Fontconfig Developers Reference, Version 2.4.2

Keith Packard
HP Cambridge Research Lab

Table of Contents

DESCRIPTION

Fontconfig is a library designed to provide system-wide font configuration, customization and application access.

FUNCTIONAL OVERVIEW

Fontconfig contains two essential modules, the configuration module which builds an internal configuration from XML files and the matching module which accepts font patterns and returns the nearest matching font.

FONT CONFIGURATION

The configuration module consists of the FcConfig datatype, libexpat and FcConfigParse which walks over an XML tree and ammends a configuration with data found within. From an external perspective, configuration of the library consists of generating a valid XML tree and feeding that to FcConfigParse. The only other mechanism provided to applications for changing the running configuration is to add fonts and directories to the list of application-provided font files.

The intent is to make font configurations relatively static, and shared by as many applications as possible. It is hoped that this will lead to more stable font selection when passing names from one application to another. XML was chosen as a configuration file format because it provides a format which is easy for external agents to edit while retaining the correct structure and syntax.

Font configuration is separate from font matching; applications needing to do their own matching can access the available fonts from the library and perform private matching. The intent is to permit applications to pick and choose appropriate functionality from the library instead of forcing them to choose between this library and a private configuration mechanism. The hope is that this will ensure that configuration of fonts for all applications can be centralized in one place. Centralizing font configuration will simplify and regularize font installation and customization.

FONT PROPERTIES

While font patterns may contain essentially any properties, there are some well known properties with associated types. Fontconfig uses some of these properties for font matching and font completion. Others are provided as a convenience for the applications rendering mechanism.

Property Definitions

Property	CPP Symbol	Type	Description
family	FC_FAMILY	String	Font family names
familylang	FC_FAMILYLANG	String	Language cooresponding to each family name
style	FC_STYLE	String	Font style. Overrides weight and slant
stylelang	FC_STYLELANG	String	Language cooresponding to each style name
fullname	FC_FULLNAME	String	Font face full name where different from family and family + style
fullnamelang	FC_FULLNAMELANG	String	Language cooresponding to each fullname
slant	FC_SLANT	Int	Italic, oblique or roman
weight	FC_WEIGHT	Int	Light, medium, demibold, bold or black

size	FC_SIZE	Double	Point size
width	FC_WIDTH	Int	Condensed, normal or expanded
aspect	FC_ASPECT	Double	Stretches glyphs horizontally before hinting
pixelsize	FC_PIXEL_SIZE	Double	Pixel size
spacing	FC_SPACING	Int	Proportional, dual-width, monospace or charcell
foundry	FC_FOUNDRY	String	Font foundry name
antialias	FC_ANTIALIAS	Bool	Whether glyphs can be antialiased
hinting	FC_HINTING	Bool	Whether the rasterizer should use hinting
hintstyle	FC_HINT_STYLE	Int	Automatic hinting style
verticallayout	FC_VERTICAL_LAYOUT	Bool	Use vertical layout
autohint	FC_AUTOHINT	Bool	Use autohinter instead of normal hinter
globaladvance	FC_GLOBAL_ADVANCE	Bool	Use font global advance data
file	FC_FILE	String	The filename holding the font
index	FC_INDEX	Int	The index of the font within the file
ftface	FC_FT_FACE	FT_Face	Use the specified FreeType face object
rasterizer	FC_RASTERIZER	String	Which rasterizer is in use
outline	FC_OUTLINE	Bool	Whether the glyphs are outlines
scalable	FC_SCALABLE	Bool	Whether glyphs can be scaled
scale	FC_SCALE	Double	Scale factor for point->pixel conversions
dpi	FC_DPI	Double	Target dots per inch
rgba	FC_RGBA	Int	unknown, rgb, bgr, vrgb, vbgr, none - subpixel geometry
minspace	FC_MINSPACE	Bool	Eliminate leading from line spacing
charset	FC_CHARSET	CharSet	Unicode chars encoded by the font
lang	FC_LANG	String	List of RFC-3066-style languages this font supports
fontversion	FC_FONTVERSION	Int	Version number of the font
capability	FC_CAPABILITY	String	List of layout capabilities in the font
embolden	FC_EMBOLDEN	Bool	Rasterizer should synthetically embolden the font

Datatypes

Fontconfig uses abstract datatypes to hide internal implementation details for most data structures. A few structures are exposed where appropriate.

