# 第一章　X 资源的使用和说明

## 1·1　X 资源的使用和说明

Xlib 资源管理器为用户说明和操作其喜爱的对象(如几何形状、颜色、字体)提供了一个工具集。X 工具箱和 Xlib 的 XGetDefault 例程都提供了调用该工具集的简单编程界面。

资源(在 X 的老版本中也称为"默认值")只是＜名字，值＞对，它们被频繁地用于控制特定程序或子系统的外观或功能，并为用少量的工作剪裁整个应用集提供了一种方便的方法。

在 X 的前几个版本中，默认值被存放于每台机器上的每个用户起始目录的·Xdefaults 文件中，它除了需要复制默认值的副本外，不适用于条件说明(尤其是当用户既使用单色又使用彩色显示器时)。

在 X11 中，这些问题可以通过使用在 X 协议中的窗口特性机制，把资源存放于所有客户都可用的服务器来加以解决。因此，用户可根据所使用的具体显示器动态地规定默认值，这对在单色和彩色显示器上设置不同的默认值非常有用。此外，一种从命令行来说明资源的新约定业已建立(该约定被 X 工具箱的所有客户支持)，用户也可按需利用。

资源的说明形式通常是"程序·名字·子名字·等等：值"。每种资源可以在资源文件中作为一行说明，也可以在命令行中作为–xrm 的一个实参说明。这些名字具有层次结构，通常是对应着应用程序中的主结构(所指结构常常是像窗口、控制面板、菜单、滚动杆等等对象)。各种子名字被称为成份，并从左到右按通用到专用的顺序说明。

如果我们将 xmh 应用程序作为一个例子，我们可以看见其显示画面中称为"窗格"的段所构成，其中每段都依次包含着命令按钮。在"toc"(目录)窗格中的"include"按钮(用于检索新邮件)可按下列形式命名：

| 程序 | 窗格 | 对象组 | 子对象 |
|---|---|---|---|
| xmh | toc | buttons | include |

对象的指定全名(如此例中的"xmh·toc·buttons·include")被称为该对象的"实例"名，此外，每个成份属于一个可用"类"名加以说明的相似部件集(如所有的窗格集，所有的按钮集等等)。在上例中，如果我们假定 xmh 程序是几种"Xmh"程序的类型之一(就像"gnuemacs"和"microemacs"可被认为是"Emacs"程序的类的实例一样)，那么我们就可以构造下列的类名：

| 应用程序类型 | 顶级域类型 | 二级对象类型 | 三级对象类型 |
|---|---|---|---|
| Xmh | VPaned | Box | Cammand |

习惯上，实例名成份以小写字母打头，而类名成份则以大写字母打头。由多个字构成的成份，其第一个字后各个字要以大写字母打头，且将它们不带空格接续在一起。因此，图标像素映像的实例可以称之为"iconPixmap"，而图标像素映像的类应称之为"IconPixmap"。这里所提及的字母大写很重要，因为在同一说明中，资源名可能既包含实例名成份又包含类名成份(比如，"gnuemacs·VPaned·cursorColor: blue")。

对于给定对象的所有版本，类名允许有被指定的默认值，对给定的某个版本实例名则只容许有某个值，它取代其类值，并可用于说明类名规则的例外情况。例如，我们可用下列形式将以垂直窗格表示的按钮盒中的所有命令按钮(除"include"外)的背景颜色定义为蓝色；"include"的背景颜色定义为红色：

```
*VPaned.Box.Command.Background: blue
xmh.toc.buttons.include.background: red
```

此外，资源名的层次也可不必全部指明。在上例中，我们列出了每个成份，但相当麻烦。实际上，我们可以不必给出每个对象集(滑块杆，编辑窗或其它类型)的全部说明，只要用"*"号(通配符)来替代其中的某些中介成份及其"·"分隔符即可，一般而言，在你忘记某些中介成份或在层次结构中间插入新层次的情况下，使用"*"号替代这些成份及其"·"分隔符不失为一个好主意。例如，我们可以将上例缩写成如下形式：

```
Xmh*VPaned*Background: blue
xmh*toc.buttons.include.background: red
```

正确区别类和实例是十分重要的。例如许多正文应用程序都想在背景、前景、边框、游标和光标或标记配上某种颜色。通常是将背景置成某种颜色，所有其它属性置成另一种颜色，以便在单色显示器上可以看见这些属性。为了便于彩色显示器用户设置某种或所有属性，我们可将这些颜色组成如下几类：

| 实例名 | 类名 |
|---|---|
| background | Background |
| foreground | Foreground |
| borderColor | Foreground |
| pointerColor | Foreground |
| cursorColor | Foreground |

