

МИНИСТЕРСТВО ОБРАЗОВАНИЯ КРАСНОЯРСКОГО КРАЯ

**КРАЕВОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«КРАСНОЯРСКИЙ КОЛЛЕДЖ ОТРАСЛЕВЫХ ТЕХНОЛОГИЙ
И ПРЕДПРИНИМАТЕЛЬСТВА»**

РАССМОТРЕНО

методической комиссией
протокол № 6 от «20» июня 2024 г.

УТВЕРЖДЕНО

Директор КГБПОУ «Красноярский колледж
отраслевых технологий и
предпринимательства»

_____/Н. В. Журова/
Приказ № 01-60-2П от «01» июля 2024 г.

**АДАптиРОВАННАЯ ПРОГРАММА ПОДГОТОВКИ
СПЕЦИАЛИСТОВ СРЕДНЕГО ЗВЕНА**

09.02.07 Информационные системы и программирование

на базе среднего общего образования

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ПРОВЕДЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ И ЛАБОРАТОРНЫХ РАБОТ**

ОП.01 Операционные системы и среды

Зам. директора по УР _____ / Е.В. Миля /
подпись

Красноярск 2024

Организация-разработчик: КГБПОУ «Красноярский колледж отраслевых технологий и предпринимательства»

Разработчики: Лавренков Семен Сергеевич, преподаватель, преподаватель

СОДЕРЖАНИЕ

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
2. ОБЩИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ И ОФОРМЛЕНИЮ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ
3. МЕТОДИКА ПРОВЕДЕНИЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ И ЛАБОРАТОРНЫХ РАБОТ
4. СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ

1.ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания к проведению практических занятий и лабораторных работ по учебной дисциплине ОП.01 Операционные системы и среды предназначены для обучающихся СПО по специальности 09.02.07 Информационные системы и программирование.

Уровень профессиональной подготовки по специальности 09.02.07 Информационные системы и программирование, определяемый ФГОС СПО, предусматривает владение практическими навыками выбора материалов для профессиональной деятельности.

Для успешного выполнения и защиты работы обучающиеся должны иметь:

умения	– управлять параметрами загрузки операционной системы; – выполнять конфигурирование аппаратных устройств; – управлять учетными записями, настраивать параметры рабочей среды пользователей; – управлять дисками и файловыми системами, настраивать сетевые параметры, управлять
знания	– основные понятия, функции, состав и принципы работы операционных систем; – архитектуры современных операционных систем; – особенности построения и функционирования семейств операционных систем «Unix» и «Windows»; – принципы управления ресурсами в операционной системе; – основные задачи администрирования и способы их выполнения в изучаемых операционных системах.

Выполнение лабораторных работ способствует формированию и развитию общих и профессиональных компетенций:

Код	Наименование общих компетенций
ОК 1	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 2	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности
ОК 5	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста
ОК 9	Использовать информационные технологии в профессиональной деятельности
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языке

Код	Наименование профессиональных компетенций
ПК 6.4	Оценивать качество и надежность функционирования информационной системы в соответствии с критериями технического задания.
ПК 6.5	Осуществлять техническое сопровождение, обновление и восстановление данных информационной системы в соответствии с техническим заданием
ПК 7.2	Осуществлять администрирование отдельных компонент серверов
ПК 7.3	Формировать требования к конфигурации локальных компьютерных сетей и серверного оборудования, необходимые для работы баз данных и серверов
ПК 7.5	Проводить аудит систем безопасности баз данных и серверов с использованием регламентов по защите информации

Формируемые личностные результаты в ходе освоения общеобразовательной дисциплины: ЛР 03, ЛР 04 ЛР 06, ЛР 07, ЛР 10, ЛР 11, ЛР 13, ЛР 14, ЛР 15, ЛР 16, ЛР 17.

2. ОБЩИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ И ОФОРМЛЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторные работы выполняются обучающимися по графику, составленному в соответствии с рабочей программой учебной дисциплины ОП.01 Операционные системы и среды.

Учебная дисциплина ОП.01 Операционные системы и среды зависит от содержания лабораторных работ, которые соответствуют более глубокому освоению дисциплины, закреплению теоретических знаний и прививают обучающимся практические навыки самостоятельной работы.

Задача лабораторных работ– закрепить теоретические знания обучающихся.

Согласно учебного плана по специальности и программы учебной дисциплины на лабораторные (практические) занятия обучающихся выделено 18 академических часов, из них:

№ и наименование разделов (тем)	Количество часов
Тема 1 Установка и настройка операционной системы Linux	4
Тема 2 Управление процессами в ос Linux	2
Тема 3. Структура операционной системы Windows XP	3
Тема 3. Управление памятью и вводом/выводом в ОС	3
Тема 4. Ознакомление с сетевыми функциями операционной системы	3
Тема 4. Установка виртуальной компьютерной сети на основе операционных систем Windows	3
Итого:	18

3. МЕТОДИКА ПРОВЕДЕНИЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Целью практических занятий является приобретение практических навыков работы с ПК, а именно работы с программами MS Office и работа с интернет ресурсами.

Целью лабораторных работ является отработка обучающимися практических навыков по созданию и заполнению путевых листов, технологических карт и оформлению презентаций, изучению специализированных программ и работа в сети Интернет.

Исходя, из поставленных целей в работе будут решаться следующие задачи:

Закрепление знаний по:

1. основные понятия, функции, состав и принципы работы операционных систем;
2. архитектуры современных операционных систем;
3. особенности построения и функционирования семейств операционных систем «Unix» и «Windows»;
4. принципы управления ресурсами в операционной системе;
5. основные задачи администрирования и способы их выполнения в изучаемых операционных системах.

Ознакомиться:

1. с приемами создания, обработки и преобразованию информации;
2. с прикладными возможностями телекоммуникационных технологий и ИКТ;
3. с приемами описания и моделирования информационных процессов.

При выполнении лабораторной работы формируются навыки:

1. работы с прикладным программным обеспечением;
2. организации взаимодействия в локальной сети;
3. поиска информации в сети Internet;
4. презентации собственной деятельности и результатов работы.

Научиться пользоваться:

1. прикладным программным обеспечением;
2. ресурсами локальной и глобальной сети.

4. СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа «Установка и настройка операционной системы Linux» (4 часа)

Цель занятия: Приобрести опыт установки операционной системы Linux

ИНСТРУКЦИОННАЯ КАРТА

Материально - техническое оснащение

Оборудование:

Аппаратная часть: персональный компьютер, сетевой или локальный принтер.

Программная часть: программа VirtualBox, установочный диск либо образ диска с ОС Linux Runtu Light на базе Ubuntu 12.04.

Характер выполнения работы:

1. Закрепить знания о работе с программой VirtualBox.

2. Создать виртуальную машину исходя из предоставленной информации о минимальных аппаратных требованиях предлагаемой к установке и изучению операционной системы (ОС).
3. Установить ОС на виртуальный компьютер. Разобрать процесс установки ОС на этапы.
4. Познакомиться с основными группами программ входящих в состав ОС.

Практические задания:

Системные требования Runtu Light 12.04:

- Процессор: Pentium 3, 700MHz;
- Оперативная память: 256 Mb (32-bit)
- Свободное дисковое пространство: 3 Гбайт HDD + 256 Мбайт для swap.
- Видеоадаптер: 64 MB памяти;
- Устройство чтения DVD-дисков.

Создадим виртуальную машину, учитываем тип операционной системы, а также минимальные системные требования.

Загружаем предлагаемый образ для установки Linux Ubuntu

Контрольные вопросы:

1. Что такое Linux?
2. Что такое дистрибутив?
3. Перечислите основные дистрибутивы Linux. Объясните в чем их отличие?
4. Какую файловую систему использует для работы установленный Вами дистрибутив?
5. Перечислите основные этапы установки операционной системы?

Критерии оценивания отчета:

Оценка «5» ставится, если учащийся выполняет работу в полном объеме с соблюдением необходимой последовательности проведения опытов и измерений; самостоятельно и рационально монтирует необходимое оборудование; все опыты проводит в условиях и режимах, обеспечивающих получение правильных результатов и выводов; соблюдает требования правил безопасности труда; в отчете правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ погрешностей.

Оценка «4» ставится, если выполнены требования к оценке «5», но было допущено два - три недочета, не более одной негрубой ошибки и одного недочёта.

Оценка «3» ставится, если работа выполнена не полностью, но объем выполненной части таков, позволяет получить правильные результаты и выводы: если в ходе проведения опыта и измерений были допущены ошибки.

Оценка «2» ставится, если работа выполнена не полностью и объем выполненной части работы не позволяет сделать правильных выводов: если опыты, измерения, вычисления, наблюдения производились неправильно.

Лабораторная работа «Управление процессами в ос Linux» (2 часа)

Цель занятия: приобрести практический опыт запуска, отслеживания выполнения и завершения процессов в современной операционной системе Windows с использованием Диспетчера задач и командного интерпретатора Windows; приобрести практические навыки работы с системным реестром.

ИНСТРУКЦИОННАЯ КАРТА

Материально - техническое оснащение

Оборудование: Аппаратная часть: персональный компьютер с правами администратора. Программная часть: программа VirtualBox, виртуальная машина с установленной ОС Windows 8/10, текстовый процессор Microsoft Word.

Теоретическая часть. Лабораторная работа посвящена процессам операционной системы Linux. Поскольку администрирование операционной системы в конечном счете сводится к управлению процессами. Каждый раз, когда вы запускаете на выполнение программу, вы начинаете то, что в литературе именуется как *процесс*. Или другими словами – процессом называется выполняемая в данный момент программа или ее потомки. Каждый процесс запускается от имени какого-то пользователя. Процессы, которые стартовали при загрузке, обычно выполняются от имени пользователей root или nobody.

Каждый пользователь может управлять поведением процессов, им запущенных. При этом пользователь root может управлять всеми процессами — как запущенными от его имени, так и процессами, порожденными другими пользователями операционной системы. Управление процессами осуществляется с помощью утилит, а также посредством некоторых команд командной оболочки (shell).

Каждый процесс в системе имеет уникальный номер — идентификационный номер процесса (Process Identification, PID). Этот номер используется ядром операционной системы, а также некоторыми утилитами для управления процессами.

Выполнение процесса на переднем плане и в фоновом режиме

Процессы могут выполняться на переднем плане (foreground) — режим по умолчанию и в фоновом режиме (background). На переднем плане в каждый момент да текущего терминала может выполняться только один процесс. Однако пользователь может перейти в другой виртуальный терминал и запустить на выполнение еще один процесс, а на другом терминале еще один и т. д. Процесс переднего плана – это процесс, с которым вы взаимодействуете, он получает информацию с клавиатуры (стандартный ввод) и посылает результаты на ваш экран (стандартный вывод).

Фоновый процесс после своего запуска благодаря использованию специальной команды командной оболочки отключается от клавиатуры и экрана, т. е. не ожидает ввода данных со стандартного ввода и не выводит информацию на стандартный вывод, а командная оболочка не ожидает окончания запущенного процесса, что позволяет пользователю немедленно запустить еще один процесс.

Обычно фоновые процессы требуют очень большого времени для своего завершения и не требуют вмешательства пользователя во время существования процесса. К примеру, компиляция программ или архивирование большого объема информации — кандидаты номер один для перевода процесса в фоновый режим.

Процессы так же могут быть отложенными. Отложенный процесс — это процесс, который в данный момент не выполняется и временно остановлен. После того как вы остановили процесс, в дальнейшем вы можете его продолжить как на переднем плане, так и в фоновом режиме. Возобновление приостановленного процесса не изменит его состояния — при возобновлении он начнется с того места, на котором был приостановлен.

Для выполнения программы в режиме переднего плана достаточно просто набрать имя программы в командной строке и запустить ее на выполнение. После этого вы можете работать с программой.

Для запуска программы в качестве фонового процесса достаточно набрать в командной строке имя программы и в конце добавить знак амперсанта (&), отделенный пробелом от имени программы и ее параметров командной строки, если таковые имеются. Затем программа запускается на выполнение. В отличие от запуска программы в режиме переднего плана мы получим приблизительно следующее сообщение:

```
/home/student# yes > /dev/null &  
[1] 123  
/home/student#
```

Оно состоит из двух чисел и приглашения командной строки. Таким образом, мы запустили программу выполняться в фоновом режиме и получили возможность запустить с той же самой консоли на выполнение еще какую-то программу.

Число [1] означает номер запущенного нами фонового процесса. Как вы узнаете несколько позже, с его помощью можно будет производить манипуляции с нашим фоновым процессом.

Значение 123 показывает идентификационный номер (PID) нашего процесса. Отличия этих двух чисел достаточно существенные. Номер фонового процесса уникален только для пользователя, запускающего данный фоновый процесс. То есть если у нас в системе три пользователя решили запустить фоновый процесс (первый для текущего сеанса) — в результате у каждого пользователя появится фоновый процесс с номером [1]. Напротив, идентификационный номер процесса (PID) уникален для всей операционной системы и однозначно идентифицирует в ней каждый процесс. Спрашивается, для чего тогда вводить нумерацию фонового процесса для пользователя? Для удобства. Номер фонового процесса хранится в переменных командной оболочки пользователя и позволяет не забивать голову цифрами типа 2693 или 1294, а использовать переменные вида %1, %2. Однако допускается пользоваться и идентификационным номером процесса.

Для проверки состояния фоновых процессов можно воспользоваться командой командной оболочки — jobs.

```
/home/student# jobs
[1]+  Running                  yes >/dev/null &
/home/student#
```

Из вышеприведенного примера видно, что у пользователя в данный момент запущен один фоновый процесс, и он выполняется.

