|  |  |
| --- | --- |
| УТВЕРЖДЕН  07623974.37015-01 91 01-ЛУ |  |
|  |  |

Программный комплекс для работы

со структурированными файлами

SIO

**Руководство пользователя**

**07623974.37015-01 91 01**

Листов 35

2022

АННОТАЦИЯ

Документ является руководством пользователя по программному комплексу для работы со структурированными файлами «SIO» (ПК «SIO»). В документе рассматривается организация хранения объектов в структурированных файлах формата SIO, библиотека доступа к объектам sio и графическая утилита sioview для просмотра содержимого структурированных файлов формата SIO.

**СОДЕРЖАНИЕ**

[1. Назначение и область применения ПК «SIO» 5](#_Toc107921053)

[2. Описание формата «Sio» 6](#_Toc107921054)

[2.1. Формат хранения данных 6](#_Toc107921055)

[2.2. Типы и формат хранимых объектов 6](#_Toc107921056)

[2.3. Хранение метаинформации (каталог объектов) 6](#_Toc107921057)

[2.4. Заголовок файла 8](#_Toc107921058)

[2.5. Сегмент каталога 8](#_Toc107921059)

[2.6. Строки каталога 8](#_Toc107921060)

[3. Библиотека доступа к структурированным файлам «Sio» 9](#_Toc107921061)

[3.1. Функции для работы с файлом 9](#_Toc107921062)

[3.1.1. Открытие файла SIOFOP 9](#_Toc107921063)

[3.1.2. Закрытие файла SIOFCL 10](#_Toc107921064)

[3.2. Функции для работы с каталогом 10](#_Toc107921065)

[3.2.1. Сохранение каталога SIOFCAT 10](#_Toc107921066)

[3.2.2. Печать каталога SIOFCAT 10](#_Toc107921067)

[3.2.3. Определение длины каталога SIOLENCAT 11](#_Toc107921068)

[3.2.4. Вывод строки каталога SIOGETCAT 11](#_Toc107921069)

[3.2.5. Вывод имени объекта по номеру строки каталога SIOGETITEMNAME 11](#_Toc107921070)

[3.2.6. Вывод типа объекта по номеру строки каталога SIOGETITEMTYPE 12](#_Toc107921071)

[3.2.7. Вывод формата элемента объекта по номеру строки каталога SIOGETITEMFMT 12](#_Toc107921072)

[3.3. Функции для работы с именованными записями 12](#_Toc107921073)

[3.3.1. Запись именованной записи SIORWR, SIORIWR 12](#_Toc107921074)

[3.3.2. Чтение именованной записи SIORRD, SIORIRD 13](#_Toc107921075)

[3.3.3. Определение длины именованной записи SIORGETLEN 14](#_Toc107921076)

[3.4. Функции для работы с группами 14](#_Toc107921077)

[3.4.1. Определение группы SIOGDEF 14](#_Toc107921078)

[3.4.2. Определение поля группы SIOGDEFP 15](#_Toc107921079)

[3.4.3. Запись экземпляра группы SIOGWR 15](#_Toc107921080)

[3.4.4. Чтение экземпляра группы SIOGRD, SIOGIRD 16](#_Toc107921081)

[3.5. Функции для работы с массивами 17](#_Toc107921082)

[3.5.1. Определение массива SIOADEF 17](#_Toc107921083)

[3.5.2. Запись массива SIOAWR 17](#_Toc107921084)

[3.5.3. Чтение массива SIOARD 18](#_Toc107921085)

[3.5.4. Определение длины массива SIOAGETLEN 18](#_Toc107921086)

[3.5.5. Определение размерности массива SIOAGETDIMS 19](#_Toc107921087)

[3.5.6. Определение длины по измерению SIOAGETDIM 19](#_Toc107921088)

[3.6. Диагностика 19](#_Toc107921089)

[3.6.1. Получение сообщения об ошибке SIOGETMSG 19](#_Toc107921090)

[4. Примеры использования 21](#_Toc107921091)

[4.1. Работа с именованными записями 21](#_Toc107921092)

[4.2. Работа с группами 22](#_Toc107921093)

[4.3. Работа с массивами 24](#_Toc107921094)

[4.4. Сборка прикладной программы с использованием библиотеки sio 27](#_Toc107921095)

[5. Графическая утилита просмотра данных «Sioview» 28](#_Toc107921096)

[5.1. Вызов утилиты и открытие файла для просмотра 28](#_Toc107921097)

[5.2. Выбор объекта для просмотра 30](#_Toc107921098)

[5.3. Просмотр именованной записи 30](#_Toc107921099)

[5.4. Просмотр многомерного массива 30](#_Toc107921100)

[Перечень сокращений 34](#_Toc107921101)

[Перечень ссылочных документов 35](#_Toc107921102)

# Назначение и область применения ПК «SIO»

ПК «SIO» предназначен для хранения и доступа к структурированным данным в файлах, расположенных на файловых системах компьютеров и вычислительных систем. ПК «SIO» состоит из библиотеки sio и графической утилиты sioview.

