

Mobile_logs

Лог

Логи – это записи либо сообщения в виде текста. У нас в этом тексте записываются все действия пользователя или как отвечает система на действия пользователя, соответственно, вся та информация, что вы делаете, куда нажимаете на самом устройстве, в приложении – всё это пишется в логи.

Логирование

Логированием называют запись логов. Оно позволяет ответить на вопросы, что происходило, когда и при каких обстоятельствах. Без логов сложно понять, из-за чего появляется ошибка, если она возникает периодически и только при определенных условиях. Чтобы облегчить задачу администраторам и программистам, в лог записывается информация не только об ошибках, но и о причинах их возникновения.

Уровни логирования

Error

На этом уровне информируются ошибки работы системы.

Записи этого уровня требуют быстрого вмешательства разработчика — на такие ошибки нужно реагировать максимально быстро.

Например, `SpannableStringBuilder` – это ошибка приложения, которая говорит нам о том, что текстовое поле, то есть `Span`, наш элемент, он не может быть нулевым либо пустым.

“ `SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length` ”

Второй вариант – это системная ошибка `ZeroHung`. Данная ошибка говорит о том, что у нас происходит утечка памяти, она может быть как от какого-то действия с приложением, так и от самого приложения.

“ `[ZeroHung]zrhung_get_config: Get config failed for wp[0x0008]]` ”

Warning

На этом уровне отображаются записи, сообщающие о каком-то неожиданном поведении, требующем внимания, или о ситуации, которая незнакома системе.

Например, ошибка из приложения: мы пытаемся декодировать видео в нужное нам качество, в нужный формат, и у нас на этом происходит ошибка.

“ [OMX.hisi.video.decoder.avc] setting nBufferCountActual to 16 failed: -2147483648 “

Второй вариант, например, BroadcastQueue. Это ошибка системная, ошибка работы какого-то виджета на вашем устройстве.

“ BroadcastQueue: Permission Denial: broadcasting Intent ”

Info

Это уровень логов, на котором нам приходят записи чисто информационного характера о работе системы. Например, в этот уровень будут приходить ваши запросы, которые отправляют приложения на сервер.

Например, такое сообщение об уровне заряда батареи на устройстве

“ APwBatteryMonitor: screen off start battery: 100 ”

А это сообщение говорит о том, что экран устройства был выключен

“ HwBatteryService: intent = Intent { act=android.intent.action.SCREEN_OFF flg=0x58200010 } ”

Ещё в логи этого уровня входят запросы от клиента на сервер: хедеры, тело запросов, которые отправляет клиент, и ответы сервера.

“ okhttp.OkHttpClient: <-- 200 https://domainname/api/v1/smith/deals (1691ms)

okhttp.OkHttpClient: server: nginx/1.15.9

okhttp.OkHttpClient: date: Thu, 23 Sep 2021 19:41:17 GMT

okhttp.OkHttpClient: content-type: application/json

okhttp.OkHttpClient: vary: Accept-Encoding

okhttp.OkHttpClient: strict-transport-security: max-age=15724800; includeSubDomains

okhttp.OkHttpClient: {"key":{"key":value,"name":""},"key":value,"key":value}

okhttp.OkHttpClient: <-- END HTTP ”

Debug

Это уровень сообщений, в которых передаётся информация о процессах отладки или шагах работы крупных процессов.

Например, в записи ниже сказано, что пользователь нажимал на кнопку уменьшения или увеличения громкости:

*“ **MediaSessionService: dispatchVolumeKeyEvent** ”*

Ещё пример: если ваше приложение использует сокет-сессию, то на уровне **DEBUG** мы можем увидеть, когда сессия начинается и заканчивается:

*“ **b\$b: WebSocket connected** ”*

Verbose

Сообщения такого уровня уточняют или раскрывают действия. Verbose — уровень самого низкого приоритета. Выбирая такой уровень отображения логов, мы будем видеть записи и со всех предыдущих уровней.

