAILTO功能。 |
| RefuseManualStart, RefuseManualStop | 本服务不允许手动启动和手动停止，只能被依赖时的启动和停止，如果手动启动或停止，则会报错。有些特殊的服务非常关键，或者某服务作为一个大服务的一部分，为了保证安全，都可以使用该特性。例如，系统审计服务auditd.service中配置了不允许手动停止指令RefuseManualStop，network.target中配置了不允许手动启动指令RefuseManualStart。 |
| AllowIsolated | 允许使用systemctl isolate切换到本服务，只配置在target中。一般来说，用户服务是绝不可能用到这一项的。 |
| ConditionPathExists, AssertPathExists | 要求给定的绝对路径文件已经存在，否则不做任何事(condition)或进入failed状态(assert)，可在路径前使用!表示条件取反，即不存在时才启动服务。 |
| ConditionPathIsDirectory, AssertPathIsDirectory | 如上，路径存在且是目录时启动。 |
| ConditionPathIsReadWrite, AssertPathIsReadWrite | 如上，路径存在且可读可写时启动。 |
| ConditionDirectoryNotEmpty, AssertDirectoryNotEmpty | 如上，路径存在且是非空目录时启动。 |
| ConditionFileNotEmpty, AssertFileNotEmpty | 如上，路径存在且是非空文件时启动。 |
| ConditionFileIsExecutable, AssertFileIsExecutable | 如上，路径存在且是可执行普通文件时启动。 |

对于自定义的服务配置文件来说，需要定义的常见指令包括Description、After、Wants及可能需要的条件判断类指令。

[Service]区块：启动行为定义

|  |  |
| --- | --- |
| EnvironmentFile | 指定当前服务的环境参数文件(路径)，如EnviromentFile=-/etc/sysconfig/xxx，连词号表示抑制错误，即发生错误时，不影响其他命令的执行 |
| Environment | 后面接多个不同的shell变量，如Environment=DATA\_DIR=/dir/data |
| User | 设置服务运行的用户 |
| Group | 设置服务运行的用户组 |
| WorkingDirectory | 设置服务运行的路径 |
| ExecStart | 定义启动服务时执行的命令 |
| ExecStop | 定义停止服务时执行的命令 |
| ExecStartPre | 定义启动服务前执行的命令 |
| ExecStartPost | 定义启动服务后执行的命令 |
| ExecStopPost | 定义停止服务后执行的命令 |
| ExecReload | 定义重启服务时执行的命令 |
| Type | 字段定义启动类型，可以设置的值如下：   * simple（默认值）：ExecStart字段启动的进程为主进程，即直接启动服务进程； * forking：ExecStart字段将以fork()方式启动，此时父进程将会退出，子进程将成为主进程（例如用shell脚本启动服务进程）； * oneshot：类似于simple，但只执行一次，Systemd 会等它执行完，才启动其他服务； * dbus：类似于simple，但会等待 D-Bus 信号后启动； * notify：类似于simple，启动结束后会发出通知信号，然后 Systemd 再启动其他服务； * idle：类似于simple，但是要等到其他任务都执行完，才会启动该服务。一种使用场合是为让该服务的输出，不与其他服务的输出相混合； |
| RemainAfterExit | 设为yes，表示进程退出以后，服务仍然保持执行 |
| KillMode | 定义 Systemd 如何停止服务，可以设置的值如下：   * control-group（default）：当前控制组里面的所有子进程，都会被杀掉； * process：只杀主进程； * mixed：主进程将收到 SIGTERM 信号，子进程收到 SIGKILL 信号； * none：没有进程会被杀掉，只是执行服务的 stop 命令 |
| Restart | 定义了服务退出后，Systemd 的重启方式，可以设置的值如下（对于守护进程，推荐设为on-failure。对于那些允许发生错误退出的服务，可以设为on-abnormal）：   * no（default）：退出后不会重启； * on-success：只有正常退出时（退出状态码为0），才会重启； * on-failure：非正常退出时（退出状态码非0），包括被信号终止和超时，才会重启； * on-abnormal：只有被信号终止和超时，才会重启； * on-abort：只有在收到没有捕捉到的信号终止时，才会重启 * on-watchdog：超时退出，才会重启；always：不管是什么退出原因，总是重启 |
| RestartSec | 表示 Systemd 重启服务之前，需要等待的秒数 |

[Install]区块：服务安装定义

|  |  |
| --- | --- |
| Install指令 | 含义 |
| WantedBy | 本服务设置开机自启动时，在被依赖目标的.wants目录下创建本服务的软链接。例如WantedBy = multi-user.target时，将在/etc/systemd/multi-user.target.wants目录下创建本服务的软链接。 |
| RequiredBy | 类似WantedBy，但是是在.requires目录下创建软链接。 |
| Alias | 指定创建软链接时链接至本服务配置文件的别名文件。例如reboot.target中配置了Alias=ctrl-alt-del.target，当执行enable时，将创建/etc/systemd/system/ctrl-alt-del.service软链接并指向reboot.target。 |
| DefaultInstance | 当是一个模板服务配置文件时(即文件名为Service\_Name@.service)，该指令指定该模板的默认实例。例如trojan@.service中配置了DefaultInstall=server时，systemctl enable trojan@.service时将创建名为trojan@server.service的软链接。 |

[Install]段落相关的指令只在systemctl enable/disable操作时有效。如果期望服务开机自启动，一般只配置一个WantedBy指令，如果不期望服务开机自启动，则Install段落通常省略。