

Программирование

Лекция 7

Петров Александр Владимирович
Фёдоров Станислав Алексеевич

**(по материалам Веренинова Игоря
Андреевича с изменениями и
дополнениями на Fortran 08 и UML)**

Сентябрь 2013

Назначение файлов

Исходные данные

Промежуточные вычисления

Результаты вычислений

Преобразование данных

Обмен данными с другими программами

Внешние и внутренние файлы

Внешний файл –
именованная область на внешнем носителе.

C:\docum.inf

D:\DATA\results.txt

E:\geometry.dat

Внутренний файл –
символьная строка или массив.

character(100) buffer

characrer(1000) temp(10)

Файловые записи

Запись – единица обмена данными между программой и внешней памятью.

Расположены в файле последовательно.

Форматные записи
внутреннее □ внешнее представление

Неформатные записи
внутренне представление

Запись конец файла
последняя запись в файле.

Форматные файлы

Содержат форматные записи.

Каждая запись оканчивается управляющими символами (возврат каретки, перевод строки).

Возможность "ручного" редактирования.

Скорость обработки файлов низкая.

Большой объём файлов.

Внешние и внутренние файлы.

Неформатные файлы

Содержат неформатные записи.

Отсутствует возможность "ручного" редактирования.

Скорость обработки файлов высокая.

Меньший объём файлов.

Внешние файлы.

Двоичные файлы

Содержат данные в двоичном представлении.

Длина записи равна 1 байту.

Отсутствует возможность "ручного" редактирования.

Эффективны для хранения больших объёмов данных (хранение промежуточных вычислений).

Внешние файлы.

Последовательный доступ

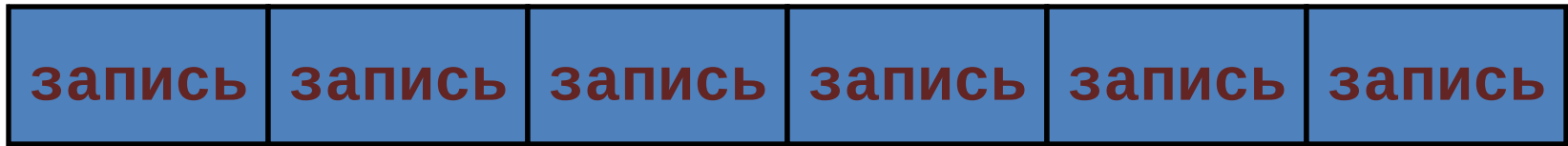


Доступ к данным по порядку

Записи переменной длины

Добавление новой записи – в конец файла

Прямой доступ



файловый
указатель

Доступ к данным произвольный

Записи одинаковой длины

Внешний файл
может быть прямого и последовательного доступа

Внутренний файл
только последовательного доступа

Оператор Open

```
open (unit   = u           , &  
      file   = name        , &  
      err    = label       , &  
      iostat = i-var       , &  
      ...)
```

Создает устройство ввода/вывода с номером **u** и подсоединяет к нему внешний файл **name**.

```
open (unit = 1, file = "data.txt")  
open (unit = 2, file = "D:\DOCUM\price.inf")
```

Параметры оператора Open

uint - номер устройства

file - имя файла

err - метка на оператор обработки ошибки

iostat - номер ошибки (0 - отсутствие).

... - один или несколько спецификаторов (≈ 40)

Параметры оператора Open

СПЕЦИФИКАТОРЫ

ACCESS

ACTION

ASSOCIATEVARIABLE

ASYNCHRONOUS

BLANK

BLOCKSIZE

BUFFERCOUNT

BUFFERED

CARRIAGECONTROL

CONVERT

DECIMAL

DEFAULTFILE

DELIM

DISPOSE

ENCODING

FORM

IOFOCUS

MAXREC

MODE

NAME

NEWUNIT

NOSHARED

ORGANIZATION

PAD

POSITION

READONLY

RECL

RECORDSIZE

RECORDTYPE

ROUND

SHARE

SHARED

SIGN

STATUS

TITLE

TYPE

USEROPEN

Примеры оператора Open

! двоичный файл

```
open(1,file = "backup.bin",form = 'binary')
```

! добавление записей в конец файла

```
open(2,file = "history.txt",access = 'append')
```

! только для чтения

```
open(3,file = "data.txt", action = 'read')
```

! файл должен существовать

```
open(4,file = "geometry.dat", status = 'old')
```

