РОССИЙСКАЯ АКАДЕМИЯ НАУК

ИНСТИТУТ
ПРОБЛЕМ БЕЗОПАСНОГО
РАЗВИТИЯ АТОМНОЙ ЭНЕРГЕТИКИ

RUSSIAN ACADEMY OF
SCIENCES

NUCLEAR
SAFETY INSTITUTE

Препринт № NSI–16–93

Preprint NSI–16–93

В.В. Варенков, В.А. Первичко, А.Г.Пошков

САМОДОКУМЕНТИРУЕМЫЙ ФАЙЛ ДАННЫХ

Москва
1993

Moscow
1993

УДК 681.3.06

В.В. Варенков, В.А. Первичко, А.Г. Попков САМОДОКУМЕНТИРУЕМЫЙ ФАЙЛ ДАННЫХ. Препринт № NSI-16-93. Москва: Институт проблем безопасного развития атомной энергетики РАН, 1993. 19 с.

Аннотация

В работе рассматривается самодокументируемый файл данных, содержащий, кроме данных, описание своей структуры. При помощи описываемых процедур и утилит вычислительные пакеты могут работать с унифицированными файлами и иметь универсальные программные средства для подготовки входных данных и обработки полученных результатов.

V.V. VARENKOV, V.A. PERVICHKO, A.G. POPKOV SELF-DESCRIPTIVENESS DATA FILE. Preprint NSI-16-93. Moscow: Nuclear Safety Institute, June 1993. 19 p.

Abstract

The paper is devoted to self-descriptiveness file that contains data together with description of their structure. With help of described procedures and utilities applied packages can work with unified files and have universal program tools for preparing of input data and for processing of obtained results.

САМОДОКУМЕНТИРУЕМЫЙ ФАЙЛ ДАННЫХ

*В.В. Варенков, В.А. Первичко, А.Г. Попков**
Институт Проблем Безопасного Развития Атомной Энергетики РАН
Большая Тульская 52, Москва 113191, Россия

17 Июня 1993 года

Аннотация

В работе рассматривается самодокументируемый файл данных, содержащий, кроме данных, описание своей структуры. При помощи описываемых процедур и утилит вычислительные пакеты могут работать с унифицированными файлами и иметь универсальные программные средства для подготовки входных данных и обработки полученных результатов.

1 Введение

Одной из компонент больших программных кодов, предназначенных для математического моделирования физических процессов является подсистема работы с входными и выходными файлами данных, включающая в себя модули чтения/записи, контроля, преобразования данных. Во входных файлах содержится некоторая исходная информация, содержимое выходных файлов используется для просмотра, представления данных в виде графических зависимостей или для преобразования данных в целях получения дополнительной информации: экстремальных значений, интегральных величин, временных зависимостей и т.п. В качестве примера традиционной организации работы с входными/выходными данными можно привести программные коды по моделированию тяжелых аварий на атомных электростанциях CORCON [1], WECHSL [2].

Среди процедур визуализации и обработки результатов расчетов, можно выделить набор базовых, не зависящих от предметной области. Таковыми, например, являются процедуры рисования одномерных графиков, расчетных сеток, линий уровня, векторных полей. Специфика предметной области проявляется в типах и обозначениях величин, их размерностях, зависимостях одних величин от других, в последовательности действий по визуализации и обработке.

В файлах данных часто к числовой информации добавляют некоторый комментарий с названиями величин, указанием на особенности расчетного варианта и т.п. Однако различие в форматах файлов данных и отсутствие стандарта в описательной информации затрудняет создание универсальных средств обработки данных и передачу данных между различными программами.

Авторами разработан специальный формат файлов: SDF (Self–Descriptiveness Format). Для краткости файлы с такой структурой будем называть SDF–файлами.

SDF–файл — это файл данных, содержащий, кроме самих данных, описание своей структуры. При помощи предлагаемых в работе процедур и утилит вычислительные пакеты могут работать с унифицированными самодокументируемыми SDF–файлами и иметь универсальные программные средства для подготовки входных данных и обработки полученных результатов.

Разработка SDF–файлов была ориентирована на поддержку работ с пакетами прикладных программ в области математической физики, однако SDF–файлы могут быть полезны и для других целей, например, для организации баз данных в научных и инженерных приложениях.

SDF–файлы по своей структуре можно рассматривать как базы данных, но они отличаются от традиционных баз данных прежде всего ориентацией на вычислительную математику, а не на работу с документами или иной текстовой информацией. Добавление описательной

*E–mail: pbl@ibrae.msk.su

части (атрибутов) к данным позволяет считать их объектами и производить работу с ними на разных уровнях абстракции. Будем называть данные вместе с описательной частью переменными.

Описываемые в работе программные средства реализованы на персональном компьютере IBM PC/AT с графическим дисплеем типа EGA или VGA. При разработке программных средств использовались языки программирования C и FORTRAN

2 Структура SDF–файла

SDF–файл состоит из двух разделов: заголовка и собственно данных. Заголовок, в свою очередь, состоит из фиксированной части, содержащей ключевую информацию о SDF–файле, и части, включающей описания отдельных полей (переменных). Значения переменных, описанных в заголовке, образуют запись. Раздел данных SDF–файла состоит из последовательности записей с одинаковой структурой (рис. 1).

заголовок	описание файла описание переменной 1 описание переменной 2 .... описание переменной n
запись 1	переменная 1 переменная 2 .... переменная n
запись 2	переменная 1 переменная 2 .... переменная n
	.....
запись m	переменная 1 переменная 2 .... переменная n

Рис. 1. Структура SDF–файла

При моделировании эволюционных задач запись соответствует определенному моменту времени, а переменные записи характеризуют основные параметры процесса на этот момент.

