Школа-семинар

«Расширенные возможности пакета OpenFOAM»

ВВЕДЕНИЕ

М.В. Крапошин (НИЦ Курчатовский Институт)
О.И. Самоваров (Институт Системного Программирования РАН)
С.В. Стрижак (ГОУ ВПО МГТУ им. Баумана)

ЦЕЛИ КУРСА

- I. Ознакомление с использованием метода конечных объемов при решении задач вычислительной гидродинамики и его реализацией в пакете OpenFOAM.
- II. Освоение инструментария разработчика, предоставляемого платформой UniHUB при разработке приложений с использованием объектно-ориентированного языка программирования C++.
- III. Получение навыков по разработке приложений, использующих библиотеку OpenFOAM.
- IV. Изучение путей доработки модулей пакета OpenFOAM.

СОДЕРЖАНИЕ КУРСА

ДЕНЬ ПЕРВЫЙ

1. ВВЕДЕНИЕ
2. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ЯЗЫКА C++
3. РАЗРАБОТКА ПРИЛОЖЕНИЙ НА ЯЗЫКЕ C++ В СРЕДЕ UNIHUB
4. ПРИЛОЖЕНИЯ OPENFOAM
5. САМОСТОЯТЕЛЬНАЯ РАБОТА

ДЕНЬ ВТОРОЙ

6. ДЕТАЛЬНЫЙ ОБЗОР СТРУКТУРЫ ИСХОДНОГО КОДА OPENFAOM
7. ПОШАГОВЫЙ РАЗБОР ИСХОДНОГО КОДА РЕШАТЕЛЯ
8. САМОСТОЯТЕЛЬНАЯ РАБОТА

СОДЕРЖАНИЕ РАЗДЕЛА

- Цели и задачи курса, содержание курса (да)
- Метод конечного объема (да)
- Реализация метода конечного объема (да)
- Средства разработки (компиляция, системы сборки (да, намечено, ссылка на SVN?)
- О средах распределенной разработки (намечено)
- Обзор модулей OpenFOAM (да, намечено)

ПРЕДМЕТ РАССМОТРЕНИЯ

$$\frac{1}{c^2}\frac{\partial^2 p'}{\partial t^2} - \Delta p' = S \qquad \frac{\partial u}{\partial t} + \nabla\cdot(\boldsymbol{V} u) = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\boldsymbol{V}\rho) = 0$$

$$\rho C\frac{\partial T}{\partial t} + \rho C\nabla\cdot(\boldsymbol{V} u) - \nabla\cdot\lambda\nabla T = 0$$

$$\frac{\partial \boldsymbol{V}}{\partial t} + \nabla\cdot(\boldsymbol{V}\boldsymbol{V}) - \nabla\cdot\nu\left(\nabla\boldsymbol{V} + (\nabla\boldsymbol{V})^T\right) = -\nabla p$$

КАК РЕШАТЬ ЧИСЛЕННО?

- Представить расчетную область (пространство + время) в виде дискретного набора точек (областей)
  - Метод конечных разностей — заменить производные на их разностные аналоги
  - Метод конечных объемов — рассматривать интегральные соотношения для примитивных объемов, линеаризовать нелинейные уравнения
  - Метод конечных элементов — представить искомое решение в виде суперпозиции функций формы
- Другие методы — дискретный метод Галеркина и пр. - комбинации предыдущих

МЕТОД КОНЕЧНОГО ОБЪЁМА (1)

- Рассматривается интегральная формулировка — уравнения сохранения
- Для перехода от интегралов по объему к интегралам по поверхности используется теорема Остроградского-Гаусса
- Расчетная область разбивается на контрольные не пересекающиеся объемы, ограниченные плоскостями
- Точное соблюдение консервативности

МЕТОД КОНЕЧНОГО ОБЪЁМА (2)

Уравнения рассматриваются в интегральной формулировке.