Библиотека sio содержит набор функций, обеспечивающих создание и открытие структурированного файла, создание в нем именованных объектов, получение информации об этих объектах, запись и чтение информации из прикладных программ в объекты структурированного файла.

Графическая утилита sioview предназначена для вывода в интерактивном режиме информации об содержащихся в структурированном файле объектах и просмотра содержимого этих объектов.

Функциональные возможности:

* хранение именованных записей (одномерных векторов);
* хранение многомерных массивов;
* использование 64-битной адресной информации (позволяет обеспечить хранение информации, объемом ограниченной только возможностями используемой файловой системы и системы хранения);
* доступ к объектам из прикладных программ, написанных на языках C, C++ в операционных системах семейства Linux и Windows;
* интерактивный просмотр данных в графической системе ОС Linux (СПО Супер-ЭВМ, ЗОС «Арамид», ОС «Astra Linux», RHEL и т.д.) и ОС Windows (XP, Windows 7, Windows 10);
* просмотр содержимого многомерных массивов в табличном виде в произвольном двумерном сечении;
* независимость доступа к данным от используемой операционной системы.

Область применения – моделирующие и информационно-вычислительные комплексы, численное моделирование физических процессов.

В отличие от ближайшего аналога HDF [1], ПК «SIO» обладает более простым интерфейсом и предъявляет минимальные требования к системному окружению.

ПК «SIO» может применяться на серверах или АРМ с архитектурой x86\_64, многопроцессорных вычислительных системах (Супер-ЭВМ) с архитектурой процессора x84\_64. Требования к оперативной памяти определяются объемами хранимых объектов.

Язык программирования: C, python, графическая библиотека: Tk-8X.

# Описание формата «Sio»

* 1. Формат хранения данных

В соответствии с областью применения формат хранения данных должен удовлетворять следующим требованиям:

1. возможность хранения произвольного количества именованных данных разного типа и размера;
2. возможность использования данных без дополнительного преобразования на ЭВМ с разной аппаратной платформой и различными операционными системами (семейство ОС Windows, семейство ОС Linux: ОС «Astra Linux», ЗОС «Арамид», СПО Супер-ЭВМ, RHEL и т.д.).

Полное описание формата хранения данных приведено в документе [2].

* 1. Типы и формат хранимых объектов

Текущая версия позволяет хранить следующие типы данных:

* именованные записи – одномерные вектора, содержащие элементы заданного формата;
* пронумерованные структурированные группы данных (структуры), в которых поля содержат вектора элементов заданного формата;
* многомерные массивы, содержащие элементы заданного формата.

Каждый хранимый объект имеет имя.

Структуры, кроме этого имеют порядковый номер. Именованные записи, поля структуры и массивы имеют формат элементов.

Массивы имеют описание размерностей в виде: количество измерений и число элементов по каждому измерению.

В текущей версии поддерживаются следующие форматы, представляемые двузначным десятичным числом:

* 32-битное целое (24);
* 64-битное целое (28);
* 32-битное вещественное (14);
* 64-битное вещественное (18).
  1. Хранение метаинформации (каталог объектов)

Для обеспечения возможности хранения произвольного количества объектов каталог файла имеет сегментную структуру. При создании файла создается заголовок файла и первый сегмент каталога. Размер сегмента может указываться пользователем в параметрах процедуры создания файла. Сегмент имеет строчную структуру. Каждая строка сегмента каталога описывает один объект или его атрибуты. После первого сегмента каталога в файле размещаются объекты, содержащиеся в этом сегменте. После исчерпания первого сегмента в конец файла помещается новый сегмент (экстент каталога) такого же размера. Далее размещаются объекты этого сегмента. Эта процедура повторяется и после исчерпания экстента.

Схема размещения информации в файле представлена на рис. 1.

|  |
| --- |
|  |
| Рис. 1 – Размещение информации в файле формата SIO |

* 1. Заголовок файла

Заголовок файла размещается в первых 24 байтах файла и содержит следующую информацию:

* код версии формата SIO (в текущей версии – 2);
* тип размещения байтов в памяти (по умолчанию LITTLE ENDIAN). Текущая версия не поддерживает другие типы размещения байтов в памяти;
* размер сегмента каталога;
* число экстентов каталога;
* общее число заполненных строк в каталоге;
* дату сборки библиотеки.
  1. Сегмент каталога

Первый сегмент каталога размещается после заголовка файла. Экстенты каталога размещаются после ранее записанных объектов. Каждый экстент именуется и включается в предыдущий сегмент каталога.

Сегмент каталога содержит в себе фиксированное количество строк, равное ранее заданному размеру сегмента.

Согласно типам хранимых объектов существуют следующие типы строк каталога:

* описание именованной записи;
* описание группы;
* описание экземпляра группы;
* описание поля группы;
* описание массива;
* описание размерности массива;
* описание экстента каталога.
  1. Строки каталога

Каждая строка каталога содержит следующую информацию:

* тип строки;
* имя объекта;
* численный параметр, связанный с объектом;
* размер объекта;
* формат элементов объекта;
* адресная информация.

