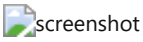
Only the original README is guaranteed to be up-to-date.

scrcpy (v1.22)

*ausgesprochen "**scr**een **c**o**py**"*

Diese Anwendung liefert sowohl Anzeige als auch Steuerung eines Android-Gerätes über USB (oder über TCP/IP). Dabei wird kein *root* Zugriff benötigt. Die Anwendung funktioniert unter *GNU/Linux*, *Windows* und *macOS*.



Dabei liegt der Fokus auf:

- **Leichtigkeit**: native, nur Anzeige des Gerätedisplays
- **Leistung**: 30~120fps, abhängig vom Gerät
- **Qualität**: 1920×1080 oder mehr
- **Geringe Latenz**: 35~70ms
- **Kurze Startzeit**: ~1 Sekunde um das erste Bild anzuzeigen
- **Keine Aufdringlichkeit**: Es wird keine installierte Software auf dem Gerät zurückgelassen
- **Nutzervorteile**: kein Account, keine Werbung, kein Internetzugriff notwendig
- **Freiheit**: gratis und open-source

Die Features beinhalten:

- Aufnahme
- Spiegeln mit ausgeschaltetem Bildschirm
- Copy&Paste in beide Richtungen
- Einstellbare Qualität
- Gerätebildschirm als Webcam (V4L2) (nur Linux)
- Simulation einer physischen Tastatur (HID) (nur Linux)
- Simulation einer physischen Maus (HID) (nur Linux)
- OTG Modus (nur Linux)
- und mehr...

Voraussetzungen

Das Android-Gerät benötigt mindestens API 21 (Android 5.0).

Es muss sichergestellt sein, dass adb debugging auf dem Gerät aktiv ist.

Auf manchen Geräten müssen zudem weitere Optionen aktiv sein um das Gerät mit Maus und Tastatur steuern zu können.

Installation der App

Zusammenfassung

- Linux: `apt install scrcpy`
- Windows: download (siehe README)
- macOS: `brew install scrcpy`

Direkt von der Source bauen: BUILD (vereinfachter Prozess (englisch))

Linux

Auf Debian und Ubuntu:

```
apt install scrcpy
```

Auf Arch Linux:

```
pacman -S scrcpy
```

Ein Snap package ist verfügbar: `scrcpy` .

Für Fedora ist ein COPR package verfügbar: `scrcpy` .

Für Gentoo ist ein Ebuild verfügbar: `scrcpy/` .

Die App kann zudem manuell gebaut werden (vereinfachter Prozess (englisch)).

Windows

Für Windows ist der Einfachheit halber ein vorgebautes Archiv mit allen Abhängigkeiten (inklusive `adb`) vorhanden.

- README

Es ist zudem in Chocolatey vorhanden:

```
choco install scrcpy
choco install adb    # falls noch nicht vorhanden
```

Und in Scoop:

```
scoop install scrcpy
scoop install adb    # falls noch nicht vorhanden
```

Die App kann zudem manuell gebaut werden.

macOS

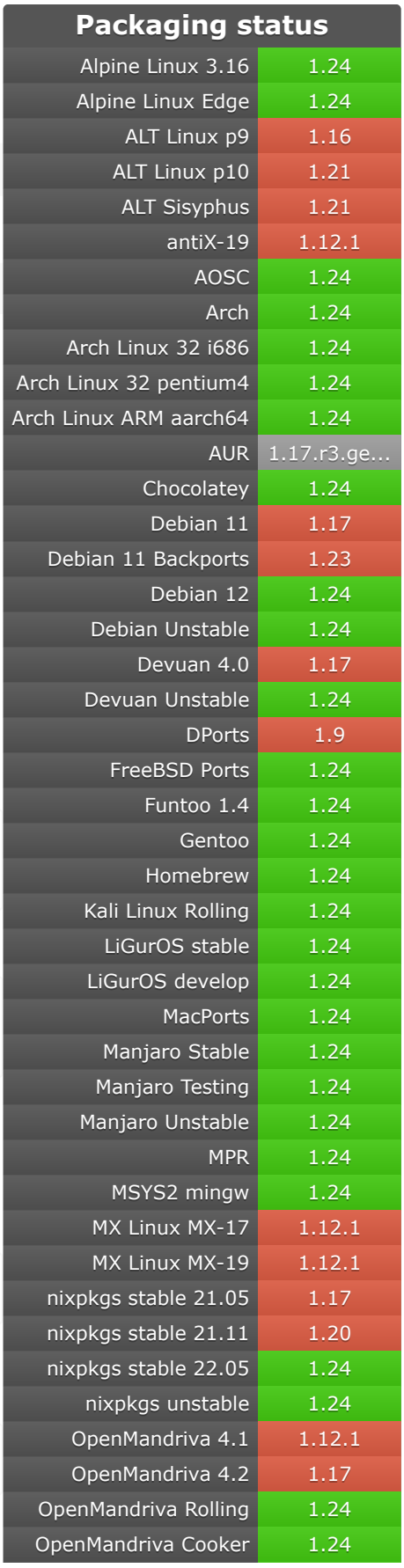
Die Anwendung ist in Homebrew verfügbar. Installation:

```
brew install scrcpy
```