Соотношение консервативности выполняется точно

Скорость изменения φ в объеме V	+	Поток через границу dS объема dV	=	Источник в dV

$$\underbrace{\frac{\partial \rho \phi}{\partial t}}_{\text{производная по времени}} + \underbrace{\nabla\cdot(\rho \boldsymbol{U} \phi)}_{\text{конвекционный член}} - \underbrace{\nabla\cdot(\Gamma_\phi \nabla \phi)}_{\text{диффузионный член}} = \underbrace{S_\phi(\phi)}_{\text{источник}}$$

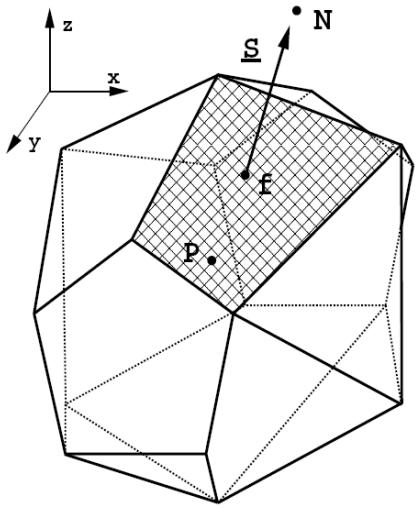
Скорость изменения φ в объеме dV	+	Диффузия + конвекция через dS	=	Источник в dV

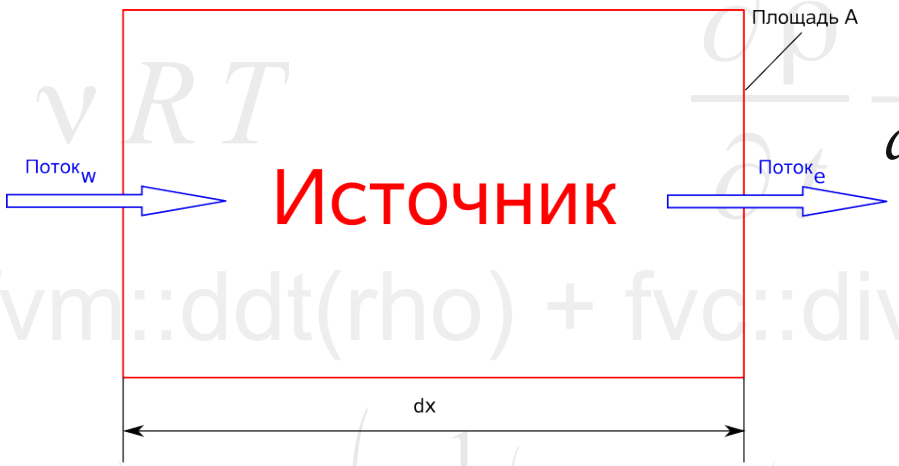
В основе — теорема Остроградского - Гаусса

$$\int_V \nabla * \psi \partial V = \int_S \partial \boldsymbol{S} * \psi$$

$$V \frac{d\psi}{dt} = -\left(\sum_f F_c(\psi) + \sum_f F_D(\psi)\right) + S_V(\psi)$$

МЕТОД КОНЕЧНОГО ОБЪЁМА (3)





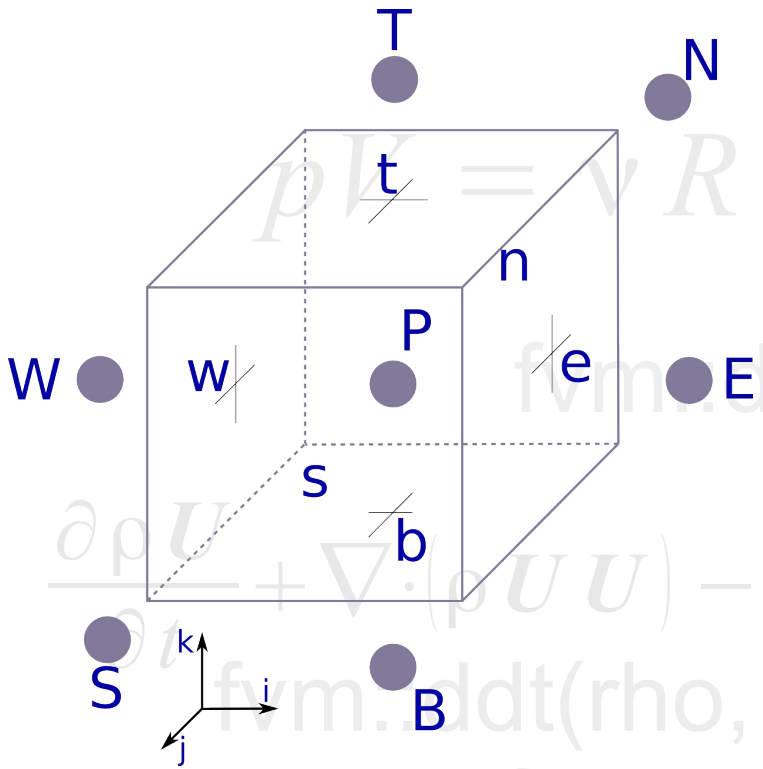
$$a_P \phi_P - \sum_F a_F \phi_F = b_P$$

