DOI: 10.15514/ISPRAS-2019-31(5)-2

Интроспекция конфигурации периферийных устройств эмулятора QEMU

Н.И. Фурсова, ORCID: 0000-0001-6817-4670 <natalia.fursova@ispras.ru>
П.М. Довгалюк, ORCID: 0000-0003-2483-5718 <pavel.dovgaluk@ispras.ru>

Новгородский Государственный университет имени Ярослава Мудрого
173003, Россия, Великий Новгород, ул. Санкт-Петербургская, д.41

Аннотация. QEMU – широко используемый и достаточно точный эмулятор, способный эмулировать десятки гостевых систем. Эмуляция системы предполагает настройку виртуальных устройств, которые в большом количестве поддержаны в QEMU, что влечет за собой очень длинную и запутанную строку запуска эмулятора. При использовании детерминированного воспроизведения ситуация усложняется не только дополнительными и не вполне очевидными параметрами, но и необходимостью синхронизации строк запуска записи и воспроизведения. Машины могут иметь разный набор устройств в зависимости от платформы и даже версии эмулятора. В статье рассматривается получение информации об устройствах эмулятора QEMU через QEMU Machine Protocol для использования этих данных в графическом интерфейсе. Графический интерфейс QemuGUI поддерживает полный цикл работы с эмулятором: создание и настройка виртуальной машины, запуск в обычном режиме и в режимах детерминированного воспроизведения, взаимодействие с машиной через монитор QEMU.

Ключевые слова: QEMU; GUI; QMP; графический интерфейс; эмулятор

Для цитирования: Фурсова Н.И., Довгалюк П.М. Интроспекция конфигурации периферийных устройств эмулятора QEMU. Труды ИСП РАН, том 31, вып. 5, 2019 г., стр. 25-36. DOI: 10.15514/ISPRAS-2019-31(5)-2

Благодарности: работа выполнена при поддержке гранта РФФИ (18-07-00900А) и в рамках выполнения государственного задания Минобрнауки России № 2.6146.2017/8.9.

Introspection of QEMU emulator peripherals configuration

N.I. Fursova, ORCID: 0000-0001-6817-4670 <natalia.fursova@ispras.ru>
P.M. Dovgalyuk, ORCID: 0000-0003-2483-5718 <pavel.dovgaluk@ispras.ru>

Yaroslav-the-Wise Novgorod State University
41, Sankt-Peterburgskaya st., Velikiy Novgorod, 173003, Russia

Abstract. QEMU is a widely used and fairly accurate emulator capable of emulating dozens of guest systems. Emulation of the system involves the configuration of virtual devices, which are supported in large numbers in QEMU, which entails a very long and complicated command line to start the emulator. When using deterministic replay, the situation is complicated not only by additional and not quite obvious parameters, but also by the need to synchronize recording and retrace launch command lines. Machines can have a different set of devices depending on the platform and even the version of the emulator. The article describes obtaining information about the devices of the QEMU emulator through the QEMU Machine Protocol for using this data in a graphical interface. The QemuGUI graphical interface supports the full cycle of work with the emulator: creating and configuring a virtual machine, starting in normal mode and in deterministic replay mode, interacting with the machine through a QEMU monitor.

Keywords: QEMU;, GUI; QMP; emulator

For citation: Fursova N.I., Dovgalyuk P.M. Introspection of QEMU emulator peripherals configuration. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 5, 2019, pp. 25-36 (in Russian). DOI: 10.15514/ISPRAS-2019-31(5)-2

Acknowledgements. The work was partially supported by the Ministry of Education and Science of Russia, research project No. 2.6146.2017/8.9, and by the Russian Foundation of Basic Research (research grant 18–07–00900 A).

1. Введение

Эмулятор QEMU [1, 2] является одним из передовых эмуляторов для разработчиков средств анализа кода. Интересен он тем, что является программным обеспечением с открытым исходным кодом и поддерживает большое количество архитектур гостевых систем.

В связи с тем, что QEMU эмулирует аппаратное обеспечение довольно точно, он позволяет указывать огромное количество настроек, описывающих устройства машины, которые необходимо передавать в командную строку запуска. Настройки эти разной степени сложности, а командная строка может раздуваться до невероятных размеров [3, 4].

