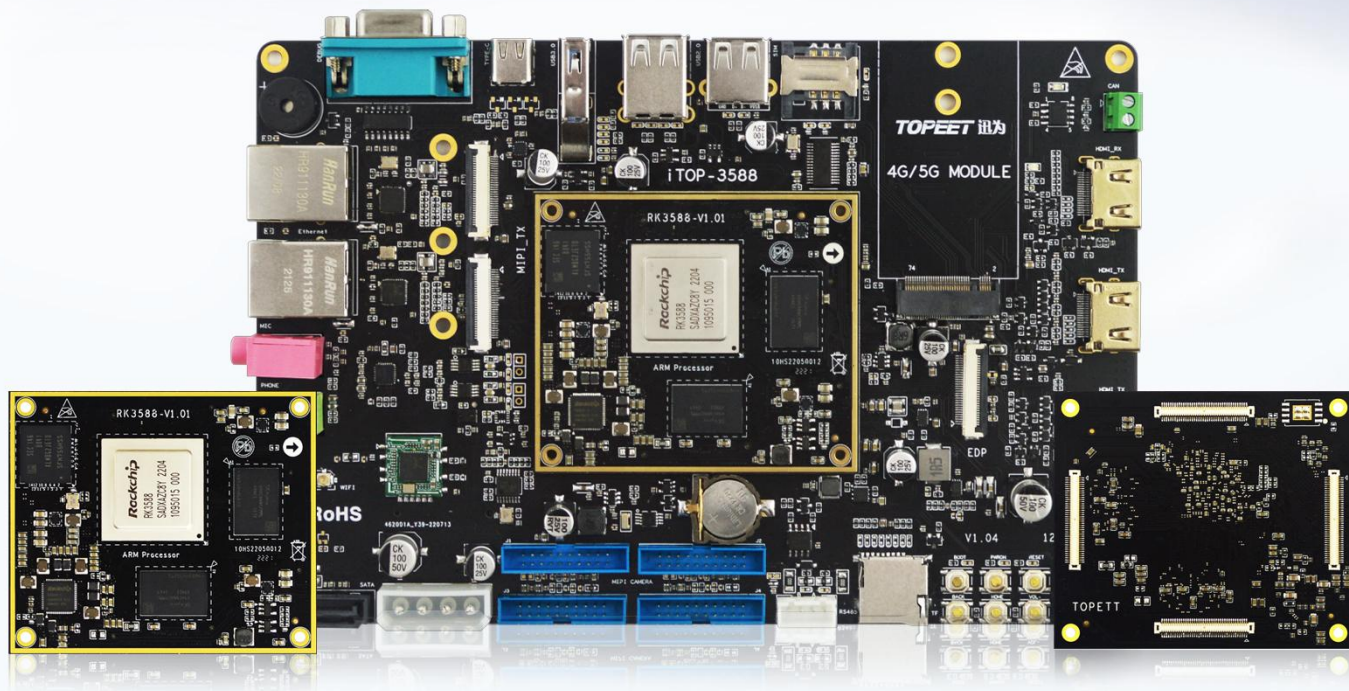


强大的 AI 能力 更快更强

超长供货周期 | 7X24 小时稳定运行 | 8K 视频编解码



iTOP-RK3588 开发板使用手册

八核 64 位 CPU | 主频 2.4GHz | NPU 算力 6T | 4800 安防级别 ISP

更新记录

更新版本	修改内容
V1.0	初版

目录

更新记录	2
目录	3
版权声明	4
更多帮助	5
第 1 章 Ubuntu 系统开发	6
1.1 ubuntu 系统安装 docker	6
1.2 ubuntu 终端密码登录	8
1.3 设置静态 IP	9
1.4 创建自启动程序	11

