Превращаем Android-смартфон в USB-клавиатуру, мышь и флешку

Android-смартфон при подключении к компьютеру через USB может выглядеть как медиаустройство, модем или хранилище файлов. В этой статье мы научимся делать из него USB-клавиатуру, мышь и флешку.

ВАЖНО!!! Все описанные в статье действия потребуют «взлома» смартфона и получения на нём прав суперпользователя (root). В процессе «взлома» вы можете потерять на устройстве все данные или вовсе необратимо сломать его. Будьте осторожны! Обдумывайте свои действия, не делайте того, о чём можете пожалеть!

Быстрый старт

Сначала практика, потом теория? Нет проблем. Если вы не понаслышке знакомы с мобильными технологиями и хорошо разбираетесь с архитектурой Android, то эта глава специально для вас. Если же вы новичок, то лучше сначала прочитать главу «Подготовка смартфона», а потом вернуться сюда.

Предполётная проверка

Для того чтобы приступить к экспериментам необходимо:

1. Android-смартфон с разблокированной учётной записью root.
2. Установленный на смартфон эмулятор терминала termux.
3. Настроенный в termux OpenSSH сервер.
4. Смартфон должен находиться в Wi-Fi сети, доступной с компьютера.

Проверка считается пройденной, если вы можете подключиться к смартфону по SSH и успешно выполнить на нём команду «su». Если с этим есть проблемы, то переходите в раздел «Подготовка смартфона», там разберёмся, как что настроить.

Переконфигурирование USB-порта смартфона в режим клавиатура, мышь и флешка

В termux выполните скрипт, который настроит USB-порт смартфона под наши задачи. *Примечание. Все скрипты и команды в этой статье (если не сказано другого) нужно запускать из-под учётной записи root.*

```
#!/system/bin/sh
# add_profile.sh

PROFILE_NAME="xxx"
PROFILE_BASEPATH="/config/usb_gadget/"
PROFILE_PATH="${PROFILE_BASEPATH}${PROFILE_NAME}"
PROFILE_CONFIG_NAME="b.1"
STORAGE_FILE_NAME="/data/data/com.termux/files/home/storage_file.img"
STORAGE_FILE_SIZE="100M"

# More info about USB Report Descriptors:
# - Human Interface Devices (HID) Specifications and Tools: https://www.usb.org/hid
# - USB HID Report Descriptor: https://github.com/tmk/tmk_keyboard/wiki/USB:-HID-
Report-Descriptor
# - Device Class Definition for Human Interface Devices (HID). B.1.
# - -  https://usb.org/sites/default/files/hid1_11.pdf
```

```
KEYBOARD_REPORT_DESC="\
\\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\
\\x15\\x00\\x25\\x01\\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\
\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01\\x05\\x08\\x19\\x01\
\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\x03\\x95\\x06\
\\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\
\\x81\\x00\\xc0"
# Version for online parser https://eleccelerator.com/usbdescreqparser/
# 0x05 0x01 0x09 0x06 0xa1 0x01 0x05 0x07 0x19 0xe0 0x29 0xe7\
# 0x15 0x00 0x25 0x01 0x75 0x01 0x95 0x08 0x81 0x02 0x95 0x01\
# 0x75 0x08 0x81 0x03 0x95 0x05 0x75 0x01 0x05 0x08 0x19 0x01\
# 0x29 0x05 0x91 0x02 0x95 0x01 0x75 0x03 0x91 0x03 0x95 0x06\
# 0x75 0x08 0x15 0x00 0x25 0x65 0x05 0x07 0x19 0x00 0x29 0x65\
# 0x81 0x00 0xc0

MOUSE_REPORT_DESC="\
\\x05\\x01\\x09\\x02\\xa1\\x01\\x09\\x01\\xa1\\x00\\x05\\x09\
\\x19\\x01\\x29\\x05\\x15\\x00\\x25\\x01\\x95\\x05\\x75\\x01\
\\x81\\x02\\x95\\x01\\x75\\x03\\x81\\x01\\x05\\x01\\x09\\x30\
\\x09\\x31\\x09\\x38\\x15\\x81\\x25\\x7F\\x75\\x08\\x95\\x03\
\\x81\\x06\\xc0\\xc0"
# Version for online parser https://eleccelerator.com/usbdescreqparser/
# 0x05 0x01 0x09 0x02 0xa1 0x01 0x09 0x01 0xa1 0x00 0x05 0x09\
# 0x19 0x01 0x29 0x05 0x15 0x00 0x25 0x01 0x95 0x05 0x75 0x01\
# 0x81 0x02 0x95 0x01 0x75 0x03 0x81 0x01 0x05 0x01 0x09 0x30\
# 0x09 0x31 0x09 0x38 0x15 0x81 0x25 0x7F 0x75 0x08 0x95 0x03\
# 0x81 0x06 0xc0 0xc0


# Device parameters
VID="0xDEAD"
PID="0xBEAF"
MANUFACTURER="DEAD"
PRODUCT="BEAF"
SERIAL_NUMBER="AQAK-10"
PROFILE_CONFIGURATION_STR="Multi-Device"


echo "Creating new USB profile..."
# Fileds info: https://www.kernel.org/doc/Documentation/ABI/testing
mkdir -p $PROFILE_PATH
cd $PROFILE_PATH
if [ $? -eq 0 ]; then
    # Base Class 00h (Device): https://www.usb.org/defined-class-codes
    echo 0x00   > bDeviceClass    # 00 - Class Defined by Interface ( composite )
    echo 0x00   > bDeviceProtocol # Base Class 00h (Device):
https://www.usb.org/defined-class-codes
    echo 0x00   > bDeviceSubClass #

    echo 0x40   > bMaxPacketSize0 # = 64 bytes. Auto assigned by Android, can't
change
    echo 0x0200 > bcdUSB          # = USB 2.0. Auto assigned by Android, can't
change
    echo 0x0333 > bcdDevice       # Device release number. Can be any
```

```
    echo $VID   > idVendor
    echo $PID   > idProduct
else
    echo "USB profile create fail."
    exit 1
fi

mkdir -p "$PROFILE_PATH/strings/0x409"
cd "$PROFILE_PATH/strings/0x409/"
if [ $? -eq 0 ]; then
    echo $MANUFACTURER  > manufacturer
    echo $PRODUCT       > product
    echo $SERIAL_NUMBER > serialnumber
else
    echo "strings create fail"
    exit 1
fi

# create keyboard function
mkdir -p "$PROFILE_PATH/functions/hid.keyboard"
cd "$PROFILE_PATH/functions/hid.keyboard"
if [ $? -eq 0 ]; then
    echo 1 > protocol # Keyboard
    echo 1 > subclass # Boot Interface Subclass
    echo 8 > report_length
    echo -ne $KEYBOARD_REPORT_DESC > report_desc
else
    echo "keyboard function create fail"
    exit 1
fi

# create mouse function
mkdir -p "$PROFILE_PATH/functions/hid.mouse"
cd "$PROFILE_PATH/functions/hid.mouse"
if [ $? -eq 0 ]; then
    echo 2 > protocol # Mouse
    echo 1 > subclass # Boot Interface Subclass
    echo 4 > report_length
    echo -ne $MOUSE_REPORT_DESC > report_desc
else
    echo "mouse function create fail"
    exit 1
fi

# create mass_storage image file
if [ ! -f "$STORAGE_FILE_NAME" ]; then
    fallocate -l $STORAGE_FILE_SIZE $STORAGE_FILE_NAME
fi

# create mass_storage function
mkdir -p "$PROFILE_PATH/functions/mass_storage.0"
cd "$PROFILE_PATH/functions/mass_storage.0/lun.0"
if [ $? -eq 0 ]; then
    echo 1 >../stall
    echo 0 > cdrom
```

```
    echo 0 > removable
    echo 0 > ro
    echo 0 > nofua
    echo "$STORAGE_FILE_NAME" > file
else
    echo "mass_storage function create fail"
    exit 1
fi

# DISABLE ALL USB Profiles!
find $PROFILE_BASEPATH  -name UDC -type f -exec sh -c 'echo "" >  "$@"' _ {} \;

# enable functions
# Device can have many configurations, like c.1, a.1, etc., but host chose from it
# Usually, device have only one config
mkdir -p "${PROFILE_PATH}/configs/${PROFILE_CONFIG_NAME}/strings/0x409"
cd "$PROFILE_PATH/configs/${PROFILE_CONFIG_NAME}/"
if [ $? -eq 0 ]; then
    echo $PROFILE_CONFIGURATION_STR > strings/0x409/configuration
    find $PROFILE_PATH/functions/* -type d -maxdepth 0 -exec sh -c 'ln -s $@ ./' _
{} \;
else
    echo "functions enable fail"
    exit 1
fi

cd $PROFILE_PATH
ls /sys/class/udc > UDC

exit 0
```

Проверка активации функций клавиатуры и мыши

Чтобы убедиться, что всё прошло как надо в termux выполните команду:

```
ls /dev | grep hidg
```

Если в результате выполнения этой команды вы видите строки «hidg0» и «hidg1» значит функции клавиатуры и мыши активировались успешно.

