Only the original README is guaranteed to be up-to-date.

只有原版的 README是保证最新的。

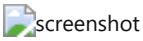
Current version is based on f4c7044

本文根据f4c7044进行翻译。

scrcpy (v1.22)

*发音为 "**scr**een **c**o**py**"*

本应用程序可以显示并控制通过 USB (或 TCP/IP) 连接的安卓设备，且不需要任何 *root* 权限。本程序支持 *GNU/Linux*, *Windows* 和 _macOS_。



本应用专注于：

- **轻量**： 原生，仅显示设备屏幕
- **性能**： 30~120fps，取决于设备
- **质量**： 分辨率可达 1920×1080 或更高
- **低延迟**： 35~70ms
- **快速启动**： 最快 1 秒内即可显示第一帧
- **无侵入性**： 不会在设备上遗留任何程序
- **用户利益**： 无需帐号，无广告，无需联网
- **自由**： 自由和开源软件

功能：

- 屏幕录制
- 镜像时关闭设备屏幕
- 双向复制粘贴
- 可配置显示质量
- 以设备屏幕作为摄像头(V4L2) (仅限 Linux)
- 模拟物理键盘 (HID) (仅限 Linux)
- 物理鼠标模拟 (HID) (仅限 Linux)
- OTG模式 (仅限 Linux)
- 更多 ……

系统要求

安卓设备最低需要支持 API 21 (Android 5.0)。

确保设备已开启 adb 调试。

在某些设备上，还需要开启额外的选项以使用鼠标和键盘进行控制。

获取本程序

概要

- Linux: `apt install scrcpy`

- Windows: [下载][direct-win64]
- macOS: `brew install scrcpy`

从源代码编译: 构建 (简化过程)

Linux

在 Debian 和 Ubuntu 上:

```
apt install scrcpy
```

在 Arch Linux 上:

```
pacman -S scrcpy
```

我们也提供 Snap 包: `scrcpy`。

对 Fedora 我们提供 COPR 包: `scrcpy`。

对 Gentoo 我们提供 Ebuild 包: `scrcpy/`。

您也可以自行构建 (简化过程)。

Windows

在 Windows 上，为简便起见，我们提供包含了所有依赖 (包括 `adb`) 的预编译包。

- README

也可以使用 Chocolatey:

```
choco install scrcpy
choco install adb    # 如果还没有 adb
```

或者 Scoop:

```
scoop install scrcpy
scoop install adb    # 如果还没有 adb
```

您也可以自行构建。

macOS

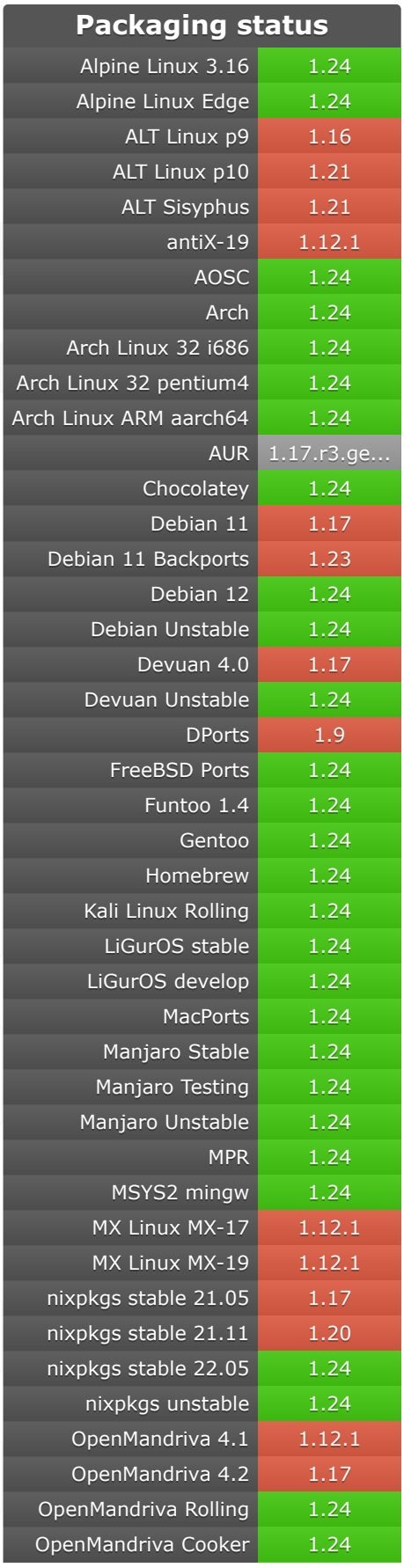
本程序已发布到 Homebrew。直接安装即可:

```
brew install scrcpy
```

