

## Содержание статьи

- [Основы работы с ADB](#)
- [Установка программ](#)
- [Бэкап приложений](#)
- [Консоль в консоли](#)
- [Создание скриншота](#)
- [Запись видео, происходящего на экране устройства](#)
- [Управление приложениями](#)
- [Системные утилиты](#)
- [Снятие логов](#)
- [Продвинутый уровень](#)
- [Снятие графического ключа, PIN, facelock](#)
- [Выводы](#)

Существует множество инструментов для работы с подключенным с помощью USB-кабеля или Wi-Fi смартфоном. Особо развитые инструменты позволяют перемещать файлы, устанавливать и удалять софт, просматривать контакты, делать скриншоты экрана и даже отправлять СМС, однако ни один графический инструмент не сравнится с мощностью, которую может дать консоль Android. В этой статье мы поговорим об ADB (Android Debug Bridge) — стандартном инструменте для отладки и работы с консолью Android с компа.

Описанные в статье команды можно выполнять непосредственно на устройстве, скачав из маркета [эмулятор терминала](#), но удобнее это делать, конечно же, с компа через adb.

## Основы работы с ADB

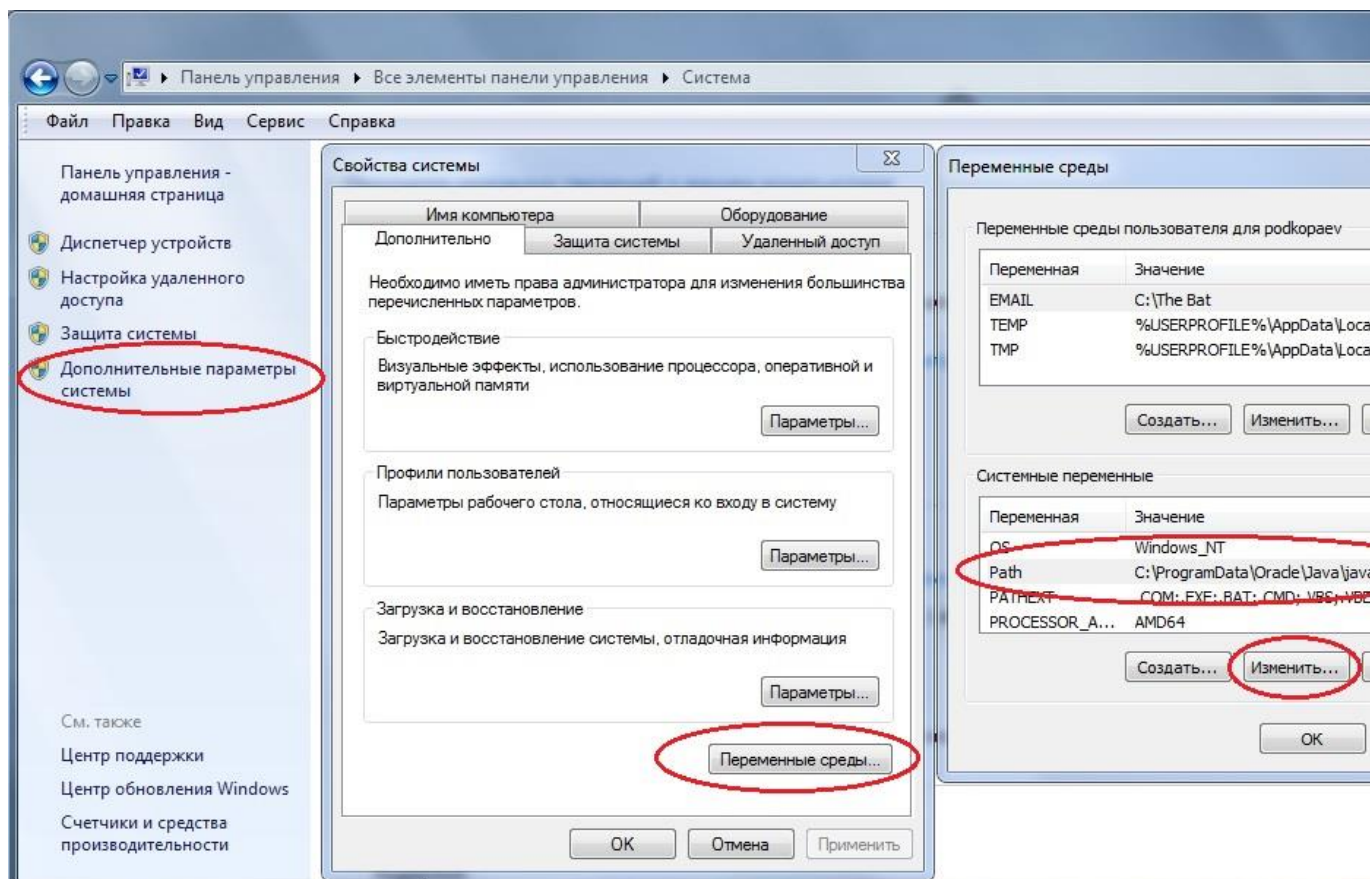
Для начала работы с ADB его следует активировать на устройстве и установить утилиту adb и драйверы на комп. Первая задача выполняется с помощью включения «Отладки по USB» в пункте настроек «Для разработчиков» (если этот пункт скрыт, нажми семь раз на номер сборки в меню «О телефоне»).

Для установки ADB на комп качаем [Adb Kit](#) и распаковываем в любую папку (рекомендую использовать названия папок без русских символов). Также скачиваем и устанавливаем [драйверы ADB](#).

Работать с adb нужно из командной строки. Нажимаем Win + R и вводим cmd, далее переходим в папку, в которой лежит adb. Для моей папки команда будет следующей:

```
cd \android
```

Чтобы не проделывать все эти манипуляции каждый раз, можно добавить нужную папку в переменную Path. Для этого необходимо зайти в «Панель управления -> Система -> Дополнительные параметры системы -> Переменные среды», найти переменную Path и добавить в конец строки, через точку с запятой, путь до папки с adb. Теперь после запуска консоли можно сразу вводить необходимые команды.