! асинхронный ввод/вывод

```
open(5,file = "tornado.dat", asynchronous = 'yes')
```

! файл доступен другим приложениям

```
open(6,file = "base.dat", share = 'denynone')
```

! файл недоступен другим приложениям

```
open(7,file = "base2.dat", share = 'denyrw')
```

Файловый ввод/вывод

Оператор **read** – чтение данных из файла

Последовательный доступ

форматные

```
read(unit, format, ...)
```

неформатные

```
read(unit, ...)
```

Прямой доступ

форматные

```
read(unit, format, rec, ...)
```

неформатные

```
read(unit, rec, ...)
```

Внутренние файлы

```
read(unit, format, ...)
```

Ввод/вывод в файлы

Оператор **write** – запись данных в файл

Последовательный доступ

форматные

```
write(unit, format, ...)
```

неформатные

```
write(unit, ...)
```

Прямой доступ

форматные

```
write(unit, format, rec, ...)
```

неформатные

```
write(unit, rec, ...)
```

Внутренние файлы

```
write(unit, format, ...)
```

ПРИМЕРЫ

! ЗАПИСЬ ДАННЫХ ВО ВНТУРЕННИЙ ФАЙЛ

```
program buffer
  character(20) buf
  integer(4) a,b,c,d

  !----- читаем формулу в символьную строку
  write(*,"(A,\)") "Enter expression....."
  read(*,"(A)") buf ! во внутренний файл

  !----- заменяем все плюсы на пробелы
  do k = 1,len(buf)
    if (buf(k:k) == '+') buf(k:k) = ' '
  end do

  read(buf,*) a,b,c,d

  write(*,*) a+b+c+d

end
```


ПРИМЕРЫ

! ЗАПИСЬ ДАННЫХ ВО ВНЕШНИЙ ФАЙЛ ПРЯМОГО ДОСТУПА

```
program random_file
real x
integer :: k = 0
  open (1, file = "C:\numbers.txt", access = 'direct', &
        recl = 10, form = 'formatted')
  do
    call random_number(x)
    k = k+1
    x = x-0.78
    write(1, "(f10.4)", rec = k) x
    if (abs(x)<0.0001) exit
  end do
  write(1, "(i10)", rec = 1) k
  close(1)
end
```

Данные записанные в

4644	-0.7545	-0.4275	-0.1131	0.1831	0.0583...
-------------	---------	---------	---------	--------	-----------

Оператор Backspace

Перемещает файловый указатель на одну запись назад в файлах последовательного доступа.

```
backspace (unit    = u      , &  
            err     = label , &  
            iostat  = i-var)
```

```
backspace(1)
```

```
backspace(2, err = 200)
```

```
backspace(3, err = 200, iostat = ios)
```

Оператор Rewind

Перемещает файловый указатель
в начало первой записи.

```
rewind (unit = u      , &  
        err  = label , &  
        iostat = i-var)
```

```
rewind(1)
```

```
rewind(2, err = 200)
```

```
rewind(3, err = 200, iostat = ios)
```

Функция EOF

Возвращает **.TRUE.** если файловый указатель установлен на запись "конец файла".
В противном случае результат **.FALSE.**

Часто используется для чтения
всех данных из файла

```
do while ( .NOT.eof(1) )  
    read(1, *) param  
    . . .  
end do
```

Функция EOF

```
program read_file
real(4), allocatable :: A(:)
real(4) AT
integer(8) :: k = 0

open(1, file = "C:\data.txt")
do while (.NOT.EOF(1)) ! подсчёт элементов массива
    read(1, *) AT
    k = k+1
end do
allocate(A(k)) ! размещение массива

rewind(1)
k = 1
do while (.NOT.EOF(1)) ! чтение элементов массива
    read(1, *) A(k)
    k = k+1
end do
end
```

Оператор Inquire

Возвращает свойства устройства, файла или папки.

```
INQUIRE (FILE=name, ERR=label, ID=idvar,      &  
          IMSG=msgvar, SIZE=sz, IOSTAT=ivar, &  
          DEFAULTFILE=def, slist)
```

```
INQUIRE (UNIT=iounit,  ERR=label, ID=idvar,  &  
          IMSG=msgvar, SIZE=sz,              &  
          IOSTAT=ivar, slist)
```

```
INQUIRE (DIRECTORY=dir, EXIST=ex,            &  
          DIRSPEC=dirspeg, ERR=label,         &  
          ID=idvar, IMSG=msgvar,              &  
          SIZE=sz, IOSTAT=ivar)
```

```
INQUIRE (IOLENGTH=len) out_item_list
```

Оператор Inquire

```
program inquire_list ! ***** Вычисление размера списка вывода
real(8) A(100)
complex(16) S(20)
integer ioL

inquire(iolength = ioL) A,S,B ! размер списка вывода = 361
end
```

```
program if_exist ! ***** Проверка существования файла
character(100) fname
logical exists

write (*, *) 'Enter the file name: '
read (*, '(a)') fname

inquire (file = fname, exist = exists)

if (.not. exists) write (*, '(2a/)') 'Cannot find ', fname
end
```

Оператор Close

Отсоединяет файл от устройства ввода/вывода и закрывает это устройство

```
close (unit    = u      , &  
        err     = label , &  
        iostat  = ivar  , &  
        status  = stat)
```

status – символьное выражение принимающее значения: **keep** или **delete**

```
program delete_file ! ***** Удаление файла  
  open(1,file = "C:\data.txt")  
  read(1,*) a,b,c  
  close(1,status = 'delete')  
end
```


Асинхронный ввод/вывод

Параллельное выполнение файлового ввода/вывода и других операторов программы.

Спецификатор **asynchronous='yes'** в операторах **open, write, read**.

```
open (1, asynchronous = 'YES')
```

! синхронный вывод

```
write(1, *, asynchronous = 'NO') A, B, C
```

! асинхронный вывод

```
write(1, *, asynchronous = 'YES') D, E, F
```

Асинхронный ввод/вывод

Переменные участвующие в асинхронном вводе/выводе получают атрибут **asynchronous**.

Явное задание атрибута при объявлении.

real, asynchronous :: MS, NV

complex, asynchronous :: TN(100,100)

Оператор **wait**

ожидает окончание асинхронной работы с файлом.

Асинхронный ввод/вывод

Неаккуратное использование асинхронной работы с файлом может приводить к гонкам данных.

```
program asyn_data_races
```

```
  real(4) :: A(100,100) = 1.0
```

```
  open(1,file = "C:\AS.txt", asynchronous = 'YES')
```

```
  do k = 1,10
```

```
    rewind(1)
```

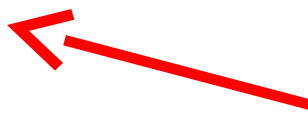
```
    write (1,*, asynchronous = 'YES') A
```

```
    A = real(k)
```

```
  end do
```

```
  close(1)
```

```
end
```



Массив А
записывается в файл
и одновременно
изменяется !

Асинхронный ввод/вывод

Содержимое файла **C:\AS.txt**
содержит разные данные !

```
...  
9.000000      9.000000      9.000000      9.000000  
9.000000      9.000000      9.000000      9.000000  
9.000000      9.000000      9.000000      9.000000  
9.000000      9.000000      9.000000      9.000000  
9.000000      9.000000      9.000000      10.00000  
10.00000      10.00000      10.00000      10.00000  
10.00000      10.00000      10.00000      10.00000  
10.00000      10.00000      10.00000      10.00000  
10.00000      10.00000      10.00000      10.00000  
10.00000      10.00000      10.00000      10.00000  
...
```

Асинхронный ввод/вывод

Используем оператор **wait**, чтобы дождаться завершения записи данных в файл.

```
program asyn_data_races

  real(4) :: A(100,100) = 1.0

  open(1,file = "C:\AS.txt", asynchronous = 'YES')

  do k = 1,10
    rewind(1)
    write (1,*, asynchronous = 'YES') A
    wait(1) ! ожидаем, когда завершится поток,
             ! отвечающий за запись данных в файл
    A = real(k)
  end do
  close(1)
end
```

Оператор Flush

Сброс логических буферов
при буферизованном выводе

```
program prog
integer, parameter :: N = 100
real A(N)
  open(1, buffered = 'yes', file = 'dat')
  write(1, *) A
  flush(1)

  call C_subroutine('dat') ! читает данные из файла dat
! ----- некоторый код
end
```

При буферизации, логически записанные данные
могут
физически не успеть попасть на диск,
что вызовет ошибку чтения данных.