你还需要在 `PATH` 内有 `adb`。如果还没有:

```
brew install android-platform-tools
```

或者通过 MacPorts，该方法同时设置好 adb:



```
sudo port install scrcpy
```

您也可以自行构建。

运行

连接安卓设备，然后执行：

```
scrcpy
```

本程序支持命令行参数，查看参数列表：

```
scrcpy --help
```

Parabola	1.24
Pardus 21	1.17
Parrot	1.17
Pisi Linux	1.24
PureOS landing	1.17
Raspbian Stable	1.17
Raspbian Testing	1.24
RPM Sphere	1.24
Scoop	1.24
SlackBuilds	1.24
Solus	1.24
Trisquel 10.0	1.12.1
Ubuntu 20.04	1.12.1
Ubuntu 22.04	1.21
Ubuntu 22.10	1.24
Void Linux x86_64	1.24

功能介绍

采集设置

降低分辨率

有时候，可以通过降低镜像的分辨率来提高性能。

要同时限制宽度和高度到某个值 (例如 1024)：

```
scrcpy --max-size 1024
scrcpy -m 1024  # 简写
```

另一边会被按比例缩小以保持设备的显示比例。这样，1920×1080 分辨率的设备会以 1024×576 的分辨率进行镜像。

修改码率

默认码率是 8 Mbps。改变视频码率 (例如改为 2 Mbps)：

```
scrcpy --bit-rate 2M
scrcpy -b 2M  # 简写
```

限制帧率

要限制采集的帧率：

```
scrcpy --max-fps 15
```

本功能从 Android 10 开始才被官方支持，但在一些旧版本中也能生效。

画面裁剪

可以对设备屏幕进行裁剪，只镜像屏幕的一部分。

例如可以只镜像 Oculus Go 的一只眼睛。

```
scrcpy --crop 1224:1440:0:0   # 以 (0,0) 为原点的 1224x1440 像素
```

如果同时指定了 `--max-size`，会先进行裁剪，再进行缩放。

锁定屏幕方向

要锁定镜像画面的方向：

```
scrcpy --lock-video-orientation      # 初始（目前）方向
scrcpy --lock-video-orientation=0    # 自然方向
scrcpy --lock-video-orientation=1    # 逆时针旋转 90°
scrcpy --lock-video-orientation=2    # 180°
scrcpy --lock-video-orientation=3    # 顺时针旋转 90°
```

只影响录制的方向。

窗口可以独立旋转。

编码器

一些设备内置了多种编码器，但是有的编码器会导致问题或崩溃。可以手动选择其它编码器：

```
scrcpy --encoder OMX.qcom.video.encoder.avc
```

要列出可用的编码器，可以指定一个不存在的编码器名称，错误信息中会包含所有的编码器：

```
scrcpy --encoder _
```

采集

屏幕录制

可以在镜像的同时录制视频：

```
scrcpy --record file.mp4
scrcpy -r file.mkv
```

仅录制，不显示镜像：

```
scrcpy --no-display --record file.mp4
scrcpy -Nr file.mkv
# 按 Ctrl+C 停止录制
```