## Добавление adb в переменную Path

Проверим наше подключение к телефону с помощью следующей команды (она должна вывести список подключенных устройств):

adb devices

С ADB можно работать через Wi-Fi. Для этого нужны права root и приложение [WiFi ADB](#). Запускаем приложение, жмем переключатель и подключаемся к смартфону с помощью команды connect и показанного приложением IP-адреса:

```
adb connect IP-адрес
```

Далее работа с ADB ничем не отличается.

Скопировать вывод консоли после выделения мышкой, а также вставить скопированную команду или имя файла в консоль можно правой кнопкой мыши. Включается в свойствах консоли.

## Установка программ

ADB можно использовать для установки приложений без необходимости копировать их на смартфон. Достаточно выполнить такую команду:

```
adb install d:/downloads/имя_файла.apk
```

В команду также можно добавить дополнительные ключи. Полезными будут `-e` — переустановить приложение с сохранением данных и `-d` — установить версию меньше текущей.

Программы можно и удалять, но для этого нужно знать название пакета (как узнать, расскажу чуть позже). На примере игры Angry Birds Seasons команда будет выглядеть так:

```
adb uninstall com.rovio.angrybirdsseasons
```

## Бэкап приложений

В Android есть встроенные функции бэкапа, которые также можно запустить с помощью командной строки. Для этого используется команда `adb backup` и набор опций:

```
adb backup [опции] <приложения>
```

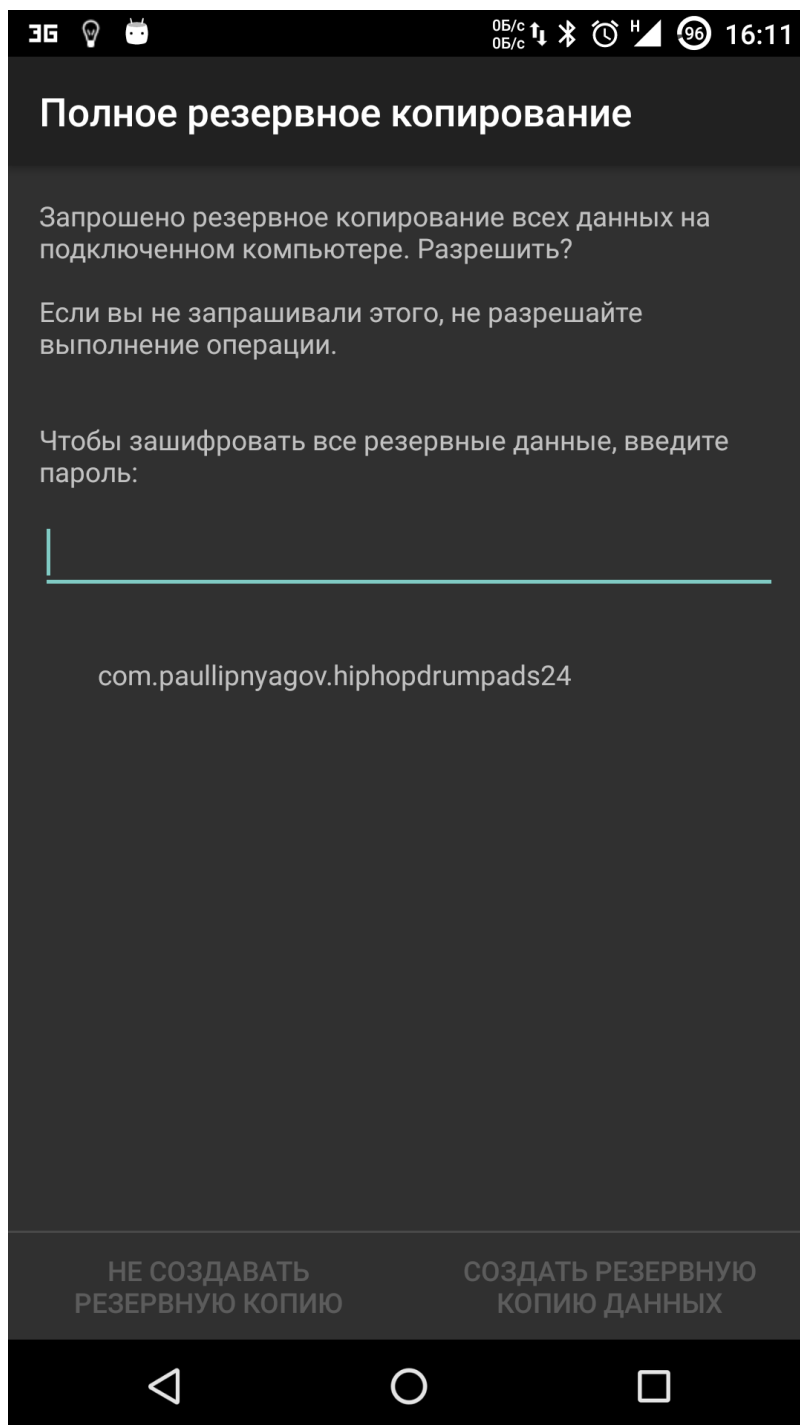
- *-f* указывает имя создаваемого файла и его расположение на компе. При отсутствии ключа будет создан файл backup.ab в текущем каталоге;
- *-apk/-noapk* указывает, включать ли в бэкап только данные приложения или сам .apk тоже (по умолчанию не включает);
- *-obb/-noobb* указывает, включать ли в бэкап расширения .obb для приложений (по умолчанию не включает);
- *-shared/-nosystem* указывает, включать ли в бэкап содержимое приложения на SD-карте (по умолчанию не включает);
- *-all* указывает на необходимость бэкапа всех установленных приложений;
- *-system/-nosystem* указывает, включать ли в бэкап системные приложения (по умолчанию включает);
- — перечень пакетов для бэкапа.