FcChar8, FcChar16, FcChar32, FcBool

These are primitive datatypes; the FcChar* types hold precisely the number of bits stated (if supported by the C implementation). FcBool holds one of two CPP symbols: FcFalse or FcTrue.

FcMatrix

An FcMatrix holds an affine transformation, usually used to reshape glyphs. A small set of matrix operations are provided to manipulate these.

```
typedef struct _FcMatrix {
        double xx, xy, yx, yy;
} FcMatrix;
```

FcCharSet

An FcCharSet is an abstract type that holds the set of encoded unicode chars in a font. Operations to build and compare these sets are provided.

FcType

Tags the kind of data stored in an FcValue.

FcValue

An FcValue object holds a single value with one of a number of different types. The 'type' tag indicates which member is valid.

```
typedef struct _FcValue {
        FcType type;
        union {
                const FcChar8 *s;
                int i;
                FcBool b;
                double d;
                const FcMatrix *m;
                const FcCharSet *c;
        } u;
} FcValue;
```

FcValue Members

Type	Union member	Datatype
FcTypeVoid	(none)	(none)
FcTypeInteger	i	int
FcTypeDouble	d	double
FcTypeString	s	char *
FcTypeBool	b	b
FcTypeMatrix	m	FcMatrix *
FcTypeCharSet	c	FcCharSet *

FcPattern

holds a set of names with associated value lists; each name refers to a property of a font. FcPatterns are used as inputs to the matching code as well as holding information about specific fonts. Each property can hold one or more values; conventionally all of the same type, although the interface doesn't demand that.

FcFontSet

```
typedef struct _FcFontSet {
        int nfont;
        int sfont;
        FcPattern **fonts;
} FcFontSet;
```

An FcFontSet contains a list of FcPatterns. Internally fontconfig uses this data structure to hold sets of fonts. Externally, fontconfig returns the results of listing fonts in this format. 'nfont' holds the number of patterns in the 'fonts' array; 'sfont' is used to indicate the size of that array.

FcStrSet, FcStrList

FcStrSet holds a list of strings that can be appended to and enumerated. Its unique characteristic is that the enumeration works even while strings are appended during enumeration. FcStrList is used during enumeration to safely and correctly walk the list of strings even while that list is edited in the middle of enumeration.

FcObjectSet

```
typedef struct _FcObjectSet {
        int nobject;
        int sobject;
        const char **objects;
} FcObjectSet;
```

holds a set of names and is used to specify which fields from fonts are placed in the the list of returned patterns when listing fonts.

FcObjectType

```
typedef struct _FcObjectType {
        const char *object;
        FcType type;
} FcObjectType;
```

marks the type of a pattern element generated when parsing font names. Applications can add new object types so that font names may contain the new elements.

FcConstant

```
typedef struct _FcConstant {
    const FcChar8 *name;
    const char *object;
    int value;
} FcConstant;
```

Provides for symbolic constants for new pattern elements. When 'name' is seen in a font name, an 'object' element is created with value 'value'.

FcBlanks

holds a list of Unicode chars which are expected to be blank; unexpectedly blank chars are assumed to be invalid and are elided from the charset associated with the font.

FcFileCache

holds the per-user cache information for use while loading the font database. This is built automatically for the current configuration when that is loaded. Applications must always pass '0' when one is requested.