Остановка и возобновление процесса

Помимо прямого указания выполнять программу в фоновом режиме, существует еще один способ перевести процесс в фоновый режим. Для этого мы должны выполнить следующие действия:

Запустить процесс выполняться на переднем плане.

Остановить выполнение процесса.

Продолжить процесс в фоновом режиме.

Для выполнения программы введем ее имя в командной строке и запустим на выполнение. Для остановки выполнения программы необходимо нажать на клавиатуре следующую комбинацию клавиш — <Ctrl>+<Z>. После этого вы увидите на экране следующее:

```
/home/student# yes > /dev/null
ctrl+Z
[1]+  Stopped                  yes >/dev/null
/home/student#
```

Мы получили приглашение командной строки. Для того чтобы перевести выполнение процесса в фоновый режим, необходимо выполнить следующую команду:

```
bg %1
```

Причем необязательно делать это сразу после остановки процесса, главное правильно указать номер остановленного процесса.

Для того чтобы вернуть процесс из фонового режима выполнения на передний план, достаточно выполнить следующую команду:

```
fg %1
```

В том случае, если вы хотите перевести программу в фоновый или, наоборот, на передний план выполнения сразу после остановки процесса, можно выполнить соответствующую программу без указания номера остановленного процесса.

Существует большая разница между фоновым и остановленным процессом. Остановленный процесс не выполняется и не потребляет ресурсы процесса, однако занимает оперативную память или пространство свопинга. В фоновом же режиме процесс продолжает выполняться.

Как остановить выполнение фонового процесса? Использование комбинации клавиш <Ctrl>+<Z> не поможет, поскольку процесс находится в фоновом режиме и не реагирует на ввод данных с консоли. Для решения этой проблемы следует переместить процесс на передний план, а затем остановить.

Завершение работы процесса

Ну вот, вы научились запускать и останавливать выполнение процессов, а также переводить исполняемый процесс в фоновый режим и в режим переднего плана. Однако вы, возможно, не умеете завершать работу процесса.

Вариант первый. Если процесс интерактивный, как правило, в документации или прямо на экране написано, как корректно завершить программу.

Вариант второй. В том случае, если вы не знаете, как завершить текущий процесс (не фоновый), можно воспользоваться клавиатурной комбинацией `<Ctrl>+<C>`. Попробуйте также комбинацию клавиш `<Ctrl>+<Break>`. А для остановки фонового процесса можно перевести его на передний план, а затем уже воспользоваться вышеприведенными клавиатурными комбинациями.

Вариант третий и самый действенный. В том случае, если вам не удалось прекратить выполнение процесса вышеприведенными способами – например, программа зависла или "слетел" терминал — для завершения процесса можно воспользоваться следующими командами: `kill`, `killall`.

Команда `kill` может получать в качестве аргумента как номер процесса, так и идентификационный номер (PID) процесса. Таким образом, команда:

```
/home/student# kill 123
```

 эквивалентна команде:

```
/home/student# kill %1
```

Можно видеть, что не надо использовать "%", когда вы обращаетесь к работе по идентификационному номеру (PID) процесса.

С помощью команды `killall` можно прекратить выполнение нескольких процессов сразу, имеющих одно и то же имя. Например, команда `killall mc` прекратит работу всех программ `mc`, запущенных от имени данного пользователя.

Для того чтобы завершить работу процесса, вам надо быть его владельцем. Это сделано в целях безопасности. Если бы одни пользователи могли завершать процессы других пользователей, открылась бы возможность исполнения в системе множества злонамеренных действий. Пользователь `root` может завершить работу любого процесса в операционной системе.

Программы, используемые для управления процессами

Существует достаточно большое количество утилит, используемых для управления тем или иным способом процессами, исполняемыми в операционной системе. Здесь мы рассмотрим только основные такие утилиты. В табл. 1 приведен список основных программ, тем или иным образом предназначенных для управления процессами.

Таблица 1. Программы управления процессами

Программа	Описание
<code>at</code>	Выполняет команды в определенное время
<code>batch</code>	Выполняет команды тогда, когда это позволяет загрузка системы
<code>cron</code>	Выполняет команды по заранее заданному расписанию
<code>crontab</code>	Позволяет работать с файлами <code>crontab</code> отдельных пользователей
<code>kill</code>	Прекращает выполнение процесса
<code>nice</code>	Изменяет приоритет процесса перед его запуском
<code>nohup</code>	Позволяет работать процессу после выхода пользователя из системы
<code>ps</code>	Выводит информацию о процессах
<code>renice</code>	Изменяет приоритет работающего процесса
<code>w</code>	Показывает, кто в настоящий момент работает в системе и с какими программами

nohup

Эта утилита позволяет организовать фоновый процесс, продолжающий свою работу даже тогда, когда пользователь отключился от терминала, в отличие от команды `&`, которая этого не позволяет. Для организации фонового процесса необходимо выполнить следующую команду:

`nohup выполняемая_фоновая_команда &`

Во вновь запущенном терминале процесс нельзя увидеть с помощью команды `jobs`, так как команда `jobs` выводит список процессов текущего терминала, поэтому после подключения к терминалу необходимо использовать команду `ps` с параметром `-A`.

ps

Программа `ps` предназначена для получения информации о существующих в операционной системе процессах. У этой команды есть множество различных опций, но мы остановимся на самых часто используемых. Для получения подробной информации смотрите man-страницу этой программы.

Простой запуск `ps` без параметров выдаст список программ, выполняемых на терминале. Обычно этот список очень мал:

PID	TTY	TIME	CMD
885	ttyl	00:00:00	login
893	ttyl	00:00:00	bash
955	ttyl	00:00:00	ps

Что означает полученная информация?

Первый столбец — `pid` (идентификационный номер процесса). Как уже упоминалось, каждый выполняющийся процесс в системе получает уникальный идентификатор, с помощью которого производится управление процессом. Каждому вновь запускаемому на выполнение процессу присваивается следующий свободный PID. Когда процесс завершается, его номер освобождается. Когда достигнут максимальный PID, следующий свободный номер будет взят из наименьшего освобожденного.

Следующий столбец — `tty` — показывает, на каком терминале процесс выполняется. Запуск команды без параметров `ps` покажет процессы, выполняемые на текущем терминале.

Столбец `time` показывает, сколько процессорного времени выполняется процесс. Оно не является фактическим временем с момента запуска процесса, поскольку Linux — это многозадачная операционная система. Информация, указанная в столбце `time`, показывает время, реально потраченное процессором на выполнение процесса.

Столбец `CMD` показывает, что же это за программа. Отображается только имя программы, опции командной строки не выводятся.

Для получения расширенного списка процессов, выполняемых в системе, используется следующая команда:

`ps -ax` (приведена часть распечатки)

PID	TTY	STAT	TIME	COMMAND
1	?	S	0:04	init
437	?	s	0:00	syslogd -m 0
442	?	s	0:00	klogd -2
885	tty1	s	0:00	login — root
1067	pts/0	R	0:00	ps -ax

Как можно видеть, список запущенных процессов в системе велик и достаточно сильно зависит от конфигурации операционной системы. Опции, заданные программе в этом примере, заставляют ее выводить не только имена программ, но и список опций, с которыми были запущены программы.

Появился новый столбец — `STAT`. В этом столбце отображается состояние (status) процесса. Полный список состояний вы можете прочитать в описании программы `ps`. Опишем самые важные состояния:

- буква R обозначает запущенный процесс, исполняющийся в данный момент времени;
- буква s обозначает спящий (sleeping) процесс — процесс ожидает какое-то событие, необходимое для его активизации;
- буква z используется для обозначения "зомбированных" процессов (zombied) — это процессы, родительский процесс которых прекратил свое существование, оставив дочерние процессы рабочими.

Обратите внимание на колонку TTY. Как вы, наверное, заметили, многие процессы, расположенные в верхней части таблицы, в этой колонке содержат знак "?" вместо терминала. Так обозначаются процессы, запущенные с более не активного терминала. Как правило, это всякие системные сервисы.

Если вы хотите увидеть еще больше информации о выполняемых процессах, попробуйте выполнить команду:

```
ps -aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	1.2	0.2	1412	520	9	S	14:51	0:04	init
root	o	0.0	0.0	0	0	9	SW	14:51	0:00	[keventd]
root	3	0.0	0.0	0	0	9	SW	14:51	0:00	[kapr-idled]
root	4	0.0	0.0	0	0	7	SWN	14:51	0:00	[ksoftirqd_CPU0]
root	5	0.0	0.0	0	0	9	SW	14:51	0:00	[kswapd]
root	6	0.0	0.0	0	0		SW	14:51	0:00	[kreclaimd]
root	7	0.0	0.0	0	0	?	SW	14:51	0:00	[bdflush]
root	8	0.0	0.0	0	0	9	SW	14:51	0:00	[kupdated]
root	9	0.0	0.0	0	0	9	SW<	14:51	0:00	[mdrecoveryd]
root	13	0.0	0.0	0	0	9	SW	14:51	0:00	[kjournald]
root	437	0.0	0.2	1472	592	9	s	14:52	0:00	syslogd -m 0
root	442	0.0	0.4	1928	1040	9	s	14:52	0:00	klogd -2
rpc	462	0.0	0.2	1552	588	9	s	14:52	0:00	portmap
rpcuser	490	0.0	0.2	1596	756	7	s	14:52	0:00	rpc.statd
root	590	0.0	0.2	1396	524	9	s	14:52	0:00	/usr/sbin/apmd -p
root	647	0.0	0.4	2676	1268	9	s	14:52	0:00	/usr/sbin/sshd
root	680	0.0	0.3	2264	992	7	s	14:52	0:00	xinetd -stayalive
lp	704	0.0	0.3	2600	1020	9	s	14:52	0:00	lpd Waiting
root	732	0.0	0.7	5296	1984	9	s	14:52	0:00	sendmail: accepti
root	751	0.0	0.1	1440	492	7	s	14:52	0:00	gpm -t ps/2 -m /d
root	769	0.0	0.2	1584	660	9	s	14:52	0:00	crond
xf	835	0.0	1.4	4988	3612	9	s	14:52	0:00	xf -droppriv -da
root	853	0.0	0.2	1416	600	7	s	14:52	0:00	anacron
daemon	871	0.0	0.2	1444	568	9	s	14:52	0:00	/usr/sbin/atd
root	885	0.0	0.4	2320	1076	ttyl	s	14:52	0:00	login — root
root	886	0.0	0.1	1384	448	tty2	s	14:52	0:00	/sbin/mingetty tt
root	887	0.0	0.1	1384	448	tty3	s	14:52	0:00	/sbin/mingetty tt
root	893	0.0	0.5	2464	1312	ttyl	s	14:52	0:00	-bash
root	1037	0.0	0.7	3284	1804	ttyl	R	14:56	0:00	/usr/bin/mc -P
root	1038	0.0	0.1	1380	348	7	s	14:56	0:00	cons.saver /dev/t
root	1039	0.0	0.5	2552	1392	pts/0	s	14:56	0:00	bash -rcfile .bas
root	1068	0.0	0.3	2780	824	pts/0	R	14:57	0:00	ps -aux

Как вы видите — информации прибавилось. Появились еще следующие столбцы:

- USER — показывает, от имени какого пользователя был запущен данный процесс;
- %CPU, %MEM — показывают, сколько данный процесс занимает соответственно процессорного времени и объем используемой оперативной памяти;
- TIME — время запуска программы.

В табл. 2 приведены некоторые параметры командной строки программы ps.

Таблица 2. Параметры командной строки ps

Ключ	Описание
a	Показать процессы всех пользователей
c	Имя команды из переменной среды
e	Показать окружение
f	Показать процессы и подпроцессы
h	Вывод без заголовка
j	Формат заданий
l	"Длинный" формат вывода
m	Вывод информации о памяти
n	Числовой вывод информации
r	Только работающие процессы
s	Формат сигналов
S	Добавить время использования процессора порожденными процессами
txx	Только процессы, связанные с терминалом xx
u	Формат вывода с указанием пользователя
v	Формат виртуальной памяти
w	Вывод без обрезки информации для размещения в одной строке
x	Показать процессы без контролирующего терминала

top

Еще одна утилита, с помощью которой можно получать информацию о запущенных в операционной системе процессах. Для использования достаточно просто запустить команду top на выполнение. Эта утилита выводит на экран список процессов в системе, отсортированных в порядке убывания значений используемых ресурсов.