# Библиотека доступа к структурированным файлам «Sio»

В библиотеку sio входят функции для работы с файлами, каталогом файла и объектами. Каждая функция имеет свои параметры и код завершения. Отрицательный код завершения свидетельствует об ошибке при выполнении. При открытии файла создается дескриптор HF\* (описан в sio2.h), все операции с файлом после этого производятся с этим дескриптором, поданном в параметрах. Некоторые параметры в текущей версии не являются актуальными и заложены для дальнейшего развития ПК. Они далее обозначаются как skip и не объясняются.

* 1. Функции для работы с файлом
     1. Открытие файла SIOFOP

**Вызов**:

ko = siofop(

const char filename,

int memtype,

int status,

int lcat,

int skip,

HF\*\* hf

)

**Параметры**:

* filename – имя файла;
* memtype – тип памяти, пока не используется, задается 0;
* status – режим открытия файла;
* lcat – желаемый размер сегмента каталога, при задании 0 используется умолчание;
* hf – дескриптор файла (выходной параметр).

Файл открывается (или создается), полученный дескриптор файла используется в дальнейших операциях с этим файлом.

Допустимые режимы открытия:

* STNEW – файл создается заново, если такой файл уже существует, информация в нем затирается. Режим доступа – чтение/запись;
* STOLD – открытие существующего файла. Режим доступа – чтение/запись;
* STUNK – открытие файла. Если такого файла нет, он создается. Режим доступа – чтение/запись;
* STRO – открытие существующего файла на чтение.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Закрытие файла SIOFCL

**Вызов:**

ko = siofcl(

HF\* hf,

int skip

)

**Параметры:**

* hf – дескриптор открытого файла.

Файл закрывается, каталог из оперативной памяти сохраняется в файле.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  1. Функции для работы с каталогом
     1. Сохранение каталога SIOFCAT

**Вызов:**

ko = sioflush(

HF\* hf

)

**Параметры:**

* hf – дескриптор открытого файла.

Каталог из оперативной памяти сохраняется в файле без закрытия файла.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Печать каталога SIOFCAT

**Вызов:**

ko = siofcat(

HF\* hf,

char\* skip

)

**Параметры:**

* hf – дескриптор открытого файла.

Каталог файла выводится в табличном виде на устройство stdout.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Определение длины каталога SIOLENCAT

**Вызов:**

ko = siolencat(

HF\* hf

)

**Параметры:**

* hf – дескриптор открытого файла.

**Результат выполнения:**

* количество строк в каталоге.
  + 1. Вывод строки каталога SIOGETCAT

**Вызов:**

ko = siogetcat(

HF\* hf,

int nl,

char\*\* name,

int\* num,

int\* type,

int\* len,

int\* fmt,

size\_t addr

)

**Параметры:**

* hf – дескриптор открытого файла;
* nl – номер запрашиваемой строки каталога;
* остальные параметры возвращают значения полей строки каталога.

Выдаются значения полей запрошенной строки каталога.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Вывод имени объекта по номеру строки каталога SIOGETITEMNAME

**Вызов:**

ko = siogetitemname(

HF\* hf,

int nl,

char\* name

)

**Параметры:**

* hf – дескриптор открытого файла;
* nl – номер строки;
* name – возвращаемое имя объекта.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Вывод типа объекта по номеру строки каталога SIOGETITEMTYPE

**Вызов:**

ko = siogetitemtype(

HF\* hf,

int nl

)

**Параметры:**

* hf – дескриптор открытого файла;
* nl – номер строки.

**Результат выполнения:**

* >0 – тип объекта;
* меньше 0 – ошибка при выполнении.
  + 1. Вывод формата элемента объекта по номеру строки каталога SIOGETITEMFMT

**Вызов:**

ko = siogetitemfmt(

HF\* hf,

int nl

)

**Параметры:**

* hf – дескриптор открытого файла;
* nl – номер строки.

Действие.

**Результат выполнения:**

* > 0 – формат объекта;
* меньше 0 – ошибка при выполнении.
  1. Функции для работы с именованными записями
     1. Запись именованной записи SIORWR, SIORIWR

**Вызов:**

ko = siorwr(

HF\* hf,

const char\* name,

void\* array,

int len,

int fmt

)

ko = sioriwr(

HF\* hf,

const char\* name,

void\* array,

int ind,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя именованной записи;
* array – указатель на записываемый массив;
* ind – смещение по области памяти в файле;
* len – длина в элементах;
* fmt – формат элемента.

Информация из массива переписывается в файл (с указанного смещения) под указанным именем.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Чтение именованной записи SIORRD, SIORIRD

**Вызов:**

ko = siorrd(

HF\* hf,

const char\* name,

void\* array,

int len,

int fmt

)

ko = siorirr(

HF\* hf,

const char\* name,

void\* array,

int ind,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя именованной записи;
* array – указатель на записываемый массив;
* ind – смещение по области памяти в файле;
* len – длина в элементах;
* fmt – формат элемента.

Информация из именованной записи (с заданным смещением) с указанным именем переписывается в массив.

