МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Н.Э. БАУМАНА
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Сёмкин П.С., Сёмкин А.П.

Методические материалы к лабораторным работам
по дисциплине
«Операционные системы»

Лабораторная работа № 1

«Установка операционной системы Ubuntu. Интерфейс пользователя»

Москва
2022 г.

ОГЛАВЛЕНИЕ

1 Цель работы

Целью работы является:

— знакомство с концепцией виртуализации;

— создание виртуальной машины и установка гостевой операционной системы Ubuntu;

— знакомство и работа в графической оболочке Unity ОС Ubuntu;

— знакомство и работа с командным интерпретатором bash ОС Ubuntu.

Продолжительность работы – 2 часа.

2 Теоретическая часть

2.1 *Создание виртуальных машин и установка гостевых операционных систем*

2.1.1 Физические и виртуальные ресурсы операционных систем

Физический ресурс - реально существует и при распределении его между процессами используются все его физические характеристики.

Примеры физических ресурсов: оперативная память (характеристики: объём ОП, время доступа, центральный процессор (быстродействие, число ядер и т.д.).

Виртуальный ресурс – некоторая программная модель, построенная на основе одного или нескольких физических ресурсов и обладающая характеристиками, отличными от характеристик физических ресурсов, на основе которых она построена. Пример виртуального ресурса: виртуальная память, которая представляется имеющей гораздо больший объём, чем физическая память, установленная на компьютере. Виртуальная память строится на основе оперативной и внешней памяти.

Виртуализация предоставляет дополнительные возможности по использованию ресурсов, предоставляя расширенный интерфейс и скрывая сложные архитектуры аппаратных средств

2.1.2 Виртуальные машины

Расширением идей виртуализации является предоставление пользователю виртуальной вычислительной машины.

Виртуальная машина — это иллюзия реальной машины, созданная с использованием программ виртуализации и средств операционной системы.

С точки зрения пользователя, виртуальные машины выглядят как реально существующие машины. На одной реальной машине может быть установлено несколько виртуальных, на каждой из которых может быть установлено собственное программное обеспечение. Согласно такой концепции было создано множество программных систем виртуализации.

Каждая виртуальная машина содержит собственную операционную систему и множество виртуальных аппаратных ресурсов.

Мультипрограммные операционные системы разделяют ресурсы системы между несколькими процессами. Каждому из этих процессов выделяется порция ресурсов реальной машины. Каждый процесс видит менее мощную машину с меньшими возможностями, чем машина, на которой действительно выполняется процесс.

Системы виртуальных машин разделяют ресурсы единственной машины по-другому. Они создают иллюзию того, что одна реальная машина - это на самом деле несколько машин. Они создают виртуальные процессоры, устройства хранения и ввода/вывода, возможно, с куда большими объёмами и возможностями, чем таковые у реальной машины.

ОС реальной машины создает среду, в которой могут работать виртуальные машины. Она обеспечивает поддержку различных операционных систем и управляет реальной машиной, лежащей в основе виртуальной машины. Она предоставляет каждому пользователю доступ к

устройствам реальной машины: процессору, оперативной памяти накопителям и устройствам ввода-вывода. ОС управляет одновременно несколькими виртуальными машинами, а не отдельными задачами или процессами.

Виртуальные машины, работающие под управлением ОС, работают почти так же, как и реальные их аналоги, но медленнее, поскольку ОС выполняет операции одновременно множества виртуальных машин.

Способность одновременно выполнять несколько операционных систем имеет множество применений. Во первых, это облегчает переход между различными операционными системами или различными версиями одной системы. Она позволяет пользователям обучаться одновременно с выполнением рабочих задач. Клиенты могут исполнять одновременно разные операционные системы.

Процессы, выполняющиеся на виртуальных машинах, управляются не ОС, а операционными системами, работающими на виртуальных машинах. Эти операционные системы выполняют все свои обычные функции, включая управление устройствами хранения, планирование загрузки процессора, управление вводом/выводом, защиту пользователей друг от друга, защиту операционной системы от пользователей, мультипрограммирование, управление процессами и задачами, обработку ошибок и так далее.

Пользователь виртуальной машины оперирует эквивалентом целой реальной машины, а не набором ресурсов, который он разделяет с множеством других пользователей.

2.1.3 Программа виртуализации Oracle VM VirtualBox

Данная программа виртуализации фирмы Oracle предназначена для создания виртуальных машин, установки на этих машинах операционных систем и работы в среде виртуальных ОС.

Гостевая операционная система в VirtualBox может быть установлена либо с использованием дистрибутива, либо импортом готовой конфигурации, сгенерированной ранее операционной системы.

1.**Создание виртуальной машины и установка гостевой ОС с использованием дистрибутива.**

Выбрать пункт меню «Создать»

Задать:

имя новой виртуальной машины

тип устанавливаемой гостевой системы;

определить количество выделяемой ей оперативной памяти;

создать виртуальный диск (фиксированного размера или динамически расширяющийся).

После создания виртуальной машины производится обычная установка гостевой операционной с установочного диска или образа ISO дистрибутива.

2. Создание виртуальной машины и установка гостевой ОС путем импорта готовой конфигурации

Выбрать пункт меню «Импорт»

Выбрать устанавливаемую конфигурацию в открытых форматах виртуализации OVA или OVF.

Открывается окно «Укажите параметры импорта»

Перечисляются устройства импортируемой конфигурации. Некоторые параметры можно изменить или отключить

После этого производится создание новой виртуальной машины и импорт выбранной конфигурации

2.2 *Графический интерфейс пользователя операционной системы Ubuntu*

2.2.1 Графическая оболочка Unity

Unity – графическая оболочка, разработанная компанией Canonical для операционной системы Ubuntu. Unity является графической оболочкой по умолчанию

2.2.2 Рабочий стол

Рабочий стол Unity состоит из следующих элементов:

1 - кнопка меню **Dash.** Открывает общее меню поиска.

6 - системная панель Unity.

