**https://adbshell.com/commands/adb-shell-netstat**

**Содержание статьи**

* [Основы работы с ADB](https://xakep.ru/2016/05/12/android-adb/#toc01.)
* [Установка программ](https://xakep.ru/2016/05/12/android-adb/#toc02.)
* [Бэкап приложений](https://xakep.ru/2016/05/12/android-adb/#toc03.)
* [Консоль в консоли](https://xakep.ru/2016/05/12/android-adb/#toc04.)
* [Создание скриншота](https://xakep.ru/2016/05/12/android-adb/#toc05.)
* [Запись видео, происходящего на экране устройства](https://xakep.ru/2016/05/12/android-adb/#toc06.)
* [Управление приложениями](https://xakep.ru/2016/05/12/android-adb/#toc07.)
* [Системные утилиты](https://xakep.ru/2016/05/12/android-adb/#toc08.)
* [Снятие логов](https://xakep.ru/2016/05/12/android-adb/#toc09.)
* [Продвинутый уровень](https://xakep.ru/2016/05/12/android-adb/#toc10.)
* [Снятие графического ключа, PIN, facelock](https://xakep.ru/2016/05/12/android-adb/#toc11.)
* [Выводы](https://xakep.ru/2016/05/12/android-adb/#toc12.)

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 мощью, 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](https://play.google.com/store/apps/details?id=jackpal.androidterm&hl=ru), но удобнее это делать, конечно же, с компа через adb.

**Основы работы с ADB**

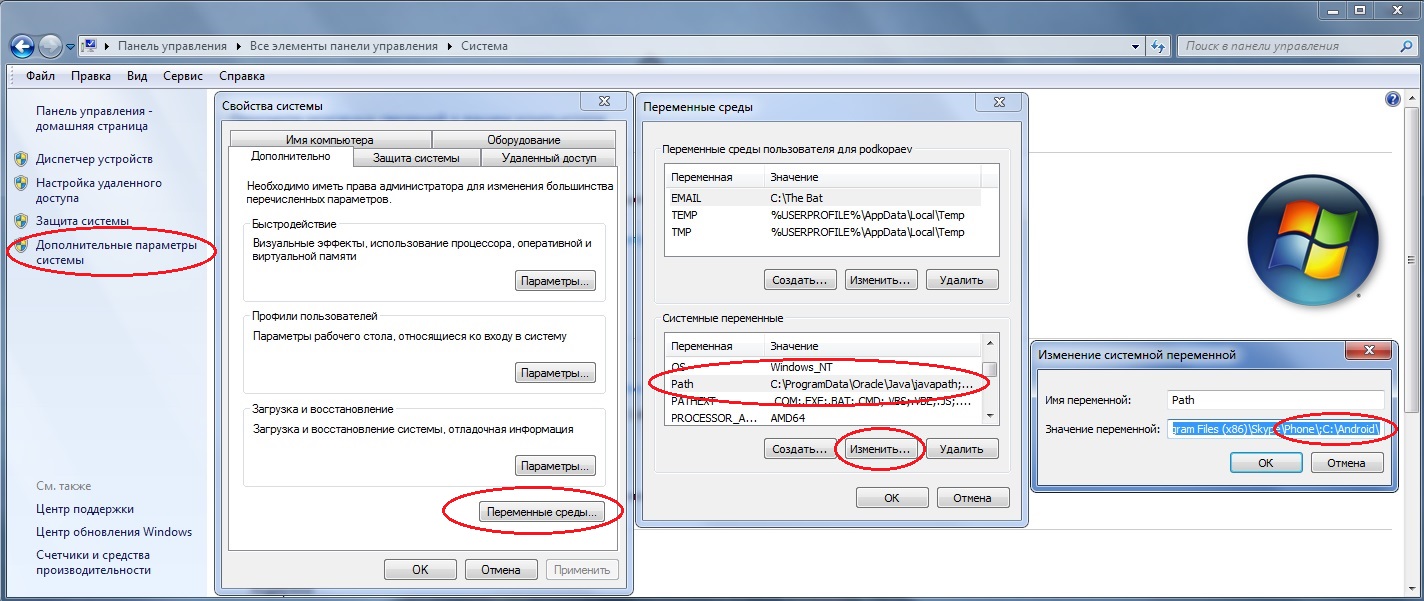
Для начала работы с ADB его следует активировать на устройстве и установить утилиту adb и драйверы на комп. Первая задача выполняется с помощью включения «Отладки по USB» в пункте настроек «Для разработчиков» (если этот пункт скрыт, нажми семь раз на номер сборки в меню «О телефоне»).