Имитация нажатия кнопки «a»

Порт переконфигурирован. Всё готово к первому шагу. Попробуем «нажать» кнопку «a». *Примечание. Не забудьте подключить смартфон к USB-порту и найти на экране компьютера строку, где должна появиться нажатая буква.*

```
#!/system/bin/sh
# print_a.sh
```

```
echo "\x00\x00\x04\x00\x00\x00\x00\x00" > /dev/hidg0 #press a
echo "\x00\x00\x00\x00\x00\x00\x00\x00" > /dev/hidg0 #release a
```

Имитация движения мыши

Теперь пошевелим курсором мыши. Следующий скрипт переместит курсор мыши на некоторое расстояние вниз. Чтобы лучше это увидеть, предварительно поставьте курсор в центр экрана.

```
#!/system/bin/sh
# mouse_move_down.sh

echo "\x00\x00\x55\x00" > /dev/hidg1
```

Проверка работы флешки

Это, пожалуй, самый простой тест. Если вы используете Windows, то откройте «Disk Management» («Управление дисками»), там появится новый диск на 100Mb (Рис. 1). Для использования его нужно просто отформатировать. Если у вас другая операционная система, то нужно выполнить аналогичные действия, но уже с вашей спецификой.

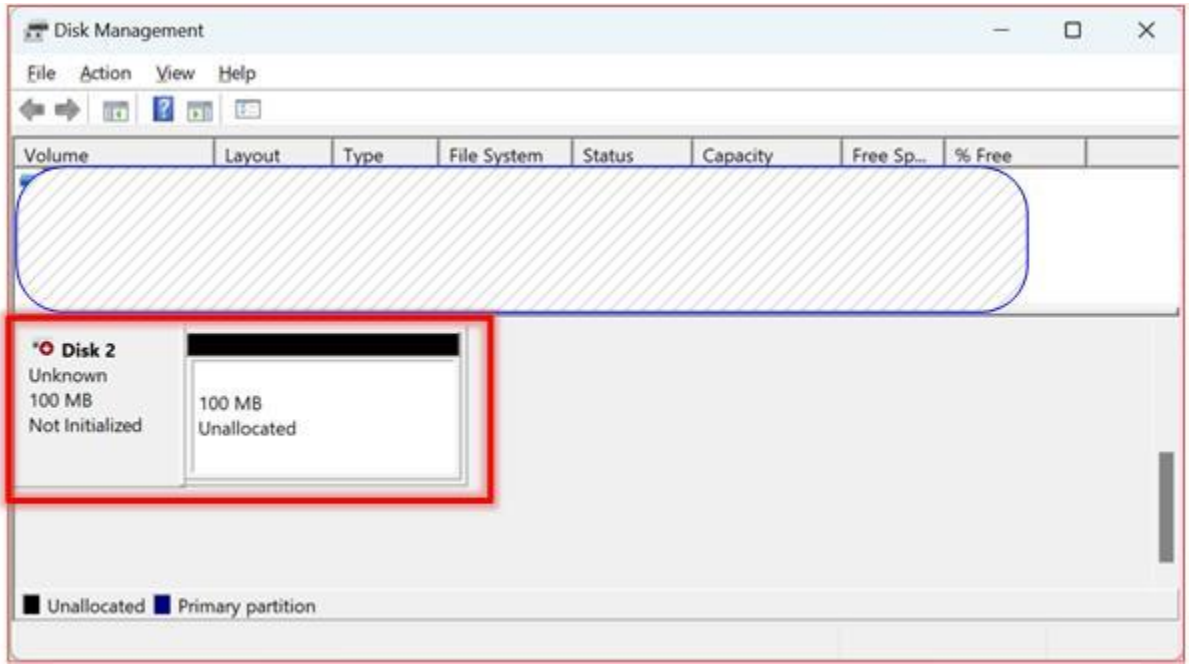

Рис. 1

Интересно? Хочется подробностей? Добро пожаловать в статью.

Подготовка смартфона

Всем, кто успешно прошёл «Быстрый старт», можно смело пропускать данную главу.

Получение root

Напоминаю, что этот процесс довольно опасен. Не делайте его на вашем основном смартфоне. Используйте отдельный смартфон для тестов, который не жалко угробить. Например, подопытным кроликом для данной статьи послужил ASUS ZenFone Max Pro M1 (Android 8.1) с разбитым экраном, который до этого без дела валялся на полке.

Процесс получения root в смартфоне, как правило, состоит из следующих этапов:

1. Активация в смартфоне отладки через USB.
2. Разблокировка загрузчика.
3. Замена стандартной прошивки восстановления на специально подготовленную (custom recovery), например, TWRP.
4. Установка из TWRP специальной утилиты Magisk.
5. Инсталляция из стандартной оболочки смартфона менеджера Magisk.

Первые этапы специфичны для каждого смартфона. Найти необходимые инструкции и прошивки вы скорее всего сможете на сайтах 4PDA или XDA Forums. Если их там нет, то Яндекс в помощь.

Хочу заострить ваше внимание на одной важной особенности получения root, о которой почему-то нигде не пишут. При перепрошивке смартфона подключать его следует только к USB 2.0 портам компьютера. Если их нет, то потребуется USB 2.0 хаб. Использование USB 3.0 портов (или более новых) может сопровождаться ворохом проблем:

- Смартфон может самопроизвольно выходить из режима быстрой загрузки.
- Утилита fastboot будет срабатывать «через раз» и выдавать ошибки: FAILED (command write failed (No error)), FAILED (status read failed (Too many links)) и прочие.

Установка termux

После получения root необходимо установить эмулятор терминала termux. С его помощью мы будем выполнять все эксперименты. Ставить termux лучше с сайта F-Droid, поскольку

в «Google play» размещена старая версия, имеющая проблемы с репозиториями.

Настройка SSH сервера в termux

Команды настройки SSH сервера должны выполняться от обычной учётной записи, root использовать не нужно.

1. Первым делом необходимо обновить termux:

```
pkg update -y
```

   Во время выполнения обновлений менеджер пакетов будет спрашивать, что делать с имеющимися конфигурационными файлами. Для наших целей лучше их обновить.

2. Установим необходимые пакеты:

```
pkg install -y termux-auth openssh
```

3. Теперь нам необходимо узнать имя пользователя, под которым работает termux. Наберём команду:

```
id
```

   Ответ будет содержать строку с идентификаторами, среди которых «uid» (user id) будет содержать требуемое имя.

4. Для того, чтобы иметь возможность авторизации в SSH по паролю, необходимо его назначить. Сделаем это с помощью команды:

```
passwd
```

5. Узнаем IP-адрес, который получил смартфон по WiFi. Выполним команду:

```
ipconfig
```

6. Запустим OpenSSH сервер:

```
sshd
```

   Всё готово для подключения по SSH. Мы знаем IP адрес смартфона, у нас есть имя пользователя и пароль, единственное порт для подключения стоит использовать 8022, так как именно его используется OpenSSH в termux по умолчанию.