**Результат выполнения:**

* >= 0 – количество прочитанных элементов;
* меньше 0 – ошибка при выполнении.
  + 1. Определение длины именованной записи SIORGETLEN

**Вызов:**

ko = siorgetlen(

HF\* hf,

const char\* name

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя записи.

**Результат выполнения:**

* >= 0 – количество элементов в именованной записи;
* меньше 0 – ошибка при выполнении.
  1. Функции для работы с группами
     1. Определение группы SIOGDEF

**Вызов:**

ko = siogdef(

HF\* hf,

const char\* name,

int numpol

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя создаваемой группы;
* numpol – число полей в группе.

Создается группа с указанным числом полей. Далее должны следовать обращения к функции определения полей группы.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Определение поля группы SIOGDEFP

**Вызов:**

ko = siogdefp(

HF\* hf,

const char\* namepol,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* namepol – имя создаваемого поля;
* len – число элементов в поле группы;
* fmt – формат элемента.

Для созданной ранее группы определяется новое поле с указанными атрибутами. Количество обращений на определение поля должно совпадать с числом полей, указанных в группе.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Запись экземпляра группы SIOGWR

**Вызов:**

ko = siogwr(

HF\* hf,

const char\* namegrp,

int num,

const char\* namepol,

void\* array,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* namegrp – имя группы;
* num – номер экземпляра группы;
* namepol – имя поля группы;
* array – указатель на массив;
* len – длина массива в элементах;
* fmt – формат элемента.

В область памяти файла, выделенного для указанного экземпляра группы, со смещением, определяемым порядковым номером поля, переписывается информация из поданного массива.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Чтение экземпляра группы SIOGRD, SIOGIRD

**Вызов:**

ko = siogrd(

HF\* hf,

const char\* namegrp,

int num,

const char\* namepol,

void\* array,

int len,

int fmt

)

ko = siogird(

HF\* hf,

const char\* namegrp,

int num,

const char\* namepol,

void\* array,

int idx,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* namegrp – имя группы;
* num – номер экземпляра группы;
* namepol – имя поля группы;
* array – указатель на массив;
* idx – смещение по области памяти для поля в файле;
* len – длина массива в элементах;
* fmt – формат элемента.

Из области памяти файла, выделенного для указанного экземпляра группы, со смещением, определяемым порядковым номером поля и указанным в параметрах смещением, переписывается информация в поданный массив.

**Результат выполнения:**

* >= 0 – количество прочитанных элементов;
* меньше 0 – ошибка при выполнении.
  1. Функции для работы с массивами
     1. Определение массива SIOADEF

**Вызов:**

ko = sioadef(

HF\* hf,

const char\* name,

int ndims,

int\* dims,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя создаваемого массива;
* ndims – количество измерений;
* dims – массив с количествами элементов по измерениям;
* fmt – формат элемента.

Создается массив с указанным числом измерений и размером, соответствующим длинам по измерениям.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Запись массива SIOAWR

**Вызов:**

ko = sioawr(

HF\* hf,

const char\* name,

void\* array,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя записываемого массива (должно быть уже создано);
* array – указатель на записываемый массив;
* len – длина массива в элементах (должна соответствовать определенной при создании массива);
* fmt – формат элемента.

Информация из массива в оперативной памяти переписывается в файл под указанным именем.

**Результат выполнения:**

* 0 – успешно;
* меньше 0 – ошибка при выполнении.
  + 1. Чтение массива SIOARD

**Вызов:**

ko = sioard(

HF\* hf,

const char\* name,

void\* array,

int len,

int fmt

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя записываемого массива (должно быть уже создано);
* array – указатель на записываемый массив;
* len – длина массива в элементах (можно указать 0, в этом случае читается весь массив);
* fmt – формат элемента

Информация из массива с указанным именем из файла переписывается в массив в оперативной памяти.

**Результат выполнения:**

* >= 0 – количество прочитанных элементов;
* меньше 0 – ошибка при выполнении.
  + 1. Определение длины массива SIOAGETLEN

**Вызов:**

ko = sioagetlen(

HF\* hf,

const char\* name

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя массива.

Выдается количество элементов в указанном массиве.

**Результат выполнения:**

* >= 0 – количество элементов в массиве;
* меньше 0 – ошибка при выполнении.
  + 1. Определение размерности массива SIOAGETDIMS

**Вызов:**

ko = sioagetdims(

HF\* hf,

const char\* name

)

**Параметры:**

* hf – дескриптор открытого файла;
* const char\* name.

**Результат выполнения:**

* >= 0 – количество измерений в массиве;
* меньше 0 – ошибка при выполнении.
  + 1. Определение длины по измерению SIOAGETDIM

**Вызов:**

ko = sioagetdim(

HF\* hf,

const char\* name,

int dim

)

**Параметры:**

* hf – дескриптор открытого файла;
* name – имя массива;
* dim – номер измерения.

Выдается число элементов массива по указанному измерению.

**Результат выполнения:**

* >= 0 – число элементов по указанному измерению;
* меньше 0 – ошибка при выполнении.
  1. Диагностика
     1. Получение сообщения об ошибке SIOGETMSG

**Вызов:**

siogetmsg(

char\* p

)

**Параметры:**

* p – указатель на массив для выдачи сообщения об ошибке.