Описание переменной включает·

- короткое имя (идентификатор);
- длинное имя (комментарий);
- дескриптор (целое число);
- тип данных (байтовый, символьный, целый, вещественный и т.п.);
- размерность (0, 1, 2, 3);
- размеры массива по x, y и z.

Короткое имя является ключевым полем и по нему осуществляется доступ, поиск и другие операции над переменной.

Дескриптор — это целое число для некоторой характеристики переменной (например, 0 — переменная определена в центрах ячеек разностной сетки, 1 — в узлах сетки)

Как правило, одна из переменных SDF–файла является ключевой для всей записи. Например, для решения нестационарных задач теплопереноса ключевой переменной является переменная "текущее время", а для базы данных по теплофизическим свойствам — "название вещества".

короткое имя	длинное имя	дескр	тип	размерность	размер по X	размер по Y	размер по Z
N	размер сетки по X	0	INT_	0	1	1	1
M	размер сетки по Y	0	INT_	0	1	1	1
L	индекс симметрии	0	INT_	0	1	1	1
time	текущее время	0	FLOAT_	0	1	1	1
dt	шаг по времени	0	FLOAT_	0	1	1	1
X	координаты X сетки	1	FLOAT_	2	51	46	1
Y	координаты Y сетки	1	FLOAT_	2	51	46	1
Te	температура	3	FLOAT_	2	51	46	1
Flux	поток	3	FLOAT_	2	51	46	1
Cv	теплоемкость	3	FLOAT_	2	51	46	1
Cap	теплопроводность	3	FLOAT_	2	51	46	1
SQ	источник	3	FLOAT_	2	51	46	1

Рис. 2. Пример набора переменных для вычислительной задачи

На рис. 2 приведен пример набора переменных для двумерной задачи теплопроводности.

В качестве основного формата SDF–файла используется двоичный, а не текстовый формат. Такой выбор основан на двух соображениях: 1) двоичные файлы занимают меньше места, что немаловажно при большом числе массивов в типичных вычислительных задачах и 2) легче позиционироваться при чтении/записи файла в файловых операциях. Двоичный формат позволяет более экономно использовать дисковое пространство и уменьшить количество обменов с диском. Однако текстовый файл обеспечивает более легкую переносимость данных на разные вычислительные машины. Поэтому целесообразно иметь программы–преобразователи из одного вида в другой.

С точки зрения уменьшения размеров SDF–файла целесообразно введение в него статической части, так как в вычислительных задачах существуют такие данные, которые не меняются в процессе счета (например, для задач механики сплошных сред на эйлеровой сетке остаются постоянными координаты узлов сетки и основные геометрические факторы). Дублирование неизменяемых массивов при большом количестве записей приведет к значительному увеличению размеров файла. Однако такой подход существенно усложняет структуру SDF–файла. Проще для хранения неменяющихся данных использовать отдельный SDF–файл с единственной записью.

Кроме используемых в SDF–файле типов данных (int, float, char) может возникнуть потребность в более сложных типах:

- тип "формула" — вычисление переменных (чисел и массивов) по формульному выражению;
- тип "ограничение" — задание условий, обеспечивающих проведение дополнительных вычислений в выделенном диапазоне параметров (например, задание объема по которому производится интегрирование).

Разумно рассматривать SDF–файлы как просто хранилища данных с некоторой описательной частью, а интерпретацию данных, в частности новые типы переменных, реализовывать на более верхнем уровне в обрабатывающей программе.

На начальных стадиях разработки не было необходимости уничтожать или вставлять данные внутри SDF–файла: вычислительные пакеты порождали выходные файлы, а программа визуализации и обработки данных их зачитывала. Однако расширение применений SDF–файлов, в частности, для создания баз данных, привело к необходимости реализации функций вставки/уничтожения. Для больших файлов эти операции занимают достаточно много времени, поэтому необходимо дальнейшее развитие структуры SDF–файла и усложнение алгоритмов для оптимизации этих функций. В перспективе возможно введение в SDF–файл средств

иерархической организации переменных. В настоящей работе этот вопрос не рассматривается.

3 Программные средства работы с SDF–файлами

Программные средства для работы с SDF–файлами включают библиотеки процедур и ряд утилит.

3.1 Библиотеки процедур SDF_FILE.C и SDF_FILE.FOR

Библиотеки SDF_FILE.C и SDF_FILE.FOR содержат набор процедур, позволяющих из пользовательских программ на языках программирования C и FORTRAN создавать, открывать и закрывать SDF–файл, заполнять его данными и их описаниями, зачитывать из SDF–файла данные и их описания по указанным номеру записи и номеру поля, а также по короткому имени поля.

3.1.1 Основные процедуры библиотеки SDF_FILE.C

Типы данных:

VOID_	0
CHAR_	1
BYTE_	2
INT_	3
FLOAT_	4
WORD_	5
LONGINT_	6
DOUBLE_	7

При вызовах процедур поля и записи нумеруются с единицы.