Если мы хотим создать бэкап всех несистемных прог, включая сами .apk, в определенное место, то команда будет выглядеть так:

```
adb backup -f c:\android\backup.ab -apk -all -nosystem
```

После ввода необходимо подтвердить начало выполнения бэкапа на самом устройстве. Для восстановления полученного бэкапа нужно выполнить соответствующую команду:

```
adb restore c:\android\backup.ab
```



Процесс бэкапа

## Консоль в консоли

Наряду с упомянутой консолью, которая является DOS-консолью под Windows, в Android существует и своя. Она вызывается через *adb shell* и представляет собой по сути стандартную Linux-консоль, но с неполным набором команд, расширить который можно, установив из маркета BusyBox. Использовать эту консоль можно двумя способами. В интерактивном режиме она запускается командой

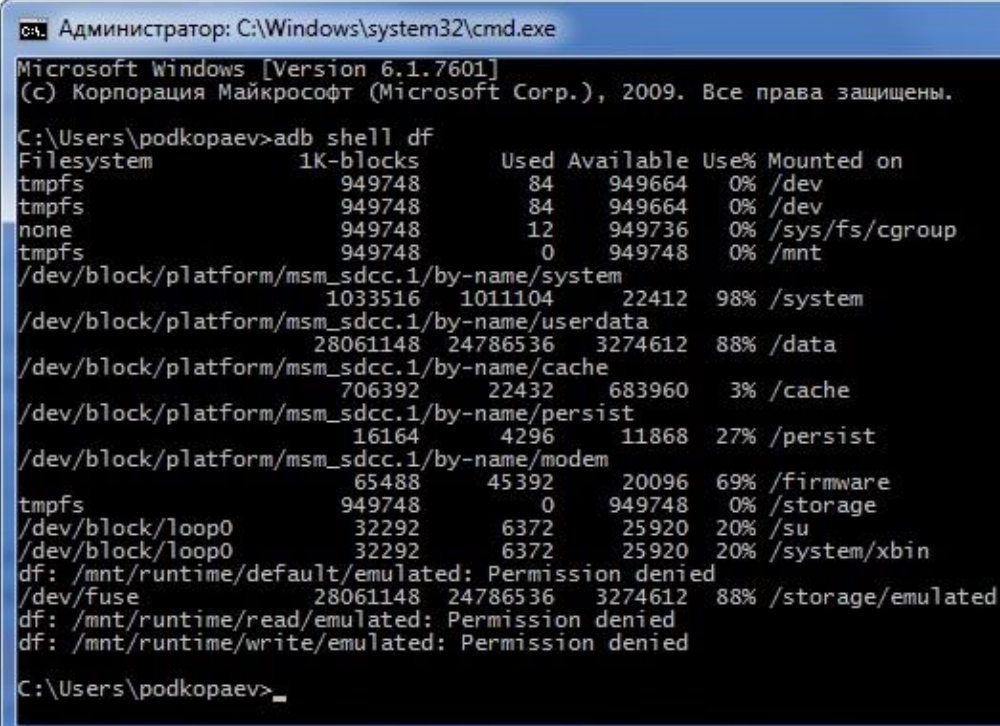
```
adb shell
```

В консоли появляется знак \$ (далее по тексту этот знак будет означать необходимость ввода предварительной команды `adb shell`), и после этого можно вводить серию команд, получая после каждой отклик. Второй способ — если необходимо ввести только одну команду, можно писать ее подряд за `adb shell`.

В шелле работают стандартные команды для копирования, перемещения и удаления файлов: *cp*, *mv* и *rm*. Можно менять каталоги (*cd*) и смотреть их содержимое (*ls*). Кроме стандартных Linux-команд, о которых можно узнать из любого справочника, в Android есть несколько своих специализированных инструментов, но, чтобы использовать некоторые из них, придется получить на смартфоне права root, а после запуска консоли выполнять команду `su`:

```
adb shell
su
```

Это нужно делать, если в ответ на какую-либо команду ты видишь строку, похожую на «access denied» или «are you root?». В случае успеха знак \$ сменится на #.



```
Администратор: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\podkopaev>adb shell df
Filesystem            1K-blocks      Used Available Use% Mounted on
tmpfs                  949748         84   949664    0% /dev
tmpfs                  949748         84   949664    0% /dev
none                  949748         12   949736    0% /sys/fs/cgroup
tmpfs                  949748          0   949748    0% /mnt
/dev/block/platform/msm_sdcc.1/by-name/system
1033516 1011104 22412    98% /system
/dev/block/platform/msm_sdcc.1/by-name/userdata
28061148 24786536 3274612    88% /data
/dev/block/platform/msm_sdcc.1/by-name/cache
706392 22432 683960    3% /cache
/dev/block/platform/msm_sdcc.1/by-name/persist
16164 4296 11868   27% /persist
/dev/block/platform/msm_sdcc.1/by-name/modem
65488 45392 20096   69% /firmware
tmpfs                  949748          0   949748    0% /storage
/dev/block/loop0       32292    6372 25920   20% /su
/dev/block/loop0       32292    6372 25920   20% /system/xbin
df: /mnt/runtime/default/emulated: Permission denied
/dev/fuse              28061148 24786536 3274612    88% /storage/emulated
df: /mnt/runtime/read/emulated: Permission denied
df: /mnt/runtime/write/emulated: Permission denied

C:\Users\podkopaev>_
```