Отдельно следует сказать о работе с детерминированным воспроизведением [5]. Детерминированное воспроизведение – это технология, позволяющая записывать и многократно воспроизводить сценарии работы эмулятора. При использовании детерминированного воспроизведения появляются дополнительные трудности, например, каждое блочное устройство должно быть оснащено прослойкой дополнительных параметров, что не всегда очевидно и также раздувает командную строку. Кроме того, пользователь может иметь необходимость пользоваться разными сборками эмулятора, что вынуждает его или постоянно заглядывать в командную строку (если он ее сохранил для многоразового использования) или поддерживать файлы журналов в строгом порядке.

Пример командной строки для записи сценария:

```
D:\\Program\\Qemu\\qemu-system-x86_64w.exe -m 512 -drive
file=F:\\temp.qcow2,if=none,id=img-direct0 -drive
driver=blkreplay,if=none,image=img-direct0,id=img-blkreplay0 -device
ide-hd,drive=img-blkreplay0,bus=ide.0 -icount
shift=7,rr=record,rrfile=F:/projects/1,rrsnapshot=first-
snapshot,rrperiod=5 -net none -qmp tcp:127.0.0.1:5500,server,nowait -
vnc 127.0.0.1:0 -drive
file=f:/DEBUG_VS2012x86/FV/ovmf.fd,if=pflash,snapshot=on -machine q35
-machine smm=on -vga std
```

То же самое, но без записи сценария:

```
D:\\Program\\Qemu\\qemu-system-x86_64w.exe -m 512 -hda D:\\image.qcow2
-net none -qmp tcp:127.0.0.1:5500,server,nowait -vnc 127.0.0.1:0 -
drive file=D:/longpath/ovmf.fd,if=pflash,snapshot=on -machine q35 -
machine smm=on -vga std
```

Запутаться в необходимых настройках может даже опытный пользователь, а найти ошибку – зачастую задача непростая.

Необходимо синхронизировать командные строки запусков записи и воспроизведения и удостовериться что при многократном запуске воспроизведения строка будет одинаковой, ведь в этом случае от перестановки мест слагаемых сумма может поменяться, например, если не указать шины, на которых будут устройства, и поменять местами устройства, QEMU распределит их самостоятельно и воспроизведение не сможет работать. Важно обеспечить детерминированность этого процесса.

Кроме того, справочный раздел QEMU (`-help`) также впечатляет своим размером и найти необходимую опцию не всегда легко и просто.

Все это вынуждает пользователей сохранять командные строки для запуска эмулятора для многоразового использования.

Отчасти именно сложность командной строки при использовании детерминированного воспроизведения сподвигла к разработке графического интерфейса для эмулятора QEMU – QemuGUI.

Цель QemuGUI – обеспечивать получение актуальной информации об устройствах QEMU и конструировать работоспособную командную строку. Под актуальной информацией понимается конфигурация устройств эмулятора и их свойства, полученные максимально автоматически для конкретного используемого эмулятора. Проблема заключается в том, что получить эти данные не является тривиальной задачей.

В статье рассматриваются возможности взаимодействия с QEMU для получения информации о доступных устройствах, а также предлагается графический интерфейс для работы с эмулятором.

2. Модель оборудования эмулятора QEMU

QEMU предоставляет огромный набор устройств. На рис. 1 представлен пример как устройства организованы в эмуляторе.