В параметре dim значение 0 означает, что переменная — константа, 1 — одномерный массив, 2 — двумерный массив, 3 — трехмерный массив.

int create_SDFfile(SDFFILE *mf, char *fname, int key, char *comment, int l_COMMENT, int l_SHORTNAME, int l_FULLNAME);
Создание SDF–файла и заполнение фиксированной части заголовка файла.

int putfield_SDFfile(SDFFILE *mf,int *key, char *ident, char *comment, int dtype, int dim, int nx, int ny, int nz);
Заполнение или замена описания поля.

int insfield_SDFfile(SDFFILE *mf, int nfield, int key, char *ident, char *fullname, int dtype, int dim, int nx, int ny, int nz);
Вставить новое поле.

int delfield_SDFfile(SDFFILE *mf, int nfield);
Уничтожить поле.

int write_SDFfile(SDFFILE *mf, int nrec, void *data[ ]);
Запись данных в SDF–файл.

int writefield_SDFfile(SDFFILE *mf, int nrec, int nfield, void *field);
Запись поля по указанным номерам записи и поля.

int writeident_SDFfile(SDFFILE *mf, int nrec, char *ident, int *nfield, void *field);
Запись поля по короткому имени поля и указанному номеру записи.

int open_SDFfile(SDFFILE *mf, char *fname, int *key, char **comment, int *nrecs, int *nfields, int *l_COMMENT, int *l_SHORTNAME, int *l_FULLNAME);
Открытие SDF–файла и чтение фиксированной части заголовка.

int getfield_SDFfile(SDFFILE *mf, int nfield, int *key, char **ident, char **fullname, int *dtype, int *dim, int *nx, int *ny, int *nz);
Чтение описания поля(переменной) по номеру.

int getident_SDFfile(SDFFILE *mf, char *ident, int *nfield, int *key, char **fullname, int *dtype, int *dim, int *nx, int *ny, int *nz);
Чтение описания поля(переменной) по короткому имени.

int read_SDFfile(SDFFILE *mf, int nrec, void *data[ ]);
Чтение записи по указанному номеру записи из SDF–файла в оперативную память.

int readfield_SDFfile(SDFFILE *mf, int nrec, int nfield, void **field);
Чтение поля по указанным номерам записи и поля. При чтении переменной оперативная память выделяется внутри процедуры.

int readident_SDFfile(SDFFILE *mf, int nrec, char *ident, int *nfield, void **field);
Чтение поля по короткому имени поля и указанному номеру записи. При чтении переменной оперативная память выделяется внутри процедуры.

int close_SDFfile(SDFFILE *mf);
Закрытие SDF–файла.

Здесь:

mf	—	адрес управляющей структуры SDF–файла;
fname	—	имя SDF–файла;
key	—	ключ для файла или переменной;
comment	—	комментарий для файла;
l_COMMENT	—	максимальная длина комментария файла;
l_SHORTNAME	—	максимальная длина короткого имени поля;
l_FULLNAME	—	максимальная длина полного имени поля;
nrec	—	номер записи;
nfield	—	номер поля;
nrecs	—	количество записей;
nfields	—	количество полей в записи;
ident	—	короткое имя для переменной;
fullname	—	полное имя переменной;
dtype	—	тип данных для переменной;
dim	—	размерность массива данных (0, 1, 2, 3);
nx	—	размер по x;
ny	—	размер по y;
nz	—	размер по z;
data	—	массив из адресов записываемых в SDF–файл или считываемых из SDF–файла переменных;
field	—	адрес переменной.

Пример использования процедур библиотеки SDF_FILE.C приводится в Приложении 1.

3.1.2 Основные процедуры библиотеки SDF_FILE.FOR

SUBROUTINE OPENSDFW(FILENAME,IKEY,COMMENT,ICOMMENT,ISHORT,IFULL)
Открытие файла для последовательной записи данных

SUBROUTINE ENDSDFW()
Закрытие файла после последовательной записи данных

SUBROUTINE INSSDF(INFIELD,IKEY,IDENT,FULLNAME,IDTYPE,IDIM,INX, INY,INZ)
Запись в файл описания поля данных по заданному номеру поля

SUBROUTINE WRFSDF(INREC,INFIELD,VFIELD)
Запись в файл поля данных по заданным номеру записи и номеру поля

SUBROUTINE OPENSDFR(FILENAME,IKEY,COMMENT,ICOMMENT,ISHORT,IFULL, INRECS,INFIELDS)
Открытие файла для чтения данных или их перезаписи.При этом зачитывается фиксированная часть заголовка

SUBROUTINE RFSDFPAR(IDENT,INFIELD,IKEY,IDTYPE,IDIM,INY,INX,INZ)
Чтение описания поля данных по его имени

SUBROUTINE RFSDF(IDENT,INREC,VFIELD)
Чтение поля данных в указанной записи по его имени

SUBROUTINE REWRFSDF(IDENT,INREC,VFIELD)
Замена поля данных в указанной записи по его имени

Здесь:

FILENAME	—	имя открываемого SDF–файла;
IKEY	—	ключ для файла или переменной;
COMMENT	—	комментарий для файла;
ICOMMENT	—	максимальная длина комментария файла;
ISHORT	—	максимальная длина короткого имени поля;
IFULL	—	максимальная длина полного имени поля;
INREC	—	номер записи;
INFIELD	—	номер поля;
INRECS	—	количество записей;
INFIELDS	—	количество полей;
IDENT	—	короткое имя для переменной;
FULLNAME	—	полное имя переменной;
IDTYPE	—	тип данных для переменной;
IDIM	—	размерность массива данных (0, 1, 2, 3);
INX	—	размер по x;
INY	—	размер по y;
INZ	—	размер по z;
VFIELD	—	идентификатор переменной.

Примеры использования процедур библиотеки SDF_FILE.FOR приводятся в Приложении 2 и Приложении 3.

