Моделирование и анализ данных
2020. Том 10. № 4. С. 60–78
DOI: https://doi.org/10.17759/mda.2020100406
ISSN: 2219-3758 (печатный)
ISSN: 2311-9454 (online)


Modelling and Data Analysis
2020. Vol 10, no. 4, pp. 60–78
DOI: https://doi.org/10.17759/mda.2020100406
ISSN: 2219-3758 (print)
ISSN: 2311-9454 (online)


## ◇◇◇◇◇◇◇ КОМПЛЕКСЫ ПРОГРАММ ◇◇◇◇◇◇◇

УДК 004.031.4

# Организация клиент-серверного взаимодействия на одной электронно-вычислительной машине

***Попков С.И.****
Московский государственный психолого-педагогический университет, г. Москва, Российская Федерация
ORCID: https://orcid.org/0000-0003-2566-1262
e-mail: rslw25@gmail.com

Проведено исследование имеющихся подходов к организации клиент-серверного взаимодействия. Особое внимание уделено практическому применению клиент-серверного взаимодействия на одной электронно-вычислительной машине. Проведен эксперимент с реализацией минимального кода сервера и применением виртуализации. Представлен результат эксперимента, описаны технические особенности и полученные результаты.

***Ключевые слова:*** клиент, сервер, виртуализация, клиент-серверное взаимодействие.

**Для цитаты:**
*Попков С.И.* Организация клиент-серверного взаимодействия на одной электронно-вычислительной машине // Моделирование и анализ данных. 2020. Том 10. № 4. С. 60–78. DOI: https://doi.org/10.17759/mda.2020100406

## 1. ВВЕДЕНИЕ

Клиент-серверная архитектура распределенных информационных систем относится к одной из наиболее востребованных в настоящее время парадигм построения веб-приложений. Этому во многом способствует постоянное развитие сети Интернет и вклад в создание современных веб-технологий со стороны консорциума W3C [1] и других организаций.

Клиент-серверная архитектура предполагает наличие сервера – узла вычислительной системы с соответствующим программным обеспечением, позволяющим

***Попков Сергей Игоревич,*** кандидат физико-математических наук, доцент факультета информационных технологий, заведующий лабораторией, Московский государственный психолого-педагогический университет, г. Москва, Российская Федерация, ORCID: https://orcid.org/0000-0003-2566-1262, e-mail: rslw25@gmail.com

обрабатывать запросы к серверу с целью получения актуализированной информации и ее дальнейшей передачи в доступном формате отправителю запроса – клиенту.

Однако, в силу различных обстоятельств, сеть Интернет может оказаться недоступной как физически (отсутствие соединения из-за аварии или качественных изменений естественных условий окружающей среды), так и логически (при работе с секретными или защищенными данными, когда компания или корпорация ограничивается локальной сетью). В последнем случае актуальной проблемой является поддержка работоспособности разработчика (или группы разработчиков) приложений в критических ситуациях, затрудняющих или делающих невозможным доступ к защищенной локальной сети без доступа к мировой сети Интернет (к частному случаю такой внештатной ситуации можно отнести удаленную работу в условиях глобальной пандемии).

В качестве одного из возможных решений для описанной ситуации можно предоставить в пользование разработчика общий снимок («snapshot») системы [2], очищенный от потенциальных уязвимостей и значимых для компании секретных данных. Однако, сам процесс такой очистки является достаточно трудоемким и содержащим потенциальные уязвимости ввиду участия человеческого фактора. Кроме того, такой подход не гарантирует совместимость вычислительных средств разработчика со снимком системы, поэтому обеспечение его работоспособности может привести к дополнительной трате ресурсов.

Другой подход предполагает абстрагирование от конкретных данных, где это возможно, и реализацию общей концепции работы системы с переносом клиент-серверной архитектуры. Это решение является более защищенным, но может оказаться неприемлемым на поздних стадиях разработки, где использование реальных рабочих данных может стать существенной необходимостью. Кроме того, этот способ требует затрат на перенос аппаратно-вычислительной среды, а также воссоздание ее работоспособности с соблюдением требований информационной безопасности.

Возможен и менее ресурсозатратный подход, если допустима атомарная автономная разработка, где участники проекта могут разрабатывать независимые узлы программного обеспечения, которые впоследствии легко соединяются между собой благодаря согласованным интерфейсам (например, микросервисная архитектура [3]). В этом случае допустима организация клиент-серверного взаимодействия на одной электронно-вычислительной машине. Цель данной работы – рассмотреть данную организацию и найти одно из возможных концептуальных решений для указанного случая.

## 2. КРАТКОЕ ОПИСАНИЕ ПРИНЦИПОВ РАБОТЫ КЛИЕНТ-СЕРВЕРНОГО ВЗАИМОДЕЙСТВИЯ БЕЗ РАЗДЕЛЕНИЯ АППАРАТНЫХ РЕСУРСОВ С ПРИМЕНЕНИЕМ ВИРТУАЛИЗАЦИИ

Для того, чтобы организовать работу клиента и сервера как взаимно независимых сущностей обычно требуется их аппаратное разделение. Действительно, для доступа

к серверу со стороны клиента используется одна программно-аппаратная среда, а для обеспечения работы сервера – другая. В связи с этим возникает вопрос, как в рамках поставленной задачи, где используется одна электронно-вычислительная машина, обеспечить разделение ресурсов? Ответом является практический прием, который позволяет изолировать логически взаимосвязанные вычислительные ресурсы, абстрагируясь от конкретной аппаратной реализации. Такой подход носит название «виртуализация».

Таким образом, на целевой машине, из основных аппаратных ресурсов, принадлежащих операционной системе, изначально установленной на машине (хостовая конфигурация, «host»), выделяется набор изолированных ресурсов, организованных таким образом, чтобы они представляли собой независимую виртуальную машину, на которую можно установить отдельную операционную систему (гостевая конфигурация, «guest»). Делается это при помощи специальных программных комплексов, включающих в себя низкоуровневые драйверы, прикладные программные и пользовательские интерфейсы. Эти комплексы носят название «гипервизоры», зачастую они написаны на нескольких языках программирования [4].

Существует множество платных и бесплатных гипервизоров, различающихся по характеристикам, поддерживаемым операционным системам, способам виртуализации, предоставляемым функциям. Среди них наиболее популярны такие программные комплексы, как Hyper-V [5], vSphere [6], VirtualBox [7], KVM [8] и др.

Подробный разбор различных программных комплексов, обеспечивающих виртуализацию, выходит за рамки данной работы. В данной работе будет использоваться VirtualBox как удобная для начинающего пользователя кросс-платформенная система с поддержкой полной аппаратной виртуализации и графическим интерфейсом. Это позволяет наглядно показать основные шаги по настройке клиент-серверного взаимодействия, но более искушенный пользователь, при необходимости, может легко перенести описанные шаги на другие средства виртуализации.

