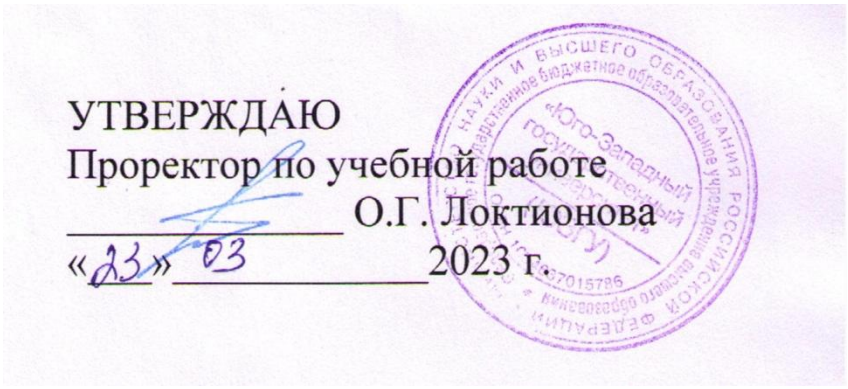
МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждения высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности



АППАРАТНЫЕ И ПРОГРАММНЫЕ СРЕДСТВА ПОСТРОЕНИЯ РАСПРЕДЕЛЕННЫХ СИСТЕМ

Методические указания по выполнению лабораторных работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.65

Составитель: А.В. Митрофанов

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Аппаратные и программные средства построения распределенных систем: методические указания по выполнению лабораторной работы по дисциплине «безопасность распределенных баз данных» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов. – Курск, 2023. – 15с. Библиогр.: с. 15.

Содержат сведения по вопросам построения распределенных систем. Указывается порядок выполнения лабораторной работы, правила, содержание отчета.

Методические указания по выполнению лабораторных работ по дисциплине «Безопасность систем баз данных», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ ____. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

Цель работы: Познакомиться с общими принципами организации распределенных ИС. Получить представление о компонентной объектной модели (COM). Научиться использовать механизм автоматизации для обработки данных с использованием внешних приложений.

Указания к выполнению лабораторной работы

Классическая компьютерная информационная система включает в себя следующие обязательные элементы:

- Информационная база, которая является основным хранилищем данных ИС. В подавляющем большинстве случаев она представляет собой *серверную базу данных*, которая управляется некоторой СУБД. Иные варианты (например, совместно используемый набор файлов) встречаются значительно реже.
- Клиентские приложения, которые используются для работы с данными и устанавливаются на ЭВМ пользователей информационной системы. Клиентские приложения не только выполняют просмотр и редактирование данных (так называемые *презентационные функции* информационной системы), но и обеспечивают решение более сложных задач: контроль корректности вводимой информации, статистическая обработка данных, формирование отчетов и т.п. (*функции ИС по прикладной обработке данных*).
- Средства обеспечения доступности данных информационной базы из клиентских приложений. К таким средствам, прежде всего, относятся средства сетевого доступа, которые обычно реализованы соответствующими средствами сетевых операционных систем (например, поддержка протоколов TCP/IP). Кроме того, сюда следует отнести средства взаимодействия пользовательского приложения и сервера баз данных. Некоторые из них были изучены в ходе выполнения предыдущих лабораторных работ (процессор баз данных BDE, технологии ODBC и ADO и т.п.).

При эксплуатации описанных информационных систем нередко возникают следующие проблемы:

- Необходимость использования ЭВМ с высокими требованиями к аппаратной части при организации рабочих мест пользователей. Клиентские приложения, которые реализуют не только презентационные, но и прикладные функции ИС, обычно требуют для своей работы довольно значительных вычислительных ресурсов. Кроме того, дополнительные ресурсы необходимы для установки и использования средств обеспечения доступа к данным.
- Необходимость приобретения (лицензирования) и установки специальных средств доступа к данным (например, компонентов BDE

или поставщиков данных OLE DB) на каждом компьютере пользователя ИС.
- Сложность конфигурирования средств доступа к данным (например, организация BDE-псевдонимов или настройка источников данных ODBC на всех пользовательских ЭВМ).
- Возможные конфликты между средствами доступа к данным, установленным для работы с различными информационными системами.

Указанные проблемы могут существенно повысить стоимость эксплуатации ИС и значительно уменьшить (или вовсе устранить) экономический эффект от ее внедрения. Для преодоления этих проблем в последние годы все чаще рекомендуется проектировать сложные информационные системы на базе многоуровневой (многозвенной) архитектуры.

Рассмотренная классическая структура информационной системы соответствует двухуровневой модели клиент-сервер. Клиентами в данном случае являются приложения, установленные на ЭВМ пользователей, а сервером – очевидно, сервер базы данных (СУБД либо сетевая ОС – если база данных состоит из множества совместно используемых файлов). Клиенты напрямую обращаются к серверу (посредством служебных средств обеспечения доступа к данным) и единственной информационной услугой, которую они запрашивают, являются хранящиеся в информационной базе данные. Большая часть прикладной обработки данных выполняется клиентским приложением самостоятельно.

Использование трехуровневой архитектуры ИС позволяет разделить функции представления и прикладной обработки данных, а также упростить и унифицировать доступ к данным из клиентских приложений. Это достигается путем выделения части функций информационной системы в отдельный промежуточный слой. В этом случае структура ИС будет состоять не из трех, а из четырех элементов:

- Данные по-прежнему хранятся в информационной базе, в роли которой чаще всего выступает серверная БД.
- Функции прикладной обработки данных сосредоточены в так называемом *сервере приложений* (*Application Server*), который предоставляет клиентским пользовательским приложениям множество информационных услуг (сервисов) высокого уровня. Серверы приложений выступают в качестве промежуточного уровня между серверами данных и клиентскими приложениями. С одной стороны, они взаимодействуют с информационной базой ИС, запрашивая необходимые данные и отсылая запросы на модификацию данных. В этом случае они выступают в качестве клиентов. С другой стороны, они взаимодействуют с клиентскими пользовательскими