Для установки ADB на комп качаем [Adb Kit](http://adbshell.com/downloads) и распаковываем в любую папку (рекомендую использовать названия папок без русских символов). Также скачиваем и устанавливаем [драйверы ADB](http://adbdriver.com/downloads).

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.

[](https://xakep.ru/wp-content/uploads/2016/05/1463036695_305c_ris.1.jpg)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](https://play.google.com/store/apps/details?id=com.ttxapps.wifiadb). Запускаем приложение, жмем переключатель и подсоединяемся 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 *-е* — 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>

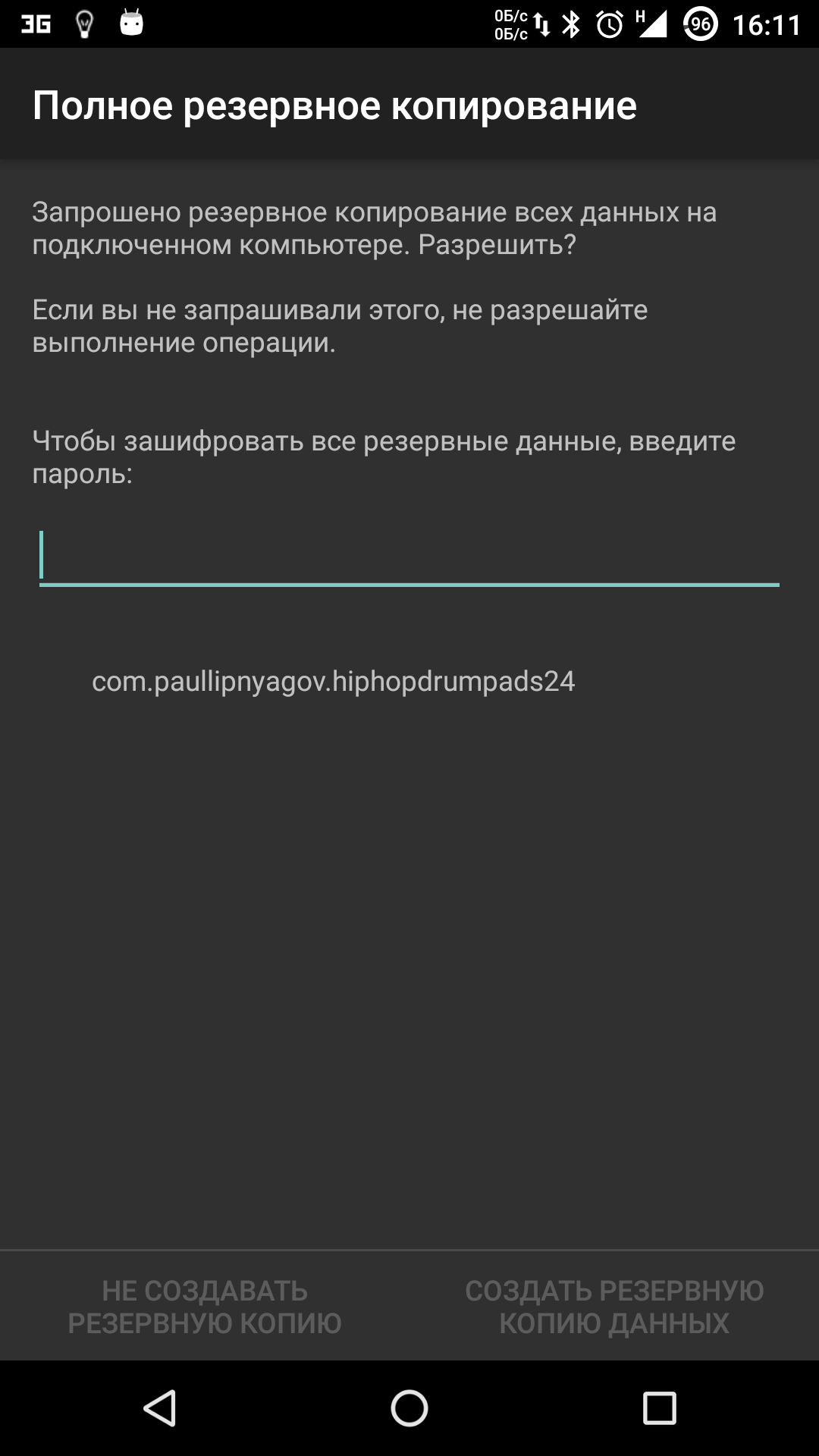
* *-f* указывает имя создаваемого файла и его расположение на компе. При отсутствии ключа будет создан файл backup.ab в текущем каталоге;
* *-apk|-noapk* указывает, включать ли в бэкап только данные приложения или сам .apk тоже (по умолчанию не включает);
* *-obb|-noobb* указывает, включать ли в бэкап расширения .obb для приложений (по умолчанию не включает);
* *-shared|-noshared* 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