Как правило, клиентские машины используют операционную систему «Windows» ввиду ее высокой популярности и удобства для пользователя с любым уровнем опыта работы за компьютером [9]. Серверные машины, наоборот, зачастую используют «Linux» из-за большего количества удобных инструментов, более высокого уровня защищенности и надежности, производительности, выгоды с точки зрения цены и модифицируемости [10]. В данной работе мы также будем придерживаться аналогичной конфигурации.

## 3. УСТАНОВКА И НАСТРОЙКА ИНСТРУМЕНТОВ ДЛЯ ОБЕСПЕЧЕНИЯ РАБОТЫ ПРОЦЕССА ВИРТУАЛИЗАЦИИ

Прежде всего, необходимо установить средство виртуализации. Данный этап в работе подробно не рассматривается. Для гипервизора VirtualBox, который используется в данной работе, существует подробное описание шагов установки в руководстве пользователя [11] и подходящие зеркальные ссылки для загрузки дистрибутива

исходя из хостовой операционной системы и текущего сетевого региона, из которого осуществляется загрузка [12].

Далее рассматривается версия VirtualBox 6.1.16 с графическим интерфейсом на базе библиотеки Qt 5.6.2. Эта версия является наиболее актуальной [13] на момент написания статьи (в начале января 2021 года). В качестве операционных систем рассматривается 64-разрядная Windows версии 10.0.20279.1 [14] (host) и 32-разрядная Linux Mint LMDE 4 [15] (guest).

После установки гипервизора необходимо настроить элементы сетевого окружения, которые будут впоследствии осуществлять виртуальное клиент-серверное взаимодействие. Для этого необходимо открыть менеджер сетей хоста (комбинация клавиш «Ctrl+H» по умолчанию из главного окна менеджера VirtualBox). В появившемся окне создать сеть «VirtualBox Host-Only Ethernet Adapter» и настроить ее параметры вручную, задав адрес сети (например, исходя из умолчаний – частный IP-адрес 192.168.56.1), а маску сети оставить стандартной (255.255.255.0). IP-адрес будем использовать статический, поэтому DHCP-сервер можно отключить. Результат должен соответствовать изображению, представленному на рис. 1.

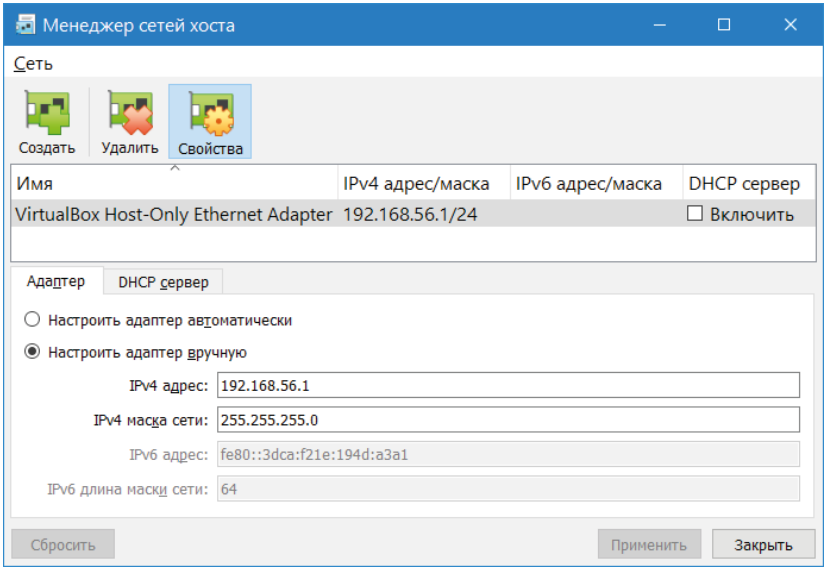


*Рис. 1. Менеджер сетей хоста*

Создание сети может потребовать предоставления прав системного администратора, поскольку эти изменения влияют на состояние хостовой операционной системы, в частности, на сетевые подключения. Открыв соответствующий раздел настроек операционной системы, необходимо убедиться в появлении нового подключения (рис. 2). Если подключение не обнаружено, может потребоваться осуществить перезапуск операционной системы.

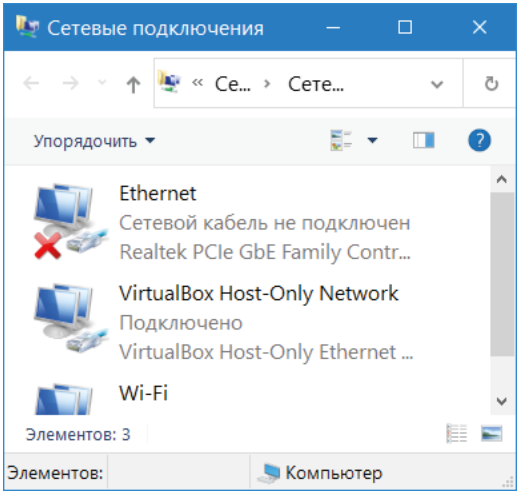


*Рис. 2. Сетевые подключения. Сеть «VirtualBox Host-Only Network» отмечена в списке соединений как подключенная*

Виртуальная сеть для взаимодействия между реальной хостовой и виртуальной гостевой машиной воспринимается как подключение по несуществующему кабелю. Сеть не подключена к сети Интернет и безопасна в использовании. На данный момент, однако, еще не создана и не подключена виртуальная машина, что видно из статистики после перезапуска адаптера в окне описания состояния соединения (рис. 3).

Перейдем к этапу создания виртуальной машины. Для начала необходимо определить тип и разрядность используемой операционной системы, которую планируется использовать в качестве гостевой. Можно просто указать общий тип (например, «Linux»), однако, делать этого не рекомендуется, поскольку тогда придется настраивать многие параметры самостоятельно. Удобнее предоставить средству виртуализации самостоятельно определить наиболее оптимальные параметры для работы гостевой операционной системы.

В случае с Linux Mint LMDE 4 известно, что эта операционная система основана на семействе дистрибутивов Debian [16]. Сам Debian является основой популярной операционной системы Ubuntu [17], что обеспечивает совместимость с большинством основных пакетов и средств, поддерживаемых дочерней операционной системой. При этом Debian – более легковесная операционная система и менее требовательная к ресурсам, что делает ее хорошим кандидатом на роль гостевой операционной системы в нашей работе для демонстрации базового примера клиент-серверного взаимодействия.

32-разрядная версия Linux Mint LMDE 4 менее требовательна к ресурсам, чем 64-разрядная, что позволяет ее использовать в фоновом режиме в роли сервера даже на старых аппаратных конфигурациях. Ограничения, накладываемые низкой разрядностью на взаимодействие с ресурсами, в рамках данной демонстрации не существенны.