$$F = \{ E, W \}$$

$$\frac{d}{dx}\phi = C_e \phi_e - C_w \phi_w \qquad C_e = (\rho u A)_e \qquad C_w = (\rho u A)_w$$

$$-\frac{d}{dx}\Gamma \frac{d}{dx}\phi = -[D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W)] \qquad D_e = \left(\frac{\Gamma A}{\Delta x}\right)_e \qquad D_w = \left(\frac{\Gamma A}{\Delta x}\right)_w$$

МЕТОД КОНЕЧНОГО ОБЪЁМА (4)



$$a_P \phi_P - \sum_F a_F \phi_F = b_P$$

$$F = \{ E, W, N, S, T, B \}$$

Система решается итеративным методов

- Метод сопряженных градиентов
- Метод Гаусса-Зейделя
- Многосеточные методы

МЕТОД КОНЕЧНОГО ОБЪЁМА (5)

- Программисту нужно реализовать:
  - a)Общее управление задачей
  - b)Поддержку расчетной сетки
  - c)Дискретизацию по пространству и времени
  - d)Хранение матрицы и полей в памяти
  - e)Учет граничных (и начальных) условий
  - f) Параллельные вычисления — синхронизацию данных
  - g)Ввод/вывод данных
  - h)Обработку и визуализацию данных

Средства реализации — выбор языка программирования

- При всем желании, выбор не велик:
  - Процедурные языки (C/Fortran)
  - Объектно-ориентированные языки (C++)
  - Языки высокого уровня (интерпретаторы) — Octave, MatLab

Реализация метода конечных объемов - OpenFOAM – Field Operation And Manipulation

a) Хранение объектов задачи - objectRegistry

b) Поддержка сетки — polyMesh, fvMesh

c) Дискретизация по времени — Time, по пространсту — GeoMesh<....>

d) Хранение матрицы — fvMatrix, fvScalarMatrix, lduMatrix

e) Граничные условия — fvPatch, fvPatchField

f) Параллельные вычисления — Pstream

g) Ввод/вывод данных — IOobject

h) Обработка данных — functionObjects, postCalc, foamCalcFunctions

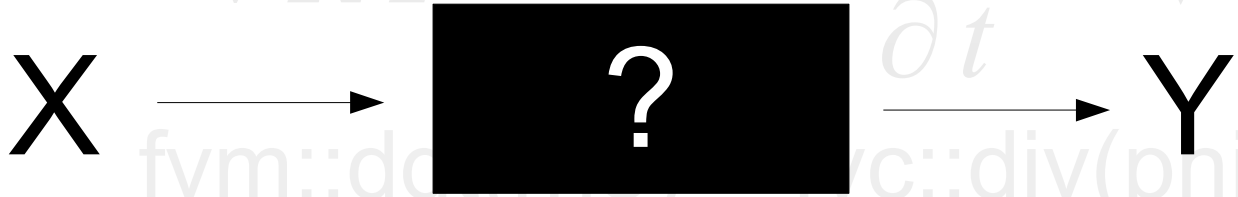
i) Визуализация — foamToVTK, ParaView

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

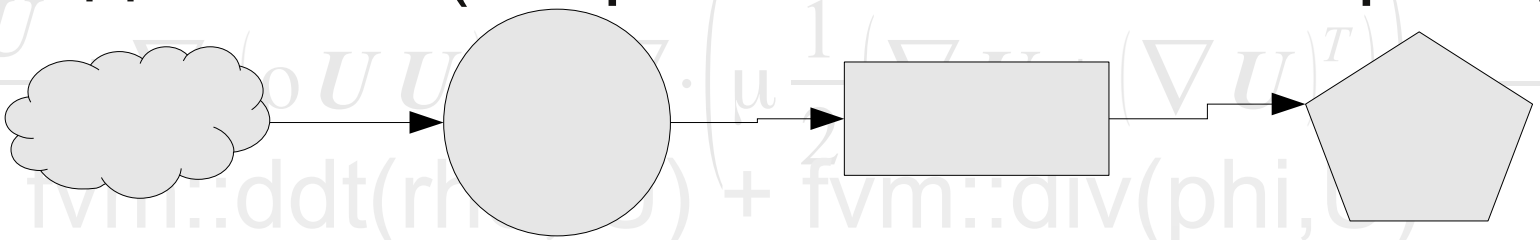
- ПРОЦЕДУРНОЕ — предполагает понимание как всей задачи в целом, так и каждой её части
- ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ — намечаются цели, которые далее можно решать некоторым (абстрактным) способом. Стратегия отделена от тактики

ТРИ КИТА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

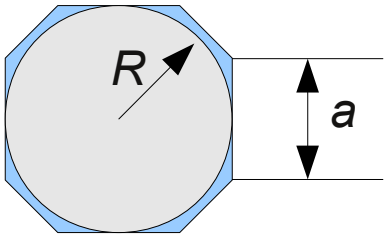
- Инкапсуляция (сокрытие данных)



- Наследование (сохранение базовых принципов)



- Полиморфизм (плюрализм методов решения)



$$S = \pi R^2 = \pi \frac{D^2}{4} = \frac{1}{4\pi} l^2 \approx n \times a$$

ОТЛИЧИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО (C++) ОТ ПРОЦЕДУРНОГО СТИЛЯ ПРОГРАММИРОВАНИЯ (F77)

Процедурное Программирование. Базовые единицы — переменная и функция. Стратегия программы определяется до написания исходного в виде блок-схем или псевдокода. Пример: Fortran, C, Pascal

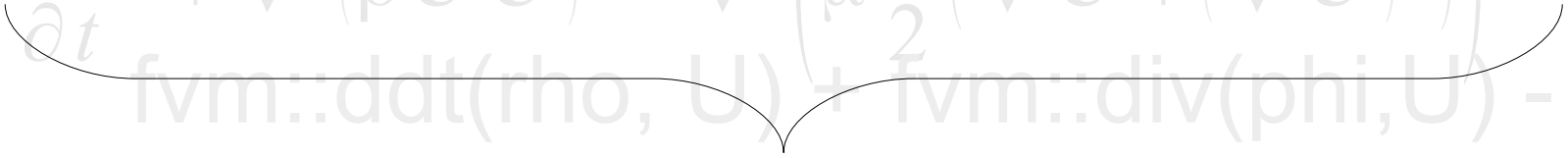
```
subroutine sum (a,b,c)
  real*4 a, b, c
  c = a + b
end subroutine
```

Объектно-ориентированное программирование. Базовая единица — класс, содержащий в себе и данные (атрибуты) класса и методы

```
class A{
  float val;
  const A& sum(const A& a, const A& b){
    val = a.val + b.val;
    return *this;
  }
};
```

ПРОЦЕСС КОМПИЛЯЦИИ

- **Препроцессор gcc** — осуществляет синтаксический анализ исходного кода, условную компиляцию
- **Транслятор gcc** — транслирует исходный код в объектные файлы
- **Сборщик gcc (ld)** — собирает объектные файлы в исполняемый либо в динамическую библиотеку



- **ELF** — Executable Linux Format — в ОС Linux формат исполняемых файлов, объектного кода и библиотек регламентируется единым стандартом

GCC — НАБОР КОМПИЛЯТОРОВ

- gcc — компиляция исходного кода языка C
- g++ — компиляция исходного кода языка C++ (cxx)
- gfortran — компиляция исходного кода Fortran форматов 77/90/95/2003
- Другие компиляторы (java, ada, obj-c)

```
gcc main.c -oc-prog -lm

g++ main.cxx -ocxx-prog -lm

gfortran main.f90 -of90-prog -lm
```

ТИПЫ ФАЙЛОВ

- Типы файлов в ОС Linux определяются не по расширению, а по содержимому (можно узнать по команде `file <имя_файла>`):
  - Файлы исходного кода (.C, .H, .cxx, .hxx, .c, .h, .cpp, .hpp, .f, .F, .f90, .py, .S)
  - Файлы объектных модулей - .o, .obj, .ko
  - Файлы статических библиотек (архив объектных модулей) — lib***.a, где *** - имя библиотеки
  - Исполняемые файлы ELF — любое расширение
  - Динамические библиотеки — lib***.so.###, где *** - имя библиотеки, ### - версия

ОПЦИИ КОМПИЛЯТОРА

- `-I` — каталоги с подключаемыми файлами
- `-L` — каталоги с библиотеками
- `-l<...>` - имя библиотеки
- `-m64` — компиляция 64-разрядных программ
- `-c` — компилировать объектный код (без сборки)

