



ESRI Shape 文件技术说明书

ESRI 白皮书—1998 年 7 月

初译: yzg
2006 年 5 月 4 日

Copyright © 1997, 1998 Environmental Systems Research Institute, Inc.
All rights reserved.
Printed in the United States of America.

The information contained in this document is the exclusive property of Environmental Systems Research Institute, Inc. This work is protected under United States copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by Environmental Systems Research Institute, Inc. All requests should be sent to Attention: Contracts Manager, Environmental Systems Research Institute, Inc., 380 New York Street, Redlands, CA 92373-8100 USA.

The information contained in this document is subject to change without notice.

U.S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is Environmental Systems Research Institute, Inc., 380 New York Street, Redlands, CA 92373-8100 USA.

In the United States and in some countries, ARC/INFO, ArcCAD, ArcView, ESRI, and PC ARC/INFO are registered trademarks; 3D Analyst, ADF, AML, ARC COGO, ARC GRID, ARC NETWORK, *Arc News*, ARC TIN, ARC/INFO, ARC/INFO LIBRARIAN, ARC/INFO—Professional GIS, ARC/INFO—The World's GIS, ArcAtlas, ArcBrowser, ArcCAD, ArcCensus, ArcCity, ArcDoc, ARCEDIT, ArcExplorer, ArcExpress, ARCPLLOT, ArcPress, ArcScan, ArcScene, ArcSchool, ArcSdl, ARCSHELL, ArcStorm, ArcTools, ArcUSA, *ArcUser*, ArcView, ArcWorld, Atlas GIS, AtlasWare, Avenue, *BusinessMAP*, DAK, DATABASE INTEGRATOR, DBI Kit, ESRI, ESRI—Team GIS, ESRI—The GIS People, FormEdit, Geographic Design System, GIS by ESRI, GIS for Everyone, GISData Server, IMAGE INTEGRATOR, *InsiteMAP*, MapCafé, MapObjects, NetEngine, PC ARC/INFO, PC ARCEDIT, PC ARCPLLOT, PC ARCSHELL, PC DATA CONVERSION, PC NETWORK, PC OVERLAY, PC STARTER KIT, PC TABLES, SDE, SML, Spatial Database Engine, StreetMap, TABLES, the ARC COGO logo, the ARC GRID logo, the ARC NETWORK logo, the ARC TIN logo, the ARC/INFO logo, the ArcCAD logo, the ArcCAD WorkBench logo, the ArcData emblem, the ArcData logo, the ArcData Online logo, the ARCEDIT logo, the ArcExplorer logo, the ArcExpress logo, the ARCPLLOT logo, the ArcPress logo, the ArcPress for ArcView logo, the ArcScan logo, the ArcStorm logo, the ArcTools logo, the ArcView 3D Analyst logo, the ArcView Data Publisher logo, the ArcView GIS logo, the ArcView Internet Map Server logo, the ArcView Network Analyst logo, the ArcView Spatial Analyst logo, the ArcView StreetMap logo, the Atlas GIS logo, the Avenue logo, the *BusinessMAP* logo, the *BusinessMAP* PRO logo, the Common Design Mark, the DAK logo, the ESRI corporate logo, the ESRI globe logo, the MapCafé logo, the MapObjects logo, the MapObjects Internet Map Server logo, the NetEngine logo, the PC ARC/INFO logo, the SDE logo, the SDE CAD Client logo, The World's Leading Desktop GIS, ViewMaker, *Water Writes*, and Your Personal Geographic Information System are trademarks; and ArcData, ARCMail, ArcOpen, ArcQuest, *ArcWatch*, ArcWeb, Rent-a-Tech, www.esri.com, and @esri.com are service marks of Environmental Systems Research Institute, Inc.

The names of other companies and products herein are trademarks or registered trademarks of their respective trademark owners.

注意：本人非英语、计算机及 GIS 专业从业人员，翻译此文仅供参考之用。
受本人水平所限，出现错误之处实属正常，如有有心人能给予指正，将不胜感激。

2006 年 5 月 4 日
YZG

ESRI Shapefile 技术说明书

ESRI 白皮书

Contents	page
Why Shapefiles?	1
Shapefile Technical Description	2
Organization of the Main File	3
Main File Record Contents	5
Organization of the Index File	5
Organization of the dBASE File	18
Glossary	20

ESRI Shapefile 技术说明书

This document defines the shapefile(.shp)spatial data format and describes why shapefiles are important. It lists the tools available in Environmental Systems Research Institute, Inc.(ESRI),software for creating shapefiles directly or converting data into shapefiles from other formats. This document also provides all the technical information necessary for writing a computer program to create shapefiles without the use of ESRI® software for organizations that want to write their own data translators.

Why Shapefiles?

A shapefile stores nontopological geometry and attribute information for the spatial features in a data set. The geometry for a feature is stored as a shape comprising a set of vector coordinates.

Because shapefiles do not have the processing overhead of a topological data structure, they have advantages over other data sources, such as faster drawing speed and edit ability. Shapefiles handle single features that overlap or that are noncontiguous. They also typically require less disk space and are easier to read and write.

Shapefiles can support point, line, and area features. Area features are represented as closed loop, double-digitized polygons. Attributes are held in a dBASE® format file. Each attribute record has a one-to-one relationship with the associated shape record.

如何创建 Shape 文件

Shape 文件可以通过以下四种方法建立：

- 导出 — 可以通过 ARC/INFO、PC ARC/INFO、Spatial Database Engine™(SDE™)、ArcView® GIS 或者 BusinessMAP™ software 等软件将任何源数据导出为 Shape 文件。
- 数字化 — 可以使用 ArcView GIS 的要素创建工具，通过直接数字化的方法生成 Shape 文件。
- 编程 — 可以使用 Avenue™(ArcView GIS)、MapObjects™、ARC Macro Language(AML™)(ARC/INFO)或者 Simple Macro Language(SML™)(PC ARC/INFO)等开发工具编程来创建 shape 文件。
- 创建程序按照文件规范直接写 shape 文件。

SDE、ARC/INFO、PC ARC/INFO、Data Automation Kit(DAK™)以及 ArcCAD® 等软件均提供了 shape 向 coverage 的转换工具，ARC/INFO 也提供了 coverage 到 shape 的转换功能。Shape 文件格式以白皮书的形式发布以便于与其它数据的交换。一些其它数据流，像 GPS 接收机的数据等等都可以存储为 shape 文件或 X、Y event tables。

Shapefile Technical Description

Shape 文件技术说明书

可以利用本文档中的说明书来创建计算机程序，用来读写 shape 文件。

ESRI 的 shape 文件由一个主文件、一个索引文件和一个 dBASE 表构成。主文件是一个可变记录长度的随机文件，文件中的每个记录描述一个包含多个顶点的 shape。在索引文件中，每个记录内容包含着与主文件中记录相对应的从主文件开始处的偏移量。dBASE 表中包含着与每个要素相对应的一条要素属性记录。几何数据与属性的一一对应关系是基于记录号来对应的。dBASE 文件中属性记录的顺序必须与主文件中的记录顺序相同。

Naming Conventions

命名规则

所有的文件遵从 8.3 的命名规则。主文件、索引文件、dBASE 文件拥有相同的文件名。文件名的开头字母必须是数字或字母（包括 a—Z、0—9），接下来是 0 到 7 个字符（包括 a—Z、0—9、_、-）。主文件的扩展名为 “.shp”，索引文件的扩展名为 “.shx”，dBASE 文件的扩展名为 “.dbf”。对于区分大小写的操作系统，文件名要全部采用小写方式。

例

- 主文件： counties.shp
- 索引文件： counties.shx
- dBASE 文件： counties.dbf

Numeric Types

数字类型

Shape 文件中存储着整数和双精度数，在接下来的文档内容中会提到以下的数据类型：

- Integer： 有符号 32 位整数（4 字节）
- Double： 有符号 64 位 IEEE 双精度浮点数（8 字节）

浮点数必须是数字值，正、负无穷大、非数字（NaN）值在 shape 文件中是不允许使用的。不过，shape 文件支持 “no data” 值的概念，但在目前这也仅仅用于 measures。在读取文件时，任何小于 -10^{38} 的浮点数被认为是 “no data” 值。

在下面的第一个部分里首先描述 shape 文件的一般结构和组织形式。第二个部分描述 shape 文件所支持类型的记录内容。

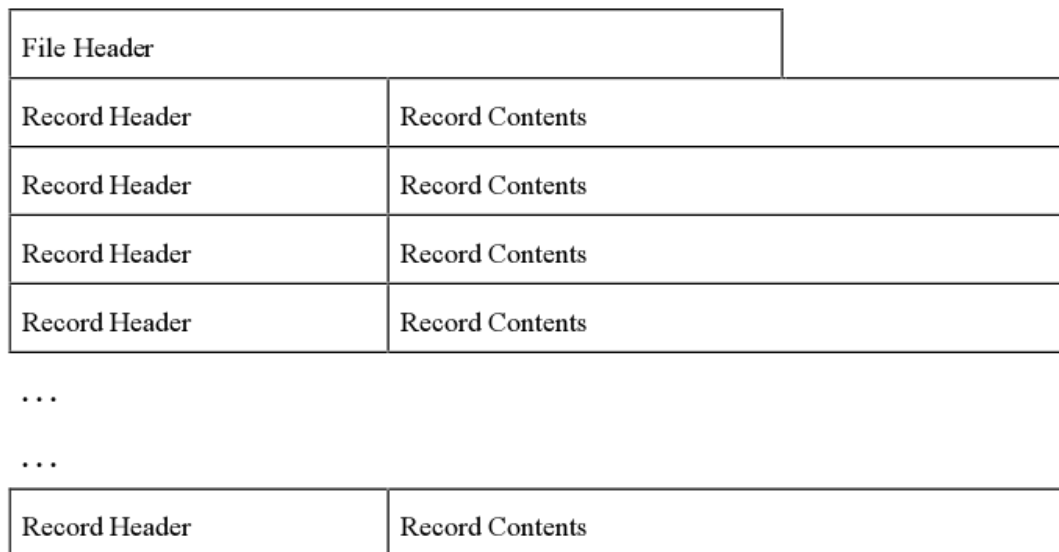
Organization of the Main File

主文件的组织

主文件（.shp）包含一个固定长度的文件头，在文件头的后面存储着可变长度的记录。每个可变长度记录由一个固定长度的记录头和跟随其后的可变长度记录内容组成。见图 1。

图 1

主文件组织



Byte Order

字节顺序

shape 文件中的所有内容可以分为两种类型：

- 数据相关的
 - 主文件记录内容
 - 主文件头数据描述区域（shape 类型、边界框等）
- 文件管理相关的
 - 文件和记录的长度
 - 记录的偏移量等等

本文档中的整数和双精度浮点数采用 little endian（PC 和 Intel）和 big endian（Sun Motorola）两种表达方式，在主文件头中构成数据描述区域的和主文件记录内容中的采用 little endian 的方式；在文件的其余部分和文件管理采用 big endian 的方式。

The Main File Header

主文件头

主文件头长度为 100 字节，表 1 中显示了文件头中的字段以及它们的字节位置、值、类型、字节顺序。在此表中，位置是从文件头算起的。

表 1 主文件头的描述

Position	Field	Value	Type	Byte Order
Byte 0	File Code	9994	Integer	Big
Byte 4	Unused	0	Integer	Big
Byte 8	Unused	0	Integer	Big
Byte 12	Unused	0	Integer	Big
Byte 16	Unused	0	Integer	Big
Byte 20	Unused	0	Integer	Big
Byte 24	File Length	File Length	Integer	Big
Byte 28	Version	1000	Integer	Little
Byte 32	Shape Type	Shape Type	Integer	Little
Byte 36	Bounding Box	Xmin	Double	Little
Byte 44	Bounding Box	Ymin	Double	Little
Byte 52	Bounding Box	Xmax	Double	Little
Byte 60	Bounding Box	Ymax	Double	Little
Byte 68*	Bounding Box	Zmin	Double	Little
Byte 76*	Bounding Box	Zmax	Double	Little
Byte 84*	Bounding Box	Mmin	Double	Little
Byte 92*	Bounding Box	Mmax	Double	Little

* Unused, with value 0.0, if not Measured or Z type

文件长度的值是用 16 位字表示的文件总长度(包括组成文件头的 50 个 16 位字)。shape 文件中所有的非空 shape 必须是相同类型的。以下是 shape 类型的值:

Value	Shape Type
0	Null Shape
1	Point
3	PolyLine
5	Polygon
8	MultiPoint
11	PointZ
13	PolyLineZ
15	PolygonZ
18	MultiPointZ
21	PointM
23	PolyLineM
25	PolygonM
28	MultiPointM
31	MultiPatch

未列入上述清单中的 Shape type 有 2、4、6 以及 33 等等, 这些是做为保留以备将来使用。目前每个 shape 文件被限定为只能包含上述类型的一种, 将来 shape 文件可能会被允许包含一种以上的 shape 类型, 如果实现了这种混合类型, 文件头中的 shape 类型字段也

采用与上面相同的定义。

主文件头中的 **Bounding Box** 储存着 **shape** 的实际范围：与 **X** 轴和 **Y** 轴（潜在的 **M** 和 **Z**）正交的包含所有 **shape** 的最小边界矩形。如果 **shape** 文件是空的（即文件不包含任何记录），则 **Xmin**、**Ymin**、**Xmax**、**Ymax** 均未指定。**Shape** 文件中具有 **measure shape** 类型的 **Mmin** 和 **Mmax** 可以被指定为“no data”值（参见第 2 页“数字类型”），也就是不包含 **measure** 值。

Record Headers

记录头

每个记录的记录头都储存着记录号和记录内容的长度，记录头的长度是 8 个字节，记录号从 1 开始。字节位置是相对于记录头算起的。见表 2。

表 2 主文件记录头的描述

Position	Field	Value	Type	Byte Order
Byte 0	Record Number	Record Number	Integer	Big
Byte 4	Content Length	Content Length	Integer	Big

记录的长度以 **word** 来计算，每个记录的长度是记录内容的长度再加上的记录头（4 个 **word**）的长度，与主文件中第 24 个字节的内容类似。

Main File Record Contents

主文件记录内容

Shape 文件的记录内容由 **shape** 类型和其后面的 **shape** 的几何数据组成。记录内容的长度取决于 **part** 的数量和 **shape** 的顶点数目。对于每个 **shape** 类型，我们先给出它的描述，然后是它在磁盘上的记录内容。在表 3 至表 16 中，**position** 是从记录内容的开始算起的。

Null Shape

空 shape

Shape 类型为 0 表示一个没有几何类据的空的（**Null**）**shape**。每个要素类型（**point**、**line**、**polygon** 等等）都支持“空状态”——在同一个 **shape** 文件中“有点”（**have point**）和“无点”（**null point**）都是合法的。空 **shape** 经常作为位置标志符来使用，在创建 **shape** 文件时使用它，稍后就会被填充为几何数据。

表 3 空 shape 记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	0	Integer	1	Little

Shape Types in X,Y Space

XY 平面上的 shape 类型

Point（点）

一个 Point 由一对双精度坐标组成，存储顺序为 X，Y。

```
Point
{
    Double      X      // X coordinate
    Double      Y      // Y coordinate
}
```

表 4 Point 记录的内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	1	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little

MultiPoint（多点）

一个 MultiPoint 描述一组点，具体如下：

```
MultiPoint
{
    Double[4]      Box      // Bounding Box
    Integer        NumPoints // Number of Points
    Point[NumPoints] Points // The Points in the set
}
```

Bounding box 的存储顺序是 Xmin, Ymin, Xmax, Ymax.

表 5 MultiPoint 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	8	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little

PolyLine（第二版中称为 Arc）

一条 Polyline 是一个按次序排列的顶点序列，包含一个或几个 part，一个 part 是由两个或两个以上的点连接而成的序列，Part 之间互相连接或不连接均可，Part 之间可以交叉也可以不交叉。

在本规范中没有禁止使用连续的相同坐标，所以要注意处理这种情况。另一方面，退

化的 (degenerate)、长度为 0 的 part 是不允许出现的。

```
PolyLine
{
    Double[4]          Box          // Bounding Box
    Integer             NumParts     // Number of Parts
    Integer             NumPoints    // Total Number of Points
    Integer[NumParts]   Parts        // Index to first Point in Part
    Point[NumPoints]    Points       // Points for all parts
}
```

PolyLine 的各个字段描述如下：

- Box:** PolyLine 的 Bounding box，按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。
- NumParts:** PolyLine 中 Part 的数目。
- NumPoints:** 所有 Part 的总点数。
- Parts:** 长度为 NumParts 的数组，存储着每个 part（注：原文中是 polyline）的起始点在 points 数组中的索引，索引从 0 开始。
- Points:** 长度为 NumPoints 的数组，按顺序存储构成 PolyLine 的所有 Part 的点。组成序号为 2 的 Part 的点紧接着序号为 1 的，依此类推。数组 Parts 中存储着每个 Part 起点的数组索引。在 points 数组中，各 Part 之间没有分隔符。

表 6 Polyline 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	3	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Note: $X = 44 + 4 * \text{NumParts}$					

Polygon（面）

Polygon 由一个或多个环组成。环是一个由 4 个或 4 个以上的顺序连接的点构成的闭合的、非自相交的回路。Polygon 可以包含多个外部环。顶点的顺序或方向表明环的哪一侧是处于 Polygon 内部的。沿着一个环的顶点顺序前进，前进方向的右侧就是这个环所在的 Polygon。在 Polygon 中由顶点组成的洞是逆时针方向的。单一且闭合的 Polygon 的结点顺序总是顺时针方向的。组成 Polygon 的 ring 就是 Polygon 的 Part。

在本规范中没有禁止使用连续相同的点，所以要注意处理这种情况。另一方面，退化的 (degenerate)、长度为 0 的 part 是不允许出现的。

Polygon 的结构与 Polyline 的结构是相同的，具体如下：

```
Polygon
{
    Double[4]           Box           // Bounding Box
    Integer             NumParts      // Number of Parts
    Integer             NumPoints     // Total Number of Points
    Integer[NumParts]   Parts         // Index to First Point in Part
    Point[NumPoints]    Points        // Points for All Parts
}
```

Polygon 中各个字段的描述如下：

Box: Polygon 的封装边界，存储顺序为：Xmin, Ymin, Xmax, Ymax。

NumParts: Polygon 中环的个数。

NumPoints: 构成所有环的点的数目。

Parts: 长度为 NumParts 的数组，存储着每个环的首点在 Points 数组中的索引，数组索引从 0 开始。

Points: 长度为 NumPoints 的数组。构成 Polygon 的每个环的点，按照首尾相连的顺序存储的。组成序号为 2 的环的点紧接着序号为 1 的环，依此类推。数组 Parts 中存储着每个环的起点的数组索引。组成不同环的点之间没有分隔符。

在图 2 的例子中表明了 polygon 的表示方法。图中显示的是内含一个岛、由 8 个顶点组成的 Polygon。

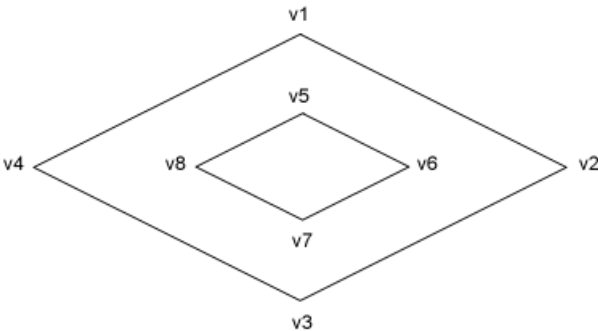
对于 Polygon 要注意以下几点：

- 环是闭合的（首点与末点必须是相同的）。
- 环在数组 points 中的次序是不重要的。
- 存储在 shapefile 中的 Polygon 必须是“干净的”的，“干净的”Polygon 是这样的：

1.不能自交叉。这意味着：属于一个环的片断不能与属于另一个环的片断交叉；组成 Polygon 的环之间可以在顶点处相接，但不能有重合片断，重合的片断被认为是交叉的。

2.含有 polygon 的 inside，并且这些 inside 是由有“正确”侧边的线组成的。沿着环的结点顺序的前进方向右侧是 polygon 的内部。对于由单一环构成的 polygon，其顶点顺序总是顺时针方向。Polygon 中构成岛的环是逆时针方向的，如果构成岛的环是顺时针方向，这时将产生一个“脏的” polygon，导致出现叠加。

图 2 Polyline 的实例



在本例中，NumParts 为 2，NumPoints 为 10。注意在图中 polygon 内的岛，组成它的点的顺序是倒向的。

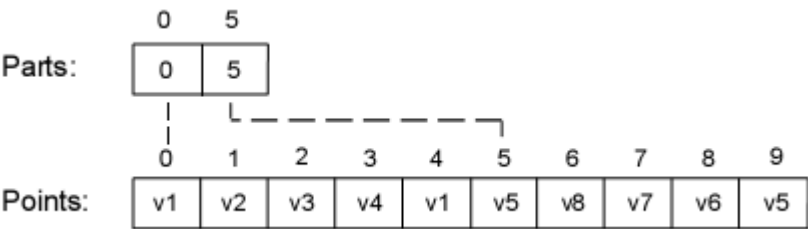


表 7 Polygon 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	5	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Note: X = 44 + 4 * NumParts					

Measured Shape Types in X,Y Space

XY 平面上的 Measured Shape 类型

这个类型的 Shape 有一个附加的坐标—M。M 的值可以是 “no data”（参见第 2 页 “数字类型”）

PointM

一个 PointM 由一对双精度坐标 X、Y 加上 M 构成。

```

PointM
{
    Double      X      // X coordinate
    Double      Y      // Y coordinate
    Double      M      // Measure
}

```

表 8 PointM 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	21	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little
Byte 20	M	M	Double	1	Little

MultiPointM

MultiPointM 描述一组 PointM, 具体如下:

```

MultiPointM
{
    Double[4]      Box      // Bounding Box
    Integer        NumPoints // Number of Points
    Point[NumPoints] Points // The Points in the set
    Double[2]      MRange   // Bounding Measure Range
    Double[NumPoints] MArray // Measures
}

```

MultiPointM 中各个字段描述如下:

- Box:** MultiPointM 的封装边界, 按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。
- NumPoints:** 点的总数。
- Points:** 长度为 NumPoints 的点数组。
- MRange:** MultiPointM 中 measure 的最小值和最大值, 按照 Mmin, Mmax 的顺序存储。
- MArray:** 长度为 NumPoints 的 measure 数组。

表 9 MultiPointM 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	28	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little
Byte X*	Mmin	Mmin	Double	1	Little
Byte X+8*	Mmax	Mmax	Double	1	Little
Byte X+16*	Marray	Marray	Double	NumPoints	Little
Note: $X = 40 + (16 * \text{NumPoints})$					
* optional					

PolyLineM

PolyLineM 由一个或几个 part 构成。一个 part 是由两个或两个以上的点连接而成的序列，Part 之间互相连接或不连接均可，Part 之间可以交叉也可以不交叉。

```

PolyLineM
{
    Double[4]           Box           // Bounding Box
    Integer             NumParts      // Number of Parts
    Integer             NumPoints     // Total Number of Points
    Integer[NumParts]   Parts         // Index to First Point in Part
    Point[NumPoints]    Points        // Points for All Parts
    Double[2]           MRange        // Bounding Measure Range
    Double[NumPoints]   MArray        // Measures for All Points
}

```

PolyLineM 中各个字段描述的如下：

- Box:** PolyLineM 的封装边界，按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。
- NumParts:** PolyLineM 中 part 的数目。
- NumPoints:** 构成所有 part 的点的总数。
- Parts:** 长度为 NumParts 的数组，存储着每个 part 的首点在 points 数组中的索引。数组索引从 0 开始。
- Points:** 长度为 NumPoints 的数组。构成 PolyLineM 的每个 Part 的点，是按照首尾相连的顺序存储的。组成序号为 2 的 Part 的点紧接着序号为 1 的 Part，依此类推。数组 Parts 中存储着每个 Part 的起点的数组索引。组成不同 Part 的点之间没有分隔符。
- MRange:** PolyLineM 中 measure 的最小值和最大值，按照 Mmin, Mmax 的顺序存储。
- MArray:** 长度为 NumPoints 的数组。PolyLineM 中每个 Part 的 measure 是按照首尾相

连的顺序存储的。第二个 Part 的 measure 紧接着第一个 Part 的 measure 存储，依此类推。数组 Parts 中存储着每个 Part 的起点的数组索引。在数组 Marray 中，不同的 part 之间没有分隔符。

表 10 PolyLineM 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	23	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y*	Mmin	Mmin	Double	1	Little
Byte Y + 8*	Mmax	Mmax	Double	1	Little
Byte Y + 16*	Marray	Marray	Double	NumPoints	Little
Note: $X = 44 + (4 * \text{NumParts})$, $Y = X + (16 * \text{NumPoints})$					
* optional					

PolygonM

PolygonM 由多个环组成。环是一个闭合的、非自相交的回路。注意这里的交叉指的是在 XY 空间中而不是在 XYM 空间中。一个 PolygonM 可以包含多个外部环，每个环都是 PolygonM 的一个 part。

PolygonM 的结构与 PolyLineM 的结构是一样的：

```
PolygonM
{
    Double[4]           Box           // Bounding Box
    Integer             NumParts      // Number of Parts
    Integer             NumPoints     // Total Number of Points
    Integer[NumParts]   Parts         // Index to First Point in Part
    Point[NumPoints]    Points        // Points for All Parts
    Double[2]           MRange        // Bounding Measure Range
    Double[NumPoints]   MArray        // Measures for All Points
}
```

PolygonM 中各个字段的描述如下：

- Box:** PolygonM 的封装边界，按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。
- NumParts:** PolygonM 中环的数目。
- NumPoints:** 构成所有环的点的总数。
- Parts:** 长度为 NumParts 的数组，存储着每个环的首点在 points 数组中的索引。数组索引从 0 开始。
- Points:** 长度为 NumPoints 的数组。构成 PolygonM 的每个环的点，是按照首尾相连

的顺序存储的。组成序号为 2 的环的点紧接着序号为 1 的环，依此类推。数组 Parts 中存储着每个环的起点的数组索引。组成不同环的点之间没有分隔符。

MRange: PolygonM 中 measure 的极小值和极大值，按照 Mmin, Mmax 的顺序存储。

MArray: 长度为 NumPoints 的数组。构成 PolyLineM 的每个环的 measure，是按照首尾相连的顺序存储的。第二个环的 measure 紧接着第一个环的 measure 存储，依此类推。数组 Parts 中存储着每个环的起始 measure 的数组索引。在数组 MRrray 中，不同的环之间没有分隔符。

关于 PolygonM，有以下几点需要重视：

- 环是闭合的（环中的第一个顶点和最后一个顶点**必须**是相同的）；
- 环在 points 数组中的次序是不重要的

表 11 PolygonM 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	25	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y*	Mmin	Mmin	Double	1	Little
Byte Y + 8*	Mmax	Mmax	Double	1	Little
Byte Y + 16*	Marray	Marray	Double	NumPoints	Little

Note: $X = 44 + (4 * \text{NumParts})$, $Y = X + (16 * \text{NumPoints})$
 * optional

Shape Type in X,Y,Z Space

在 XYZ 空间中的 Shape 类型

此类型有一个可选的坐标—M。注意，M 可以被赋值为“no data”（参见第 2 页的“数字类型”）。

PointZ

一个 PointZ 由三个双精度坐标 X、Y、Z，外加一个 measure 组成。

```
PointZ
{
    Double      X      // X coordinate
    Double      Y      // Y coordinate
    Double      Z      // Z coordinate
    Double      M      // Measure
}
```


表 12 PointZ 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	11	Integer	1	Little
Byte 4	X	X	Double	1	Little
Byte 12	Y	Y	Double	1	Little
Byte 20	Z	Z	Double	1	Little
Byte 28	Measure	M	Double	1	Little

MultiPointZ

MultiPointZ 由一组 PointZ 组成:

MultiPointZ

```
{
    Double[4]          Box          // Bounding Box
    Integer            NumPoints     // Number of Points
    Point[NumPoints]   Points       // The Points in the Set
    Double[2]          ZRange       // Bounding Z Range
    Double[NumPoints]  ZArray       // Z Values
    Double[2]          MRange       // Bounding Measure Range
    Double[NumPoints]  MArray       // Measures
}
```

Bounding box 按照 Xmin, Ymin, Xmax, Ymax 的顺序存储;

Bounding ZRange 按照 Zmin, Zmax 的顺序存储; Bounding MRange 按照 Mmin, Mmax 的顺序存储。

表 13 MultiPointZ 记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	18	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumPoints	NumPoints	Integer	1	Little
Byte 40	Points	Points	Point	NumPoints	Little
Byte X	Zmin	Zmin	Double	1	Little
Byte X+8	Zmax	Zmax	Double	1	Little
Byte X+16	Zarray	Zarray	Double	NumPoints	Little
Byte Y*	Mmin	Mmin	Double	1	Little
Byte Y+8*	Mmax	Mmax	Double	1	Little
Byte Y+16*	Marray	Marray	Double	NumPoints	Little
Note: X = 40 + (16 * NumPoints); Y = X + 16 + (8 * NumPoints)					
* optional					

PolyLineZ

一个 PolyLineZ 由一个或多个 part 构成。一个 part 是由两个或两个以上的点连接而成的序列, Part 之间互相连接或不连接均可, Part 之间可以交叉也可以不交叉。

```

PolyLineZ
{
    Double[4]          Box          // Bounding Box
    Integer            NumParts      // Number of Parts
    Integer            NumPoints     // Total Number of Points
    Integer[NumParts]  Parts         // Index to First Point in Part
    Point[NumPoints]   Points        // Points for All Parts
    Double[2]          ZRange        // Bounding ZRange
    Double[NumPoints]  ZArray        // Z Value for All Points
    Double[2]          MRange        // Bounding Measure Range
    Double[NumPoints]  MArray        // Measures
}

```

PolyLineZ 中各个字段的描述如下：

Box: PolyLineZ 的封装边界，按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。

NumParts: PolyLineZ 中 part 的数目。

NumPoints: 构成所有 part 的点的总数。

Parts: 长度为 NumParts 的数组，存储着每个 part 的首点在 points 数组中的索引。数组索引从 0 开始。

Points: 长度为 NumPoints 的数组。构成 PolyLineZ 的每个 Part 的点，是按照首尾相连的顺序存储的，第 2 个 Part 的点紧接着第 1 个 Part 的点，依此类推。数组 Parts 中存储着每个 Part 的起点的数组索引。在数组 points 中，各个 Part 之间没有分隔符。

ZRange: PolyLineZ 中 Z 的极小值和极大值，按照 Zmin, Zmax 的顺序存储。

ZArray: 长度为 NumPoints 的数组。PolyLineZ 中每个 Part 的 Z 值是按照首尾相连的顺序存储的。第二个 Part 的 Z 值紧接着第一个 Part 的 Z 值存储，依此类推。数组 Parts 中存储着每个 Part 的起始 Z 值的数组索引。在数组 ZArray 中，各个 Part 之间没有分隔符。

MRange: PolyLineZ 中 measure 的极小值和极大值，按照 Mmin, Mmax 的顺序存储。

MArray: 长度为 NumPoints 的数组。PolyLineZ 中每个 Part 的 measure 是按照首尾相连的顺序存储的。第二个 Part 的 measure 紧接着第一个 Part 的 measure 存储，依此类推。数组 Parts 中存储着每个 Part 的起始 measure 的数组索引。在数组 measure 中，各个 Part 之间没有分隔符。

表 14 PolyLineZ 的记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	13	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y + 8	Zmax	Zmax	Double	1	Little
Byte Y + 16	Zarray	Zarray	Double	NumPoints	Little
Byte Z*	Mmin	Mmin	Double	1	Little
Byte Z+8*	Mmax	Mmax	Double	1	Little
Byte Z+16*	Marray	Marray	Double	NumPoints	Little

Note: $X = 44 + (4 * \text{NumParts})$, $Y = X + (16 * \text{NumPoints})$, $Z = Y + 16 + (8 * \text{NumPoints})$
 * optional

PolygonZ

PolygonZ 由多个环组成。环是一个闭合的、非自相交的回路。一个 PolygonZ 可以包含多个外部环。每个环都是 PolygonZ 的一个 part。

PolygonZ 的结构与 PolyLineZ 的结构是相同的：

PolygonZ

```
{
    Double[4]           Box           // Bounding Box
    Integer             NumParts      // Number of Parts
    Integer             NumPoints     // Total Number of Points
    Integer[NumParts]   Parts         // Index to First Point in Part
    Point[NumPoints]    Points        // Points for All Parts
    Double[2]           ZRange        // Bounding ZRange
    Double[NumPoints]   ZArray        // Z Value for All Points
    Double[2]           MRange        // Bounding Measure Range
    Double[NumPoints]   MArray        // Measures
}
```

PolygonZ 中各个字段的描述如下：

- Box:** PolygonZ 的封装边界，按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。
- NumParts:** PolygonZ 中环的数目。
- NumPoints:** 构成所有环的点的总数。
- Parts:** 长度为 NumParts 的数组，存储着每个环的首点在 points 数组中的索引。数组索引从 0 开始。
- Points:** 长度为 NumPoints 的数组。构成 PolygonZ 的每个环的点，按照首尾相连的顺序存储。组成序号为 2 的环的点紧接着序号为 1 的环，依此类推。数组

Parts 中存储着每个环的起点的数组索引。在 points 数组中组成不同环的点之间没有分隔符。

ZRange: PolygonZ 中 Z 的极小值和极大值，按照 Zmin,Zmax 的顺序存储。

ZArray: 长度为 NumPoints 的数组。PolygonZ 中每个环的 Z 值是按照首尾相连的顺序存储的。第二个环的 Z 值紧接着第一个环的 Z 值存储，依此类推。数组 Parts 中存储着每个环的起始 Z 值的数组索引。在数组 ZArray 中，各个环之间没有分隔符。

MRange: PolygonZ 中 measure 的极小值和极大值，按照 Mmin, Mmax 的顺序存储。

MArray: 长度为 NumPoints 的数组。构成 PolyLineZ 的每个环的 measure，是按照首尾相连的顺序存储的。第二个环的 measure 紧接着第一个环的 measure 存储，依此类推。数组 Parts 中存储着每个环的起始 measure 的数组索引。在数组 MArray 中，构成不同的环的点之间没有分隔符。

关于 PolygonZ，有以下几点需要重视：

- 环是闭合的（环中的第一个顶点和最后一个顶点**必须**是相同的）；
- 环在 points 数组中的次序是不重要的。

表 15 PolygonZ 记录内容

Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	15	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y+8	Zmax	Zmax	Double	1	Little
Byte Y+16	Zarray	Zarray	Double	NumPoints	Little
Byte Z*	Mmin	Mmin	Double	1	Little
Byte Z+8*	Mmax	Mmax	Double	1	Little
Byte Z+16*	Marray	Marray	Double	NumPoints	Little
Note: $X = 44 + (4 * \text{NumParts})$, $Y = X + (16 * \text{NumPoints})$, $Z = Y + 16 + (8 * \text{NumPoints})$					
* optional					

MultiPatch

MultiPatch 由多个曲面组成，每个曲面描述一个表面。每个曲面是 MultiPatch 的一个 part，曲面的类型决定了顶点的排列顺序。MultiPatch 的 part 可以是以下类型：

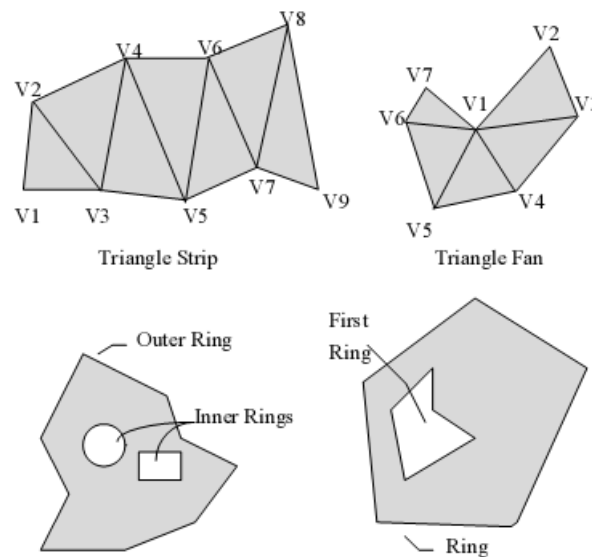
- **Triangle Strip** 相当于三角锁，除了前两个顶点外，每一个顶点都形成一个新的三角形，新的三角形总是由新顶点和它前面的两个原有顶点组成的。

- **Triangle Fan** 扇形三角，每一个顶点都形成一个新的三角形，新三角形总是由新顶点和它前面的点以及第一个顶点构成的。
- **Outer Ring** Polygon 的外环。
- **Inner Ring** Polygon 的内环。
- **First Ring** Polygon 中第一个未指定类型的环。
- **Ring** Polygon 中未指定类型的环。

一个单独的 Triangle Strip 或 Triangle Fan，描述一个单一的曲面。参见图 3。

一系列的环可以构成一个内部带岛的曲面，比较典型的是由一个外环描述曲面的外边界，内含几个内环用来描述岛。当一个内含岛的多边曲面由多个环构成，且其中的一个单独的环的类型无法确定，那么这些环必须以 First Ring 开始，然后是多个 Ring。一个环的序列如果没有以 First Ring 做为开始，则这些环被认为是没有岛的外环。

图 3 MultiPatch part 的样例



每个 part 的编码如下：

Value	Part Type
0	Triangle Strip
1	Triangle Fan
2	Outer Ring
3	Inner Ring
4	First Ring
5	Ring

```
MultiPatch
{
    Double[4]           Box           // Bounding Box
    Integer             NumParts       // Number of Parts
    Integer             NumPoints      // Total Number of Points
    Integer[NumParts]   Parts          // Index to First Point in Part
    Integer[NumParts]   PartsTypes     // Part Type
    Point[NumPoints]    Points         // Points for All Parts
    Double[2]           ZRange         // Bounding ZRange
    Double[NumPoints]   ZArray         // Z Value for All Points
    Double[2]           MRange         // Bounding Measure Range
    Double[NumPoints]   MArray         // Measures
}
```

MultiPatch 中各个字段的描述如下：

- Box:** MultiPatch 的封装边界，按照 Xmin, Ymin, Xmax, Ymax 的顺序存储。
- NumParts:** MultiPatch 中 part 的数目。
- NumPoints:** 构成所有 part 的点的总数。
- Parts:** 长度为 NumParts 的数组，存储着每个 part 的首点在 points 数组中的索引。数组索引从 0 开始。
- Points:** 长度为 NumPoints 的数组。构成 MultiPatch 的每个环的点，按照首尾相连的顺序存储。组成第二个 part 的点紧跟着第一个 part 的点，依此类推。数组 Parts 中存储着每个 part 的起点的数组索引。在 points 数组中组成不同 part 的点之间没有分隔符。
- ZRange:** arc 的 Z 值的极小值和极大值，按照 Zmin, Zmax 的顺序存储。
- ZArray:** 长度为 NumPoints 的数组。MultiPatch 中每个 part 的 Z 值是按照首尾相连的顺序存储的。第二个 part 的 Z 值紧接着第一个 part 的 Z 值存储，依此类推。数组 Parts 中存储着每个 part 的起始 Z 值的数组索引。在数组 ZArray 中，各个 part 之间没有分隔符。
- MRange:** MultiPatch 中 measure 的极小值和极大值，按照 Mmin, Mmax 的顺序存储。
- MArray:** 长度为 NumPoints 的数组。构成 MultiPatch 的每个环的 measure，是按照首尾相连的顺序存储的。第二个环的 measure 紧接着第一个环的 measure 存储，依此类推。数组 Parts 中存储着每个环的起始 measure 的数组索引。在数组 MArray 中，构成不同的环的点之间没有分隔符。

关于 MultiPatch，有以下几点需要注意：

- 如果 MultiPatch 的一个 part 是环，则这个环必须是闭合的（环中的第一个顶点和最后一个顶点**必须**是相同的）；
- 如果构成 MultiPatch 的 part 是环，则环在 points 数组中的次序是重要的：内环必须跟在外环后面；如果一个曲面由一个环序列描述，那么这些环必须以 First Ring 类型

的环作为起始环。

- 各个 part 可以共享公有边界，但彼此不能交叉和穿过（penetrate）。

表 16 MultiPatch 记录内容

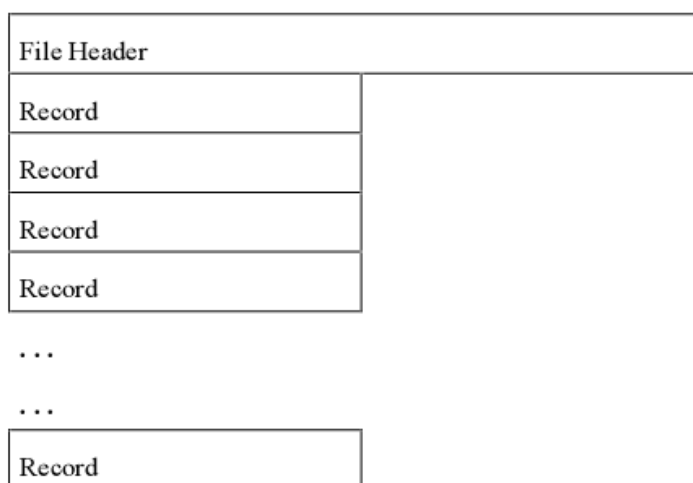
Position	Field	Value	Type	Number	Byte Order
Byte 0	Shape Type	31	Integer	1	Little
Byte 4	Box	Box	Double	4	Little
Byte 36	NumParts	NumParts	Integer	1	Little
Byte 40	NumPoints	NumPoints	Integer	1	Little
Byte 44	Parts	Parts	Integer	NumParts	Little
Byte W	PartTypes	PartTypes	Integer	NumParts	Little
Byte X	Points	Points	Point	NumPoints	Little
Byte Y	Zmin	Zmin	Double	1	Little
Byte Y+8	Zmax	Zmax	Double	1	Little
Byte Y+16	Zarray	Zarray	Double	NumPoints	Little
Byte Z*	Mmin	Mmin	Double	1	Little
Byte Z+8*	Mmax	Mmax	Double	1	Little
Byte Z+16*	Marray	Marray	Double	NumPoints	Little
Note: $W = 44 + (4 * \text{NumParts})$, $X = W + (4 * \text{NumParts})$, $Y = X + (16 * \text{NumPoints})$, $Z = Y + 16 + (8 * \text{NumPoints})$ * optional					

Organization of the Index File

索引文件的组织

索引文件(.shx)由一个长度为 100 字节的文件头引导，后面是一系列长度为 8 字节的记录。见图 4。

图 4 索引文件的组织



The Index File Header

索引文件文件头

索引文件文件头的组织形式与上面的主文件文件头的描述是一样的。文件头中存储的文件长度是以 16 字节的 word 表示的文件的总长度（文件头的 50 个 16 字节 word 加上记录个数的 4 倍）。

Index Records

索引记录

索引文件中的第 I 个记录存储着第 I 个记录在主文件中的偏移量和内容长度。表 17 中显示了文件头中的各个字段以及它们的位置、值、类型、字节顺序。表中的位置是从索引文件记录的开始算起的。

表 17 索引记录的描述

Position	Field	Value	Type	Byte Order
Byte 0	Offset	Offset	Integer	Big
Byte 4	Content Length	Content Length	Integer	Big

一个记录在主文件中的偏移量是用 16 字节的 word 来表示的，它表示从主文件开始至这个记录记录头第一个字节的 word 个数。因此，主文件中的第一个记录的偏移量是 50。

索引记录中存储的内容长度与主文件中记录头中存储的数值相同。

Organization of the dBASE File

dBASE 文件的组织

dBASE 文件(.dbf)中包含任何需要的要素属性或可供其它表连接的属性关键字。它是标准的 DBF 格式文件，广泛应用于诸多的 Windows 和 DOS 平台上基于表格的应用程序。各类字段都可被引入到表中。在这里要遵循四个（原文中为“三个”）条件：

- 文件名必须与 shape 文件和索引文件相同，扩展名必须为“.dbf”（参见第 2 页的“命名规则”）。
- 每个要素在表中必须要包含一个与之相对应的记录。
- 记录的顺序必需与要素在主文件中（*.shp）的顺序一样。
- dBASE 文件头中的年份值必须要晚于 1900 年。

更多的关于 dBASE 文件的格式，访问 INPRISE 公司，网站是 www.inprise.com。

Glossary

术语表

Key terms are defined below that will help you understand the concepts discussed in this document.

ARC/INFO

ARC/INFO software is designed for users who require a complete set of tools for processing and manipulating spatial data, including digitizing, editing, coordinate management, network analysis, surface modeling, and grid cell based modeling. ARC/INFO operates on a large variety of workstations and minicomputers. Using open standards and client/server architecture, ARC/INFO can act as a GIS server for ArcView clients.

ArcCAD

ArcCAD software brings the functionality of ARC/INFO GIS software to the AutoCAD environment, providing comprehensive data management, spatial analysis, and display tools.

ARC Macro Language (AML)

ARC Macro Language is a high-level, algorithmic language that provides full programming capabilities and a set of tools to the user interface of your application.

ArcView GIS

ArcView GIS software is a powerful, easy-to-use desktop GIS that gives you the power to visualize, explore, query, and analyze data spatially. ArcView GIS operates in both Windows desktop environments as well as a large variety of workstations.

Avenue

Avenue software is an object-oriented programming language and development environment created for use with ArcView GIS software. Avenue can be used to extend ArcView GIS software's basic capabilities and customize ArcView GIS for specific applications.

big endian byte order

Left-to-right byte ordering of an integer word. This byte-ordering method is used on many UNIX systems including Sun, Hewlett-Packard®, IBM®, and Data General AViiON®.

Bounding Box

A Bounding Box is a rectangle surrounding each shape (e.g., PolyLine) that is just large enough to contain the entire shape. It is defined as Xmin,Ymin, Xmax,Ymax.

Business MAP

BusinessMAP database mapping software for Windows allows you to create custom maps and

represent information in two- or three-dimensional charts. *BusinessMAP* read ESRI shapefiles and works with the leading contact managers, databases, and spreadsheets.

coverage

1. A digital version of a map forming the basic unit of vector data storage in ARC/INFO software. A coverage stores geographic features as primary features (such as arcs, nodes, polygons, and label points) and secondary features (such as tics, map extent, links, and annotation). Associated feature attribute tables describe and store attributes of the geographic features.
2. A set of thematically associated data considered as a unit. A coverage usually represents a single theme such as soils, streams, roads, or land use.

Date Automation Kit(DAK)

Date Automation Kit(DAK) complements ArcView GIS and other desktop mapping software by providing high-quality digitizing and data editing, topology creation, data conversion, and map projection capabilities.

feature

A representation of a geographic feature which has both a spatial representation referred to as a "shape" and a set of attributes.

index file

An ArcView GIS shapefile index file is a file that allows direct access to records in the corresponding main file.

little endian byte order

Right-to-left byte ordering of an integer word. This byte-ordering method is used on many operating file systems including DEC OSF/1TM, DEC OpenVMSTM, MS-DOS[®], and Windows NTTM.

MapObjects

MapObjects is a collection of embeddable mapping and GIS components including anActive X Control (OCX) and programmable Active X Automation objects. UseMapObjects with a variety of standard Windows development environments to buildmapping applications or add

mapping components into existing applications.

MultiPoint

A single feature composed of a cluster of point locations and a single attribute record. The group of points represents the geographic feature.

NumPoints

The count of the number of x,y vertices contained in a shape.

PC ARC/INFO

PC ARC/INFO is a full-featured GIS for PC compatibles. Like ARC/INFO software, PC ARC/INFO is used by organizations around the world for automating, managing, and analyzing geographic information. Attributes describing geographic features are stored as tabular files in dBASE format.

PolyLine

An ordered set of x,y vertices representing a line or boundary.

ring

An ordered set of x,y vertices where the first vertex is the same location as the last vertex; a closed PolyLine or a polygon.

shapefile

An ArcView data set used to represent a set of geographic features such as streets, hospital locations, trade areas, and ZIP Code boundaries. Shapefiles can represent point, line, or area features. Each feature in a shapefile represents a single geographic feature and its attributes.

Simple Macro Language (SML)

SML is PC ARC/INFO software's Simple Macro Language—a set of commands that constitute a simple programming language for building macros with some of the features of a high-level programming language such as expression evaluation, handling of input and output, and directing program flow of control.

theme

A user-defined set of geographic features. Data sources for themes in ArcView include coverages, grids, images, and shapefiles. Theme properties include the data source name, attributes of interest, a data classification scheme, and drawing methodology.

topology

The spatial relationships between connecting or adjacent coverage features (e.g., arcs, nodes, polygons, and points). For example, the topology of an arc includes its from- and to-nodes and its left and right polygons. Topological relationships are built from simple elements into complex elements: points (simplest elements) and arcs (sets of connected points) are used to represent more complex features such as areas (sets of connected arcs). Shapefiles do not explicitly record topology.

Coverages represent geographic features as topological line graphs. Topology can be useful for many GIS modeling operations that do not require coordinates. For example, to find an optimal path between two points requires a list of the arcs that connect to each other and the cost to traverse each arc in each direction. Coordinates are only needed for drawing the path after it is calculated.

vector

A Cartesian (i.e., x,y) coordinate-based data structure commonly used to represent geographic features. Each feature is represented as one or more vertices. Attributes are associated with the feature. Other data structures include raster (which associates attributes with a grid cell) and triangulated irregular networks (TINs) for surface representation.

vertex

One of a set of ordered x,y coordinates that constitutes a line.

附 部分词汇

这些词汇只是在本文中的译法，不能确保其准确性。括号中的释义出自金山词霸 2005[简明英汉词典]

Vertices: 顶点 (n.至高点, 头顶)

Corresponding: 相对应的 (adj.相应的, 通讯的)

Geometry: 几何数据 (n.几何学的)

Adhere: 遵从 (vi.粘附, 胶着, 坚持 v.坚持)

Byte Order: 字节状态

Bounding box: 边界框; 封装边界—Modeling our world

Field: 字段

Ring: 环

Inner/outer rings: 内部/外部环

Loop: 回路

Hole: 洞—应该是“岛”

Touch: 相接—Modeling our world

representation: 表示方法 (n.表示法, 表现, 陈述, 请求, 扮演, 画像, 继承, 代表)

clean: 干净的

segment: 片断

table-based: 基于表格的

Alphanumeric: (adj.文字数字的, 包括文字与数字的)

Infinity: (n.无限, 无穷大)

Nevertheless: (conj.然而, 不过 adv.仍然, 不过)

Concept: (n.观念, 概念)

Category : (n.种类, 别, [逻]范畴)

Restrict: (adj.(~ with) 严格的, 严厉的, 严谨的, 精确的)

Orthogonal: adj.直角的, 直交的

Potentially: adv.潜在地

Specify: vt.指定, 详细说明, 列入清单

place holder: n. 位置标志符 —金山在线翻译: 英汉计算机大词典

populate: vt.居住于(敏殖,填充)—金山在线翻译: 英汉计算机大词典

forbid: vt.禁止, 不许 v.禁止

consecutive: adj.连续的, 联贯的
degenerate: adj.退化的 v.退化 一不知确切含义
interior: adj.内部的, 内的 n.内部
significant: adj.有意义的, 重大的, 重要的
colinear: adj.共线的一金山在线翻译: 英汉计算机大词典
triplet: n.三个一幅, 三个一组, 三份
present: n.赠品, 礼物, 现在, 瞄准 adj.现在的, 出席的, 当面的 vt.介绍, 引见, 给, 赠送, 上演, 提出, 呈现 vi.举枪瞄准
surface patch: 曲面一金山在线翻译: 英汉计算机大词典
precede: v.领先(于), 在...之前, 先于

不清楚的:

but not along segments: 但不能有重合片断?

Has the inside of the polygon on the "correct" side of the line that defines it.;

The table must contain one record per shape feature.每个要素在表中必须要包含一个与之相对应的记录 (在表中必须要包含一个与每个要素相对应的记录?)。

The first ring of a polygon of an unspecify type. 面中第一个未指定类型的环?

the type of part controls how the order of vertices of an MultiPatch part is interpreted. 曲面的类型决定了顶点的排列顺序?

Byte Order:

high byte low byte

0x34 0x12

若此 WORD 值为 0x3412 的, 我们称之为 little-endian;

若为 0x1234 的, 我们称之为 big-endian。

MSDN 中的定义:

Byte Ordering Byte ordering Meaning

big-endian The most significant byte is on the left end of a word.

little-endian The most significant byte is on the right end of a word.

32 位整数的情况:

例如有一个 4 字节长整数:

Byte3 Byte2 Byte1 Byte0

Little Endian:

Base Address+0 Byte0

Base Address+1 Byte1

Base Address+2 Byte2

Base Address+3 Byte3

Big Endian:

Base Address+0 Byte3

Base Address+1 Byte2

Base Address+2 Byte1

Base Address+3 Byte0