```
2:55pm up 3 min, 1 user, load average: 0,06, 0,09, 0,03
32 processes: 31 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 1,1% user, 2,9% system, 0,0% nice, 95,8% idle
Mem: 255532K av, 42856K used, 212676K free, OK shrd, 8560K buff
Swap: 257000K av, OK used, 257000K free 19920K cached
```

PID	USER	PRI	N1	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	COMMAND
1	root	8	0	520	520	452	s	0,0	0,2	0:04	init
2	root	g	0	0	0	0	sw	0,0	0,0	0:00	keventd
3	root	9	0	0	0	0	sw	0,0	0,0	0:00	kapm-idled
4	root	19	19	0	0	0	SWN	0,0	0,0	0:00	ksoftirqd_CPU0
5	root	9	0	0	0	0	SW	0,0	0,0	0:00	kswapd
6	root	9	0	0	0	0	SW	0,0	0,0	0:00	kreclaimd
-j	root	9	0	0	0	0	sw	0,0	0,0	0:00	bdf flush
8	root	9	0	0	0	0	sw	0,0	0,0	0:00	kupdated
9	root	-1	-20	0	0	0	sw<	0,0	0,0	0:00	mdrecoveryd
13	root	9	0	0	0	0	sw	0,0	0,0	0:00	kjournald
437	root	g	0	592	592	496	s	0,0	0,2	0:00	syslogd
442	root	9	0	1040	1040	448	s	0,0	0,4	0:00	klogd
462	rpc	9	0	588	588	504	s	0,0	0,2	0:00	portmap
490	rpcuser	9	0	756	756	660	s	0,0	0,2	0:00	rpc.statd
590	root	8	0	524	524	464	s	0,0	0,2	0:00	apmd
647	root	9	0	1268	1268	1076	s	0,0	0,4	0:00	sshd
680	root -	9	0	1008	992	816	s	0,0	0,3	0:00	xinetd
704	lp	9	0	1020	1020	872	s	0,0	0,3	0:00	lpd

Сначала идет общесистемная информация — из нее можно узнать время запуска операционной системы, время работы операционной системы от момента последнего перезапуска системы, количество зарегистрированных в данный момент в операционной системе пользователей, а также минимальную, максимальную и среднюю загрузку операционной системы. Помимо этого, отображается общее количество процессов и их состояние, сколько процентов ресурсов системы используют пользовательские процессы и системные процессы, использование оперативной памяти и свопа.

Далее идет таблица, во многом напоминающая вывод программы ps. Идентификационный номер процесса, имя пользователя — владельца процесса, приоритет процесса, размер процесса, его состояние, используемые процессом оперативная память и ресурс центрального процесса, время выполнения и, наконец, имя процесса.

Утилита top после запуска периодически обновляет информацию о состоянии процессов в операционной системе, что позволяет нам динамически получать информацию о загрузке системы.

kill

Программа kill (в переводе с английского — убить) предназначена для послышки соответствующих сигналов указанному нами процессу. Как правило, это бывает тогда, когда некоторые процессы начинают вести себя неадекватно. Наиболее часто программа применяется, чтобы прекратить выполнение процессов.

Для того чтобы прекратить работу процесса, необходимо знать PID процесса либо его имя. Например, чтобы "убить" процесс 123, достаточно выполнить следующую команду:

```
kill 123
```

Как обычно, чтобы прекратить работу процесса, вам необходимо быть его владельцем. Самой собой, пользователь root может прекратить работу любого процесса в системе.

Иногда стандартное выполнение программы kill не справляется с поставленной задачей. Обычно это объясняется тем, что данный процесс завис либо выполняет операцию, которую с его точки зрения нельзя прервать немедленно. Для прерывания этого процесса можно воспользоваться следующей командой:

```
kill -9 123
```

Что это означает? Вообще-то программа kill предназначена для послышки процессам управляющих сигналов, одним из которых является сигнал sigterm (terminate, завершиться). Этот сигнал посылается процессу при выполнении программы kill по умолчанию. Процесс, получивший данный сигнал, должен корректно завершить свою работу (закрыть используемые файлы, сбросить буферы ввода/вывода и т. п.). Ключ -9 указывает программе kill посылать процессу другой тип сигнала — sigkill. Это приводит к тому, что процесс не производит корректного завершения, а немедленно прекращает свою жизнедеятельность. Помимо этих сигналов, в вашем распоряжении целый набор различных сигналов. Полный список сигналов можно получить, выполнив следующую команду:

- | | | | |
|-----------------|-----------------|-----------------|-----------------|
| 1) SIGHUP | 2) SIGINT | 3) SIGQUIT | 4) SIGILL |
| 5) SIGTRAP | 6) SIG7ABRT | 7) SIGBUS | 8) SIGFPE |
| 9) SIGKILL | 10) SIGUSR1 | 11) SIGSEGV | 12) SIGUSR2 |
| 13) SIGPIPE | 14) SIGALRM | 15) SIGTERM | 17) SIGCHLD |
| 18) SIGCONT | 19) SIGSTOP | 20) SIGTSTP | 21) SIGTTIN |
| 22) SIGTTOU | 23) SIGURG | 24) SIGXCPU | 25) SIGXFSZ |
| 26) SIGVTALRM | 27) SIGPROF | 28) SIGWINCH | 29) SIGIO |
| 30) SIGPWR | 31) SIGSYS | 32) SIGRTMIN | 33) SIGRTMIN+1 |
| 34) SIGRTMIN+2 | 35) SIGRTMIN+3 | 36) SIGRTMIN+4 | 37) SIGRTMIN+5 |
| 38) SIGRTMIN+6 | 39) SIGRTMIN+7 | 40) SIGRTMIN+8 | 41) SIGRTMIN+9 |
| 42) SIGRTMIN+10 | 43) SIGRTMIN+11 | 44) SIGRTMIN+12 | 45) SIGRTMIN+13 |
| 46) SIGRTMIN+14 | 47) SIGRTMIN+15 | 48) SIGRTMAX-15 | 49) SIGRTMAX-14 |
| 50) SIGRTMAX-13 | 51) SIGRTMAX-12 | 52) SIGRTMAX-11 | 53) SIGRTMAX-10 |
| 54) SIGRTMAX-9 | 55) SIGRTMAX-8 | 56) SIGRTMAX-7 | 57) SIGRTMAX-6 |

58) SIGRTMAX-5 59) SIGRTMAX-4 60) SIGRTMAX-3 61) SIGRTMAX-2
62) SIGRTMAX-1 63) SIGRTMAX

killall

Еще один вариант программы `kill`. Используется для того, чтобы завершить работу процессов, носящих одно и то же имя. К примеру, в нашей системе запущено несколько программ `tc`. Для того чтобы одновременно завершить работу этих программ, достаточно всего лишь выполнить следующую команду:

```
killall me
```

Конечно, этим не ограничивается использование данной команды. С ее помощью можно отсылать сигналы группе одноименных процессов. Для получения более подробной информации по этой команде обращайтесь к ее `man`-странице.

Изменение приоритета выполнения процессов

В операционной системе Linux у каждого процесса есть свой приоритет исполнения. Это очень удобно. Поскольку операционная система многозадачная — то для выполнения каждого процесса выделяется определенное количество времени. Для некоторых задач необходимо выделить побольше, для некоторых можно поменьше. Для этого и предназначен приоритет процесса. Управление приоритетом процесса осуществляется программами `nice` и `renice`.

nice

Программа `nice` позволяет запустить команду с предопределенным приоритетом выполнения, который задается в командной строке. При обычном запуске все задачи имеют один и тот же приоритет, и операционная система равномерно распределяет между ними процессорное время. Однако с помощью утилиты `nice` можно понизить приоритет какой-либо задачи, таким образом, предоставляя другим процессам больше процессорного времени. Повысить приоритет той или иной задачи имеет право только пользователь `root`. Синтаксис использования `nice` следующий:

```
nice -number command
```

Уровень приоритета процесса определяется параметром *number*, при этом большее его значение означает меньший приоритет процесса. Значение по умолчанию — 10, и *number* представляет собой число, на которое должен быть уменьшен приоритет.

К примеру, процесс `top` имеет приоритет, равный -5. Для того чтобы понизить приоритет выполнения процесса на десять, мы должны выполнить следующую команду:

```
nice 10 top
```

В результате процесс `top` имеет приоритет, равный 5.

Только пользователь `root` может поднять приоритет того или иного процесса, используя для этого *отрицательное* значение параметра *number*.

renice

Программа `renice`, в отличие от программы `nice`, позволяет изменить приоритет уже работающего процесса. Формат запуска программы следующий:

```
renice -number PID
```

В общем, программа `renice` работает точно так же, как и `nice`. Уровень приоритета процесса определяется параметром *number*, при этом большее его значение означает меньший приоритет процесса. Значение по умолчанию — 10, и *number* представляет собой число, на которое должен быть уменьшен приоритет процесса.

Только пользователь `root` может поднять приоритет того или иного процесса, используя для этого *отрицательное* значение параметра *number*.

Выполнение процессов в заданное время

Одна из основных задач автоматизации администрирования операционной системы — выполнение программ в заданное время или с заданной периодичностью. Конечно, можно запускать программы самостоятельно, но проводить 24 часа на работе или постоянно удаленно

запускать программы в самое неподходящее время (часа в три ночи) — безумие. Для решения этих проблем существует несколько утилит, позволяющих запускать процессы в нужное время.

at

Для запуска одной или более команд в заранее определенное время используется команда `at`. В этой команде вы можете определить время и дату запуска той или иной команды. Команда `at` требует, по меньшей мере, двух параметров — время выполнения программы и запускаемую программу с ее параметрами запуска.

Приведенный ниже пример запустит команду на выполнение в 01:01. Для этого введите все, приведенное ниже, с терминала, завершая ввод каждой строки нажатием клавиши `<Enter>` и по окончании ввода всей команды — `<Ctrl>+<D>` для ее завершения.

```
at 1:01
Is
echo "Time is 1:01"
```

Помимо времени, в команде `at` может быть также определена и дата запуска программы на выполнение.

Пользователь `root` может без ограничения применять практически любые команды. Для обычных пользователей права доступа к команде `at` определяются файлами `/etc/at.allow` и `/etc/at.deny`. В файле `/etc/at.allow` содержится список тех, кому разрешено использовать команду `at`, а в файле `/etc/at.deny` находится список тех, кому ее выполнять запрещено.

batch

Команда `batch` в принципе аналогична команде `at`. Более того, `batch` представляет собой псевдоним команды `at -b`. Для чего необходима эта команда? Представьте, вы хотите запустить резервное копирование вечером. Однако в это время система очень занята, и выполнение резервирования системы практически парализует ее работу. Для этого и существует команда `batch` — ее использование позволяет операционной системе самой решить, когда наступает подходящий момент для запуска задачи в то время, когда система не сильно загружена.

Формат команды `batch` представляет собой просто список команд для выполнения, следующих в строках за командой; заканчивается список комбинацией клавиш `<Ctrl>+<D>`. Можно также поместить список команд в файл и перенаправить его на стандартный ввод команды `batch`.

cron

`Cron` — это программа, выполняющая задания по расписанию, но, в отличие от команды `at`, она позволяет выполнять задания неоднократно. Вы определяете времена и даты, когда должна запускаться та или иная программа. Времена и даты могут определяться в минутах, часах, днях месяца, месяцах года и днях недели.

Программа `cron` запускается один, раз при загрузке системы. При запуске `cron` проверяет очередь заданий `at` и задания пользователей в файлах `crontab`. Если для запуска не было найдено заданий — следующую проверку `cron` произведет через минуту.

Для создания списка задач для программы `cron` используется команда `crontab`. Для каждого пользователя с помощью этой команды создается его собственный `crontab`-файл со списком заданий, имеющий то же имя, что и имя пользователя.

Каждая строка в файле `crontab` содержит шаблон времени и команду. Команда выполняется тогда, когда текущее время соответствует приведенному шаблону. Шаблон состоит из пяти частей, разделенных пробелами или символами табуляции, и имеет вид:

минуты часы день _ месяца месяц день _ недели задание

Первые пять полей представляют собой шаблон времени и обязательно должны присутствовать в файле. Для того чтобы программа `cron` игнорировала поле шаблона времени, поставьте в нем символ звездочки (*).

Например, шаблон ю 01 01 * * говорит о том, что команда должна быть запущена в десять минут второго каждого первого числа любого (*) месяца, каким бы днем недели оно ни было. В табл. 3 приведено описание полей таблицы задания cron.

Таблица 3. Параметры таблицы заданий программы cron

Поле	Описание
минуты	Указывает минуты в течение часа. Значения от 0 до 59
часы	Указывает час запуска задания. Значения от 0 до 23, где 0 — полночь
день	Указывает день месяца, в который должна исполняться команда
месяца	Указывает месяц, в который необходимо запускать задание. Значения лежат в пределах от 1 до 12, где 1 — январь
день недели	Указывает день недели — или как цифровое значение от 0 до 7 (0 и 7 означают воскресенье), или используя первые три буквы, например Моп
задание	Командная строка для запуска задания

Ниже приведены несколько команд, исполняемых программой cron:

– команда запускается в 1 минуту каждого часа:

```
01 * * * * /usr/bin/script
```

– команда запускается каждый день в 8:20:

```
20 8 * * * /usr/bin/script
```

– команда запускается в 6 часов каждое воскресенье:

```
00 6 * * 0 /usr/bin/script
```

– команда запускается в 7:40 каждое первое число:

```
40 7 1 * * /usr/bin/script
```

Для создания и редактирования файла заданий для программы cron используется команда crontab. Прямое редактирование файла заданий не допускается.

Команда crontab имеет следующие параметры командной строки:

- -e — позволяет редактировать компоненты файла (при этом вызывается редактор, определенный в переменной EDITOR);
- -r — удаляет текущий crontab-файл из каталога;
- -l — используется для вывода списка текущих заданий.

Cron также имеет возможность разрешать или запрещать конкретным пользователем свое использование. Для этого существуют файлы /etc/cron.allow и /etc/cron.deny, которые аналогичны описанным ранее /etc/at.allow и /etc/at.deny.

Задания.

1. Запустите программу уес в фоновом режиме с подавлением потока вывода.
2. Запустите программу уес на переднем плане с подавлением потока вывода. Приостановите выполнение программы. Заново запустите программу уес с теми же параметрами, и завершите ее выполнение.
3. Запустите программу уес на переднем плане без подавления потока вывода. Приостановите выполнение программы. Заново запустите программу уес с теми же параметрами, и завершите ее выполнение.
4. Проверьте состояния процессов, воспользовавшись командой jobs.
5. Переведите процесс, который у вас выполняется в фоновом режиме на передний план и остановите его.
6. Переведите любой ваш процесс с подавлением потока вывода в фоновый режим.
7. Проверьте состояния процессов, воспользовавшись командой jobs. Обратите внимание, что процесс

стал выполняющимся (Running) в фоновом режиме.

8. Запустите процесс в фоновом режиме, таким образом, чтобы он продолжил свою работу даже после отключения от терминала.
9. Закройте окно и заново запустите консоль. Убедитесь, что процесс продолжил свою работу.
10. Получите информацию о запущенных в операционной системе процессах с помощью утилиты `top`.
11. Запустите еще три программы `yes` в фоновом режиме с подавлением потока вывода.
12. «Убейте» два процесса: для одного используйте его PID, а для другого его идентификатор конкретного задания.
13. Попробуйте послать сигнал 1 (SIGHUP) процессу, запущенному с помощью `nohup` и обычному процессу.
14. Запустите еще несколько программ `yes` в фоновом режиме с подавлением потока вывода.
15. Завершите их работу одновременно, используя команду `killall`.
16. Запустите программу `yes` в фоновом режиме с подавлением потока вывода. Используя утилиту `nice`, запустите программу `yes` с теми же параметрами и с приоритетом, большим на 5. Сравните абсолютные и относительные приоритеты у этих двух процессов.
17. Используя утилиту `renice`, измените приоритет у одного из потоков `yes` таким образом, чтобы у обоих потоков приоритеты были равны.
18. Сделайте так, чтобы в `xx` минут `xx` часов автоматически выполнялась утилита `ls` и выводился строка текста «Lab rab 3. Zadanie 18». Учтите, что вывод будет осуществляться не на экран, а в файл: `/var/spool/mail/student` (`xx` минут `xx` часов – ближайшие несколько минут).
19. Сделайте так, чтобы в `xx` минут `xx` часов каждую пятницу автоматически выполнялась утилита `ps` и выводился строка текста «Lab rab 3. Zadanie 19». Учтите, что вывод будет осуществляться не на экран, а в файл: `/var/spool/mail/student` (`xx` минут `xx` часов – ближайшие несколько минут).
20. Сделайте так, чтобы в `xx` минут `xx` часов каждый `xx` месяц автоматически выполнялась утилита `ps` и выводился строка текста «Lab rab 3. Zadanie 20». Учтите, что вывод будет осуществляться не на экран, а в файл: `/var/spool/mail/student` (`xx` минут `xx` часов – ближайшие несколько минут, `xx` месяц – текущий месяц).

Критерии оценивания отчета:

Оценка «5» ставится, если учащийся выполняет работу в полном объеме с соблюдением необходимой последовательности проведения опытов и измерений; самостоятельно и рационально монтирует необходимое оборудование; все опыты проводит в условиях и режимах, обеспечивающих получение правильных результатов и выводов; соблюдает требования правил безопасности труда; в отчете правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ погрешностей.

Оценка «4» ставится, если выполнены требования к оценке «5», но было допущено два - три недочета, не более одной негрубой ошибки и одного недочёта.

Оценка «3» ставится, если работа выполнена не полностью, но объем выполненной части таков, позволяет получить правильные результаты и выводы: если в ходе проведения опыта и измерений были допущены ошибки.

Оценка «2» ставится, если работа выполнена не полностью и объем выполненной части работы не позволяет сделать правильных выводов: если опыты, измерения, вычисления, наблюдения производились неправильно.

Лабораторная работа «Структура операционной системы Windows XP» (3 часа)

Цель занятия: изучить структуру операционной системы Windows XP.

ИНСТРУКЦИОННАЯ КАРТА

Материально - техническое оснащение

Оборудование:

Аппаратная часть: персональный компьютер с правами администратора.
Программная часть: программа VirtualBox, виртуальная машина с установленной ОС Windows 8/10, текстовый процессор Microsoft Word.

Теоретическая часть:

Windows XP имеет модульную структуру (рис. 2.20), в которой код операционной системы и драйверы выполняются в привилегированном режиме процессора (режиме ядра), обеспечивающем полный доступ ко всей аппаратной части компьютера, а пользовательские приложения выполняются в непривилегированном режиме процессора – пользовательском режиме без прямого доступа к оборудованию компьютера. В режиме ядра работают следующие компоненты.

1. **Уровень абстрагирования от оборудования (Hardware Abstraction Layer, HAL).** Его задачей является отделение операционной системы от особенностей конкретных реализаций в аппаратном обеспечении компьютера, т. е. от различий в материнских платах, в модификациях процессоров, в наборах микросхем и др. Благодаря этому уровню управление подсистемами прерываний, прямого доступа к памяти, системными шинами и таймерами для ядра операционной системы является одинаковым. Уровень HAL реализован в системном файле *Hal.dll*.

2. **Ядро операционной системы.** Ядро содержит наиболее часто вызываемые низкоуровневые функции операционной системы: планирование и распределение ресурсов между процессами, их переключение и синхронизацию. В обязанности ядра входит также управление прерываниями и обработка ошибочных ситуаций при функционировании операционной системы. Код ядра Windows XP не разделяется на потоки, а находится только в оперативной памяти и не может быть выгружен на диск. Код ядра Windows XP находится в системном файле *Ntoskrnl.exe*.

3. **Драйверы устройств.** Драйверы представляют собой подпрограммы, транслирующие вызовы, поступившие от пользовательских программ в запросы обработки данных для конкретных устройств. Значительное число драйверов входит в состав Windows XP (они располагаются в подкаталоге *Isystem32\drivers* системного каталога и имеют тип файла *.sys, например, драйвер дисковой подсистемы находится в файле *disk.sys*), а для нестандартных периферийных устройств драйверы находятся в комплектах поставки.

4. **Исполняющая подсистема (NT Executive).** Модуль *NT Executive* состоит из *микроядра* и подсистем *диспетчеризации управления программами* с доступом к виртуальной памяти, окнам и графической подсистеме. Виртуальная память предоставляет пользовательским программам виртуальные адреса адресного пространства процессов и соответствующие физические страницы оперативной памяти компьютера. Графическая подсистема предназначена для создания оконного интерфейса, рисования элементов управления, расположенных в окнах. К исполняющей подсистеме относятся системные файлы *Ntkrnlpa.exe*, *Kernel32.dll*, *Advapi32.dll*, *User32.dll*, *Gdi32.dll*.

Операционная система Windows XP в значительной мере использует возможности процессоров, совместимых с семейством *Intel x86*. В их аппаратной архитектуре предусматривается четыре уровня привилегий выполнения кода программ от 0-го наивысшего привилегированного до 4-го пользовательского режима с ограниченным набором команд процессора. Программы режима ядра операционной системы Windows XP функционируют в нулевом, защищенном и привилегированном режиме, а остальные пользовательские программы работают в менее привилегированных режимах, находясь под контролем программ режима ядра.

Недоступные в пользовательском режиме операции и приложения обращаются к системным вызовам ядра операционной системы *Win32 API*. В состав *API* входит более 250 функций, обращение к которым осуществляется при помощи системных вызовов, основанных на подпрограммах ядра операционной системы. Все вызовы *Win32 API* обслуживаются как системными службами *NT*, так и модулем *NT Executive* – исполняющей системы Windows XP. Модуль *NT Executive* представляет собой несколько программных потоков, которые выполняются в режиме ядра. Код практически всех подсистем этого модуля находится в файле *ntoskrnl.exe* (кроме подсистемы *Win32*, код которой расположен в файле *win32k.sys*) и уровне абстрагирования от

оборудования *HAL*, который содержится в файле *hal.dll*. В модуле *NT Executive* сосредоточены все самые важные части операционной системы.

Микроядро отвечает за выделение памяти для приложений и распределение процессорного времени, т. е. за реализацию многозадачности. Для этого в состав микроядра входит *планировщик потоков (threads scheduler)*, который назначает каждому из потоков один из 32 уровней приоритета. Уровень 0 зарезервирован для системы. Уровни от 1-го до 15-го назначаются исполняемым программам, а уровни от 16-го до 31-го могут назначаться только администраторами. Планировщик делит все процессорное время на кванты фиксированного размера. При этом каждый программный поток выполняется только в течение отведенного ему времени, и если по окончании кванта он не освобождает процессор, планировщик в принудительном порядке приостанавливает этот поток и меняет программное окружение процесса, настраивая его на выполнение другого потока, обладающего тем же приоритетом. Микроядро также осуществляет всю работу, связанную с обработкой программных и аппаратных прерываний.

5. Диспетчеризация управления программами. Модуль состоит из следующего набора системных программ:

- *Диспетчер ввода-вывода* – интегрирует добавляемые в систему драйверы устройств в операционную систему *Windows XP*;

- *Диспетчер объектов* – служит для управления всеми разделяемыми ресурсами компьютера. В момент обращения приложения к какому-либо ресурсу диспетчер объектов сопоставляет с этим ресурсом объект (например, окно) и отдает приложению дескриптор^[1] (№ окна) этого объекта. Используя дескриптор, приложение взаимодействует с объектом, совершая в его отношении различные операции. Монитор системы безопасности следит при этом за тем, чтобы с объектом выполнялись только разрешенные действия;

- *Диспетчер процессов* – предоставляет интерфейс, при помощи которого другие компоненты *Windows NT Executive*, а также приложения пользовательского режима могут манипулировать процессами и потоками. Во время работы диспетчер процессов сопоставляет с каждым процессом и потоком идентификатор процесса (*PID – Process Identifier*) и идентификатор потока (*TID – Thret Identifier*) соответственно, а также таблицу адресов и таблицу дескрипторов;

- *Диспетчер виртуальной памяти* – служит для управления организации подсистемы памяти, позволяет создавать таблицы адресов для процессов и следит за корректностью использования адресного пространства приложениями. Кроме того, обеспечивает возможность загрузки в оперативную память исполняемых файлов и файлов динамических библиотек. Диспетчер виртуальной памяти представляет физическую память для пользовательских приложений – каждому процессу выделяются 4 Гб виртуального адресного пространства, из которых младшие 2 Гб используются процессом, а старшие 2 Гб (общие для всех процессов) отводятся на нужды системы. Каждый процесс работает в своем изолированном адресном пространстве и «не знает» о других процессах. Процессы обмениваются данными через разделяемую память, которая может быть спроецирована на виртуальное адресное пространство нескольких процессов. Главная задача диспетчера виртуальной памяти – организация логической памяти, размер которой больше размера физической, установленной на компьютере. Это достигается благодаря тому, что страницы памяти, к которым долго не было обращений, и которые не имеют атрибута неперемещаемых, сохраняются диспетчером в файле *pagefile.sys* на жестком диске и удаляются из оперативной памяти, освобождая ее для других приложений. В момент, когда происходит обращение к данным, находящимся в перемещенной на винчестер странице, диспетчер виртуальной памяти копирует страницу обратно в оперативную память, затем обеспечивает доступ к ней. Этот механизм обеспечивает выделение дополнительной памяти программам, которые нуждаются в ней, и при этом следит за тем, чтобы все работающие в системе программы обладали достаточным объемом физической памяти для того, чтобы продолжать функционирование;

- *Диспетчер кэша* – используется для кэшированного чтения и записи и позволяет существенно ускорить работу жестких дисков и других устройств. При этом наиболее востребованные файлы дублируются диспетчером кэша в оперативной памяти компьютера, и обращение к ним обслуживается с использованием этой копии, а не оригинала, расположенного на сравнительно медленном долговременном носителе. Кэш в *Windows XP* является единым для всех

логических дисков, вне зависимости от используемой файловой системы. Кроме того, он является динамическим, а это значит, что диспетчер управляет его размерами в зависимости от доступного объема свободной физической памяти в каждый конкретный момент;

• *Диспетчеры окон и графики* – выполняют все функции, связанные с пересылкой системных сообщений и отображением информации на экране.

Процесс функционирования *Windows XP* условно подразделяется на три фазы: процесс начальной загрузки, штатный режим работы и завершение работы. Для загрузки *Windows XP* используется следующий минимальный набор файлов:

- файлы, располагающиеся в корневом каталоге загрузочного диска: *Ntldr*, *Boot.ini*, *Bootsect.dos* (файл необходим только при использовании мультизагрузки), *Ntdetect.com*;

- файлы, располагающиеся в системном подкаталоге */system32: Ntoskrnl.exe, Hal.dll*, разделы реестра *SYSTEM*;

- файлы, располагающиеся в системном подкаталоге */system32/drivers*: (необходимые драйверы устройств).

Процесс загрузки компьютера начинается с процедуры начального тестирования оборудования (*POST – Power-On Self Test*). Код, выполняющий *POST*, зашит в базовой системе ввода-вывода (*BIOS*) каждого компьютера, при включении питания ему передается управление. Если в процессе тестирования обнаруживаются какие-либо ошибки, то *BIOS* генерирует коды ошибок (*POSTcodes*), которые отличаются для *BIOS* разных производителей, и звуковые коды. Если процедура *POST* завершается успешно, то *BIOS* передает управление главной загрузочной записи (*MBR – Master Boot Record*) и первая «аппаратная» стадия загрузки компьютера, когда процесс зависит только от аппаратуры компьютера, завершается.

Далее загрузочная запись, оперируя данными о разбиении жесткого диска на логические тома, передает управление исполняемому коду, загрузчику *Ntldr*, расположенному в загрузочном секторе. Загрузчик переходит в защищенный режим и производит необходимые для успешного функционирования манипуляции с памятью, кроме этого, *Ntldr* имеет модули, позволяющие работать с файловой системой и некоторыми другими базовыми ресурсами системы. Все другие действия выполняются с помощью вызова прерываний *BIOS*.

Если в файле *boot.ini* зарегистрировано более одной операционной системы, то после первичной инициализации загрузчик предоставляет пользователю возможность выбора путем вывода *Ntldr* приглашения о выборе операционной системы. Если выбрана операционная система *Windows XP*, загрузчик запускает файл *Ntdetect.com*. Этот компонент считывает из *CMOS*-памяти системную дату и время, после чего производит поиск и распознавание аппаратных средств, подключенных в данный момент к компьютеру. Завершив работу, *Ntdetect* возвращает управление и собранную им информацию обратно в *Ntldr*. Далее загружается и инициализируется ядро операционной системы *Ntoskrnl.exe* и уровень абстрагирования от оборудования *Hal.dll*. При инициализации ядро производит ряд действий в определенной последовательности:

- инициализация диспетчера памяти;
- инициализация диспетчера объектов;
- установка системы безопасности;
- настройка драйвера файловой системы;
- загрузка и инициализация диспетчера ввода-вывода;
- загрузка системных сервисов, которые реализуют взаимодействие с пользователем.

В состав системных сервисов входят следующие модули:

- *Smss.exe* (диспетчер сеансов) – модуль управляет другими сервисами и службами *Windows*; запускает: *Win32 (Csrss)* и некоторые системные утилиты, выполняемые на этапе загрузки; реализует графический пользовательский интерфейс и запуск процессов *Csrss.exe* и *WinLogon.exe*;

- *Csrss.exe* – модуль предназначен для организации взаимодействия между компьютером и пользователем;

- *Lsass.exe* – служба, запускаемая *WinLogon.exe* и отвечающая за безопасность системы (предоставляет возможность пользователю зарегистрироваться в системе).

После загрузки операционной системы пользователь должен пройти процедуру *аутентификации* – ввести собственное регистрационное имя (*логин*) и пароль. Процедура подключения к системе позволяет определить, обладает ли пользователь правом входа и работы с системой. Эту процедуру выполняет служба *WinLogon*. При этом в системе происходят следующие события:

- *процесс WinLogon* отображает на экране фон рабочего стола и приглашение к вводу пользователем логина и пароля. Введенные данные передаются подсистеме безопасности;
- подсистема безопасности обращается к базе данных *SAM (Security Accounts Manager)* и проверяет, обладает ли пользователь полномочиями работы с системой.

Если пользователь является авторизованным пользователем системы, то подсистема безопасности формирует для него *идентификатор доступа*, который вместе с управлением передает обратно процессу *WinLogon*. Процесс *WinLogon* посредством обращения к подсистеме *Win32* создает новый процесс для пользователя и прикрепляет ему идентификатор доступа. Каждый процесс, в дальнейшем создаваемый пользователем, отмечается принадлежащим ему идентификатором доступа, поэтому доступ пользователя к ресурсам системы контролируется и отслеживается. Благодаря обязательной процедуре подключения к системе упрощается реализация механизмов: *аудит системы* и *квоты на использование ресурсов*. Пользовательский идентификатор доступа содержит идентификатор пользователя, а также идентификаторы всех групп, к которым принадлежит данный пользователь.

Если операционная система не загружается корректно, то при нажатии в процессе загрузки *Windows XP* клавиши *F8* происходит переход в расширенное меню запуска, содержащее пункты:

- *Безопасный режим* – загрузка *Windows XP* с минимальным требуемым количеством системных файлов и драйверов устройств;
- *Безопасный режим с загрузкой сетевых драйверов* – загрузка *Windows XP* с минимальным требуемым количеством системных файлов и драйверов устройств с поддержкой подключения к сети;
- *Безопасный режим с поддержкой командной строки* – загрузка *Windows XP* с минимальным требуемым количеством системных файлов и драйверов устройств с загрузкой режима командной строки;
- *Включить протоколирование загрузки* – режим позволяет записать этапы загрузки *Windows XP* в файл *Nbtlog.txt*;
- *Включить режим VGA* – режим, загружает драйвер стандартного монитора *VGA* с разрешением 640 на 480 точек на дюйм и 16 цветами;
- *Загрузка последней удачной конфигурации* – режим, восстанавливает последнюю неиспорченную копию реестра *Windows XP*.

Практическая часть:

1. Изучить теорию по структуре ОС.
2. Сделать краткое описание составных частей структуры ОС+ (рисунки).
3. Составить письменный отчет.
4. Защитить работу.

Контрольные вопросы к защите:

1. На какие фазы условно можно разделить работу ОС *Windows*?
2. Какие цели достигаются на уровне абстрагирования?
3. Что такое ядро системы?
4. Что такое драйвера устройств?
5. Что такое исполняющая система?
6. Что такое диспетчер ввода-вывода?
7. Что такое диспетчер объектов?
8. Что такое диспетчер процессов?
9. Что такое диспетчер кэша?
10. Опишите ряд действий выполняемых ядром при инициализации?

Критерии оценивания отчета:

Оценка «5» ставится, если учащийся выполняет работу в полном объеме с соблюдением необходимой последовательности проведения опытов и измерений; самостоятельно и рационально монтирует необходимое оборудование; все опыты проводит в условиях и режимах, обеспечивающих получение правильных результатов и выводов; соблюдает требования правил безопасности труда; в отчете правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ погрешностей.

Оценка «4» ставится, если выполнены требования к оценке «5», но было допущено два - три недочета, не более одной негрубой ошибки и одного недочёта.

Оценка «3» ставится, если работа выполнена не полностью, но объем выполненной части таков, позволяет получить правильные результаты и выводы: если в ходе проведения опыта и измерений были допущены ошибки.

Оценка «2» ставится, если работа выполнена не полностью и объем выполненной части работы не позволяет сделать правильных выводов: если опыты, измерения, вычисления, наблюдения производились неправильно.

Лабораторная работа «Управление памятью и вводом/выводом в ОС» (3 часа)

Цель занятия: Практическое знакомство с управлением вводом/выводом в операционных системах Windows и кэширования операций ввода/вывода.

ИНСТРУКЦИОННАЯ КАРТА

План проведения занятия:

1. Ознакомиться с краткими теоретическими сведениями.
2. Ознакомиться с назначением и основными функциями Диспетчера задач Windows.
3. Приобрести навыки применения командной строки Windows. Научиться запускать, останавливать и проверять работу процессов.
4. Сделать выводы о взаимосвязи запущенных процессов и оперативной памяти компьютера.
5. Подготовить отчет для преподавателя о выполнении лабораторной работы и записать его в папку «Выполнение».

Оборудование:

Аппаратная часть: персональный компьютер, сетевой или локальный принтер.

Программная часть: ОС Windows 7, текстовый процессор Microsoft Word.

Краткие теоретические сведения:

Необходимость обеспечить программам возможность осуществлять обмен данными с внешними устройствами и при этом не включать в каждую двоичную программу соответствующий двоичный код, осуществляющий собственно управление устройствами ввода/вывода, привела разработчиков к созданию системного программного обеспечения и, в частности, самих операционных систем.

Программирование задач управления вводом/выводом является наиболее сложным и трудоемким, требующим очень высокой квалификации. Поэтому код, позволяющий осуществлять операции ввода/вывода, стали оформлять в виде системных библиотечных процедур; потом его стали включать не в системы программирования, а в операционную систему с тем, чтобы в каждую отдельно взятую программу его не вставлять, а только позволить обращаться к такому коду. Системы программирования стали генерировать обращения к этому системному коду ввода/вывода и осуществлять только подготовку к собственно операциям ввода/вывода, то есть автоматизировать преобразование данных к соответствующему формату, понятному устройствам, избавляя прикладных программистов от этой сложной и трудоемкой работы. Другими словами, системы программирования вставляют в машинный код необходимые библиотечные подпрограммы

ввода/вывода и обращения к тем системным программным модулям, которые, собственно, и управляют операциями обмена между оперативной памятью и внешними устройствами.

Таким образом, управление вводом/выводом — это одна из основных функций любой ОС. Одним из средств правления вводом/выводом, а также инструментом управления памятью является диспетчер задач Windows, он отображает приложения, процессы и службы, которые в текущий момент запущены на компьютере. С его помощью можно контролировать производительность компьютера или завершать работу приложений, которые не отвечают.

При наличии подключения к сети можно также просматривать состояние сети и параметры ее работы. Если к компьютеру подключились несколько пользователей, можно увидеть их имена, какие задачи они выполняют, а также отправить им сообщение.

Также управлять процессами можно и «вручную» при помощи командной строки.

Команды Windows для работы с процессами:

- at — запуск программ в заданное время
- Schtasks — настраивает выполнение команд по расписанию
- Start — запускает определенную программу или команду в отдельном окне.
- Taskkill — завершает процесс
- Tasklist — выводит информацию о работающих процессах

Для получения более подробной информации, можно использовать центр справки и поддержки или команду help (например: help at)

- command.com — запуск командной оболочки MS-DOS
- cmd.exe — запуск командной оболочки Windows

Ход работы:

Задание 1. Работа с Диспетчером задач Windows 7.

1. Запустите Windows 7
2. Запуск диспетчера задач можно осуществить двумя способами:
 - 1) Нажатием сочетания клавиш Ctrl+Alt+Del. При использовании данной команды не стоит пренебрегать последовательностью клавиш. Появится меню, в котором курсором следует выбрать пункт «Диспетчер задач».
 - 2) Переведите курсор на область с показаниями системной даты и времени и нажмите правый клик, будет выведено меню, в котором следует выбрать «Диспетчер задач».
3. Будет выведено окно как на рис. 1.

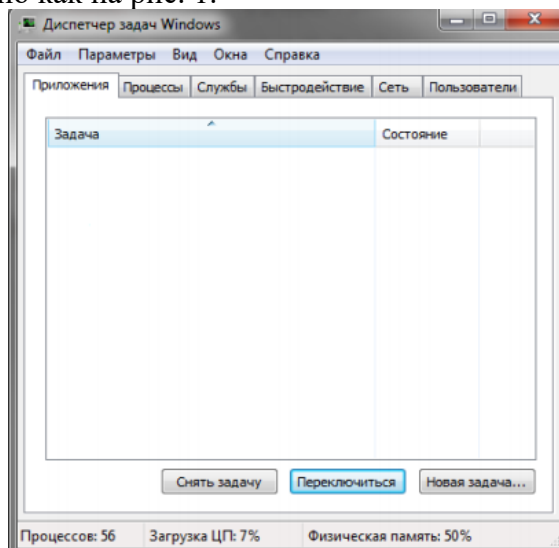


Рис. 1. Диспетчер задач Windows 7.

4. В диспетчере задач есть 6 вкладок:
 1. Приложения
 2. Процессы
 3. Службы

4. Быстродействие

5. Сеть

6. Пользователи

- Вкладка «Приложения» отображает список запущенных задач (программ) выполняющиеся в настоящий момент не в фоновом режиме, а также отображает их состояние. Также в данном окне можно снять задачу переключиться между задачами и запустить новую задачу при помощи соответствующих кнопок.

- Вкладка «Процессы» отображает список запущенных процессов, имя пользователя запустившего процесс, загрузку центрального процессора в процентном соотношении, а также объем памяти используемого для выполнения процесса. Также присутствует возможность отображать процессы всех пользователей, либо принудительного завершения процесса. Процесс — выполнение пассивных инструкций компьютерной программы на процессоре ЭВМ.

- Вкладка «Службы» показывает, какие службы запущены на компьютере. Службы

- — приложения, автоматически запускаемые системой при запуске ОС Windows и выполняющиеся вне зависимости от статуса пользователя.

- Вкладка «Быстродействие» отображает в графическом режиме загрузку процессора, а также хронологию использования физической памяти компьютера. Очень эффективным инструментом наблюдения является «Монитор ресурсов». С его помощью можно наглядно наблюдать за каждой из сторон «жизни» компьютера. Подробное изучение инструмента произвести самостоятельно, интуитивно.

- Вкладка «Сеть» отображает подключенные сетевые адаптеры, а также сетевую активность.

- Вкладка «Пользователи» отображает список подключенных пользователей.

- Потренируйтесь в завершении и повторном запуске процессов.

- Разберите мониторинг загрузки и использование памяти.

- Попробуйте запустить новые процессы при помощи диспетчера, для этого можно использовать команды: cmd, msconfig.

5. После изучения диспетчера задач:

Задание 2. Командная строка Windows.

1. Для запуска командной строки в режиме Windows следует нажать:

(Пуск) > «Все программы» > «Стандартные» > «Командная строка»

2. Поработайте выполнением основных команд работы с процессами: запуская, отслеживая и завершая процессы.

Основные команды

Schtasks — выводит выполнение команд по расписанию

Start — запускает определенную программу или команду в отдельном окне. Taskkill — завершает процесс

Tasklist — выводит информацию о работающих процессах

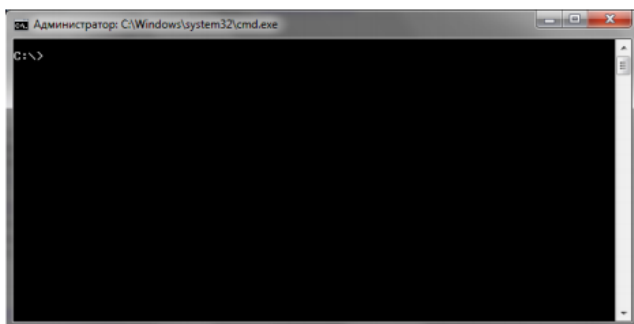


Рис. 2. Командная строка Windows 7.

3. В появившемся окне (рис. 2) наберите:

cd\ — переход в корневой каталог;

cd windows – переход в каталог Windows.

dir — просмотр содержимого каталога.

В данном каталоге мы можем работать с такими программами как «WordPad» и «Блокнот».

4. Запустим программу «Блокнот»:

C:\Windows > start notepad.exe

Отследим выполнение процесса: C:\Windows > tasklist

Затем завершите выполнение процесса: C:\Windows > taskkill /IM notepad.exe

5. Самостоятельно, интуитивно, найдите команду запуска программы WordPad.

Необходимый файл запуска найдите в папке Windows.

6. Выполнение задания включить в отчет по выполнению лабораторной работы.

Задание 3. Самостоятельное задание.

1. Отследите выполнение процесса explorer.exe при помощи диспетчера задач и командной строки.

2. Прдемонстрируйте преподавателю завершение и повторный запуск процесса explorer.exe из:

- Диспетчера задач;
- Командной строки.

3. Выполнение задания включить в отчет по выполнению лабораторной работы.

Контрольные вопросы:

1. Дайте понятие процессу в операционной системе.

2. Дайте понятие службе в операционной системе.

3. Причислите основные команда работы с процессами при помощи командной строки.

Критерии оценивания отчета:

Оценка «5» ставится, если учащийся выполняет работу в полном объеме с соблюдением необходимой последовательности проведения опытов и измерений; самостоятельно и рационально монтирует необходимое оборудование; все опыты проводит в условиях и режимах, обеспечивающих получение правильных результатов и выводов; соблюдает требования правил безопасности труда; в отчете правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ погрешностей.

Оценка «4» ставится, если выполнены требования к оценке «5», но было допущено два - три недочета, не более одной негрубой ошибки и одного недочёта.

Оценка «3» ставится, если работа выполнена не полностью, но объем выполненной части таков, позволяет получить правильные результаты и выводы: если в ходе проведения опыта и измерений были допущены ошибки.

Оценка «2» ставится, если работа выполнена не полностью и объем выполненной части работы не позволяет сделать правильных выводов: если опыты, измерения, вычисления, наблюдения производились неправильно.

Лабораторная работа «Ознакомление с сетевыми функциями операционной системы » (3 часа)

Цель занятия: Практическое знакомство с управлением вводом/выводом в операционных системах Windows и кэширования операций ввода/вывода.

ИНСТРУКЦИОННАЯ КАРТА

Указания к проведению лабораторной работы

Связь локального компьютера с Интернетом, локальной сетью или другим компьютером обеспечивает компонент «Сетевые подключения». Это средство позволяет получать доступ к ресурсам и функциональным возможностям компьютеров сети. Операции создания, настройки,

сохранения подключений и наблюдения за ними выполняются в папке «Сетевые подключения». Чтобы настроить параметры TCP/IP и открыть компонент Сетевые подключения.

1. Нажмите кнопку Пуск, выберите пункт Панель управления, а затем дважды щелкните значок Сетевые подключения.

2. Выделите подключение, которое требуется настроить, и затем в группе Типичные сетевые задачи щелкните ссылку Изменить параметры этого подключения.

3. Выполните одно из следующих действий.

- В случае подключения по локальной сети выберите в списке Отмеченные компоненты используются этим подключением вариант Протокол Интернета (TCP/IP) и нажмите кнопку Свойства.

- В случае подключения удаленного доступа, входящего подключения или подключения VPN откройте вкладку Сеть. В списке Отмеченные компоненты используются этим подключением выберите вариант Протокол Интернета (TCP/IP) и нажмите кнопку Свойства.

4. Выполните одно из следующих действий.

- Чтобы параметры IP были назначены автоматически, установите переключатель в положение Получить IP-адрес автоматически и нажмите кнопку ОК.

- Чтобы вручную указать IP-адрес или адрес сервера DNS, выполните следующие действия.

- Установите переключатель в положение Использовать следующий IP-адрес и в поле IP-адрес введите соответствующий адрес и маску сети.

- Установите переключатель в положение Использовать следующие адреса DNS-серверов и введите в полях Предпочитаемый DNS-сервер и Альтернативный DNS-сервер адреса основного и (необязательно) дополнительного серверов DNS.

5. Чтобы настроить параметры DNS, WINS и IP, нажмите кнопку Дополнительно.

- По возможности желательно использовать автоматическую настройку параметров IP (с помощью DHCP), что продиктовано следующими соображениями.

- Служба DHCP включена по умолчанию.

- При изменении местоположения компьютера не требуется вручную изменять параметры IP.

- Автоматическое назначение параметров IP используется для всех подключений; при этом отпадает необходимость в настройке параметров DNS, WINS и т. п.

Служебные программы протоколов TCP/IP

Служебные программы TCP/IP обеспечивают подключение к сетям и проверку работы сетевого подключения. Служебные программы и команды выполняются в режиме командной строки, для перехода в который необходимо выполнить переход к разделу стандартных программ и выбрать пункт Командная строка или, что удобнее, запустить программу cmd.exe через Пуск/Выполнить. В данной консоли вводятся команды и формируются выходные данные. Список служебных команд протоколов TCP/IP довольно обширен [1 - 4]. В рамках данной ознакомительной работы рассматриваются команды **ipconfig**, **ping**, **tracert**.

Команда ipconfig

Команда ipconfig служит для отображения всех текущих параметров сетевых подключений компьютера к сети TCP/IP и обновления параметров служб DHCP и DNS.

При вызове команды ipconfig без параметров выводится только IP-адрес, маска подсети и основной шлюз для каждого сетевого адаптера (Рисунок 7.1).

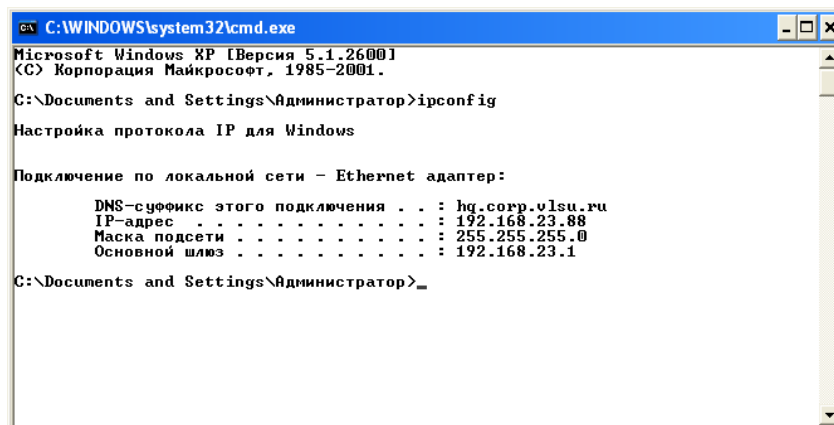


Рисунок 7.1 Краткая информация о сетевых подключениях

Некоторые примеры команды **ipconfig**:

ipconfig /?

Отображение справки по данной команде в командной строке.

ipconfig/all

Вывод полной конфигурации TCP/IP для всех адаптеров (рисунок 7.2). Адаптеры могут представлять собой физические интерфейсы, такие как установленные сетевые адаптеры, или логические интерфейсы, такие как подключения удаленного доступа.

ipconfig/renew [*адаптер*]

Обновление конфигурации DHCP для всех адаптеров (если адаптер не задан) или для заданного *адаптера*. Данный параметр доступен только на компьютерах с адаптерами, настроенными для автоматического получения IP-адресов. Чтобы указать адаптер, введите без параметров имя, выводимое командой **ipconfig**. (рисунок 7.3)

ipconfig/release [*адаптер*]

Отправка сообщения DHCPRELEASE серверу DHCP для освобождения текущей конфигурации DHCP и удаление конфигурации IP-адресов для всех адаптеров (если адаптер не задан) или для заданного *адаптера*. Этот адаптер отключает протокол TCP/IP для адаптеров, настроенных для автоматического получения IP-адресов. Чтобы указать адаптер, введите без параметров имя, выводимое командой **ipconfig**.

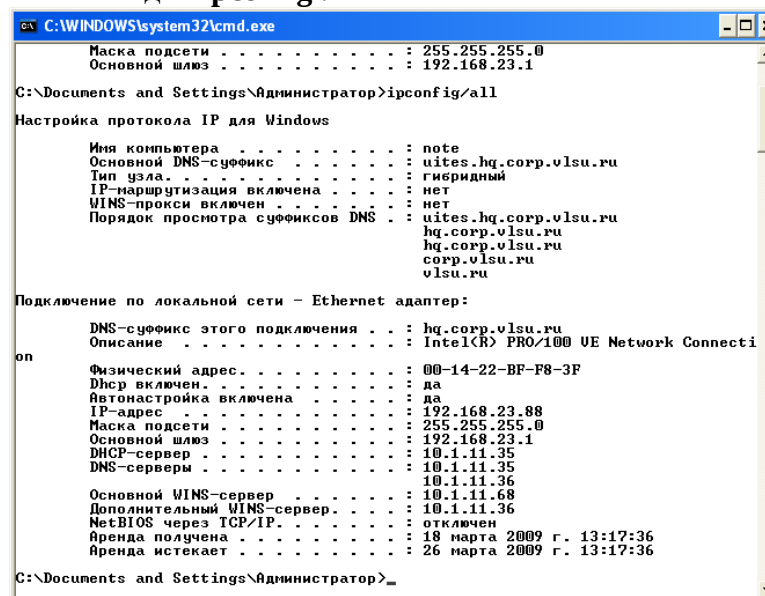


Рисунок 7.2 Полная информация о сетевых подключениях

ipconfig/registerdns

Динамическая регистрация вручную имен DNS и IP-адресов, настроенных на компьютере. Этот параметр полезен при устранении неполадок в случае отказа в регистрации имени DNS или

при выяснении причин неполадок динамического обновления между клиентом и DNS-сервером без перезагрузки клиента.

Для просмотра и обновления IP-адреса можно воспользоваться окном «Сетевые подключения». Для этого откройте окно Сетевые подключения, щелкните правой кнопкой мыши сетевое подключение, выберите команду **Состояние**, а затем откройте вкладку **Поддержка**.

Данная команда доступна только на компьютерах с адаптерами, настроенными для автоматического получения IP-адресов. Это позволяет пользователям определять, какие значения конфигурации были получены с помощью DHCP, APIPA или другой конфигурации.

Если имя *адаптер* содержит пробелы, его следует заключать в кавычки (т. е. "имя адаптера").

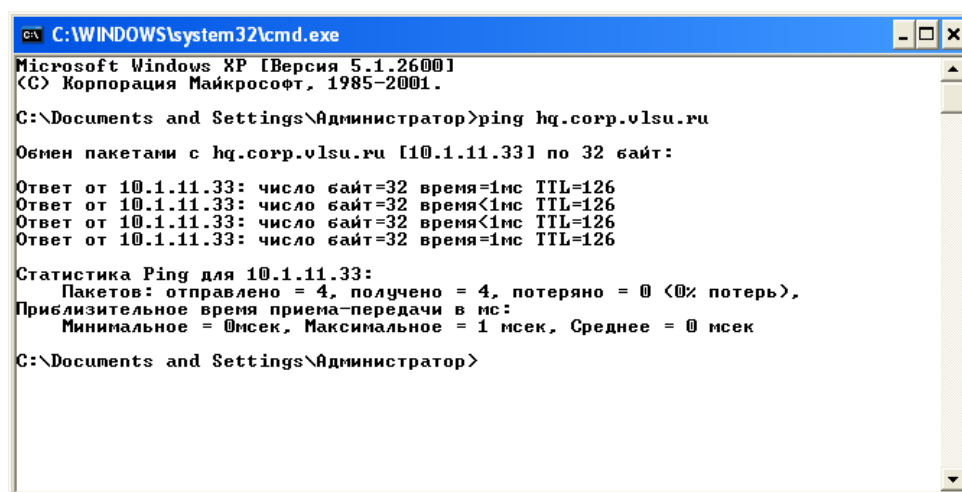
В именах адаптеров, задаваемых для команды ipconfig, поддерживается использование подстановочного знака звездочки (*) для задания имен, начинающихся с указанной строки или содержащих указанную строку. Например, имя **Подкл*** будет включать все адаптеры, начинающиеся со строки «Подкл», а имя ***сет*** — все адаптера, содержащие строку «сет».

Эта команда доступна, только если в свойствах сетевого адаптера в объекте Сетевые подключения в качестве компонента установлен протокол Интернета (TCP/IP).

Команда Ping

Команда Ping с помощью отправки сообщений с эхо-запросом по протоколу ICMP проверяет соединение на уровне протокола IP с другим компьютером, поддерживающим TCP/IP. После каждой передачи выводится соответствующее сообщение с эхо-ответом (рисунок 7.3).

Ping - это основная TCP/IP-команда, используемая для устранения неполадки в соединении, проверки возможности доступа и разрешения имен. Команда ping, запущенная без параметров, выводит справку.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\Documents and Settings\Администратор>ping hq.corp.vlsu.ru

Обмен пакетами с hq.corp.vlsu.ru [10.1.11.33] по 32 байт:

Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время<1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время<1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126

Статистика Ping для 10.1.11.33:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
    Приблизительное время приема-передачи в мс:
        Минимальное = 0 мсек, Максимальное = 1 мсек, Среднее = 0 мсек

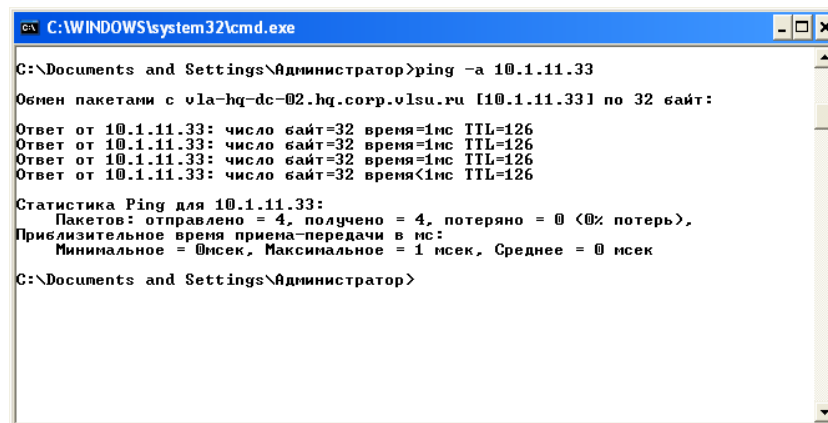
C:\Documents and Settings\Администратор>
```