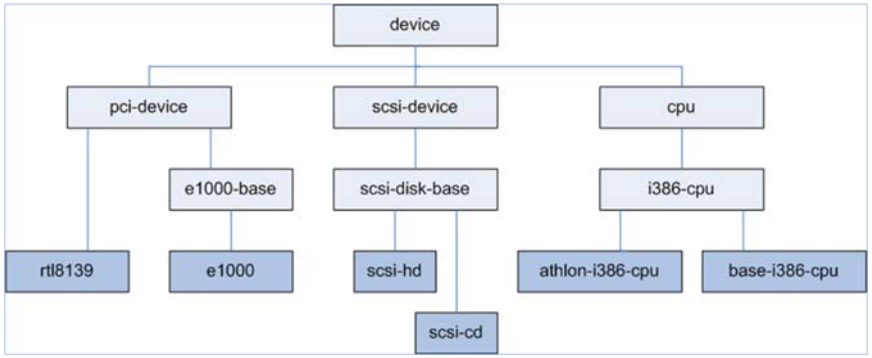

Рис. 1. Фрагмент дерева устройств QEMU
Fig. 1. The part of QEMU devices tree

Мы хотим получать список устройств и их характеристики, установленные по умолчанию, а также те, которые могут быть добавлены в эту конкретную машину.

Модель устройств должна включать шины, на которые могут быть определены устройства, сами устройства и их свойства. Она будет иметь древовидную структуру.

Примером конечного результата может послужить вывод команды монитора QEMU «`info qtree`». Ниже приведен небольшой фрагмент.

```
(qemu) info qtree
bus: main-system-bus
  type System
  dev: i440FX-pcihost, id ""
    pci-hole64-size = 2147483648 (2 GiB)
    short_root_bus = 0 (0x0)
    x-pci-hole64-fix = false
    bus: pci.0
      type PCI
      dev: e1000, id ""
```



```
        mac = "52:54:00:12:34:56"
        vlan = <null>
        ...
        netdev = "device-47"
  ...
  dev:...
```

Эта команда позволяет получить информацию обо всех устройствах, установленных на машине в данный момент. Большая часть устройств определена по умолчанию. В целом, такие данные позволили бы получить желаемую структуру устройств, но использование команды монитора не является хорошим решением, потому что вывод ответа ориентирован на пользователя, подвержен изменениям со стороны разработчика, а его разбор может быть осуществлен только на жестких сравнениях слов, используемых в выводе.

Кроме того, перебрать все возможные конфигурации машин невозможно, и какие-то устройства или шины можно упустить. Да и в целом решение получится крайне неудачным с архитектурной точки зрения. Для таких целей необходимо использовать форматы данных, предусмотренные для машинной обработки, например JSON, но для QMP (разд. 3) подобной команды нет.

3. Взаимодействие с QEMU

QEMU предоставляет два канала связи – монитор [6, 7] и QEMU Machine Protocol (QMP) [8]. Монитор предназначен для пользователей, позволяет вводить команды в привычном человеку виде и получать такие же ответы. Через QMP пользователь тоже может отправлять запросы эмулятору, однако они задаются в формате JSON [9] и больше предназначены для взаимодействия между программами.

Пример простой команды остановки машины:

- монитор: `stop`
- QMP: `{“execute” : “stop”}`

И монитор, и QMP предоставляют обширный набор команд для получения информации об эмуляторе и его состоянии. QMP это машинный протокол, поэтому в работе используется он.

Команды QMP делятся на несколько групп, в нашем случае интерес представляют команды из группы запросов (query-commands) и команды управления состоянием эмулятора.

4. Применение QMP для получения информации об устройствах виртуальной машины

QEMU эмулирует большое количество оборудования, которое может меняться от версии к версии или от настроек машины. Важно иметь актуальные списки моделей устройств, доступных в машине, поэтому вариант списать их с одного эмулятора и использовать повсеместно не подходит. Таким образом появилась необходимость запрашивать у QEMU какие устройства доступны в конкретной текущей версии.

Для частичного решения этой задачи подходит QMP. Наибольший интерес вызывают так называемые `query-commands`, команды, представляющие информацию по конкретным запросам.

QMP предоставляет достаточно большое количество команд для интроспекции конфигурации эмулятора. Мы используем следующие команды:

- `query-machines` – список машин;
- `query-cpu-definitions` – список процессоров;

- `qom-list-types` – информация о всех устройствах эмулятора. Предоставляется в виде списка соответствий «устройство» – «родитель»;
- `device-list-properties` – подробная информация о конкретном устройстве, заданном параметром.