于是，若要使用两种颜色来构造应用程序运行于"单色"方式，只需使用下列两种说明：

```
obj*Background: blue
obj*Foreground: red
```

若你希望将光标的颜色置为黄色，而游标和边框保持与前景一样的颜色，只需加上一个新的资源说明：

```
obj*cursorColor: yellow
```

由于类名和实例名可以由大小写(case)来识别，因而两种类型名都可以用单一资源说明来给出。本手册的 Xlib 部分第 10 节给出了下列附加例子(注意："xmh"程序可以使用其它名字，此处仅作为示例)：

```
xmh*background: red
*command.font: 8×13
*command.background: blue
*Command.Foreground: green
xmh*toc*Command.activeForeground: black
```

其中，"xmh * toc * Command·activeForeground"资源层次结构，说明了在 xmh 应用程序的"toc"(目录)窗格中，所包含的 Command 类全部成份的某种特定颜色资源(这里是说明活动的前景颜色)。虽然这种形式功能很强，而对于资源管理器的众多复杂的客户、用文档笼统地说明全部容易，而如何单独地描述它们目前还是一个问题。人们的最终目标是，零件应像今天的命令一样文档化：每个零件资源应有实例名、类名和可容许值的描述。这样，应用文档只需要描述如何组成零件就行了。但在此之前，要得到说明资源的信息，只好去查寻如下资料：

1. 应用程序手册页
2. 应用所使用的零件之文档
3. 在/usr/lib/X11/app defaults/目录下的应用资源文件
4. 在应用程序或零件源中的 XtResource 表。

在 X11 版本中，你可以灵活地在各处定义默认值。资源管理器从下列地方获取资源的说明：

1. 通常存放在/usr/lib/x11/app defaults/目录下的专用资源文件。
2. 对用X工具箱编写的程序，由环境变量XAPPLRESDIR(默认值为 $ HOME)说明的目录下的专用资源文件。
3. 在0号屏幕的根窗口上的RESOURCE MANAGER特性；其资源说明用xrdb程序存放。如果该特性没有定义，则读取 $ HOME/·Xdefaults 文件，以便与 X10 兼容(虽然资源说明格式已作了某些改变)。
4. 存放在由环境变量XENVIRONMENT设置的某文件中的用户专用默认值。
5. -xrm 命令行选项(对用 X 工具箱编写的程序)。

资源通常是在启动 X 的 shell 文件中使用"xrdb"程序从一文件装入到 RESOURCE_MANAGER 特性的。使用 xdm 显示管理器的站点大多提供了自动装入用户指定的资源的能力。详见你的系统管理器。

我们在每台机器上都建立了一个名为"xsetup"的 shell 程序，它用于启动该机器上相

应的程序(如终端仿真器、宿主机的邮箱、时钟、负载均衡监视器等)。例如，我们在台式工作站上可以使用下列 shell 程序获取基本环境的工作情况：

```
#! /bin/sh
xrdb -loadHOME/.Xresources
stty erase'^?'
xmodmap-p $ HOME/.xmodmap
xset b 100 400 c 50 s 1800
uwm&
xclock -geometry 48×48-1+1 &
xload -geometry 48×48-1+100 &
```

其中，xrdb 程序从 $ HOME/.Xresources 文件中读取全局默认值。我们希望被每台机器上的客户使用的下列资源也定义在该文件中：

```
bitmap *Dashed: off
XTerm *multiScroll: on
XTerm *jumpScroll: on
XTerm *reverseWrap: on
XTerm *curses: on
XTerm *font: 6×10
emacs *Geometry: 80×65-0-0
emacs *Background: #5b7686
emacs *Foreground: white
emacs *Cursor: white
emacs *BorderColor: white
emacs *Font: 6×10
```

我们将机器专用的默认值(通常它使我们易于识别各台机器上的窗口颜色)放在 /.Xenv 文件中，并在.login 文件中设置 XENVIRONMENT 变量来指向该文件。例如，在一台台式工作站上我们可使用下列默认值：

```
XTerm *Background: black
XTerm *Foreground: green
XTern *BorderColor: white
xclock *analog: on
xclock *borderWhite: 0
xclock *padding: 2
xclock *Background: black
xclock *Foreground: red
xload *Background: black
```

```
xload *Foreground: cyan
```

为使相互一致，我们可在所用的每台机器上创建一个"xsetup shell"文件和一个·Xenv 文件(若我们想要机器专用默认值的话)，这样，在我们注册(login)后，就可以马上键入"xsetup"，开始正常的工作。继续上述示例，我们可把服务器上使用的 xsetup 文件描述如下:

```
#! /bin/sh
xterm -geometry 80×55+0+0&
xterm -geometry 80×65+488+1&
xterm -geometry 80×20+0-0&
xload = 48×48-1+150&
xbiff -rv=48×48-1+50&
```