Решение проблем с OpenSSH в termux

Если с компьютера подключиться к смартфону по SSH не удаётся, то можно попробовать следующие варианты:

1. Перезагрузить OpenSSH, последовательно выполнив команды:

```
pkill sshd
sshd
```

2. Перезагрузить смартфон.

Одной из возможных причин недоступности SSH в termux может быть установка на смартфоне брандмауэра, блокирующего любые входящие подключения. В данном случае необходимо развернуть на компьютере SSH сервер, со смартфона подключится к нему SSH клиентом с пробросой портов, а затем с компьютера SSH клиентом подключится в проброшенный порт.

При сохранении проблем попробуйте поискать ответы в официальном руководстве termux по настройке удалённого доступа.

Настройка прав root в Magisk

Первый раз, набрав команду «su», в termux сработает оповещение от Magisk, которое запросит вашего одобрения наделения (grant) приложения termux root доступом. Необходимо согласиться.

Настройка USB-порта в Andoid

Особенности использования привилегий root в Android

Если очень упростить, то можно сказать, что операционная система Android – это слоёный пирог (Рис. 2), где всю низкоуровневую работу выполняет Linux, а приложения работают в виртуальной машине Android runtime.

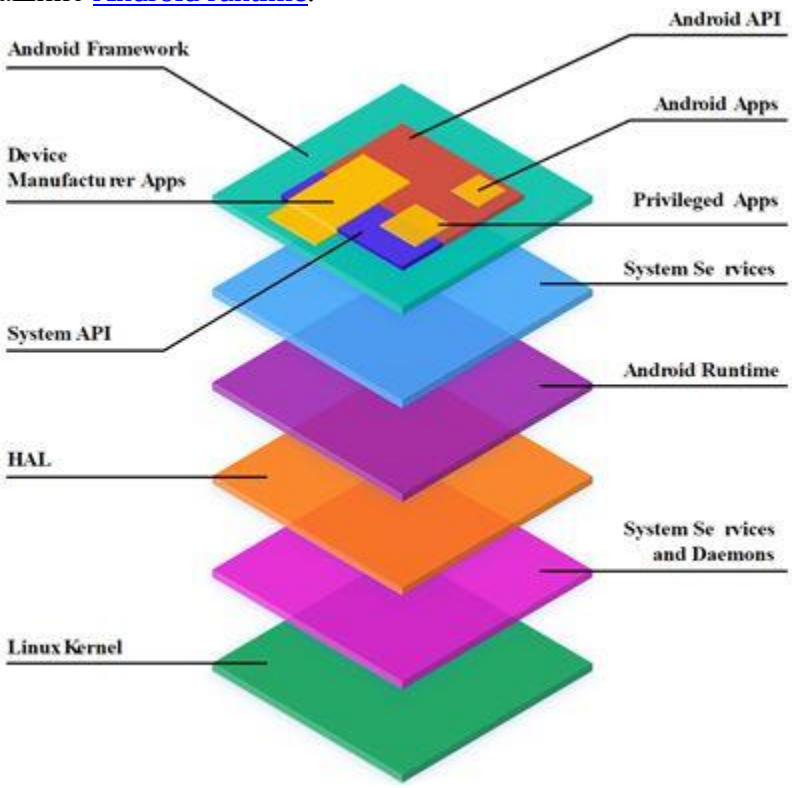

Рис. 2

Приложения бывают обычные, привилегированные (системные) и разработанные поставщиком устройства (Device manufacturer apps). Перенастройка USB-порта недоступна обычным приложениям, а повышение уровня доступа до привилегированного сопряжено с определёнными сложностями.

Но есть другой вариант. Дело в том, что в виртуальной машине Android runtime существует API, позволяющий выполнять Linux-команды, а уже с их помощью можно сделать практически всё что угодно. Ограничением будут лишь полномочия пользователя, под которым работает приложение, и механизмы разграничения доступа самого Linux (например, SELinux).

Процедура перенастройки USB-интерфейса с помощью вызова Linux-команд является привилегированной и недоступна для обычного приложения, требуется root-доступ. Несмотря на то что мы ранее получили root, это вовсе не значит, что все наши приложения стали запускаться из-под пользователя root. Они всё так же запускаются из-под

ограниченной учётной записи, и то, чего нельзя было делать ранее (до получения root), нельзя делать и сейчас. Единственным важным отличием стало то, что теперь (после получения root) есть возможность выполнить команду «su», которая запускает привилегированную командную оболочку (shell), где как раз и можно выполнить все необходимые нам действия.

Что значит запустить от root

Всякий раз, когда требуется выполнить команду или скрипт из-под учётной записи root это означает, что предварительно нужно повысить полномочия с помощью команды «su», а потом выполнять требуемые действия.

Linux USB Gadget

В терминах Linux-ядра смартфон, оснащённый настраиваемым USB-контроллером, называется Linux USB Gadget. Виртуальная файловая система configfs — наиболее удобный способ его конфигурирования.

Смартфоны бывают разные, и в некоторых из них, например, в старых моделях подобного функционала может и не быть. Для того чтобы удостовериться, что ваш смартфон позволяет настраивать USB-порт через configfs, необходимо проверить опции Linux-ядра:

```
#!/system/bin/sh

echo KERNEL_VERSION=`(uname -r |cut -d '-' -f1 )`
gunzip -c /proc/config.gz | grep -i configfs
```

Для наших целей на выходе скрипта должны присутствовать строки:

```
…
CONFIG_USB_CONFIGFS=y
…
CONFIG_USB_CONFIGFS_MASS_STORAGE=y
…
CONFIG_USB_CONFIGFS_F_HID=y
…
```

Они означают, что Linux-ядро поддерживает настройку USB через configfs, и вдобавок поддерживаются режимы MASS_STORAGE (флешка) и HID (клавиатура, мышь).

Настройка USB-интерфейса через configfs

Определить точку монтирования configfs можно с помощью команды:

```
mount | grep configfs
```

Обычно в Android это «/config». Настройки USB-интерфейса лежат в каталоге «/config/usb_gadget/».

Вот здесь становится интересно. Если посмотреть содержимое этого каталога:

```
ls /config/usb_gadget/
```

то можно увидеть, что в нём существует единственный подкаталог «g1» – это так называемый системный профиль USB-порта, созданный оболочкой Android. Кроме него, в каталоге «/config/usb_gadget/» можно создать сколь угодно других профилей, но активным

может быть только один. В связи с этим у нас есть две стратегии конфигурирования USB-порта:

1. Мы можем создавать новый профиль и затем делать его активным, отключив старый профиль (так сделано в разделе «Быстрый старт»).
2. Мы можем добавлять/удалять функции в уже существующем активном профиле.

Каждый подход имеет свои плюсы и минусы. Создавая и активируя новый профиль, поддерживающий клавиатуру, мышь и флешку, мы будем вынуждены отключить системный профиль, что, например, может привести к тому, что отладка по USB будет недоступна.

Изменяя функционал существующего, например, системного профиля, мы, в некоторых случаях можем получить плавающие «глюки» в работе USB, а также конкуренцию с оболочкой Android по настройке USB-порта. Поэтому каждая конкретная ситуация будет требовать своего решения.

Использование виртуальной файловой системы configfs для настройки USB-порта имеет ряд особенностей:

1. За исключением имени профиля, названия всех файлов и каталогов предопределены. Внутри configfs нельзя создать произвольный файл или каталог.
2. Форматы записываемых в файлы данных предопределены.
3. Для удаления каталогов следует использовать «rmdir», предварительно удалив из него все внутренности. Использование «rm -rf» может повлечь проблемы.
4. Изменения в configfs пропадают после перезагрузки устройства.