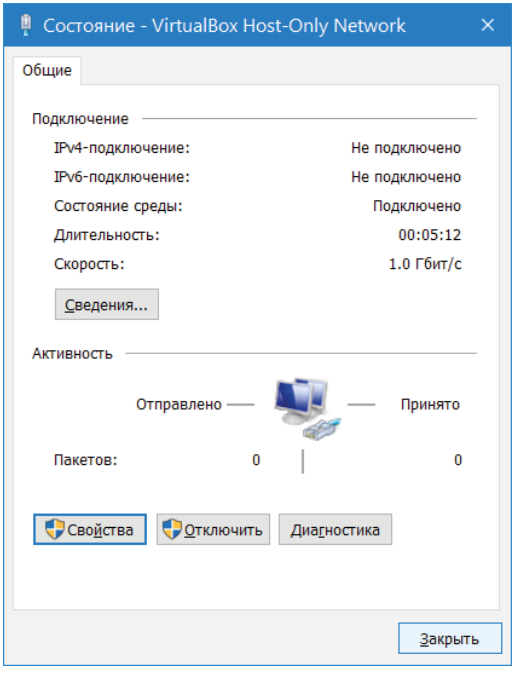


*Рис. 3. Состояние сети*

Создадим виртуальную машину (комбинация клавиш «Ctrl+N» по умолчанию из главного окна менеджера VirtualBox), назовем ее произвольным образом (например, «SelfServer»), укажем папку для хранения данных о виртуальной машине и заполним остальные поля согласно рис. 4.

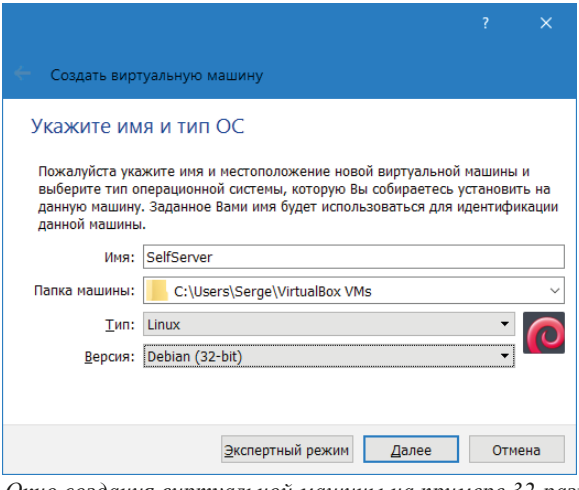


*Рис. 4. Окно создания виртуальной машины на примере 32-разрядной операционной системы из семейства Debian*

Объем памяти установим 1 ГБ (1024 МБ), виртуальный жесткий диск сделаем динамическим, размер установим 16 ГБ (чтобы гарантировать достаточное количество места для всех необходимых служб). Динамический жесткий диск будет работать медленнее фиксированного, однако, будет занимать меньше места (до тех пор, пока не будет полностью заполнен) и будет создан быстрее (для фиксированного виртуального жесткого диска необходимо сразу выделить весь запрошенный объем на физическом жестком диске).

После того, как все необходимые данные будут заполнены, а подготовка файлов для работы виртуальной машины будет завершена, в главном окне менеджера VirtualBox отобразится соответствующая виртуальная машина и краткое описание ее основных настроек с возможностью внесения изменений (рис. 5).

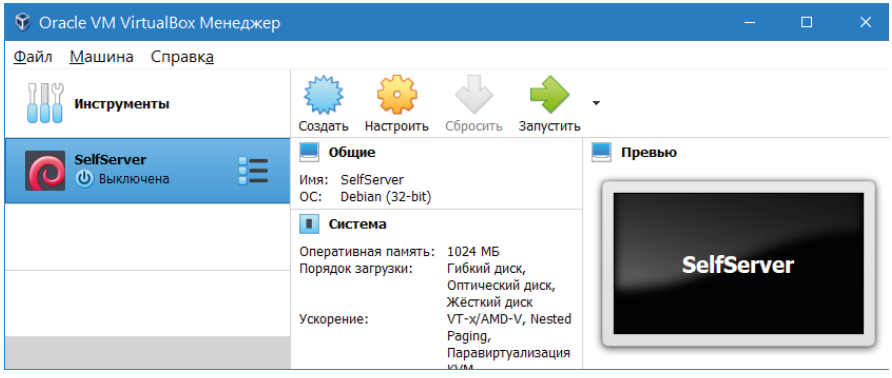


*Рис. 5. Состояние созданной виртуальной машины*

По умолчанию виртуальная машина подключена с помощью метода NAT («Network Address Translation», метод трансляции сетевых адресов) через виртуальный адаптер к той же сети, к которой подключена хостовая машина. Необходимо дополнительно подключить ранее созданную сеть для обеспечения клиент-серверного взаимодействия. Для этого в настройках виртуальной машины на вкладке «Сеть» устанавливаем переключатель «Включить сетевой адаптер» в установленное положение, выбираем тип подключения «Виртуальный адаптер хоста», имя ранее созданного и зарегистрированного в хостовой операционной системе виртуального сетевого адаптера «VirtualBox Host-Only Ethernet Adapter». Пример работающих настроек приведен на рис. 6.

По умолчанию общие папки не установлены. Если конфигурация виртуальной машины импортируется из внешнего источника, необходимо убедиться в отсутствии общих папок для обеспечения достоверности проводимых процедур: любое взаимодействие между клиентом и сервером должно осуществляться строго по выделенной для этого виртуальной сети – без доступа к сети Интернет и без использования дополнительных внешних механизмов, упрощающих решение задачи и нарушающих чистоту эксперимента.