3.2 Утилиты обработки SDF–файлов

Рассмотренные ниже утилиты реализованы на языке C и позволяют пользователю получить информацию о содержимом SDF–файла, просмотреть его, преобразовать в текстовый файл и обратно.

SDFINFO	—	выдача информации о содержимом SDF–файла,
SDFVIEW	—	просмотр содержимого SDF–файла;
TXTSDF	—	преобразование SDF–файла из текстового вида в двоичный;
SDFTXT	—	преобразование SDF–файла из текстового вида в двоичный;
DATSDF	—	преобразование произвольного текстового файла данных с регулярной структурой в SDF–формат

Утилита SDFINFO выдает содержимое заголовка SDF–файла и описания переменных.

Утилита SDFVIEW служит для просмотра содержимого SDF–файла в в некоторых стандартных форматах. Общение с пользователем происходит в диалоговом режиме с использованием "мыши"

Утилиты TXTSDF и SDFTXT осуществляют преобразования между двоичным и текстовым представлением SDF–файла. Текстовый вид может быть полезен при передаче SDF–файлов на другие типы вычислительных машин, предотвращая проблемы с разнообразием двоичных представлений целых и вещественных чисел. Текстовый вид также позволяет человеку рассматривать данные в любом текстовом редакторе.

До сих пор одним из наиболее популярных языков программирования остается FORTRAN С целью использования данных, полученных из программ на этом языке, была создана утилита DATSDF, преобразующая текстовый файл данных с результатами вычислений в SDF–формат.

Описанные средства входят в программный комплекс для решения систем дифференциальных уравнений в частных производных, созданный в Лаборатории математического моделирования и дают основу для стандартизации информационных потоков. Так, интерактивный пакет обработки и визуализации выходных данных ПОСТПРОЦЕССОР работает с SDF–файлами, создаваемыми из любых вычислительных пакетов комплекса.

В интерактивных пакетах пользователь задает шаг сброса данных в SDF–файл. Счет можно прерывать на любой момент времени, ввести другое значение шага, формируя таким образом нужную последовательность записей в SDF–файле.

В программах обработки данных переменные из SDF–файла образуют список в оперативной памяти, который пользователь может видеть и выбирать ту или иную переменную для визуализации или обработки. Это дает основу для повышения уровня взаимодействия "человек–ЭВМ"

3.2.1 Утилита SDFINFO

Утилита SDFINFO выдает на экран дисплея информацию из заголовка SDF–файла и описания содержащихся в нем переменных. Вывод может быть перенаправлен в файл на диск стандартными средствами операционной системы.

Синтаксис:
SDFINFO file.sdf
file.sdf — имя SDF–файла.

На рис.3 приведен пример выдаваемой информации.

SDFINFO v1.0 12.06.92 Copyright(c) 1992
FILE: TEPLO.SDF created 12.04.92
Файл результатов расчетов программы TEPLO.EXE
KEY=0 L_SHORTNAME=8 L_FULLNAME=40 NFIELDS=12 NRECS=10

NAME	FULL NAME	KEY	TYPE	DIM	NX	NY	NZ
N	размер сетки по X	0	INT_	0	1	1	1
M	размер сетки по Y	0	INT_	0	1	1	1
L	индекс симметрии	0	INT_	0	1	1	1
time	текущее время	0	FLOAT_	0	1	1	1
dt	шаг по времени	0	FLOAT_	0	1	1	1
X	координаты X сетки	1	FLOAT_	2	51	46	1
Y	координаты Y сетки	2	FLOAT_	2	51	46	1
Te	температура	3	FLOAT_	2	51	46	1
Flux	поток	3	FLOAT_	2	51	46	1
Cv	теплоемкость	3	FLOAT_	2	51	46	1
Cap	теплопроводность	3	FLOAT_	2	51	46	1
SQ	источник	3	FLOAT_	2	51	46	1

Рис. 3. Пример работы утилиты SDFINFO

3.2.2 Утилита SDFVIEW

Утилита SDFVIEW служит для просмотра содержимого SDF–файла в в некоторых стандартных форматах.

Синтаксис:
SDFVIEW file.sdf
file.sdf — имя SDF–файла.

Работа с утилитой происходит в интерактивном режиме с использованием устройства Microsoft Mouse (рис.4).

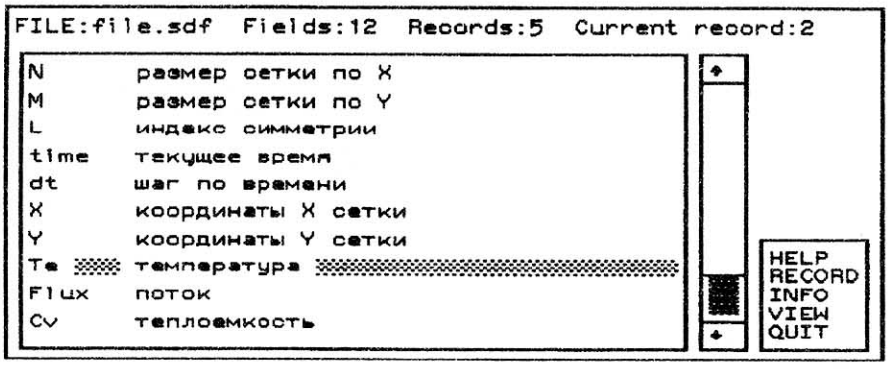

Рис. 4. Вид экрана дисплея при работе с утилитой SDFVIEW

На экране дисплея появляется список переменных, который пользователь может прокручивать на экране и выбирать ту или иную переменную для просмотра.