Вывод свободного пространства на устройстве командой `adb shell df`



```
Администратор: C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\podkopaev>adb shell
shell@flo:/ $ ls -l /dev/block/platform/msm_sdcc.1/by-name/
/dev/block/platform/msm_sdcc.1/by-name/: Permission denied
1|shell@flo:/ $ su
root@flo:/ # ls -l /dev/block/platform/msm_sdcc.1/by-name/
lrwxrwxrwx root      root      2016-04-20 09:25  DDR -> /dev/block/mmcblk0p26
lrwxrwxrwx root      root      2016-04-20 09:25  aboot -> /dev/block/mmcblk0p12
lrwxrwxrwx root      root      2016-04-20 09:25  abootb -> /dev/block/mmcblk0p19
lrwxrwxrwx root      root      2016-04-20 09:25  boot -> /dev/block/mmcblk0p14
lrwxrwxrwx root      root      2016-04-20 09:25  cache -> /dev/block/mmcblk0p23
lrwxrwxrwx root      root      2016-04-20 09:25  fsg -> /dev/block/mmcblk0p8
lrwxrwxrwx root      root      2016-04-20 09:25  m9kefs1 -> /dev/block/mmcblk0p5
lrwxrwxrwx root      root      2016-04-20 09:25  m9kefs2 -> /dev/block/mmcblk0p6
lrwxrwxrwx root      root      2016-04-20 09:25  m9kefs3 -> /dev/block/mmcblk0p7
lrwxrwxrwx root      root      2016-04-20 09:25  m9kefsc -> /dev/block/mmcblk0p28
lrwxrwxrwx root      root      2016-04-20 09:25  metadata -> /dev/block/mmcblk0p29
lrwxrwxrwx root      root      2016-04-20 09:25  misc -> /dev/block/mmcblk0p24
lrwxrwxrwx root      root      2016-04-20 09:25  modemst1 -> /dev/block/mmcblk0p2
lrwxrwxrwx root      root      2016-04-20 09:25  modemst2 -> /dev/block/mmcblk0p3
lrwxrwxrwx root      root      2016-04-20 09:25  pad -> /dev/block/mmcblk0p16
lrwxrwxrwx root      root      2016-04-20 09:25  persist -> /dev/block/mmcblk0p4
lrwxrwxrwx root      root      2016-04-20 09:25  radio -> /dev/block/mmcblk0p1
lrwxrwxrwx root      root      2016-04-20 09:25  recovery -> /dev/block/mmcblk0p25
lrwxrwxrwx root      root      2016-04-20 09:25  rpm -> /dev/block/mmcblk0p13
lrwxrwxrwx root      root      2016-04-20 09:25  rpmb -> /dev/block/mmcblk0p20
lrwxrwxrwx root      root      2016-04-20 09:25  sb11 -> /dev/block/mmcblk0p9
lrwxrwxrwx root      root      2016-04-20 09:25  sb12 -> /dev/block/mmcblk0p10
lrwxrwxrwx root      root      2016-04-20 09:25  sb12b -> /dev/block/mmcblk0p17
lrwxrwxrwx root      root      2016-04-20 09:25  sb13 -> /dev/block/mmcblk0p11
lrwxrwxrwx root      root      2016-04-20 09:25  sb13b -> /dev/block/mmcblk0p18
lrwxrwxrwx root      root      2016-04-20 09:25  ssd -> /dev/block/mmcblk0p27
lrwxrwxrwx root      root      2016-04-20 09:25  system -> /dev/block/mmcblk0p22
lrwxrwxrwx root      root      2016-04-20 09:25  tz -> /dev/block/mmcblk0p15
lrwxrwxrwx root      root      2016-04-20 09:25  tzb -> /dev/block/mmcblk0p21
lrwxrwxrwx root      root      2016-04-20 09:25  userdata -> /dev/block/mmcblk0p30
root@flo:/ # _
```

Пример работы команды `ls` для вывода информации о разделах

## Создание скриншота

Выполняется одной строчкой:

```
adb shell screencap /sdcard/screen.png
```

После этого картинку нужно выдернуть из устройства командой `adb pull`:

```
adb pull /sdcard/screen.png
```

В recovery скриншот можно сделать следующей командой:

```
adb pull /dev/graphics/fb0
```

Затем необходимо преобразовать файл `fb0` в нормальное изображение с помощью `FFmpeg`, который нужно скачать и положить в папку с `adb`.

Расширение необходимо ставить своего устройства:

```
ffmpeg -f rawvideo -pix_fmt rgb32 -s 1080x1920 -i fb0 fb0.png
```

## Запись видео, происходящего на экране устройства

```
adb shell screenrecord --size 1280x720 --bit-rate 6000000 --time-limit 10 --verbose /sdcard/video.mp4
```

Данная команда начнет записывать видео с разрешением 1280 x 720 (если не указать, то будет использовано нативное разрешение экрана устройства), с битрейтом 6 Мбит/с, длиной 20 с (если не указать, то будет выставлено максимальное значение 180 с), с показом логов в консоли. Записанное видео будет находиться в /sdcard (файл video.mp4).

Все запущенные из консоли и в *adb shell* процессы, занимающие некоторое время для выполнения, можно прервать с помощью комбинации Ctrl + C. Выйти из шелла и вернуться к выполнению обычных команд adb — Ctrl + D.

## Управление приложениями

Для управления приложениями используются две команды: **pm** (package manager) — менеджер пакетов и **am** (activity manager) — менеджер активностей. У данных команд есть немало ключей, которые можно посмотреть на [портале разработчиков](#). Остановимся на некоторых. Для начала получим список установленных на устройстве приложений в виде названий пакетов, которые пригодятся позже:

```
$ pm list packages
```

Добавив в конец **-s**, ты увидишь только системные приложения, **-3** — только сторонние, **-f** покажет пути установки пакетов, а **-d** — отключенные приложения. Далее, зная названия пакетов, можно совершать над ними различные насильственные действия :). Например, отключить ненужный календарь:

```
$ pm disable com.google.android.calendar
```

Очистить данные:



```
$ pm clear com.dropbox.android
```

Ну а совсем удалить можно так:

```
$ pm uninstall com.dropbox.android
```

Для использования activity manager понадобятся более глубокие знания структуры Android и понимание того, что такое [Activity](#) и [Intent](#). Это позволит тебе запускать различные приложения, например браузер или настройки:

```
$ am start -n com.android.browser/.BrowserActivity
$ am start -n com.android.settings/.Settings
```

Завершить работу приложения можно противоположной командой:

```
$ am kill com.android.browser
```

Ну а убить все запущенные приложения — такой командой:

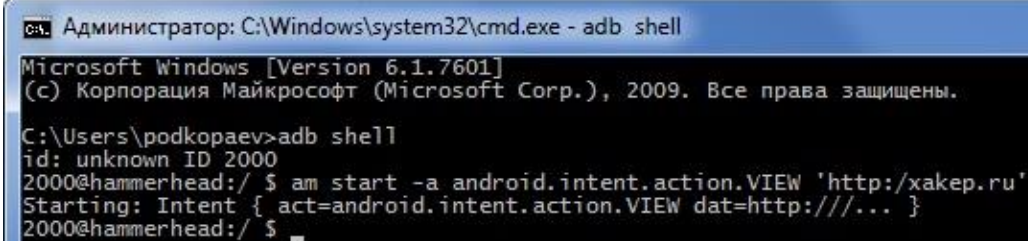
```
$ am kill-all
```

Тот же activity manager поможет сделать звонок на нужный номер телефона:

```
$ am start -a android.intent.action.CALL tel:123
```