录制时会包含“被跳过的帧”，即使它们由于性能原因没有实时显示。设备会为每一帧打上 *时间戳*，所以 包时延抖动 不会影响录制的文件。

v4l2loopback

在 Linux 上，可以将视频流发送至 v4l2 回环 (loopback) 设备，因此可以使用任何 v4l2 工具像摄像头一样打开安卓设备。

需安装 `v4l2loopback` 模块：

```
sudo apt install v4l2loopback-dkms
```

创建一个 v4l2 设备：

```
sudo modprobe v4l2loopback
```

这样会在 `/dev/videoN` 创建一个新的视频设备，其中 `N` 是整数。（更多选项 可以用来创建多个设备或者特定 ID 的设备）。

列出已启用的设备：

```
# 需要 v4l-utils 包
v4l2-ctl --list-devices

# 简单但或许足够
ls /dev/video*
```

使用一个 v4l2 漏开启 scrcpy：

```
scrcpy --v4l2-sink=/dev/videoN
scrcpy --v4l2-sink=/dev/videoN --no-display  # 禁用窗口镜像
scrcpy --v4l2-sink=/dev/videoN -N            # 简写
```

(将 `N` 替换为设备 ID，使用 `ls /dev/video*` 命令查看)

启用之后，可以使用 v4l2 工具打开视频流：

```
ffplay -i /dev/videoN
vlc v4l2:///dev/videoN   # VLC 可能存在一些缓冲延迟
```

例如，可以在 OBS 中采集视频。

缓冲

可以加入缓冲，会增加延迟，但可以减少抖动 (见 #2464)。

对于显示缓冲：

```
scrcpy --display-buffer=50  # 为显示增加 50 毫秒的缓冲
```

对于 V4L2 漏:

```
scrcpy --v4l2-buffer=500    # 为 v4l2 漏增加 500 毫秒的缓冲
```

连接

TCP/IP （无线）

Scrcpy 使用 `adb` 与设备通信，并且 `adb` 支持通过 TCP/IP 连接到设备（设备必须连接与电脑相同的网络）。

自动配置

参数 `--tcpip` 允许自动配置连接。这里有两种方式。

对于传入的 adb 连接，如果设备（在这个例子中以192.168.1.1为可用地址）已经监听了一个端口（通常是5555），运行：

```
scrcpy --tcpip=192.168.1.1        # 默认端口是5555
scrcpy --tcpip=192.168.1.1:5555
```

如果adb TCP/IP（无线） 模式在某些设备上不被启用（或者你不知道IP地址），用USB连接设备，然后运行：

```
scrcpy --tcpip     # 无需其他参数
```

这将会自动寻找设备IP地址，启用TCP/IP模式，然后在启动之前连接到设备。

手动配置

或者，可以通过 `adb` 使用手动启用 TCP/IP 连接：

1. 将设备和电脑连接至同一 Wi-Fi。
2. 打开 设置 → 关于手机 → 状态信息，获取设备的 IP 地址，也可以执行以下的命令：

   ```
   adb shell ip route | awk '{print $9}'
   ```

3. 启用设备的网络 adb 功能：`adb tcpip 5555`。
4. 断开设备的 USB 连接。
5. 连接到您的设备：`adb connect DEVICE_IP:5555` _(将 `DEVICE_IP` 替换为设备 IP)_。
6. 正常运行 `scrcpy`。

降低比特率和分辨率可能很有用：

```
scrcpy --bit-rate 2M --max-size 800
scrcpy -b2M -m800  # 简写
```

多设备

如果 `adb devices` 列出了多个设备，您必须指定设备的 *序列号*：

```
scrcpy --serial 0123456789abcdef
scrcpy -s 0123456789abcdef  # 简写
```

如果设备通过 TCP/IP 连接：

```
scrcpy --serial 192.168.0.1:5555
scrcpy -s 192.168.0.1:5555  # 简写
```

您可以同时启动多个 *scrcpy* 实例以同时显示多个设备的画面。

在设备连接时自动启动

您可以使用 AutoAdb:

```
autoadb scrcpy -s '{}'
```

隧道

要远程连接到设备，可以将本地的 adb 客户端连接到远程的 adb 服务端 (需要两端的 *adb* 协议版本相同)。

远程ADB服务器

要连接到一个远程ADB服务器，让服务器在所有接口上监听：

```
adb kill-server
adb -a nodaemon server start
# 保持该窗口开启
```