Es wird `adb` benötigt, auf welches über `PATH` zugegriffen werden kann. Falls noch nicht vorhanden:

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

Es ist außerdem in MacPorts vorhanden, welches adb bereits aufsetzt:



```
sudo port install scrcpy
```

Die Anwendung kann zudem manuell gebaut werden.

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

Ausführen

Ein Android-Gerät anschließen und diese Befehle ausführen:

```
scrcpy
```

Dabei werden Kommandozeilenargumente akzeptiert, aufgelistet per:

```
scrcpy --help
```

Funktionalitäten

Aufnahmekonfiguration

Größe reduzieren

Manchmal ist es sinnvoll, das Android-Gerät mit einer geringeren Auflösung zu spiegeln, um die Leistung zu erhöhen.

Um die Höhe und Breite auf einen Wert zu limitieren (z.B. 1024):

```
scrcpy --max-size 1024
scrcpy -m 1024  # short version
```

Die andere Größe wird dabei so berechnet, dass das Seitenverhältnis des Gerätes erhalten bleibt. In diesem Fall wird ein Gerät mit einer 1920×1080-Auflösung mit 1024×576 gespiegelt.

Ändern der Bit-Rate

Die Standard-Bitrate ist 8 Mbps. Um die Bitrate zu ändern (z.B. zu 2 Mbps):

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

Limitieren der Bildwiederholrate

Die Aufnahme-Bildwiederholrate kann begrenzt werden:

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

Dies wird offiziell seit Android 10 unterstützt, kann jedoch bereits auf früheren Versionen funktionieren.

Zuschneiden

Der Geräte-Bildschirm kann zugeschnitten werden, sodass nur ein Teil gespiegelt wird.

Dies ist beispielsweise nützlich, um nur ein Auge der Oculus Go zu spiegeln:

```
scrcpy --crop 1224:1440:0:0   # 1224x1440 am Versatz (0,0)
```

Falls `--max-size` auch festgelegt ist, wird das Ändern der Größe nach dem Zuschneiden angewandt.

Feststellen der Videoorientierung

Um die Orientierung während dem Spiegeln festzustellen:

```
scrcpy --lock-video-orientation     # ursprüngliche (momentane) Orientierung
scrcpy --lock-video-orientation=0   # normale Orientierung
scrcpy --lock-video-orientation=1   # 90° gegen den Uhrzeigersinn
scrcpy --lock-video-orientation=2   # 180°
scrcpy --lock-video-orientation=3   # 90° mit dem Uhrzeigersinn
```

Dies beeinflusst die Aufnahmeausrichtung.

Das Fenster kann auch unabhängig rotiert werden.

Encoder

Manche Geräte besitzen mehr als einen Encoder. Manche dieser Encoder können dabei sogar zu Problemen oder Abstürzen führen. Die Auswahl eines anderen Encoders ist möglich:

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

Um eine Liste aller verfügbaren Encoder zu erhalten (eine Fehlermeldung gibt alle verfügbaren Encoder aus):

```
scrcpy --encoder _
```

Aufnahme

Aufnehmen von Videos

Es ist möglich, das Display während des Spiegelns aufzunehmen:

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

Um das Spiegeln während des Aufnehmens zu deaktivieren:

```
scrcpy --no-display --record file.mp4
scrcpy -Nr file.mkv
# Unterbrechen der Aufnahme mit Strg+C
```

"Übersprungene Bilder" werden aufgenommen, selbst wenn sie in Echtzeit (aufgrund von Performancegründen) nicht dargestellt werden. Die Einzelbilder sind mit *Zeitstempeln* des Gerätes versehen are, sodass eine Paketverzögerungsvariation nicht die Aufnahmedatei beeinträchtigt.

v4l2loopback

Auf Linux ist es möglich, den Video-Stream zu einem v4l2 loopback Gerät zu senden, sodass das Android-Gerät von jedem v4l2-fähigen Tool wie eine Webcam verwendet werden kann.