版权声明

本文档版权归北京迅为电子有限公司所有。未经本公司书面许可，任何单位和个人无权以任何形式复制、传播、转载本文档的任何内容，违者将被追究法律责任。

更多帮助

注意事项与维护

- ❖ 请注意和遵循标注在产品上的所有警示和指引信息；
- ❖ 请勿带电插拔核心板及外围模块；
- ❖ 使用产品之前，请仔细阅读本手册，并妥善保管，以备将来参考；
- ❖ 请使用配套电源适配器，以保证电压、电流的稳定；
- ❖ 请勿在冷热交替环境中使用本产品，避免结露损坏元器件；
- ❖ 请保持产品干燥，如果不慎被任何液体泼溅或浸润，请立刻断电并充分晾干；
- ❖ 请勿使用有机溶剂或腐蚀性液体清洗本产品；
- ❖ 请勿在多尘、脏乱的环境中使用本产品，如果长期不使用，请包装好本产品；
- ❖ 如果在震动场景使用，请做好核心板与底板的固定，避免核心板跌落损坏；
- ❖ 请勿在通电情况下，插拔核心板及外围模块(特别是串口模块)；
- ❖ 请勿自行维修、拆解本产品，如产品出现故障应及时联系本公司进行维修；
- ❖ 请勿自行修改或使用未经授权的配件，由此造成的损坏将不予保修；

资料的更新

为了确保您的资料是最新状态，请密切关注我们的动态，我们将会通过微信公众号和 QQ 群推送。

关注“迅为电子”微信公众号，不定期分享教程、资料 and 行业干货及产品一线资料。

迅为新媒体账号

官网：<https://www.topeetboard.com>

知乎：<https://www.zhihu.com/people/topeetabc123>

CSDN：<https://blog.csdn.net/BeiJingXunWei>



售后服务政策

1. 如产品使用过程中出现硬件故障可根据售后服务政策进行维修
2. 服务政策：参见官方网售后服务说明
<https://www.topeetboard.com/sydyml/Service/bx.html>

送修地址：

1. 地址：北京市海淀区永翔北路 9 号中国航发大厦三层
2. 联系人：迅为开发板售后服务部
3. 电话：010-85270716
4. 邮编：100094
5. 邮寄须知：建议使用顺丰、圆通或韵达，且不接受任何到付

技术支持范围

1. 了解产品的软、硬件资源提供情况咨询
2. 产品的软、硬件手册使用过程中遇到的问题
3. 下载和烧写更新系统过程中遇到的问题
4. 产品用户的资料丢失、更新后重新获取
5. 产品的故障判断及售后维修服务。

PS：（由于嵌入式系统知识范围广泛，我们无法保证对各种问题都能一一解答，部分内容无法供技术支持，只能提供建议。）

技术支持

1. 周一至周五：（法定节假日除外）
上午 9:00 ~ 11:30 / 下午 13:30 ~ 17:30
2. QQ 技术交流群：
824412014
822183461
95631883
861311530

第 1 章 Ubuntu 系统开发

1.1 ubuntu 系统安装 docker

使用 Docker 等容器需要 CGROUP、BRIDGE 等功能，如果 Kernel 中相关配置没有打开，安装或运行 Docker 会出现各种错误。默认迅为提供的源码，内核支持使用 docker 容器。

Docker 开源团队提供了一个脚本，可以检测 Kernel 配置是否符合 Docker 的运行要求，源地址：<https://github.com/moby/moby/blob/master/contrib/check-config.sh>

迅为也提供了检测脚本，在网盘资料“iTOP-3588 开发板/02_【iTOP-RK3588 开发板】开发资料/09_Linux 系统开发配套资料/03 Ubuntu 系统开发配套资料/01_Ubuntu 系统安装 Docker 配套资料”目录下。拷贝检测脚本到源码内核目录下，输入如下命令：

```
chmod 777 check-config.sh
./check-config.sh kernel/arch/arm64/configs/rockchip_linux_defconfig
```