其对应的·Xenv 文件为:

```
XTerm *Background:  #COO
XTerm *BorderColor: white
tinyxterm *Font:    3×5
tinyxtern *Geometry:80×24
bigxterm *Font:     9×15
bigxterm *Geometry: 80×55
xload *Background:  black
xload *Foreground:  red
xbiff *borderWidth: 0
xbiff *Background:  white
xbiff *Foreground:  blue
xbiff *reverseVideo:on
```

**注意**: 使用可替代的选择实例名"tinyxterm"和"bigxterm"很重要，它使我们只要通过调用"xterm—name tinyxterm"或"xterm—name bigxterm"，就可快速获得 xterm 的一个不同的"品味"。

xrdb 程序使用 C 预处理程序来提供基于所用服务器的配置条件，这对你从同一帐号使用不同类型的服务器(如单色和彩色)很有用。详见 xrdb(1)手册页。

### 1·1·1 更多的信息

从如下资料可以查看更多的有关信息:

- X(1)
- xrdb(1)
- Xlib的第10节

- XGetDefaults(3X)
- X 工具箱应用手册页的第4节

# 第二章　X 的逻辑字体描述约定 V1.3

## 2.1　引言

XWIN 客户应用程序必须能够移植到具有不同文件系统、命名约定和字体库的各个服务器上，这已成为一种实际需求。但是，由 X 窗口系统协议版本 11 所定义的字体访问请求既没有对字体名规定与服务器无关的约定，也没有为逻辑描述印刷字体提供适当的字体特征。

XWIN 客户必须能动态地确定在任一给定服务器上可用的字体是什么字号、风格等。只有这样，才能向用户呈现可理解的信息，或者选择合理的字体回溯方法。人们希望大多数公共查询，可以在免于为打开每个字体并对该字体特征进行检查而付出开销的情况下就能完成，即通过简单的 ListFonts 请求来完成[例如，若某用户选择了 Helvetica+活字体族，那么客户应用程序就应能向服务器查询所有的 Helvetica 字体并只呈现该字体族的 setwidths(活字宽度)、weights(笔划粗细)、slants(倾斜度)、pointsizes(点大小)和可用的字符集]。

本文档给出了一个标准逻辑字体描述(XLFD)和在 X 协议中要使用的约定，使客户可以在所有 X 服务器之间以一致的方式访问屏幕活字函数库。除了用字体的 FontName 来完整地说明一个给定字体外，XLFD 还提供了更详细描述该字体的一个标准的 FontProperties 基本集(字体特征集)。

当处理用某些未知字体创建于一些外来服务器上的文档时，XLFD 为出版和其它应用进行合理的字体匹配或替代，提供一个适当的印刷字体特征集(如 CAP_HEIGHT, X_HEIGHT 和 RELATIVE_SETWIDTH 等)。此外，某些客户也需要该信息来自动放置下标及确定最小的大写字母高度、建议的标题、字间间隔值等等。

文中的示例仅用作解释性说明。

## 2.2　需求和目标

本说明符合具有实现以下各项的标准逻辑字体描述的短期和长远目标：

- 提供支持简单模式匹配的唯一描述字体名
- 支持多个字体销售商、任意字符集及编码
- 与X服务器和操作系统/文件系统无关
- 为任意复杂的字体匹配/替代提供足以描述字体的信息
- 可扩展性

### 2·2·1 唯一描述字体名

为了保证字体名在合理的概率范围内的唯一性，以免意造出新的登记组织，应使字体名足够长并进行详尽描述。字体名必须独自地预先考虑和处理有关分辨率/与字号有关的主模、多个销售商的字体库等等。

名字本身应结构化成服从简单的模式匹配和语法分析，并允许 XWIN 客户限制向服务器上所有可能字体中的某一子集字体的查询。

### 2·2·2 支持多个销售商和字符集

字体名和字体特征应能识别出不同字体销售商所提供的字体，但可以共用相同的名字。我们期望有一个高度竞争的字体市场，这样用户就可以根据他们的具体需求从众多字体中选购所需的字体。

许多销售商销售每种字体时，都连同为该字体设计的所有象形文字符(glyphs)一起向用户提供，而其中的字符集映像则由编码向量规定。一些服务器实现可能迫使这些字符集映像静态地映射到字体数据中的专有或标准字符集上，而另一些服务器实现又可能希望在服务器中动态地完成映射。在支持多个字符集的服务器环境中，对允许某字体请求去说明/标识指定字符集映射的字体名，必须提供一些相应措施。