Представление переменной на экране зависит от ее типа.

Если переменная — целое или вещественное число, то оно высвечивается на экране. Аналогично с текстовой строкой.

Если переменная — вещественный массив, то для его визуализации используется таблица. Элементы массива высвечиваются в окне; массив прокручивается вверх–вниз при движении курсором вверх из первой строки окна и при движении курсором вниз из последней строки. Если массив — двух– или трех– мерный, то он представляется как одномерный. Справа от чисел высвечивается графическое представление массива (по оси абсцисс откладываются номера индексов массива)

Команды, выбираемые пользователем из меню:

HELP	—	выдача информации об утилите SDFVIEW;
RECORD	—	установка номера текущей записи;
INFO	—	высветить описательную часть переменной (аттрибуты);
VIEW	—	высветить значение переменной;
QUIT	—	конец работы.

3.2.3 Утилита TXTSDF

Утилита TXTSDF преобразует данные из специального текстового представления, названного авторами SDT–файлами в двоичные SDF–файлы.

Синтаксис:
TXTSDF file.sdt file.sdf
file.sdt — имя текстового SDT–файла,
file.sdf — имя двоичного SDF–файла.

Формат текстового файла позволяет достаточно удобно ввести различные данные и, преобразовав файл в SDF–файл, использовать их для расчетов.Основные принципы этого формата рассмотрим на конкретном примере.

```
TIME1
InitTim 13980.000000
StopTim 20700.099609
TimeSte 5.000000
I1 IntLvl 3 123 67 1000
R1 RealLvl 4 123.000000 67.889999 1000.099976 1000.200012
I2 LEVblck 3 2
1 2
3 4
5 6

R2 DATtime 3 2
0.000000 10.000000
100.000000 20.000000
0.000000 0.000000

R2 SDFtime 3 2
0.000000 10.000000
100.000000 20.000000
1000.000000 50.000000

SCRpage 1
R2 SCRtime 3 2
0.000000 10.000000
100.000000 20.000000
1000.000000 50.000000

END
```

Текстовый файл состоит из блоков данных, соответствующих записям SDF–файла. Количество таких блоков в файле неограничено. Каждый блок данных содержит имя переменной, ее характеристики (тип, размерность) и значение.

В приведенном выше примере файл состоит только из одного блока данных, содержащего 10 переменных.

TIME1	—	имя блока данных;
InitTim, StopTim, TimeSte, IntLvl, RealLvl, LEVblck, DATtime, SDFtime, SCRpage, SCRtime	—	имена переменных;
END	—	признак конца блока.

Допустимы следующие описатели типов данных:

- I0, R0, C0 — переменная является целой, вещественной или строковой константой соответственно. Если спецификатор типа отсутствует, то тип переменной определяется по записи значения: если есть хоть одна буква, то тип строковый, в противном случае, если есть десятичная точка, то тип вещественный, иначе — целый.

- I1, R1, C1 — переменная является одномерным целым, вещественным или строковым массивом соответственно. В этом случае после имени переменной обязательно указывается его длина.

- I2, R2 — переменная является двумерным целым или вещественным массивом соответственно. За именем переменной обязательно указываются его размерности.

После имени переменной или, если это необходимо, после описания размерности, следуют значения переменной.

Следует отметить, что предложенный формат текстового файла является базовым и нуждается в дальнейшем развитии.

3.2.4 Утилита SDFTXT

Утилита SDFTXT преобразует двоичный SDF–файл в текстовый SDT–файл.

Синтаксис:
SDFTXT file.sdf file.txt
file.sdf — имя двоичного SDF–файла;
file.sdt — имя текстового SDT–файла.

Текстовый файл имеет описанный выше формат и достаточно удобен для просмотра из любого текстового редактора.

3.2.5 Утилита DATSDF

Утилита DATSDF преобразует текстовые файлы данных, получаемые в результате работы любых расчетных программ в SDF–формат.

Использование для вычислений разных типов компьютеров и разных языков программирования порождает разнообразие в двоичном представлении чисел. Текстовый формат файлов данных в значительной степени снимает эту проблему Для представления текста, как правило, используется ASCII–кодировка. Могут возникать трудности с количеством значащих цифр в текстовом представлении числа (точность). Текстовый формат увеличивает длину файла в 2–3 раза по сравнению с двоичным представлением.

Основной целью утилиты DATSDF является подготовка файлов для работы с пакетом ПОСТПРОЦЕССОР, осуществляющим визуализацию и обработку выходных данных расчетных программ.

Утилита DATSDF может применяться для преобразования данных типа DAT из пакетов GRAPHER или SURFER.

Перед преобразованием текстового файла данных в SDF–формат необходимо создать файл с описанием структуры данных. Для этого можно воспользоваться любым редактором текстов.

Синтаксис:
DATSDF file.def file.dat file.sdf
file.def — входной текстовый файл с описанием структуры файла данных,
file.dat — входной текстовый файл данных;
file.sdf — выходной SDF–файл.

Формат файла описания структуры FILE.DEF:

- Первая строка — комментарий для def–файла. Эта строка не записывается в выводной SDF–файл.

- Вторая строка — комментарий для выводного SDF–файла.

- Третья строка — 3 целых числа и необязательный символ "|" :

первое число	—	ключ для файла (например, номер пакета, создавшего данные или тип компьютера);
второе число	—	максимальная длина короткого имени поля (идентификатора);
третье число	—	максимальная длина полного имени поля;
символ "|"	—	признак размещения данных по столбцам, если символ за третьей цифрой другой или отсутствует, то данные размещаются последовательно друг за другом.