[](https://xakep.ru/wp-content/uploads/2016/05/1463036758_205d_ris.3.png)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](https://busybox.net/downloads/BusyBox.html) который можно, установив из маркета BusyBox. Использовать эту консоль можно двумя способами. В интерактивном режиме она запускается командой

adb shell

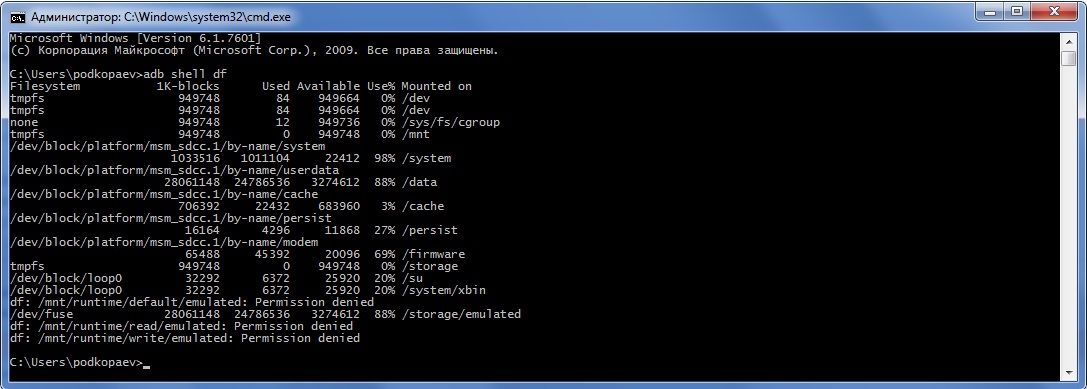
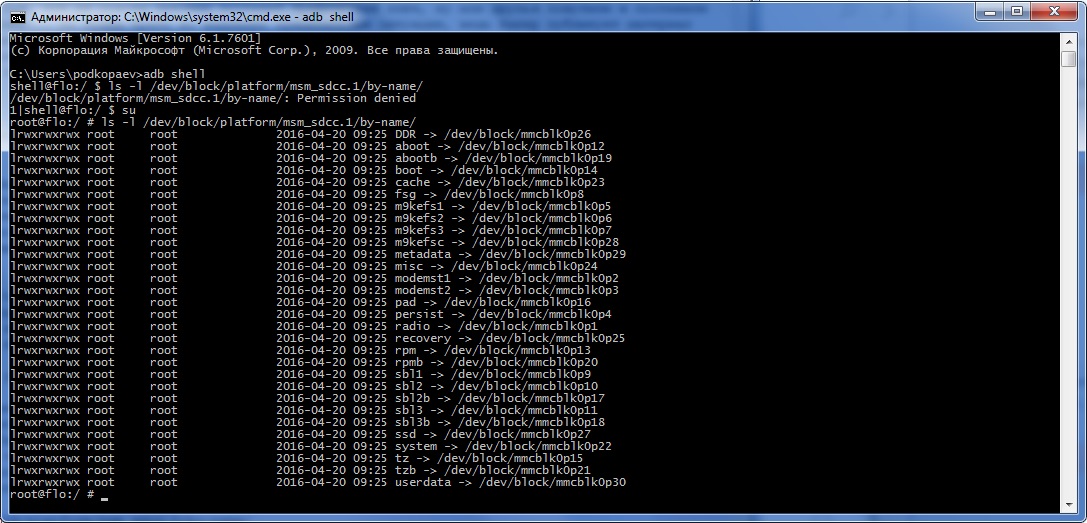
В консоли появляется знак $ (далее по тексту этот знак будет означать необходимость ввода предварительной команды adb shell), и после этого можно вводить серию команд, получая после каждой отклик. Второй способ — если необходимо ввести только одну команду, можно писать ее подряд за adb shell.