Все исполняемые файлы могут быть собраны из объектного кода или с помощью ld или одним из компиляторов (определяют набор подключаемых библиотек по умолчанию)

```
gcc obj1.o obj2.o obj3.o -oexec
```

СИСТЕМЫ СБОРКИ

- Система сборки make — makefile — устанавливает зависимости между исходными файлами — как и в какой последовательности выполнять компиляцию и сборку. Сложный и неоднозначный синтаксис.
- Надстройки над make:
  - autoconf — построение makefile по имеющемуся скрипту
  - cmake — построение makefile по имеющемуся файлу настроек CMakeLists
  - wmake — построение makefile в соответствии с файлами Make/files и Make/options

ЗАВИСИМОСТИ МЕЖДУ ФАЙЛАМИ

fprotos.hxx

```
double circleArea (double r);
double squareArea (double a);
```

circle.cxx

```
#include <fprotos.hxx>
#include <cmath>

double circleArea (double r)
{
    return r*r*M_PI;
}
```

square.cxx

```
#include <fprotos.hxx>

double squareArea (double r)
{
    return r*r;
}
```

main.cxx

```
#include <iostream>
#include <fprotos.hxx>

int main (int argc, char * argv[])
{
    std::cout <<
    "Area of circle with unit radius is:"
    << circleArea(1.0) << std::endl;
    std::cout <<
    "Area of square with side length "
     << 2 << " is: " << squareArea(2.0)
     << std::endl;
}
```

MAKEFILE

```
CXX  = g++ -c -I. -O3
LINK = g++ -lm -m64

main: square.obj circle.obj main.obj
        $(LINK) main.obj square.obj
circle.obj -omain

circle.obj: circle.cxx fprotos.hxx
        $(CXX) circle.cxx -ocircle.obj

square.obj: square.cxx fprotos.hxx
        $(CXX) square.cxx -osquare.obj

main.obj: main.cxx fprotos.hxx
        $(CXX) main.cxx -omain.obj


all: main


clean:
        rm -rf *.obj
        rm -rf main
```

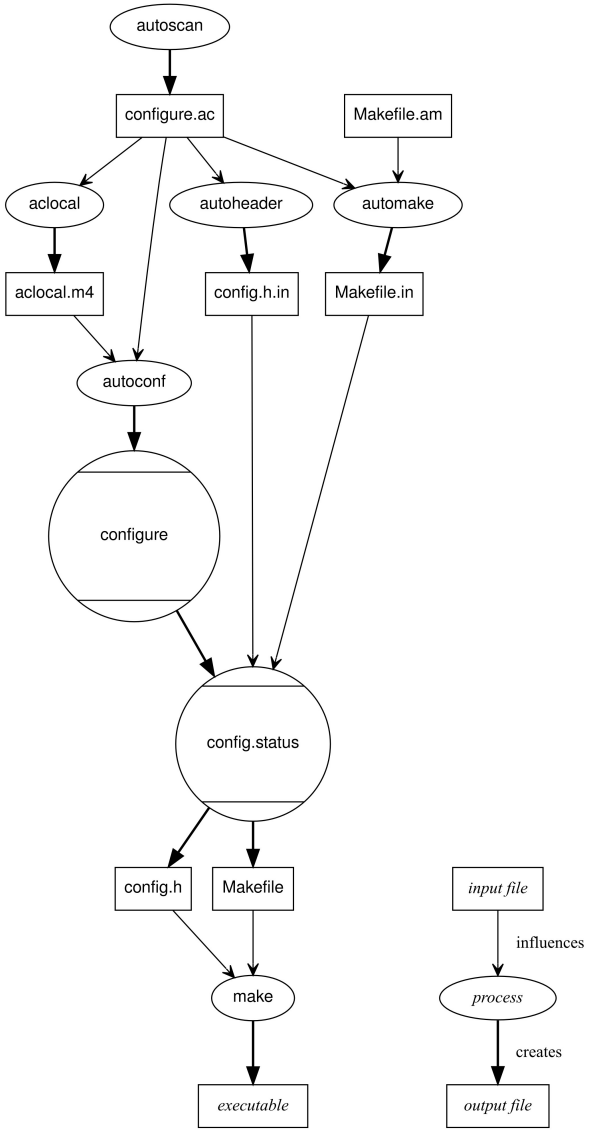
Можно определять переменные и использовать их

Каждая запись — это «цель», её зависимости (те файлы, от которых зависит цель) и правила сборки (компиляции) цели

По умолчанию, первая цель является главной (собирается по команде **make**)

Другие цели —
make all: обрать цель main
make clean: очистить рабочий каталог

AUTOCONF/CONFIGURE



- Материалы взяты из Wikipedia
- Полное название системы — GNU Autotools
- Autoconf читает configure.in или configure.ac и Makefile.am для генерации скрипта configure и Makefile.in
- Полученный скрипт configure читает Makefile.in и генерирует Makefile
- По правилам Makefile выполняется компиляция
- Makefile.in создается из Makefile.am с использованием Automake

CMAKE

- Утилита использует единый файл CmakeLists.txt

```
#
# Minimum cmake version
#
cmake_minimum_required (VERSION 2.6)

project (test_main)

include_directories (.)

add_executable (main main.cxx circle.cxx
square.cxx fprotos.hxx)

target_link_libraries(main m)

#END_OF_FILE
```

СИСТЕМА СБОРКИ WMAKE

- Файлы исходного кода имеют имя .H или .C
- Список компилируемых файлов: Make/files
- Опции компиляции Make/options
- Каталог lnInclude — ссылки на все файлы исходного кода библиотеки

laplacianFoam.C

```
EXE = $(FOAM_APPBIN)/laplacianFoam
```

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude

EXE_LIBS = \
    -lfiniteVolume \
    -lIduSolvers
```

ШАГИ СБОРКИ — MAKE, CONFIGURE, CMAKE, WMAKE

- Сбор информации о системном окружении (cmake, autoconf/configure, wmake)
- Сбор целей — make / wmake
- Установка файлов в файловой структуре — make install / wmake

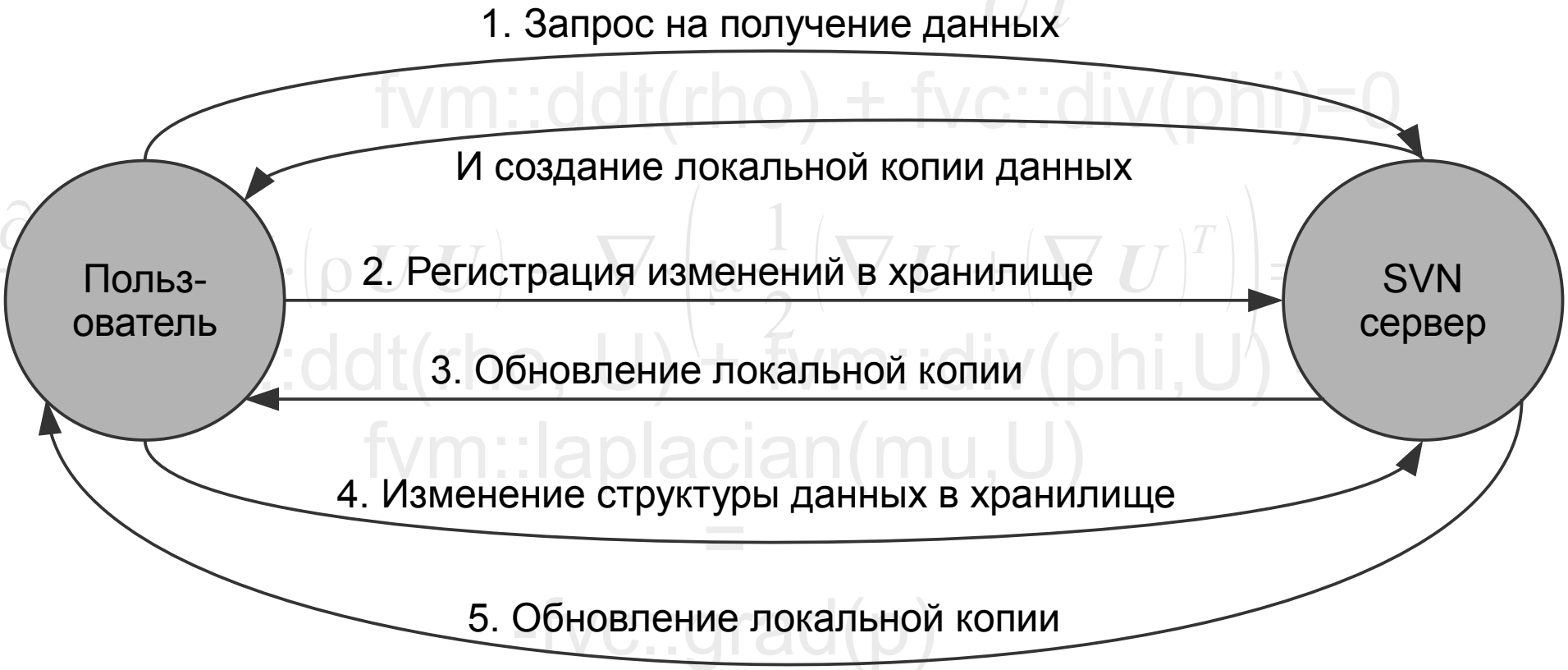
- В отличие от других систем, Wmake объединяет в себя все три этапа — анализ системного окружения, компиляция, установка

РАСПРЕДЕЛЕННАЯ РАЗРАБОТКА

- SVN — Система регистрации изменений
- КОРОТКО ОБ ECLIPSE
- КОРОТКО О ПРИЛОЖЕНИЯХ В UNIHUB

СИСТЕМА УЧЕТА ВЕРСИЙ SVN

SVN — SubVersioN — открытая система контроля изменений, её клиент может работать под любой ОС, включая Windows



Основные команды SVN

Получение справки:
```
svn help
```

Получение справки для команды svn:
```
svn help <имя_команды>
```

Команды основного цикла

1) Запрос на получение копии
```
svn checkout <имя_сервера>/<путь_к_данным>
```

2) Регистрация локальных изменений в хранилище
```
svn commit
```

3) Обновление локальной копии
```
svn update
```

4) Изменений структуры данных хранилища
```
svn add, svn del
```

В ~/.bashrc вставить строку export EDITOR=/usr/bin/mcedit

СТРУКТУРА КАТАЛОГОВ КУРСА

Скачать презентации курса и его рабочие материалы

svn co https://unihub.ru/tools/unicfdc2/svn/trunk/data

Папки Odp, Pdf, Swf, Odt — презентации

Папка Literature — подбор .pdf, полезных при изучении OpenFOAM и MCC

Подпапки Files — рабочие материалы курса

СТРУКТУРА КАТАЛОГОВ OPENFOAM (1)

- Рассматривается на примере OpenFOAM-1.6-ext
- Базовые переменные OpenFOAM:
  - FOAM_APPBIN (FOAM_USER_APPBIN) — исполняемые файлы OpenFOAM (_USER_ - локальные по отношению к пользователю)
  - FOAM_LIBBIN (FOAM_USER_LIBBIN) — библиотеки файлы OpenFOAM (_USER_ - локальные по отношению к пользователю)
  - FOAM_INST_DIR — каталог, в который установлен OpenFOAM
  - FOAM_SRC — каталог с исходным кодом OpenFOAM
  - FOAM_TUTORIALS — каталог с примерами OpenFOAM
- `export | grep FOAM_` - вывести все переменные, начинающиеся на FOAM_
- Или так: `export | grep -e "FOAM"` — вывести все переменные, содержащие FOAM в своем имени

СТРУКТУРА КАТАЛОГОВ OPENFOAM (2)

- **Macros** — макросы издательской системы LaTeX для оформления документации
- **ThirdParty** — используемые в OpenFOAM сторонние приложения (исходные коды в rpmBuild/SOURCES)
  - **packages/ParMGridGen-1.0** — библиотека генерации грубых сеток на основе заданной (для геометрических многосеточных методов)
  - **packages/ParMetis-3.1.1** — библиотека метода декомпозиции сетки ParMetis
  - **packages/ParaView-3.8.1** — приложение для визуализации ParaView
  - **packages/cmake-2.8.3** — система сборки CMake
  - **packages/libccmio-2.6.1** — библиотека для чтения/записи формата CCM CD-ADAPCO
  - **packages/mesquite-2.1.2** — библиотека для улучшения качества сеток
  - **packages/metis-5.0pre2** — библиотека декомпозиции по пространству методом metis
  - **packages/openmpi-1.4.3** — реализация стандарта MPI OpenMPI
  - **packages/scotch_5.1.10b** — библиотека декомпозиции по пространству методом scotch

СТРУКТУРА КАТАЛОГОВ OPENFOAM (3)

- **applications** — исходный код приложений OpenFOAM
  - **solvers** — решатели OpenFOAM
  - **utilities** — утилиты для пре- и постпроцессинга
- **bin** — скрипты OpenFOAM
- **lib** — скомпилированные библиотеки OpenFOAM
- **doc** — документация OpenFOAM
  - **Doxygen** — автоматически генерируемая документация к исходному коду (с содержание, глоссарием и перекрестными ссылками)
  - **Guides-a4** — руководства пользователя и программиста в формате Pdf
- **etc** — конфигурационные файлы (скрипты) OpenFOAM
- **tutorials** — примеры OpenFOAM (настроенные тестовые задачи)
- **wmake** — настройки, скрипты и исполняемые файлы системы сборки wmake

СТРУКТУРА КАТАЛОГОВ OPENFOAM (4)

- **src** — исходный код библиотек OpenFOAM
  - **ODE** — средства численного интегрирования ОДУ
  - **Pstream — работа с параллельными процессами (MPI)**
  - **OSspecific** — взаимодействие с системой (файлы, потоки выполнения, системное время и пр.)
  - **OpenFOAM** — базовая библиотека (примитивы величин (скаляр, вектор, тензор и др.), матрицы, поля, работа с памятью, размерность величин, база данных объектов и т. д.)
  - **finiteVolume** — метод конечного объёма (дискретизация полей методом К.О., конечно-объёмная сетка и матрица, поверхностная сетка, объемные и поверхностные поля)