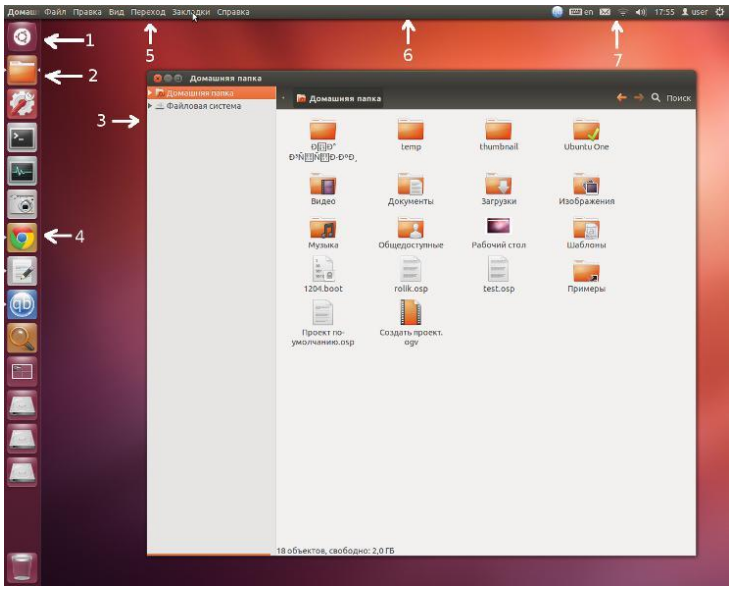
2, 4 - кнопки быстрого запуска программ. Работающие программы отмечены треугольником. Активная программа отмечена двумя треугольниками.

1, 2, 4 - элементы панели быстрого запуска (Launcher).

3 - окно активной программы.

5 - меню активной программы (появляется при наведении курсора мыши или нажатии клавиши Alt).

7 – область уведомления(трей).



Вверху экрана размещается системная панель, которая содержит (слева направо) пространство для меню программ и системный трей.

По левой границе экрана размещается вертикальная панель с кнопками быстрого запуска приложений (Launcher), и там же отражаются иконки работающих программ и кнопка открытия меню поиска Dash. Панель запуска совмещена с панелью задач.

В окнах программ нет строки меню. Меню динамически загружается в системную панель Unity, то есть в верхней панели размещается меню той программы, которая сейчас активна. Это меню видно, если навести на панель курсор мыши или нажать клавишу Alt.

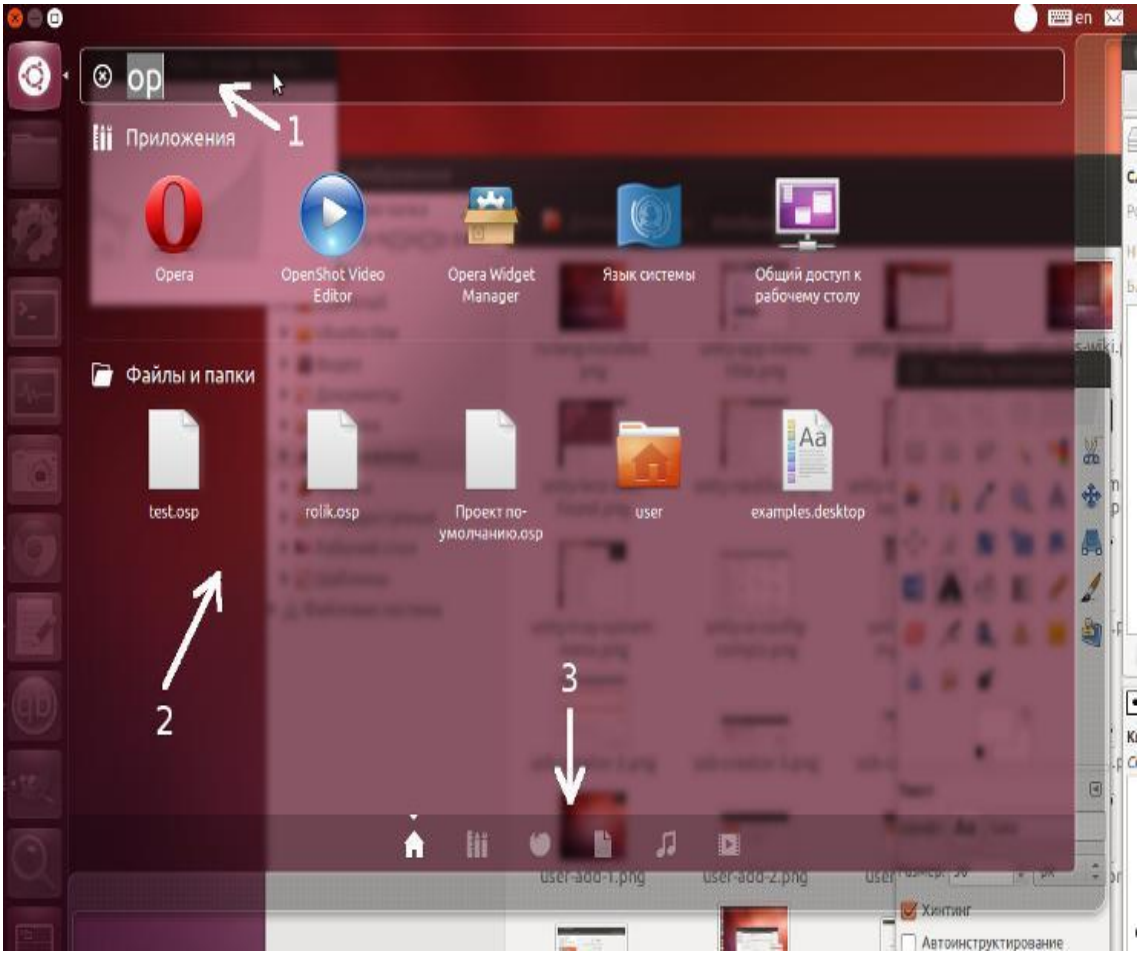
2.2.3 Меню поиска Dash

Открывается нажатием кнопки в панели быстрого запуска Launcher или нажатием клавиши "Win" на клавиатуре.