При ошибочном выполнении последней выполняемой функции с помощью функции siogetmsg можно получить текстовое сообщение, указывающее причину ошибки.

# Примеры использования

* 1. Работа с именованными записями

Приведем пример программы, которая работает с именованными записями:

#include "../sio2.h"

#include <stdio.h>

#include <stdlib.h>

void err()

{

char msg[80];

siogetmsg(msg);

puts(msg);

exit(0);

}

const int L = 1000000;

HF \*hf;

int \*mi;

long \*ml;

float \*mf;

double \*md;

char \*fname = "tdocrec.sio";

void read\_array()

{

int i, l;

if ((l = siorrd(hf,"i1",mi,L,24)) < 0) err();

if ((l = siorrd(hf,"f2",mf,L,14)) < 0) err();

for (i = 0; i < 2; i++) printf("i1,f2 %d %f\n",mi[i], mf[i]);

if ((l = siorrd(hf,"l3",ml,L,28)) < 0) err();

if ((l = siorrd(hf,"d4",md,L,18)) < 0) err();

for (i = 0; i < 2; i++) printf("l3,d4 %ld %f\n",ml[i], md[i]);

} /\* read\_array \*/

void write\_array()

{

int i, l;

for (i = 0; i < L; i++) mi[i]=1000+i;

for (i = 0; i < L; i++) mf[i]=2000.0+i;

if ((l = siorwr(hf,"i1",mi,L,24)) < 0) err();

if ((l = siorwr(hf,"f2",mf,L,14)) < 0) err();

for (i = 0; i < L; i++) ml[i]=3000+i;

for (i = 0; i < L; i++) md[i]=4000.0+i;

if ((l = siorwr(hf,"l3",ml,L,28)) < 0) err();

if ((l = siorwr(hf,"d4",md,L,18)) < 0) err();

} /\* write\_array \*/

main(int argc, char \*\*argv)

{

const int lcat = 15;

int reg, prpak = 0, ko;

mi = (int\*) malloc(L\*sizeof(int));

mf = (float\*) malloc(L\*sizeof(float));

ml = (long\*) malloc(L\*sizeof(long));

md = (double\*)malloc(L\*sizeof(double));

if (argc > 1) {

reg = atoi(argv[1]);

if (reg == 1) {

ko = siofop(fname,0,STNEW,lcat,0,&hf);

write\_array();

} else {

ko = siofop(fname,0,STUNK,lcat,0,&hf);

read\_array();

}

} else {

ko = siofop(fname,0,STRO,lcat,0,&hf);

}

if (ko < 0) err();

siofcat(hf,0,0);

siofcl(hf,prpak);

} /\* main \*/

Эта программа, в зависимости от параметра запуска, вызывает функцию write\_array для открытого в режиме записи файла с заново создаваемым содержимым, либо функцию read\_array для чтения именованных записей из существующего файла. Формируется 4 именованных записей разного типа.

Вывод этой программы в режиме чтения показывает два первых элемента каждой именованной записи и каталог формированного файла:

[s4145@localhost tests]$ ./tdocrec 2

i1,f2 1000 2000.000000

i1,f2 1001 2001.000000

l3,d4 3000 4000.000000

l3,d4 3001 4001.000000

name number type length format address

----------------------------+------+----------------+------------

i1 record 1000000 int\*4 736

f2 record 1000000 real\*4 4000736

l3 record 1000000 int\*8 8000736

d4 record 1000000 real\*8 16000736

Total : 24000736

----------------------------+------+----------------+------------

* 1. Работа с группами

Приведем пример программы, которая работает с группами:

#include "../sio2.h"

#include <stdio.h>

#include <stdlib.h>

void err()

{

char msg[80];

siogetmsg(msg);

puts(msg);

exit(0);

}

const int L = 1000000;

HF \*hf;

int \*mi;

float \*mf;

long \*ml;

double \*md;

char \*fname = "tdocgrp.sio";

void def\_field()

{

if (siogdef(hf,"gr1",2) < 0) err();

if (siogdefp(hf,"p1i",L,24) < 0) err();

if (siogdefp(hf,"p1f",L,14) < 0) err();

if (siogdef(hf,"gr2",2) < 0) err();

if (siogdefp(hf,"p2l",L,28) < 0) err();

if (siogdefp(hf,"p2d",L,18) < 0) err();

} /\* def\_field \*/

void write\_field()

{

int i, l, n;

for (n = 1; n <= 2; n++) {

for (i = 0; i < L; i++) mi[i]=1000 +i+n\*10000+100000;

for (i = 0; i < L; i++) mf[i]=2000.0+i+n\*10000+100000;

if ((l = siogwr(hf,"gr1",n,"p1i",mi,L,24)) < 0) err();

if ((l = siogwr(hf,"gr1",n,"p1f",mf,L,14)) < 0) err();

for (i = 0; i < L; i++) ml[i]=1000 +i+n\*10000+200000;

for (i = 0; i < L; i++) md[i]=2000.0+i+n\*10000+200000;

if ((l = siogwr(hf,"gr2",n,"p2l",ml,L,28)) < 0) err();

if ((l = siogwr(hf,"gr2",n,"p2d",md,L,18)) < 0) err();