Несколько слов о конфигурации USB-устройств

Перед тем как приступить к конфигурированию USB-порта, нам потребуется разобраться с принципами функционирования USB-устройств. Первое, на чём хочется заострить внимание, это то, что наша идея превратить смартфон одновременно в клавиатуру, мышь и флешку не является чем-то экстраординарным. Никого ведь не смущают МФУ, совмещающие в себе одновременно фикции принтера, сканера и факса. Подобные USB-устройства, обладающие множеством функций, в терминах USB, называются композитными. Кстати, множество игровых мышек также являются композитными устройствами, сочетающими в себе функции мышки и клавиатуры (Рис. 3). Поэтому наш гибрид – вполне заурядная вещь в мире USB.

Рис. 3

Поскольку к USB-шине может быть подключено множество самых разных устройств, компьютер (USB-хост) должен понимать, что к нему подключилось и как с этим работать. Для этого USB-устройство обменивается с компьютером набором дескрипторов:

1. Дескриптор устройства, содержащий информацию о том, какая версия USB поддерживается устройством, классе устройства, идентификаторы производителя и устройства, а также ряд других параметров.
2. Дескрипторы конфигураций устройства. Спецификация USB позволяет устройствам иметь несколько конфигураций, например, одна конфигурация может подразумевать питание устройства от встроенного блока питания, а вторая – питание через шину USB. Операционная система компьютера может выбрать одну из конфигураций устройства в зависимости от своих настроек и параметров работы. Например, ноутбук при питании от сети может активировать конфигурацию устройства с питанием через шину USB, а при питании от батареи использовать конфигурацию, подразумевающую питание устройства через встроенный блок питания.
3. Дескрипторы строк. Содержат текстовую информацию, ассоциированную с устройством.
4. Дескрипторы отчёта HID устройства (USB HID Report descriptor). Данный вид дескриптора применяется только к HID (Human Interface Device) устройствам, таким как мышь, клавиатура, джойстик и т. д. А вот USB-флешка, например, подобный дескриптор отправлять не будет, поскольку не является HID устройством. Если максимально упростить, то в самом дескрипторе содержаться сведения о том, как интерпретировать поступающую с устройства информацию.

Таким образом, конфигурирование USB-порта с помощью configfs сводится к тому, чтобы предоставить Linux-ядру необходимую информацию, на базе которой оно сформирует

требуемый набор дескрипторов.

Файловая структура configfs

Настало время разобраться со структурой configfs в части конфигурирования USB-порта. Для этого создадим в каталоге /config/usb_gadget/ новый профиль устройства и назовём его «xxx»:

```
mkdir /config/usb_gadget/xxx
```

Сразу после выполнения этой команды в каталоге «/config/usb_gadget/xxx» была создана следующая файловая структура:

Файл или каталог	**Описание**

UDC	Файл, указывающий на активность профиля. Если файл пуст, то профиль не активен, если нет, то в нём указывается имя драйвера устройства, ассоциированного с этим профилем.
	Для отключения профиля используется команда:
	`echo "" > UDC`
	Для активации профиля:
	`ls /sys/class/udc > UDC`

Параметры, используемые для формирования дескриптора устройства	
bDeviceClass	Файлы, содержащие шестнадцатеричные коды USB-устройств. Перечень возможных кодов приведён на сайте USB-IF
bDeviceProtocol	
bDeviceSubClass	
bMaxPacketSize0	Максимальный размер пакета для конечной точки 0. Допустимые размеры 8, 16, 32, 64. По факту, вне зависимости от указанного здесь значения Android USB Gadget будет использоваться значение 64.

Файл или каталог	**Описание**
bcdUSB	Номер спецификации USB, с которой совместимо устройство. По факту, вне зависимости от указанного здесь значения Android USB Gadget будет использоваться значение совместимости с USB 2.0.
bcdDevice	Номер версии устройства.
idVendor	Параметры, используемые операционной системой для нахождения драйвера для устройства. idVendor назначается организацией USB-IF, idProduct назначается производителем.
idProduct	
Параметры, используемые для формирования дескрипторов конфигурации и строковых дескрипторов	
functions/	В данном каталоге содержатся подкаталоги, описывающие определённую функцию USB-устройства. Подкаталог «hid.keyboard» описывает клавиатуру, «hid.mouse» – мышь, «mass_storage.0» – флешку. Детальное содержимое этих подкаталогов будет рассмотрено далее.
configs/	Каталог содержит подкаталоги, соответствующие конфигурации устройства. Подкаталог с конфигурацией должен иметь имя, состоящее из буквы и цифры, например, «b.1».
config/<подкаталог с конфигурацией устройства>/	Здесь содержатся символические ссылки на каталоги функций из папки «functions/», а также ряд файлов, содержимое которых для нас неважно. Наличие символической ссылки делает функцию активной. Например, если в рассматриваемом каталоге будет символическая ссылка на каталог …functions/hid.mouse, это будет означать, что функция hid.mouse активна.
strings/	Содержит подкаталоги с описанием устройства на заданных языках. Имя подкаталога соответствует коду языка. Коды языков определены организацией USB.org. Например, код 0x409 соответствует английскому языку (en-US).
strings/<код языка>/manufacturer	Файл, содержащий текстовое название производителя

Файл или каталог	Описание
strings/<код языка>/manufacturer	Файл, содержащий текстовое название продукта
strings/<код языка>/serialnumber	Файл, содержащий серийный номер устройства
os_desc/	Каталог отвечает за поддержку функции «OS Description». В наших экспериментах она использоваться не будет. По умолчанию функция отключена.

Структура каталогов «hid.mouse» и «hid.keyboard»

Описание функций мышки и клавиатуры имеют одинаковую структуру, различающуюся содержимым.

Файл	Описание
protocol	Номер протокола: 1 – клавиатура, 2 – мышь
subclass	Номер подкласса. В нашем случае всегда будет 1, указывающий, что это подкласс Boot Interface. Другими словами, создаваемая нами клавиатура и мышь будут совместимы даже с BIOS.
report_length	Длина отчёта о состоянии устройства. Определяется дескриптором отчёта HID устройства. В нашем случае: 8 – клавиатура, 4 – мышь.
report_desc	Двоичный код дескрипторов отчёта HID-устройств. Далее рассмотрим их более подробно.

В наших экспериментах мы используем дескрипторы отчётов HID-устройств, которые кочуют по различным open-source проектам, но тем не менее показали себя вполне работоспособными.

Дескриптор отчёта HID-устройства клавиатура

```
0x05, 0x01,         // Usage Page (Generic Desktop Ctrls)
0x09, 0x06,         // Usage (Keyboard)
0xA1, 0x01,         // Collection (Application)
0x05, 0x07,         //   Usage Page (Kbrd/Keypad)
0x19, 0xE0,         //   Usage Minimum (0xE0)
0x29, 0xE7,         //   Usage Maximum (0xE7)
0x15, 0x00,         //   Logical Minimum (0)
0x25, 0x01,         //   Logical Maximum (1)
0x75, 0x01,         //   Report Size (1)
0x95, 0x08,         //   Report Count (8)
0x81, 0x02,         //   Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null
```

```
Position)
0x95, 0x01,         //   Report Count (1)
0x75, 0x08,         //   Report Size (8)
0x81, 0x03,         //   Input (Const,Var,Abs,No Wrap,Linear,Preferred State,No Null
Position)
0x95, 0x05,         //   Report Count (5)
0x75, 0x01,         //   Report Size (1)
0x05, 0x08,         //   Usage Page (LEDs)
0x19, 0x01,         //   Usage Minimum (Num Lock)
0x29, 0x05,         //   Usage Maximum (Kana)
0x91, 0x02,         //   Output (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null
Position,Non-volatile)
0x95, 0x01,         //   Report Count (1)
0x75, 0x03,         //   Report Size (3)
0x91, 0x03,         //   Output (Const,Var,Abs,No Wrap,Linear,Preferred State,No
Null Position,Non-volatile)
0x95, 0x06,         //   Report Count (6)
0x75, 0x08,         //   Report Size (8)
0x15, 0x00,         //   Logical Minimum (0)
0x25, 0x65,         //   Logical Maximum (101)
0x05, 0x07,         //   Usage Page (Kbrd/Keypad)
0x19, 0x00,         //   Usage Minimum (0x00)
0x29, 0x65,         //   Usage Maximum (0x65)
0x81, 0x00,         //   Input (Data,Array,Abs,No Wrap,Linear,Preferred State,No
Null Position)
0xC0,               // End Collection

// 63 bytes
```