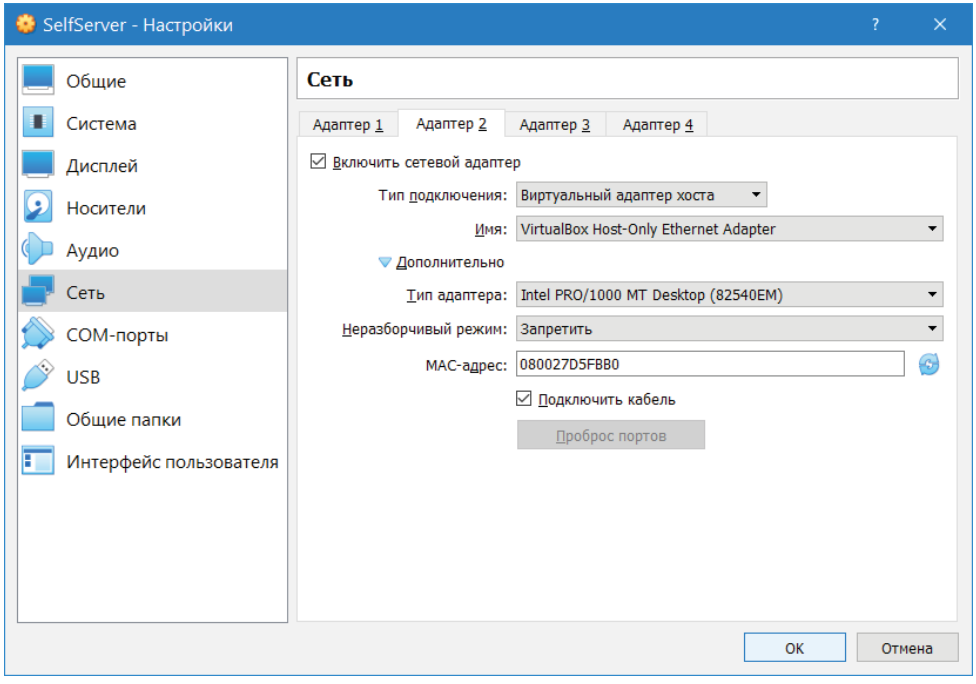


*Рис. 6. Настройки сетевого адаптера*

С домашней страницы сайта операционной системы Linux Mint LMDE 4 необходимо скачать официальный дистрибутив [18] в виде образа загрузочного диска в формате ISO [19], после чего запустить установщик и провести инсталляцию операционной системы на гостевую виртуальную машину. Более подробное описание процедуры выходит за рамки данной работы.

На данном этапе предполагается, что гостевая операционная система установлена и корректно работает на виртуальной машине, а сама машина успешно запускается. Иногда могут возникать проблемы с инициализацией виртуальной машины – как правило, они связаны с конфликтами между различными средствами виртуализации. В этом случае для корректного проведения описываемого эксперимента необходимо отключить или деинсталлировать другие средства виртуализации.

В некоторых случаях на хостовой операционной системе Windows 10 возможно возникновение ошибки средства виртуализации VirtualBox при запуске виртуальной машины. Код ошибки «VERR_INTNET_FLT_IF_NOT_FOUND» указывает на проблему с инициализацией и подключением виртуального сетевого адаптера «VirtualBox Host-Only Ethernet Adapter». Само окно выглядит как на рис. 7.

Эта ошибка является достаточно распространенной и решается переподключением виртуального сетевого адаптера на уровне хостовой операционной системы в окне состояния виртуального сетевого адаптера.

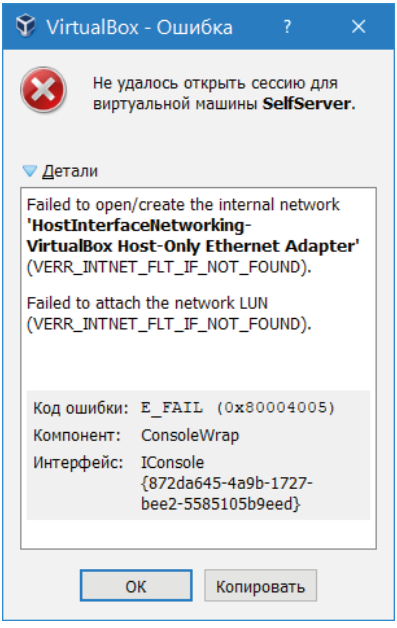


*Рис. 7. Окно с сообщением об ошибке*

В следующем разделе будет рассмотрен процесс настройки виртуальной машины.

## 4. ПРОЦЕСС НАСТРОЙКИ ВИРТУАЛЬНОЙ МАШИНЫ ДЛЯ ПОДДЕРЖКИ РАБОТЫ СЕРВЕРНЫХ СЛУЖБ И ОСУЩЕСТВЛЕНИЯ КЛИЕНТ-СЕРВЕРНОГО ВЗАИМОДЕЙСТВИЯ

После процедуры установки операционная система сообщает о проблемах подключения к сети хоста. Чтобы выявить проблему, можно проверить конфигурацию сети с помощью инструмента «ifconfig», как показано на рис. 8.

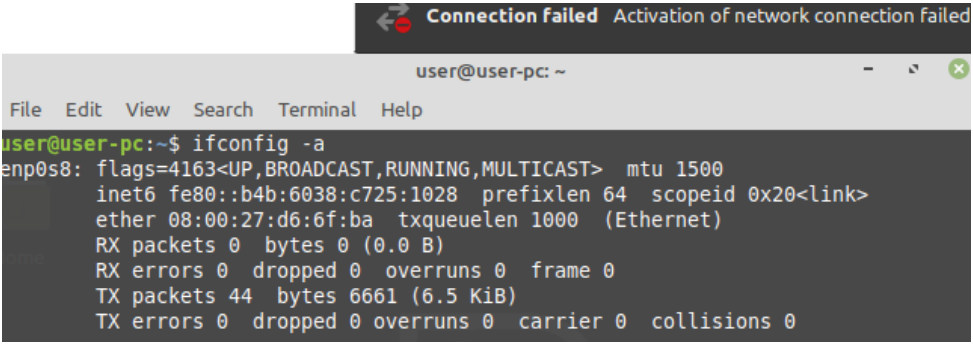


*Рис. 8. Результат применения ifconfig после установки*

Несмотря на то, что сеть на стороне виртуальной машины обнаруживается сразу, она остается недоступной для использования без конфигурации сетевого интерфейса и определения статического адреса для виртуальной машины, по которому она будет определяться в сети. В приводимом примере выбран адрес, согласующийся с адресом сети и маской, а именно – 192.168.56.10. При необходимости можно определить собственный адрес, исходя из конкретной конфигурации сети.

Для того, чтобы закрепить искомый адрес за гостевой виртуальной машиной, необходимо добавить в файл, отвечающий за регистрацию сетевых интерфейсов, соответствующие строки с командами. Эти команды будут инициализированы при следующем запуске операционной системы или инициализации сетевых интерфейсов. Пример внесенных изменений показан на рис. 9 (добавлены строки 4–7).

Повторный запуск инструмента «ifconfig» демонстрирует появление новой записи для конфигурации интерфейса сети хоста “inet 192.168.56.10 netmask 255.255.255.0 broadcast 192.168.56.255”. Теперь сетевой интерфейс настроен и позволяет обеспечить клиент-серверное взаимодействие. В качестве сервера выступает гостевая виртуальная машина с операционной системой на базе Linux (Debian), в качестве клиента – хостовая операционная система Windows 10, оснащенная современным браузером (в описываемом примере используется Google Chrome 87.0.4280.88).