приложениями, выдавая по их запросам результаты обработки данных или транслируя полученные запросы на модификацию данных в информационную базу. В этом случае они выступают в качестве серверов. Серверы приложений являются объектами высокого уровня, поэтому их реализация может не зависеть от специфики конкретной операционной системы или СУБД. Это обеспечивает целый ряд важных достоинств многоуровневой архитектуры ИС, в том числе:
    - Возможность подключения к серверу приложений клиентских приложений, работающих под управлением различных ОС.
    - Возможность простого и единообразного подключения клиентских приложений. Для их взаимодействия с сервером приложений может быть использована простая, но достаточно эффективная технология, не требующая установки сложного программного обеспечения на пользовательские ЭВМ. Все средства доступа к данным информационной базы (менеджеры драйверов, поставщик данных для конкретной СУБД, библиотеки, реализующие прикладной программный интерфейс и т. п.) в этом случае сосредоточены на уровне сервера приложений, а не на уровне клиентов.
- Клиентские приложения реализуют только *презентационные функции* ИС (просмотр и редактирования данных). Это соответствует концепции так называемого «тонкого» клиента, предъявляющего минимальные требования к аппаратной части пользовательских ЭВМ.
- Средства обеспечения доступности данных теперь распределены между клиентами и серверами приложений. Средства сетевого взаимодействия по-прежнему необходимы на всех уровнях, однако средства взаимодействия с базами данных необходимы только на уровне серверов приложений (см. выше).

Развитием трехуровневой архитектуры является так называемая *многоуровневая* (*n-уровневая*) *организация вычислений* (*Multi-tier computing*), когда информационная система состоит из большого количества удаленных друг от друга объектов (серверов), каждый из которых может предоставлять другим объектам (клиентам) разнообразные информационные услуги. При этом различные серверы могут работать под управлением различных ОС или использовать для хранения данных СУБД различных типов. Для синхронизированного и согласованного функционирования таких объектов в состав информационной системы также включают специальные служебные сервисы – системы мониторинга и управления распределенными транзакциями, агенты запуска и управления серверами и др.

В настоящее время наиболее популярны следующие технологии организации распределенных информационных систем:

- Технология COM и ее развитие (DCOM, COM+);

- Технология MIDAS (на базе компонентной объектной модели COM);
- Технология CORBA;
- Платформа .NET.

Сокращение COM означает Component Object Model, что можно перевести с английского языка как компонентная объектная модель. Технология была разработана фирмой Microsoft в начале 90-ых годов прошлого века для обеспечения взаимодействия между различными приложениями, запущенными на одном или даже на разных компьютерах[1].

Сущность технологии COM заключается в программировании с использованием *компонентов* – подход, который знаком всем программистам, использующим в своей работе среду разработки Delphi или C++ Builder. Под компонентом в данном случае понимается законченный (и откомпилированный) объект со своими свойствами и методами, который может легко встраиваться в различные приложения и распространяться как отдельный продукт. Компоненты, созданные в соответствии со спецификацией COM, могут функционировать в различной языковой и операционной средах. Это значит, что если разработчик оформил некоторый набор функций как объект COM, то функциями этого объекта могут воспользоваться программисты самых разных языков программирования: C++, Delphi, Visual Basic и т. д. – достаточно, чтобы соответствующая среда разработки поддерживала технологию COM. По этой причине модель COM может являться базовой для создания распределенных информационных систем – составляющие ее объекты могут быть реализованы с использованием различных технологий и инструментов программирования, однако их взаимодействие может осуществляться в соответствии со спецификацией COM (посредством определенных интерфейсов и протоколов).

Принципы обращения к свойствам и методам COM-объекта из других приложений полностью соответствуют модели клиент-сервер. Компонент, в котором реализованы некоторые полезные свойства и методы, выступает в качестве COM-сервера, а обращающиеся к нему приложения – в качестве клиентов. Одно и то же приложение, очевидно, может выступать и в качестве клиента, и в качестве COM-сервера, в зависимости от характера взаимодействия с другими приложениями в каждый конкретный момент времени. Чтобы взаимодействие «клиент-сервер» между произвольной парой приложений состоялось успешно, необходимо выполнение следующих условий:

- Приложение-клиент должно «иметь представление» о тех сервисах (полезных свойствах и методах), которые предоставляет приложение-сервер. Если бы речь шла о приложениях, написанных на одном языке программирования, то для решения этой проблемы достаточно было

бы распространять вместе с компонентом описания соответствующих ему классов. Например, для компонента, написанного на C++, описания его классов хранятся в заголовочных файлах с расширением “*.h*”. Поскольку технология COM является не зависимой от языка программирования, то для определения сервисных функций COM-объекта используется специальный язык – *язык описания интерфейса* (*IDL, Interface Description Language*[2]).

- Все COM-объекты должны иметь уникальные имена, по которым их можно отличать друг от друга и обращаться к ним. При этом уникальность должна носить глобальный характер, поскольку одни и те же COM-объекты могут использоваться десятками тысяч разработчиков в сотнях тысяч различных приложений. Для именования объектов COM используются так называемые глобальные уникальные идентификаторы (*Globally unique identifier*, *GUID*[3]). Глобальный уникальный идентификатор представляет собой 128-разрядное число, которое генерируется с использованием алгоритмов получения случайных чисел и специальной хеш-функции. Вероятность повторения глобального уникального идентификатора такой разрядности чрезвычайно мала. Таким образом, для того чтобы все COM-серверы были поименованы уникальными именами, разработчикам достаточно использовать один и тот же заранее определенный алгоритм генерации глобальных уникальных идентификаторов. В ОС Windows для генерирования GUID используются функции API **UuidCreate** и **CoCreateGuid**. Глобальный уникальный идентификатор принято записывать в следующем формате:

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX,

например:

B502D1BE-9A57-11d0-8FDE-00C04FD9189D

- Операционная система должна хранить список доступных COM-объектов и информацию о месте их хранения (то есть путь к соответствующим исполняемым модулям, в которых реализованы сервисы этих объектов). В ОС Windows для этих целей используется специальный подраздел системного реестра: HKEY_CLASSES_ROOT\CLSID\, у которого в качестве вложенных подразделов выступают имена (глобальные уникальные идентификаторы) зарегистрированных в системе объектов, например:

{00000000-0E4D-0463-87B5-D411BEBE0010}

{00000001-4FEF-40D3-B3FA-E0531B897F98}

{00000010-0000-0010-8000-00AA006D2EA4}

{00000100-0000-0010-8000-00AA006D2EA4}

Каждый из указанных разделов содержит описание свойств соответствующего COM-объекта. Например, свойство **ProgID** содержит программный идентификатор объекта – кодовое наименование, которое присваивается разработчиком и которое, в отличие от глобального уникального идентификатора, несет некоторую смысловую нагрузку. Чтобы задать местонахождение исполняемого модуля для объекта COM используются свойства **InprocServer32** (если реализация объекта располагается в DLL-файле), **LocalServer32** (если объект реализован как EXE-файл) или **RemoteServer32** (для объектов, расположенных на удаленных ЭВМ в компьютерной сети).

- Операционная система, под управлением которой работает COM-сервер, должна иметь возможность единообразного получения адреса зарегистрированного COM-объекта, чтобы передавать соответствующую ссылку запросившим ее клиентам. Кроме того, для взаимодействия клиента с удаленными COM-серверами необходим специальный протокол сетевого взаимодействия, реализующий, в том числе, правила аутентификации и авторизации.

На основании указанного выше можно сделать вывод, что создание полноценных COM-объектов – это весьма сложная задача, требующая от разработчика глубоких знаний в области системного и объектно-ориентированного программирования[4].

Среда разработки Borland C++ Builder предоставляют программисту возможность не только создавать собственные, но и эффективно использовать уже существующие COM-объекты в своих приложениях. В некоторых случаях взаимодействие создаваемого приложения с объектами COM реализуется незаметно («прозрачно») для разработчика, в других – требует от него осознанных действий.

Полезным примером практического и осознанного использования технологии COM является использование *механизма автоматизации* для взаимодействия с внешними приложениями. Механизм автоматизации (*Automation*[5]) позволяет разработчикам приложения привлекать для обработки данных функциональные возможности других приложений. Например, для формирования отчетов часто используются возможности Microsoft Excel или Microsoft Word.

Взаимодействие в данном случае осуществляется между *объектами* и *контроллерами* автоматизации. Объект автоматизации – это объект,

созданный по технологии COM, в котором реализован доступ к полезным свойствам и методам некоторого приложения или его части. Контроллер автоматизации – это приложение, которое имеет доступ к свойствам и методам COM-объекта и использует их для обработки своих данных. В указанном ранее примере, приложение разработчика – это контроллер, а Microsoft Excel либо Microsoft Word – объекты автоматизации.

Рассмотрим общую схему работы контроллера автоматизации при взаимодействии с каким-либо приложением Microsoft Office:

1. Проверить, запущена ли копия приложения-сервера (объекта автоматизации).
2. В зависимости от результатов проверки (либо исходя из конкретной ситуации) запустить копию приложения-сервера либо подключиться к уже имеющейся копии.
3. Если необходимо, сделать окно приложения-сервера видимым.
4. Выполнить какие-то действия с приложением-сервером (например, создать или открыть документы, изменить их данные, сохранить документы и т. п.).
5. Закрыть приложение-сервер, если его копия была запущена данным контроллером, или отключиться от него, если контроллер подключился к уже имеющейся копии (или если работа с приложением будет продолжена).

Ниже приведен программный код для указанной схемы (в качестве объекта автоматизации выбрано приложение Microsoft Excel, которое запускается по нажатию на кнопку Button1 и закрывается по нажатию кнопки Button2):

```
#include <comobj.hpp>
...
TForm1 *Form1;
// глобальные переменные модуля
Variant Server;
bool ServerIsRunning;
...
void __fastcall TForm1::Button1Click(TObject *Sender)
{
AnsiString ServerProgID;
IUnknown *UnknownInterface;
HResult Res;
// Свойство ProgID для приложения Microsoft Excel = "Excel.Application"
ServerProgID = "Excel.Application";
// Проверяем наличие запущенной копии приложения
Res = GetActiveObject(ProgIDToClassID(ServerProgID),0,&UnknownInterface);
if SUCCEEDED(Res)
{
// подключаемся к существующему экземпляру приложения-сервера
Server = GetActiveOleObject(ServerProgID);
```

```
ServerIsRunning = true;
}
else
{
// запускаем новый экземпляр приложения-сервера
Server = CreateOleObject(ServerProgID);
ServerIsRunning = false;
}
// Показываем окно приложения-сервера на экране
Server.OlePropertySet("Visible", true);
try
{
//----------------------------------------
// Здесь выполняются некоторые действия
// с объектами приложения Office
// ----------------------------------------
}
catch(...)
{
// Сообщение об ошибке работы с MS Excel
}
Button1->Enabled = false;
Button2->Enabled = true;
}
//---------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
// отключаемся либо закрываем приложение-сервер
if (ServerIsRunning)
Server= Unassigned;
else
Server.OleProcedure("Quit");
Button2->Enabled = false;
Button1->Enabled = true;
}
```

Для обращения к свойствам и методам объектов автоматизации можно использовать следующие методы класса Variant:

- Variant OlePropertyGet(const String& name, TAutoArgsBase* args = 0) – получить значение свойства *name* некоторого COM-объекта. В качестве второго параметра метода обычно передаются дополнительные аргументы;
- **void** OlePropertySet(const String& name, TAutoArgsBase& args) – установить значение свойства *name* некоторого COM-объекта. В качестве второго параметра метода обычно передается устанавливаемое значение;
- **void** OleProcedure (const String& name, TAutoArgsBase* args = 0) – запустить метод (процедуру) с именем *name* для некоторого COM-

объекта. В качестве второго параметра метода обычно передаются дополнительные параметры процедуры;

- Variant OleFunction (const String& name, TAutoArgsBase* args = 0) – запустить метод (функцию) с именем *name* для некоторого COM-объекта. В качестве второго параметра метода обычно передаются дополнительные параметры функции;

Используя перечисленные методы, можно обращаться к тем объектам приложения Microsoft Excel (Workbooks, WorkSheets, Cells и т. п.), которые изучались в лабораторной работе № 2 курса «Информационные системы».

Примеры:

1. Открываем новую рабочую книгу

```
Server.OlePropertyGet("WorkBooks").OleProcedure("add");
```

2. Устанавливаем ширину колонки “B” равной 40 на первом рабочем листе:

```
Variant Sheet1 = Server.OlePropertyGet("WorkSheets",1);

Variant ColumnB =

Sheet1.OlePropertyGet("Cells",1,2).OlePropertyGet("EntireColumn");

ColumnB.OlePropertySet("ColumnWidth",40);
```

3. Устанавливаем значение ячейки с RC-адресом (Row, Column) равным значению переменной (или Variant-объекта) Data:

```
Sheet1.OlePropertyGet("Cells",Row,Column).OlePropertySet("Value",Data);
```

4. Устанавливаем жирный шрифт для ячейки с RC-адресом (Row, Column)

```
Sheet1.OlePropertyGet("Cells",Row,Column).

OlePropertyGet("Font").OlePropertySet("Bold",true);
```

5. Закрываем последнюю открытую рабочую книгу. Если в нее были внесены изменения, то пользователь будет предупрежден об их возможной потере.

```
Variant WorkBook = Server.OlePropertyGet("WorkBooks",
```

Server.OlePropertyGet("WorkBooks").OlePropertyGet("Count"));

WorkBook.OleProcedure("close");

6. Закрываем последнюю открытую рабочую книгу без сохранения изменений и без предупреждения пользователя.

Variant WorkBook = Server.OlePropertyGet("WorkBooks",

Server.OlePropertyGet("WorkBooks").OlePropertyGet("Count"));

WorkBook.OleProcedure("close",**false**);

Задания к лабораторной работе

1. Исследовать перечень COM-объектов, зарегистрированных на компьютере, на котором выполняется лабораторная работа. Для этого запустить редактор системного реестра (это можно, сделать, например, через команду **Выполнить** кнопки **Пуск**; имя приложения редактора реестра – *regedit.exe*). В редакторе реестра найти раздел HKEY_CLASSES_ROOT и подраздел CLSID. В отчет о выполнении лабораторной работы включить рисунок с частью системного реестра, отображающий перечень нескольких первых объектов COM.
2. Найти в списке зарегистрированных COM-объектов объекты со следующими глобальными уникальными идентификаторами:

{00000011-0000-0010-8000-00AA006D2EA4}

{00000100-0000-0010-8000-00AA006D2EA4}

{00024500-0000-0000-C000-000000000046}

В отчете о выполнении лабораторной работы указать, какие из указанных объектов были обнаружены. Для найденных объектов привести значения свойств **ProgID** и **InprocServer32/LocalServer32/RemoteServer32.**

3. С использованием механизма автоматизации доработать БД-приложение из лабораторной работы № 5 таким образом, чтобы у пользователя была возможность экспортировать в Microsoft Excel данные из основной таблицы приложения. Обратить внимание на удобство восприятия экспортированных данных (предусмотреть выделение заголовков столбцов, подбор ширины ячеек и т.п.).
4. В отчете отразить ход выполнения работы, результаты тестирования и сделанные выводы.

Контрольные вопросы

1. Двухзвенная архитектура ИС.
2. Достоинства и недостатки двухзвенной архитектуры.
3. Особенности трехзвенной архитектуры ИС.
4. Распределенные (многозвенные) ИС.
5. Принципы технологии COM.
6. Понятие глобального уникального идентификатора.
7. Механизм автоматизации как средство межпроцессного взаимодействия.
8. Общая схема работы контроллера автоматизации.
9. Приведите примеры «прозрачного» использования механизма автоматизации при разработке приложений в Borland C++ Builder.
10. Функции и методы для взаимодействия с объектами автоматизации в Borland C++ Builder.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. – СПб: Питер, 2003. – 877 с.: ил.

2. Тель Ж. Введение в распределенные алгоритмы. Пер. с англ. – М.: МЦНМО, 2009. – 616 с.

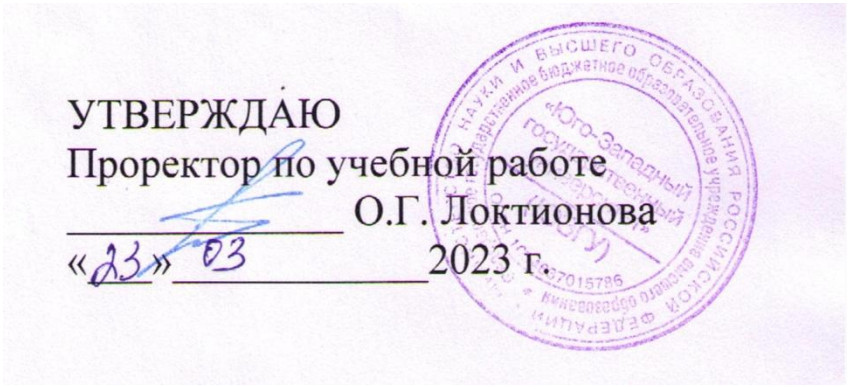
3. Топорков В. В. Модели распределенных вычислений. – М.: Физматлит, 2004. – 320 с.

4. Эндрюс Г. Р. Основы многопоточного, параллельного и распределенного программирования. Пер. с англ. – М.: Издательский дом "Вильямс", 2003. – 512 с.: ил.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждения высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности



ФАЙЛОВАЯ СИСТЕМА NFS

Методические указания по выполнению лабораторных работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.65

Составитель: А.В. Митрофанов

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Файловая система NFS: методические указания по выполнению лабораторной работы по дисциплине «безопасность распределенных баз данных» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов. – Курск, 2023. – 15 с.: – Библиогр.: с. 15.

Содержат сведения по вопросам работы в файловой системе NFS. Указывается порядок выполнения лабораторной работы, правила содержание отчета.

Методические указания по выполнению лабораторных работ по дисциплине «Безопасность систем баз данных», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ ____. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

Цель работы

Ознакомление с файловой системой Linux, её структурой, именами и содержанием каталогов. Приобретение практических навыков по применению команд для работы с файлами и каталогами, по управлению процессами (и работами), по проверке использования диска и обслуживанию файловой системы.

Указания к выполнению лабораторной работы

Команды для работы с файлами и каталогами

Команда для создания текстовых файлов.

Для создания текстового файла удобно воспользоваться командой touch.

Формат команды:

touch имя-файла

Команды просмотра текстовых файлов. Для просмотра небольших файлов

удобно пользоваться командой cat.

Формат команды:

cat имя-файла

Для просмотра больших файлов используйте команду less — она позволяет осуществлять постраничный просмотр файлов (длина страницы соответствует размеру экрана).

Формат команды:

less имя-файла

Для управления процессом просмотра можно использовать следующие управляющие клавиши:

- Space — переход на следующую страниц
- ENTER — сдвиг вперёд на одну строку,
- b — возврат на предыдущую страницу,
- h — обращение за подсказкой,
- q — выход в режим командной строки.

Для просмотра начала файла можно воспользоваться командой head. По умолчанию она выводит первые 10 строк файла.

Формат команды:

head [-n] имя-файла,

где n — количество выводимых строк.

Команда tail выводит несколько (по умолчанию 10) последних строк файла.

Формат команды:

tail [-n] имя-файла,

где n — количество выводимых строк.

Копирование файлов и каталогов

Копирование файлов и каталогов осуществляется при помощи команды cp.

Формат команды:

cp [-опции] исходный_файл целевой_файл

Примеры:

1 Копирование файла в текущем каталоге. Скопировать файл ~/abc1 в файл april

и в файл may:

cd

touch abc1

cp abc1 april

cp abc1 may

2 Копирование нескольких файлов в каталог. Скопировать файлы april и may в

каталог monthly:

mkdir monthly

cp april may monthly

3 Копирование файлов в произвольном каталоге. Скопировать файл monthly/may

в файл с именем june:

cp monthly/may monthly/june

ls monthly

Опция i в команде cp выведет на экран запрос подтверждения о перезаписи файла, если на место целевого файла вы поставите имя уже существующего файла.

Команда cp с опцией r (recursive) позволяет копировать каталоги вместе с входящими в них файлами и каталогами.

Примеры:

1 Копирование каталогов в текущем каталоге. Скопировать каталог monthly в каталог monthly.00:

mkdir monthly.00

cp -r monthly monthly.00

2 Копирование каталогов в произвольном каталоге. Скопировать каталог

monthly.00 в каталог /tmp

cp -r monthly.00 /tmp

Перемещение и переименование файлов и каталогов

Команды mv и mvdir предназначены для перемещения и переименования файлов и каталогов.

Формат команды mv:

mv [-опции] старый_файл новый_файл

Примеры:

1 Переименование файлов в текущем каталоге. Изменить название файла april

на july в домашнем каталоге:

cd mv april july

2 Перемещение файлов в другой каталог. Переместить файл july в каталог

monthly.00:

mv july monthly.00

ls monthly.00

Результат:

april july june may

Если необходим запрос подтверждения о перезаписи файла, то нужно использовать опцию i.

3 Переименование каталогов в текущем каталоге. Переименовать каталог

monthly.00 в monthly.01

mv monthly.00 monthly.01

4 Перемещение каталога в другой каталог. Переместить каталог monthly.01в каталог reports:

mkdir reports

mv monthly.01 reports

5 Переименование каталога, не являющегося текущим. Переименовать каталог

reports/monthly.01 в reports/monthly:

mv reports/monthly.01 reports/monthly

Права доступа

Каждый файл или каталог имеет права доступа (табл.1).

Таблица 1

Права доступа

Право	Обозначение	Файл	Каталог
Чтение	r	Разрешены просмотр и копирование	Разрешён просмотр списка входящих файлов
Запись	w	Разрешены изменение и переименование	Разрешены создание и удаление файлов
Выполнение	x	Разрешено выполнение файла (скриптов и/или программ)	Разрешён доступ в каталог и есть возможность сделать его текущим

В сведениях о файле или каталоге указываются:

– тип файла (символ (-) обозначает файл, а символ (d) — каталог);

– права для владельца файла (r— разрешено чтение, w— разрешена запись, x—

разрешено выполнение, -— право доступа отсутствует);

– права для членов группы (r — разрешено чтение, w — разрешена запись, x —

разрешено выполнение, -— право доступа отсутствует);

– права для всех остальных (r— разрешено чтение, w— разрешена запись, x—

разрешено выполнение, -— право доступа отсутствует).

Примеры:

1 Для файла (крайнее левое поле имеет значение -) владелец файла имеет право на чтение и запись (rw-), группа, в которую входит владелец файла, может читать

файл (r--), все остальные могут читать файл (r--):

-rw-r--r--

2 Только владелец файла имеет право на чтение, изменение и выполнение файла:

-rwx------

3 Владелец каталога (крайнее левое поле имеет значение d) имеет право на просмотр, изменение и доступа в каталог, члены

группы могут входить и просматривать его, все остальные — только входить в каталог:

drwxr-x--x

Изменение прав доступа

Права доступа к файлу или каталогу можно изменить, воспользовавшись командой chmod. Сделать это может владелец файла (или каталога) или пользователь

с правами администратора.

Формат команды:

chmod режим имя_файла

Режим (в формате команды) имеет следующие компоненты структуры и способ

записи:

= установить право

- лишить права

+ дать право

r чтение

w запись

x выполнение

u (user) владелец файла

g (group) группа, к которой принадлежит владелец файла

o (others) все остальные

В работе с правами доступа можно использовать их цифровую запись (восьмеричное значение) вместо символьной (табл. 2).

Таблица 2

Формы записи прав доступа

Двоичная	Восьмеричная	Символьная
111	7	rwx
110	6	rw-
101	5	r-x
100	4	r--
011	3	-wx
010	2	-w-
001	1	--x
000	0	---

Примеры:

1 Требуется создать файл ~/may с правом выполнения для владельца:

cd touch may

ls -l may

chmod u+x may

ls -l may

2 Требуется лишить владельца файла ~/may права на выполнение:

chmod u-x may

ls -l may

3 Требуется создать каталог monthly с запретом на чтение для членов группы и всех остальных пользователей:

cd

mkdir monthly

chmod g-r, o-r monthly

4 Требуется создать файл ~/abc1 с правом записи для членов группы:

cd

touch abc1

chmod g+w abc1

Анализ файловой системы

Файловая система в Linux состоит из фалов и каталогов. Каждому физическому носителю соответствует своя файловая система.

Существует несколько типов файловых систем. Перечислим наиболее часто встречающиеся типы:

– ext2fs (second extended filesystem);
– ext2fs (third extended file system);
– ext4 (fourth extended file system);
– ReiserFS;
– xfs;
– fat (file allocation table);
– ntfs (new technology file system).

Для просмотра используемых в операционной системе файловых систем можно воспользоваться командой mount без параметров. В результате её применения

можно получить примерно следующее:

```
mount
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec)
udev on /dev type tmpfs (rw,nosuid)
devpts on /dev/pts type devpts (rw,nosuid,noexec)
/dev/sda1 on /mnt/a type ext3 (rw,noatime)
/dev/sdb2 on /mnt/docs type reiserfs (rw,noatime)
shm on /dev/shm type tmpfs (rw,noexec,nosuid,nodev)
usbfs on /proc/bus/usb type usbfs
(rw,noexec,nosuid,devmode=0664,devgid=85)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc
(rw,noexec,nosuid,nodev)
nfsd on /proc/fs/nfs type nfsd (rw,noexec,nosuid,nodev)
```

В данном случае указаны имена устройств, названия соответствующих им точек

монтирования (путь), тип файловой системы и параметрами монтирования.

В контексте команды mount устройство—специальный файл устройства, с помощью которого операционная система получает доступ к аппаратному устройству.

Файлы устройств обычно располагаются в каталоге /dev, имеют сокращённые имена (например, sdaN, sdbN или hdaN, hdbN, где N — порядковый номер устройства,

sd — устройства SCSI, hd — устройства MFM/IDE).

Точка монтирования— каталог (путь к каталогу), к которому присоединяются файлы устройств.

Другой способ определения смонтированных в операционной системе файловых систем — просмотр файла/etc/fstab. Сделать это можно например с помощью команды cat:

cat /etc/fstab

/dev/hda1 / ext2 defaults 1 1

/dev/hda5 /home ext2 defaults 1 2

/dev/hda6 swap swap defaults 0 0

/dev/hdc /mnt/cdrom auto umask=0,user,noauto,ro,exec,users 0 0

none /mnt/floppy supermount dev=/dev/fd0,fs=ext2:vfat,--,

sync,umask=0 0 0

none /proc proc defaults 0 0

none /dev/pts devpts mode=0622 0 0

В каждой строке этого файла указано:

– имя устройство;

– точка монтирования;

– тип файловой системы;

– опции монтирования;

– специальные флаги для утилиты dump;

– порядок проверки целостности файловой системы с помощью утилиты fsck.

Для определения объёма свободного пространства на файловой системе можно воспользоваться командой df, которая выведет на экран список всех файловых

систем в соответствии с именами устройств, с указанием размера и точки монтирования. Например:

df

Filesystem 1024-blocks Used Available Capacity Mounted on

/dev/hda3 297635 169499 112764 60% /

С помощью команды fsck можно проверить (а в ряде случаев восстановить)

целостность файловой системы:

Формат команды:

fsck имя_устройства

Пример:
fsck /dev/sda1

Задание к лабораторной работе

1 Выполните все примеры, приведённые в первой части описания лабораторной работы.

2 Выполните следующие действия, зафиксировав в отчёте по лабораторной работе используемые при этом команды и результаты их выполнения:

2.1. Скопируйте файл /usr/include/sys/io.h в домашний каталог и назовите его equipment. Если файла io.h нет, то используйте любой другой файл в каталоге /usr/include/sys/ вместо него.

2.2. В домашнем каталоге создайте директорию ~/ski.plases.

2.3. Переместите файл equipment в каталог ~/ski.plases.

2.4. Переименуйте файл ~/ski.plases/equipment в ~/ski.plases/equiplist.

2.5. Создайте в домашнем каталоге файл abc1 и скопируйте его в каталог ~/ski.plases, назовите его equiplist2.

2.6. Создайте каталог с именем equipment в каталоге ~/ski.plases.

2.7. Переместите файлы ~/ski.plases/equiplist и equiplist2 в каталог ~/ski.plases/equipment.

2.8. Создайте и переместите каталог ~/newdir в каталог ~/ski.plases и назовите его plans.

3 Определите опции команды chmod, необходимые для того, чтобы присвоить перечисленным ниже файлам выделенные права доступа, считая, что в начале таких прав нет:

3.1. drwxr--r-- ... australia

3.2. drwx--x--x ... play

3.3. -r-xr--r-- ... my_os

3.4. -rw-rw-r-- ... feathers

При необходимости создайте нужные файлы.

4 Проделайте приведённые ниже упражнения, записывая в отчёт по лабораторной работе используемые при этом команды:

4.1. Просмотрите содержимое файла /etc/password.

4.2. Скопируйте файл ~/feathers в файл ~/file.old.

4.3. Переместите файл ~/file.old в каталог ~/play.

4.4. Скопируйте каталог ~/play в каталог ~/fun.

4.5. Переместите каталог ~/fun в каталог ~/play и назовите его games.

4.6. Лишите владельца файла ~/feathers права на чтение.

4.7. Что произойдёт, если вы попытаетесь просмотреть файл ~/feathers командой cat?

4.8. Что произойдёт, если вы попытаетесь скопировать файл ~/feathers?

4.9. Дайте владельцу файла ~/feathers право на чтение.

4.10. Лишите владельца каталога ~/play права на выполнение.

4.11. Перейдите в каталог ~/play. Что произошло?

4.12. Дайте владельцу каталога ~/play право на выполнение.

5 Прочитайте man по командам mount, fsck, mkfs, kill и кратко их охарактеризуйте, приведя примеры.

Содержание отчёта

Отчёт должен включать:

1 титульный лист;

2 формулировку цели работы;

3 описание результатов выполнения задания:

– снимки экрана (скриншоты) с результатами выполнения команд;

– ответы на вопросы;

4 выводы, согласованные с целью работы;

5 ответы на вопросы.

Контрольные вопросы

1 Дайте характеристику каждой файловой системе, существующей на жёстком

диске компьютера, на котором вы выполняли лабораторную работу.

2 Приведите общую структуру файловой системы и дайте характеристику каждой

директории первого уровня этой структуры.

3 Какая операция должна быть выполнена, чтобы содержимое некоторойфайловой

системы было доступно операционной системе?

4 Назовите основные причины нарушения целостности файловой системы. Как

устранить повреждения файловой системы?

5 Как создаётся файловая система?

6 Дайте характеристику командам, которые позволяют просмотреть текстовые

файлы.

7 Приведите основные возможности команды cp в Linux.

8 Назовите и дайте характеристику командам перемещения и переименования

файлов и каталогов.

9 Что такое права доступа? Как они могут быть изменены?

При ответах на вопросы используйте дополнительные источники информации

по теме.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Таненбаум Э., ван Стеен М. Распределенные системы. Принципы и парадигмы. – СПб: Питер, 2003. – 877 с.: ил.

2. Тель Ж. Введение в распределенные алгоритмы. Пер. с англ. – М.: МЦНМО, 2009. – 616 с.

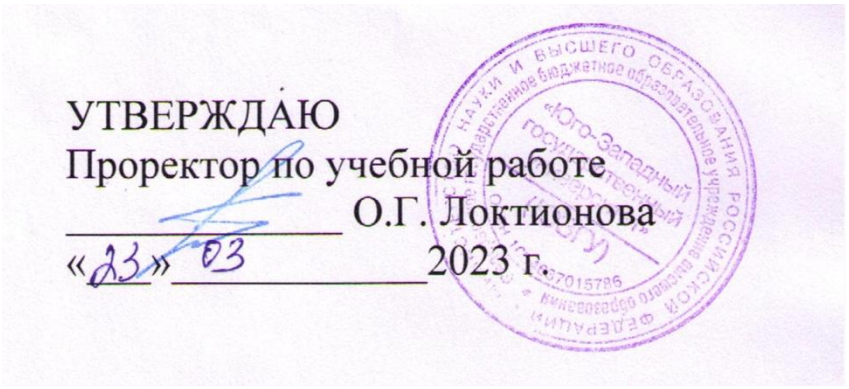
3. Топорков В. В. Модели распределенных вычислений. – М.: Физматлит, 2004. – 320 с.

4. Эндрюс Г. Р. Основы многопоточного, параллельного и распределенного программирования. Пер. с англ. – М.: Издательский дом "Вильямс", 2003. – 512 с.: ил.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательноеучреждения
высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности



ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ ВИДЕОКАРТЫ С ПОДДЕРЖКОЙ ТЕХНОЛОГИИ CUDA В СРЕДЕ MICROSOFT VISUAL STUDIO

Методические указания по выполнению лабораторных работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.65

Составитель: А.В. Митрофанов

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Определение параметров видеокарты с поддержкой технологии Cuda в среде Microsoft Visual Studio: методические указания по выполнению лабораторной работы по дисциплине «безопасность распределенных баз данных» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов. – Курск, 2023. – 10с. Библиогр.: с. 10.

Содержат сведения по разработке программных средств с использованием инструментария NVidia CUDA на современных языках программирования высокого уровня в среде Microsoft Visual Studio.

Методические указания по выполнению лабораторных работ по дисциплине «Безопасность систем баз данных», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ ____. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

Основные теоретические положения

Цель работы: иться с особенностями интеграции инструментария CUDA в среде разработки Мпознакомicrosoft Visual Studio, научиться определять основные параметры видеокарт, влияющие на скорость вычислений.

Настройка среды Microsoft Visual Studio

Инструментарий CUDA предоставляет возможность интеграции в состав популярной среды разработки Microsoft Visual Studio, что избавляет разработчика от необходимости напрямую обращаться к компилятору `nvcc` с использованием командной строки или разрабатывать `make`-файлы. Для создания проекта с поддержкой CUDA необходимо (подразумевается, что инструментарий CUDA установлен на машине):

1. Создать новый проект.
2. Добавить к нему необходимые `.cu`-файлы.
3. Для каждого из них указать инструмент, с помощью которого производится компиляция (правый клик по файлу в дереве проекта → Properties → General → Tool → в выпадающем списке выбрать «CUDA Runtime API»):

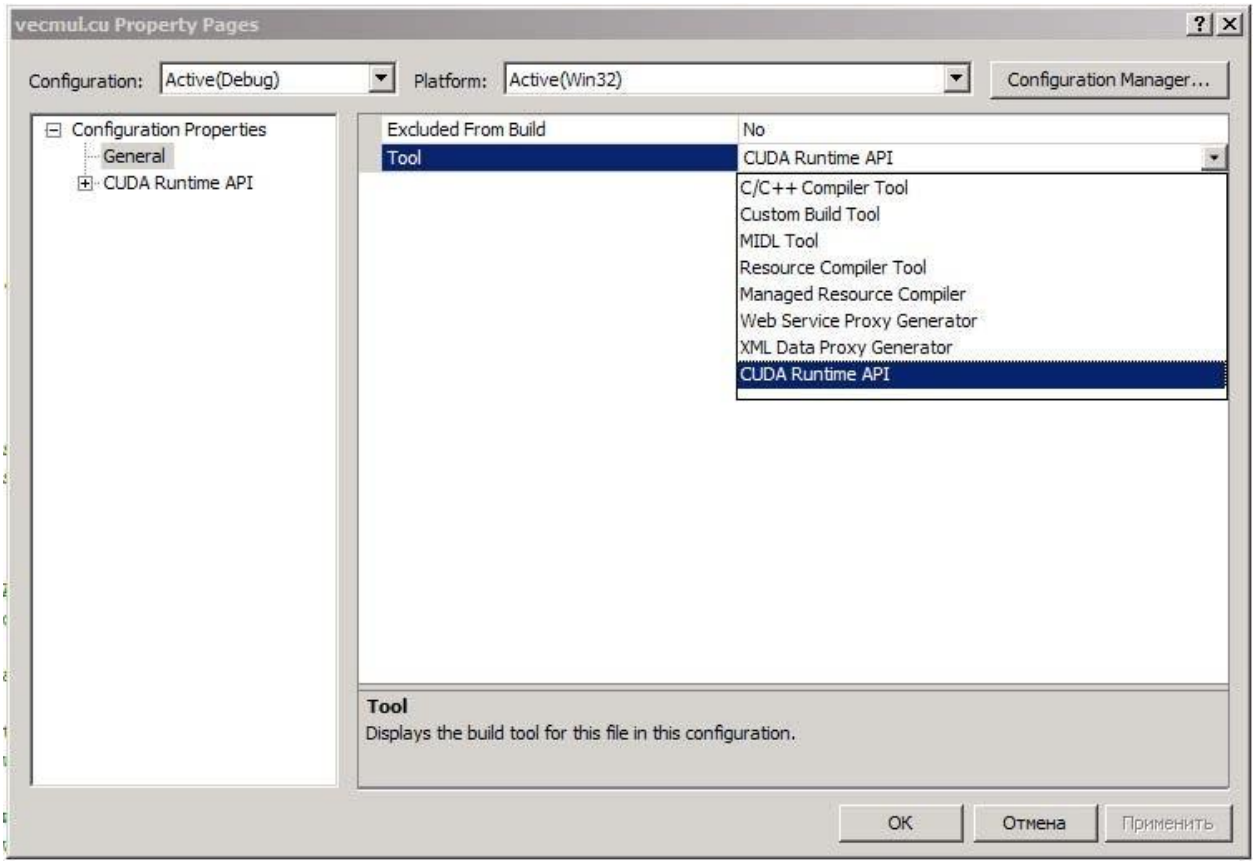


Рис. 1. Выбор типа компиляции для `.cu`-файлов

4. Указать путь к необходимым статически подключаемым библиотекам (выбрать Главное меню → Project → Properties → Configuration properties → Linker → Input → Additional dependencies, указать значение

```
"C:\Program Files\NVIDIA GPU Computing
Toolkit\CUDA\v5.0\lib\Win32\cudart.lib"
```

(в кавычках, путь показан на примере Windows XP x86!)

Другой вариант – указать в коде программы ссылку на библиотеку:

```
#pragma comment(lib,"cudart.lib")
```

5. Указать путь к расположению инструментария CUDA (выбрать Главное меню → Project → Properties → Configuration properties →

Linker → Additional library directories, указать значение `$(CUDA_LIB_PATH)`).

6. Указать путь к расположению заголовочных файлов CUDA (выбрать Главное меню → Project → Properties → Configuration properties → C/C++ → Additional include directories, указать значение

```
"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v5.0\include\"
```

(в кавычках, путь показан на примере Windows XP x86!)

После выполнения указанных действий среда разработки Microsoft Visual Studio будет самостоятельно вызывать компилятор `nvcc` для `.cu`-файлов, осуществлять компиляцию и сборку проекта.

Определение параметров видеокарты

Для определения числа установленных в системе видеокарт с поддержкой технологии CUDA используется функция `cudaGetDeviceCount()`:

```
int device_count;
cudaGetDeviceCount(&device_count);
```

Для определения свойств видеокарты используется функция `cudaGetDeviceProperties()`, принимающая в качестве параметра номер видеокарты и возвращающая в структуре `cudaDeviceProp` интересующие значения:

```
cudaDeviceProp dp;
cudaGetDeviceProperties(&dp, 0);  // Определение параметров GPU с номером 0
```

Например, для определения наименований установленных в системе видеокарт с поддержкой CUDA и их вычислительных возможностей можно использовать следующий код:

```
int device_count;
cudaDeviceProp dp;

cudaGetDeviceCount(&device_count);
cout << "CUDA device count: " << device_count << "\n";

for (int i=0; i<device_count; i++)
{
      cudaGetDeviceProperties(&dp, i);

      cout << i << ": " << dp.name << " with CUDA compute compatibility " <<
           dp.major << "." << dp.minor << "\n";
}
```

Задание

1. Создать в среде Microsoft Visual Studio проект, состоящий из пары файлов (`.cpp` и `.cu`) из предыдущей работы. Установить в среде необходимые настройки. Убедиться в том, что проект успешно собирается и запускается.
2. Определить число видеокарт с поддержкой технологии CUDA и следующие параметры видеокарты:
    - наименование;
    - общий объем графической памяти;
    - объем памяти констант;
    - объем разделяемой памяти в пределах блока;
    - число регистров в пределах блока;
    - размер WARP’а;
    - максимально допустимое число потоков в блоке;
    - версию вычислительных возможностей;
    - число потоковых мультипроцессоров;
    - тактовую частоту ядра;

- частоту памяти видеокарты;
- объем кэша второго уровня;
- ширину шины памяти видеокарты;
- максимальную размерность при конфигурации потоков в блоке и блоков в сетке.

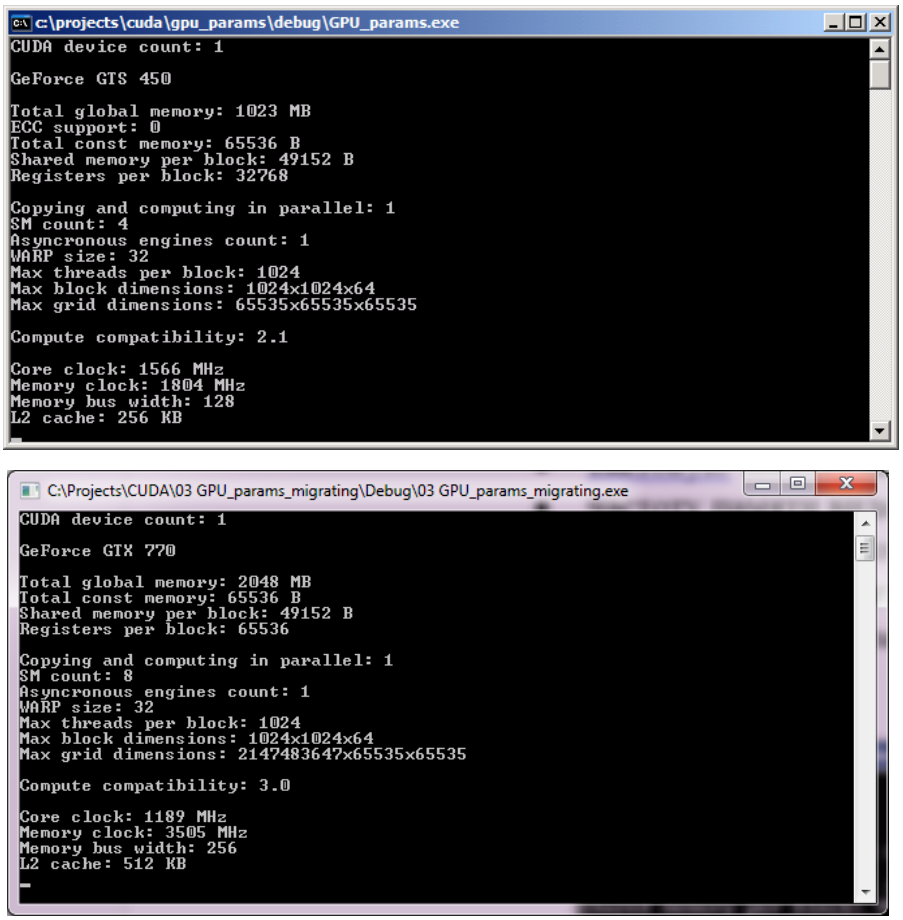


Рис. 2. Основные параметры видеокарты, влияющие на производительность вычислений

Содержание отчета

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг программы.
5. Скриншоты с результатами работы программы.
6. Выводы.

Контрольные вопросы

1. Для чего предназначен инструментарий CUDA?
2. Как производится программирование NVidia GPU с использованием CUDA?
3. Как производится сборка программы для ее запуска на NVidia GPU?
4. Как инструментарий CUDA интегрируется в состав среды Microsoft Visual Studio?

Библиографический список

1. Емельянов С.Г., Ватутин Э.И., Панищев В.С., Титов В.С. Процедурно-модульное программирование на Delphi: учебное пособие. М.: Аргамак-Медиа, 2014. 352 с.
2. Зотов И.В., Ватутин Э.И., Борзов Д.Б. Процедурно-ориентированное программирование на С++: учебное пособие. Курск: КурскГТУ, 2008. 211 с.