- Четвертая и следующие строки — содержат описания полей в следующем формате:

короткое имя	—	последовательность символов;
дескриптор	—	целое число с характеристой поля (например, =0 если значения относятся к узлам расчетной сетки и =1 если к ячейкам);
тип поля	—	CHAR–символьный, INT–целый, FLOAT–вещественный;
размерность	—	0–константа, 1–одномерный массив, 2–двумерный, 3–трехмерный;
размер	—	три целых числа через символ "*" : nx*ny*nz, если размерность меньше 3, то по отсутствующим измерениям нужно ставить число 1,
полное имя	—	все оставшиеся символы до конца строки.

- Элементы описания разделяются пробелами.
- Если короткое имя — символ "–", то это поле не записывается в выходной SDF–файл.

Формат файла данных FILE.DAT:

- Текстовый файл данных содержит целые числа, вещественные числа и символьные идентификаторы, разделенные пробелами, табуляцией или переносом строки.
- Данные могут располагаться последовательно, т.е. вначале идут элементы первого поля, затем второго, третьего и т.д.
- Данные могут располагаться вертикально: первая строка содержит первые элементы полей, вторая — вторые и т.д. При этом расположении все поля должны содержать одинаковое количество элементов.
- Количество элементов на одной строке и количество разделителей может быть любым.
- При считывании всех данных, описанных в def–файле производится вывод записи в выходной SDF–файл, после этого из входного файла начинает считываться новая запись.
- Если достигнут конец входного dat–файла, а формирование текущей записи неокончено и она уже содержит по крайней один элемент, то выдается сообщение об ошибке.

Пример 1 Файл RASPLAV.DEF:

```
Файл RASPLAV.DEF — описание структуры файла RASPLAV.DAT
База данных по веществам для пакета "РАСПЛАВ"
1 8 40
 Sreda_C    0   CHAR    1   9*1*1    Тип вещества
 Sreda      0   CHAR    1   9*1*1    Вещество
 Srtuct     1   CHAR    1   51*1*1   Состав (компоненты)
 Struct_P   1   CHAR    1   51*1*1   Состав (весовые проценты)
 Ro         2   FLOAT   1   1*1*1    Плотность
 C          2   FLOAT   1   1*1*1    Теплоемкость
 Cappa      2   FLOAT   1   1*1*1    Теплопроводность
 Tsol       3   FLOAT   1   1*1*1    Температура "солидус"
 Tliq       3   FLOAT   1   1*1*1    Температура "ликвидус"
```

Пример 2. Файл RASPLAV.DAT:

БЕТОН бетон 1
Fe_2O_3+TiO_2+K_2O+Na_2O+CaO+MgO+SiO_2+Al_2O_3+CO_2+H_2O+SO_2
6.3+1.1+5.4+1.8+8.8+6.2+55.2+8.3+2.5+4.2+0.2
1.0 2.0 3.0 4.0 5.0 6.0
БЕТОН бетон 2
Fe_2O_3+TiO_2+K_2O+Na_2O+CaO+MgO+SiO_2+Al_2O_3+CO_2+H_2O+SO_2
6.3+1.1+5.4+1.8+8.8+6.2+55.2+8.3+2.5+4.2+0.2
1.01 2.01 3.01 4.02 5.02 6.02
МЕТАЛЛ сталь
Fe+Cr+Ni+Si+Cu+Mo+Mn+C+P+S
71.8+18.5+8.25+0.5+0.25+0.25+1.0+0.004+0.002+0.001
21.0 22.2 23.3 24.4 25.5 26.6

Пример 3. Файл FLUX.DEF·

Файл FLUX.DEF — описание структуры файла FLUX.DAT
Data from "GRAFER"
1 6 20 |

X	0	FLOAT	2	26*26*1	X–координата
Y	0	FLOAT	2	26*26*1	Y–координата
F	1	FLOAT	2	26*26*1	Поток
–	3	FLOAT	2	26*26*1	Поле 4
–	3	FLOAT	2	26*26*1	Поле 5

4 Работа с переменными в оперативной памяти

Использование SDF–файлов в новых программных проектах предполагает наличие дополнительных средств для работы с переменными в оперативной памяти.

Работа с переменными в оперативной памяти поддерживается динамическими списками. Для работы со списками разработана библиотека функций LIST Ограниченные размеры оперативной памяти и большие размеры массивов данных приводят к необходимости разработки стратегий по ее оптимальному использованию. Программная подсистема "ДИСПЕТЧЕР ПАМЯТИ" организует при необходимости сброс переменных во "внешнюю память": диски, extended memory, expanded memory.

Преобразование переменных в оперативной памяти поддерживается программными средствами по обработке формульных выражений. Для этой цели служит библиотека функций VFORMULA.

Разработанные программные средства легли в основу пакета ПОСТПРОЦЕССОР, осуществляющего визуализацию и обработку данных вычислительного эксперимента.

На основе этих же средств разрабатываются базы данных, ориентированные на научные и инженерные применения.

В вычислительных пакетах часто требуется уметь сохранять и восстанавливать текущее состояние задачи, а также сбрасывать некоторое подмножество переменных задачи для обработки.