警告：所有客户端与ADB服务器的交流都是未加密的。

假设此服务器可在 192.168.1.2 访问。 然后，从另一个终端，运行 scrcpy：

```
export ADB_SERVER_SOCKET=tcp:192.168.1.2:5037
scrcpy --tunnel-host=192.168.1.2
```

默认情况下，scrcpy使用用于 `adb forward` 隧道建立的本地端口（通常是 `27183`，见 `--port`）。它也可以强制使用一个不同的隧道端口（当涉及更多的重定向时，这在更复杂的情况下可能很有用）：

```
scrcpy --tunnel-port=1234
```

SSH 隧道

为了安全地与远程ADB服务器通信，最好使用SSH隧道。

首先，确保ADB服务器正在远程计算机上运行：

```
adb start-server
```

然后，建立一个SSH隧道：

```
# 本地  5038 --> 远程  5037
# 本地 27183 <-- 远程 27183
ssh -CN -L5038:localhost:5037 -R27183:localhost:27183 your_remote_computer
# 保持该窗口开启
```

在另一个终端上，运行scrcpy：

```
export ADB_SERVER_SOCKET=tcp:localhost:5038
scrcpy
```

若要不使用远程端口转发，可以强制使用正向连接（注意是 `-L` 而不是 `-R` ）：

```
# 本地  5038 --> 远程  5037
# 本地 27183 <-- 远程 27183
ssh -CN -L5038:localhost:5037 -L27183:localhost:27183 your_remote_computer
# 保持该窗口开启
```

在另一个终端上，运行scrcpy：

```
export ADB_SERVER_SOCKET=tcp:localhost:5038
scrcpy --force-adb-forward
```

类似地，对于无线连接，可能需要降低画面质量：

```
scrcpy -b2M -m800 --max-fps 15
```

窗口设置

标题

窗口的标题默认为设备型号。可以通过如下命令修改：

```
scrcpy --window-title "我的设备"
```

位置和大小

您可以指定初始的窗口位置和大小：

```
scrcpy --window-x 100 --window-y 100 --window-width 800 --window-height 600
```

无边框

禁用窗口边框：

```
scrcpy --window-borderless
```

保持窗口在最前

您可以通过如下命令保持窗口在最前面：

```
scrcpy --always-on-top
```

全屏

您可以通过如下命令直接全屏启动 scrcpy：

```
scrcpy --fullscreen
scrcpy -f  # 简写
```

全屏状态可以通过 `MOD+f` 随时切换。

旋转

可以通过以下命令旋转窗口：

```
scrcpy --rotation 1
```

可选的值有：

- `0`：无旋转
- `1`：逆时针旋转 90°
- `2`：旋转 180°
- `3`：顺时针旋转 90°

也可以使用 MOD+← *(左箭头)* 和 MOD+→ *(右箭头)* 随时更改。

需要注意的是， *scrcpy* 中有三类旋转方向：

- MOD+r 请求设备在竖屏和横屏之间切换 (如果前台应用程序不支持请求的朝向，可能会拒绝该请求)。
- `--lock-video-orientation` 改变镜像的朝向 (设备传输到电脑的画面的朝向)。这会影响录制。
- `--rotation` (或 MOD+←/MOD+→) 只旋转窗口的内容。这只影响显示，不影响录制。