FcConfig

holds a complete configuration of the library; there is one default configuration, other can be constructed from XML data structures. All public entry points that need global data can take an optional FcConfig* argument; passing 0 uses the default configuration. FcConfig objects hold two sets of fonts, the first contains those specified by the configuration, the second set holds those added by the application at run-time. Interfaces that need to reference a particulat set use one of the FcSetName enumerated values.

FcSetName

Specifies one of the two sets of fonts available in a configuration; FcSetSystem for those fonts specified in the configuration and FcSetApplication which holds fonts provided by the application.

FcResult

Used as a return type for functions manipulating FcPattern objects.

```
FcResult Values
  Result Code             Meaning
  ------------------------------------------------------------
  FcResultMatch           Object exists with the specified ID
  FcResultNoMatch         Object doesn't exist at all
  FcResultTypeMismatch    Object exists, but the type doesn't match
  FcResultNoId            Object exists, but has fewer values
                          than specified
  FcResultOutOfMemory     Malloc failed
```

FcAtomic

Used for locking access to config files. Provides a safe way to update configuration files.

FUNCTIONS

These are grouped by functionality, often using the main datatype being manipulated.

Initialization

These functions provide some control over how the library is initialized.

FcInitLoadConfig

Name

`FcInitLoadConfig` — load configuration

Synopsis

```
#include <fontconfig.h>

FcConfig * FcInitLoadConfig(void);
```

Description

Loads the default configuration file and returns the resulting configuration. Does not load any font information.

Version

Fontconfig version 2.4.2

FcInitLoadConfigAndFonts

Name

`FcInitLoadConfigAndFonts` — load configuration and font data

Synopsis

```
#include <fontconfig.h>

FcConfig * FcInitLoadConfigAndFonts(void);
```

Description

Loads the default configuration file and builds information about the available fonts. Returns the resulting configuration.

Version

Fontconfig version 2.4.2

FcInit

Name

`FcInit` — initialize fontconfig library

Synopsis

```
#include <fontconfig.h>

FcBool FcInit(void);
```

Description

Loads the default configuration file and the fonts referenced therein and sets the default configuration to that result. Returns whether this process succeeded or not. If the default configuration has already been loaded, this routine does nothing and returns FcTrue.

Version

Fontconfig version 2.4.2

FcFini

Name

FcFini — finalize fonconfig library

Synopsis

```
#include <fontconfig.h>

void FcFini(void);
```

Description

Frees all data structures allocated by previous calls to fontconfig functions. Fontconfig returns to an uninitialized state, requiring a new call to one of the FcInit functions before any other fontconfig function may be called.

Version

Fontconfig version 2.4.2

FcGetVersion

Name

FcGetVersion — library version number

Synopsis

```
#include <fontconfig.h>

int FcGetVersion(void);
```

Description

Returns the version number of the library.

Version

Fontconfig version 2.4.2

FcInitReinitialize

Name

FcInitReinitialize — re-initialize library

Synopsis

```
#include <fontconfig.h>

FcBool FcInitReinitialize(void);
```

Description

Forces the default configuration file to be reloaded and resets the default configuration.

Version

Fontconfig version 2.4.2

FcInitBringUptoDate

Name

FcInitBringUptoDate — reload configuration files if needed

Synopsis

```
#include <fontconfig.h>

FcBool FcInitBringUptoDate(void);
```

Description

Checks the rescan interval in the default configuration, checking the configuration if the interval has passed and reloading the configuration if when any changes are detected.

Version

Fontconfig version 2.4.2

FcPattern

An FcPattern is an opaque type that holds both patterns to match against the available fonts, as well as the information about each font.

FcPatternCreate

Name

`FcPatternCreate` — Create a pattern

Synopsis

```
#include <fontconfig.h>

FcPattern * FcPatternCreate(void);
```

Description

Creates a pattern with no properties; used to build patterns from scratch.

Version

Fontconfig version 2.4.2

FcPatternDestroy

Name

`FcPatternDestroy` — Destroy a pattern

Synopsis

```
#include <fontconfig.h>
void FcPatternDestroy(FcPattern *p);
```

Description

Destroys a pattern, in the process destroying all related values.