Для этих целей разработан модуль OUTSDF.C, включающий следующие функции:

int WriteVars(char *fname,int nrec);
запись текущего состояния задачи в запись nrec SDF–файла с именем fname;

int ReadVars(char *fname,int nrec);
чтение текущего состояния задачи из записи nrec SDF–файла с именем fname;

int CreateOUTfile(char *fname);
создать выходной SDF–файл с именем fname;

int WriteOUTfile(void);
сбросить выходные данные на текущий момент времени в SDF–файл.

int CloseOUTfile(void);
закрыть SDF–файл.

Все перечисленные функции возвращают 0 в случае успешной работы или код ошибки, отличный от нуля.

Полный набор переменных определяется в массиве vars: static PROBLEMvar vars[NVAR];

Константа NVAR задает общее количество переменных задачи, а константа NOUTVARS задает максимальное количество выходных переменных из первого множества.

Структура PROBLEMvar определяется следующим образом:

```
typedef struct {
        char shortname[L_SHORTNAME+1];   —  короткое имя переменной
        char fullname[L_FULLNAME+1];     —  полное имя переменной
        int key;                         —  дескриптор
        int dtype;                       —  тип данных
        int dim;                         —  размерность (0, 1, 2 или 3)
        int nx;                          —  размер по X
        int ny;                          —  размер по Y
        int nz;                          —  размер по Z
        int out;                         —  =1, если переменная–выходная
        int ptr_flag;                    —  =1 — адрес, =2 — адрес адреса
        void *ptr;                       —  указатель (см. ptr_flag)
    }   PROBLEMvar;
```

Можно ввести в пакет средства для выбора пользователем подмножества сбрасываемых в выходной файл переменных (из общего набора имеющихся в пакете).

Заключение

Рассмотренный формат хранения данных (SDF–файлы) позволяет эффективно осуществлять взаимодействие вычислительных модулей с входными/выходными потоками данных. Разработанный набор сервисных программ обеспечивает формирование таких файлов из произвольной расчетной программы, написанной на языках программирования FORTRAN и C. Предложенная структура SDF–файла дает возможность строить универсальные программные средства по обработке и визуализации результатов численного моделирования.

Дальнейшее развитие идеи SDF–файла включает в себя:

- введение возможностей по созданию иерархии переменных;
- введение более эффективных процедур вставки и удаления записей и переменных из SDF–файла;
- усовершенствование формата текстового файла;
- развитие возможностей утилиты DATSDF по работе с более сложными форматами файлов данных;
- перенесение созданных средств на другие типы вычислительной техники, в частности, на рабочие станции.

Литература

[1] Cole R.K., Kelly D.P., Ellis M.A. CORCON–MOD2: A Computer Program for Analysis of Molten–core Concrete Interactions. NUREG/CR–3920, SAND84–1246, 1984.

[2] M.Reinmann, S.Stiefel. The WECHSL-Mod2 Code: A Computer Program for the Interaction of a Core Melt with Concrete including the Long Term Behaviour. KfK 4477, 1989.

[3] И.Флорес. Структуры и управление данными. М:"Финансы и статистика", 1983.

Приложение 1

Пример использования библиотеки SDF_FILE.C

Сброс данных в SDF-файл из программы на языке C.

```
#include <stdio.h>
#include <math.h>
#include "sdf_file.h"

float mas1[15], mas2[15], mas3[15];
int   bmas[30][30];
float constanta;

main()
{
SDFFILE in;
int i,j,l_COMMENT=80,l_SHORTNAME=8,l_FULLNAME=40;
void *data[5];

data[0] = mas1;   data[1] = mas2;   data[2] = mas3;
data[3] = bmas;   data[4] = &constanta;

for (i=0;i<15;i++) mas1[i] = sin(i/20.0);
for (i=0;i<15;i++) mas2[i] = cos(i/20.0);
for (i=0;i<15;i++) mas3[i] = cos(i/50.0);
for(i=0;i<30;i++)
   for(j=0;j<30;j++)
     bmas[i][j] = (i*j)+111;
constanta=sin(1/100);

if((i=create_SDFfile(&in,"test.dat",2,"file 1",
            l_COMMENT,l_SHORTNAME,l_FULLNAME)) != 0)
    { printf(" Error %d!\n",i); getch(); exit(0); }

insfield_SDFfile(&in, 1, 1,"Te", "temper",FLOAT_,1,15,1,1);
insfield_SDFfile(&in, 2, 1,"SQ", "flux", FLOAT_,1,15,1,1);
insfield_SDFfile(&in, 3, 2,"K", "koefficient",FLOAT_,1,15,1,1);
insfield_SDFfile(&in, 4, 2,"MAS","array",INT_,2,30,30,1);
insfield_SDFfile(&in, 5, 3,"CON","const",FLOAT_,1,1,1,1);

if((i=write_SDFfile(&in, 1, data)) != 0)
       { printf(" Error %d!\n",i); getch(); exit(0); }
if((i=write_SDFfile(&in, 2, data)) != 0)
       { printf(" Error %d!\n",i); getch(); exit(0); }

close_SDFfile(&in);
}
```