其他镜像设置

只读

禁用电脑对设备的控制 (任何可与设备交互的方式：如键盘输入、鼠标事件和文件拖放)：

```
scrcpy --no-control
scrcpy -n
```

显示屏

如果设备有多个显示屏，可以选择要镜像的显示屏：

```
scrcpy --display 1
```

可以通过如下命令列出所有显示屏的 id：

```
adb shell dumpsys display   # 在输出中搜索 "mDisplayId="
```

控制第二显示屏需要设备运行 Android 10 或更高版本 (否则将在只读状态下镜像)。

保持常亮

阻止设备在连接时一段时间后休眠：

```
scrcpy --stay-awake
scrcpy -w
```

scrcpy 关闭时会恢复设备原来的设置。

关闭设备屏幕

可以通过以下的命令行参数在关闭设备屏幕的状态下进行镜像：

```
scrcpy --turn-screen-off
scrcpy -S
```

或者在任何时候按 `MOD+o`。

要重新打开屏幕，按下 `MOD+Shift+o`。

在Android上， `电源` 按钮始终能把屏幕打开。为了方便，对于在 *scrcpy* 中发出的 `电源` 事件 (通过鼠标右键或 `MOD+p`)，会 (尽最大的努力) 在短暂的延迟后将屏幕关闭。设备上的 `电源` 按钮仍然能打开设备屏幕。

还可以同时阻止设备休眠：

```
scrcpy --turn-screen-off --stay-awake
scrcpy -Sw
```

退出时息屏

scrcpy 退出时关闭设备屏幕：

```
scrcpy --power-off-on-close
```

显示触摸

在演示时，可能会需要显示 (在物理设备上的) 物理触摸点。

Android 在 *开发者选项* 中提供了这项功能。

Scrcpy 提供一个选项可以在启动时开启这项功能并在退出时恢复初始设置：

```
scrcpy --show-touches
scrcpy -t
```

请注意这项功能只能显示 *物理* 触摸 (用手指在屏幕上的触摸)。

关闭屏保

Scrcpy 默认不会阻止电脑上开启的屏幕保护。

关闭屏幕保护：

```
scrcpy --disable-screensaver
```

输入控制

旋转设备屏幕

使用 `MOD+r` 在竖屏和横屏模式之间切换。

需要注意的是，只有在前台应用程序支持所要求的模式时，才会进行切换。

复制粘贴

每次安卓的剪贴板变化时，其内容都会被自动同步到电脑的剪贴板上。

所有的 `Ctrl` 快捷键都会被转发至设备。其中：

- `Ctrl+c` 通常执行复制
- `Ctrl+x` 通常执行剪切
- `Ctrl+v` 通常执行粘贴 (在电脑到设备的剪贴板同步完成之后)

大多数时候这些按键都会执行以上的功能。

但实际的行为取决于设备上的前台程序。例如，*Termux* 会在按下 `Ctrl+c` 时发送 SIGINT，又如 *K-9 Mail* 会新建一封邮件。

要在这种情况下进行剪切，复制和粘贴 (仅支持 Android >= 7)：

- `MOD+c` 注入 `COPY` (复制)
- `MOD+x` 注入 `CUT` (剪切)
- `MOD+v` 注入 `PASTE` (粘贴) (在电脑到设备的剪贴板同步完成之后)

另外，`MOD+Shift+v` 会将电脑的剪贴板内容转换为一串按键事件输入到设备。在应用程序不接受粘贴时 (比如 *Termux*)，这项功能可以派上一定的用场。不过这项功能可能会导致非 ASCII 编码的内容出现错误。

警告： 将电脑剪贴板的内容粘贴至设备 (无论是通过 `Ctrl+v` 还是 `MOD+v`) 都会将内容复制到设备的剪贴板。如此，任何安卓应用程序都能读取到。您应避免将敏感内容 (如密码) 通过这种方式粘贴。

一些设备不支持通过程序设置剪贴板。通过 `--legacy-paste` 选项可以修改 `Ctrl+v` 和 `MOD+v` 的工作方式，使它们通过按键事件 (同 `MOD+Shift+v`) 来注入电脑剪贴板内容。

要禁用自动剪贴板同步功能，使用 `--no-clipboard-autosync`。

双指缩放

模拟“双指缩放”：`Ctrl+`_按下并拖动鼠标_。

在按住 `Ctrl` 时按下鼠标左键，直到松开鼠标左键前，移动鼠标会使屏幕内容相对于屏幕中心进行缩放或旋转 (如果应用支持)。

具体来说，*scrcpy* 会在鼠标位置，以及鼠标以屏幕中心镜像的位置分别生成触摸事件。

物理键盘模拟 (HID)

默认情况下，scrcpy 使用安卓按键或文本注入，这在任何情况都可以使用，但仅限于ASCII字符。

在 Linux 上，scrcpy 可以模拟为 Android 上的物理 USB 键盘，以提供更好地输入体验 (使用 USB HID over AOAv2)：禁用虚拟键盘，并适用于任何字符和输入法。

不过，这种方法仅支持 USB 连接以及 Linux平台。

启用 HID 模式：

```
scrcpy --hid-keyboard
scrcpy -K  # 简写
```

如果失败了 (如设备未通过 USB 连接)，则自动回退至默认模式 (终端中会输出日志)。这即允许通过 USB 和 TCP/IP 连接时使用相同的命令行参数。

在这种模式下，原始按键事件 (扫描码) 被发送给设备，而与宿主机按键映射无关。因此，若键盘布局不匹配，需要在 Android 设备上进行配置，具体为 设置 → 系统 → 语言和输入法 → 实体键盘。

物理鼠标模拟 (HID)

与物理键盘模拟类似，可以模拟一个物理鼠标。 同样，它仅在设备通过 USB 连接时才有效，并且目前仅在 Linux 上受支持。

默认情况下，scrcpy 使用 Android 鼠标事件注入，使用绝对坐标。 通过模拟物理鼠标，在Android设备上出现鼠标指针，并注入鼠标相对运动、点击和滚动。

启用此模式:

```
scrcpy --hid-mouse
scrcpy -M  # 简写
```

您还可以将 `--forward-all-clicks` 添加到 转发所有点击.

启用此模式后，计算机鼠标将被"捕获"（鼠标指针从计算机上消失并出现在 Android 设备上）。

特殊的捕获键，`Alt` 或 `Super`，切换（禁用或启用）鼠标捕获。 使用其中之一将鼠标的控制权交还给计算机。

OTG

可以仅使用物理键盘和鼠标模拟 (HID) 运行 _scrcpy_，就好像计算机键盘和鼠标通过 OTG 线直接插入设备一样。

在这个模式下，*adb* (USB 调试)是不必要的，且镜像被禁用。

启用 OTG 模式:

```
scrcpy --otg
# 如果有多个 USB 设备可用，则通过序列号选择
scrcpy --otg -s 0123456789abcdef
```

只开启 HID 键盘 或 HID 鼠标 是可行的:

```
scrcpy --otg --hid-keyboard              # 只开启 HID 键盘
scrcpy --otg --hid-mouse                 # 只开启 HID 鼠标
scrcpy --otg --hid-keyboard --hid-mouse  # 开启 HID 键盘 和 HID 鼠标
# 为了方便，默认两者都开启
scrcpy --otg                             # 开启 HID 键盘 和 HID 鼠标
```

像 `--hid-keyboard` 和 `--hid-mouse` 一样，它只在设备通过 USB 连接时才有效，且目前仅在 Linux 上支持。

文本注入偏好

输入文字的时候，系统会产生两种事件:

- *按键事件*，代表一个按键被按下或松开。
- *文本事件*，代表一个字符被输入。

程序默认使用按键事件来输入字母。只有这样，键盘才会在游戏中正常运作 (例如 WASD 键)。

但这也有可能造成一些问题。如果您遇到了问题，可以通过以下方式避免:

```
scrcpy --prefer-text
```

(但这会导致键盘在游戏中工作不正常)

相反，您可以强制始终注入原始按键事件：

```
scrcpy --raw-key-events
```

该选项不影响 HID 键盘 (该模式下，所有按键都发送为扫描码)。

按键重复

默认状态下，按住一个按键不放会生成多个重复按键事件。在某些游戏中这通常没有实际用途，且可能会导致性能问题。

避免转发重复按键事件：

```
scrcpy --no-key-repeat
```

该选项不影响 HID 键盘 (该模式下，按键重复由 Android 直接管理)。

右键和中键

默认状态下，右键会触发返回键 (或电源键开启)，中键会触发 HOME 键。要禁用这些快捷键并把所有点击转发到设备：

```
scrcpy --forward-all-clicks
```