1 - строка поиска. По мере ввода символов будут отображаться программы и файлы имена которых содержат эти символы. Строка поиска инициируется сразу после открытия Dash,

2 - найденные через поиск программы и файлы.

3 - линзы (Dash Lens) переключатели поиска из глобального состояния (искать везде) в специфическую категорию - поиск только файлов, только программ и т. д.



2.2.4 Панель запуска программ Unity (Launcher)

Программы запускаются либо кнопками непосредственно с панели быстрого запуска (Launcher), либо через общее меню Dash. Запущенные программы отображаются в той же панели быстрого запуска, такими же значками, только их значки выделены светлым треугольником или несколькими (если несколько окон программы).

Чтобы вставить кнопку запуска программы в панель быстрого запуска (Launcher):

1. *Запустить программу.*
2. *Когда ее значок появится в боковой панели нужно на нем н*

кною кнопкую кнопку мыши и выбрать пункт меню "Прикрепить к панели

Или:

1. *открыть меню Dash*
2. *найти в нем значок программы*
3. *мышью перетащить этот значок в Launcher*

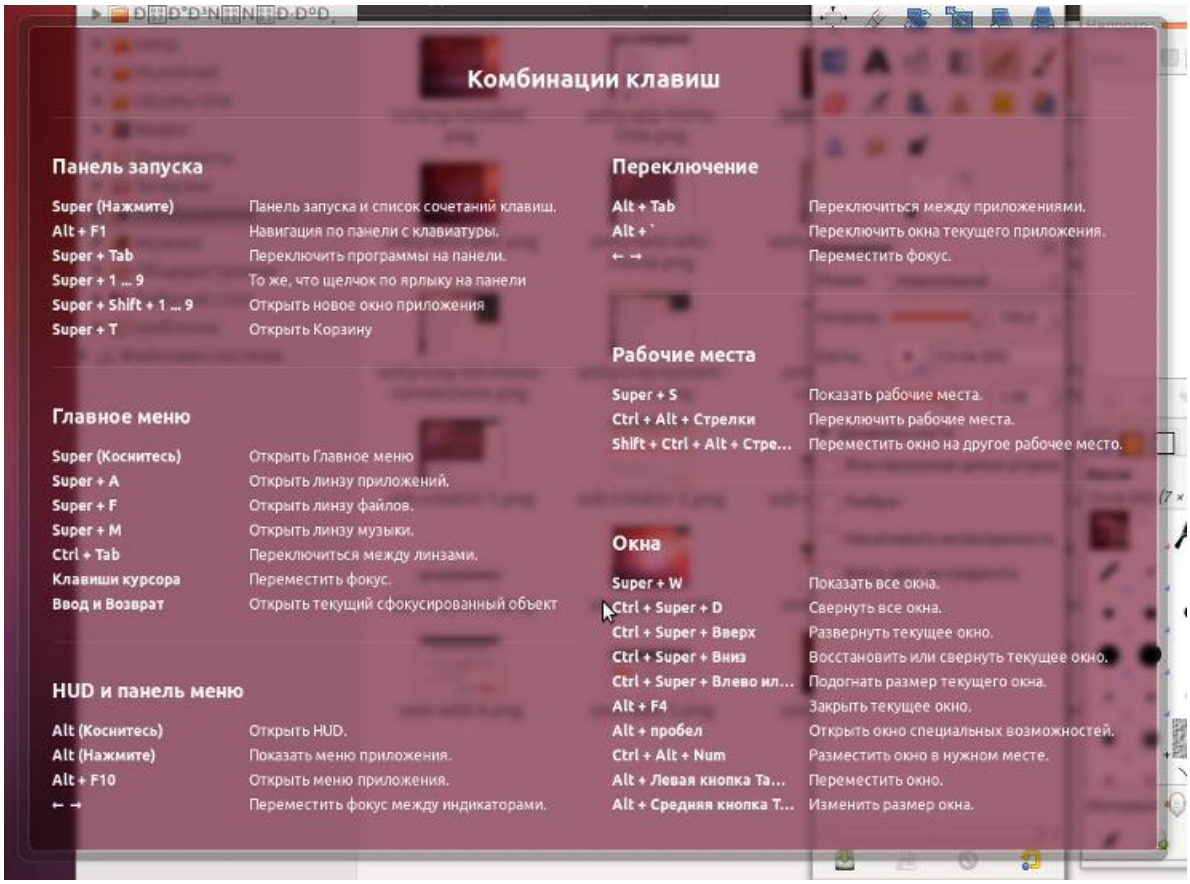
Чтобы удалить кнопку запуска программы из панели запуска нужно сделать обратное - нужно на нем нажать правую кнопку мыши и снова выбрать пункт меню "Прикрепить к панели" - у постоянных значков в этой строке птичка

2.2.5 Использование клавиатуры в Unity

- **F10** - активизировать трей. Затем можно перемещаться по элементам трея стрелками.
- **Win** - открыть меню программ (Dash) с поиском. По мере того как вводят буквы в поле, программа выдает подходящие варианты. Поиск работает как по английскому языку (оригинальному имени программ) так по русскому - именам ярлыков программ.
- **Alt+Tab** - переход между окнами запущенных программ.
- **Win+D** - свернуть все окна и освободить рабочий стол. ***В Ubuntu 11.10 сделали Ctrl+Alt+D.***
- **Win+R** - диалог "Выполнить". В нем можно вписать однострочную терминальную команду.
- ***Alt+F2*** **-** поиск по истории введенных команд.
- **Tab** или стрелки вверх\вниз\вправо\влево - перемещение в меню.
- **Win+S** - показать 4 рабочих стола сразу.
- **Win+Tab** - при нажатии этого сочетания на каждой кнопке появляется номер, нажав этот номер на клавиатуре можно запустить\перейти в эту программу. Нужно удерживать нажатой **Win** до ввода цифры.
- **Ctrl+L** - изменить режим адресной строки в Наутилусе с "табов" на текстовый.