Das Modul `v4l2loopback` muss dazu installiert werden:

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

Um ein v4l2 Gerät zu erzeugen:

```
sudo modprobe v4l2loopback
```

Dies erzeugt ein neues Video-Gerät in `/dev/videoN`, wobei `N` ein Integer ist (mehr Optionen sind verfügbar um mehrere Geräte oder Geräte mit spezifischen Nummern zu erzeugen).

Um die aktivierten Geräte aufzulisten:

```
# benötigt das v4l-utils package
v4l2-ctl --list-devices

# simpel, kann aber ausreichend
ls /dev/video*
```

Um scrcpy mithilfe eines v4l2 sink zu starten:

```
scrcpy --v4l2-sink=/dev/videoN
scrcpy --v4l2-sink=/dev/videoN --no-display  # Fenster mit Spiegelung ausschalten
scrcpy --v4l2-sink=/dev/videoN -N            # kurze Version
```

(`N` muss mit der Geräte-ID ersetzt werden, welche mit `ls /dev/video*` überprüft werden kann)

Einmal aktiv, kann der Stream mit einem v4l2-fähigen Tool verwendet werden:

```
ffplay -i /dev/videoN
vlc v4l2:///dev/videoN   # VLC kann eine gewisse  Bufferverzögerung herbeiführen
```

Beispielsweise kann das Video mithilfe von OBS aufgenommen werden.

Buffering

Es ist möglich, Buffering hinzuzufügen. Dies erhöht die Latenz, reduziert aber etwaigen Jitter (see #2464).

Diese Option ist sowohl für Video-Buffering:

```
scrcpy --display-buffer=50  # fügt 50ms Buffering zum Display hinzu
```

als auch V4L2 sink verfügbar:

```
scrcpy --v4l2-buffer=500    # fügt 500ms Buffering für v4l2 sink hinzu
```

Verbindung

TCP/IP Kabellos

Scrcpy verwendet `adb`, um mit dem Gerät zu kommunizieren. `adb` kann sich per TCP/IP mit einem Gerät verbinden. Das Gerät muss dabei mit demselben Netzwerk wie der Computer verbunden sein.

Automatisch

Die Option `--tcpip` erlaubt es, die Verbindung automatisch zu konfigurieren. Dabei gibt es zwei Varianten.

Falls das Gerät (verfügbar unter 192.168.1.1 in diesem Beispiel) bereit an einem Port (typically 5555) nach einkommenden adb-Verbindungen hört, dann führe diesen Befehl aus:

```
scrcpy --tcpip=192.168.1.1       # Standard-Port ist 5555
scrcpy --tcpip=192.168.1.1:5555
```

Falls adb TCP/IP auf dem Gerät deaktiviert ist (oder falls die IP-Adresse des Gerätes nicht bekannt ist): Gerät per USB verbinden, anschließend diesen Befehl ausführen:

```
scrcpy --tcpip    # ohne weitere Argumente
```

Dies finden automatisch das Gerät und aktiviert den TCP/IP-Modus. Anschließend verbindet sich der Befehl mit dem Gerät bevor die Verbindung startet.

Manuell

Alternativ kann die TCP/IP-Verbindung auch manuell per `adb` aktiviert werden:

1. Gerät mit demselben Wifi wie den Computer verbinden.
2. IP-Adresse des Gerätes herausfinden, entweder über Einstellungen → Über das Telefon → Status, oder indem dieser Befehl ausgeführt wird:

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

3. Aktivieren von adb über TCP/IP auf dem Gerät: `adb tcpip 5555`.
4. Ausstecken des Geräts.
5. Verbinden zum Gerät: `adb connect DEVICE_IP:5555` *(`DEVICE_IP` ersetzen)*.
6. `scrcpy` wie normal ausführen.