Например, у нас есть служба управления окнами на экране приложения. И на уровне **Verbose** мы можем увидеть подробности её работы.

Открытие окна:

WindowManager: addWindow

Закрытие окна:

WindowManager: Removing Window

А меняя звук на устройстве, мы увидим, как растёт или падает значение:

AudioManager: getStreamVolume streamType: 3 volume: 10

Группы логов

Есть два вида логов:

- **Crash logs** — файл, в котором хранятся записи только об ошибках экстренного завершения программы — по-простому, когда приложение крашнулось.
- **Logs** — простые логи, или журнал событий. Это файл, в котором хранятся системные записи и ответы устройства на действие пользователя.

Программы для сбора логов

Android

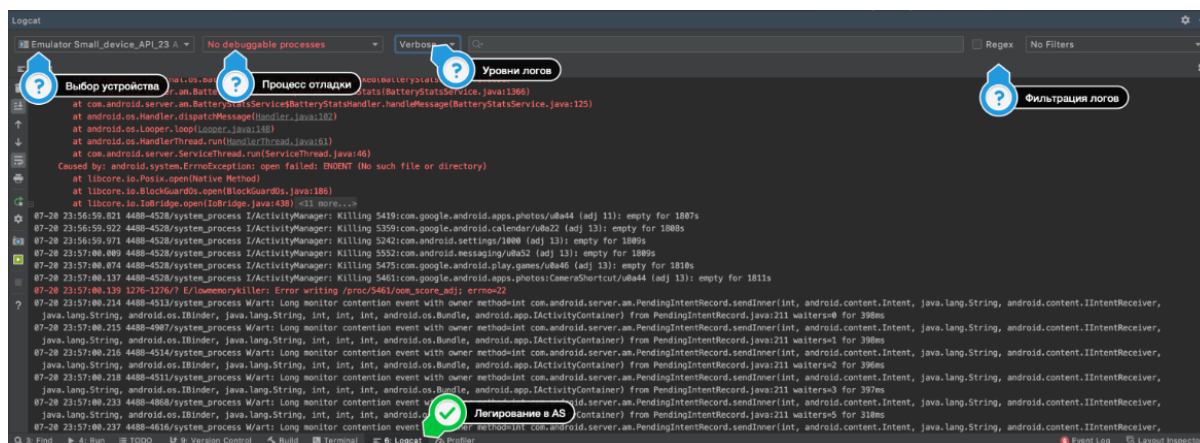
Android Studio - Logcat

Первый — **Logcat** в составе Android Studio, самый известный и широко используемый.

Для снятия логов нам необходимо перевести устройство в режим разработчика/отладки. Для этого нужно:

- найти в настройках номер нашего билда или ОС (в зависимости от устройства),
- около десяти раз нажать на эту информацию,

- при появлении сообщения о том, не хотим ли мы перевести устройство в режим разработчика, нажать «Ок».
1. Выбираем вкладку Logcat (переходим к сообщениям в реальном времени).
 2. В окошке выбираем телефон, с которого снимаем логи.
 3. На этой вкладке выбираем логи определённого приложения. Если нужно снять вообще все логи со всех приложений и системы, эту вкладку стоит не трогать. Рядом с ней можно выбрать уровень логирования (вкладка Verbose на скрине).
 4. В поле поиска, где мы можем фильтровать выдачу, разрешено писать что угодно — от названия пакета до частей вроде fatal.



Terminal - Logcat

Второй способ — выгрузка логов с самого устройства. Кроме режима разработчика нам нужно подключить устройство к ПК через USB и установить ADB — Android Debug Bridge.

Открываем терминал и пишем две команды.