Рисунок 7.3 Выполнение команды Ping

Примеры использования команды с параметрами

Ping -t Задаёт для команды ping отправку сообщений с эхо-запросом к точке назначения до тех пор, пока команда не будет прервана. Для прерывания команды и вывода статистики нажмите комбинацию CTRL-BREAK.

Ping -a Задаёт разрешение обратного имени по IP-адресу назначения. В случае успешного выполнения выводится имя соответствующего узла (рисунок 7/4).



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\Администратор>ping -a 10.1.11.33
Обмен пакетами с v1a-hq-dc-02.hq.corp.vlsu.ru [10.1.11.33] по 32 байт:

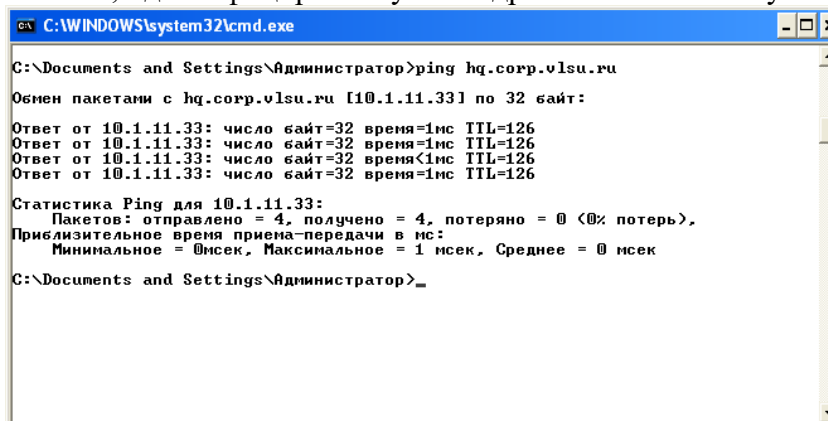
Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время<1мс TTL=126

Статистика Ping для 10.1.11.33:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 1 мсек, Среднее = 0 мсек
C:\Documents and Settings\Администратор>
```

Рисунок 7.4 Выполнение команды Ping для разрешения имени по IP-адресу

Ping имя_конечного_компьютера

Задаёт точку назначения, идентифицированную IP-адресом или именем узла (рисунок 7.5).



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\Администратор>ping hq.corp.vlsu.ru
Обмен пакетами с hq.corp.vlsu.ru [10.1.11.33] по 32 байт:

Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время<1мс TTL=126
Ответ от 10.1.11.33: число байт=32 время=1мс TTL=126

Статистика Ping для 10.1.11.33:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 1 мсек, Среднее = 0 мсек
C:\Documents and Settings\Администратор>
```

Рисунок 7.5 Выполнение команды Ping с определением IP-адреса по имени ping/?

Отображает справку в командной строке.

Команда ping позволяет проверить имя и IP-адрес компьютера. Если проверка IP-адреса успешная, и проверка имени — нет, то имеет место проблема разрешения имен. В этом случае с помощью запросов DNS (Domain Name System) или с помощью методов разрешения имен NetBIOS проверьте, чтобы имя задаваемого компьютера было разрешено в локальном файле Hosts.

Эта команда доступна, только если в свойствах сетевого адаптера в объекте Сетевые подключения в качестве компонента установлен протокол Интернета (TCP/IP).

Команда Tracert

Команда Tracert определяет путь до точки назначения с помощью посылки в точку назначения эхо-сообщений протокола Control Message Protocol (ICMP) с постоянным увеличением значений срока жизни (Time to Live, TTL). Выведенный путь — это список ближайших интерфейсов маршрутизаторов, находящихся на пути между узлом источника и точкой назначения. Ближний интерфейс представляют собой интерфейс маршрутизатора, который является ближайшим к узлу отправителя на пути. Запущенная без параметров, команда tracert выводит справку.

Примеры использования команды с параметрами

Tracert **-d** предотвращает попытки команды tracert разрешения IP-адресов промежуточных маршрутизаторов в имена. Увеличивает скорость вывода результатов команды tracert.

tracert **-h** *максимальное_число_переходов* задает максимальное количество переходов на пути при поиске конечного объекта. Значение по умолчанию равно 30.

tracert **-j** *список_узлов* указывает для сообщений с эхо-запросом использование параметра свободной маршрутизации в заголовке IP с набором промежуточных мест назначения, указанных в *списке_узлов*. При свободной маршрутизации успешные промежуточные места назначения могут

быть разделены одним или несколькими маршрутизаторами. Максимальное число адресов или имен в списке — 9. *Список_адресов* представляет набор IP-адресов (в точечно-десятичной нотации), разделенных пробелами.

`tracert -w интервал` определяет в миллисекундах время ожидания для получения эхо-ответов протокола ICMP или ICMP-сообщений об истечении времени, соответствующих данному сообщению эхо-запроса. Если сообщение не получено в течение заданного времени, выводится звездочка (*). Таймаут по умолчанию 4000 (4 секунды).

`tracert имя_конечного_компьютера` задает точку назначения, указанную IP-адресом или именем узла.

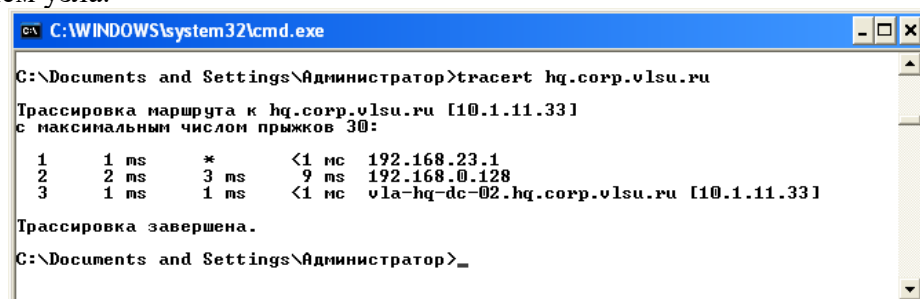


Рисунок 7.6 Трассировка маршрута к указанному компьютеру

`tracert -?` отображает справку в командной строке.

Задание на выполнение лабораторной работы

Задание выполняется в условиях, когда оба виртуальных компьютера запущены и соединены сетью.

1. Используя команду `ipconfig`, выполните следующие команды и объясните полученные результаты.

- вывести основную конфигурацию TCP/IP для всех адаптеров (введите `ipconfig`);
- вывести полную конфигурацию TCP/IP для всех адаптеров (введите `ipconfig/all`);
- обновить конфигурацию IP-адреса, назначенного DHCP-сервером, только для адаптера с условным именем **Подключение по локальной сети** (введите `ipconfig /renew "Подключение по локальной сети"`);
- сбросить кэш сопоставления имен DNS при наличии неполадок в сопоставлении имен (введите: `ipconfig /flushdns`);
- вывести код класса DHCP для всех адаптеров с именами, начинающимися со слова *Подключение* (введите `ipconfig/showclassid Подключение*`);

2. Используя команду `ping`, выполните следующие команды и объясните полученные результаты.

- Определите IP-адрес компьютера назначения по его имени. (Введите `ping имя_компьютера`)
- Определите имя компьютера назначения по его IP-адресу. (Введите `ping -a ip-address`)
- отправьте точке назначения десять сообщений с эхо-запросом, каждое из которых имеет поле данных из 1000 байт (введите `ping -n 10 -l 1000 ip-address`);

Проверьте конфигурацию TCP/IP с помощью команды `ping`.

- Чтобы быстро получить значения параметров конфигурации TCP/IP на компьютере, откройте командную строку и выполните команду **ipconfig**. С помощью сведений, отображенных

командой **ipconfig**, убедитесь, что сетевой адаптер для проверяемой конфигурации TCP/IP не находится в состоянии Сеть отключена.

- В командной строке обратитесь по адресу замыкания на себя; для этого выполните команду **ping 127.0.0.1**.

- Обратитесь командой «ping» по IP-адресу данного компьютера.

- Обратитесь командой «ping» по IP-адресу основного шлюза.

- Если команда ping не была успешно выполнена, проверьте правильность IP-адреса основного шлюза и работоспособность этого шлюза (маршрутизатора).

- Обратитесь командой «ping» по IP-адресу удаленного узла (узла, находящегося в другой подсети).

- Если команда ping не была успешно выполнена, проверьте правильность IP-адреса удаленного узла, работоспособность этого узла, а также работоспособность всех шлюзов (маршрутизаторов) между локальным компьютером и удаленным узлом.

- Обратитесь командой «ping» по IP-адресу DNS-сервера.

Если команда ping не была успешно выполнена, проверьте правильность IP-адреса DNS-сервера, работоспособность DNS-сервера, а также работоспособность всех шлюзов (маршрутизаторов) между локальным компьютером и DNS-сервером.

3. Используя команду **tracert**, выполните следующие команды и объясните полученные результаты.

- выполнить трассировку маршрута с помощью команды **tracert**.

Откройте командную строку и выполните следующую команду:

tracert имя_узла либо **tracert ip-адрес**, указав в параметре *имя_узла* или *ip-адрес*, соответственно, имя узла или IP-адрес удаленного компьютера.

- выполнить трассировку маршрута от локального компьютера к узлу **www.microsoft.com**. (название узла условно – если ваш компьютер в Интернете, используйте любой адрес, на занятиях в классе - получите от преподавателя имена узлов, доступных в сети).

Введите следующую команду: **tracert www.microsoft.com**

- выполнить трассировку маршрута от локального компьютера к узлу **www.microsoft.com** чтобы команда «**tancert**» не разрешала и не выводила на экран имена всех маршрутизаторов на пути.

Введите следующую команду: **tracert -d www.microsoft.com**

4. Окна терминала командной строки с результатами выполнения команд поместите в отчет по лабораторной работе.

Контрольные вопросы

1. Как настроить TCP/IP для получения статического IP-адреса?
2. Как настроить TCP/IP для автоматического получения IP-адреса?
3. К чему приведет последовательное выполнение команд **ipconfig/release ipconfig/renew**?
4. Как проверить исправность подключения хоста к сети с использованием команды **ping**?
5. Какая служба обеспечивает разрешение адреса по имени при выполнении команды **ping имя хоста**?
6. Самостоятельно определите, как работают и какую информацию дают при выполнении команды **Hostname, Route, NetStat**.

Лабораторная работа «Установка виртуальной компьютерной сети на основе операционных систем Windows» (3 часа)

Цель занятия: создать модель компьютерной сети предприятия на основе виртуальной машины Microsoft Virtual PC для изучения операционных систем в рамках дисциплины «Операционные системы, среды и оболочки».

Назначение виртуального компьютера в составе лабораторного комплекса

Применение виртуального компьютера позволяет создать гибкую в настройках и безопасную для реального компьютера среду, в которой студент обладает правами администратора, что позволяет изучать все аспекты применения операционных систем без вмешательства в настройки реального (физического) компьютера. Это создает уникальные возможности для изучения любых ОС в составе сети предприятия без необходимости их установки на реальном компьютере.

Под управлением основной системы могут быть одновременно запущены любые операционные системы и процесс изучения ОС, приобретения и тестирования навыков проходит на порядок быстрее. Изолированность виртуальной машины от основной операционной системы исключает возможность распространения вирусов или срабатывания вредоносных механизмов исследуемого программного обеспечения.

Виртуальный компьютер представлен файлами на диске реального компьютера и может быть легко перенесен с одного компьютера на другой.

Создав одну виртуальную машину с нужным набором программного обеспечения, в течение нескольких минут можно установить ее на все машины компьютерного класса. Ничего страшного не произойдет, если обучаемый в процессе освоения преподаваемых технологий умышленно или нечаянно разрушит подопытную среду. Для восстановления поврежденной виртуальной машины из резервной копии понадобится всего несколько минут. При выполнении лабораторных работ все тестовые сети и компьютеры, находящиеся внутри них, создадим, используя средства комплекса виртуальных машин.

Указания к выполнению лабораторной работы

Создание виртуальных компьютеров для лабораторного комплекса необходимо выполнить в следующей последовательности:

1. Реализовать действия, необходимые для установки Microsoft Virtual PC 2004/2007 на компьютере учебного класса с операционной системой Windows XP или Windows Vista. Установка выполняется путем запуска установочного файла setup.exe из дистрибутива Microsoft Virtual PC 2004/2007, являющегося бесплатным и свободно распространяемым продуктом.