Дескрипторы отчёта HID-устройства мышь

```
0x05, 0x01,         // Usage Page (Generic Desktop Ctrls)
0x09, 0x02,         // Usage (Mouse)
0xA1, 0x01,         // Collection (Application)
0x09, 0x01,         //   Usage (Pointer)
0xA1, 0x00,         //   Collection (Physical)
0x05, 0x09,         //     Usage Page (Button)
0x19, 0x01,         //     Usage Minimum (0x01)
0x29, 0x05,         //     Usage Maximum (0x05)
0x15, 0x00,         //     Logical Minimum (0)
0x25, 0x01,         //     Logical Maximum (1)
0x95, 0x05,         //     Report Count (5)
0x75, 0x01,         //     Report Size (1)
0x81, 0x02,         //     Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No
Null Position)
0x95, 0x01,         //     Report Count (1)
0x75, 0x03,         //     Report Size (3)
0x81, 0x01,         //     Input (Const,Array,Abs,No Wrap,Linear,Preferred State,No
Null Position)
0x05, 0x01,         //     Usage Page (Generic Desktop Ctrls)
0x09, 0x30,         //     Usage (X)
0x09, 0x31,         //     Usage (Y)
0x09, 0x38,         //     Usage (Wheel)
0x15, 0x81,         //     Logical Minimum (-127)
0x25, 0x7F,         //     Logical Maximum (127)
```

```
0x75, 0x08,         //     Report Size (8)
0x95, 0x03,         //     Report Count (3)
0x81, 0x06,         //     Input (Data,Var,Rel,No Wrap,Linear,Preferred State,No
Null Position)
0xC0,               //   End Collection
0xC0,               // End Collection

// 52 bytes
```

В тоже время дескрипторы, представленные в официальной документации [1].Appendix E: Example USB Descriptors for HID Class Devices для работы с Android не подошли.

Если вам захочется поэкспериментировать с дескрипторами отчёта, то можете воспользоваться официальной утилитой от USB-IF, либо онлайн-парсером от Frank Zhao.

Структура каталога mass_storage.0

В отличие от клавиатуры и мыши, каталог, описывающий функцию флешки, имеет совершенно другую структуру

Файл	**Описание**
stall	Файл-флаг, разрешающий (значение «1») остановку массовых конечных точек (bulk endpoints). По умолчанию должен быть включён. Отключать следует только при возникновении проблем.
/lun.<число>/	Подкаталог с настройками конкретной логической единицы хранения (LUN).
/lun.<число>/file	Файл, содержащий имя файла образа флешки.
ro	Файл-флаг, устанавливающий возможность записи в хранилище. 0 – запись возможна, 1 – запрещена (read-only).
removable	Файл-флаг, указывающий на то, что устройство следует интерпретировать как отчуждаемое. 1 – да, 0 – нет.
cd	Файл-флаг, указывающий на то, что устройство следует интерпретировать CD-ROM. 1 – да, 0 – нет.
nofua	Файл-флаг, указывающий на то, что устройство должно поддерживать функцию SCSI FUA. 1 – да, 0 – нет.

Устройства /dev/hidg0 и /dev/hidg1

После того как в активном профиле USB-порта активированы функции клавиатуры и мыши, в системе появятся два новых устройства /dev/hidg0 и /dev/hidg1. Первое

соответствует клавиатуре, а второе — мыши. Посылая отчёты в эти устройства, мы можем имитировать нажатия кнопок на клавиатуре или движения мыши.

Отключение функций клавиатуры или мышки в активном USB профиле приведёт к тому, что эти устройства перестанут существовать. Что такое отчёты, и как их формировать, мы поговорим далее.

Дополнительные скрипты для настройки USB-порта в смартфоне

Закрывая тему настройки USB-порта в смартфоне, рассмотрим ещё ряд вспомогательных скриптов, которые позволят вам глубже поковыряться в этой теме.

Удаление профиля

Первый скрипт позволяет удалить ранее созданный профиль USB-порта. Название профиля должно передаваться в качестве первого позиционного параметра.
Примечание. Не удаляйте системный профиль «g1», смартфон может повиснуть.

```
#!/system/bin/sh
# remove_profile.sh


PROFILE_NAME="$1"
PROFILE_BASEPATH="/config/usb_gadget/"
PROFILE_PATH="${PROFILE_BASEPATH}${PROFILE_NAME}"

if [ -z "$PROFILE_NAME" ]; then
  echo "Specify the profile name: \n$(ls $PROFILE_BASEPATH)"
  exit 1
fi

# Stop profile
echo "" > ${PROFILE_PATH}/UDC

# Disable active functions
find ${PROFILE_PATH}/configs/ -type l -delete

# Remove config
find ${PROFILE_PATH}/configs/ -type d -delete

# Remove functions
find ${PROFILE_PATH}/functions/ -type d -delete

# Remove strings
find ${PROFILE_PATH}/strings/ -type d -delete

# Remove profile
rmdir "$PROFILE_PATH"
```

Получение сведений о существующих USB профилях

Следующий скрипт позволяет получить информацию об имеющихся профилях, включая полный перечень доступных функций и список активных функций в профиле.

```
#!/system/bin/sh
# get_profiles_info.sh

for dir in /config/usb_gadget/*/
do
  echo PROFILE_PATH=$dir
  cd ${dir}configs/
  echo CONFIG_PATH="${dir}configs/$(ls -1 | head -1)/"
  cd $dir
  if [ "$?" -ne "0" ]; then
    echo "Error. Unable to change dir to $dir... exit"
    exit 1
  fi
  echo UDC=$(cat UDC)
  find ./configs/ -type l -exec sh -c 'echo FUNCTIONS_ACTIVE=$(basename $(readlink
"$@"))' _ {} \;

  for f in `ls -1 ./functions/`;
  do
    echo FUNCTIONS=$f;
  done;

  cd ./strings/0x409/;
  for vars in *
  do
    echo ${vars}=$(cat $vars)
  done
  echo "=============="
 done
```