Первая — **adb devices** — показывает подключённые устройства, которые видит ADB. В терминале выглядит так:

```
-zsh
Last login: Wed Jun 29 01:18:34 on ttys001
o.nikitina@onikitina-new ~ % adb devices
List of devices attached
7BKDU18504001505      device

o.nikitina@onikitina-new ~ % adb -s 7BKDU18504001505 logcat
```

Вводим вторую команду — **adb -s название устройства logcat**, — которая запускает утилиту Logcat для конкретного устройства. В терминале в реальном времени будут поступать логи.

```
-zsh
06-29 01:19:59.845 1666 1666 D HwCustMobileSignalControllerImpl: updateDataType, mDataNetType: 19, isCState: true
06-29 01:19:59.848 1326 1746 E : [ZeroHung]zrhung_get_config: Get config failed for wp[0x0000]
06-29 01:20:00.006 1326 1326 V AlarmManager: Received TIME_TICK alarm; rescheduling
06-29 01:20:00.008 1326 1326 I HwAlarmManagerService: hwSetAlarm listenerTag: time_tick
06-29 01:20:00.021 1666 1666 W PanelView: set notification panel padding = 1638
06-29 01:20:00.021 1666 1666 W HwBackDropView: setAnimationParamInner 0.0 0
06-29 01:20:00.026 1666 1666 W BokehDrawable: drawScrim colorB: 0 0
06-29 01:20:00.030 1666 1666 E DateView: DateView, mCurrentTime: 1656454800030
06-29 01:20:00.033 1666 1666 I EventCenter: EventCenter Get :android.intent.action.TIME_TICK
06-29 01:20:00.034 1666 1666 W ClockView1: action android.intent.action.TIME_TICK
06-29 01:20:00.034 1666 1666 I chatty : uid=10039(com.huawei.HwMultiScreenShot) com.android.systemui identical 1 line
06-29 01:20:00.034 1666 1666 W ClockView1: action android.intent.action.TIME_TICK
06-29 01:20:00.039 1666 1666 W PanelView: set notification panel padding = 1638
06-29 01:20:00.040 1666 1666 W HwBackDropView: setAnimationParamInner 0.0 0
06-29 01:20:00.045 1666 1666 W BokehDrawable: drawScrim colorB: 0 0
06-29 01:20:00.050 1666 1666 I LocalCalendar: CalendarId: 0000LocalCalender Off
06-29 01:20:00.054 1666 1666 I chatty : uid=10039(com.huawei.HwMultiScreenShot) com.android.systemui identical 1 line
06-29 01:20:00.058 1666 1666 I LocalCalendar: CalendarId: 0000LocalCalender Off
06-29 01:20:00.064 1666 1666 W PanelView: set notification panel padding = 1638
06-29 01:20:00.064 1666 1666 W HwBackDropView: setAnimationParamInner 0.0 0
06-29 01:20:00.065 1888 1888 I HwLauncher: Model onReceive intent=Intent { act=android.intent.action.TIME_TICK flg=0x50200014 hwFlg=0x900 (has extras) }
06-29 01:20:00.065 1888 1888 I HwLauncher: Model onReceive user=UserHandle{0}
06-29 01:20:00.068 1666 1666 W BokehDrawable: drawScrim colorB: 0 0
06-29 01:20:00.074 1666 1922 I KeyguardStatusView: Runnable for refresh
06-29 01:20:00.102 1326 9458 V BroadcastQueue: Finished with ordered broadcast BroadcastRecord{ef08532 u-1 android.intent.action.TIME_TICK}
```

1. В первом столбце — дата и время поступления записи.
2. Во втором — обозначения уровней логирования. Например, D — это Debug.
3. В третьем показываются названия инструмента, утилиты, пакета, от которых поступает сообщение, а также расшифровка того, что вообще происходит.