- Удерживая нажатой клавишу "**Win**" (в Linux она называется **Super**) и нажимая клавишу с цифрой можно запустить одну из программ которые закреплены в панели быстрого запуска. Таким же способом можно перемещаться между окнами работающих программ

Увидеть список клавиатурных сокращений можно если нажать и удерживать нажатой клавишу "Win" (Super):

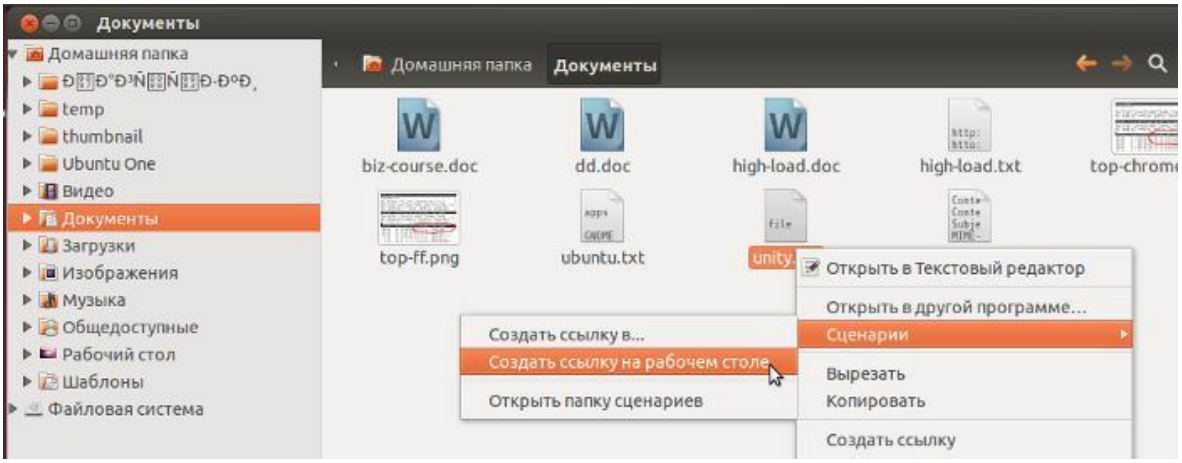


Переопределить сочетания клавиш можно через апплет "Клавиатура" и через CompizConfig.

2.2.6 Типичные задачи в Ubuntu Unity

- **Создать ярлык файла на Рабочем столе Ubuntu Unity**
  - *Нажать правую кнопку на файле и в меню выбрать пункт "Создать ссылку"*

- *Скопировать эту ссылку на рабочий стол.*



- **Создать ярлык программы на Рабочем столе Ubuntu Unity**

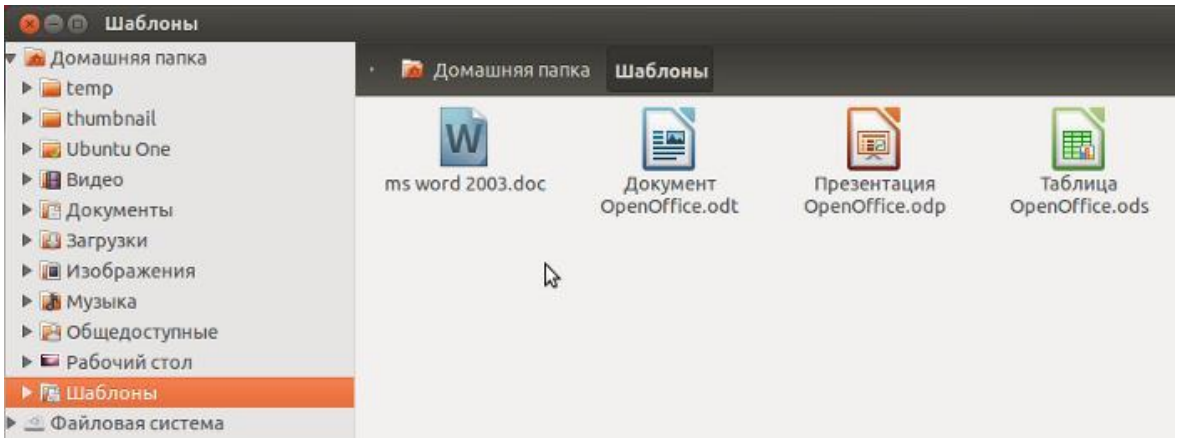
- *открыть меню поиска Dash*
- *найти в нем значок программы*
- *мышью перетащить этот значок на Рабочий стол*

Или можно перетащить мышью значок из панели (Launcher) на Рабочий стол.

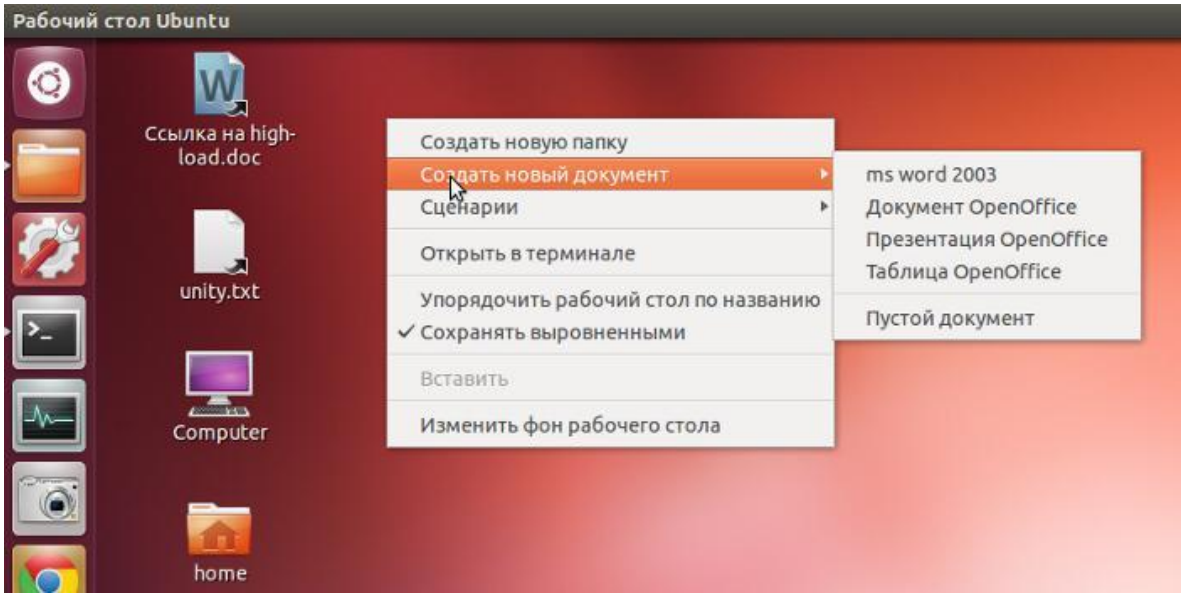
- **Добавить новые файлы в меню "Создать новый файл"**