Version

Fontconfig version 2.4.2

FcPatternEqual

Name

`FcPatternEqual` — Compare patterns

Synopsis

```
#include <fontconfig.h>
FcBool FcPatternEqual(const FcPattern *pa, const FcPattern *pb);
```

Description

Returns whether *pa* and *pb* are exactly alike.

Version

Fontconfig version 2.4.2

FcPatternEqualSubset

Name

`FcPatternEqualSubset` — Compare portions of patterns

Synopsis

```
#include <fontconfig.h>

FcBool FcPatternEqualSubset(const FcPattern *pa, const FcPattern *pb,
const FcObjectSet *os);
```

Description

Returns whether *pa* and *pb* have exactly the same values for all of the objects in *os*.

Version

Fontconfig version 2.4.2

FcPatternHash

Name

FcPatternHash — Compute a pattern hash value

Synopsis

```
#include <fontconfig.h>

FcChar32 FcPatternHash(const FcPattern *p);
```

Description

Returns a 32-bit number which is the same for any two patterns which are equal.

Version

Fontconfig version 2.4.2

FcPatternAdd

Name

FcPatternAdd — Add a value to a pattern

Synopsis

```
#include <fontconfig.h>

FcBool FcPatternAdd(FcPattern *p, const char *object, FcValue value,
FcBool append);
```

Description

Adds a single value to the list of values associated with the property named ‘object. *If ‘append* is FcTrue, the value is added at the end of any existing list, otherwise it is inserted at the begining. ‘value’ is saved (with FcValueSave) when inserted into the pattern so that the library retains no reference to any application-supplied data structure.

Version

Fontconfig version 2.4.2

FcPatternAddWeak

Name

FcPatternAddWeak — Add a value to a pattern with weak binding

Synopsis

```
#include <fontconfig.h>

FcBool FcPatternAddWeak(FcPattern *p, const char *object, FcValue
value, FcBool append);
```

Description

FcPatternAddWeak is essentially the same as FcPatternAdd except that any values added to the list have binding *weak* instead of *strong*.

Version

Fontconfig version 2.4.2

FcPatternAdd-Type

Name

FcPatternAddInteger, FcPatternAddDouble, FcPatternAddString, FcPatternAddMatrix, FcPatternAddCharSet, FcPatternAddBool — Add a typed value to a pattern

Synopsis

```
#include <fontconfig.h>

FcBool FcPatternAddInteger(FcPattern *p, const char *object, int i);
FcBool FcPatternAddDouble(FcPattern *p, const char *object, double d);
FcBool FcPatternAddString(FcPattern *p, const char *object, const char
*s);
FcBool FcPatternAddMatrix(FcPattern *p, const char *object, const
FcMatrix *m);
FcBool FcPatternAddCharSet(FcPattern *p, const char *object, const
FcCharSet *c);
FcBool FcPatternAddBool(FcPattern *p, const char *object, FcBool b);
```

Description

These are all convenience functions that insert objects of the specified type into the pattern. Use these in preference to FcPatternAdd as they will provide compile-time typechecking. These all append values to any existing list of values.

Version

Fontconfig version 2.4.2

FcPatternGet

Name

FcPatternGet — Return a value from a pattern

Synopsis

```
#include <fontconfig.h>

FcResult FcPatternGet(FcPattern *p, const char *object, int id,
FcValue *v);
```

Description

Returns in *v* the *id*'th value associated with the property *object*. The value returned is not a copy, but rather refers to the data stored within the pattern directly. Applications must not free this value.