### 2·2·3 与服务器、操作系统及文件系统无关

要求一特定字体的 XWIN 客户程序应能够使用描述名，而不需知道服务器所用文件系统或其它存放处。但服务器可以将给定的字体名翻译成文件名语法形式，这种形式知道如何处理才不致于影响字体名唯一性。此翻译算法应是可逆的(如何做此翻译完全依赖于具体的实现)。

### 2·2·4 支持任意复杂的字体匹配/替代

除了字体名外，XLFD 还应定义具有对所有字体都一致的回溯方法的描述字体特征的一张标准表。这样，当要求客户应用程序处理所需的不知名字体时，它才能获得字体专用的格式/显示数据，并完成字体匹配/替代任务。

### 2·2·5 可扩展性

XLFD 必须具有可扩展性，这样才能向相依从的字体添加新的和/或私有的描述字体特征，而不必废弃现存的 XWIN 客户或服务器的具体实现。

## 2.3 X 的逻辑字体描述

XLFD 由 FontName(字体名)和一个可选用的 FontProperties(字体特征)变量表这两个基本成份组成。其中 FontName 给出了唯一识别 X 协议请求(如 OpenFont，ListFonts 等等)中的字体所需要的所有字体信息;FontProperties(字体特征)变量表则更详细地描述一个字体。FontName 用于字体查询并在某些 X 协议请求中作为数据返回；在 X 社团的字符位映像发布格式说明(BDFV2.1)中，FontName 也被说明为 FONT 项的数据值。

FontProperties 应用于逐个字体，并作为 XFontStruct 数据结构中的一部分，在某些 X 协议请求中以数据返回。FontProperty 名及其有关数据值还可以作为 BDF2.1 说明中的 STARTPROPERTIES…ENDPROPERTIES 表的各项出现。

### 2.3.1 FontName(字体名)

在逻辑上，FontName 由两个字符串构成，一个是前缀 FontNameRegistry，另一个是跟在其后的 FontNameSuffix。FontNameSuffix 是一个 X 登记名，它用于识别拥有规定的 FontNameSuffix 语法和语义的登记权限。

依从于此说明的所有字体名将使用一个定义成连字符"-"串的 FontNameRegistry 前缀，即 ISO8859-1 HYPHEN(行/列 02/13)。所有形为"+version-"的 FontNameRegistry 前缀由 X 社团保留作为将来扩充 XLFD 字体名之用，其中 Version 是将来某些 XLFD 说明的版本。如果需要的话，对当前 XLFD 字体名的扩充可以通过在当前结构附加一新域而构成。每个新域用现有的域定界符分界，其它 FontNameRegistry 前缀的可用性或支持其它记录方式的字体的可用性与服务器的具体实现有关。

在 X 协议的说明中，FontName 要以字符串表示。因此，在名字中数字域的值应以等效字符串表示。所有的 FontNameSuffix 域也将定义成 FontProperties(字体特征)，其中的数字特性值则表示成相应的带符号或无符号整数。

#### 2.3.1.1 FontName(字体名)语法

FontName(字体名)是一个结构化的可解析字符串(X 数据类型 STRING8)，其巴科斯-诺尔范式语法描述如下:

**表 2-1: FontName(字体名)语法**

| | |
|---|---|
| FontName$_{2.2}$ = | XFontNameRegistry *XFontNamesuffix* \| |
| | PrivFontNameRegistry PrivFontNameSuffix |
| XFontNameRegistry$_{2.2}$ = | Delim \| XFNExtPrefix Version XFNDelim |
| XFontNameSuffix$_{2.2}$ = | FOUNDRY XFNDelim FAMILY_NAME XFNDelim |
| | WEIGHT_NAME XFNDelim SLANT XFNDelim |

| | |
|---|---|
| | SETWIDTH_NAME XFNDelim ADD_STYLE_NAME |
| | XFNDelim PIXEL_SIZE XFNDelim POINT_SIZE |
| | XFNDelim RESOLUTION_X XFNDelim RESOLUTION_Y |
| | XFNDelim SPACING XFNDelim AVERAGE_WIDTH |
| | XFNDelim CHARSET_REGISTRY XFNDelim |
| | CHARSET_ENCODING |
| Version$_{12}$ = | STRING8–the XLFD version defining an extension to font name syntax, e.g. "2.0" |
| XFNExtPrefix$_{12}$ = | OCTET–the value of ISO8859–1 PLUS (Column/Row 02/13) |
| XFNDelim$_{12}$ = | OCTET–the value of ISO8859–1 HYPHEN (Column /Row 02/13) |
| PrivFontNameRegistry$_{12}$ = | STRING8–other than those strings reserved by XLFD |
| PrivFontNameSuffix$_{12}$ = | STRING8 |