"Из коробки" в этом меню можно создавать только один тип файла - текстовый. Но в это меню можно добавить любые другие типы файлов. Для нужно создать пустые файлы нужного формата в папке "Шаблоны", которая находится в Домашней папке пользователя.

Например, в программе LibreOffice Writer создаётся новый пустой файл и сохраняется этот файл в формате MS Office 2003, в папку "Шаблоны". После этого в меню "Создать новый файл" появится новая строка.



После этого в контекстном меню появятся новые пункты:



2.2.7 Переключение между открытыми окнами программ

Перейти из одного окна в другое можно разными способами:

- Нажать на клавиатуре клавиши **Alt + Tab**. Откроется список иконок программ, в этом списке можно перемещаться, последовательно нажимая и отпуская клавишу **Tab** (клавиша **Alt** все время нажата). Для открытия выбранного окна нужно отпустить клавишу **Alt**.
- Кликнуть левой кнопкой мыши на иконке программы в панели запуска (вертикальная панель по левой границе экрана).

Для того чтобы свернуть все окна и освободить рабочий стол нужно нажать на клавиатуре клавиши **Ctrl + Windows + D**. Через программу **Ubuntu Tweak** можно вывести значок "Свернуть все окна" на панель запуска.

2.3 *Интерфейс командной строки операционной системы Ubuntu*

Для интерпретации и выполнения команд пользователя в интерфейсе командной строки предназначены командные процессоры (интерпретаторы команд).

В Unix и Linux системах существует несколько командных процессоров. В ОС Ubuntu интерпретатором команд по умолчанию является командный процессор **bash** (Bourne Again Shell), наиболее распространённый интерпретатор имеющий удобный интерфейс.

При отсутствии графического интерфейса, зайдя в систему, обычно сразу попадают в интерпретатор команд.

При использовании графического режима самый распространённый способ доступа к интерпретатору – использование окна терминала

2.3.1 Окна терминала

Чтобы открыть окно терминала при использовании Unity, надо выбрать значок Dashboard (Панель управления), а затем начать печатать текст в поле поиска. Найти появившийся на панели управления значок **Терминал** и выбрать его. После этого откроется окно **gnome-terminal**, в котором будет приглашение интерпретатора команд bash.

Чтобы получить информацию о *текущим интерпретаторе* команд, необходимо ввести команду **echo $SHELL** .

Для получения информации о *текущем пользователе (открывшим окно),* необходимо ввести команду **whoami,**

Для получения информации о *текущем каталоге,* надо ввести команду **pwd**

Имя пользователя и имя хоста отображаются в строке заголовка и приглашении.

Окно **gnome-terminal** позволяет получить доступ к интерпретатору команд, и предоставляет управление интерпретаторами команд.

Примеры:

- Выбрать в строке меню (подвести курсор к верхней строчке стола).**Файл ► Открыть вкладку**, чтобы открыть другой интерпретатор команд на новой вкладке;
- Выполнить команду **Файл ► Открыть терминал**, чтобы открыть новое окно терминала;
- Выбрать **Терминал ► Задать заголовок**, чтобы задать новый заголовок в строке заголовка.

Для работы с окном терминала можно использовать сочетания клавиш:

- открыть интерпретатор команд на новой вкладке, нажав **Shifts Ctrl+T**;
- открыть новое окно терминала с помощью **Shift+Ctrl+N**;
- закрыть вкладку, нажав **Shift-Ctrl-W**;
- закрыть окно терминала посредством **Shift+Ctrl-Q.**
- выделить текст и скопировать его, нажав **Shift-Ctrl-C**, а затем вставьте его в то же самое или другое окно, нажав **Shift-Ctrl-V** либо среднюю кнопку мыши.

Другие сочетания клавиш для управления окнами терминала предусматривают использование **F11** для переключения окна в *полноэкранный режим* и обратно.

- Нажать **Ctrl+Shift**++, чтобы **увеличить** (сделать текст крупнее), или **Ctrl+-** (это Ctrl и знак минуса), чтобы **уменьшить** (сделать текст мельче).
- **Переключаться между вкладками** можно с помощью **Ctrl+Page Up** и Ctrl+Page Down (соответственно предыдущая и следующая вкладка), либо с использованием **Alt+1, Alt+2, Alt+3** и т. д. для перехода на первую, вторую, третью (и т. д.) вкладку.
- Нажать **Ctrl+D**, чтобы выйти из интерпретатора команд, что

приведет к закрытию текущей вкладки или всего окна терминала (если соответствующая вкладка окажется последней).

Окно gnome-terminal также поддерживает профили (выбрать **Редактировать ► Профили**). Одни настройки профилей — косметические (позволяют выделять текст жирным шрифтом, обеспечивать мерцание курсора, звуковые сигналы терминала, использовать цвета, изображения и прозрачность). Другие настройки — функциональные. Например, по умолчанию терминал позволяет прокручивать 512 строк, сохраняя их в памяти. Некоторые пользователи хотят прокручивать еще дальше и согласны выделить на это больше памяти.