А так можно открыть страницу в браузере:

```
$ am start -a android.intent.action.VIEW 'http://xakep.ru'
```



```
Администратор: C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\podkopaev>adb shell
id: unknown ID 2000
2000@hammerhead:/ $ am start -a android.intent.action.VIEW 'http://xakep.ru'
Starting: Intent { act=android.intent.action.VIEW dat=http://... }
2000@hammerhead:/ $ _
```

Запуск браузера из консоли

А с помощью вариации предыдущей команды можно отправить СМС:

```
$ am start -a android.intent.action.SENDTO -d sms:HOME_ТЕЛЕФОНА --
es sms_body "ТЕКСТ_СМС" --ez exit_on_sent true
```

```
$ input keyevent 22
$ input keyevent 66
```

В данной команде *input keyevent* эмулирует нажатие кнопок и может использоваться как для хардварных, так и для кнопок внутри приложения. В нашем примере 22 соответствует перевод фокуса вправо (джойстик вправо — dpad right), а 66 — Enter.

С помощью команды *input* можно, например, разблокировать телефон. Для этого необходимо ввести:

```
$ input keyevent 82
```

Погасит экран *keyevent 26*, что соответствует нажатию кнопки Power. Можно также поэкспериментировать с цифрами 3 — Home, 4 — Back, 24 — Volume Up, 25 — Volume Down, 27 — физическая кнопка Camera. Последнюю кнопку можно передать и через широковещательные сообщения (полный список широковещательных сообщений ты найдешь [тут](#)):

```
$ am broadcast -a android.intent.action.CAMERA_BUTTON
```

Другое широковещательное сообщение переведет телефон в режим самолета:

```
$ am broadcast -a android.intent.action.AIRPLANE_MODE --ez state true
```

Но данная команда не сработает на последних версиях Android. Для управления питанием и беспроводными коммуникациями там используется утилита *svc*. Например, включить передачу данных через мобильную сеть или управлять Wi-Fi можно через команды

```
$ svc data enable
$ svc wifi disable
```

Также можно заставить смартфон оставаться включенным при подключении к USB-порту/зарядке/Wi-Fi-сети или всегда:

```
$ svc power stayon usb
$ svc power stayon ac
$ svc power stayon wireless
$ svc power stayon true
```

Возвращаясь к команде *input*, стоит выделить еще одну команду для вставки текста в текущее поле. Кому-то это может показаться более

привлекательным способом набора текста с компа, чем нажимать на кнопки небольшой области экрана. Выглядит команда так:

```
$ input text "Текст для вставки"
```

Кроме опции `text`, у команды `input` есть и другие. Полная форма команды такова:

```
$ input [<source>] <command> [<arg>...]
```

В качестве источника можно указывать `trackball`, `joystick`, `touchnavigation`, `mouse`, `keyboard`, `gamepad`, `touchpad`, `dpad`, `stylus`, `touchscreen`. В качестве команд будут:

- `text` (Default: `touchscreen`) [`delay`]
- `keyevent` [`–longpress`] ... (Default: `keyboard`)
- `tap` (Default: `touchscreen`)
- `swipe` [`duration(ms)`] (Default: `touchscreen`)
- `press` (Default: `trackball`)
- `roll` (Default: `trackball`)

Как видно из команд, можно, хотя и с трудом, управлять устройством через команды `input touch` и `input swipe` при разбитом экране, если не поддерживается работа мышки через USB-OTG. Например, вытянуть шторку с уведомлениями получится так (отсчет координат идет от левого верхнего угла):

```
$ input swipe 10 10 10 1000
```

А так можно узнать разрешение экрана:

```
$ dumpsys window | \sed -n '/mUnrestrictedScreen/ s/^(.*) \([0-9][0-9]*\)x\([0-9][0-9]*\)\/\1 \2/p'
```

Для Nexus 5 разрешение выдаст 1080 x 1920. Тогда нажать на кнопку «Меню приложений» стандартного лаунчера от Google, которая находится над кнопкой «Домой», можно так:

```
$ input touchscreen tap 500 1775
```

## Скрипты

Выполнение всех описываемых в статье серий команд можно автоматизировать. Для этого вставляем их в текстовый файл (строки, следующие за `adb shell`), который имеет в начале строку `#!/system/bin/sh`,

сохраняем с расширением sh и закидываем на устройство. После этого можно запускать скрипт через тот же adb:

```
adb shell sh /sdcard/имя_файла.sh
```

## Системные утилиты

Кратко остановлюсь на нескольких полезных командах (работоспособность некоторых, однако, может зависеть от версии прошивки и модели телефона).

**Изменение DPI.** Не требует root и работает на Android 5.0+. Стандартное значение для Nexus 5 — 480. При значении 420 на рабочем столе стокового лаунчера помещается пять иконок в ряд вместо четырех:

```
$ wm density 420 && adb reboot
```

**Подключение /system в режиме записи.** Для части команд, которые меняют системные файлы, необходимо сначала перемонтировать раздел /system на запись. Это необходимо в том числе при удалении системных приложений. Перемонтирование выполняется следующей командой:

```
$ su  
# mount -o rw,remount /system
```

**Мягкая перезагрузка:**

```
$ setprop ctl.restart zygote
```

**Перевод смартфона в режим энергосбережения Doze (Android M+):**

```
$ dumpsys battery unplug  
$ dumpsys deviceidle step
```

...повторяем действия, пока не увидим idle.