}

} /\* write\_field \*/

void read\_field()

{

int i, l, n;

for (n = 1; n <= 2; n++) {

if ((l = siogrd(hf,"gr1",n,"p1i",mi,L,24)) < 0) err();

if ((l = siogrd(hf,"gr1",n,"p1f",mf,L,14)) < 0) err();

for (i = 0; i < 2; i++) printf("gr1[%d] p1i=%d p1f=%f\n", n, mi[i], mf[i]);

if ((l = siogrd(hf,"gr2",n,"p2l",ml,L,28)) < 0) err();

if ((l = siogrd(hf,"gr2",n,"p2d",md,L,18)) < 0) err();

for (i = 0; i < 2; i++) printf("gr2[%d] p2i=%ld p2f=%f\n", n, ml[i], md[i]);

}

} /\* read\_field \*/

main(int argc, char \*\*argv)

{

const int lcat = 15;

int reg, prpak = 0, ko;

mi = (int\*) malloc(L\*sizeof(int));

mf = (float\*) malloc(L\*sizeof(float));

ml = (long\*) malloc(L\*sizeof(long));

md = (double\*)malloc(L\*sizeof(double));

if (argc > 1) {

reg = atoi(argv[1]);

if (reg == 1) {

ko = siofop(fname,0,STNEW,lcat,0,&hf);

def\_field();

write\_field();

} else {

ko = siofop(fname,0,STUNK,lcat,0,&hf);

read\_field();

}

}

else ko = siofop(fname,0,STRO,lcat,0,&hf);

if (ko < 0) err();

siofcat(hf,0,0);

siofcl(hf,prpak);

} /\* main \*/

Эта программа, в зависимости от параметра запуска, вызывает функции определения групп и записей двух экземпляров этих групп для открытого в режиме записи файла с заново создаваемым содержимым или функцию чтения этих групп из существующего файла. Формируется по два экземпляра двух групп с полями разного типа.

Вывод этой программы показывает содержимое первых элементов полей каждого экземпляра групп и каталог формированного файла:

[s4145@localhost tests]$ ./tdocgrp 0

gr1[1] p1i=111000 p1f=112000.000000

gr1[1] p1i=111001 p1f=112001.000000

gr2[1] p2i=211000 p2f=212000.000000

gr2[1] p2i=211001 p2f=212001.000000

gr1[2] p1i=121000 p1f=122000.000000

gr1[2] p1i=121001 p1f=122001.000000

gr2[2] p2i=221000 p2f=222000.000000

gr2[2] p2i=221001 p2f=222001.000000

name number type length format address

----------------------------+------+----------------+------------

gr1 group 8000000 bytes 2 РїРѕР»РµР№ flds

p1i field 1000000 int\*4

p1f field 1000000 real\*4

gr2 group 16000000 bytes 2 РїРѕР»РµР№ flds

p2l field 1000000 int\*8

p2d field 1000000 real\*8

gr1 1ekz.gr. 8000000 bytes 736

gr2 1ekz.gr. 16000000 bytes 8000736

gr1 2ekz.gr. 8000000 bytes 24000736

gr2 2ekz.gr. 16000000 bytes 32000736

Total : 48000736

----------------------------+------+----------------+------------

* 1. Работа с массивами

Приведем пример программы, которая работает с массивами:

#include "../sio2.h"

#include <stdio.h>

#include <stdlib.h>

#define LM 100

#define L1 100

#define L2 50

#define L3 10

const int D3 = 3;

int dims3[3];

double d3\_1[L1][L2][L3];

double d3\_2[L1][L2][L3];

void err(int ko)

{

char msg[80];

printf("Error %d\n", ko);

siogetmsg(msg);

puts(msg);

exit(0);

} /\* err \*/

void PrintArray3(char\*name, double a[L1][L2][L3])

{

int i, j, k, l;

printf("== %s dim=%d ==\n", name, 3);

for (i = 0; i < L1; i++) {

if (i > 0 && i < L1-1) continue;

for (j = 0; j < L2; j++) {

if (j > 0 && j < L2-1) continue;

k = 0;

printf("%02d-%02d-%02d: ", i, j, k);

for (k = 0; k < L3; k++) {

if (k > 1 && k < L3-2) continue;

printf(" %9.0f", a[i][j][k]);

}

printf("\n");

}

}

}

void Prepare() {

int i, j, k, l, m;

dims3[0] = L1;

dims3[1] = L2;

dims3[2] = L3;

for (i = 0; i < L1; i++) {

for (j = 0; j < L2; j++) {

for (k = 0; k < L3; k++) {

d3\_1[i][j][k] = (double) i\*LM\*LM + j\*LM + k;

d3\_2[i][j][k] = (double) i\*LM\*LM + j\*LM + k + LM\*LM\*LM;