Es kann sinnvoll sein, die Bit-Rate sowie dei Auflösung zu reduzieren:

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

Mehrere Geräte

Falls mehrere Geräte unter `adb devices` aufgelistet werden, muss die *Seriennummer* angegeben werden:

```
scrcpy --serial 0123456789abcdef
scrcpy -s 0123456789abcdef  # kurze Version
```

Falls das Gerät über TCP/IP verbunden ist:

```
scrcpy --serial 192.168.0.1:5555
scrcpy -s 192.168.0.1:5555  # kurze Version
```

Es können mehrere Instanzen von *scrcpy* für mehrere Geräte gestartet werden.

Autostart beim Verbinden eines Gerätes

Hierfür kann AutoAdb verwendet werden:

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

Tunnel

Um sich zu einem entfernten Gerät zu verbinden, kann der `adb` Client mit einem remote-`adb`-Server verbunden werden (Voraussetzung: Gleiche Version des `adb`-Protokolls).

Remote ADB Server

Um sich zu einem Remote-`adb`-Server zu verbinden: Der Server muss auf allen Ports hören

```
adb kill-server
adb -a nodaemon server start
# Diesen Dialog offen halten
```

Warnung: Die gesamte Kommunikation zwischen adb und den Geräten ist unverschlüsselt.

Angenommen, der Server ist unter 192.168.1.2 verfügbar. Dann kann von einer anderen Kommandozeile scrcpy aufgeführt werden:

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

Standardmäßig verwendet scrcpy den lokalen Port für die Einrichtung des `adb forward`-Tunnels (typischerweise `27183`, siehe `--port`). Es ist zudem möglich, einen anderen Tunnel-Port zuzuweisen (sinnvoll in Situationen, bei welchen viele Weiterleitungen erfolgen):

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

SSH Tunnel

Um mit einem Remote-`adb`-Server sicher zu kommunizieren, wird ein SSH-Tunnel empfohlen.

Sicherstellen, dass der Remote-`adb`-Server läuft:

```
adb start-server
```

Erzeugung eines SSH-Tunnels:

```
# local  5038 --> remote  5037
# local 27183 <-- remote 27183
ssh -CN -L5038:localhost:5037 -R27183:localhost:27183 your_remote_computer
# Diesen Dialog geöffnet halten
```

Von einer anderen Kommandozeile aus scrcpy ausführen:

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

Um das Aktivieren von Remote-Weiterleitung zu verhindern, kann eine Vorwärts-Verbindung verwendet werden (`-L` anstatt von `-R`):

```
# local  5038 --> remote  5037
# local 27183 --> remote 27183
ssh -CN -L5038:localhost:5037 -L27183:localhost:27183 your_remote_computer
# Diesen Dialog geöffnet halten
```

Von einer anderen Kommandozeile aus scrcpy ausführen:

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

Wie für kabellose Verbindungen kann es sinnvoll sein, die Qualität zu reduzieren:

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

Fensterkonfiguration

Titel

Standardmäßig ist der Fenstertitel das Gerätemodell. Der Titel kann jedoch geändert werden:

```
scrcpy --window-title 'Mein Gerät'
```

Position und Größe

Die anfängliche Fensterposition und Größe können festgelegt werden:

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

Rahmenlos

Um den Rahmen des Fensters zu deaktivieren:

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

Immer im Vordergrund

Um das Fenster immer im Vordergrund zu halten:

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

Vollbild

Die Anwendung kann direkt im Vollbildmodus gestartet werden:

```
scrcpy --fullscreen
scrcpy -f  # kurze Version
```

Das Vollbild kann dynamisch mit `MOD`+`f` gewechselt werden.

Rotation

Das Fenster kann rotiert werden:

```
scrcpy --rotation 1
```

Mögliche Werte sind:

- `0` : keine Rotation
- `1` : 90 grad gegen den Uhrzeigersinn
- `2` : 180 grad
- `3` : 90 grad mit dem Uhrzeigersinn

die Rotation kann zudem dynamisch mit `MOD`+← *(links)* and `MOD`+→ *(rechts)* angepasst werden.

scrcpy schafft 3 verschiedene Rotationen:

- `MOD`+`r` erfordert von Gerät den Wechsel zwischen Hochformat und Querformat (die momentane App kann dies verweigern, wenn die geforderte Ausrichtung nicht unterstützt wird).
- `--lock-video-orientation` ändert die Ausrichtung der Spiegelung (die Ausrichtung des an den Computer gesendeten Videos). Dies beeinflusst eventuelle Aufnahmen.
- `--rotation` (or `MOD`+←/`MOD`+→) rotiert nur das Fenster, eventuelle Aufnahmen sind hiervon nicht beeinflusst.