При запуске виртуальной машины появляется консоль управления виртуальными компьютерами (рисунок 1.1), предоставляющая возможность установки любых операционных систем и работы с ними после установки, как по отдельности, так и в составе компьютерной сети.

2. Для установки операционной системы на виртуальной платформе необходимо выбрать пункт New и далее Create a virtual machine. При переносе уже имеющейся машины на другой компьютер выбираем пункт Add an existing virtual machine, позволяющий добавить в данный контейнер ранее созданный виртуальный компьютер.

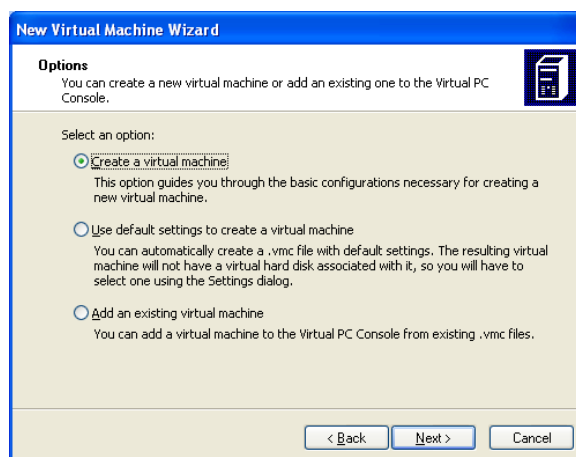


Рисунок 1.1 Выбор варианта установки виртуального компьютера

При установке выбираем объем оперативной памяти, достаточной для функционирования устанавливаемой ОС, и вариант создания жесткого диска машины, как это показано на рисунках 1.2 и 1.3.

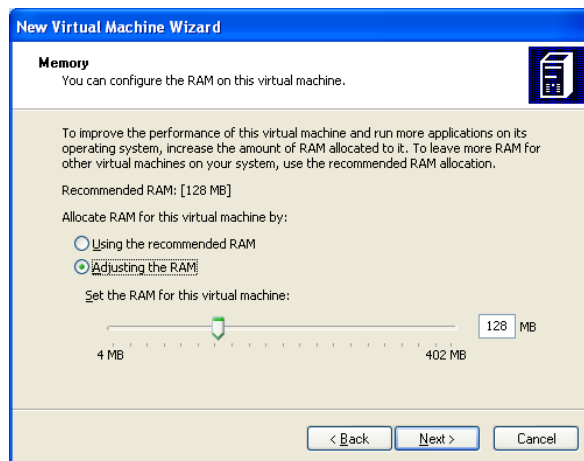


Рисунок 1.2 Выбор объема оперативной памяти виртуальной машины

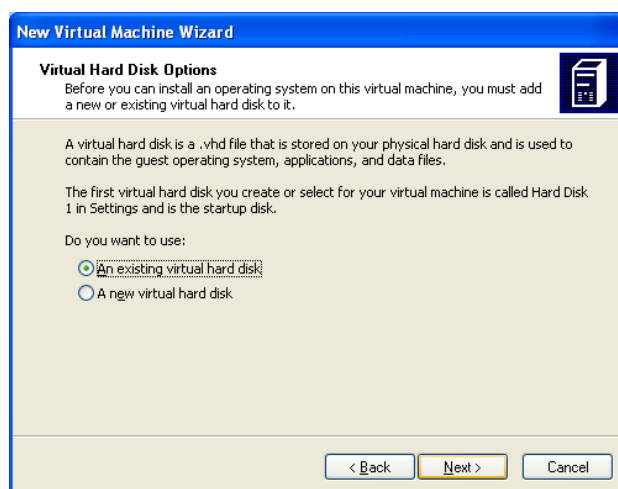


Рисунок 1.3 Выбор варианта создания жесткого диска виртуальной машины

Машину можно установить на новый виртуальный жесткий диск (A new virtual hard disk) или использовать диск, созданный ранее (An existing virtual hard disk).

После создания новой виртуальной машины ее имя появляется в консоли и ее можно запустить для установки операционной системы (рисунок 1.4).

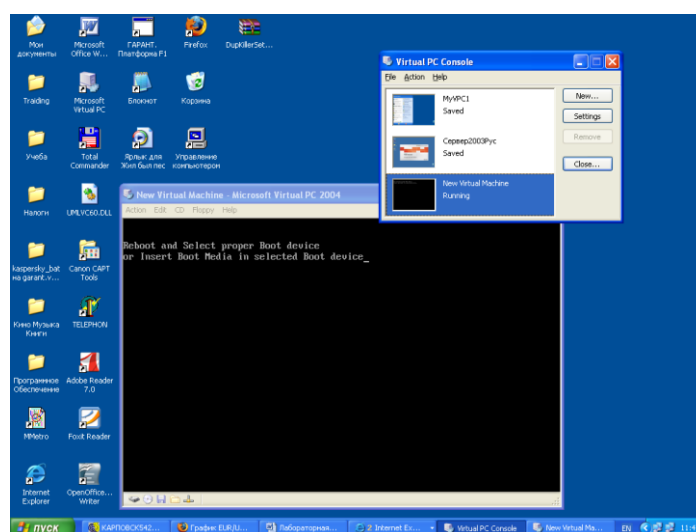


Рисунок 1.4 Запуск виртуальной машины для установки операционной системы

Для установки операционной системы необходимо иметь ее дистрибутив на CD/DVD диске или файл образа в формате ISO.

В первом случае в меню виртуального компьютера CD выберите Use physical drive и виртуальная машина будет использовать привод реального компьютера. При наличии ISO – образа выбираем пункт Capture ISO image и указываем соответствующий файл.

Далее установка операционной системы ничем не отличается от ее установки на реальном компьютере.

При установке ОС необходимо выбрать размер виртуального жесткого диска, достаточный для установки операционной системы и предполагаемого к установке ПО, а при форматировании диска обязательно выбрать опцию «Форматировать раздел в системе NTFS». Далее при установке следует выбирать стандартные настройки, предлагаемые по умолчанию. По окончании установки системы новый виртуальный компьютер появляется в консоли Управления виртуальной машиной и может быть запущен кнопкой Start.

Аналогично устанавливается операционная система Windows Server 2003. При установке выбираем все варианты по умолчанию, так как настройки сети и серверов предполагается выполнить позже в ходе выполнения лабораторных работ.

Среда, моделирующая компьютерную сеть предприятия, образуется при одновременном запуске виртуальных машин с серверной и клиентской операционными системами. Для организации сетевого взаимодействия виртуальных компьютеров необходимо настроить виртуальные сетевые соединения. Для этого используется раздел Settings установки параметров Virtual PC, в котором предусмотрено несколько режимов настройки сети. Для того, чтобы создать изолированную от реальной системы виртуальную сеть выберем режим Local only, в котором виртуальные машины взаимодействуют только между собой.

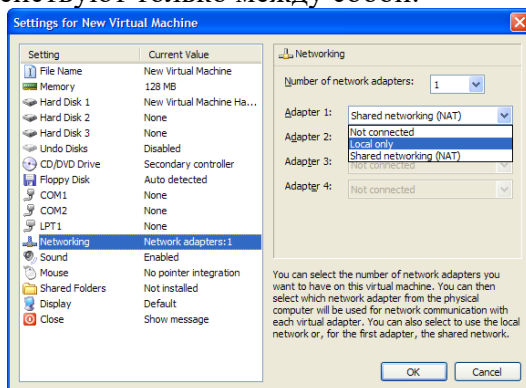


Рисунок 1.5 Выбор варианта сетевого взаимодействия виртуальных машин

Далее необходимо настроить протокол TCP/IP на сервере и клиентской машине. Настройка протокола TCP/IP на сервере в данном случае сводится к установлению постоянного (статического) IP-адреса виртуального компьютера с операционной системой Windows Server.

Для этого откроем окно свойств подключения по локальной сети и выберем компонент Протокол Интернета (TCP/IP) (рисунок 1.6).

Настроим свойства протокола, установив IP-адрес 192.168.1.1 и маску сети 255.255.255.0. Аналогично задается IP-адрес для клиентской машины. Установим его равным 192.168.1.10. В качестве шлюза по умолчанию укажите адрес сервера.

Проверим взаимодействие виртуальных компьютеров по сети. Для этого в режиме командной строки (выполняя Пуск/Выполнить/cmd.exe) введем команду проверки функционирования сети ping 192.168.1.1 на машине – клиенте и ping 192.168.1.10 на сервере.

Если сетевые адаптеры и протоколы взаимодействуют верно, то результаты выполнения команд будут показывать наличие обмена данными.

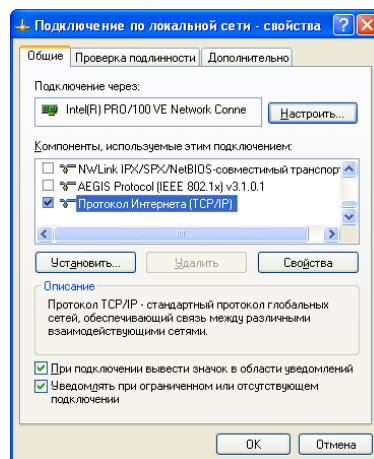


Рисунок 1.6 Выбор протокола TCP/IP для настройки IP-адресов

Таким образом, у нас получается рабочий макет стандартной схемы локальной сети масштаба предприятия. Для простоты понимания учебного примера проведена миниатюризация, заключающаяся в том, что в каждую сеть помещен только один компьютер. Этого достаточно для демонстрации обсуждаемых концепций и изучения ОС в составе лабораторного комплекса.

Задание на выполнение лабораторной работы

Одна из бригад студентов, выбранная преподавателем, выполняет следующее задание:

1. Установить виртуальную машину.
2. Установить на виртуальном компьютере операционную систему Windows XP или Vista.
3. Установить на следующем виртуальном компьютере операционную систему Windows Server 2003 или Windows Server 2008.
4. Настроить сетевое соединение компьютеров, как это предписано в методических указаниях.
5. Проверить взаимодействие компьютеров через сеть.
6. Создать копии файлов виртуальных машин на мобильном запоминающем устройстве – DVD – диске или Flash – носителе.
7. Остальные бригады студентов получают копии файлов виртуальных машин и создают на их основе виртуальные машины на своих рабочих местах.
8. Каждая бригада отчитывается перед преподавателем, демонстрируя работу виртуальных компьютеров, установленных ими на рабочем месте студента.

Контрольные вопросы

1. Назовите преимущества использования виртуальной машины при изучении операционных систем.
2. Назовите основные шаги установки виртуального компьютера.
3. Как установить виртуальную машину с параметрами по умолчанию?
4. Как установить виртуальную машину с использованием файлов имеющейся виртуальной машины?
5. Назовите способы установки операционных систем на виртуальную машину.
6. Каким образом выполняется выбор режимов работы сетевых адаптеров виртуальной машины?
7. Каким образом можно установить созданную виртуальную машину на другом компьютере?
8. Почему в данном случае в качестве файловой системы виртуальных машин необходимо выбирать систему NTFS?

Информационные источники

Печатные издания

1. Батаев А.В. Операционные системы и среды. Учебник Изд. Академия. 2020г.

Электронные издания (электронные ресурсы):

1. Российское образование. Федеральный портал. [Электронный ресурс]. Режим доступа: <http://www.edu.ru>
2. Единая коллекция цифровых образовательных ресурсов. [Электронный ресурс]. Режим доступа: <http://school-collection.edu.ru>
3. Социальная сеть работников образования. [Электронный ресурс]. Режим доступа: <http://nsportal.ru>
4. Электронная информационная образовательная среда. [Электронный ресурс]. Режим доступа: <http://edu.dvgups.ru>
5. Открытый урок. Первое сентября. [Электронный ресурс]. Режим доступа: <http://festival.1september.ru>
6. Педагогическое сообщество «урок.рф». [Электронный ресурс]. Режим доступа: <http://урок.рф>
7. Инфоурок. Ведущий образовательный портал России. [Электронный ресурс]. Режим доступа: <https://infourok.ru>
8. Профобразование. [Электронный ресурс]. Режим доступа: <http://проф-обр.рф>
9. Учебно-методический кабинет. [Электронный ресурс]. Режим доступа: <http://ped-kopilka.ru>
10. Единое окно доступа к образовательным ресурсам. [Электронный ресурс]. Режим доступа: <http://window.edu.ru>
11. Электронное обучение, компьютерная филология. Информационные технологии. [Электронный ресурс]. Режим доступа: <http://it.lang-study.com/>

Дополнительные источники:

1. ЭБС «Юрайт»: Советов, Б. Я. Информационные технологии: учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский. — 7-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2020. — 327 с. — (Профессиональное образование). — ISBN 978-5-534-06399-8. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: <https://urait.ru/bcode/450686>
2. Гохберг Г.С. , Зафиевский А.В. , Короткин А.А. Информационные технологии: Издание: учебник для среднего профессионального образования/ Гохберг Г.С. , Зафиевский А.В. , Короткин А.А – 3-е изд. стер. М.: Изд.центр «Академия», 2020 – 240 с.
3. Михеева Е.В. Информационные технологии в профессиональной деятельности. Технические специальности: учебник для студ. Учреждений сред.проф. образования/Е.В. Михеева, О.И. Титова. – 2-е изд., стер. – М.: Издательский центр «Академия», 2015. – 416с. Режим доступа: <http://www.academia-moscow.ru/reader/?id=168074&demo=Y>.
4. Михеева Е. В. Практикум по информационным технологиям в профессиональной деятельности.- М.: Изд.центр «Академия», 2014. – 256 с. Режим доступа: <http://www.academia-moscow.ru/reader/?id=106719>.