}

}

}

}

HF \*hf;

char \*fname = "tdocarr.sio";

void Write() {

const int lcat = 20;

int ko;

ko = siofop(fname,0,STNEW,lcat,0,&hf);

if (ko < 0) err(ko);

ko = sioadef(hf, "d3\_1", 3, dims3, 18);

if (ko < 0) err(ko);

ko = sioawr(hf, "d3\_1", d3\_1, L1\*L2\*L3, 18);

if (ko < 0) err(ko);

ko = sioadef(hf, "d3\_2", 3, dims3, 18);

if (ko < 0) err(ko);

ko = sioawr(hf, "d3\_2", d3\_2, L1\*L2\*L3, 18);

if (ko < 0) err(ko);

ko = siofcat(hf, 0, 0);

ko = siofcl(hf,0);

if (ko < 0) err(ko);

}

void PrintfDims(char\* name)

{

int dim, ndims, i, n;

char s[20], \*p=s;

p[0] = 0x00;

ndims = sioagetdims(hf, name);

for (i = 0; i < ndims; i++) {

dim = sioagetdim(hf, name, i);

n = sprintf(p,"%d ", dim);

p += n;

}

printf("== %s dim=%d sizes=%s\n", name, ndims, s);

}

void Read()

{

int ko;

int lcat = 20;

ko = siofop(fname,0,STOLD,lcat,0,&hf);

if (ko < 0) err(ko);

ko = sioard(hf, "d3\_1", d3\_1, 0, 18);

if (ko < 0) err(ko);

PrintfDims("d3\_1");

PrintArray3("d3\_1", d3\_1);

ko = sioard(hf, "d3\_2", d3\_2, 0, 18);

if (ko < 0) err(ko);

PrintfDims("d3\_2");

PrintArray3("d3\_2", d3\_2);

ko = siofcl(hf,0);

if (ko < 0) err(ko);

}

int main(int argc, char \*\*argv)

{

Prepare();

Write();

Read();

return 0;

} /\* main \*/

Эта программа создает и записывает два статических трехмерных массива, выводит каталог файла, а затем читает эти массивы и выводит крайние значения по измерениям:

[s4145@localhost tests]$ ./tdocarr

name number type length format address

----------------------------+------+----------------+------------

d3\_1 3D array 50000 real\*8 976

d3\_1 0 dim 100 real\*8 0

d3\_1 1 dim 50 real\*8 0

d3\_1 2 dim 10 real\*8 0

d3\_2 3D array 50000 real\*8 400976

d3\_2 0 dim 100 real\*8 0

d3\_2 1 dim 50 real\*8 0

d3\_2 2 dim 10 real\*8 0

Total : 800976

----------------------------+------+----------------+------------

== d3\_1 dim=3 sizes=100 50 10

== d3\_1 dim=3 ==

00-00-00: 0 1 8 9

00-49-00: 4900 4901 4908 4909

99-00-00: 990000 990001 990008 990009

99-49-00: 994900 994901 994908 994909

== d3\_2 dim=3 sizes=100 50 10

== d3\_2 dim=3 ==

00-00-00: 1000000 1000001 1000008 1000009

00-49-00: 1004900 1004901 1004908 1004909

99-00-00: 1990000 1990001 1990008 1990009

99-49-00: 1994900 1994901 1994908 1994909

* 1. Сборка прикладной программы с использованием библиотеки sio

Пусть исходные файлы библиотеки расположены в каталоге /wdsio, а прикладная программа, использующая это библиотеку – в каталоге /wdprog. Для сборки программы и библиотеки в ОС должен быть установлен компилятор с языка С.

Тогда сборка библиотеки в ОС Linux при использовании компилятора gcc может быть выполнена командами:

$ cd /wdsio

$ gcc –o sio2.c

А сборка программы с библиотекой sio:

$ cd /wdprog

$ gcc –o prog proc.c $wdsio/sio2.o

Сборка приложения в ОС Windows производится аналогично согласно руководству пользователя по используемому компилятору

# Графическая утилита просмотра данных «Sioview»

Графическая утилита sioview предназначена для интерактивного просмотра содержимого файла формата SIO и хранящихся в нем объектов в среде X11 (ОС Linux) или ОС Windows.

Для использования утилиты в ОС должны быть установлены пакеты Python3 с модулем Tkinter для Python, обеспечивающим работу с Tk.

* 1. Вызов утилиты и открытие файла для просмотра

Для использования утилиты в ОС должны быть установлены пакеты, содержащие python версии 3, графическую библиотеку Tk версии 8.Х и модель tkinter.Утилита sioview вызывается из командной строки графического терминала в ОС Linux или из командной строки ОС Windows:

$ python3 sioview2.py

В параметрах запуска может быть указано имя просматриваемого файла:

$ python3 sioview2.py tests/array.sio

Предварительно должен быть установлен путь доступа к утилите.

При вызове утилиты создается ее главное окно (см. рис. 2), содержащее краткую информацию об утилите и две кнопки: «Выход» и «Открыть».

|  |
| --- |
|  |
| Рис. 2 – Головное окно утилиты |

Кнопка «Выход» используется для завершения работы утилиты.