Приложение 2

Пример использования библиотеки SDF_FILE.FOR

Сброс данных в SDF-файл из программы на языке FORTRAN

```
C************************************************
C   Example file
C   Updated:   26.04.93
C   Copyright(c) 1993 by  Valery Pervichko
C   USE NDP/MS FORTRAN SUBROUTINES for SDF FORMAT
C************************************************
        PROGRAM WRITE
        REAL*4    R4(5)/1.,2.,3.,4.,5./
        REAL*8    R8(6)/6.,7.,8.,9.,10.,11./
        INTEGER*2 I2(5)/11,12,13,14,15/
        INTEGER*4 I4(6)/16,17,18,19,20,21/
        INTEGER*2 IV/22/
        REAL*4    RV/23./
        INTEGER*2 I22(4,3)/24,25,26,27,28,29,30,31,32,33,34,35/
        REAL*4    R42(3,4)/24.1,25.1,26.1,27.1,28.1,29.1,
     *                     30.1,31.1,32.1,33.1,34.1, 35.1/

        CALL OPENSDFW('cooc.sdf',1,'example 01',80,8,40)

        CALL INSSDF(1,1, 'varR42','var1',4,2,3,4,1)
        CALL INSSDF(2,1, 'varR4','var2',4,1,5,1,1)
        CALL INSSDF(3,1, 'varR8','var3',7,1,6,1,1)
        CALL INSSDF(4,1, 'varI22','var4',3,2,4,3,1)
        CALL INSSDF(5,1, 'varI2V','var5',3,1,1,1,1)
        CALL INSSDF(6,1, 'varI2','var6',3,1,5,1,1)
        CALL INSSDF(7,1, 'varI4','var7',6,1,6,1,1)
        CALL INSSDF(8,1, 'varR4V','var8',4,1,1,1,1)

        CALL WRFSDF(1,1, R42)
        CALL WRFSDF(1,2, R4)
        CALL WRFSDF(1,3, R8)
        CALL WRFSDF(1,4, I22)
        CALL WRFSDF(1,5, IV)
        CALL WRFSDF(1,6, I2)
        CALL WRFSDF(1,7, I4)
        CALL WRFSDF(1,8, RV)

        do 1 i=1,5
        R4(i)=R4(i)*2
        I2(i)=I2(i)*2
1       continue
        do 2 i=1,6
        R8(i)=R8(i)*2
        I4(i)=I4(i)*2
2       continue
        do 3 i=1,3
        do 3 j=1,4
        R42(i,j)=R42(i,j)*2
        I22(j,i)=I22(j,i)*2
3       continue
        IV=IV*2
```

```
      RV=RV*2

      CALL WRFSDF(2,1, R42)
      CALL WRFSDF(2,2, R4)
      CALL WRFSDF(2,3, R8)
      CALL WRFSDF(2,4, I22)
      CALL WRFSDF(2,5, IV)
      CALL WRFSDF(2,6, I2)
      CALL WRFSDF(2,7, I4)
      CALL WRFSDF(2,8, RV)

      CALL ENDSDFW()
      STOP
      END
```

Приложение 3

Пример использования библиотеки SDF_FILE.FOR

Чтение и замена данных в SDF-файле из программы на языке FORTRAN

```
C***********************************************
C   Example file
C   Updated:   26.04.93
C   Copyright(c) 1993 by  Valery Pervichko
C   USE NDP/MS FORTRAN SUBROUTINES for SDF FORMAT
C***********************************************
      PROGRAM READ
      CHARACTER COMMENT*11
      REAL*4    R42(3,4)
      REAL*8    R8(6)
      INTEGER*4 I4(6)
      INTEGER*2 IV
      REAL*4    RV
      INTEGER*2 I22(4,3)
      INTEGER*4 NFIELD,KEY,DTYPE,DIM,NX,NY,NZ

      CALL OPENSDFR('cooc.sdf',IKEY,COMMENT,ICOMMENT,
     *  ISHORT,IFULL,INRECS,INFIELDS)

      open(unit=11,file='tst.tst',status='new')
      write(11,*) IKEY,COMMENT,ICOMMENT,ISHORT,IFULL,
     *  INRECS,INFIELDS

      CALL RFSDFPAR('varR42',NFIELD,KEY,DTYPE,DIM,NY,NX,NZ)
      WRITE(11,*) NFIELD,KEY,DTYPE,DIM,NY,NX,NZ
      CALL RFSDFVAL(1,R42)
      do 101 i=1,3
101   WRITE(11,*) (R42(i,j),j=1,4)

      CALL RFSDF('varR42',2,R42)
      do 102 i=1,3
102   WRITE(11,*) (R42(i,j),j=1,4)
      CALL RFSDF('varR8',2,R8)
      WRITE(11,*) R8
      CALL RFSDF('varI22',2,I22)
```

```
      do 103 i=1,4
103   WRITE(11,*) (I22(i,j),j=1,3)
      CALL RFSDF('varR8',1,R8)
      WRITE(11,*) R8
      CALL RFSDF('varI22',1,I22)
      do 104 i=1,4
104   WRITE(11,*) (I22(i,j),j=1,3)
      CALL RFSDF('varR4V',2,RV)
      WRITE(11,*) RV
      CALL RFSDF('varR4V',1,RV)
      WRITE(11,*) RV
      CALL RFSDF('var2V',3,IV)
      WRITE(11,*) IV
      CALL RFSDF('varI4',2,I4)
      WRITE(11,*) I4
      CALL RFSDF('varI4',1,I4)
      WRITE(11,*) I4
      CALL RFSDF('varI2V',2,IV)
      WRITE(11,*) IV
      CALL RFSDF('varI2V',1,IV)
      WRITE(11,*) IV

      do 111 i=1,3
      do 111 j=1,4
      R42(i,j)=R42(i,j)*3
111   CONTINUE

      CALL REWRFSDF('varR42',1,R42)
      CALL RFSDF('varR42',1,R42)

      do 112 i=1,3
112   WRITE(11,*) (R42(i,j),j=1,4)
      CALL ENDSDFR()
      STOP
      END
```