Ниже приведены примеры команд и вывода.

```
-> {“execute”: “query-machines”}
<- {"return": [{"hotpluggable-cpus": true, "name": "pc-i440fx-2.2", "cpu-max": 255}, {"hotpluggable-cpus": true, "name": "pc-q35-2.4", "cpu-max": 255}, ... {"hotpluggable-cpus": true, "name": "pc-i440fx-2.8", "cpu-max": 255}, ... {"hotpluggable-cpus": true, "name": "pc-1.0", "cpu-max": 255}]}
-> "execute": "device-list-properties", "arguments": {"typename": "e1000"}}
<- {"return": [{"name": "bootindex", "type": "int32"}, {"name": "mitigation", "description": "on/off", "type": "bool"}, {"name": "addr", "description": "Slot and optional function number, example: 06.0 or 06", "type": "int32"}, ... {"name": "vlan", "description": "Integer VLAN id to connect to", "type": "int32"}, {"name": "romfile", "type": "str"}, {"name": "rombar", "type": "uint32"}, {"name": "autonegotiation", "description": "on/off", "type": "bool"}, {"name": "netdev", "description": "ID of a netdev to use as a backend", "type": "str"}]}
```

Используя команду `qom-list-types` можно построить дерево всех устройств, однако для полноты картины не будет хватать шин. Вывод этой команды содержит только названия устройств. Характеристики каждого устройства можно запрашивать с помощью команды `device-list-properties`. Однако, в дереве будут отображены все возможные устройства, доступные для этой машины, а не только относящиеся к текущей конфигурации, то же относится и к характеристикам устройств.

Команды, аналогичной команде монитора `info qtree`, предоставляющей список устройств, которые установлены в машине в данный момент, в QMP нет, и именно поэтому решение частичное.

5. Реализация

В рамках этой работы разрабатывался графический интерфейс для QEMU. Все вышеописанные манипуляции нашли применение в этом приложении.

Графические интерфейсы для эмулятора QEMU разрабатывались и ранее, например, AQEMU [6], QtEmu [7], JavaQemu[8]. Однако последние обновления этих инструментов зафиксированы несколько лет назад, они не поддерживают работу с детерминированным воспроизведением, а также не предоставляют возможности загрузить несколько версий эмулятора для удобного использования разных сборок. В нашем случае детерминированное воспроизведение является важным аспектом, поэтому было принято решение разрабатывать собственную графическую оболочку.

5.1 Управление состоянием эмулятора

Эмулятор может находиться в трех состояниях: работающий, остановленный, выключенный. Управление состоянием реализуется с помощью команд QMP: `cont` (возобновляет работу эмулятора после остановки), `stop` (пауза), `shutdown` (завершение работы эмулятора).

Эмулятор QEMU может быть запущен с опцией `-S`, что означает, что он будет запущен остановленным. Так как команды действия посылаются эмулятору посредством кнопок графического интерфейса, то необходимо чтобы их состояние было корректным

(активна/неактивна). Поэтому при старте эмулятор получает команду запроса статуса (`query-status`), дающую информацию о его текущем состоянии.

5.2 Получение информации об устройствах

Получить полную структуру устройств и шин через QMP на данный момент нельзя, поэтому пока получены только списки машин, моделей процессоров и сетевых карт.

Данные об этих устройствах соответствуют конкретной сборке QEMU и извлекаются в момент добавления этой сборки в графический интерфейс. Данные собираются и распределяются по конфигурационным файлам.

Для списка машин и моделей процессоров есть соответствующие команды `query-machines` и `query-cpu-definitions` соответственно. Списки машин и моделей процессоров обширны и неопытный пользователь может растеряться. Команда запроса информации о машинах имеет необязательный параметр `is-default`, который дает возможность узнать, какая машина использовалась бы QEMU по умолчанию. К сожалению, для процессора такого параметра нет, поскольку процессор по умолчанию зависит от машины. Поэтому в список процессоров добавлен вариант `default`, который отключит опцию командной строки, отвечающую за процессор, и QEMU назначит его по умолчанию самостоятельно.

Информация о других устройствах не может быть получена за один запрос. Сначала необходимо получить список всех устройств, затем запросить информацию по каждому из них и, исходя из результата, собирать нужные данные. Список устройств получаем командой `qom-list-types`. Далее команда `device-list-properties` должна отработать с каждым устройством из этого списка. Так как мы собирали информацию о сетевых картах, то в свойствах устройств искали указывающий на это параметр, а именно тип устройства `netdev`.

На данный момент структура дерева поддерживаемых устройств задана жестко. Машина создается с определенным минимальным набором устройств и шин (CPU, Memory, PCI, IDE). Эти устройства и шины не могут быть удалены. Некоторые устройства считаются неизменяемыми (например, машина), остальные могут быть отредактированы. Собственно, нередактируемые устройства в дереве не отображаются. Данные для редактирования некоторых устройств берутся из конфигурационных файлов, полученных с помощью QMP (модели ЦП, сетевые карты).

Добавляются устройства непосредственно на шину. Какие именно устройства могут быть добавлены, зависит от типа шины. Например, на шину PCI на данный момент можно добавить контроллеры IDE, SCSI и Network.

5.3 Определение списка снимков для запуска детерминированного воспроизведения

Снимки системы сохраняются в оверлей-файле. Этот файл является надстройкой над образом диска, в котором сохраняются все изменения, которые пользователь произвел с диском. Использование оверлея позволяет файлу образа диска оставаться неизменным. Чтобы пользователь мог указывать с какого снимка он хочет начать воспроизведение, мы должны предоставить ему список снимков, содержащихся в оверлее.

Детерминированное воспроизведение было разработано в ИСП РАН, и, на данный момент, в основную ветку QEMU приняты не все патчи. В связи с этим некоторую функциональность нельзя использовать в QemuGUI, рассчитанным на все версии QEMU (начиная с 2.8).

В данном случае речь идет о команде QMP `info_snapshots`, реализованной в ИСП РАН.

```
 -> { "execute": "info_snapshots" }
 <- { "return": [
          {
             "id": "1",
             "name": "retrace_0",
             "date": "2017-03-31",
             "time": "16:47:22",
             "vm-time": "00:00:00.000",
             "icount": 123
          },
          {
             ...
          }
       ]}
```

Вывод этой команды дает исчерпывающую информацию о снимках, но, к сожалению, с любыми версиями эмулятора воспользоваться ей пока что нельзя.

Другим путем получения этой информации является использование утилиты `qemu-img`, поставляемой вместе с эмулятором. Плюсом такого подхода является возможность получения списка снимков без запуска эмулятора (что не получилось бы при использовании команды QMP), а минусом вывод, разбор которого основан исключительно на том, как он выглядит. В случае изменения формата вывода разработчиками QEMU, мы столкнемся с необходимостью его переделывать. Сами разработчики эмулятора рекомендуют всегда использовать QMP.

Пример запроса и вывода:

```
->      qemu-img info C:/overlay0.ovl
<-      image: C:/overlay0.ovl
file format: qcow2
virtual size: 2.0G (2145386496 bytes)
disk size: 2.1M
cluster_size: 65536
backing file: C:/VBoxFolder/BSD.img
Snapshot list:
ID    TAG     VM SIZE    DATE                 VM CLOCK       ICOUNT
1     init    1.5M       2019-09-30 18:57:35  00:00:00.000   0
2     init    0          2019-09-30 18:57:35  00:00:00.000   0
Format specific information:
        compat: 1.1
        lazy refcounts: false
        refcount bits: 16
        corrupt: false
```

Следовательно, из всей этой информации нам нужен только раздел `Snapshot list`.

5.4 Описание графического интерфейса

Был реализован графический интерфейс [9], поддерживающий детерминированное воспроизведение. Главное окно программы представлено на рис. 2. В интерфейс можно добавлять неограниченное количество сборок QEMU, причем предусмотрена проверка совместимости исполняемого файла эмулятора и машины. Если машина не совместима с экземпляром эмулятора, то запустить его не удастся ни в обычном режиме, ни в режиме детерминированного воспроизведения.

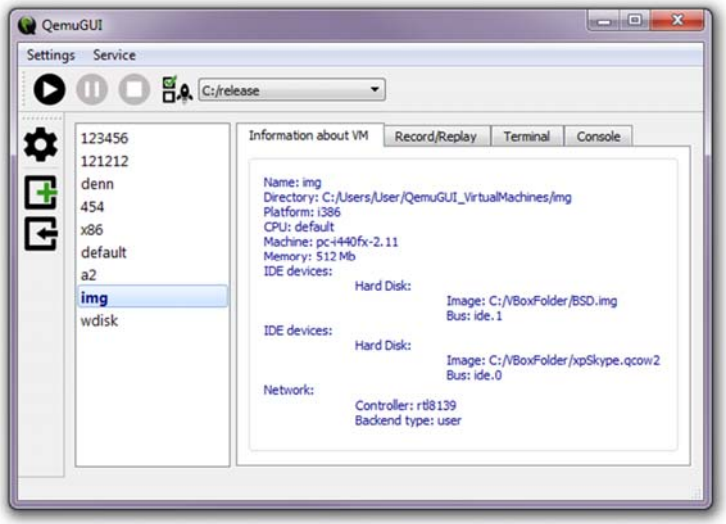

Рис. 2. Главное окно программы QemuGUI
Fig. 2. QemuGUI main window

QemuGUI позволяет создавать машины и конфигурировать их, окно создания машины представлено на рис. 3, а конфигурирования на рис. 4. В один момент времени может быть запущен только один экземпляр QEMU.

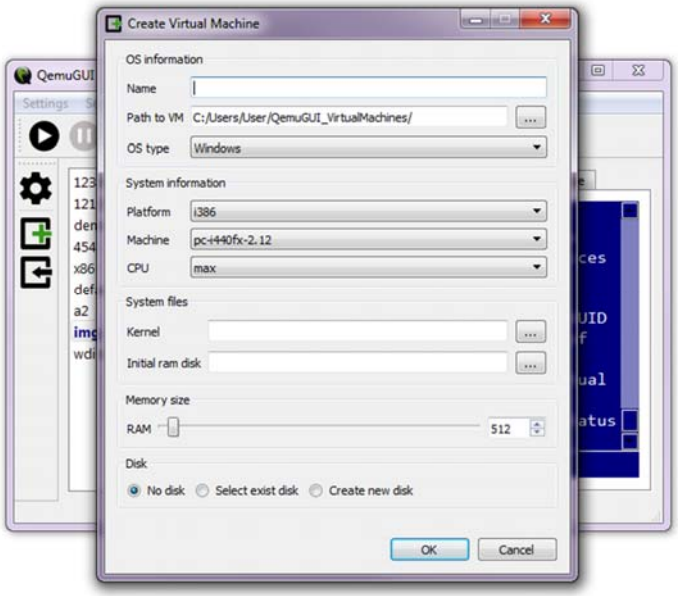

Рис. 3. Форма создания новой машины
Fig. 3. Create machine wizard

Так как на данный момент поддерживается достаточно узкий круг устройств, есть возможность написать дополнительные опции командной строки. Такие поля есть в форме редактирования устройства (в таком случае опции сохранятся) и в окне `Run options` (изменения этого окна не сохраняются). Помимо дополнительной командной строки в этом окне можно включить лог-файлы QEMU, а также установить флаги: запускать остановленной, запускать в режиме отладки и запускать с опцией `snapshot`. Окно представлено на рис. 5.

Основная область главного окна QemuGUI содержит четыре вкладки: информация о машине, детерминированное воспроизведение, монитор и консоль для вывода информации от эмулятора. На данный момент туда выводится командная строка при запуске QEMU.

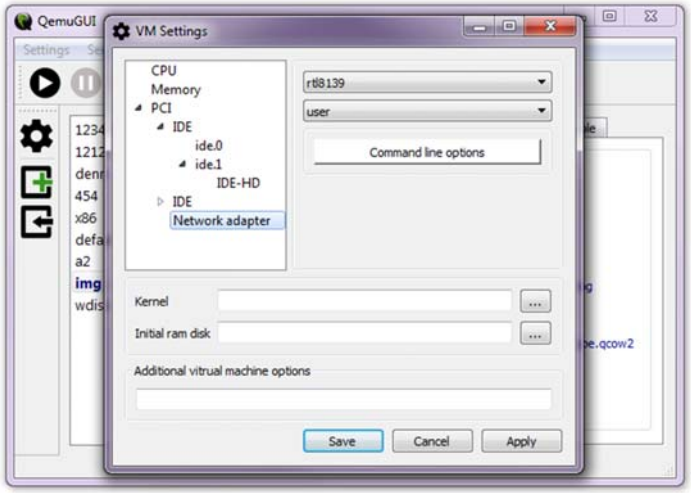

Рис. 4. Дерево устройств и их настройка
Fig. 4. Tree of devices and their settings