Кнопка «Открыть» позволяет открыть новый файл для просмотра. При этом выдается диалог выбора файла (см. рис. 3). В верхнем поле диалога можно указать шаблон имени файла для просмотра. Кнопка «Filter» активизирует этот шаблон.

|  |
| --- |
|  |
| Рис. 3 – Диалог выбора файла |

После выбора файла (или если файл указан в параметре утилиты) создается окно просмотра каталога, представленное на рис. 4. В этом окне показаны все строки каталога, описывающие хранящиеся в файле объекты. Для каждого объекта указаны его имя, тип, количество и формат элементов, адрес размещения в файле. Для массивов дополнительно указана их размерность.

|  |
| --- |
|  |
| Рис. 4 – Окно с каталогом файла |

На рис. 4 показан каталог файла, содержащего одну именованную запись «rec1», одномерный массив «d1», двумерный массив «d2» и пятимерный массив «d5».

* 1. Выбор объекта для просмотра

Для просмотра объекта нужно мышью выбрать строку, содержащую требуемый объект, и либо дважды нажать левую кнопку мыши, либо правой кнопкой открыть контекстное меню операций над объектом и выбрать операцию «Просмотр».

* 1. Просмотр именованной записи

Именованная запись, являясь одномерным вектором, для удобства просмотра показывается в виде таблицы с горизонтальной построчной разверткой по 10 элементов (см. рис.5). Диапазон выдачи задается в полях ввода «от» и «до». После изменения диапазона необходимо нажать кнопку «Перевыдать».

|  |
| --- |
|  |
| Рис. 5 – Просмотр именованной записи |

В левом столбце таблицы выдается индекс первого и последнего элемента, отображенного в строке.

* 1. Просмотр многомерного массива

Просмотр многомерных массивов любой размерности производится по одним алгоритмам, однако есть особенности выдачи массивов разных размерностей.

Одномерный массив выдается в столбце таблице (см. рис. 6). Строка таблицы имеет ширину 1. Для массива можно указать диапазон выдачи.

|  |
| --- |
|  |
| Рис. 6 – Просмотр одномерного массива |

Двумерный массив выдается в табличном виде. Одно измерение выдается по столбцам, второе по строкам (см. рис. 7). Заголовок первого столбца показывает порядок вывода измерений. Значения в этом столбце показывают индекс в массиве первого в строке элемента. Заголовки остальных столбцов показывают индекс по другому измерению. Порядок вывода по измерениям, как показано далее, может быть изменен пользователем.

|  |
| --- |
|  |
| Рис. 7 – Просмотр двумерного массива |

Массивы остальных размерностей выдаются также в табличном виде. При этом выдается двумерное сечение по произвольно выбранным измерениям (для пятимерного массива см. рис. 8).

|  |
| --- |
|  |
| Рисунок 8 – Просмотр многомерного массива |

В заголовке первого столбца, как и в случае двумерного массива, перечислены измерения массива в порядке их отображения. Смысл значений в первом столбце и заголовках остальных столбцов такой же, как и в случае двумерного массива.

В левой нижней части окна расположен фрейм управления выводом.

Пример фрейма управления в случае вывода сечения по 1 и 2 измерению (измерения нумеруются, начиная с 0) приведен на рис. 9.

|  |
| --- |
|  |
| Рис. 9 – Фрейм управления выводом |

Верхняя строка фрейма содержит имя файла. Следующая строка содержит имя выбранного объекта, формат его элементов и количество элементов в объекте.

Две следующие строки позволяют выбрать измерения для формирования сечения. Для этого используется комбобокс в левой части строки. После выбора измерения справа от комбобокса будет выдано символическое обозначение этого измерения (начиная с латинской буквы «I») и число элементов в нем. Также выдаются два поля ввода для определения диапазона вывода по выбранному измерению.

Остальные строки описывают оставшиеся измерения (при их наличии). Для каждого измерения выдается его символическое имя и количество элементов по этому измерению. Также выдается поля ввода для индекса по измерению и кнопки «-1» и «+1», позволяющие уменьшать или увеличивать на единицу текущее значение индекса.

**Перечень сокращений**

|  |  |  |
| --- | --- | --- |
| АРМ | – | Автоматизированное рабочее место |
| ЗОС | – | Защищенная операционная система |
| ОС | – | Операционная система |
| ПК | – | Программный комплекс |
| СПО | – | Системное программное обеспечение |

**Перечень ссылочных документов**

1. Hierarchical data formats – what is HDF5. Электронный ресурс: https://www.neonscience.org/about-hdf5.
2. 07623974.00015-01 91 01 Программный комплекс для работы со структурированными файлами SIO. Описание программы. Снежинск, 2022 г. (свидетельство о государственной регистрации программы для ЭВМ № 2022666734 от 06.09.2022)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего  листов  (страниц)  в документе | №  документа | Входящий  № сопрово-  дительного  документа  и дата | Подп. | Дата |
| Изм. | изме-нен  ных | заме-  ненных | новых | анулиро-  ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |