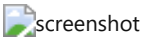
*Only the original README is guaranteed to be up-to-date.*

# scrcpy (v1.19)

このアプリケーションはUSB(もしくはTCP/IP経由)で接続されたAndroidデバイスの表示と制御を提供します。このアプリケーションは *root* でのアクセスを必要としません。このアプリケーションは *GNU/Linux* 、 *Windows* そして *macOS* 上で動作します。


screenshot

以下に焦点を当てています:

- **軽量** (ネイティブ、デバイス画面表示のみ)
- **パフォーマンス** (30~60fps)
- **クオリティ** (1920x1080以上)
- **低遅延** (35~70ms)
- **短い起動時間** (初回画像を1秒以内に表示)
- **非侵入型** (デバイスに何もインストールされていない状態になる)

## 必要要件

AndroidデバイスはAPI21(Android 5.0)以上。

Androidデバイスでadbデバッグが有効であること。

一部のAndroidデバイスでは、キーボードとマウスを使用して制御する追加オプションを有効にする必要がある。

## アプリの取得

### Linux

Debian (*testing* と *sid*) とUbuntu(20.04):

```
apt install scrcpy
```

Snapパッケージが利用可能: `scrcpy`

Fedora用COPRパッケージが利用可能: `scrcpy`

Arch Linux用AURパッケージが利用可能: `scrcpy`

Gentoo用Ebuildが利用可能: `scrcpy`

自分でビルドも可能（心配しないでください、それほど難しくはありません。）

### Windows

Windowsでは簡単に、（`adb` を含む）すべての依存関係を構築済みのアーカイブを利用可能です。

- README

Chocolateyでも利用可能です:

```
choco install scrcpy
choco install adb    # まだ入手していない場合
```

Scoopでも利用可能です:

```
scoop install scrcpy
scoop install adb    # まだ入手していない場合
```

また、アプリケーションをビルドすることも可能です。

### macOS

アプリケーションはHomebrewで利用可能です。ただインストールするだけです。

```
brew install scrcpy
```

`PATH` からアクセス可能な `adb` が必要です。もし持っていない場合はインストールしてください。

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

`adb` はMacPortsからでもインストールできます。

```
sudo port install scrcpy
```

また、アプリケーションをビルドすることも可能です。

## 実行

Androidデバイスを接続し、実行:

```
scrcpy
```

次のコマンドでリストされるコマンドライン引数も受け付けます:

```
scrcpy --help
```

## 機能

### キャプチャ構成

#### サイズ削減

Androidデバイスを低解像度でミラーリングする場合、パフォーマンス向上に便利な場合があります。

幅と高さをある値(例：1024)に制限するには:

| Packaging status | |
|---|---|
| Alpine Linux 3.16 | 1.24 |
| Alpine Linux Edge | 1.24 |
| ALT Linux p9 | 1.16 |
| ALT Linux p10 | 1.21 |
| ALT Sisyphus | 1.21 |
| antiX-19 | 1.12.1 |
| AOSC | 1.24 |
| Arch | 1.24 |
| Arch Linux 32 i686 | 1.24 |
| Arch Linux 32 pentium4 | 1.24 |
| Arch Linux ARM aarch64 | 1.24 |
| AUR | 1.17.r3.ge... |
| Chocolatey | 1.24 |
| Debian 11 | 1.17 |
| Debian 11 Backports | 1.23 |
| Debian 12 | 1.24 |
| Debian Unstable | 1.24 |
| Devuan 4.0 | 1.17 |
| Devuan Unstable | 1.24 |
| DPorts | 1.9 |
| FreeBSD Ports | 1.24 |
| Funtoo 1.4 | 1.24 |
| Gentoo | 1.24 |
| Homebrew | 1.24 |
| Kali Linux Rolling | 1.24 |
| LiGurOS stable | 1.24 |
| LiGurOS develop | 1.24 |
| MacPorts | 1.24 |
| Manjaro Stable | 1.24 |
| Manjaro Testing | 1.24 |
| Manjaro Unstable | 1.24 |
| MPR | 1.24 |
| MSYS2 mingw | 1.24 |
| MX Linux MX-17 | 1.12.1 |
| MX Linux MX-19 | 1.12.1 |
| nixpkgs stable 21.05 | 1.17 |
| nixpkgs stable 21.11 | 1.20 |
| nixpkgs stable 22.05 | 1.24 |
| nixpkgs unstable | 1.24 |
| OpenMandriva 4.1 | 1.12.1 |
| OpenMandriva 4.2 | 1.17 |
| OpenMandriva Rolling | 1.24 |
| OpenMandriva Cooker | 1.24 |

```
scrcpy --max-size 1024
scrcpy -m 1024  # 短縮版
```