Добавление функций клавиатуры, мыши и флешки к системному профилю USB-порта

Как упоминалось ранее, настраивать USB-порт можно как путём добавления нового профиля, так и активируя функции в уже имеющемся профиле. Представленный ниже скрипт добавляет функции клавиатуры, мыши и флешки к системному USB-профилю.

```
#!/system/bin/sh
# enable_func_in_sys_profile.sh

PROFILE_NAME="g1"
PROFILE_BASEPATH="/config/usb_gadget/"
PROFILE_PATH="${PROFILE_BASEPATH}${PROFILE_NAME}"
PROFILE_CONFIG_NAME="b.1"
STORAGE_FILE_NAME="/data/data/com.termux/files/home/storage_file.img"
STORAGE_FILE_SIZE="100M"

KEYBOARD_REPORT_DESC="\
\\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\
\\x15\\x00\\x25\\x01\\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\
\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01\\x05\\x08\\x19\\x01\
\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\x03\\x95\\x06\
\\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\
\\x81\\x00\\xc0"

MOUSE_REPORT_DESC="\
\\x05\\x01\\x09\\x02\\xa1\\x01\\x09\\x01\\xa1\\x00\\x05\\x09\
\\x19\\x01\\x29\\x05\\x15\\x00\\x25\\x01\\x95\\x05\\x75\\x01\
\\x81\\x02\\x95\\x01\\x75\\x03\\x81\\x01\\x05\\x01\\x09\\x30\
\\x09\\x31\\x09\\x38\\x15\\x81\\x25\\x7F\\x75\\x08\\x95\\x03\
\\x81\\x06\\xc0\\xc0"


# create keyboard function
mkdir -p "$PROFILE_PATH/functions/hid.keyboard"
cd "$PROFILE_PATH/functions/hid.keyboard"
if [ $? -eq 0 ]; then
    echo 1 > protocol # Keyboard
    echo 1 > subclass # Boot Interface Subclass
    echo 8 > report_length
    echo -ne $KEYBOARD_REPORT_DESC > report_desc
else
    echo "keyboard function create fail"
    exit 1
fi

# create mouse function
mkdir -p "$PROFILE_PATH/functions/hid.mouse"
cd "$PROFILE_PATH/functions/hid.mouse"
if [ $? -eq 0 ]; then
    echo 2 > protocol # Mouse
    echo 1 > subclass # Boot Interface Subclass
    echo 4 > report_length
```

```
    echo -ne $MOUSE_REPORT_DESC > report_desc
else
    echo "mouse function create fail"
    exit 1
fi

# create mass_storage image file
if [ ! -f "$STORAGE_FILE_NAME" ]; then
    fallocate -l $STORAGE_FILE_SIZE $STORAGE_FILE_NAME
fi

# create mass_storage function
mkdir -p "$PROFILE_PATH/functions/mass_storage.0"
cd "$PROFILE_PATH/functions/mass_storage.0/lun.0"
if [ $? -eq 0 ]; then
    echo 1 >../stall
    echo 0 > cdrom
    echo 0 > removable
    echo 0 > ro
    echo 0 > nofua
    echo "$STORAGE_FILE_NAME" > file
else
    echo "mass_storage function create fail"
    exit 1
fi

# Disable current profile
echo "" > $PROFILE_PATH/UDC

# Enable keyboard, mouse and mass storage functions
cd "$PROFILE_PATH/configs/$PROFILE_CONFIG_NAME/"
if [ $? -eq 0 ]; then
    ln -s "$PROFILE_PATH/functions/hid.keyboard" ./
    ln -s "$PROFILE_PATH/functions/hid.mouse" ./
    ln -s "$PROFILE_PATH/functions/mass_storage.0" ./
else
    echo "functions enable fail"
    exit 1
fi

# Enable current profile
ls /sys/class/udc > $PROFILE_PATH/UDC

exit 0
```

Отключение функций клавиатуры, мыши и флешки в системном USB-профиле

Это скрипт антагонист предыдущему. Он отключает (но не удаляет) ранее созданные функции в системном профиле.

```
#!/system/bin/sh
# disable_func_in_sys_profile.sh

PROFILE_NAME="g1"
```

```
PROFILE_BASEPATH="/config/usb_gadget/"
PROFILE_PATH="${PROFILE_BASEPATH}${PROFILE_NAME}"
PROFILE_CONFIG_NAME="b.1"

# Stop profile
echo "" > "${PROFILE_PATH}/UDC"

rm "${PROFILE_PATH}/configs/$PROFILE_CONFIG_NAME/hid.keyboard"
rm "${PROFILE_PATH}/configs/$PROFILE_CONFIG_NAME/hid.mouse"
rm "${PROFILE_PATH}/configs/$PROFILE_CONFIG_NAME/mass_storage.0"

# Start profile
ls /sys/class/udc > "${PROFILE_PATH}/UDC"

exit 0
```

Использование USB-порта в Android

Разбираемся с работой клавиатуры

USB-клавиатура при изменении своего состояния (нажатии или отпуске клавиши) посылает отчёт следующего формата ([1].B.1 Protocol 1 (Keyboard)):

Байт	**Назначение**
0	Модификатор
1	Резерв
2	Код клавиши 1
3	Код клавиши 2
4	Код клавиши 3
5	Код клавиши 4
6	Код клавиши 5
7	Код клавиши 6

Модификаторы, хранящиеся в Байте 0 отчёта, имеют формат ([1].8.3 Report Format for Array Items):

Бит	Назначение
0	LEFT CTRL
1	LEFT SHIFT
2	LEFT ALT
3	LEFT GUI
4	RIGHT CTRL
5	RIGHT SHIFT
6	RIGHT ALT
7	RIGHT GUI

В байтах (2-7) указываются коды нажатых клавиш, которые определены в спецификации [2].Keyboard/Keypad Page (0x07). Из-за большого объёма данных коды всех клавиш приводить здесь не будем, ограничимся лишь фрагментом (Рис. 4), необходимым для понимания принципа работы.

Usage ID	Usage Name	Usage Type	AT-101	PC-AT	Mac	Unix	Boot
00-00	*Reserved*						
01	Keyboard ErrorRollOver[1]	Sel	*N/A*	✓	✓	✓	4/101/104
02	Keyboard POSTFail[1]	Sel	*N/A*	✓	✓	✓	4/101/104
03	Keyboard ErrorUndefined[1]	Sel	*N/A*	✓	✓	✓	4/101/104
04	Keyboard a and A[2]	Sel	31	✓	✓	✓	4/101/104
05	Keyboard b and B	Sel	50	✓	✓	✓	4/101/104
06	Keyboard c and C[2]	Sel	48	✓	✓	✓	4/101/104
07	Keyboard d and D	Sel	33	✓	✓	✓	4/101/104
08	Keyboard e and E	Sel	19	✓	✓	✓	4/101/104
09	Keyboard f and F	Sel	34	✓	✓	✓	4/101/104

Рис. 4

Сама спецификация в открытом доступе, так что вы легко сможете её найти. Кроме официальной спецификации, можете посмотреть GitHub-репозиторий tmk_keyboard, содержащий визуальную карту кодов кнопок. На первых этапах он, наверное, будет более понятным.

Со справочной информацией покончено, перейдём к анализу принципа работы. Как вы скорее всего уже заметили, USB-клавиатура может обрабатывать одновременно нажатие 6 клавиш, а с учётом клавиш модификаторов это количество увеличивается до 14. В этой связи протокол её работы ([1].Appendix C: Keyboard Implementation) имеет определённую специфику: передаются только изменения состояния клавиатуры. Другими словами, если пользователь нажал кнопку, изменение произошло, если удерживает, то ничего не меняется, если отпустил, то опять изменение.

Чтобы с этим разобраться рассмотрим ряд примеров:

1. В скрипте, имитирующем нажатие кнопки «a», мы имитировали действия пользователя, при котором он сначала нажал кнопку «a», а потом отпускал все нажатые кнопки. По факту мы посылали в клавиатуру два отчёта:
    - 00 00 04 00 00 00 00 00 – нажатие кнопки «a»;
    - 00 00 00 00 00 00 00 00 – отпуск всех ранее нажатых кнопок.
2. Теперь усложним. Допустим мы хотим имитировать ввод заглавной «A». Для этого нам нужно послать отчёт, содержащий код буквы «a» с активным модификатором Left Shift (хотя можно выбрать и Right Shift). Отчёт для ввода заглавной «A» будет выглядеть так:
    - 02 00 04 00 00 00 00 00 – ввод «A».

3. Что будет, если просто пошлём отчёт в нажатой «A» и больше ничего слать не будем? А произойдёт следующие. ПО компьютера, к которому подключён смартфон подождёт 0,5 сек, а затем будет генерировать нажатия «A» и будет это происходить до тех пор, пока клавиатура не отправит код отпуска всех клавиш.
4. Давайте теперь разберёмся, что значит «отпустить кнопку» и «отпустить все кнопки». Понять это можно на примере. Пользователь нажимает кнопку «a», клавиатура посылает отчёт «00 00 04 00 00 00 00 00». Затем, не отпуская её он нажимает «m» – тогда будет послан код «00 00 04 10 00 00 00 00». Теперь он, продолжая удерживать «m» отпускает «a» – будет послан отчёт «00 00 10 00 00 00 00 00». То есть, код «a» (04) пропал, а его место занял код «m» (10). Когда он отпустит «m» будет послан код «00 00 00 00 00 00 00 00». Таким образом, кода отпуска какой-то конкретной клавиши не существует. Зато есть код, обозначающий, что все клавиши отпущены.