Если запускать gnome-terminal вручную, то можно добавить параметры.

Примеры:

- Запустить терминал с тремя открытыми вкладками:

$ gnome-terminal --tab --tab --tab

- Запустить терминал размером 80 символов на 20 строк:

  $ gnome-terminal --geometry 80x20

- Запустить терминал с более крупным шрифтом:

  $ gnome-terminal --zoom=2

Помимо окна gnome-terminal, есть окна множества других терминалов:

xterm (базовый эмулятор терминала, сопутствующий системе X Window System),

aterm (эмулятор терминала, созданный по образцу эмулятора Afterstep XVT VT l02)

konsole (эмулятор терминала, поставляемый вместе с рабочим столом KDE).

Проект рабочего стола Enlightenment предлагает терминал **eterm,** который включает такие функции, как журналы сообщений на фоне экрана.

2.3.2 Конфигурирование интерпретатора команд

После открытия интерпретатора команд, его среда конфигурируется исходя из того, какой пользователь запустил этот интерпретатор команд. Настройки

интерпретатора команд **bash** для интерпретаторов команд всех пользователей располагаются в нескольких файлах. Можно сделать так, чтобы собственные версии этих файлов переопределяли системные настройки. Эти настройки содержатся в файлах двух типов - **файлах запуска** и **файлах инициализации.**

bash задействует файлы запуска в случае с любым интерпретатором команд, который является тем, что назначается при входе в систему. Эти файлы определяют настройки, которые будут применяться при каждом входе в систему

bash задействует файлы инициализации в случае с интерпретаторами команд, которые работают в интерактивном режиме, то есть без запуска сценария интерпретатора команд.

bash применяет общесистемный файл запуска **/etc/profile**, а также некоторые файлы с точкой из домашнего каталога пользователя для индивидуальных настроек (при наличии таковых): **.bash_profile**, **.bash_login** и **.profile**. Он также использует расположенные в каталоге **/etc/profile.d** сценарии, имена которых оканчиваются на **. sh**.

bash задействует общесистемный файл инициализации **/etc/bash.bashrc**, а также файл **.bashrc** из домашнего каталога (для индивидуальных настроек). Эти файлы задействуются при каждом новом открытии интерпретатора команд bash.

При выходе из интерпретатора команд, назначаемого при входе в систему (например, из виртуальной консоли), будут выполнены все команды, указанные в файле **~/. bash_logout**. Модификация настроек в этих файлах приведет к перманентному изменению настроек интерпретатора команд пользователя, однако не повлияет на интерпретаторы команд, которые уже запушены (прочие интерпретаторы команд используют другие конфигурационные файлы).

Есть множество способов, посредством которых можно просматривать и изменять среду интерпретатора команд.

2.3.3 История bash

В bash, как и в других интерпретаторах команд, имеется такая встроенная функция, как история, позволяющая вам просматривать, изменять и повторно

использовать команды, которые выполнялись ранее.

Она может оказаться очень полезной, поскольку многие команды Linux длинные и сложные.

При запуске bash он считывает файл **~/.bash_history** и загружает его в память. Для этого файла задается значение $HISTFILE.

Во время сеанса bash команды добавляются в историю, которая содержится в памяти. При выходе из bash история, находящаяся в памяти, записывается обратно в файл **.bash_hi story**.

Количество команд, размещаемых в истории во время сеанса bash, задается посредством **$HISTFILE**, а количество команд, фактически сохраняемых в файле истории, - посредством **$HISTFILESIZE**;

$ echo $HISTFILE $HISTSIZE $HISTFILESIZE

Чтобы просмотреть всю историю, необходимо ввести команду **history**. Для отображения определенного количества предыдущих команд в истории необходимо указать после **history** нужное число.

Для перемещения среди команд в истории используются клавиши «**вверх**» и «**вниз**». Как только команда отобразится на экране, можно воспользоваться клавиатурой для редактирования текущей команды так же, как и любой другой, посредством клавиш «**влево**», «**вправо**», **Delete, Backspace** и т. д.

2.3.4 Создание простых сценариев интерпретатора команд

Сценарии интерпретатора команд подходят для автоматизации повторяющихся задач.

bash и другие интерпретаторы команд включают базовые конструкции, встречающиеся в языках программирования, например циклы, проверки, операторы case и т. д. Основное отличие состоит в том, что в случае с интерпретаторами команд имеют место переменные только одного типа - строковые.

Сценарии интерпретатора команд представляют собой простые текстовые файлы, содержащие команды, функции, псевдонимы или любые

другие компоненты, выполнение которых можно запустить из интерпретатора команд. Преимущество сценария интерпретатора команд заключается в том, что его можно использовать для формирования целого набора команд, благодаря чему их можно легко применять снова (без необходимости заново печатать) или даже выполнять автоматически.

Можно создавать сценарии интерпретатора команд, используя любой текстовый редактор (например, vi). Чтобы запустить файл сценария интерпретатора команд, этот файл должен быть выполняемым.

К примеру, если создадаётся сценарий интерпретатора команд с файловым именем **myscript.sh**, то можно сделать его выполняемым так:

$ chmod u+x myscript.sh

Кроме того, первая строка ваших сценариев bash всегда должна иметь следующий вид:

#!/bin/bash

Знак # в данном случае означает начало комментария.

Синтаксис **#!** выступает в качестве комментария для интерпретаторов команд, которые не понимают этот особый синтаксис.

Часть **/bin/bash** сообщает функционирующему интерпретатору команд, будь то bash или другой, какую программу следует использовать для запуска сценария (поскольку исторически сложилось, что не все системы включали в себя bash, часто /bin/bash используют в роли команды для запуска сценария)

Как и в случае с любой командой, помимо необходимости быть выполняемым, создаваемый сценарий интерпретатора команд также должен быть либо указан в **PATH,** либо идентифицироваться посредством своего полного или относительного пути при запуске.

