Логи

Логи – это текстовые файлы, в которых записываются все действия пользователя. Например, какие кнопки он нажимает в приложении и как на это оно реагирует в ответ.

Записи в логах формируются в хронологическом порядке. Самая свежая — внизу.

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.

Логи на мобильных устройствах бывают нескольких уровней:

- ERROR
- WARN
- INFO
- DEBUG
- VERBOSE

Они представлены по уровню важности — от самого высокого к самому низкому, — и каждый следующий уровень включает в себя предыдущий. Например, VERBOSE содержит в себе логи всех остальных.

Примечание: уровни более применимы к логам на Android, потому что именно там такое разделение встречается чаще.

Рассмотрим подробнее каждый уровень:

1. **Error (ERROR)**

На этом уровне информируются ошибки работы системы.

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

Как пример, такая запись в логе:

“ SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero length ”

Это ошибка, в которой говорится, что строковый элемент span не может быть пустым.

Или вот:

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

Эта системная ошибка сообщает, что происходит утечка памяти при взаимодействии с каким-то элементом или приложением.

2. **Warning (WARN)**

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

Например, сообщение ниже — запись из тестового приложения:

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

Мы пытаемся декодировать запись в какой-то формат, но его нет. Ошибка сообщает о неуспешной попытке настройки видеоплеера в нужном формате.

Ещё пример:

“ BroadcastQueue: Permission Denial: broadcasting Intent ”

Эта системная ошибка говорит о сбое в работе одного из виджетов на устройстве.

3. **Info (INFO)**

На этот уровень приходят записи информационного характера, например о работе системы.

Допустим, 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/smth/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 ”

Такие записи могут помочь вам в понимании какого-то бага или в разборе задачи при условии, что вы не можете перехватить трафик или не знаете, какие запросы отправляются на бэкенд.

4. **Debug (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 ”

Сначала мы видим запись о самом факте нажатия на кнопку, далее оно расшифровывается подробнее:

{ action=ACTION_DOWN, keyCode=KEYCODE_VOLUME_UP }

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

“ b$b: WebSocket connected ”

5. **Verbose (VERBOSE)**

Сообщения такого уровня уточняют или раскрывают действия.

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

На этом уровне мы можем посмотреть системные подробности наших действий. Например, при включении геолокации в записи отобразится текущая геолокация.

GnssLocationProvider: reportLocation Location [...]

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

AudioManager: getStreamVolume streamType: 3 volume: 10

Каждое нажатие, то есть изменение звука, будет отражаться новым сообщением.

Verbose — уровень самого низкого приоритета. Выбирая такой уровень отображения логов, мы будем видеть записи и со всех предыдущих уровней.

Примечание: разработчики приложения самостоятельно покрывают действия логами, определяют уровни, а также какие сообщения какому из них соответствуют.

Инструменты для снятия логов: Android

Расскажем о трёх способах.

1. 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)
- несколько раз нажать на эту информацию
- при появлении сообщения о том, не хотим ли мы перевести устройство в режим разработчика, нажать «Ок»

Примечание: алгоритм может отличаться в зависимости от производителя устройства, потому что у многих из них свои надстройки на ОС Android.

Дальше подключаем устройство по USB к ПК и устанавливаем Android Studio.
Следующие шаги:

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.