结果如下图所示。结果主要是两部分，Generally Necessary: 表示必要的配置，如果有显示 missing 的地方，需要在内核配置中打开，重新编译烧录内核以支持 Docker。Optional Features: 是可选配置，根据需要打开。

```
root@ubuntu:/home/topeet/linux/rk3588_linux# ./check-config.sh kernel/arch/arm64/configs/rockchip_linux_defconfig
info: reading kernel config from kernel/arch/arm64/configs/rockchip_linux_defconfig ...

Generally Necessary:
- cgroup hierarchy: properly mounted [/sys/fs/cgroup]
- apparmor: enabled and tools installed
- CONFIG_NAMESPACES: enabled
- CONFIG_NET_NS: enabled
- CONFIG_PID_NS: enabled
- CONFIG_IPC_NS: enabled
- CONFIG_UTS_NS: enabled
- CONFIG_CGROUPS: enabled
- CONFIG_CGROUP_CPUACCT: enabled
- CONFIG_CGROUP_DEVICE: enabled
- CONFIG_CGROUP_FREEZER: enabled
- CONFIG_CGROUP_SCHED: enabled
- CONFIG_CPUSETS: enabled
- CONFIG_MEMCG: enabled
- CONFIG_KEYS: enabled
- CONFIG_VETH: enabled
- CONFIG_BRIDGE: enabled
- CONFIG_BRIDGE_NETFILTER: enabled
- CONFIG_IP_NF_FILTER: enabled
- CONFIG_IP_NF_TARGET_MASQUERADE: enabled
- CONFIG_NETFILTER_XT_MATCH_ADDRTYPE: enabled
- CONFIG_NETFILTER_XT_MATCH_CONNTRACK: enabled
- CONFIG_NETFILTER_XT_MATCH_IPVS: enabled
- CONFIG_NETFILTER_XT_MARK: enabled
- CONFIG_IP_NF_NAT: enabled
- CONFIG_NF_NAT: enabled
- CONFIG_POSIX_MQUEUE: enabled
- CONFIG_CGROUP_BPF: enabled
```

在 ubuntu 系统中默认已经提供了 docker 安装脚本 install_docker.sh，只需要在命令行输入 install_docker.sh 命令即可安装，安装过程如下所示：

```
root@topeet:~$ install_docker.sh
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
软件包 docker-ce 未安装, 所以不会被卸载
E: 无法定位软件包 docker-engine
命中:1 https://repo.huaweicloud.com/ubuntu-ports focal InRelease
命中:2 https://repo.huaweicloud.com/ubuntu-ports focal-security InRelease
命中:3 https://repo.huaweicloud.com/ubuntu-ports focal-updates InRelease
命中:4 https://repo.huaweicloud.com/ubuntu-ports focal-backports InRelease
获取:5 https://repo.huaweicloud.com/ubuntu-ports focal/main Translation-zh_CN [113 kB]
获取:6 https://repo.huaweicloud.com/ubuntu-ports focal/main arm64 DEP-11 Metadata [489 kB]
0% [6 Components-arm64 store 0 B] [正在等待报头]
```

安装完成之后输入以下命令查看 docker 的状态, 如下图所示:

systemctl status docker

```
root@topeet:~$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Sat 2024-01-13 06:48:23 UTC; 36s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 9949 (dockerd)
      Tasks: 11
     Memory: 24.8M
    CGroup: /system.slice/docker.service
            └─9949 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>

1月 13 06:48:22 topeet systemd[1]: Starting Docker Application Container Engine>
1月 13 06:48:22 topeet dockerd[9949]: time="2024-01-13T06:48:22.230591126Z" lev>
1月 13 06:48:22 topeet dockerd[9949]: time="2024-01-13T06:48:22.231707626Z" lev>
1月 13 06:48:23 topeet dockerd[9949]: time="2024-01-13T06:48:23.056036167Z" lev>
1月 13 06:48:23 topeet dockerd[9949]: time="2024-01-13T06:48:23.190887972Z" lev>
1月 13 06:48:23 topeet dockerd[9949]: time="2024-01-13T06:48:23.273437522Z" lev>
1月 13 06:48:23 topeet dockerd[9949]: time="2024-01-13T06:48:23.331148445Z" lev>
1月 13 06:48:23 topeet dockerd[9949]: time="2024-01-13T06:48:23.331573111Z" lev>
1月 13 06:48:23 topeet dockerd[9949]: time="2024-01-13T06:48:23.409895536Z" lev>
1月 13 06:48:23 topeet systemd[1]: Started Docker Application Container Engine.
lines 1-21/21 (END)
```

输入指令, 出现下图表示 docker 安装成功:

docker run hello-world

```
root@topeet:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
478afc919002: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (arm64v8)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

root@topeet:~$
```

至此 docker 就安装测试完成了。

1.2 ubuntu 终端密码登录

默认情况下 ubuntu 系统终端会自动登录 root 用户，某些情况下可能需要输入密码登录的情况，所以本小节将完善 ubuntu 终端密码登录的方法。

开发板烧写 ubuntu 系统，开发板启动之后，输入以下命令修改开机设置选项：

```
sudo vi /lib/systemd/system/serial-getty@.service
```

```
25 # rescue.target or starts rescue.service from multi-user.target or
26 # graphical.target.
27 Conflicts=rescue.service
28 Before=rescue.service
29
30 [Service]
31 # The '-o' option value tellsagetty to replace 'login' arguments with an
32 # option to preserve environment (-p), followed by '--' for safety, and then
33 # the entered username.
34 # ExecStart=/sbin/agetty -o '-p -- \\u' --keep-baud 115200,38400,9600 %I $TT
ERM
35 ExecStart=/sbin/agetty --autologin root --noclear %I $TERM
36 Type=idle
37 Restart=always
38 UtmpIdentifier=%I
```