Закончим с клавиатурой, рассмотрев скрипт, вызывающий блокнот и печатающий в нём «Hello World!!».

```
#!/system/bin/sh
# print_hello_world.sh

echo "\x08\x00\x15\x00\x00\x00\x00\x00" > /dev/hidg0 # press Win+R
sleep 1s # wait until window appears
echo "\x00\x00\x00\x00\x00\x00\x00\x00" > /dev/hidg0 # release key
echo "\x00\x00\x11\x00\x00\x00\x00\x00" > /dev/hidg0 # press n
echo "\x00\x00\x12\x00\x00\x00\x00\x00" > /dev/hidg0 # press o
echo "\x00\x00\x17\x00\x00\x00\x00\x00" > /dev/hidg0 # press t
echo "\x00\x00\x08\x00\x00\x00\x00\x00" > /dev/hidg0 # press e
echo "\x00\x00\x13\x00\x00\x00\x00\x00" > /dev/hidg0 # press p
echo "\x00\x00\x04\x00\x00\x00\x00\x00" > /dev/hidg0 # press a
echo "\x00\x00\x07\x00\x00\x00\x00\x00" > /dev/hidg0 # press d
echo "\x00\x00\x37\x00\x00\x00\x00\x00" > /dev/hidg0 # press .
echo "\x00\x00\x08\x00\x00\x00\x00\x00" > /dev/hidg0 # press e
echo "\x00\x00\x1B\x00\x00\x00\x00\x00" > /dev/hidg0 # press x
echo "\x00\x00\x08\x00\x00\x00\x00\x00" > /dev/hidg0 # press e
echo "\x00\x00\x28\x00\x00\x00\x00\x00" > /dev/hidg0 # press Enter
echo "\x00\x00\x00\x00\x00\x00\x00\x00" > /dev/hidg0 # release key
sleep 3s # wait until window appears
echo "\x02\x00\x0B\x00\x00\x00\x00\x00" > /dev/hidg0 # press H
echo "\x00\x00\x08\x00\x00\x00\x00\x00" > /dev/hidg0 # press e
echo "\x00\x00\x0F\x00\x00\x00\x00\x00" > /dev/hidg0 # press l
echo "\x00\x00\x0F\x00\x00\x00\x00\x00" > /dev/hidg0 # press l
echo "\x00\x00\x12\x00\x00\x00\x00\x00" > /dev/hidg0 # press o
echo "\x00\x00\x2C\x00\x00\x00\x00\x00" > /dev/hidg0 # press Space
echo "\x02\x00\x1A\x00\x00\x00\x00\x00" > /dev/hidg0 # press W
echo "\x00\x00\x12\x00\x00\x00\x00\x00" > /dev/hidg0 # press o
echo "\x00\x00\x15\x00\x00\x00\x00\x00" > /dev/hidg0 # press r
echo "\x00\x00\x0F\x00\x00\x00\x00\x00" > /dev/hidg0 # press l
echo "\x00\x00\x07\x00\x00\x00\x00\x00" > /dev/hidg0 # press d
echo "\x02\x00\x1E\x00\x00\x00\x00\x00" > /dev/hidg0 # press !
echo "\x00\x00\x00\x00\x00\x00\x00\x00" > /dev/hidg0 # release key
```

Особенности использования скриптов для виртуальной клавиатуры

Человек, когда работает за компьютером, видит результаты своей работы, а скрипт, генерирующий нажатия клавиш – нет, это порождает ряд проблем:

1. Заранее неизвестно, какая клавиатурная раскладка активна на компьютере. Скрипт вместо английских букв может начать вводить русские и наоборот. Не известны также горячие кнопки переключения, это может быть как Shift+Alt, так и Ctrl+Shift или другие.
2. Если скрипт запускает программы, то им нужно время на отрисовку и получение фокуса ввода. Причём время запуска в общем случае может сильно варьироваться. Кнопки, нажатые до отрисовки требуемой программы, в лучшем случае могут уйти «в молоко», а худшем – натворить разных плохих дел.

Разбираемся с работой мыши

Мышь, как и клавиатура, на каждое изменение своего состояния посылает отчёт. Формат отчёта определён в спецификации [1].B.2 Protocol 2 (Mouse). После экспериментов над Android, превращённым в мышь, спецификацию отчёта пришлось дополнить, и в итоге она стала выглядеть так:

Байт	**Описание**
0	Состояние кнопок
1	Перемещение по X («+» – влево, «-» – вправо)
2	Перемещение по Y («+» – вверх, «-» – вниз)
3	Колёсико («+» – вверх, «-» – вниз)

Байты 1-3 могут принимать значения от -127 до 127. Значения кодируются дополнительным кодом, где 00 hex = 0 dec, 01 hex = 1 dec, 7F hex = 127, 81 hex = -127 dec, FF hex = -1 dec, 80 hex – игнорируется. Байт состояния кнопок мыши расшифровывается так (1 – означает нажатие кнопки, 0 – отпуск):

Бит	**Описание**
0	Левая кнопка
1	Правая кнопка
2	Средняя кнопка

Бит	Описание
3	Навигация назад
4	Навигация вперёд
5-7	Неопределенно

Фактическое перемещение курсора мыши осуществляется операционной системой в зависимости от текущего положения курсора и значений изменения координат, полученных из USB-отчёта от мыши (Рис. 5).

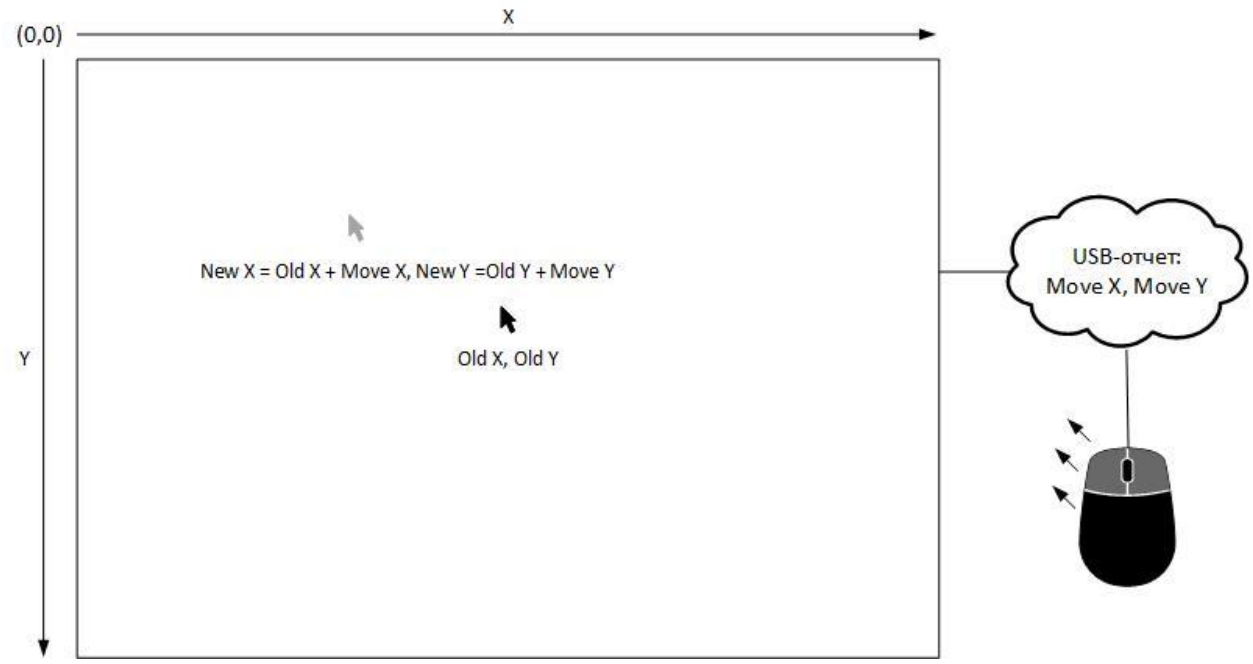

Рис. 5

Система координат начинается в верхнем левом углу экрана. Ось X растёт слева направо, ось Y сверху вниз. Новое положение курсора вычисляется по формулам:

$$NewX = OldX + MoveX$$

$$NewY = OldY + MoveY$$

где New X – новое значение координаты X курсора, Old X – предыдущие значение координаты X курсора, Move X – изменение значения координаты X курсора, полученное из USB отчёта мыши. Значения New Y, Old Y и Move Y – аналогичны, но для координаты Y. Если значение Move X отрицательное, то курсор смещается влево, если положительное, то вправо. Если значение Move Y отрицательное, то курсор подымается вверх, иначе смещается вниз.

Колёсико мышки обрабатывается по принципу: положительное значение – скроллинг вверх, отрицательное – скроллинг вниз. Величина скроллллинга определяется значением параметра «колёсико» в USB-отчёте.

С мышкой не всё так просто. Есть ряд нюансов, которые необходимо учитывать:

1. Получаемые координаты курсора мыши являются виртуальными. Чтобы операционная система могла перерисовать курсор в новом положении экрана, она переводит их в экранные значения с учётом разрешения экрана и работы функций акселерации, которая ускоряет перемещения экранного курсора в зависимости от интенсивности движения физической мыши. На более резкий сдвиг мыши экранный курс отодвинется дальше.
2. Мышь отправляет USB-отчёты с определённой частотой, и на перемещение экранного курсора также тратится время. Это может привести к неожиданным последствиям. Рассмотрим скрипт, который должен нарисовать в MS Paint квадрат:

```
#!/system/bin/sh
# mouse_draw_rect.sh

echo "\x01\x00\x00\x00" > /dev/hidg1 # Press LMB
echo "\x01\x7F\x00\x00" > /dev/hidg1 # Move MAX Right
echo "\x01\x01\x00\x00" > /dev/hidg1 # Move MIN Right
echo "\x01\x00\x00\x00" > /dev/hidg1 # Press LMB
echo "\x01\x00\x7F\x00" > /dev/hidg1 # Move MAX Down
echo "\x01\x00\x01\x00" > /dev/hidg1 # Move MIN Down
echo "\x01\x00\x00\x00" > /dev/hidg1 # Press LMB
echo "\x01\x81\x00\x00" > /dev/hidg1 # Move MAX Left
echo "\x01\xFF\x00\x00" > /dev/hidg1 # Move MIN Left
echo "\x01\x00\x00\x00" > /dev/hidg1 # Press LMB
echo "\x01\x00\x81\x00" > /dev/hidg1 # Move MAX Up
echo "\x01\x00\xFF\x00" > /dev/hidg1 # Move MIN Up
echo "\x00\x00\x00\x00" > /dev/hidg1 # Release LMB
```

Здесь сначала нажимается левая кнопка мыши, затем курсор с зажатой кнопкой двигается вправо, затем вниз, потом вверх, где и отпускается кнопка. При этом между изменениями движения курсора вставлен отчёт, который просто удерживает нажатой левую кнопку мыши. Если этого не делать, то в некоторых случаях вместо ожидаемого квадрата на экране можно увидеть треугольник.

Имитация действий человека и обнаружение ботов

Рассмотрим ряд признаков, которые позволят отличить управление клавиатурой и мышью реальным человеком от управления этими же устройствами ботом, использующим Android как средство ввода.
Поведенческие признаки:

1. Послав отчёты «02 00 04 00 00 00 00 00» и «00 00 00 00 00 00 00 00» бот обеспечит ввод одной заглавной «A». Но клавиатура, управляемая человеком, посылала бы другие коды. Вначале был бы послан код нажатия Left Shift – «02 00 00 00 00 00 00 00», затем при нажатом Left Shift была бы нажата «a» – код «02 00 04 00 00 00 00 00», потом отпущена «a» – код «02 00 00 00 00 00 00 00», а затем отпущен Left Shift – «00 00 00 00 00 00 00 00».
2. Кроме того, у всех людей разные размеры ладоней, моторика пальцев и привычки. Временные задержки между нажатием и отпуском клавиш, относительные задержки между вводом различных букв, предпочтения по используемым модификаторам образуют уникальный клавиатурный подчерк конкретного человека.
3. Мышь, управляемая человеком, отличается от мыши, управляемой ботом. Реальная мышь обычно генерирует гигантское количество отчётов на малейшее перемещение. Причём в самих отчётах изменение координат происходит на 1 или чуть больше. Бот же будет генерировать небольшое число отчётов с большим изменением координат.
4. Человек практически не в состоянии провести курсор по прямой, будут хоть и небольшие, но биения в обеих координатах. Бот же в свою очередь будет практически всегда перемещать курсор по прямым линиям.

Аппаратные признаки:

1. Кроме поведенческих признаков ботов можно выявлять и по аппаратным особенностям, анализируя обмен USB-пакетами. В ходе экспериментов установлено, что Android имеет определённые ограничения на имитацию USB-устройств. Например, полностью сымитировать существующую игровую мышь на нём невозможно. Реальная мышь в дескрипторах устройства сигнализирует о работе в режиме USB 1.1, в то время как Android всегда отдаёт USB 2.0. Кроме этого есть ещё множество других демаскирующих потенциального бота признаков.

Создание родного (native) приложения

В этой статье повсеместно используются скрипты, и у вас может сложиться мнение, что, кроме как из termux, взаимодействовать с виртуальной клавиатурой и мышью никак нельзя, но это не так. Как уже отмечалось в начале статьи, из родного API Andoird можно выполнять Linux команды. Для наших целей нужно выполнить команду «su», что приведёт к формированию оболочки. Затем, используя перенаправление потоков ввода-вывода, ей можно подавать на вход любые другие команды, включая скрипты, и получать результаты их выполнения. Поэтому вы можете смело брать написанный здесь код и выполнять его из своего приложения. Пример того как работать с «su» в Android, вы можете найти в этой статье.

Все скрипты, приведённые в этой статье, являются свободным ПО и могут использоваться без каких-либо ограничений.

Сценарии использования

Мы научились нажимать смартфоном кнопки клавиатуры, шевелить курсором мыши и получили новую флешку, что с этим всем делать? А применений нашему комбайну на самом деле очень много:

1. Для системных администраторов:
   - Смартфон, как портативное устройство ввода (клавиатура, мышь).
     Особенно актуальным будет данный сценарий при обслуживании техники вне офиса, например, банкоматов, платёжных терминалов, инфокиосков и т.д.
   - Многофункциональный LiveUSB.
     На смартфоне можно создать несколько образов загрузочных флешек под различные задачи. Затем по необходимости, можно выбирать нужный образ простым изменением настроек в скриптах переконфигурирования USB-интерфейса.
2. Автоинсталлятор.
   - С помощью смартфона можно проводить автоинсталляцию ПО. Для этого необходимые дистрибутивы записываются на флешку, а для клавиатуры и мыши пишутся скрипты, управляющие процессом инсталляции.
3. Для разработчиков:
   - Автотестер.
     С помощью смартфона, имитируя действия клавиатуры и мыши, можно выполнять тестирование разрабатываемого ПО.
4. Для обычных людей
   - Имитатор работы.
     При работе на удалёнке некоторые компании проверяют, чтобы человек постоянно что-то делал за компьютером. С помощью смартфона можно имитировать пользовательскую активность, не дающую, например, мессенджерам перейти в статус «Отошёл».

Инструменты исследования

В ходе исследования очень помогли инструменты:

1. Wireshark с модулем USBCap. Знаменитый сниффер, который может использоваться и для отладки USB-устройств. С его помощью в режиме реального времени можно увидеть весь информационный обмен между компьютером и устройством, включая USB дескрипторы, отчёты и прочее.
2. Python библиотека hid-tools. Имеет целую коллекцию инструментов для работы с USB HID устройствами. Основной изюминкой библиотеки является возможность получения от устройства данных (дескрипторов, отчётов) в понятном для человека виде.

Список источников

1. Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID) Firmware Specification—5/27/01 Version 1.11
2. HID Usage Tables FOR Universal Serial Bus (USB) Version 1.5