3 Выполнение работы

3.1 *Задание*

1. Создать виртуальную машину в среде менеджера виртуальных машин **Oracle VM VirtualBox**.
2. Установить операционную систему **Ubuntu**
3. Познакомиться с графической оболочкой и интерпретатором команд операционной системы
4. Создать текстовый файл и файл сценария вывода содержимого файла
5. Проверить выполнение сценария

3.2 *Порядок выполнения работы*

- Войти в систему под учётной записью s**tud_XX, где XX -** индекс группы. Пароль **studXX**
- Запустить программу менеджера виртуальных машин **Oracle VM VirtualBox**.

- *Удалить установленные раннее виртуальные машины:*

Машины - Удалить

- *Выполнить настройку папки для установки виртуальных машин:*

Файл – Настройки – Папка для машин

d:\Users\studXX\VirtualBox VMs

3.2.1 Создание виртуальной машины Ubuntu

Машины – Создать

Имя: **Ubuntu**

Тип**: Linux**

Версия: **Ubuntu (64 bit)**

Объём памяти **1024 МБ**

Создать новый виртуальный жёсткий диск

Тип **VDI (VirtualBox Disk Image)**

Формат хранения **Динамический виртуальный жёсткий диск**

Размер файла **10,00 ГБ**

3.2.2 Настройка виртуальной машины Ubuntu

1. Подключить образ оптического диска с дистрибутивом операционной системы:

Настроить - Носители

Контроллер IDE

Привод: Первичный мастер IDE

Выбрать файл образа D**:\ОС\ДИСТРИБУТИВЫ\ubuntu-16.04-desktop-i386.iso**

2. Установить порядок загрузки системы

Настроить - Система

Порядок загрузки: **CD/DVD**

Жёсткий диск

3.2.3 Установка операционной системы Ubuntu

1. Запустить виртуальную машину **Ubuntu**

2. Установить операционную систему Ubuntu с дистрибутива

Выбрать язык **Русский**

Установить Ubuntu

Скачать обновления при установке

Стереть диск и установить Ubuntu

Ваше имя : **student**

Пароль: **student**

Входить в систему автоматически

3.2.4 Работа с ОС Ubuntu

1. Запустить виртуальную машину **Ubuntu**

2. Ознакомиться с основными элементами графической оболочки Unity и освоить основные приёмы запуска программ.

3. Прикрепить к панели быстрого запуска кнопки запуска программ **Терминал** и **Текстовый редактор**

4. Открыть интерпретатор команд в окне программы **Терминал**

5. Познакомиться с синтаксисом и выполнением команд интерпретатора

6. Создать в домашнем каталоге текстовый файл содержащий **ФИО студента и учебную группу**

7. Написать сценарий интерпретатора команд, который выводит на экран содержимое файла. Имя выводимого файла должно задаваться в параметре сценария. Перед выводом содержимого файла необходимо напечатать заголовок, содержащий имя выводимого файла, текущую дату и время.

8. Создать файл сценария с помощью команды **touch** и текстового редактора **vi**

touch /TMP/script.sh

vi /TMP/script.sh

Первая строка сценария **bash** должна иметь следующий вид:

#! /bin/bash

После создания файла необходимо сделать его выполнимым:

chmod u+x /tmp/myscript.sh

и проверить выполнение сценария:

/tmp/myscript.sh

4 Контрольные вопросы

1. В чём различие между физическими и виртуальными ресурсами?
2. Что такое виртуальная машина?

3. В чём отличие мультипрограммных систем и систем виртуальных машин?
4. Для чего предназначена программа VirtulBox?
5. Как может быть установлена гостевая операционная система?

5 ЛИТЕРАТУРА

1. Э. Таненбаум. Современные операционные системы. 3-е изд – Спб.: Питер, 2010, 116 с.: ил.

2. Х.М. Дейтел, П. Дж. Дейтел, Д.Р. Чофнес Операционные системы. Часть 1. Основы и принципы: Третье издание. Пер. с англ. – М.: ООО «Бином-Пресс», 2011 г. – 1024 с.: ил.

3. Робачевский А.М. Операционная система UNIX.-СПб.: БХВ-Петербург, 2001. – 528 с.:ил.

4. Негус К. Ubuntu и Debian Linux для продвинутых. 2-е изд. – СПб.: Питер,2014. -384 с.: ил.

6 Приложение

6.1 *Основные команды BASH:*

help - справка о встроенных командах BASH.

help help – о том как получать справку.

man – (manual) вызов справки команд (man –help , man <команда>).

info – информация о командах (info <команда>).

bash – запуск нового командного интерпретатора .

uname -a – показать тип ОС и характеристики (все ключи -animporsv).

who и w – показать подключенных пользователей

env – показать переменные среды

echo - вывести аргументы командной строки на стандартный вывод (-n не выводится конец строки)

PS1=”>>” – изменение подсказки в командной строке (новая “>>”).

alias – назначение алиасов командам (можно командам с параметрами)

unalias – удаление алиасов из списка

cal/ncal – вызов календаря

date – вызов текущей даты

clear – очистка страницы экрана командной строки

exit – завершение текущего командного интерпретатора или скрипта.

free – показать информацию о занятой памяти.

write – посылка сообщения другому пользователю

vi – вызов текстового редактора

ps – список процессов в системе

kill <pid> - завершение работы процесса с номером <pid>

6.2 *Редактор vi для текстового терминала*

Редактор vi используется при работе в текстовом режиме - т. е. d текстовом терминале или в программе-эмуляторе терминала типа gnome-terminal, xterm, dtterm или putty (в системах Windows).

Редактор vi может находиться в одном из двух режимов - **командном режиме** или **режиме ввода**.