文件拖放

安装APK

将 APK 文件 (文件名以 `.apk` 结尾) 拖放到 *scrcpy* 窗口来安装。

不会有视觉反馈，终端会输出一条日志。

将文件推送至设备

要推送文件到设备的 `/sdcard/Download/`，将 (非 APK) 文件拖放至 *scrcpy* 窗口。

不会有视觉反馈，终端会输出一条日志。

在启动时可以修改目标目录：

```
scrcpy --push-target=/sdcard/Movies/
```

音频转发

Scrcpy 不支持音频。请使用 sndcpy。

另见 issue #14。

快捷键

在以下列表中, `MOD` 是快捷键的修饰键。 默认是 (左) `Alt` 或 (左) `Super`。

您可以使用 `--shortcut-mod` 来修改。可选的按键有 `lctrl`、`rctrl`、`lalt`、`ralt`、`lsuper` 和 `rsuper`。例如：

```
# 使用右 Ctrl 键
scrcpy --shortcut-mod=rctrl

# 使用左 Ctrl 键 + 左 Alt 键，或 Super 键
scrcpy --shortcut-mod=lctrl+lalt,lsuper
```

Super 键通常是指 Windows 或 Cmd 键。

操作	快捷键
全屏	MOD+f
向左旋转屏幕	MOD+← *(左箭头)*
向右旋转屏幕	MOD+→ *(右箭头)*
将窗口大小重置为1:1 (匹配像素)	MOD+g
将窗口大小重置为消除黑边	MOD+w | *双击左键*[1]
点按 主屏幕	MOD+h | *中键*
点按 返回	MOD+b | *右键*[2]
点按 切换应用	MOD+s | *第4键*[3]
点按 菜单 (解锁屏幕)[4]	MOD+m
点按 音量+	MOD+↑ *(上箭头)*
点按 音量-	MOD+↓ *(下箭头)*
点按 电源	MOD+p
打开屏幕	*鼠标右键*[2]
关闭设备屏幕 (但继续在电脑上显示)	MOD+o
打开设备屏幕	MOD+Shift+o
旋转设备屏幕	MOD+r
展开通知面板	MOD+n | *第5键*[3]
展开设置面板	MOD+n+n | *双击第5键*[3]
收起通知面板	MOD+Shift+n
复制到剪贴板[5]	MOD+c
剪切到剪贴板[5]	MOD+x
同步剪贴板并粘贴[5]	MOD+v
注入电脑剪贴板文本	MOD+Shift+v

打开/关闭FPS显示 (至标准输出)	`MOD`+`i`
捏拉缩放	`Ctrl`+*按住并移动鼠标*
拖放 APK 文件	从电脑安装 APK 文件
拖放非 APK 文件	将文件推送至设备

[1]双击黑边可以去除黑边。
[2]点击鼠标右键将在屏幕熄灭时点亮屏幕，其余情况则视为按下返回键 。
[3]鼠标的第4键和第5键。
[4]对于开发中的 react-native 应用程序，`MENU` 触发开发菜单。
[5]需要安卓版本 Android >= 7。

有重复按键的快捷键通过松开再按下一个按键来进行，如"展开设置面板"：

1. 按下 `MOD` 不放。
2. 双击 `n`。
3. 松开 `MOD`。

所有的 `Ctrl`+*按键* 的快捷键都会被转发到设备，所以会由当前应用程序进行处理。

自定义路径

要使用指定的 *adb* 二进制文件，可以设置环境变量 `ADB`：

```
ADB=/path/to/adb scrcpy
```

要覆盖 `scrcpy-server` 的路径，可以设置 `SCRCPY_SERVER_PATH`。

要覆盖图标，可以设置其路径至 `SCRCPY_ICON_PATH`。

为什么叫 *scrcpy*？

一个同事让我找出一个和 gnirehtet 一样难以发音的名字。

`strcpy` 源于 **str**ing （字符串）； `scrcpy` 源于 **scr**een （屏幕）。

如何构建?

请查看 BUILD。

常见问题

请查看 FAQ。

开发者

请查看开发者页面。

许可协议

相关文章

- Introducing scrcpy
- Scrcpy now works wirelessly