默认情况下起作用的为上面框起来的内容，表示默认登录 root，如果想要通过输入密码登录只需要注释该行，然后取消上一行的注释即可：

```
25 # rescue.target or starts rescue.service from multi-user.target or
26 # graphical.target.
27 Conflicts=rescue.service
28 Before=rescue.service
29
30 [Service]
31 # The '-o' option value tellsagetty to replace 'login' arguments with an
32 # option to preserve environment (-p), followed by '--' for safety, and then
33 # the entered username.
34 ExecStart=/sbin/agetty -o '-p -- \u' --keep-baud 115200,38400,9600 %I $TERM
35 # ExecStart=/sbin/agetty --autologin root --noclear %I $TERM
36 Type=idle
```

保存退出之后重新启动，需要输入系统才能启动，如下图所示：

1.3 设置静态 IP

在一些情况下，需要设置静态 IP，本小节将描述 Ubuntu 系统下设置静态 IP 的方法。

烧写完 Ubuntu 系统之后如下图所示：

```
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.10.110 aarch64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

  ____ _
 |  _ \| | | | | |
 | |_) | |_| |
 |  _<|  _<|
 |_| \_| \_|_|_|

System load:   13%           Up time:           0 min
Memory usage:  9% of 7.74G   IP:            172.17.0.1 192.168.1.24
CPU temp:      47°C          Usage of /:     26% of 22G
```

由于 dns 文件/etc/resolv.conf 是一个指向/tmp/resolv.conf 的软链接，使用静态获取 IP 的方式不能正常解析 DNS 文件，所以需要使用以下命令删掉软链接文件并重新创建，如下图所示：

```
rm -rf /etc/resolv.conf
echo nameserver 8.8.8.8 >>/etc/resolv.conf

root@topeet:~$ rm -rf /etc/resolv.conf
root@topeet:~$
root@topeet:~$ echo nameserver 8.8.8.8 >>/etc/resolv.conf
```

最后使用 “vi /etc/network/interfaces” 命令修改网络配置文件，添加以下内容

```
auto eth0
iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 8.8.8.8
```

```
up route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1 dev eth0

auto eth1
iface eth1 inet static
address 192.168.1.51
netmask 255.255.255.0
dns-nameservers 8.8.8.8
up route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1 dev eth1
```

添加完成如下图所示:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source /etc/network/interfaces.d/*
auto eth0
iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 8.8.8.8
up route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1 dev eth0

auto eth1
iface eth1 inet static
address 192.168.2.51
netmask 255.255.255.0
dns-nameservers 8.8.8.8
up route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.1.1 dev eth1
```

至此，关于静态 ip 的设置就完成，接下来重启开发板，重启完成之后分别使用 “ifconfig” 命令查看 ip 地址，如下图所示：

```
root@iTOP-RK3588:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.50 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 240e:341:d09d:7100:b0a2:daff:feac:65d3 prefixlen 64 scopeid 0x0<global>
    inet6 fe80::b0a2:daff:feac:65d3 prefixlen 64 scopeid 0x20<link>
    ether b2:a2:da:ac:65:d3 txqueuelen 1000 (Ethernet)
    RX packets 33 bytes 3023 (2.9 KiB)
    RX errors 0 dropped 2 overruns 0 frame 0
    TX packets 23 bytes 1730 (1.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 74

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.51 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::aca2:daff:feac:65d3 prefixlen 64 scopeid 0x20<link>

    inet6 240e:341:d09d:7100:aca2:daff:feac:65d3 prefixlen 64 scopeid 0x0<global>
    ether ae:a2:da:ac:65:d3 txqueuelen 1000 (Ethernet)
    RX packets 42 bytes 3186 (3.1 KiB)
    RX errors 0 dropped 3 overruns 0 frame 0
    TX packets 11 bytes 962 (962.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 114
```

可以看到 eth0 和 eth1 的地址已经成功设置为了 192.168.1.50 和 192.168.2.51，然后分别受用以下命令测试两个网口的网络是否正常，如下图所示：