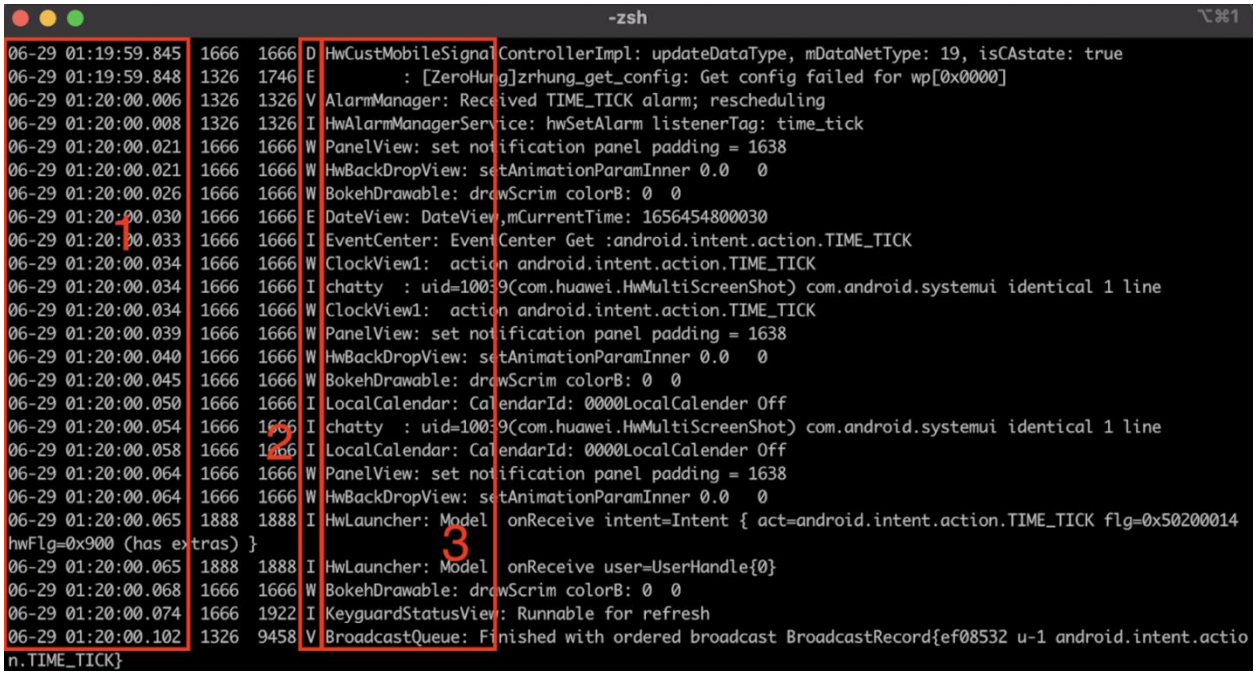
На экране будут видны логи с подключенного устройства.

2. 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.
Вводим вторую команду — **adb -s *название устройства* logcat**, — которая запускает утилиту Logcat для конкретного устройства. В терминале в реальном времени будут поступать логи.

Как их читать?



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.

3. 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.**

Прерываем выполнение команды (например, на Mac это Control+C). Лог добавляется в папку. Очень похоже на предыдущий терминал, но файл обновляется, пока в терминале действует команда.

Инструменты снятия логов: iOS

1. 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.

Подобное оформление логов мы уже где-то видели, но построение информации в выдаче немного отличается. Есть дата и время и данные о том, с какого устройства снята информация: имя компьютера, элемент системы, с которого пришло сообщение, и его расшифровка.

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

Записи можно отфильтровать по конкретному процессу (вашему приложению):

1. Устанавливаем xCode.

2. Подключаем устройство к ПК по USB.

3. Открываем xCode → Windows → Devices and Simulators.

Дальше нажимаем у устройства Open Console и видим панель с названием устройства, информацией о модели и ОС: все приложения, которые установлены на устройстве; версия устройства; пакет приложения устройства.

Логи поступают в реальном времени, но их удобно отслеживать. У нас есть три столбца:

1. «Время» — время поступления сообщения

2. «Процесс» — с какой части системы/приложения пришло сообщение

3. «Сообщение» — описание события, сервисная информация

В инструменте есть поиск для фильтрации выдачи. Ещё есть полезная кнопка «Приостановить» — она останавливает поток логов.

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

Ещё одно важное достоинство iMazing — возможность сохранять логи (по кнопке «Сохранить»).

Выходные данные фильтрации журнала

Журналы Android делятся на следующие 7 уровней：

- V: подробный (низкий приоритет) - (подробный: 2)
- D: Отладка - (Отладка: 3)
- I: Информация - (Информация: 4)
- W: Предупреждение - (Предупреждение: 5)
- E: Ошибка - (Ошибка: 6)
- F: серьезная ошибка - (фатальный: 7)
- S: Без звука (высший приоритет, никогда ничего не выводить) - (Без звука)

При отладке журналов мы можем управлять выводом журналов на разных уровнях.

Ниже приводится нормальный вывод журнала Android.

Журнал： 02-05 12:44:15.357 17533 17545 D ActivityThread: caller system = false

Формат журнала:<дата> <время> <PID> <TID> <уровень журнала> <тег тега журнала> <содержимое журнала>.

** - pid - это идентификатор процесса; tid - это идентификатор потока*