**Батарейка в процентах (Android 4.4+):**

```
$ content insert --uri content://settings/system --bind  
name:s:status_bar_show_battery_percent --bind value:i:1
```

## Снятие логов

Очень часто, когда для решения проблемы пользователь обращается на форум устройства, там его просят скинуть логи работы телефона или приложения. Отвечают за это две утилиты: **logcat** и **dmesg**. Первая позволяет увидеть системные сообщения в реальном времени, а вторая постфактум покажет работу ядра, включая сообщения ошибок ввода-вывода, загрузку драйверов, подключение USB-устройств и так далее. Полный лог можно вывести сразу в файл следующей командой:

```
adb logcat > logcat.txt
```

Все события будут записываться непрерывно по мере работы устройства. Остановить запись можно стандартной комбинацией Ctrl + C. Однако в лог попадает вся информация, что сильно затрудняет поиск нужной. Поэтому для работы обычно используют [набор ключей и фильтров](#), подходящих к конкретной ситуации. Существует семь приоритетов сообщений по мере возрастания: V — Verbose, D — Debug, I — Info, W — Warning, E — Error, F — Fatal, S — Silent. Например, для вывода всех сообщений с приоритетом *E* и выше следует ввести:

```
adb logcat *:E
```

После этого можно запускать проблемное приложение и смотреть, что именно вызывает ошибку. Также поддерживается вывод информации из альтернативных буферов. Этим способом можно посмотреть, что приложения делают в фоне и, например, какие события происходят после включения экрана:

```
adb logcat -b events
```



```
Администратор: C:\Windows\system32\cmd.exe - adb shell
l vibrate=null sound=null defaults=0x0 flags=0x6a color=0x00000000 category=service actions=1 vis=PR
04-20 13:08:24.405 837 1893 I notification_expansion: [0|com.painless.pc|1|null|10098,0,0,8193017
04-20 13:08:24.435 1011 1011 I sysui_status_bar_state: [1,1,0,0,1]
04-20 13:08:24.486 215 215 I sf_frame_dur: [NavigationBar,37,1,1,0,0,0,1]
04-20 13:08:24.649 837 837 I power_screen_broadcast_done: [0,341,1]
04-20 13:08:26.073 837 1893 I am_kill : [0,12984,com.google.android.deskclock,15,empty #17]
04-20 13:08:26.090 837 1343 I am_proc_died: [0,12984,com.google.android.deskclock]
04-20 13:08:33.631 837 857 I am_pss : [1510,10027,com.google.android.googlequicksearchbox,12583
04-20 13:08:33.663 837 857 I am_pss : [30725,10055,com.google.android.talk,92730368,89772032]
04-20 13:08:33.689 837 857 I am_pss : [24277,10111,com.bitcubate.root.busybox.complete,24543232
04-20 13:08:33.720 837 857 I am_pss : [17657,10140,com.vkontakte.android,59897856,58392576]
04-20 13:08:33.741 837 857 I am_pss : [14610,10000,eu.chainfire.supersu,4213760,3452928]
04-20 13:08:33.763 837 857 I am_pss : [14591,10046,com.google.android.apps.fitness,6105088,5357
04-20 13:08:33.787 837 857 I am_pss : [13815,10156,ru.yandex.metro,9203712,6213632]
04-20 13:08:52.163 2492 12877 I c2dm : [0,NULL,7,6]
04-20 13:08:52.286 837 857 I am_pss : [2492,10011,com.google.android.gms.persistent,47450112,45
04-20 13:08:52.318 837 857 I am_pss : [13143,10071,com.google.android.gm,57329664,53002240]
04-20 13:08:52.324 2492 12883 I c2dm : [1,NULL,7,6]
04-20 13:08:52.348 837 857 I am_pss : [16061,10027,com.google.android.googlequicksearchbox:sear
04-20 13:08:52.369 837 857 I am_pss : [14577,10003,android.process.acore,8404992,7462912]
04-20 13:08:52.389 837 857 I am_pss : [12358,10011,com.google.process.gapps,4913152,4046848]
04-20 13:08:57.112 2492 12877 I c2dm : [-8,com.google.android.gsf.gtalkservice:,8,7]
04-20 13:08:57.161 837 857 I am_pss : [14697,10034,com.google.android.calendar,12292096,1138278
04-20 13:08:57.180 837 857 I am_pss : [14732,10002,com.android.providers.calendar,4623360,37806
04-20 13:09:20.325 837 857 I am_pss : [1574,10011,com.google.android.gms,43096064,40255488]
04-20 13:09:20.357 837 857 I am_pss : [21560,10067,com.google.android.apps.plus,34450432,337305
04-20 13:09:20.360 837 857 I sync : [com.google.android.gm.email.provider,0,1,-1096739447]
04-20 13:09:20.509 837 857 I sync : [com.google.android.gm.email.provider,1,1,-1096739447]
04-20 13:09:32.820 837 957 I sysui_histogram: [power_double_tap_interval,69531]
04-20 13:09:32.821 837 837 I power_screen_state: [1,0,0,0]
04-20 13:09:32.821 837 837 I screen_toggled: 1
04-20 13:09:32.821 837 837 I power_screen_broadcast_send: 1
04-20 13:09:32.822 1011 1011 I sysui_histogram: [note_load,7]
04-20 13:09:32.822 837 848 I notification_panel_revealed: 7
04-20 13:09:32.825 837 1134 I notification_visibility: [0|com.google.android.apps.inbox|0|john.br
04-20 13:09:32.825 837 1134 I notification_visibility: [-1|android|17040353|null|1000,1,529737,52
04-20 13:09:32.825 837 1134 I notification_visibility: [0|net.dinglish.android.taskerm|214748364
04-20 13:09:32.825 837 1134 I notification_visibility: [0|com.painless.pc|1|null|10098,1,8198597
04-20 13:09:32.857 837 837 I notification_enqueue: [10106,1326,net.dinglish.android.taskerm,214
l vibrate=null sound=null defaults=0x0 flags=0x6a color=0x00000000 category=service actions=1 vis=PR
04-20 13:09:32.923 837 857 I am_pss : [10923,10101,ru.megafon.mlk,10547200,9510912]
04-20 13:09:33.042 837 837 I power_screen_broadcast_done: [1,221,1]
04-20 13:09:33.129 215 215 I sf_frame_dur: [ColorFade,27,1,0,0,0,0]
04-20 13:09:36.251 1011 1011 I sysui_lockscreen_gesture: [1,228,4078]
04-20 13:09:36.356 837 1691 I notification_visibility: [0|com.google.android.apps.inbox|0|john.br
04-20 13:09:36.357 837 1691 I notification_visibility: [-1|android|17040353|null|1000,0,533269,53
04-20 13:09:36.357 837 1691 I notification_visibility: [0|net.dinglish.android.taskerm|214748364
04-20 13:09:36.357 837 1691 I notification_visibility: [0|com.painless.pc|1|null|10098,0,82002129
04-20 13:09:36.357 837 1131 I notification_panel_hidden: []
04-20 13:09:36.359 1011 1011 I screen_toggled: 2
04-20 13:09:36.365 837 1618 I am_resume_activity: [0,144493372,3428,com.google.android.googlequic
04-20 13:09:36.441 837 1343 I notification_expansion: [0|com.painless.pc|1|null|10098,0,1,8200221
04-20 13:09:36.448 1011 1011 I sysui_status_bar_state: [0,0,0,0,1]
04-20 13:09:36.462 1510 1510 I am_on_resume_called: [0,com.google.android.launcher.GEL]
04-20 13:09:36.465 2492 12877 I c2dm : [-8,com.google.android.gsf.gtalkservice:,8,8]
04-20 13:09:36.477 837 1089 I force_gc: Binder
04-20 13:09:36.562 837 837 I notification_enqueue: [10093,13663,ru.yandex.yandexnavi,19810816,NU
exnavi/0x7f0300ae vibrate=null sound=null defaults=0x0 flags=0x22 color=0x00000000 vis=PUBLIC),0]
```

Вывод команды adb logcat -b events

## Продвинутый уровень

В одной из своих статей я показывал, как можно доставать информацию из баз данных различных приложений. Ну а теперь посмотрим, как проделать это прямо из консоли, не качая базы на комп и не устанавливая на устройство просмотрщики баз. Для этого используется команда **sqlite3**. Выведем на экран историю браузера Chrome:



```
$ cd /data/data/com.android.chrome
$ su
# sqlite3 app_chrome/Default/History
> .schema urls
> select * from urls where url like "%android%";
```

Чтобы база читалась, необходимо выгрузить браузер из работающих приложений. Прервать выполнение скрипта sqlite можно, нажав Ctrl + Z, а выйти — командой .quit. Если в ответ на команду ты получишь ошибку /system/bin/sh: sqlite3: not found, значит, на смартфоне нет sqlite3 и ее придется скачать, закинуть в /system/bin и дать файлу все права. Я использую [sqlite3](#), который вытащил когда-то из Titanium Backup.

```

Администратор: C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\podkopaev>adb shell
id: unknown ID 2000
2000@hammerhead:/ $ cd /data/data/com.android.chrome
2000@hammerhead:/data/data/com.android.chrome $ su
sqlite3 app_chrome/Default/History
SQLite version 3.7.6.3-Titanium
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .schema urls
CREATE TABLE urls(id INTEGER PRIMARY KEY,url LONGVARCHAR,title LONGVARCHAR,visit_count INTEGER DEFAULT 1,last_visit_time INTEGER NOT NULL,hidden INTEGER DEFAULT 0 NOT NULL,favicon_id INTEGER DEFAULT 0 NOT NULL);
CREATE INDEX urls_url_index ON urls (url);
sqlite> select * from urls where url like "%android%";
182|http://www.androidpolice.com/|Android Police - Android News, Apps, Games, Phones, Tablets|21|1|1
224|http://www.androidcentral.com/nexus-5-models-whats-different-between-two|Nexus 5 models 820 and 13088937251541739|1|0
262|http://www.pocketables.com/2012/05/guide-to-game-controllers-on-android.html|Guide to game controllers on Android|819|1|0
264|http://developer.android.com/reference/android/content/Intent.html|Intent | Android Developers|0
265|http://developer.android.com/guide/components/intents-filters.html|Intents and Intent Filters |
271|http://www.androidcentral.com/best-android-travel-apps|The top 10 best travel apps for Android |
280|http://www.xda-developers.com/android/android-wear-tools-smartwatch/?utm_source=feedburner&utm_medium=social&utm_campaign=android-wear-tools-smartwatch|Easily Manage Your Smartwatch with Android Wear Tools - XDA Forums|0|0|13088937251541739|1|0
281|http://www.xda-developers.com/android/top-5-android-wear-apps-xdatv/?utm_source=feedburner&utm_medium=social&utm_campaign=android-wear-apps-xdatv|Top 5 Android Wear Apps - XDA Developer TV|0|0|13088937251541739|1|0
282|http://rominirani.com/2014/07/12/tutorial-cast-your-android-wear-screen-to-all/|Tutorial : Cast your Android Wear screen to all devices|937251603309|1|0
284|http://forums.getpebble.com/discussion/5767/android-pebbledialer-new-call-controls-for-your-pebble-or-your-pebble! - Pebble Forums|0|0|13088937251604177|1|0
285|http://forums.getpebble.com/discussion/8053/android-notification-center-for-pebble-ultimate-notification-center-for-pebble - Ultimate notification replacement - Pebble Forums|0|0|13088937251605912|1|0
290|http://www.xda-developers.com/android/exit-the-chromecast-walled-garden-with-kyocast/?utm_source=feedburner&utm_medium=social&utm_campaign=android-wear-apps-xdatv|Exit the Chromecast Walled Garden with KyoCast |XDA Developers|0|0|13088937251605912|1|0
298|https://developers.google.com/android/nexus/images#yakju|Factory Images for Nexus Devices - Nexus Developer|0|0
311|http://forums.getpebble.com/discussion/6304/android-root-keep-for-pebble-display-google-keep-not-keep-for-pebble - Display Google Keep notes on your Pebble! - Pebble Forums|0|0|13088937251623658|1|0
314|http://forums.getpebble.com/discussion/7954/android-pebble-messenger-whatsapp-sms-quick-response-app/SMS, quick responses + typing!) - Pebble Forums|0|0|13088937251624679|1|0
315|http://forums.getpebble.com/discussion/10373/android-20-01-14-sdk-2-0-yanc-yet-another-notification-center - Yet another Notification Center - BETA - Pebble Forums|0|0|13088937251625142|1|0
317|http://forums.getpebble.com/discussion/8053/android-notification-center-for-pebble-ultimate-notification-center-for-pebble - Ultimate notification replacement - Pebble Forums|0|0|13088937251626250|1|0
319|http://www.androidpolice.com/2014/07/05/how-to-android-wear-enable-debugging-take-screenshots-unlock-the-bootloader-and-root-the-lg-g-watch/|Android Wear: Enable Debugging, Take Screenshots, Unlock The Bootloader, And Root The LG G Watch|326|1|0
326|http://www.xda-developers.com/android/flashcast-makes-chromecast-rooting-easy/|FlashCast Makes Chromecast Rooting Easy|8937251630450|1|0
341|http://www.androidauthority.com/fallout-1-and-2-android-536594/|Play Fallout 1 and Fallout 2 on Android|0|0|13088937251635257|1|0
347|http://www.androidauthority.com/android-authority-this-week-june-28-2015-620706/|Android Authority This Week: June 28, 2015|0|0|13088937251638406|1|0
366|http://www.xda-developers.com/android/unlock-root-moto-360-xdatv/?utm_source=feedburner&utm_medium=social&utm_campaign=android-wear-apps-xdatv|How to Unlock, Root, and Restore Your Moto 360|0|0|13088937251648473|1|0
1572|http://www.androidpolice.com/2016/01/14/psa-google-fixed-a-factory-reset-protection-bypass-bug-in-the-january-security-update/|A Factory Reset Protection Bypass Bug In The January Security Update|0|0|13097952484377387|0|0
1673|http://www.androidauthority.com/|Android Authority|3|3|13103797239930116|0|0

```

История браузера Chrome

Также с помощью sqlite3 можно выдернуть все контакты с телефона. Для этого в консоли на компе должен использоваться шрифт Lucida Console и перед началом выполнения команд необходимо перевести кодировку на UTF-8. Иначе вместо русских букв будут отображаться непонятные символы. Сами команды выглядят так:

```

chcp 65001
adb shell
$ su

```

```
# cd /data/data/com.android.providers.contacts/databases
# sqlite3 contacts2.db
> select t1.raw_contact_id,t1.normalized_number,t2.display_name
from phone_lookup as t1, raw_contacts as t2 where
t1.raw_contact_id=t2._id Order by display_name;
```

Если все сделано правильно, то в консоли ты увидишь таблицу с порядковым номером записи, номером телефона и контактами, отсортированными по имени. Для контактов с более одного номера будет несколько записей подряд.

```
Администратор: C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\podkopaev>chcp 65001
Active code page: 65001

C:\Users\podkopaev>adb shell
id: unknown ID 2000
2000@hammerhead:/ $ su
root@hammerhead:/ # cd /data/data/com.android.providers.contacts/databases
sqlite3 contacts2.db
SQLite version 3.7.6.3-Titanium
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select t1.raw_contact_id,t1.normalized_number,t2.display_name from phone_lookup as t1, raw_c
by display_name;
402|+861:|55|Austin Si
522|+861:|55|Austin Si
199|+792:| |Banan
1068|+784:|6|Dmitry
1068|+79:|6|Dmitry
912|+798:| |E I
338|+792:| |E.I.
338|+798:| |E.I.
51|+7902:| |Gangsta
51|+7937:| |Gangsta
1286|+184:|3|Jeffrey Borek
326|+792:| |John Doe
492|+792:| |John Doe
1327|+74:|9|Kelly
1060|+79:|9|Larita Prescott
1060|+44:|67|Larita Prescott
1195|+44:|67|Larita Prescott
256|+784:| |Linde
321|8846:| |Linde
321|+784:| |Linde
334|+792:| |Maman
334|5422:| |Maman
334|+798:| |Maman
925|+792:| |Maman
107|+790:| |Morgan
10|+7927:| |Mrak
200|+793:| |Nataliya
523|+793:| |Nataliya
40|+7927:| |RUTO
935|+792:| |RUTO
1181|+79:|3|RUTO
26|+7927:| |at
247|+792:| |Zazzoo
959|+792:| |Zazzoo
122|+790:| |fedya
498|+790:| |fedya
1160|+79:| |hisamu
1198|+79:| |hisamu
255|+793:| |nAbu
538|+793:| |nAbu
239|+796:| |warcraft
500|+796:| |warcraft
92|+7842:| |автовазбанк
```

Вывод контактов из базы contacts2.db



Можно вывести данные не на экран, а сразу в текстовый файл. Для этого команды нужно изменить:

```
adb shell
$ su
# cd /data/data/com.android.providers.contacts/databases
# sqlite3 contacts2.db "select
t1.raw_contact_id,t1.normalized_number,t2.display_name from
phone_lookup as t1, raw_contacts as t2 where
t1.raw_contact_id=t2._id;" > /sdcard/contacts.txt
```

Альтернативный способ вывода контактов в файл — команда, требующая установленного BusyBox:

```
content query --uri content://contacts/phones --projection
number:name --sort "name ASC"| awk -F= '{gsub(/[-()
name]/, "", $2); print $2 " "$3}'| sed 's/,//g' >/sdcard/contacts.txt
```

## Снятие графического ключа, PIN, facelock

Допустим, ты забыл PIN или не совсем трезвым поставил графический ключ, ну или друзья пошутили и поставили распознавание по лицу... Так вот, если устройство по какой-то причине заблокировано, то блокировку можно снять (при условии включенной отладки по USB) через ту же консоль:

```
adb shell
$ su
# cd /data/system
# rm *.key
```

Команда удалит все пароли и графические ключи. Сами файлы в зависимости от прошивки и модели устройства могут быть: gesture.key, password.key, cm\_gesture.key, personalpattern.key, personalbackuppinn.key. Также за блокировку отвечают файлы locksettings.db, locksettings.db-shm, locksettings.db-wal.

После этого достаточно перезагрузить устройство и ввести любой ключ, пароль. Если это не помогает, можно попробовать следующее:

```
adb shell
$ cd /data/data/com.android.providers.settings/databases
$ sqlite3 settings.db
> update system set value=0 where name='lock_pattern_autolock';
> update system set value=0 where
name='lockscreen.lockedoutpermanently';
```

## Выводы

Как видишь, с помощью ADB можно сделать много интересного. И чем больше пользуешься консолью, тем быстрее можно выполнить некоторые действия без установки дополнительного софта на устройство. Надеюсь, данная статья помогла разобраться с ADB и подтолкнула к чтению документации и поиску новых полезных команд.