```
ping www.baidu.com -I eth0
ping www.baidu.com -I eth1
```

```

root@iTOP-RK3588:~$ ping www.baidu.com -I eth0
PING www.baidu.com(240e:83:205:58:0:ff:b09f:36bf) from 240e:341:d09d:7100:b0a2:daff:feac:65d3 eth0: 56 data bytes
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=3 ttl=54 time=17.4 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=4 ttl=54 time=17.4 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=5 ttl=54 time=17.5 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=6 ttl=54 time=17.3 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=7 ttl=54 time=17.7 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=8 ttl=54 time=17.6 ms
^C64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=9 ttl=54 time=17.5 ms

--- www.baidu.com ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8069ms
rtt min/avg/max/mdev = 17.314/17.587/18.331/0.282 ms
root@iTOP-RK3588:~$
root@iTOP-RK3588:~$
root@iTOP-RK3588:~$ ping www.baidu.com -I eth1
PING www.baidu.com(240e:83:205:58:0:ff:b09f:36bf) from 240e:341:d09d:7100:aca2:daff:feac:65d3 eth1: 56 data bytes
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=1 ttl=54 time=11.9 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=2 ttl=54 time=11.8 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=3 ttl=54 time=12.0 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=4 ttl=54 time=11.9 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=5 ttl=54 time=12.0 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=6 ttl=54 time=11.9 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=7 ttl=54 time=12.0 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=8 ttl=54 time=11.9 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=9 ttl=54 time=11.9 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=10 ttl=54 time=12.0 ms
64 bytes from 240e:83:205:58:0:ff:b09f:36bf: icmp_seq=11 ttl=54 time=12.0 ms
^C
--- www.baidu.com ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 11218ms
rtt min/avg/max/mdev = 11.779/11.941/12.033/0.076 ms
root@iTOP-RK3588:~$

```

1.4 创建自启动程序

1.在/root 目录下创建一个要启动的脚本文件 myscript.sh，并添加如下内容，如下图所示：

```
vi myscript.sh
```

```
#!/bin/bash  
echo 11111111111111111111111111111111 > /root/test  
echo 11111111111111111111111111111111 > /root/test  
echo 11111111111111111111111111111111 > /root/test
```

2.然后，为该脚本添加可执行权限，以使其可以运行。使用以下命令为脚本添加可执行权限：

```
chmod +x myscript.sh
```

```
root@topeet:~$ chmod +x myscript.sh
root@topeet:~$ ls
公共 模板 视频 图片 文档 下载 音乐 桌面 myscript.sh
root@topeet:~$
```

3.接下来,创建一个 `systemd` 服务文件来运行脚本。使用以下命令创建并编辑一个新文件:

```
vi /etc/systemd/system/myscript.service
```

4.然后，在编辑器中输入以下内容：

[Unit]

Description=My Script

After=network.target

[Service]

ExecStart=/root/myscript.sh //自启动文件路径，根据自己具体实际路径修改

[Install]

WantedBy=multi-user.target

```
Description=My Script
After=network.target
[Service]
ExecStart=/root/myscript.sh
[Install]
WantedBy=multi-user.target
```

5.保存并关闭文件，然后使用以下命令重新加载 `systemd` 配置：

```
systemctl daemon-reload
```

```
root@topeet:~$ systemctl daemon-reload
root@topeet:~$
root@topeet:~$
```

6.接下来，使用以下命令启用服务：

```
systemctl enable myscript.service
```

```
root@topeet:~# systemctl enable myscript.service
Created symlink /etc/systemd/system/multi-user.target.wants/myscript.service → /etc/systemd/system/myscript.service.
root@topeet:~#
```

7.最后，使用以下命令启动服务或重启系统:

```
systemctl start myscript.service
```

```
root@topeet:~$ systemctl start myscrip.service
root@topeet:~$
```

重启系统后可以看到在 `/root` 目录下生成了一个名为 `test` 的文件，文件内容如下所示：

[illegible]

注如需禁止自启动，输入 `systemctl disable myscript.service`，这个就会取消链接，下次开机时不会自启动。如需要完全移除，删除这个 `myscript.service` 以及它的脚本。

