IRBIS64.dll

Материал из Wikipedia

`IRBIS64.dll` – программная библиотека для доступа к базам данных ИРБИС 64 и обработки данных с использованием языка форматирования.

`IRBIS64.dll` использует менеджер памяти ShareMem.

Содержание

Общий порядок работы с базами данных ИРБИС с помощью библиотеки `IRBIS64.dll`

Для работы с базами данных ИРБИС с помощью библиотеки `IRBIS64.dll` предусмотрен следующий порядок:

- Инициализировать переменную типа `TIrbisSpace` с помощью функции `IrbisInit`. Эта переменная предназначена для хранения контекста работы с базой и требуется при вызове большинства функций библиотеки `IRBIS64.dll`.
- Инициализировать структуры `isisucw` и `isisacw` с помощью функции `irbis_uatab_init`. Эти структуры требуются для корректной работы некоторых команд форматирования.
- Открыть базу данных с помощью функций `IrbisInitMST`, `IrbisInitTerm`, `IrbisInitInvContext`.
- Работать с открытой базой данных.
- Закрыть базу данных с помощью функций `IrbisCloseMst`, `IrbisCloseTerm` или `IrbisClose`.

Для проверки успешности выполнения `IrbisInitMST`, `IrbisInitTerm` и других функций библиотеки `IRBIS64.dll` принято определять константу **`ZERO = 0`**.

```
// Инициализировать запись TIrbisSpace и вернуть указатель на неё.
function IrbisInit: PIrbisSpace; export;
```

```
// Открыть файлы MST и XRF.
// Параметры:
// SP – указатель на переменную типа TIrbisSpace;
// DataBase – полный путь к файлам MST и XRF с указанием имени файла, но без расширения
// (предполагается, что эти файлы находятся в одной папке, одинаково называются и отличаются
// расширением);
// ANumberShelfs – количество полок.
// В случае успешного открытия возвращает ZERO.
function IrbisInitMST(SP: PIrbisSpace; DataBase: Pchar; ANumberShelfs: integer): integer; export;
```

```
// Открыть инверсный файл.
// Параметры:
// SP – указатель на переменную типа TIrbisSpace;
// DataBase – полный путь к инверсному файлу с указанием имени файла, но без расширения.
// В случае успешного открытия возвращает ZERO.
function IrbisInitTerm(SP: PIrbisSpace; DataBase: Pchar): integer; export;
```

После окончания работы с библиотекой `IRBIS64.dll` необходимо освободить занятые во время работы ресурсы с помощью процедуры `IrbisClose`.

Примечание: процедура `IrbisClose` в том числе запускает процедуры `IrbisCloseMST`, `IrbisCloseTerm`.

```
// Освободить ресурсы, занятые во время работы с библиотекой IRBIS64.dll:
// закрыть файлы, освободить память.
// Параметры:
// SP – указатель на переменную типа TIrbisSpace.
procedure IrbisClose(SP: PIrbisSpace); export;
```

```
// Закрыть файлы MST и XRF.
// Параметры:
// SP – указатель на переменную типа TIrbisSpace.
procedure IrbisCloseMST(SP: PIrbisSpace); export;
```

```
// Закрыть инверсный файл.
// Параметры:
// SP – указатель на переменную типа TIrbisSpace.
procedure IrbisCloseTerm(SP: PIrbisSpace); export;
```

Работа с записями

```
function IrbisMaxMFN(PS: PIrbisSpace): longint; export;
```

Вернуть число, равное максимальному номеру MFN из имеющихся в базе записей + 1. Это тот номер MFN, который будет присвоен первой новой записи, добавленной в базу.

```
function IrbisRecord(PS: PIrbisSpace; Shelf, mfn: longint): integer; export;
```

Читать из базы данных запись с указанным MFN и поместить на полку.

Параметры:

- SP – указатель на переменную типа TIrbisSpace.
- Shelf – полка, на которую будет помещена запись.
- mfn – MFN записи, которую необходимо прочитать.
- В случае успешного выполнения возвращает ZERO.

Примечание: аналог функции RECORD в ISIS.

Пример – перебор всех записей в базе данных:

```
var
  mfn, maxMFN: Integer;
  rr: Integer;
begin
  maxMFN := IrbisMaxMfn(CurSpace);
  for mfn := 1 to maxMFN - 1 do begin
    rr := IrbisRecord(CurSpace, 0, mfn);
    if rr <> ZERO then continue;
```

```
  ...
end;
```

Чтение записи

```
function IrbisMFN(PS: PIrbisSpace; shelf: integer): longint; export;
```

Вернуть MFN записи, находящейся на полке.

```
function IrbisNFields(PS: PIrbisSpace; Shelf: integer): integer; export;
```

Вернуть количество вхождений полей (считая также повторения).

Примечание: аналог NFIELDS.

```
function Irbisnocc(PS: PIrbisSpace; Shelf, met: integer): integer; export;
```

Вернуть число повторений поля с заданной меткой.

Примечание: аналог NOCC.

```
function Irbisfldtag(PS: PIrbisSpace; Shelf, nf: integer): integer; export;
```

Вернуть метку поля по порядковому номеру поля в записи.

Примечание: аналог FLDTAG.

```
function IrbisFieldN(PS: PIrbisSpace; shelf, met, occ: integer): integer; export;
```

Возвращает порядковый номер поля с заданной меткой. Если поле отсутствует в записи, то возвращает 0.

Примечание: аналог функции FIELDN в ISIS.

```
function IrbisField(PS: PIrbisSpace; shelf, nf: integer; subfields: PChar): Pchar; export;
```

Получает элемент данных записи, помещённой на полку с помощью вызова IrbisRecord или аналога ISISRLOCK.

Параметры:

- nf – порядковый номер;
- subfields – подполе или подполя (указываются только символы, идентифицирующие подполе, без знака ^).

Примечание: аналог функции FIELD в ISIS.

Пример чтения подполя ^B 1-го повторения 952-го поля записи с указанным mfn:

```
var
  rr: Integer;
begin
  rr := IrbisRecord(CurSpace, 0, mfn);
```

```
 if rr <> ZERO then exit;

 s := string(IrbisField(CurSpace, 0, IrbisFieldN(CurSpace, 0, 952, 1), 'B'));
```

Изменение записи

```
// nf - порядковый номер поля в справочнике
// если nf<1 или nf>кол-ва полей, то поле добавляется последним
// возврат: 0 - успешное завершение <>0 - не успешное
// АНАЛОГ ISIS FLDADD
function Irbisfldadd(PS: PIrbisSpace; Shelf,met, nf: integer; pole: Pchar): integer;export;
```

```
// замена поля
// возврат: 0 - успешно; <>0 - неуспешно
// если pole=nil () - удаляет!!
// аналог FLDREP
function Irbisfldrep(PS: PIrbisSpace; Shelf,nf: integer; pole: Pchar): integer;export;
```

```
// опустошает запись
function Irbisfldempty(PS:PIrbisSpace;Shelf:integer):integer;export;
```

```
// изменитьномер записи
function Irbischangemfn(PS: PIrbisSpace; shelf,newmfn: integer):integer;export;
```

Добавление новой записи

```
// формирование заготовки для новой записи с номером Amfn
function Irbisnewrec(PS: PIrbisSpace; shelf: integer): integer; export;
```

Функции форматёра

Форматёр – модуль, реализующий набор функций для обработки данных с использованием языка форматирования.

```
// Задать формат для дальнейшей обработки.
// Line - формат.
function Irbis_InitPFT(PS: PIrbisSpace; Line: PChar): integer; far;
```

```
// Выполнить формат.
// shelf - полка с записью
// FmtExitDLL = 'IRBIS64'
// PS.workerbuf - результат выполнения формата.
function Irbis_Format(PS: PIrbisSpace; shelf, alt_shelf, trm_shelf, LwLn: Integer; FmtExitDLL : PChar): integer; far;
```

Пример использования языка форматирования с динамически формируемой записью (без сохранения в БД):

```
var
  IrbisSpace: PIrbisSpace;
  value: String;
  formatResult: String;
begin

...

Irbisnewrec(IrbisSpace, 0);
Irbisfldadd(IrbisSpace, 0, 952, 0, PChar('^b' + value));
Irbis_InitPFT(IrbisSpace, '&uf(''+3C'',"TXT="v952^b)');
Irbis_Format(IrbisSpace, 0, 1, 0, MAXRECORD, 'IRBIS64');
```

```
SetString(formatResult, IrbisSpace.workerbuf, strlen(IrbisSpace.workerbuf));
```

Функции работы с инверсным файлом

Поисковый ключ term должен иметь длину >=255 используйте тип TKey.

```
// Найти термин term.
// Возвращает:
// 0 - термин найден
// TERM_NOT_EXISTS = -202 в term - следующий термин
// TERM_LAST_IN_LIST = -203  в term - #0
// TERM_FIRST_IN_LIST = -204  в term - первый термин словаря
function Irbisfind(PS: PIrbisSpace; term: Pchar): integer; export;
```

```
// дать следующий термин
// возврат 0 и term TERM_LAST_IN_LIST и #0
function Irbisnxtterm(PS: PIrbisSpace; term: Pchar): integer; export;
```

```
// дать предидущий термин
// возврат 0 и term TERM_FIRST_IN_LIST и #0
function Irbisprevterm(PS: PIrbisSpace; term: Pchar): integer; export;
```

```
// Возвращает число ссылок на термин.
// Функция предназначена для использования после вызова Irbisfind.
function Irbisnposts(PS: PIrbisSpace): longint; export;
```

```
// Разобрать ссылку на: mfn, tag, occ, cnt.
// Возможные значения opt:
// opt = 1 функция возвращает mfn (номер записи);
// opt = 2 функция возвращает tag (метка поля);
// opt = 3 функция возвращает occ (повторение поля);
// opt = 4 функция возвращает cnt (номер слова).
// Возвращает: в случае успеха положительное значение (больше ноля).
function IrbisPosting(PS: PIrbisSpace; opt: smallint): longint; export;
```

```
// инициализация работы со ссылками - не вызывается напрямую процедура скрыта в FIND
function Irbisinitpost(PS: PIrbisSpace): integer; export;
```

```
// Перейти к следующей ссылке.
// Возвращает: 0 - успешно выполнено; -1 - конец списка.
function IrbisNxtPost(PS: PIrbisSpace): integer; export;
```

```
// установить указатель постингов на заданном чтобы читать дальше NXTPost с него!!!
function IrbisFindPosting(PS: PIrbisSpace; const Term: PChar; const posting: TifpItemPosting): integer; export;
```

Пример поиска термина в инверсном файле при помощи функции Irbisfind:

```
procedure example(IrbisSpace: PIrbisSpace; prefixedTermin: String);
var
  isLongTermin: Boolean;
  prefixedTermin_: String;
  res: Integer;
  irbisKey: TIrbisKey;
  currentKey: String;
begin
  isLongTermin := length(prefixedTermin) >= LONGTERM;
  if not isLongTermin then begin   // искомый термин может поместиться в словаре
    prefixedTermin_ := prefixedTermin;
  end else begin                 // искомый термин не может поместиться в словаре
    prefixedTermin_ := Copy(prefixedTermin, 0, LONGTERM);
```

```
  end;

  res := IrbisFind(IrbisSpace, irbisKey);
  if (res = ZERO) and (not isLongTermin) then begin
    // термин найден
  end else begin
    // Термин не найден, но искомый термин такой длины,
    // что если и был помещён в словарь, то оказался обрезан.
    // Попробуем перебрать термины индексного файла,
    // которые могут быть началом искомого термина.
    while true do begin
      SetString(currentKey, irbisKey, strlen(irbisKey));

      // если найденный термин индексного файла не соответствует началу искомого термина,
      // то прекращаем перебор
      if not havePrefix(prefixedTermin_, currentKey, False) then
        break;

      // перебрать записи, на которые указывают постинги
      for i := 1 to IrbisNPosts(IrbisSpace) do begin
        // откроем запись
        if IrbisNxtPost(IrbisSpace) <> 0 then break;
        if IrbisPosting(IrbisSpace, 1) <= 0 then
          continue;
        if IrbisRecord(IrbisSpace, shelf, IrbisPosting(IrbisSpace, 1)) <> ZERO then
          continue;

        // здесь можно читать содержимое записи, чтобы проверить, содержит ли запись искомый длинный термин.
      end;

      // Если термин был последним в индексном файле, то прекращаем перебор
      if res = -203 then break;

      // Взять следующий термин из индексного файла
      res := Irbisnxtterm(IrbisSpace, irbisKey);
    end;
  end;
```

Функции, осуществляющие запись в БД

```
// опустошение БД
function IrbisDBEmptyTime(IrbisSpace:PIrbisSpace;seconds:integer):integer;export;
```

```
// заблокировать БД
function IrbisLockDBTime(IrbisSpace:PIrbisSpace;seconds:integer):integer;export;
```

```
// снять блокировку БД
function IrbisUnLockDBTime(IrbisSpace:PIrbisSpace;seconds:integer):integer;export;
```

```
// базовая функция записи с полки с актуализацией и разблокировкой
function IrbisRecUpdateTime(IrbisSpace:PIrbisSpace;Shelf:integer;KeepLock:integer;Updif:boolean;seconds:integer;
  var result_update:integer;var result_updif:integer):integer;export;
```

```
// базовая функция актуализации записи
function IrbisRecIfUpdateTime(IrbisSpace:PIrbisSpace;Shelf, mfn:integer;seconds:integer):integer;export;
```

```
// базовая функция блокировки записи
// аналог RECORD
function IrbisRecLockTime(IrbisSpace: PIrbisSpace; Shelf,mfn: longint;seconds:integer): integer;export;
```

```
// базовая функция разблокировки записи
function IrbisRecUnLockTime(IrbisSpace:PIrbisSpace;mfn:integer;seconds:integer):integer;export;
```

Ссылки

См. также:

- Базы данных ИРБИС
- Язык форматирования системы ИРБИС
- UNIFOR

Источник — «http://wiki.elnit.org/index.php/IRBIS64.dll»
Категории: Программные модули ИРБИС | Базы данных ИРБИС | ИРБИС 64