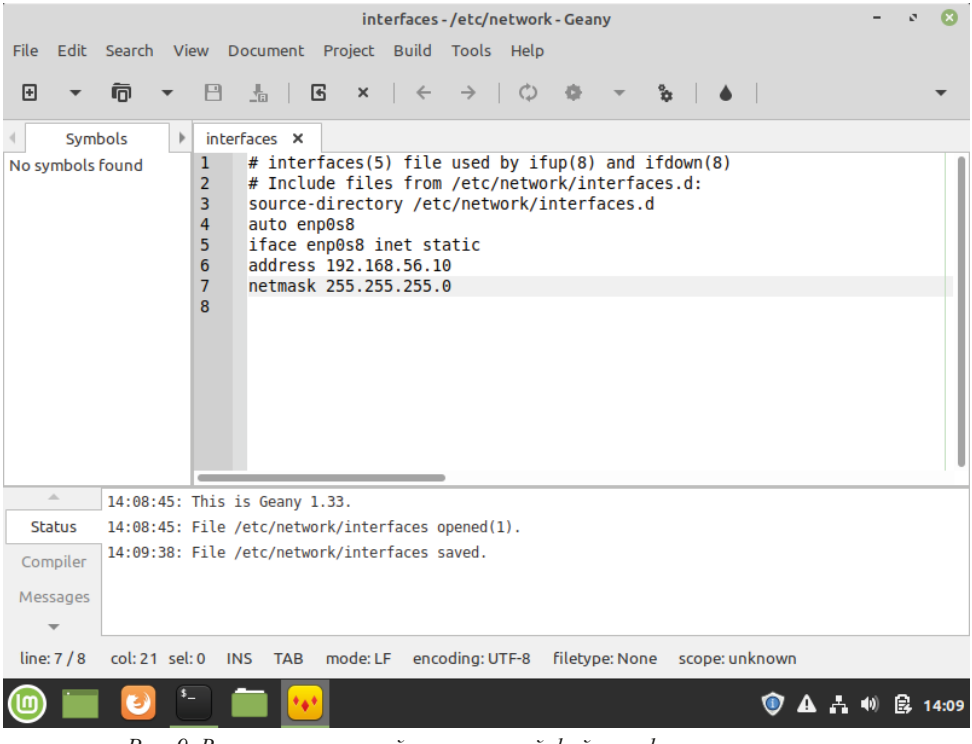


*Рис. 9. Внесение изменений в системный файл конфигурации сетевых интерфейсов ОС Linux Mint LMDE 4 с использованием IDE Geany*

Для того, чтобы убедиться, что сервер готов к выполнению своих функций, добавим для него роль SSH-сервера. SSH (“Secure Shell” [20], защищенная оболочка) – протокол для предоставления функций удаленного управления операционной системой. Позволяет безопасно передавать данные в незащищенной среде. Как правило, под соединение SSH выделяется порт TCP с номером 22, хотя при самостоятельной организации соединения рекомендуется менять стандартный номер порта во избежание bruteforce-атак со стороны потенциального взломщика [21]. Однако, в случае изолированной сети хоста для локально развернутой виртуальной машины подобными мерами безопасности можно пренебречь. Для проводимого в рамках данной работы эксперимента функционал SSH-сервера позволит убедиться в возможности полноценно взаимодействовать с сервером на стороне клиента, а также позволит подготовить необходимые файлы, описывающие сценарии взаимодействия с сервером.

Для установки SSH-сервера выполним в консоли команду “sudo apt install ssh”, после завершения установки проверить работу соответствующих служб можно, подключившись к серверу по локальной сети (через localhost командой “ssh localhost”). В этом случае виртуальная машина должна обратиться сама к себе и создать SSH-сессию, из которой можно выйти с помощью команды “logout” – в этом случае управление вернется в вызывающую консоль.

Для разрешения внешнего доступа следует добавить соответствующее правило во встроенный брандмауэр операционной системы, которое разрешит доступ к порту TCP извне. Для настройки правил доступа в Linux Mint можно воспользоваться как приложением с графическим интерфейсом, так и консолью. В проводимом эксперименте использовалась консольная команда “sudo iptables -A INPUT -p tcp –dport ssh -j ACCEPT”. Псевдоним «ssh» обозначает номер порта 22.

Для внешнего доступа к серверу потребуется клиент SSH, одним из наиболее популярных приложений в данной категории является PuTTY [22]. Данное приложение предоставляет различные настройки и позволяет выбрать требуемый протокол доступа, а также возможность работы с собственным форматом данных, не ограниченным конкретными спецификациями (режим “raw”).

Для проверки работы сервера SSH и возможности доступа к виртуальной машине не потребуется менять протокол и режимы работы. Достаточно убедиться, что виртуальная машина запущена, а в окне конфигурации подключения указать искомый адрес для подключения – 192.168.56.10.

Диалоговое окно настройки клиента SSH PuTTY с указанием необходимых данных о параметрах подключения перед получением доступа к виртуальной машине изображено на рис. 10.

Подключение осуществляется после нажатия кнопки “Open” в диалоговом окне. Открывается консольное окно, похожее на стандартный консольный интерфейс гостевой операционной системы. После авторизации (с указанием имени уполномоченного пользователя гостевой операционной системы и соответствующего пароля) в этом окне можно выполнить произвольные команды, для которых у соответствующей учетной записи имеется разрешение.

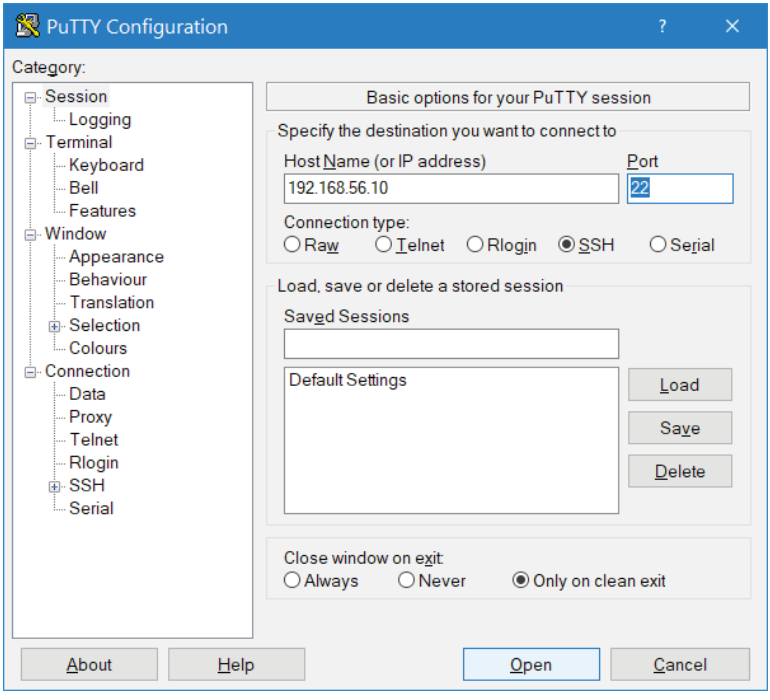


*Рис. 10. Окно настройки SSH-клиента*

Выполним команду создания каталога “mkdir” и вывода списка файлов на экран “ls” (рис. 11).

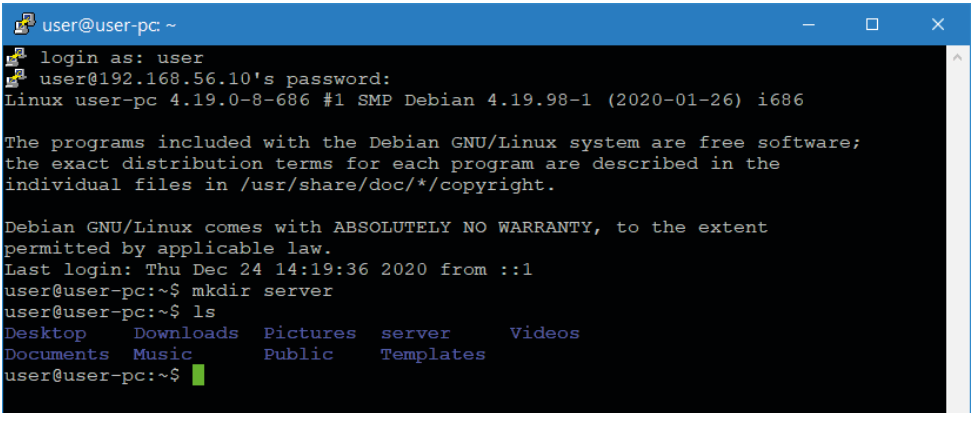


*Рис. 11. Окно консоли PuTTY*

В результате была создана папка “server”. Если открыть файловый менеджер в гостевой операционной системе, можно увидеть, что папка действительно была создана (рис. 12):

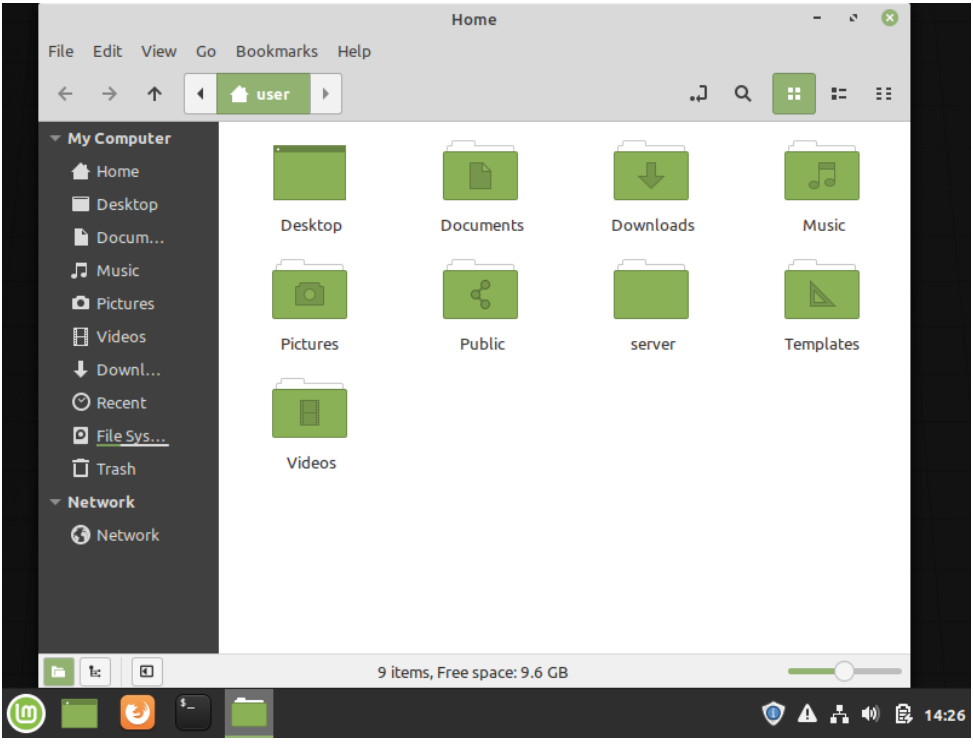


*Рис. 12. Папка «server», созданная через подключение SSH, отображается в файловом менеджере гостевой операционной системы*

Убедившись в том, что соединение настроено корректно, и сеть позволяет корректно осуществлять клиент-серверное взаимодействие, перейдем непосредственно к практической части эксперимента и создадим простейший сервер.

## 5. ПРОЦЕСС НАСТРОЙКИ ВИРТУАЛЬНОЙ МАШИНЫ ДЛЯ ПОДДЕРЖКИ РАБОТЫ СЕРВЕРНЫХ СЛУЖБ И ОСУЩЕСТВЛЕНИЯ КЛИЕНТ-СЕРВЕРНОГО ВЗАИМОДЕЙСТВИЯ

Исходный код для сервера (на базе фреймворка Flask [23]) показан на рис. 13.

Весь сценарий клиент-серверного взаимодействия, описанный исходным кодом в файле base.py, можно разбить на три части:

1) инициализация – загрузка необходимых для работы сервера библиотек, объектов и функций, запуск веб-приложения на основе выбранного фреймворка (строки 1–2);
2) создание веб-формы, отображаемой на стороне клиента и состоящей из кнопки отправки данных и поля ввода для сообщения со стороны пользователя (строки 4–11);

3) обработка результата, полученного со стороны клиента: передача клиенту информации о статусе обработки результата и сохранение полученного сообщения в файл на стороне сервера (строки 13–19).

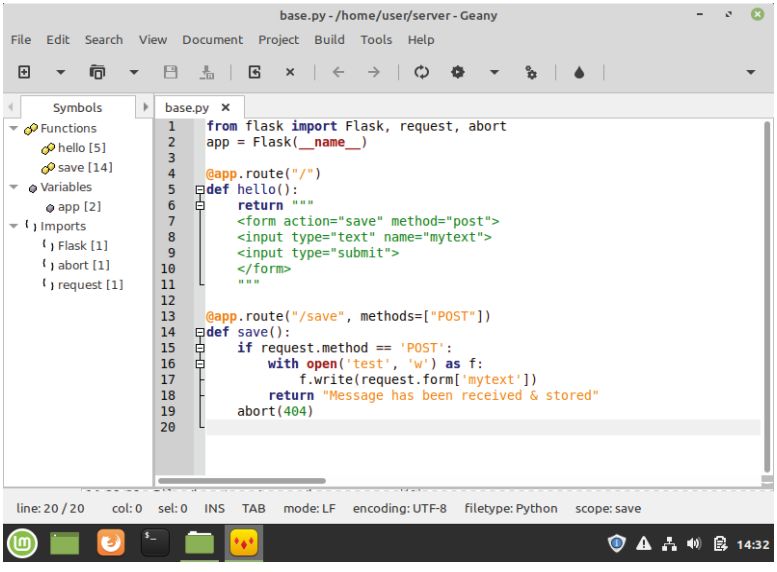


*Рис. 13. Исходный код сервера*

Для того, чтобы разрешить доступ к произвольному порту сервера со стороны клиента, необходимо задать соответствующие правила для брандмауэра. Это можно сделать через консольный интерфейс, воспользовавшись командами iptables и ufw. Для Linux Mint доступен графический интерфейс для добавления соответствующих правил (рис. 14).

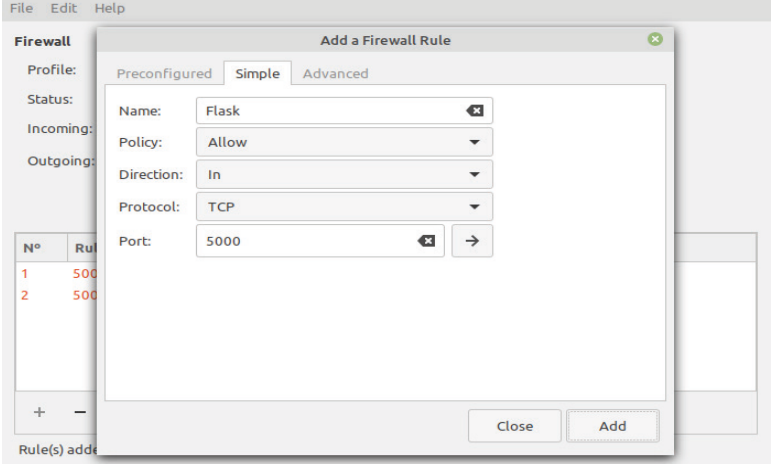


*Рис. 14. Добавление правил брандмауэра через графический интерфейс Linux Mint*

Установка фреймворка Flask и настройка брандмауэра выходит за рамки данной статьи; в качестве программной основы для сервера можно выбрать любое другое программное средство, настройки также будут специфичны для конкретных параметров конфигурации сервера.

В консоли сервера зададим переменную окружения FLASK_APP равной значению “base.py” (“export FLASK_APP=base.py”) и запустим приложение Flask из каталога, в котором хранятся файлы сервера, для всех доступных IP-адресов (“flask run –host=0.0.0.0”), после чего, убедившись, что в стандартном потоке вывода появилась строка «Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)», и сервер запущен корректно, на стороне клиента (хостовой операционной системы) необходимо открыть браузер и ввести адрес, привязанный к серверу, с портом веб-приложения сервера (по умолчанию для Flask это порт с номером 5000). Таким образом, адрес будет «http://192.168.56.10:5000». Результатом будет отображение минималистичной формы ввода данных, как показано на рис. 15.

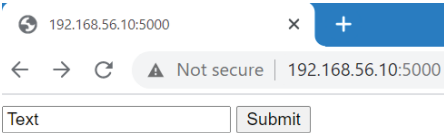


*Рис. 15. Форма ввода данных*

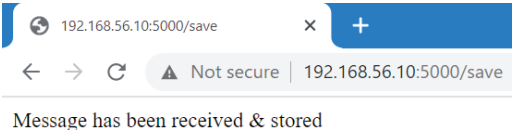


*Рис. 16. Сообщение об успешной отправке данных*

Введя в поле текст «Text» и нажав кнопку отправки на стороне клиента, пользователь инициирует на стороне сервера запуск обработчика и ответ, представленный на рис. 16.

На стороне сервера будет сгенерирован файл «test», его содержимым будет полученное сообщение (рис. 17). Это доказывает возможность клиент-серверного взаимодействия между хостовой и гостевой системами в качестве клиента и сервера, соответственно; клиент может как изменять состояние сервера (на примере создания файла с произвольным содержимым), так и получать данные от него (на примере получаемых сообщений в браузере).

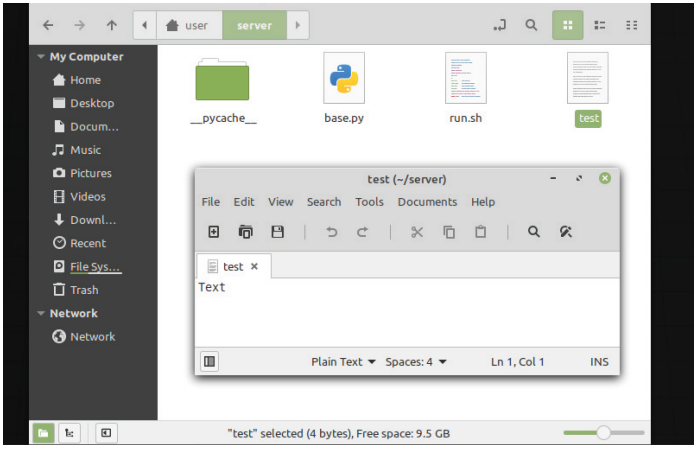


*Рис. 17. Сгенерированный в ходе клиент-серверного взаимодействия файл*

Продемонстрированный пример простейшего клиент-серверного взаимодействия легко можно расширить для более комплексных и полезных прикладных сценариев, например, фоновой автоматизации процессов сбора данных в сети, конвертации данных, а также процедуре отладки клиент-серверной архитектуры.

## 6. ЗАКЛЮЧЕНИЕ

В статье рассмотрена организация клиент-серверного взаимодействия на одной электронно-вычислительной машине, описан механизм виртуализации и продемонстрировано его использование с помощью одного из инструментов виртуализации, продемонстрирована настройка сети для осуществления клиент-серверного взаимодействия, показано практическое применение организации клиент-серверного взаимодействия на одной электронно-вычислительной машине в ходе эксперимента с запуском минимального кода сервера и обменом сообщениями между хостовой и гостевой системами; предложены идеи практически значимых расширений проведенного эксперимента для задач автоматизации процессов и отладки.

***Литература***

1. W3C – URL: https://www.w3.org/ (дата обращения 14.01.2020 г.)
2. Snapshot Red Hat Virtualization – URL: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.0/html/virtual_machine_management_guide/sect-snapshots (дата обращения 14.01.2020 г.)
3. Microservice Architecture – URL: https://microservices.io/ (дата обращения 14.01.2020 г.)
4. What is a hypervisor? – URL: https://www.vmware.com/topics/glossary/content/hypervisor (дата обращения 14.01.2020 г.)
5. Introduction to Hyper-V on Windows 10 – URL: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/ (дата обращения 14.01.2020 г.)
6. What is vSphere? – URL: https://www.vmware.com/products/vsphere.html (дата обращения 14.01.2020 г.)
7. Oracle VM VirtualBox – URL: https://www.virtualbox.org/ (дата обращения 14.01.2020 г.)
8. Linux KVM – URL: https://www.linux-kvm.org/page/Main_Page (дата обращения 14.01.2020 г.)
9. Windows – URL: https://www.microsoft.com/en-us/windows (дата обращения 14.01.2020 г.)
10. Linux vs. Microsoft Windows Servers – URL: https://www.volico.com/linux-vs-microsoft-windows-servers/ (дата обращения 14.01.2020 г.)
11. Oracle® VM VirtualBox® User Manual – URL: https://www.virtualbox.org/manual/UserManual.html (дата обращения 14.01.2020 г.)
12. Download VirtualBox – URL: https://www.virtualbox.org/wiki/Downloads (дата обращения 14.01.2020 г.)
13. Changelog for VirtualBox 6.1 – URL: https://www.virtualbox.org/wiki/Changelog (дата обращения 14.01.2020 г.)
14. Announcing Windows 10 Insider Preview Build 20279 – URL: https://blogs.windows.com/windows-insider/2020/12/14/announcing-windows-10-insider-preview-build-20279/ (дата обращения 14.01.2020 г.)
15. Release Notes for LMDE 4 – URL: https://linuxmint.com/rel_debbie.php (дата обращения 14.01.2020 г.)

16. Debian – The Universal Operating System – URL: https://www.debian.org/ (дата обращения 14.01.2020 г.)
17. Ubuntu: Enterprise Open Source and Linux – URL: https://ubuntu.com/ (дата обращения 14.01.2020 г.)
18. Download LMDE 4 Debbie – URL: https://www.linuxmint.com/download_lmde.php (дата обращения 14.01.2020 г.)
19. Editions for Linux Mint 4 “Debbie” – URL: https://linuxmint.com/release.php?id=37 (дата обращения 14.01.2020 г.)
20. SSH (Secure Shell) – URL: https://www.ssh.com/ssh/ (дата обращения 14.01.2020 г.)
21. Смена порта SSH-сервера как мера защиты от брутфорса – URL: https://putty.org.ru/articles/change-default-sshd-port.html (дата обращения 14.01.2020 г.)
22. PuTTY: Telnet/SSH Клиент – URL: https://putty.org.ru/ (дата обращения 14.01.2020 г.)
23. Flask – URL: https://flask.palletsprojects.com/ (дата обращения 14.01.2020 г.)

# Organization of Client-Server Interaction on a Single Computer

***Sergei I. Popkov****
Moscow State University of Psychology and Education, Moscow, Russia
ORCID: https://orcid.org/0000-0003-2566-1262
e-mail: rslw25@gmail.com

A research of existing approaches to the organization of client-server interaction is conducted. Special attention is paid to the practical application of client-server interaction on a single computer. An experiment was conducted with the implementation of minimal server code and the application of virtualization. The result of the experiment is presented, the technical features and the results obtained are described.

***Keywords:*** client, server, virtualization, client-server interaction.

**For citation:**
Popkov S.I. Organization of Client-Server Interaction on a Single Computer. *Modelirovanie i analiz dannykh = Modelling and Data Analysis*, 2020. Vol. 10, no.4, pp. 60–78. DOI: https://doi.org/10.17759/mda.2020100406 (In Russ., abstr. in Engl.).

***References***

1. W3C – URL: https://www.w3.org/ (req. date 14.01.2020 г.)
2. Snapshot Red Hat Virtualization – URL: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.0/html/virtual_machine_management_guide/sect-snapshots (req. date 14.01.2020)
3. Microservice Architecture – URL: https://microservices.io/ (req. date 14.01.2020)
4. What is a hypervisor? – URL: https://www.vmware.com/topics/glossary/content/hypervisor (req. date 14.01.2020)
5. Introduction to Hyper-V on Windows 10 – URL: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/ (req. date 14.01.2020)
6. What is vSphere? – URL: https://www.vmware.com/products/vsphere.html (req. date 14.01.2020)
7. Oracle VM VirtualBox – URL: https://www.virtualbox.org/ (req. date 14.01.2020)
8. Linux KVM – URL: https://www.linux-kvm.org/page/Main_Page (req. date 14.01.2020)
9. Windows – URL: https://www.microsoft.com/en-us/windows (req. date 14.01.2020)
10. Linux vs. Microsoft Windows Servers – URL: https://www.volico.com/linux-vs-microsoft-windows-servers/ (req. date 14.01.2020)
11. Oracle® VM VirtualBox® User Manual – URL: https://www.virtualbox.org/manual/UserManual.html (req. date 14.01.2020)
12. Download VirtualBox – URL: https://www.virtualbox.org/wiki/Downloads (req. date 14.01.2020)

***Sergei I. Popkov,** PhD in Physical and Mathematical Sciences, Assistant Professor of the faculty of information technologies, head of the laboratory, Moscow State University of Psychology and Education, Moscow, Russia, ORCID: https://orcid.org/0000-0003-2566-1262 , e-mail: rslw25@gmail.com

13. Changelog for VirtualBox 6.1 – URL: https://www.virtualbox.org/wiki/Changelog (req. date 14.01.2020)
14. Announcing Windows 10 Insider Preview Build 20279 – URL: https://blogs.windows.com/windows-insider/2020/12/14/announcing-windows-10-insider-preview-build-20279/ (req. date 14.01.2020)
15. Release Notes for LMDE 4 – URL: https://linuxmint.com/rel_debbie.php (req. date 14.01.2020)
16. Debian – The Universal Operating System – URL: https://www.debian.org/ (req. date 14.01.2020)
17. Ubuntu: Enterprise Open Source and Linux – URL: https://ubuntu.com/ (req. date 14.01.2020)
18. Download LMDE 4 Debbie – URL: https://www.linuxmint.com/download_lmde.php (req. date 14.01.2020)
19. Editions for Linux Mint 4 “Debbie” – URL: https://linuxmint.com/release.php?id=37 (req. date 14.01.2020)
20. SSH (Secure Shell) – URL: https://www.ssh.com/ssh/ (req. date 14.01.2020)
21. SSH-server port changing as an anti-bruteforce defensive measure – URL: https://putty.org.ru/articles/change-default-sshd-port.html (req. date 14.01.2020 г.)
22. PuTTY: Telnet/SSH Client – URL: https://putty.org.ru/ (req. date 14.01.2020)
23. Flask – URL: https://flask.palletsprojects.com/ (req. date 14.01.2020)