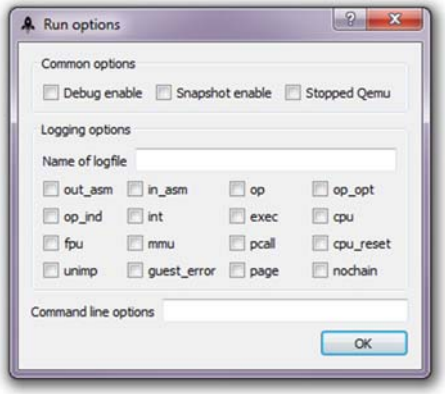

Рис.5. Окно настроек для запуска QEMU
Fig. 5. QEMU launch settings window

На вкладке детерминированного воспроизведения находится список текущих записанных сценариев, а также непосредственно кнопки для записи и воспроизведения, представлено на рис. 6. Если сценарий записывается с опцией создания снимка, то при воспроизведении будет предложен выбор снимка, с которого будет начато воспроизведение. По умолчанию создается снимок `init`, если была включена опция автоматического создания снимков, они будут называться `auto_N` и будут представлены в выпадающем списке.

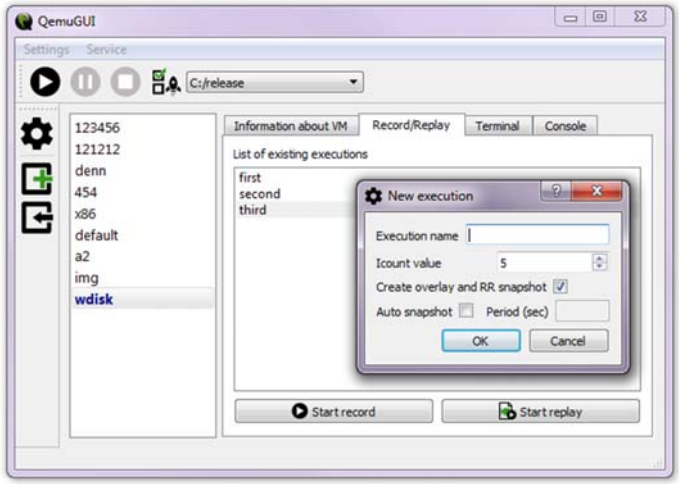

Рис. 6. Создание нового журнала
Fig. 6. Create execution wizard

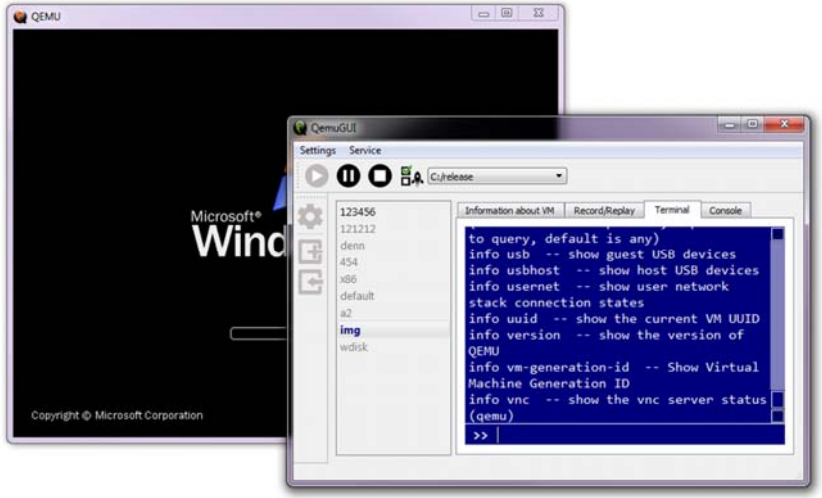

Рис. 7. Монитор QEMU
Fig. 7. QEMU monitor

Вкладка с монитором представлена на рис. 7. Наличие монитора в интерфейсе делает использование эмулятора более удобным. Внешний вид терминала можно менять в настройках. Также в настройках можно менять порты для подключения QMP и монитора.