При запуске редактор начинает работу в командном режиме. В этом режиме все, что набирают на клавиатуре, интерпретируется как команда. Команды в vi короткие - почти все состоят из одной или двух букв. Некоторые команды редактора vi отображаются в служебной строке. Служебной строкой в vi считается последняя строка экрана. Большинство команд редактора никак не отображаются, хотя выполняются. Так можно легко стереть или модифицировать текст неожиданным образом. Помните: не надо случайных нажатий на клавиши в командном режиме!

Vi отображает вводимую команду в служебной строке, если это команда поиска или если команда вводится в режиме совместимости с редактором ed. При

отображении чего-либо в служебной строке экрана строка файла, которая раньше показывалась в этой строке, никак не меняется.

Для перехода в **режим ввода** надо выполнить команду **a, i** или **o**. После этого можно будет начинать ввод текста. Текст начнет вводиться в позиции курсора, в позиции, следующей за позицией курсора, или в начале новой строки, которая появится под текущей строкой, соответственно.

В режиме ввода все, что набирается на клавиатуре, за исключением клавиши Esc, интерпретируется как набираемый текст. **Esc** - это переход из режима ввода в режим команд.

В режиме ввода не всегда можно пользоваться клавишами передвижения по тексту (стрелками, **PgUp. PgDn. Home, End** и т. д.).Если терминал настроен не совсем корректно, то нажатие, например, клавиши «стрелка вверх», Vi может воспринять как нажатие Esc. Поэтому может случиться так. что, нажав стрелку вверх в режиме ввода, вы незаметно для себя перейдете в режим команд. Если вы продолжите вводить текст, он будет воспринят как команда. Таким образом, иногда можно передвигаться по тексту в режиме ввода, но лучше сначала выйти в командный режим, затем перейти по тексту в нужное место и снова перейти в режим ввода.

Переход по тексту в командном режиме выполняется клавишами передвижения по тексту (стрелками, PgUp, PgDn, Home, End), а также Ctrl+F (forward, вперед на страницу). Ctrl-B (backward, назад на страницу).

Все команды, начинающиеся с двоеточия, - это команды режима совместимости с редактором ed. Ввод двоеточия интерпретируется vi как переход к этому режиму. Как только вводят двоеточие, оно отобразится в начале служебной строки, и оставшуюся часть команды набирают, видя ее в этой строке. В качестве служебной строки используется последняя строка экрана.

Перед любой командой vi, за исключением команд, начинающихся с двоеточия, можно набрать число, которое будет интерпретироваться как требование повторить идущую за ним команду это число раз.

Например:

15dd означает: вырезать в буфер пятнадцать строк, начиная с текущей.

Важная часть команд - команды поиска и замены. Обычный поиск выполняется командой **/образец**, - т. е. знак «слэш», за которым следует образец для поиска. При поиске назад по файлу используется вопросительный знак вместо слэша. Продолжение поиска - слэш или вопросительный знак без образца. Если в образце встретится слэш или вопросительный знак, vi сочтет их ограничителем образца и проигнорирует остаток образца.

Поиск и замена выполняется командой

:s/образеw/на_что_менять/

В ней указывается образец для поиска и строка, которой следует его заменить. После завершающего слэша может стоять модификатор g. Если его не поставить заменен будет только первый образец в строке. Если в строке есть еще подстроки, отвечающие образцу, они останутся нетронутыми. Если поставить модификатор g. будут заменены все полстроки, отвечающие образцу.

Все команды режима совместимости с редактором ed могут быть предварены выражением n,m, где n,m — номера строк, ограничивающих диапазон выполнения команды. Если такого выражения нет, команда выполняется для текущей строки. Например:

:23,33s/black/white/g

означает, что нужно заменить все вхождения black на white во всех строках с 23 по 33 включительно. В выражении, описывающем диапазон, допустимы числа и символы «.» и «$», а также арифметические выражения. Символ *«.»* обозначает текущую строку, символ «$» — последнюю строку файла. Напрмер:

:.,.+7s/black/white/g

означает замену black на white в восьми строках, начиная с текущей. Допустим, мы находимся в первой строке. Тогда выражение

., .+7

фактически означает

1,1+7

т. е. 1,8

Команда

:.,$s/yellow/blue/g

означает замену yellow на blue во всех строках, начиная с текущей и до конца файла.

В режиме совместимости можно не только выполнять команды поиска и замены. Здесь есть еще команда удаления - **d.**

Команда **:1, .d** удалит все строки с первой по текущую.

Диапазон может состоять из одной строки.

Например, команда **:4d** означает требование удалить четвертую строку.

Ввод числа за двоеточием без всяких команд означает переход к строке с указанным номером, например :**56** вызовет переход к 56-й строке.

6.3 *Основные команды редактора vi*

	Начало ввода
a	перейти в режим ввода, начать ввод в позиции, следующей за позицией курсора
i	перейти в режим ввода, начать ввод в позиции курсора
0	перейти в режим ввода, добавить пустую строку под текущей строкой и начать ввод в новой строке
	Операции с буфером
dd	вырезать текущую строку в буфер
ч	вырезать текущий символ в буфер
x	отменить последнее действие
yy	копировать текущую строку в буфер
p	вставить строку из буфера под текущей
	Замена символа
rn	заменить символ в позиции курсора на п
	Перемещение по тексту
0	перейти в начало строки
$	перейти в конец строки
j	перейти на строку вниз
k	перейти на строку вверх
h	перейти на символ влево
I	перейти на символ вправо
%	курсор стоит в позиции символа «скобка» (круглая, квадратная, угловая или фигурная) перейти в позицию соответствующей второй
Ctrl+G	показать текущую позицию (номер строки) в файле
	Поиск и замена
/	поиск вперед
?	поиск назад
:s/что/на что/[g]	поиск и замена
	Запись и завершение редактирования
ZZ	выход из редактора, запись файла
:q	выйти из редактора
:q!	выйти без записи
:w	записать изменения
:wq	записать и выйти

:w!	записать, даже если нет права записи в файл (срабатывает, только если выполнена от имени root)
:w имя файла	записать в файл с другим именем; в дальнейшем будет считаться, что редактируется именно этот файл с другим именем
Esc	переход в командный режим