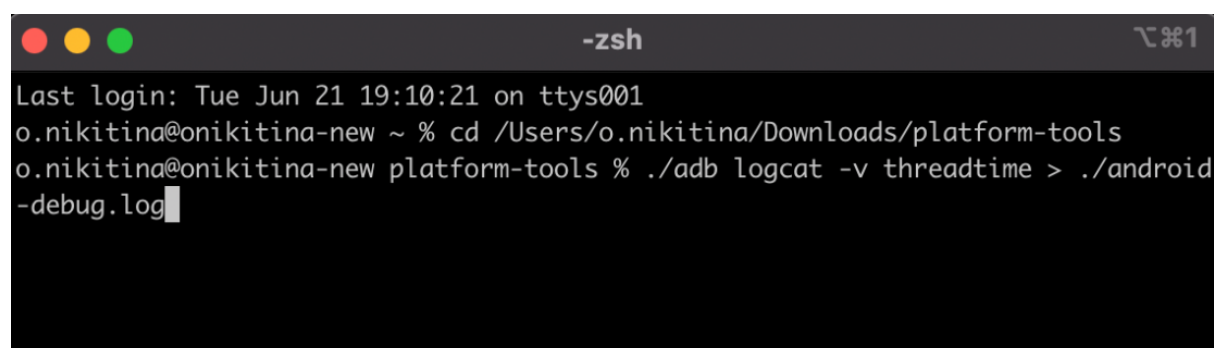
SDK Platform Tools

Третий инструмент — SDK Platform Tools. Процесс его установки практически аналогичен предыдущим двум:

- переводим телефон в режим разработчика,
- подключаем к ПК по USB,
- скачиваем на ПК папку SDK PT (под свою ОС),
- открываем папку SDK PT в терминале.

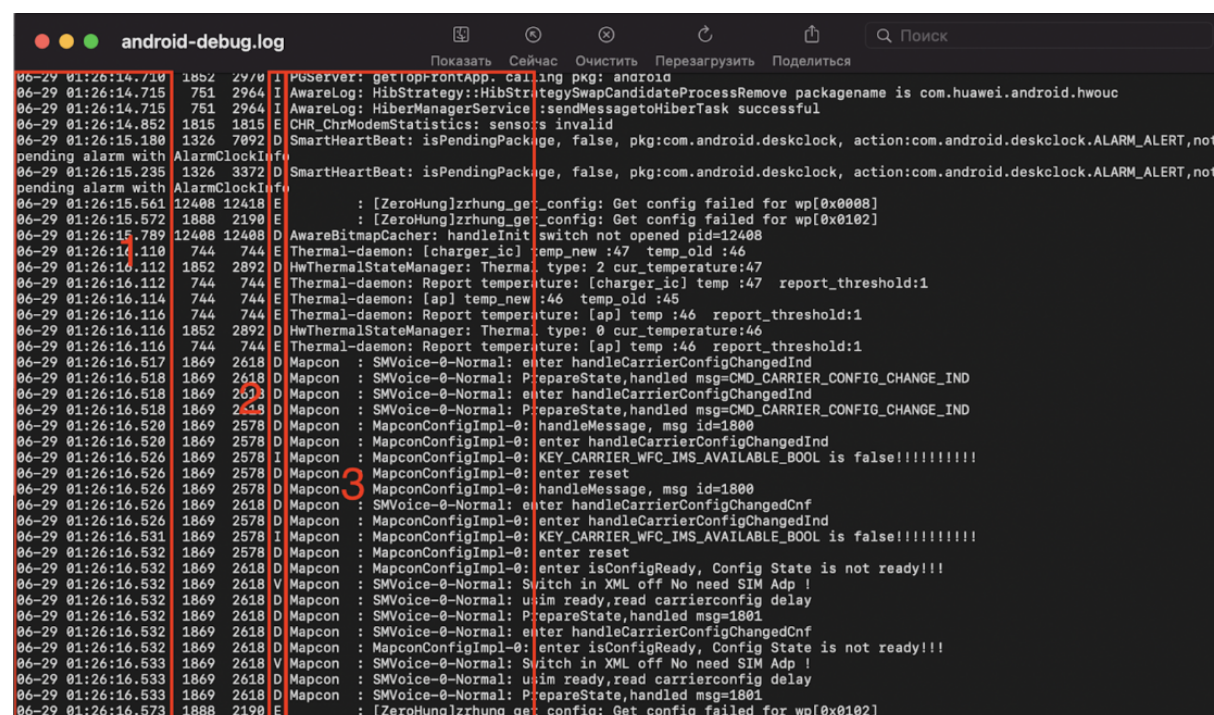
Теперь пишем команду `./adb logcat -v threadtime > ./android-debug.log`.

В терминале это выглядит так:



```
-zsh
Last login: Tue Jun 21 19:10:21 on ttys001
o.nikitina@onikitina-new ~ % cd /Users/o.nikitina/Downloads/platform-tools
o.nikitina@onikitina-new platform-tools % ./adb logcat -v threadtime > ./android-debug.log
```

Прерываем выполнение команды. Лог добавляется в папку. Открываем.



```
06-29 01:26:14.710 1852 2970 I PGServer: getIopFrontApp. calling pkg: android
06-29 01:26:14.715 751 2964 I AwareLog: HibStrategy::HibStrategySwapCandidateProcessRemove packagename is com.huawei.android.hwouc
06-29 01:26:14.715 751 2964 I AwareLog: HibManagerService :sendMessageToHiberTask successful
06-29 01:26:14.852 1815 1815 E CHR_ChrModemStatistics: sensors invalid
06-29 01:26:15.180 1326 7092 D SmartHeartBeat: isPendingPackage, false, pkg:com.android.deskclock, action:com.android.deskclock.ALARM_ALERT,not
pending alarm with
06-29 01:26:15.235 1326 3372 D SmartHeartBeat: isPendingPackage, false, pkg:com.android.deskclock, action:com.android.deskclock.ALARM_ALERT,not
pending alarm with
06-29 01:26:15.561 12408 12418 E : [ZeroHung]zrhung_get_config: Get config failed for wp[0x0008]
06-29 01:26:15.572 1888 2198 E : [ZeroHung]zrhung_get_config: Get config failed for wp[0x0102]
06-29 01:26:15.789 12408 12408 D AwareBitmapCacher: handleInit switch not opened pid=12408
06-29 01:26:16.118 744 744 E Thermal-daemon: [charger_ic] temp_new:47 temp_old:46
06-29 01:26:16.112 1852 2892 D HwThermalStateManager: Thermal type: 2 cur_temperature:47
06-29 01:26:16.112 744 744 E Thermal-daemon: Report temperature: [charger_ic] temp :47 report_threshold:1
06-29 01:26:16.114 744 744 E Thermal-daemon: [ap] temp_new:46 temp_old:45
06-29 01:26:16.116 744 744 E Thermal-daemon: Report temperature: [ap] temp :46 report_threshold:1
06-29 01:26:16.116 1852 2892 D HwThermalStateManager: Thermal type: 0 cur_temperature:46
06-29 01:26:16.116 744 744 E Thermal-daemon: Report temperature: [ap] temp :46 report_threshold:1
06-29 01:26:16.517 1869 2613 D Mapcon : SMVoice-0-Normal: enter handleCarrierConfigChangedInd
06-29 01:26:16.518 1869 2613 D Mapcon : SMVoice-0-Normal: PrepareState,handled msg=CMD_CARRIER_CONFIG_CHANGE_IND
06-29 01:26:16.518 1869 2613 D Mapcon : SMVoice-0-Normal: enter handleCarrierConfigChangedInd
06-29 01:26:16.518 1869 2613 D Mapcon : SMVoice-0-Normal: PrepareState,handled msg=CMD_CARRIER_CONFIG_CHANGE_IND
06-29 01:26:16.528 1869 2578 D Mapcon : MapconConfigImpl-0: handleMessage, msg id=1800
06-29 01:26:16.528 1869 2578 D Mapcon : MapconConfigImpl-0: enter handleCarrierConfigChangedInd
06-29 01:26:16.526 1869 2578 I Mapcon : MapconConfigImpl-0: KEY_CARRIER_WFC_IMS_AVAILABLE_BOOL is false!!!!!!!!!!!!
06-29 01:26:16.526 1869 2578 D Mapcon : MapconConfigImpl-0: enter reset
06-29 01:26:16.526 1869 2613 D Mapcon : SMVoice-0-Normal: enter handleCarrierConfigChangedCnf
06-29 01:26:16.526 1869 2578 D Mapcon : MapconConfigImpl-0: enter handleCarrierConfigChangedInd
06-29 01:26:16.531 1869 2578 I Mapcon : MapconConfigImpl-0: KEY_CARRIER_WFC_IMS_AVAILABLE_BOOL is false!!!!!!!!!!!!
06-29 01:26:16.532 1869 2578 D Mapcon : MapconConfigImpl-0: enter reset
06-29 01:26:16.532 1869 2613 D Mapcon : MapconConfigImpl-0: enter isConfigReady, Config State is not ready!!!
06-29 01:26:16.532 1869 2613 V Mapcon : SMVoice-0-Normal: Switch in XML off No need SIM Adp !
06-29 01:26:16.532 1869 2613 D Mapcon : SMVoice-0-Normal: uim ready,read carrierconfig delay
06-29 01:26:16.532 1869 2613 D Mapcon : SMVoice-0-Normal: PrepareState,handled msg=1801
06-29 01:26:16.532 1869 2613 D Mapcon : SMVoice-0-Normal: enter handleCarrierConfigChangedCnf
06-29 01:26:16.532 1869 2613 D Mapcon : MapconConfigImpl-0: enter isConfigReady, Config State is not ready!!!
06-29 01:26:16.533 1869 2613 V Mapcon : SMVoice-0-Normal: Switch in XML off No need SIM Adp !
06-29 01:26:16.533 1869 2613 D Mapcon : SMVoice-0-Normal: uim ready,read carrierconfig delay
06-29 01:26:16.533 1869 2613 D Mapcon : SMVoice-0-Normal: PrepareState,handled msg=1801
06-29 01:26:16.573 1888 2198 E : [ZeroHung]zrhung_get_config: Get config failed for wp[0x0102]
```