一方のサイズはデバイスのアスペクト比が維持されるように計算されます。この方法では、1920x1080のデバイスでは1024x576にミラーリングされます。

#### ビットレート変更

ビットレートの初期値は8Mbpsです。ビットレートを変更するには(例:2Mbpsに変更):

```
scrcpy --bit-rate 2M
scrcpy -b 2M  # 短縮版
```

#### フレームレート制限

キャプチャするフレームレートを制限できます:

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

この機能はAndroid 10からオフィシャルサポートとなっていますが、以前のバージョンでも動作する可能性があります。

#### トリミング

デバイスの画面は、画面の一部のみをミラーリングするようにトリミングできます。

これは、例えばOculus Goの片方の目をミラーリングする場合に便利です。:

```
scrcpy --crop 1224:1440:0:0   # オフセット位置(0,0)で1224x1440
```

もし `--max-size` も指定されている場合、トリミング後にサイズ変更が適用されます。

#### ビデオの向きをロックする

ミラーリングの向きをロックするには:

```
scrcpy --lock-video-orientation      # 現在の向き
scrcpy --lock-video-orientation=0    # 自然な向き
scrcpy --lock-video-orientation=1    # 90°反時計回り
scrcpy --lock-video-orientation=2    # 180°
scrcpy --lock-video-orientation=3    # 90°時計回り
```

この設定は録画の向きに影響します。

ウィンドウは独立して回転することもできます。

#### エンコーダ

いくつかのデバイスでは一つ以上のエンコーダを持ちます。それらのいくつかは、問題やクラッシュを引き起こします。別のエンコーダを選択することが可能です:

| | |
|---|---|
| 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 |

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

利用可能なエンコーダをリストするために、無効なエンコーダ名を渡すことができます。エラー表示で利用可能なエンコーダを提供します。

```
scrcpy --encoder _
```

### キャプチャ

#### 録画

ミラーリング中に画面の録画をすることが可能です:

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

録画中にミラーリングを無効にするには:

```
scrcpy --no-display --record file.mp4
scrcpy -Nr file.mkv
# Ctrl+Cで録画を中断する
```

"スキップされたフレーム"は(パフォーマンス上の理由で)リアルタイムで表示されなくても録画されます。

フレームはデバイス上で *タイムスタンプされる* ため パケット遅延のバリエーション は録画されたファイルに影響を与えません。

#### v4l2loopback

Linuxでは、ビデオストリームをv4l2ループバックデバイスに送信することができます。 v4l2loopbackのデバイスにビデオストリームを送信することで、Androidデバイスをウェブカメラのようにv4l2対応ツールで開くこともできます。

`v4l2loopback` モジュールのインストールが必要です。

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

v4l2デバイスを作成する。

```
sudo modprobe v4l2loopback
```

これにより、新しいビデオデバイスが `/dev/videoN` に作成されます。 ( `N` は整数) (複数のデバイスや特定のIDのデバイスを作成するために、より多くのオプションが利用可能です。 多くのオプションが利用可能で複数のデバイスや特定のIDのデバイスを作成できます。

有効なデバイスを一覧表示する:

```
# v4l-utilsパッケージが必要
v4l2-ctl --list-devices
```

```
# シンプルですが十分これで確認できます
ls /dev/video*
```

v4l2シンクを使用してscrcpyを起動する。

```
scrcpy --v4l2-sink=/dev/videoN
scrcpy --v4l2-sink=/dev/videoN --no-display  # ミラーリングウィンドウを無効化する
scrcpy --v4l2-sink=/dev/videoN -N            # 短縮版
```

( `N` をデバイス ID に置き換えて、 `ls /dev/video*` で確認してください) 有効にすると、v4l2対応のツールでビデオストリームを開けます。

```
ffplay -i /dev/videoN
vlc v4l2:///dev/videoN   # VLCではバッファリングの遅延が発生する場合があります
```

例えばですが OBSの中にこの映像を取り込めことができます。

#### Buffering

バッファリングを追加することも可能です。これによりレイテンシーは増加しますが、ジッターは減少します。（参照 #2464)

このオプションでディスプレイバッファリングを設定できます。

```
scrcpy --display-buffer=50  # ディスプレイに50msのバッファリングを追加する
```

V4L2の場合はこちらのオプションで設定できます。

```
scrcpy --v4l2-buffer=500    # add 500 ms buffering for v4l2 sink
```

### 接続

#### ワイヤレス

*Scrcpy* はデバイスとの通信に `adb` を使用します。そして `adb` はTCP/IPを介しデバイスに接続することができます:

1. あなたのコンピュータと同じWi-Fiに接続します。
2. あなたのIPアドレスを取得します。設定 → 端末情報 → ステータス情報、もしくは、このコマンドを実行します:

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

3. あなたのデバイスでTCP/IPを介したadbを有効にします: `adb tcpip 5555`
4. あなたのデバイスの接続を外します。
5. あなたのデバイスに接続します: `adb connect DEVICE_IP:5555` ( *`DEVICE_IP` は置き換える)*
6. 通常通り `scrcpy` を実行します。

この方法はビットレートと解像度を減らすのにおそらく有用です:

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

#### マルチデバイス

もし `adb devices` でいくつかのデバイスがリストされる場合、 *シリアルナンバー* を指定する必要があります:

```
scrcpy --serial 0123456789abcdef
scrcpy -s 0123456789abcdef  # 短縮版
```

デバイスがTCP/IPを介して接続されている場合:

```
scrcpy --serial 192.168.0.1:5555
scrcpy -s 192.168.0.1:5555  # 短縮版
```

複数のデバイスに対して、複数の *scrcpy* インスタンスを開始することができます。

#### デバイス接続での自動起動

AutoAdbを使用可能です:

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

#### SSHトンネル

リモートデバイスに接続するため、ローカル `adb` クライアントからリモート `adb` サーバーへ接続することが可能です(同じバージョンの *adb* プロトコルを使用している場合):

```
adb kill-server    # 5037ポートのローカルadbサーバーを終了する
ssh -CN -L5037:localhost:5037 -R27183:localhost:27183 your_remote_computer
# オープンしたままにする
```

他の端末から:

```
scrcpy
```

リモートポート転送の有効化を回避するためには、代わりに転送接続を強制することができます( `-R` の代わりに `-L` を使用することに注意):

```
adb kill-server    # 5037ポートのローカルadbサーバーを終了する
ssh -CN -L5037:localhost:5037 -L27183:localhost:27183 your_remote_computer
# オープンしたままにする
```

他の端末から:

```
scrcpy --force-adb-forward
```

ワイヤレス接続と同様に、クオリティを下げると便利な場合があります:

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

### ウィンドウ構成

#### タイトル

ウィンドウのタイトルはデバイスモデルが初期値です。これは変更できます:

```
scrcpy --window-title 'My device'
```

#### 位置とサイズ

ウィンドウの位置とサイズの初期値を指定できます:

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

#### ボーダーレス

ウィンドウの装飾を無効化するには:

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

#### 常に画面のトップ

scrcpyの画面を常にトップにするには:

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

#### フルスクリーン

アプリケーションを直接フルスクリーンで開始できます:

```
scrcpy --fullscreen
scrcpy -f  # 短縮版
```

フルスクリーンは、次のコマンドで動的に切り替えることができます `MOD+f`

#### 回転

ウィンドウは回転することができます:

```
scrcpy --rotation 1
```

設定可能な値:

- `0` : 回転なし
- `1` : 90° 反時計回り
- `2` : 180°
- `3` : 90° 時計回り

回転は次のコマンドで動的に変更することができます。 MOD+←(左) 、 MOD+→(右)

*scrcpy* は3つの回転を管理することに注意:

- MOD+rはデバイスに縦向きと横向きの切り替えを要求する(現在実行中のアプリで要求している向きをサポートしていない場合、拒否することがある)
- `--lock-video-orientation` は、ミラーリングする向きを変更する(デバイスからPCへ送信される向き)。録画に影響します。
- `--rotation` (もしくはMOD+←/MOD+→)は、ウィンドウのコンテンツのみを回転します。これは表示にのみに影響し、録画には影響しません。

## 他のミラーリングオプション

### Read-only　リードオンリー

制御を無効にするには(デバイスと対話する全てのもの:入力キー、マウスイベント、ファイルのドラッグ&ドロップ):

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

### ディスプレイ

いくつか利用可能なディスプレイがある場合、ミラーリングするディスプレイを選択できます:

```
scrcpy --display 1
```

ディスプレイIDのリストは次の方法で取得できます:

```
adb shell dumpsys display   # search "mDisplayId=" in the output
```

セカンダリディスプレイは、デバイスが少なくともAndroid 10の場合にコントロール可能です。(それ以外ではリードオンリーでミラーリングされます)

### 起動状態にする

デバイス接続時、少し遅れてからデバイスのスリープを防ぐには:

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

scrcpyが閉じられた時、初期状態に復元されます。

### 画面OFF

コマンドラインオプションを使用することで、ミラーリングの開始時にデバイスの画面をOFFにすることができます:

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

もしくは、MOD+oを押すことでいつでもできます。

元に戻すには、`MOD+Shift+o`を押します。

Androidでは、`POWER`ボタンはいつでも画面を表示します。便宜上、`POWER`がscrcpyを介して(右クリックもしくは`MOD+p`を介して)送信される場合、(ベストエフォートベースで)少し遅れて、強制的に画面を非表示にします。ただし、物理的な`POWER`ボタンを押した場合は、画面は表示されます。

このオプションはデバイスがスリープしないようにすることにも役立ちます:

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

#### タッチを表示

プレゼンテーションの場合(物理デバイス上で)物理的なタッチを表示すると便利な場合があります。

Androidはこの機能を *開発者オプション* で提供します。

*Scrcpy* は開始時にこの機能を有効にし、終了時に初期値を復元するオプションを提供します:

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

(デバイス上で指を使った) *物理的な* タッチのみ表示されることに注意してください。

#### スクリーンセーバー無効

初期状態では、scrcpyはコンピュータ上でスクリーンセーバーが実行される事を妨げません。

これを無効にするには:

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

### 入力制御

#### デバイス画面の回転

`MOD+r`を押すことで、縦向きと横向きを切り替えます。

フォアグラウンドのアプリケーションが要求された向きをサポートしている場合のみ回転することに注意してください。

#### コピー-ペースト

Androidのクリップボードが変更される度に、コンピュータのクリップボードに自動的に同期されます。

`Ctrl`のショートカットは全てデバイスに転送されます。特に:

- `Ctrl+c` 通常はコピーします
- `Ctrl+x` 通常はカットします
- `Ctrl+v` 通常はペーストします(コンピュータとデバイスのクリップボードが同期された後)

通常は期待通りに動作します。

しかしながら、実際の動作はアクティブなアプリケーションに依存します。例えば、*Termux* は代わりに`Ctrl+c`でSIGINTを送信します、そして、*K-9 Mail* は新しいメッセージを作成します。

このようなケースでコピー、カットそしてペーストをするには(Android 7以上でのサポートのみですが):

- `MOD+c` `COPY` を挿入
- `MOD+x` `CUT` を挿入
- `MOD+v` `PASTE` を挿入(コンピュータとデバイスのクリップボードが同期された後)

加えて、`MOD+Shift+v`はコンピュータのクリップボードテキストにキーイベントのシーケンスとして挿入することを許可します。これはコンポーネントがテキストのペーストを許可しない場合(例えば *Termux*)に有用ですが、非ASCIIコンテンツを壊す可能性があります。

**警告:** デバイスにコンピュータのクリップボードを(`Ctrl+v`または`MOD+v`を介して)ペーストすることは、デバイスのクリップボードにコンテンツをコピーします。結果としてどのAndoridアプリケーションもそのコンテンツを読み取ることができます。機密性の高いコンテンツ(例えばパスワードなど)をこの方法でペーストすることは避けてください。

プログラムでデバイスのクリップボードを設定した場合、一部のデバイスは期待どおりに動作しません。`--legacy-paste` オプションは、コンピュータのクリップボードテキストをキーイベントのシーケンスとして挿入するため(`MOD+Shift+v`と同じ方法)、`Ctrl+v`と`MOD+v`の動作の変更を提供します。

#### ピンチしてズームする

"ピンチしてズームする"をシミュレートするには: `Ctrl`+*クリック&移動*

より正確にするには、左クリックボタンを押している間、`Ctrl`を押したままにします。左クリックボタンを離すまで、全てのマウスの動きは、(アプリでサポートされている場合)画面の中心を基準として、コンテンツを拡大縮小および回転します。

具体的には、scrcpyは画面の中央を反転した位置にある"バーチャルフィンガー"から追加のタッチイベントを生成します。

#### テキストインジェクション環境設定

テキストをタイプした時に生成される2種類のイベントがあります:

- *key events*　はキーを押したときと離したことを通知します。
- *text events*　はテキストが入力されたことを通知します。

初期状態で、文字はキーイベントで挿入されるため、キーボードはゲームで期待通りに動作します(通常はWASDキー)。

しかし、これは問題を引き起こすかもしれません。もしこのような問題が発生した場合は、この方法で回避できます:

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

(しかしこの方法はゲームのキーボードの動作を壊します)

#### キーの繰り返し

初期状態では、キーの押しっぱなしは繰り返しのキーイベントを生成します。これらのイベントが使われない場合でも、この方法は一部のゲームでパフォーマンスの問題を引き起す可能性があります。

繰り返しのキーイベントの転送を回避するためには:

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

#### 右クリックと真ん中クリック

初期状態では、右クリックはバックの動作(もしくはパワーオン)を起こし、真ん中クリックではホーム画面へ戻ります。このショートカットを無効にし、代わりにデバイスへクリックを転送するには:

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

### ファイルのドロップ

#### APKのインストール

APKをインストールするには、( `.apk` で終わる)APKファイルを *scrcpy* の画面にドラッグ&ドロップします。

見た目のフィードバックはありません。コンソールにログが出力されます。

#### デバイスにファイルを送る

デバイスの `/sdcard/Download` ディレクトリにファイルを送るには、(APKではない)ファイルを *scrcpy* の画面にドラッグ&ドロップします。

見た目のフィードバックはありません。コンソールにログが出力されます。

転送先ディレクトリを起動時に変更することができます:

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

### 音声転送

音声は *scrcpy* では転送されません。sndcpyを使用します。

issue #14も参照ください。

## ショートカット

次のリストでは、`MOD`でショートカット変更します。初期状態では、(left)`Alt`または(left)`Super`です。

これは `--shortcut-mod` で変更することができます。可能なキーは `lctrl` 、 `rctrl` 、 `lalt` 、 `ralt` 、 `lsuper` そして `rsuper` です。例えば:

```
# RCtrlをショートカットとして使用します
scrcpy --shortcut-mod=rctrl

# ショートカットにLCtrl+LAltまたはLSuperのいずれかを使用します
scrcpy --shortcut-mod=lctrl+lalt,lsuper
```

*`Super`は通常`Windows`もしくは`Cmd`キーです。*

| アクション | ショートカット |
| --- | --- |
| フルスクリーンモードへの切り替え | `MOD`+`f` |
| ディスプレイを左に回転 | `MOD`+`←` *(左)* |
| ディスプレイを右に回転 | `MOD`+`→` *(右)* |

| | |
|---|---|
| ウィンドウサイズを変更して1:1に変更(ピクセルパーフェクト) | MOD+g |
| ウィンドウサイズを変更して黒い境界線を削除 | MOD+w \| *ダブルクリック*[1] |
| HOMEをクリック | MOD+h \| *真ん中クリック* |
| BACKをクリック | MOD+b \| *右クリック*[2] |
| APP_SWITCHをクリック | MOD+s \| *4クリック*[3] |
| MENU (画面のアンロック)をクリック | MOD+m |
| VOLUME_UPをクリック | MOD+↑ *(上)* |
| VOLUME_DOWNをクリック | MOD+↓ *(下)* |
| POWERをクリック | MOD+p |
| 電源オン | *右クリック*[2] |
| デバイス画面をオフにする(ミラーリングしたまま) | MOD+o |
| デバイス画面をオンにする | MOD+Shift+o |
| デバイス画面を回転する | MOD+r |
| 通知パネルを展開する | MOD+n \| *5ボタンクリック*[3] |
| 設定パネルを展開する | MOD+n+n \| *5ダブルクリック*[3] |
| 通知パネルを折りたたむ | MOD+Shift+n |
| クリップボードへのコピー[3] | MOD+c |
| クリップボードへのカット[3] | MOD+x |
| クリップボードの同期とペースト[3] | MOD+v |
| コンピュータのクリップボードテキストの挿入 | MOD+Shift+v |
| FPSカウンタ有効/無効(標準入出力上) | MOD+i |
| ピンチしてズームする | Ctrl+ *クリック&移動* |

*[1]黒い境界線を削除するため、境界線上でダブルクリック*
*[2]もしスクリーンがオフの場合、右クリックでスクリーンをオンする。それ以外の場合はBackを押します.*
*[3]4と5はマウスのボタンです、もしあなたのマウスにボタンがあれば使えます.*
*[4]Android 7以上のみ.*

キーを繰り返すショートカットはキーを離して2回目を押したら実行されます。例えば「設定パネルを展開する」を実行する場合は以下のように操作する。

1. MOD キーを押し、押したままにする.
2. その後に nキーを2回押す.
3. 最後に MODキーを離す.

全てのCtrl+ キー ショートカットはデバイスに転送されます、そのためアクティブなアプリケーションによって処理されます。

## カスタムパス

特定の *adb* バイナリを使用する場合、そのパスを環境変数 `ADB` で構成します:

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

`scrcpy-server` ファイルのパスを上書きするには、 `SCRCPY_SERVER_PATH` でそのパスを構成します。

## なぜ *scrcpy*?

同僚が私に、gnirehtetのように発音できない名前を見つけるように要求しました。

`strcpy` は**str**ingをコピーします。 `scrcpy` は**scr**eenをコピーします。

## ビルド方法は?

BUILDを参照してください。

## よくある質問

FAQを参照してください。

## 開発者

開発者のページを読んでください。

## ライセンス

```
Copyright (C) 2018 Genymobile
Copyright (C) 2018-2022 Romain Vimont

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

## 記事

- Introducing scrcpy
- Scrcpy now works wirelessly