除下列情况外，所有域值都按 ISO8859–1 图形字符串构成：

- HYPHEN(02/13)，XLFD字体名定界符字符。
- QUESTIONMARK(03/15)和ASTERISK(02/10)，X协议字体名通配符(wildcard)字符。

对字母大小写区别是允许的，这只关系到人们的可读性。X 服务器将完成与大小写无关的字体名查询/打开请求的匹配。整个字体名字符串不得多于 255 个字符。我们建议客户在构造字体名查询模式时，要显式地包含所有的域定界符，以避免不可预料的结果。值得注意的是：SPACE(空格)是 FontName 域的一个有效字符。例如，FAMILY_NAME 可以是 ITC Avant Garde Gothic 这种格式。

### 2·3·1·2 字体名(FontName)域定义

**FOUNDRY x登记**

FOUNDRY 是一个 x 登记名，它可以是数字化的并提供字体数据的数字型活字车间(foundry)之名字或标识符;或者是上一次修改字体形状或字体尺寸信息的组织之标识符。

这种区别是必要的，因为给定字体设计的许可权可能来自一个源头(如 ITC)，而其数字化和定型则可能由许多不同活字的供应商提供。最初设计的每个数字版本在尺寸和形状上一般都与"理想"的最初字体数据有些区别，这一方面是由于每个字体活字车间好坏都有它自己的标准和受制约于当时输出技术所能得到的实际活字字样；另一方面是由于这些车间自己察觉到市场的需要所进行的调整。

活字供应商应根据 X 社团规定的登记过程，向 X 社团在 FontName 域登记一个适当的名字。

X 社团应规定登记活字车间名字的过程，并维护和及时公布已登记名字的记录，以便用于 XLFD 字体名和字体特征。

**FAMILY_NAME 字符串**

FAMILY_NAME 是一个字符串，它用于标识活字字样设计的范围或"族"，这些字样设计范围就成为一个基本印刷风格的所有变种。FAMILY_NAME 必须使用全拼写，需要时以空格分隔各个字，它的名字必须是人们易于理解的并表示为宜于字体用户去标识活字字样族。

供给和维护宜于此域和字体特征的字符串、将恰当的合法称号捆在一给定名字上以及防止侵犯他人的版权或商标等工作都取决于活字供应商。按约定，不翻译 FAMILY_NAME，如果考虑活字字样族名作为一个有效的部分，FAMILY_NAME 可以包括一个设计所有权的标记(参见下面示例)。

FAMILY_NAME 的示例为:

```
Helvetica+
ITC Avant Garde Gothic
Times
Times Roman
Bitstream Amerigo
Stone
```

**WEIGHT_NAME 字符串**

WEIGHT_NAME 是一个根据 FOUNDRY 的判断而识别字体的印刷笔划粗细(即字体的标准黑色)的字符串，它的名字必须是人们易于理解的并适于向字体用户表示的。

对此域的解释尚存在一些问题，因为对印刷笔划粗细的判断传统上依赖于上述活字字样族的整个设计[即某字体的 DemiBold(半黑体)笔划粗细可能在印刷感受上几乎等价于另一族的黑体字体]。

因为 WEIGHT_NAME 是人们易于理解的字体全名的一个重要组成部分，所以把它作为一个任意字符串来捕获。但它不应用于字体匹配/替代。XWIN 客户应用程序应使用相对笔划粗细的字体特征(RELATIVE_WEIGHT&WEIGHT)。这两个特征为这一用途相应地给出了已编码的相对笔划粗细和计算的笔划粗细。

**SLANT(倾斜度)代码串**

SLANT 是一个代码串。它表示用于字体中的活字字样设计的整个姿态。

SLANT 的编码如下:

表 2-2: 倾斜度编码

| 代码 | 英文 | 描述 |
|---|---|---|
| "R" | Roman | 正体设计 |
| "I" | Italic | 斜体设计，对垂直线按顺时针方向倾斜 |
| "O" | Oblique | 倾斜了的正体设计，对垂直线按顺时针方向倾斜 |
| "RI" | Reverse Italic | 斜体设计，对垂直线按逆时针方向倾斜。 |
| "RO" | Reverse Oblique | 倾斜了的正体设计，对垂直线按逆时针方向倾斜。 |
| "OT" | Other | 其它 |

SLANT 代码仅仅是为了方便编程，因而在把它们交给用户之前，通常是将其转化成人们易于理解的等价形式。

**SETWIDTH_NAME 字符串**

SETWIDTH_NAME 是一个字符串，按照 FOUNDRY 的判断，它给出字体的印刷比例宽度(即字体的每个水平单元的标称宽度)。像 WEIGHT_NAME 一样，对此域或字体特征的解释尚存在一些问题，因为设计者对设置宽度的判断传统上依赖于上述活字字样族的整个设计。所以，一般 XWIN 客户应用程序应使用 RELATIVE_SETWIDTH 字体特征、该特征为字体匹配或替代给出了相对的编码比例宽度，或计算出此比例宽度。