СТРУКТУРА КАТАЛОГОВ OPENFOAM (5)

- **src** — исходный код библиотек OpenFOAM
  - **OSspecific** — взаимодействие с системой (файлы, потоки выполнения, системное время и пр.)
  - **OpenFOAM** — базовая библиотека (примитивы величин (скаляр, вектор, тензор и др.), матрицы, поля, работа с памятью, размерность величин, база данных объектов и т. д.)
  - **Pstream** — работа с параллельными процессами (MPI)
  - **finiteVolume** — метод конечного объёма (дискретизация полей методом К.О., конечно-объёмная сетка и матрица, поверхностная сетка, объемные и поверхностные поля)

ПРИЛОЖЕНИЯ ДЛЯ ПРЕ- И ПОСТ- ПРОЦЕССИНГА

$FOAM_APP/utilities

errorEstimation

Оценка погрешности численного решения уравнений

mesh

Утилиты для работы с сеткой

miscellaneous

Разнообразные утилиты, не отнесенные к другим группам

parallelProccesing

Декомпозиция и сбор расчетной области при параллельных вычислениях

postProcessing

Обработка результатов расчетов

preProcessing

Подготовка исходных данных

surface

Работа с поверхностями сеток

thermophysical

Расчет термодинамических параметров

Все вспомогательные приложения OpenFOAM имеют такую же структуру построения, как и решатели, но при этом отсутствует блок, связанный с решением СЛАУ. Утилиты предназначены для подготовки исходных данных и обработки результатов

ТИПЫ РЕШАТЕЛЕЙ ПО МОДЕЛЯМ

Класс задачи	Описание
DNS	Прямое численное моделирование течения несжимаемой жидкости (dnsFoam)
basic	**Простейшие задачи (потенциальное течение, транспорт скаляра)**
combustion	Задачи с горением и химическими реакциями (например, сжиганием топлива в двигателе)
compressible	Турбулентное течение сжимаемых сред (дозвуковые, транзвуковые и сверхзвуковые)
discreteMethods	Задачи с использованием дискретных методов (например Монте-Карло) для исследования течения жидкостей
electromagnetics	Задачи магнитогидродинамики
financial	Экономические задачи (например, уравнение Блэка-Шоулза)
heatTransfer	Турбулентное течение жидкости с теплообменом и учетом плавучести
incompressible	**Турбулентное течение несжимаемой жидкости**
lagrangian	Течение жидкостей с примесями, представленных частицами Лагранжа
multiphase	Движение многофазных частиц, в том числе с фазовыми превращениями
stressAnalysis	Задачи анализа прочности с использованием метода конечного объёма

СПАСБО ЗА ВНИМАНИЕ!

$$pV = \nu R T$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{U} = 0$$

fvm::ddt(rho) + fvc::div(phi)=0

$$\frac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho \boldsymbol{U} \boldsymbol{U}) - \nabla \cdot \left(\mu \frac{1}{2} \left(\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T \right) \right) = -\nabla p$$

fvm::ddt(rho, U) + fvm::div(phi,U) -
fvm::laplacian(mu,U)
=
-fvc::grad(p)