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 #.

[](https://xakep.ru/wp-content/uploads/2016/05/1463036774_b07b_ris.7.jpg)Вывод свободного пространства на устройстве командой adb shell df  
[](https://xakep.ru/wp-content/uploads/2016/05/1463036780_caaf_ris.10.jpg)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](http://developer.android.com/intl/ru/tools/help/shell.html). 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 [Avtivity](http://developer.android.com/intl/ru/reference/android/app/Activity.html) и [Intent](http://developer.android.com/intl/ru/tools/help/shell.html" \l "IntentSpec). 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:

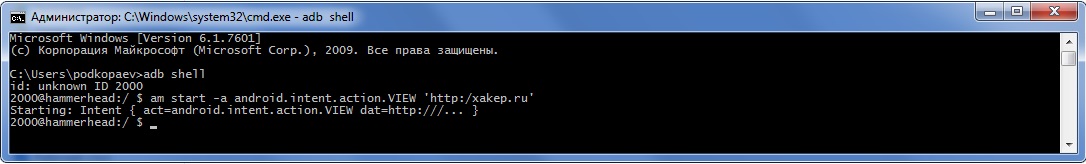
$ am kill-all

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'

[](https://xakep.ru/wp-content/uploads/2016/05/1463036811_709b_ris.6.jpg)Запуск браузера из консоли

А с помощью вариации предыдущей команды можно отправить СМС:

$ am start -a android.intent.action.SENDTO -d sms:НОМЕР\_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](http://goo.gl/hAzMh)):

$ 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 х 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:

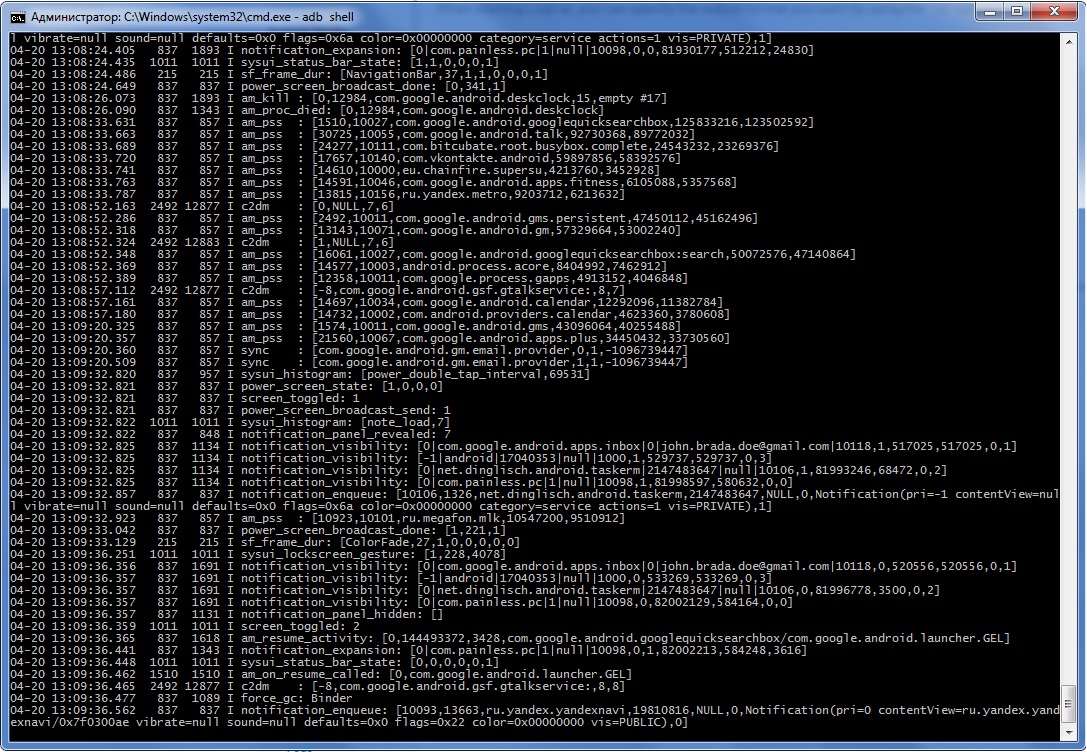
adb logcat > logcat.txt

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](http://developer.android.com/intl/ru/tools/debugging/debugging-log.html), 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 *Е* 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

[](https://xakep.ru/wp-content/uploads/2016/05/1463036864_1a47_ris.8.jpg)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