1. В первом столбце — дата и время,

2. Во втором — уровни логов,
3. В третьем — указание на то, от какой части системы поступают данные, лог и его расшифровка/подробности

Очень похоже на предыдущий терминал, но файл обновляется, пока в терминале действует команда.

iOS

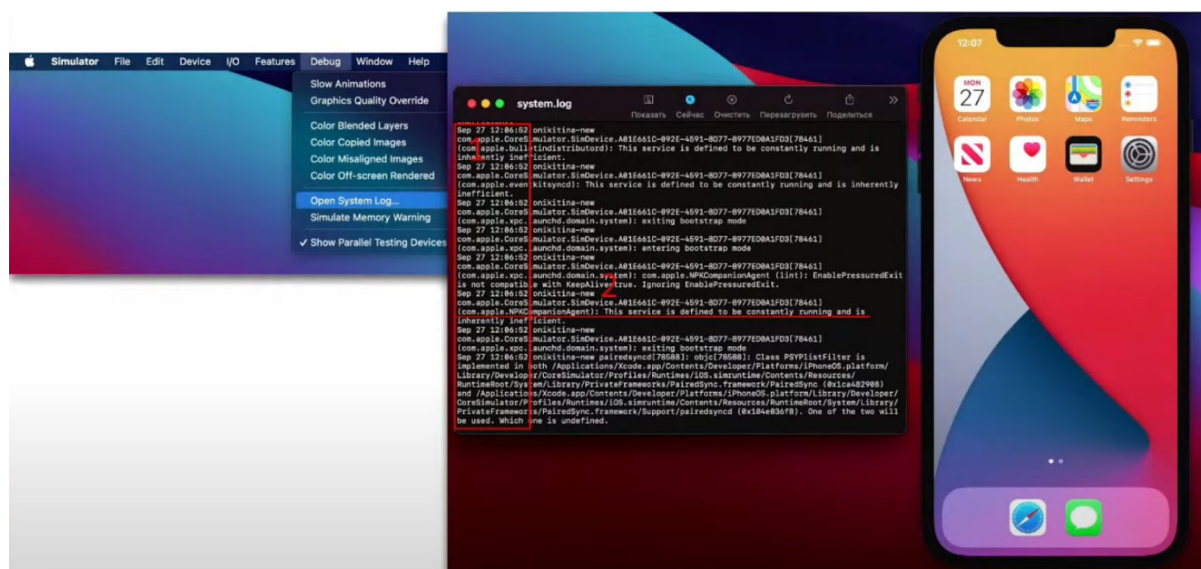
xCode

— интегрированная среда разработки (IDE), в которую встроен нужный нам инструмент Simulator.

Как использовать инструмент:

1. Устанавливаем xCode.
2. В системной строке нажимаем xCode → Open Developer Tools → Simulator.
3. Устанавливаем приложение.
4. В самом симуляторе выбираем Debug → Open System Log.

Мы будем видеть логи в реальном времени:



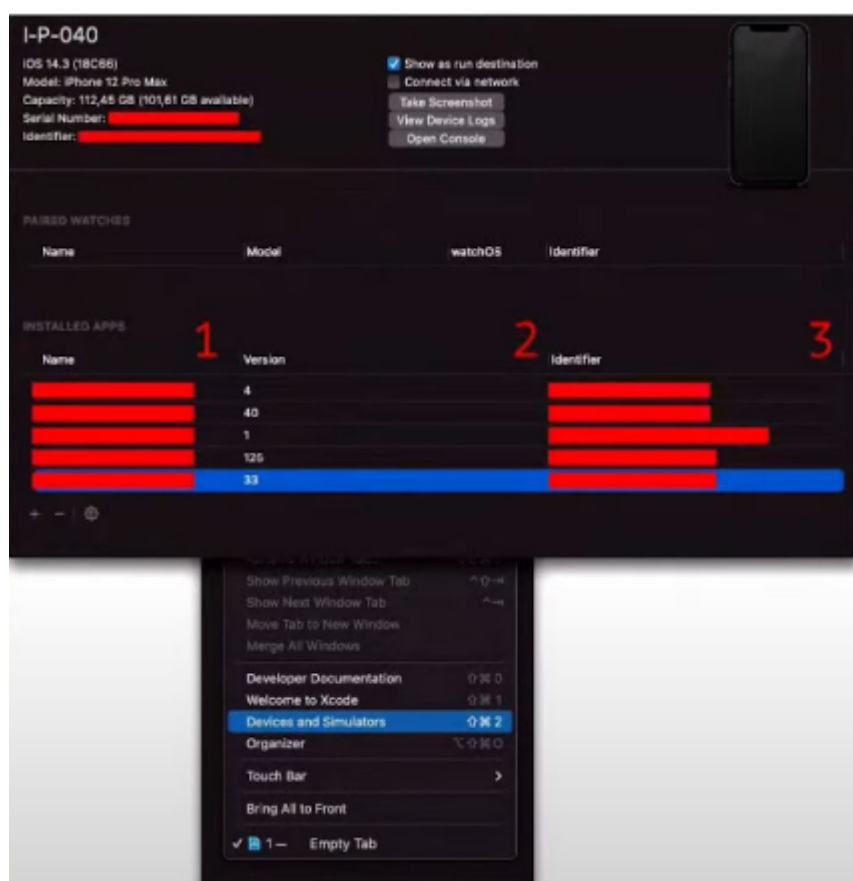
Есть дата и время (1) и данные (2) о том, с какого устройства снята информация: имя компьютера, элемент системы, с которого пришло сообщение, и его расшифровка.

Но первый способ работает только с симуляторами. Если необходимо снимать логи с реального устройства, в этом может помочь раздел **Devices and**

Simulators.

Это Devices and Simulator. Устанавливаем xCode, подключаем устройство по USB, тут уже важно, чтобы было реальное устройство. В самом xCode открываем вкладку Window, там выбираем подвкладку Devices and Simulator. Нажимаем у устройства «Open Console». На панели видим название нашего устройства, какая у него операционная система, модель, и правее этой кнопки нам интересно «Open Console».

Под цифрой 1 мы видим все приложения, которые дополнительно установлены на наши устройства, в колонке под цифрой 2 – версия этого устройства, которую разработчик указывает. Третье пишется URL нашего устройства. То есть, например, у вашего разработчика, у вашей компании, у вашего приложения есть свой URL, по которому он ходит. Соответственно, здесь он отражен.

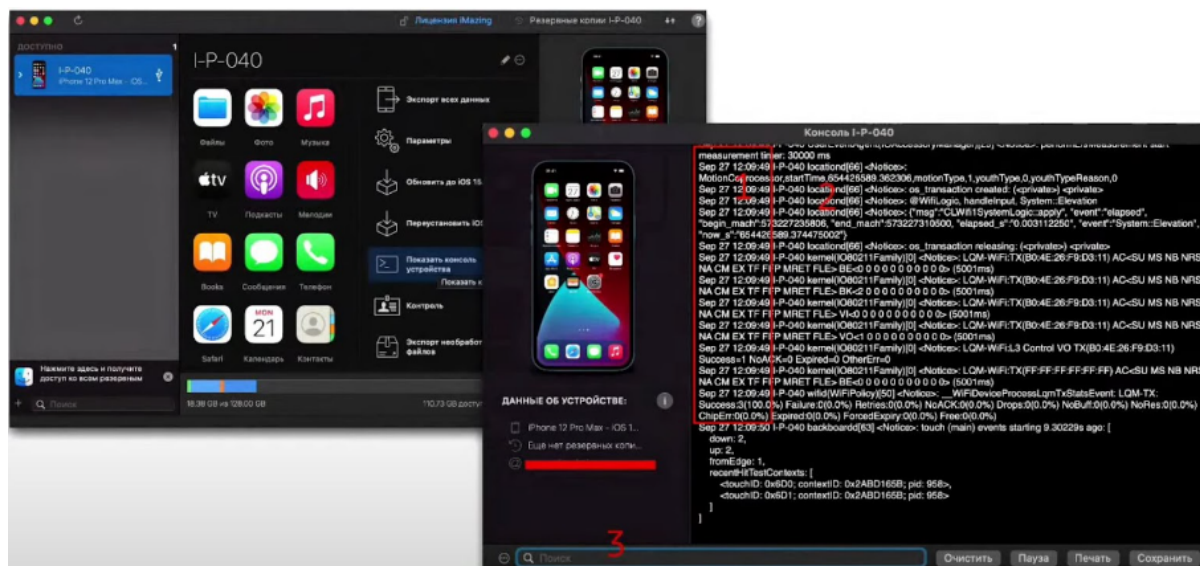


iMazing

Утилита iMazing поможет снимать iOS-логи для тех, у кого установлен Windows. Приложение платное, но часть функциональности доступна бесплатно.

Например, за снятие логов устройства платить не нужно.

В меню выбираем «Показать консоль устройства». В открывшемся окне приходят записи логов в реальном времени со всего устройства.



1. дата и время получения сообщения
2. имя телефона, информация, с какой части устройства пришло сообщение, и описание
3. поисковая строка для фильтрации выдачи

Структура файла логов мобильного устройства

Лог-файл — файл с последовательными событиями, которые осуществлены пользователем на интернет-ресурсе или сервере. Структура файла зависит от устройства и инструмента для снятия логов .

В файле, чаще всего, может содержаться следующая информация:

1. дата и время запроса
2. тип устройства
3. уровень лога
4. текст лога ошибки
5. название системы, отправляющей информацию в лог.