SETWIDTH_NAME 的示例为:

```
Normal
Condensed
Narrow
Double Wide
```

**ADD_STYLE_NAME 字符串**

ADD_STYLE_NAME 是一个字符串，它识别不被其它域捕获的附加印刷风格的信息。我们需要它来唯一地识别字体。

ADD_STYLE_NAME 不是一个活字字样分类域，且仅用于唯一性识别，它的使用不限于对印刷风格的区分。

下面是 ADD_STYLE_NAME 的示例:

```
Serif
Sans Serif
Informal
Decorated
```

**PIXEL_SIZE 整型串**

PIXEL_SIZE 是一个无符号整型串的印刷字体度量尺寸(以设备像素表示)，它针对特定的 POINT_SIZE 和 RESOLUTION_Y 给出了字体活字身的大小，PIXEL_SIZE 通常与附加垂直空隙一起考虑为字体设计的一个部分(不过要注意的是，该值不必等于字体边框盒的高度)。PIXEL_SIZE 在 0～∞范围内取值。

PIXEL_SIZE 通常应由这样的 XWIN 客户应用程序使用，这些程序需要根据相关设备大小来询问字体，但不关心为字体设计的点的大小或垂直分辨率。

**POINT_SIZE 整型串**

POINT_SIZE 是一个无符号整型串的印刷字体度量尺寸(以设备无关单位表示)，它给出了为字体设计的字体活字的大小。通常，它与附加垂直空隙一起考虑为字体设计的一个部分(不过要注意的是，POINT_SIZE 不必等于字体边框盒的高度)。POINT_SIZE 用带小数点的十进制数表示(其中点的数目在 X 协议中定义，或 72.27 点＝1 英寸)，其取值范围是 0～∞。

POINT_SIZE 和 RESOLUTION_Y 应由 XWIN 客户应用程序使用，以根据与设备无关大小来询问字体，来保持屏幕上的正文大小不变，而不管用于该字体的 PIXEL_SIZE 如何。

**RESOLUTION_X(X 分辨率)整型串和 RESOLUTION_Y(Y 分辨率)整型串**

RESOLUTION_X 和 RESLOUTION_Y 都是无符号整型串，是为字体设计而定义的水平和垂直分辨率，并用每英寸上的像素/点数表示(dpi)。由于必须为带各种不同比率(如 1:1，4:3，2:1 等等)的显示器设计出各自的位映像字体，因此需要水平和垂直值。

由于 X 允许具有极不同视屏特性(水平和垂直分辨率，屏幕和像素大小，像素形状等等)的服务器随意访问相同字体库，因此分开像素/点大小和分辨率是必需的。这样，为 75dpi 设计 14 点字体(字体大约为 14 个像素大小)和为 150dpi 设计的 14 点字体(字体大约为 28 个像素大小)的字体名必须有所区别。此外，在实现连续缩放部分或全部字体外形的服务器上，POINT_SIZE 和 RESOLUTION_Y 会帮助服务器区分正文、标题及显示尺寸或其它印刷情况的各种字体主模。

**SPACING 代码串**

SPACING 是一个表示字体的转义类代码串，即是为单一间隔(固定间距)、按比例调整间隔(可变间距)或字符单元(一个符合传统数据处理字符单元字体模型的专用单一间隔字体)。

**表 2-3: 间隔编码**

| 代码 | 英文串 | 描述 |
|---|---|---|
| "P" | Proportional | 每个象形文字符的逻辑字符宽度都不同的字体。注意：按比例调整的 |

| | | |
|---|---|---|
| | | 字体尺寸不受其它限制。 |
| "M" | Monospace | 逻辑字符宽度不变的字体(即字体的所有字符宽度都等于max_bounds.width)。单一间隔的字体尺寸不受其它限制; |
| "C" | CharCell | 符合标准打字机字符单元模式的单一间隔字体，即字体的象形文字符可以由 XWIN 客户程序模型化成相同宽度和高度的"盒子"，这些"盒子"从左到右并排映象而形成正文串，或从顶到底映象形成正文行。按定义，所有的象形文字符都有相同的逻辑字符宽度，且在字符单元外它们没有"跳点"..没能颗粒状斑点(即对每个字符而言，带正字体尺寸时：0<=left_bearing<=right_bearing<=width;带负字体尺寸时：width<=left_bearing<=right_bearing<=0)而且字体在垂直方向的扩展没有超过垂直空隙(即对每个字符而言：ascent<=font_ascent& descent<=font_descent)。单元的高度=font_descent+font_ascent;且宽度=AVERAGE_WIDTH。 |