Version

Fontconfig version 2.4.2

FcPatternGet-Type

Name

FcPatternGetInteger, FcPatternGetDouble, FcPatternGetString, FcPatternGetMatrix, FcPatternGetCharSet, FcPatternGetBool — Return a typed value from a pattern

Synopsis

```
#include <fontconfig.h>

FcResult FcPatternGetInteger(FcPattern *p, const char *object, int n,
int *i);
FcResult FcPatternGetDouble(FcPattern *p, const char *object, int n,
double *d);
FcResult FcPatternGetString(FcPattern *p, const char *object, int n,
char **consts);
FcResult FcPatternGetMatrix(FcPattern *p, const char *object, int n,
FcMatrix **s);
FcResult FcPatternGetCharSet(FcPattern *p, const char *object, int n,
FcCharSet **c);
FcResult FcPatternGetBool(FcPattern *p, const char *object, int n,
FcBool *b);
```

Description

These are convenience functions that call FcPatternGet and verify that the returned data is of the expected type. They return FcResultTypeMismatch if this is not the case. Note that these (like FcPatternGet) do not make a copy of any data structure referenced by the return value. Use these in preference to FcPatternGet to provide compile-time typechecking.

Version

Fontconfig version 2.4.2

FcPatternBuild

Name

FcPatternBuild, FcPatternVaBuild — Create patterns from arguments

Synopsis

```
#include <fontconfig.h>

FcPattern * FcPatternBuild(FcPattern *orig, ...);
FcPattern * FcPatternVaBuild(FcPattern *orig, va_list va);
```

Description

Builds a pattern using a list of objects, types and values. Each value to be entered in the pattern is specified with three arguments:

1. Object name, a string describing the property to be added.
2. Object type, one of the FcType enumerated values
3. Value, not an FcValue, but the raw type as passed to any of the FcPatternAdd<type> functions. Must match the type of the second argument.

The argument list is terminated by a null object name, no object type nor value need be passed for this. The values are added to 'pattern', if 'pattern' is null, a new pattern is created. In either case, the pattern is returned. Example

```
pattern = FcPatternBuild (0, FC_FAMILY, FcTypeString, "Times", (char *) 0);
```

FcPatternVaBuild is used when the arguments are already in the form of a varargs value.

Version

Fontconfig version 2.4.2

FcPatternDel

Name

FcPatternDel — Delete a property from a pattern

Synopsis

```
#include <fontconfig.h>

FcBool FcPatternDel(FcPattern *p, const char *object);
```

Description

Deletes all values associated with the property 'object', returning whether the property existed or not.

Version

Fontconfig version 2.4.2

FcPatternRemove

Name

FcPatternRemove — Remove one object of the specified type from the pattern

Synopsis

```
#include <fontconfig.h>

FcBool FcPatternRemove(FcPattern *p, const char *object, int id);
```

Description

Removes the value associated with the property 'object' at position 'id', returning whether the property existed and had a value at that position or not.

Version

Fontconfig version 2.4.2

FcPatternPrint

Name

FcPatternPrint — Print a pattern for debugging

Synopsis

```
#include <fontconfig.h>

void FcPatternPrint(const FcPattern *p);
```

Description

Prints an easily readable version of the pattern to stdout. There is no provision for reparsing data in this format, it's just for diagnostics and debugging.

Version

Fontconfig version 2.4.2

FcDefaultSubstitute

Name

`FcDefaultSubstitute` — Perform default substitutions in a pattern

Synopsis

```
#include <fontconfig.h>

void FcDefaultSubstitute(FcPattern *pattern);
```

Description

Supplies default values for underspecified font patterns:

- Patterns without a specified style or weight are set to Medium
- Patterns without a specified style or slant are set to Roman
- Patterns without a specified pixel size are given one computed from any specified point size (default 12), dpi (default 75) and scale (default 1).

Version

Fontconfig version 2.4.2

FcNameParse

Name

FcNameParse — Parse a pattern string

Synopsis

```
#include <fontconfig.h>

FcPattern * FcNameParse(const char *name);
```

Description

Converts *name* from the standard text format described above into a pattern.