Andere Spiegel-Optionen

Lesezugriff

Um die Steuerung (alles, was mit dem Gerät interagieren kann: Tasten, Mausklicks, Drag-and-drop von Dateien) zu deaktivieren:

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

Anzeige

Falls mehrere Displays vorhanden sind, kann das zu spiegelnde Display gewählt werden:

```
scrcpy --display 1
```

Die Liste an verfügbaren Displays kann mit diesem Befehl ausgegeben werden:

```
adb shell dumpsys display   # Nach "mDisplayId=" in der Ausgabe suchen
```

Das zweite Display kann nur gesteuert werden, wenn das Gerät Android 10 oder höher besitzt. Ansonsten wird das Display nur mit Lesezugriff gespiegelt.

Wach bleiben

Um zu verhindern, dass das Gerät nach einer Weile in den Ruhezustand übergeht (solange es eingesteckt ist):

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

Der ursprüngliche Zustand wird beim Schließen von scrcpy wiederhergestellt.

Bildschirm ausschalten

Es ist möglich, beim Starten des Spiegelns mithilfe eines Kommandozeilenarguments den Bildschirm des Gerätes auszuschalten:

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

Oder durch das Drücken von `MOD`+`o` jederzeit.

Um das Display wieder einzuschalten muss `MOD`+`Shift`+`o` gedrückt werden.

Auf Android aktiviert der `POWER` Knopf das Display immer. Für den Komfort wird, wenn `POWER` via scrcpy gesendet wird (über Rechtsklick oder `MOD`+`p`), wird versucht, das Display nach einer kurzen Zeit wieder auszuschalten (falls es möglich ist). Der physische `POWER` Button aktiviert das Display jedoch immer.

Dies kann zudem nützlich sein, um das Gerät vom Ruhezustand abzuhalten:

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

Ausschalten beim Schließen

Um den Gerätebildschirm abzuschalten, wenn scrcpy geschlossen wird:

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

Anzeigen von Berührungen

Für Präsentationen kann es sinnvoll sein, die physischen Berührungen anzuzeigen (auf dem physischen Gerät).

Android stellt dieses Feature in den *Entwickleroptionen* zur Verfügung.

Scrcpy stellt die Option zur Verfügung, dies beim Start zu aktivieren und beim Schließen auf den Ursprungszustand zurückzusetzen:

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

Anmerkung: Nur *physische Berührungen* werden angezeigt (mit dem Finger auf dem Gerät).

Bildschirmschoner deaktivieren

Standardmäßig unterbindet scrcpy nicht den Bildschirmschoner des Computers.

Um den Bildschirmschoner zu unterbinden:

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

Eingabesteuerung

Geräte-Bildschirm drehen

`MOD`+`r` drücken, um zwischen Hoch- und Querformat zu wechseln.

Anmerkung: Dis funktioniert nur, wenn die momentan geöffnete App beide Rotationen unterstützt.

Copy-paste

Immer, wenn sich die Zwischenablage von Android ändert wird dies mit dem Computer synchronisiert.

Jedes `Strg` wird an das Gerät weitergegeben. Insbesonders:

- `Strg`+`c` kopiert typischerweise
- `Strg`+`x` schneidet typischerweise aus
- `Strg`+`v` fügt typischerweise ein (nach der Computer-zu-Gerät-Synchronisation)

Dies funktioniert typischerweise wie erwartet.

Die wirkliche Funktionsweise hängt jedoch von der jeweiligen Anwendung ab. Beispielhaft sendet *Termux* SIGINT bei `Strg`+`c`, und *K-9 Mail* erzeugt eine neue Nachricht.

Um kopieren, ausschneiden und einfügen in diesen Fällen zu verwenden (nur bei Android >= 7 unterstützt):

- `MOD`+`c` gibt `COPY` ein
- `MOD`+`x` gibt `CUT` ein
- `MOD`+`v` gibt `PASTE` ein (nach der Computer-zu-Gerät-Synchronisation)

Zusätzlich erlaubt es `MOD`+`Shift`+`v` den momentanen Inhalt der Zwischenablage als eine Serie von Tastenevents einzugeben. Dies ist nützlich, fall die Applikation kein Einfügen unterstützt (z.B. *Termux*). Jedoch kann nicht-ASCII-Inhalt dabei zerstört werden.