**AVERAGE_WIDTH 整型串**

AVERAGE_WIDTH 是一个无符号整型串的印刷字体度量尺寸值，它给出字体中所有象形文字符的无笔划粗细算术平均宽度，(以 1/10 个像素为单位)。注意：对单一间隔字体和字符单元字体，这是字体中所有象形文字符的宽度。

**CHARSET_REGISTRYx_登记名**

**CHARSET_ENCODING 登记名**

一种字符集，它用于对 X 社团字符集记录所维护的字体的象形文字符(并隐含着该字体的象形文字符的成熟古体)进行编码。CHARSET_REGISTRY 是一个 x_登记名，它识别拥有指定编码的登记权限;CHARSET_ENCODING 也是一个登记名，它识别由该登记权限所定义的编码字符集。

虽然 X 协议没有明确地提供关于字符集编码的知识，但出于性能和方便性的原因，服务器实现者更希望把某些所有权知识或工业标准字符集嵌入它们的字体库中。CHARSET_REGISTRY 和 CHARSET_ENCODING 域/特性，允许 XWIN 客户的一个字体请求去指定在支持多个字符集的服务器环境中的一个专用字符集映象。任何具体字符集的可获用性与字体和服务器的实现有关。

为了避免在定义字符集名时发生冲突，我们建议 CHARSET_REGISTRY/CHARSET_ENCODING 名对应按下列约定构成:

**表 2-4：字符集名语法**

| | |
|---|---|
| CharsetRegistry::= | StdCharsetRegistryName \| PrivCharsetRegistryName |
| CharsetEncoding::= | StdCharsetEncodingName \| PrivCharsetEncodingName |
| StdCharsetRegistryName::= | StdOrganizationId StdNumber \| StdOrganizationId StdNumber Dot Year |
| PrivCharsetRegistryName::= | OrganizationId STRING8 |
| StdCharsetEncodingName::= | STRING8–numeric part # of referenced standard |
| PrivCharsetEncodingName::= | STRING8 |
| StdOrganizationId::= | STRING8–the registered name or acronym of the referenced standard organization |
| StdNumber::= | STRING8–referenced standard number |
| OrganizationId::= | STRING8–the registered name or acronym of the organization |
| Dot::= | "."–ISO 8859–1 FULL STOP (Colmn/Row2/14) |
| Year::= | STRING8–numeric–year, e.g. 1989 |

X 社团应维护并及时公布这样的字符集名的记录，以便按 XLFD 的规定用于 X 协议的字体名和字体特征。

ISO Latin1 字符集应被 X 社团登记为 CHARSET–REGISTRY–CHARSET–ENCODING 值对："ISO8859–1"。

### 2.3.1.3 示例

下面的字体名示例由屏幕字体(随 R3 服务器销售)衍生出来。

**表 2-5：X 字体名示例**

| 字体 | X字体名 |
|---|---|
| Courier 8pt | Adobe–Courier–Medium–R–Normal–8–80–75–75–M–50–ISO8859–1 @75dpi |
| Courier 10pt | Adobe–Courier–Medium–R–Normal–10–100–75–75–M–60–ISO8859–1 |

| | |
|---|---|
| Courier 12pt | Adobe-Courier-Medium-R-Normal-12-120-75-75-M-70-ISO8859-1 |
| Courier 14pt | Adobe-Courier-Medium-R-Normal-14-140-75-75-M-90-ISO8859-1 |
| Courier 18pt | Adobe-Courier-Medium-R-Normal-18-180-75-75-M-110-ISO8859-1 |
| Courier 24pt | Adobe-Courier-Medium-R-Normal-24-240-75-75-M-60-ISO8859-1 |
| Courier Bold 10pt | Adobe-Courier-Bold-R-Normal-10-100-75-75-M-150-ISO8859-1 |
| Courier BoldOblique 10pt | Adobe-Courier-Bold-O-Normal-10-100-75-75-M-60-ISO8859-1 |
| Courier Oblique 10pt | Adobe-Courier-Medium-O-Normal-10-100-75-75-M-60-ISO8859-1 |
| Times Roman 10pt @100dpi | Adobe-Times-Medium-R-Normal-14-100-100-100-P-74-ISO8859-1 |
| Times Bold 10pt | Adobe-Times-Bold-R-Normal-14-100-100-100-P-76-ISO8859-1 |
| Times BoldItalic 10pt | Adobe-Times-Bold-I-Normal-14-100-100-100-P-77-ISO8859-1 |
| Times Italic 10pt | Adobe-Times-Medium-I-Normal-14-100-100-100-P-73-ISO8859-1 |
| Charter 12pt @75dpi | Bitstream-Charter-Medium-R-Normal-12-120-75-75-P-68-ISO8859-1 |
| Charter BoldItalic 12pt | Bitstream-Charter-Bold-I-Normal-12-120-75-75-P-75-ISO8859-1 |
| Charter Italic 12pt | Bitstream-Charter-Medium-I-Normal-12-120-75-75-P-66-ISO8859-1 |
| Charter Bold 12pt | Bitstream-Charter-Bold-R-Normal-12-120-75-75-P-76-ISO8859-1 |
| SYMBOL 8pt@100dpi | Adobe-Symbol-Medium-R-Normal-11-80-100-100-P-61-Adobe-FONTSPECIFIC |
| Symbol 10pt | Adobe-Symbol-Medium-R-Normal-14-100-100-100-P-85-Adobe-FONTSRECIFIC |
| Symbol 12pt | Adobe-Symbol-Medium-R-Normal-17-120-100-100-P-95-Adobe-FONTSPECIFIC |
| Symbol 14pt | Adobe-Symbol-Medium-R-Normal-20-140-100-100-P-107-Adobe-FONTSPECIFIC |
| Symbol 18pt | Adobe-Symbol-Medium-R-Normal-25-180-100-100-P-142-Adobe-FONTSPECIFIC |
| Symbol 24pt | Adobe-Symbol-Medium-R-Normal-34-240-100-100-P-191-Adobe-FONTSPECIFIC |

### 2.3.2 字体特征(FontProperties)

所有字体特征都是可选的，但它们一般都包括字体名域和其它有用的逐个字体描述/用法的信息。这种信息对灵巧地使用字体会有用处。对一种给定的字体，在没有定义其特征时，XLFD 将指定标准 X 字体特征的一个可扩展集，它们的解释和回溯规则。其目的是向客户应用程序提供足够的字体信息，以便对自动格式化/显示方式作出决定，得到满意的印刷品。

附加的标准 X 字体特征定义可在将来加以规定，而私有特征可能随时存在于 X 字体

中。私有字体特征应定义成符合于 X 协议所规定的通用机制，以避免重选的名字空间和二义的特征名。也就是说私有字体特征名应形如：

ISO8859-1 UNDERSCORE(Column/Row05/15)、即前面是组织的标识符，其后跟 UNDERSCORE，最后以特征名结束。

下面给出了 X 字体特征的巴科斯范式语法描述：

**表 2-6: X FontProperties 语法**

| | |
|---|---|
| Properties::= | OptFontPropList |
| OptFontPropList::= | NULL \| OptFontProp OptFontPropList |
| OptFontProp::= | PrivateFontProp \| XFontProp |
| PrivateFontProp::= | STRING8 \| Underscore OrganizationId Underscore STRING8 |
| XFontProp::= | FOUNDRY \|FAMILY_NAME \| WEIGHT_NAME \|SLANT \| SETWIDTH_NAME \| ADD_STYLE_NAME \| PIXEL_SIZE \| POINT_SIZE \| RESOLUTION_X \| RESOLUTION_Y \| SPACING \| AVERAGE_WIDTHCHARSET_REGISTRY \| CHARSET_ ENCODING QUAD_WIDTH \| RESOLUTION \| MIN_ SPACENORM_SPACE \| MAX_SPACE \| END_ SPACESUPERSCRIPT_X \| SUPERSCRIPT_Y SUBSCRIPT_X \| SUBSCRIPT_Y \| UNDERLINE_POSITION \| UNDERLINE_ THICKNESS STRIKEOUT_ASCENT \| SYRIKEOUT_DESCENT ITALIC_ANGLE \| X_HEIGHT \| WEIGHT \| FACE_NAME \| COPYRIGHT \| AVG_CAPITAL_WIDTH AVG_LOWERCASE_ WIDTH \| RELATIVE_SETWIDTH RELATIVE_WEIGHT \| CAP_HEIGHT \|SUPERSCRIPT_SIZE \| FIGURE_WIDTH \| SUBSCRIPT_SIZE \|SMALL_CAP_SIZE \| NOTICE DESTINAITON |
| Underscore::= | OCTET-the value of ISO8859-1 UNDERSCORE character (Column / Row 05/15) |
| OrganizationId::= | STRING8-the registered name or acronym of the organization |

### 2·3·2·1 FOUNDRY ATOM

除特征类型是 ATOM(原子)外，其它都与 FontName(字体名)定义的相同。如果 FOUNDRY 不作为一个字体特征提供，则不能计算或默认它。