Все данные и настройки, используемые QemuGUI, хранятся в конфигурационных файлах в формате XML.

К недостаткам QemuGUI можно отнести ограниченный набор виртуальных устройств, поддерживаемых для добавления в дерево. Для смягчения этого недостатка в интерфейсе присутствует поле для дополнительных опций эмулятора, с помощью которого можно вручную дополнить конфигурацию машины.

Графический интерфейс находится в открытом доступе [10].

6. Заключение

В результате был разработан графический интерфейс для эмулятора QEMU, позволяющий получать актуальные сведения о некоторых устройствах, а также помогающий пользователям более удобно и эффективно работать с QEMU, в частности, и с детерминированным воспроизведением. Графическая оболочка работает под операционными системами Windows и Linux.

References

[1]. QEMU the FAST! processor emulator. Available at: https://www.qemu.org (accessed 07.11.2019)
[2]. QEMU code. Available at: https://git.qemu.org/git/qemu.git (accessed 12.10.2019)
[3]. QEMU interface introspection: From hacks to solutions. Markus Armbruster. KVM Forum 2015. Available at: https://events.static.linuxfound.org/sites/events/files/slides/armbru-qemu-introspection.pdf (accessed 07.11.2019)
[4]. QEMU User Documentation. Available at: https://qemu.weilnetz.de/doc/qemu-doc.html (accessed 12.10.2019)
[5]. Pavel Dovgalyuk. Deterministic Replay of System's Execution with Multi-target QEMU Simulator for Dynamic Analysis and Reverse Debugging. In Proc. of 16th European Conference on Software Maintenance and Reengineering, 2012, vol. 1, pp. 553-556.
[6]. QEMU Monitor documentation. Available at: https://en.wikibooks.org/wiki/QEMU/Monitor (accessed 07.11.2019)
[7]. QEMU Monitor. Available at: http://people.redhat.com/pbonzini/qemu-test-doc/_build/html/topics/pcsys_005f monitor.html (accessed 07.11.2019)
[8]. QMP Documentation. Available at: https://wiki.qemu.org/Documentation/QMP (accessed я 07.11.2019)
[9]. Introducing JSON. Available at: http://www.json.org/ (accessed 07.11.2019)
[10]. AQEMU. Available at: https://sourceforge.net/projects/aqemu/ (accessed 07.11.2019)
[11]. QtEmu. Available at: https://qtemu.org/ (дата обращения 07.11.2019)
[12]. JavaQemu. Available at: https://sourceforge.net/projects/javaqemu/ (accessed 07.11.2019)
[13]. Qt Documentation. Available at: https://doc.qt.io/qt-5/qt5-intro.html (accessed 07.11.2019)
[14]. QemuGUI source. Available at: https://github.com/ispras/qemu-gui (accessed 07.11.2019)

Информация об авторах / Information about authors

Наталья Игоревна ФУРСОВА – старший научный сотрудник, старший преподаватель, кандидат технических наук. Сфера научных интересов: интроспекция и инструментирование виртуальных машин, динамический анализ кода, эмуляторы.

Natalia Igorevna FURSOVA – senior researcher, senior lecturer, Ph.D in Technical Sciences. Research interests: virtual machines introspection and instrumentation, dynamic analysis of code, emulators.

Павел Михайлович ДОВГАЛЮК – старший научный сотрудник, доцент, кандидат технических наук. Сфера научных интересов: интроспекция и инструментирование виртуальных машин, динамический анализ кода, эмуляторы.

Pavel Michailovich DOVGALYUK – senior researcher, associate professor, Ph.D. in Technical Sciences. Research interests: virtual machines introspection and instrumentation, dynamic analysis of code, emulators.