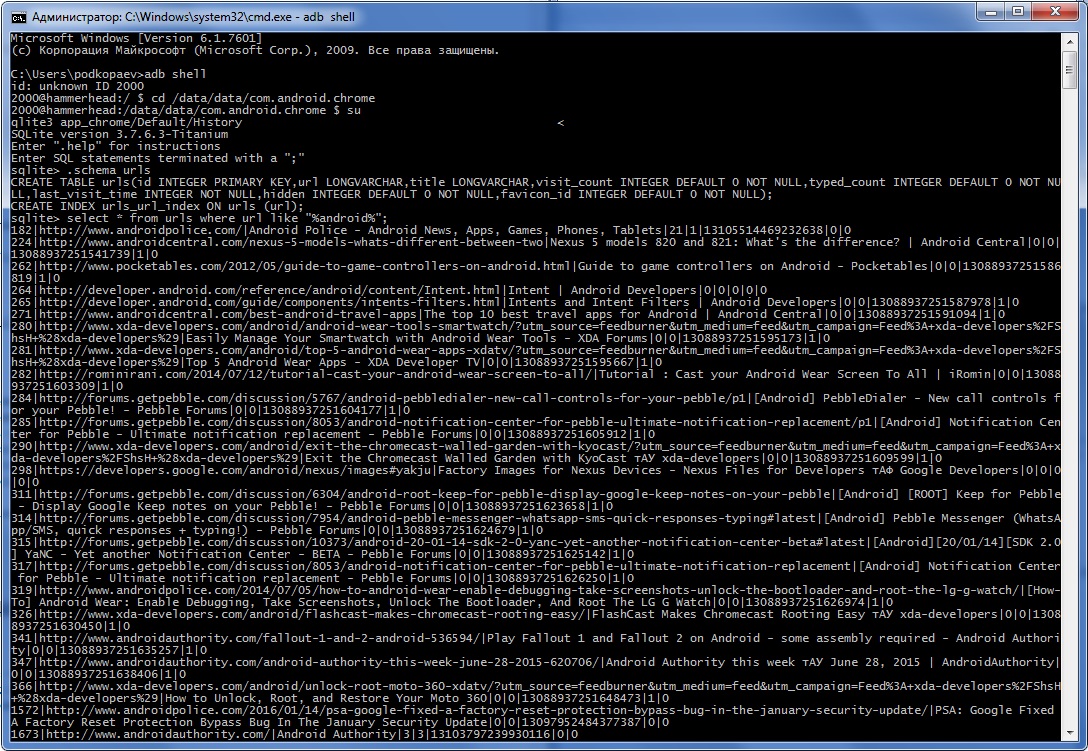
$ su

# 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](https://goo.gl/AUMYNs), который вытащил когда-то из Titanium Backup.

[](https://xakep.ru/wp-content/uploads/2016/05/1463036876_fa04_ris.4.jpg)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

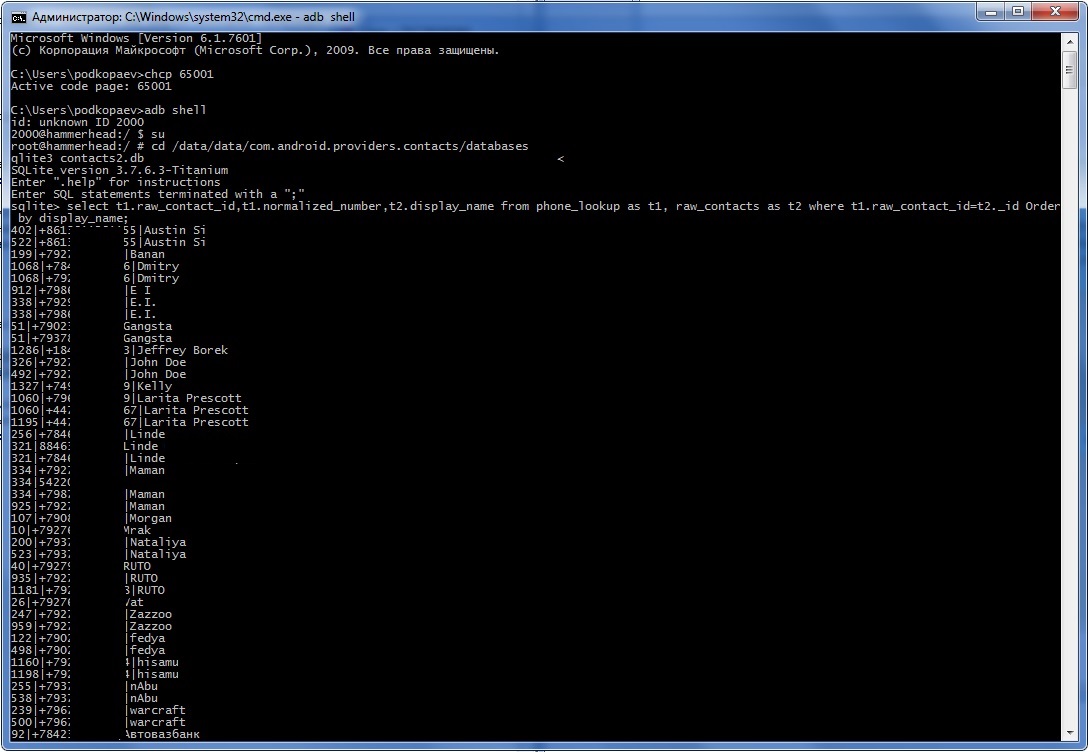
$ su

# 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.

[](https://xakep.ru/wp-content/uploads/2016/05/1463036891_da8f_ris.5.jpg)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, personalbackuppin.key. Также за блокировку отвечают файлы locksettings.db, locksettings.db-shm, locksettings.db-wal.

После этого достаточно перегрузить 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.