WARNUNG: Das Einfügen der Computer-Zwischenablage in das Gerät (entweder mit `Strg`+`v` oder `MOD`+`v`) kopiert den Inhalt in die Zwischenablage des Gerätes. Als Konsequenz kann somit jede Android-Applikation diesen Inhalt lesen. Das Einfügen von sensiblen Informationen wie Passwörtern sollte aus diesem Grund vermieden werden.

Mache Geräte verhalten sich nicht wie erwartet, wenn die Zwischenablage per Programm verändert wird. Die Option `--legacy-paste` wird bereitgestellt, welche das Verhalten von `Strg`+`v` und `MOD`+`v` so ändert, dass die

Zwischenablage wie bei `MOD+Shift+v` als eine Serie von Tastenevents ausgeführt wird.

Um die automatische Synchronisierung der Zwischenablage zu deaktivieren: `--no-clipboard-autosync`.

Ziehen zum Zoomen

Um "Ziehen-zum-Zoomen" zu simulieren: `Strg`+*klicken-und-bewegen*.

Genauer: `Strg` halten, während Linksklick gehalten wird. Solange Linksklick gehalten wird, skalieren und rotieren die Mausbewegungen den Inhalt (soweit von der jeweiligen App unterstützt).

Konkret erzeugt scrcpy einen am Mittelpunkt des Displays gespiegelten, "virtuellen" Finger.

Simulation einer physischen Tastatur mit HID

Standardmäßig verwendet scrcpy Android-Tasten oder Textinjektion. Dies funktioniert zwar immer, jedoch nur mit ASCII.

Auf Linux kann scrcpy mithilfe von HID eine physische Tastatur simulieren, um eine bessere Eingabeerfahrung zu gewährleisten (dies nutzt USB HID over AOAv2): Die virtuelle Tastatur wird deaktiviert, es funktioniert für alle Zeichen und mit IME.

Dies funktioniert jedoch nur, wenn das Gerät über USB verbunden ist. Zudem wird dies momentan nur unter Linux unterstützt.

Um diesen Modus zu aktivieren:

```
scrcpy --hid-keyboard
scrcpy -K  # kurze Version
```

Falls dies auf gewissen Gründen fehlschlägt (z.B. Gerät ist nicht über USB verbunden), so fällt scrcpy auf den Standardmodus zurück (mit einer Ausgabe in der Konsole). Dies erlaubt es, dieselben Kommandozeilenargumente zu verwenden, egal ob das Gerät per USB oder TCP/IP verbunden ist.

In diesem Modus werden rohe Tastenevents (scancodes) an das Gerät gesendet. Aus diesem Grund muss ein nicht passenden Tastaturformat in den Einstellungen des Android-Gerätes unter Einstellungen → System → Sprache und Eingabe → Physical keyboard konfiguriert werden.

Diese Einstellungsseite kann direkt mit diesem Befehl geöffnet werden:

```
adb shell am start -a android.settings.HARD_KEYBOARD_SETTINGS
```

Diese Option ist jedoch nur verfügbar, wenn eine HID-Tastatur oder eine physische Tastatur verbunden sind.

Simulation einer physischen Maus mit HID

Ähnlich zu einer Tastatur kann auch eine Maus mithilfe von HID simuliert werden. Wie zuvor funktioniert dies jedoch nur, wenn das Gerät über USB verbunden ist. Zudem wird dies momentan nur unter Linux unterstützt.

Standardmäßig verwendet scrcpy Android Maus Injektionen mit absoluten Koordinaten. Durch die Simulation einer physischen Maus erscheint auf dem Display des Geräts ein Mauszeiger, zu welchem die Bewegungen, Klicks und Scrollbewegungen relativ eingegeben werden.

Um diesen Modus zu aktivieren:

```
scrcpy --hid-mouse
scrcpy -M  # kurze Version
```

Es kann zudem `--forward-all-clicks` übergeben werden, um alle Mausklicks an das Gerät weiterzugeben.

Wenn dieser Modus aktiv ist, ist der Mauszeiger des Computers auf dem Fenster gefangen (Zeiger verschwindet von Computer und erscheint auf dem Android-Gerät).

Spezielle Tasteneingaben wie `Alt` oder `Super` ändern den Zustand des Mauszeigers (geben diesen wieder frei/fangen ihn wieder ein). Eine dieser Tasten kann verwendet werden, um die Kontrolle der Maus wieder zurück an den Computer zu geben.

OTG

Es ist möglich, *scrcpy* so auszuführen, dass nur Maus und Tastatur, wie wenn diese direkt über ein OTG-Kabel verbunden wären, simuliert werden.

In diesem Modus ist *adb* nicht nötig, ebenso ist das Spiegeln der Anzeige deaktiviert.

Um den OTG-Modus zu aktivieren:

```
scrcpy --otg
# Seriennummer übergeben, falls mehrere Geräte vorhanden sind
scrcpy --otg -s 0123456789abcdef
```

Es ist möglich, nur HID-Tastatur oder HID-Maus zu aktivieren:

```
scrcpy --otg --hid-keyboard              # nur Tastatur
scrcpy --otg --hid-mouse                 # nur Maus
scrcpy --otg --hid-keyboard --hid-mouse  # Tastatur und Maus
# Der Einfachheit halber sind standardmäßig beide aktiv
scrcpy --otg                             # Tastatur und Maus
```

Wie `--hid-keyboard` und `--hid-mouse` funktioniert dies nur, wenn das Gerät per USB verbunden ist. Zudem wird dies momentan nur unter Linux unterstützt.

Textinjektions-Vorliebe

Beim Tippen von Text werden zwei verschiedene Events generiert:

- *key events*, welche signalisieren, ob eine Taste gedrückt oder losgelassen wurde;
- *text events*, welche signalisieren, dass Text eingegeben wurde.

Standardmäßig werden key events verwendet, da sich bei diesen die Tastatur in Spielen wie erwartet verhält (typischerweise für WASD).

Dies kann jedoch Probleme verursachen. Trifft man auf ein solches Problem, so kann dies mit diesem Befehl umgangen werden:

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

Dies kann jedoch das Tastaturverhalten in Spielen beeinträchtigen/zerstören.

Auf der anderen Seite kann jedoch auch die Nutzung von key events erzwungen werden:

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

Diese Optionen haben jedoch keinen Einfluss auf eine etwaige HID-Tastatur, da in diesem modus alle key events als scancodes gesendet werden.

Wiederholen von Tasten

Standardmäßig löst das gedrückt halten einer Taste das jeweilige Event mehrfach aus. Dies kann jedoch zu Performanceproblemen in manchen Spielen führen.

Um das Weitergeben von sich wiederholenden Tasteneingaben zu verhindern:

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

This option has no effect on HID keyboard (key repeat is handled by Android directly in this mode).

Rechtsklick und Mittelklick

Standardmäßig löst Rechtsklick BACK (wenn Bildschirm aus: POWER) und Mittelklick BACK aus. Um diese Kürzel abzuschalten und stattdessen die Eingaben direkt an das Gerät weiterzugeben:

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

Dateien ablegen

APK installieren

Um eine AKP zu installieren, kann diese per Drag-and-drop auf das *scrcpy*-Fenster gezogen werden.

Dabei erfolgt kein visuelles Feedback, ein Log wird in die Konsole ausgegeben.

Datei auf Gerät schieben

Um eine Datei nach `/sdcard/Download/` auf dem Gerät zu schieben, Drag-and-drop die (nicht-APK)-Datei auf das *scrcpy*-Fenster.

Dabei erfolgt kein visuelles Feedback, ein Log wird in die Konsole ausgegeben.

Das Zielverzeichnis kann beim Start geändert werden:

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

Audioweitergabe

Audio wird von *scrcpy* nicht übertragen. Hierfür kann sndcpy verwendet werden.

Siehe zudem issue #14.

Tastenkürzel

In der folgenden Liste ist `MOD` der Kürzel-Auslöser. Standardmäßig ist dies (links) `Alt` oder (links) `Super`.

Dies kann mithilfe von `--shortcut-mod` geändert werden. Mögliche Tasten sind `lstrg`, `rstrg`, `lalt`, `ralt`, `lsuper` und `rsuper`. Beispielhaft:

```
# Nutze rStrg als Auslöser
scrcpy --shortcut-mod=rctrl

# Nutze entweder LStrg+LAlt oder LSuper für Tastenkürzel
scrcpy --shortcut-mod=lctrl+lalt,lsuper
```

`Super` ist typischerweise die `Windows` oder `Cmd` Taste.

Aktion	Tastenkürzel	
Vollbild wechseln	`MOD`+`f`	
Display nach links rotieren	`MOD`+← *(links)*	
Display nach links rotieren	`MOD`+→ *(rechts)*	
Fenstergröße 1:1 replizieren (pixel-perfect)	`MOD`+`g`	
Fenstergröße zum entfernen der schwarzen Balken ändern	`MOD`+`w`	*Doppel-Linksklick*[1]
Klick auf `HOME`	`MOD`+`h`	*Mittelklick*
Klick auf `BACK`	`MOD`+`b`	*Rechtsklick*[2]
Klick auf `APP_SWITCH`	`MOD`+`s`	*4.-Taste-Klick*[3]
Klick auf `MENU` (Bildschirm entsperren)[4]	`MOD`+`m`	
Klick auf `VOLUME_UP`	`MOD`+↑ *(hoch)*	
CKlick auf `VOLUME_DOWN`	`MOD`+↓ *(runter)*	
Klick auf `POWER`	`MOD`+`p`	
Power an	*Rechtsklick*[2]	
Gerätebildschirm ausschalten (weiterhin spiegeln)	`MOD`+`o`	
Gerätebildschirm einschalten	`MOD`+`Shift`+`o`	
Gerätebildschirm drehen	`MOD`+`r`	
Benachrichtigungs-Bereich anzeigen	`MOD`+`n`	*5.-Taste-Klick*[3]
Erweitertes Einstellungs-Menü anzeigen	`MOD`+`n`+`n`	*Doppel-5.-Taste-Klick*[3]
Bedienfelder einklappen	`MOD`+`Shift`+`n`	
In die Zwischenablage kopieren[5]	`MOD`+`c`	
In die Zwischenablage kopieren[5]	`MOD`+`x`	
Zwischenablage synchronisieren und einfügen[5]	`MOD`+`v`	

Computer-Zwischenablage einfügen (per Tastenevents)	`MOD`+`Shift`+`v`	
FPS-Zähler aktivieren/deaktivieren (ing stdout)	`MOD`+`i`	
Ziehen zum Zoomen	`Strg`+*Klicken-und-Bewegen*	
Drag-and-drop mit APK-Datei	APK von Computer installieren	
Drag-and-drop mit Nicht-APK Datei	Datei auf das Gerät schieben	

[1]*Doppelklick auf die schwarzen Balken, um diese zu entfernen.*
[2]*Rechtsklick aktiviert den Bildschirm, falls dieser aus war, ansonsten ZURÜCK.*
[3]*4. und 5. Maustasten, wenn diese an der jeweiligen Maus vorhanden sind.*
[4]*Für react-native Applikationen in Entwicklung,* `MENU` *öffnet das Entwickler-Menü.*
[5]*Nur für Android >= 7.*

Abkürzungen mit mehreren Tastenanschlägen werden durch das Loslassen und erneute Drücken der Taste erreicht. Beispielhaft, um "Erweitere das Einstellungs-Menü" auszuführen:

1. Drücke und halte `MOD`.
2. Doppelklicke `n`.
3. Lasse `MOD` los.

Alle `Strg`+*Taste* Tastenkürzel werden an das Gerät übergeben, sodass sie von der jeweiligen Applikation ausgeführt werden können.

Personalisierte Pfade

Um eine spezifische *adb* Binary zu verwenden, muss deren Pfad als Umgebungsvariable `ADB` deklariert werden:

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

Um den Pfad der `scrcpy-server` Datei zu bearbeiten, muss deren Pfad in `SCRCPY_SERVER_PATH` bearbeitet werden.

Um das Icon von scrcpy zu ändern, muss `SCRCPY_ICON_PATH` geändert werden.

Warum *scrcpy*?

Ein Kollege hat mich dazu herausgefordert, einen Namen so unaussprechbar wie gnirehtet zu finden.

`strcpy` kopiert einen **str**ing; `scrcpy` kopiert einen **scr**een.

Selbst bauen?

Siehe BUILD.

Typische Fehler

Siehe FAQ.

Entwickler

Entwicklerseite.

Licence

Artikel (auf Englisch)

- Introducing scrcpy
- Scrcpy now works wirelessly