Version

Fontconfig version 2.4.2

FcNameUnparse

Name

FcNameUnparse — Convert a pattern back into a string that can be parsed

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcNameUnparse(FcPattern *pat);
```

Description

Converts the given pattern into the standard text format described above. The return value is not static, but instead refers to newly allocated memory which should be freed by the caller.

Version

Fontconfig version 2.4.2

FcFontSet

An FcFontSet simply holds a list of patterns; these are used to return the results of listing available fonts.

FcFontSetCreate

Name

`FcFontSetCreate` — Create a font set

Synopsis

```
#include <fontconfig.h>

FcFontSet * FcFontSetCreate(void);
```

Description

Creates an empty font set.

Version

Fontconfig version 2.4.2

FcFontSetDestroy

Name

`FcFontSetDestroy` — Destroy a font set

Synopsis

```
#include <fontconfig.h>

void FcFontSetDestroy(FcFontSet *s);
```

Description

Destroys a font set. Note that this destroys any referenced patterns as well.

Version

Fontconfig version 2.4.2

FcFontSetAdd

Name

`FcFontSetAdd` — Add to a font set

Synopsis

```
#include <fontconfig.h>

FcBool FcFontSetAdd(FcFontSet *s, FcPattern *font);
```

Description

Adds a pattern to a font set. Note that the pattern is not copied before being inserted into the set.

Version

Fontconfig version 2.4.2

FcObjectSet

An FcObjectSet holds a list of pattern property names; it is used to indiciate which properties are to be returned in the patterns from FcFontList.

FcObjectSetCreate

Name

`FcObjectSetCreate` — Create an object set

Synopsis

```
#include <fontconfig.h>
FcObjectSet * FcObjectSetCreate(void);
```

Description

Creates an empty set.

Version

Fontconfig version 2.4.2

FcObjectSetAdd

Name

`FcObjectSetAdd` — Add to an object set

Synopsis

```
#include <fontconfig.h>
FcBool FcObjectSetAdd(FcObjectSet *os, const char *object);
```

Description

Adds a proprety name to the set.

Version

Fontconfig version 2.4.2

FcObjectSetDestroy

Name

`FcObjectSetDestroy` — Destroy an object set

Synopsis

```
#include <fontconfig.h>

void FcObjectSetDestroy(FcObjectSet *os);
```

Description

Destroys an object set.

Version

Fontconfig version 2.4.2

FcObjectSetBuild

Name

`FcObjectSetBuild, FcObjectSetVaBuild` — Build object set from args

Synopsis

```
#include <fontconfig.h>

FcObjectSet * FcObjectSetBuild(const char *first, ...);
FcObjectSet * FcObjectSetVaBuild(const char *first, va_list va);
```

Description

These build an object set from a null-terminated list of property names.

Version

Fontconfig version 2.4.2

FreeType specific functions

While the fontconfig library doesn't insist that FreeType be used as the rasterization mechanism for fonts, it does provide some convenience functions.

FcFreeTypeCharIndex

Name

FcFreeTypeCharIndex — map Unicode to glyph id

Synopsis

```
#include <fontconfig.h>
#include <fcfreetype.h>

FT_UInt FcFreeTypeCharIndex(FT_Face face, FcChar32 ucs4);
```

Description

Maps a Unicode char to a glyph index. This function uses information from several possible underlying encoding tables to work around broken fonts. As a result, this function isn't designed to be used in performance sensitive areas; results from this function are intended to be cached by higher level functions.

Version

Fontconfig version 2.4.2

FcFreeTypeCharSet

Name

FcFreeTypeCharSet — compute unicode coverage

Synopsis

```
#include <fontconfig.h>
#include <fcfreetype.h>

FcCharSet * FcFreeTypeCharSet(FT_Face face, FcBlanks *blanks);
```

Description

Scans a FreeType face and returns the set of encoded Unicode chars. This scans several encoding tables to build as complete a list as possible. If 'blanks' is not 0, the glyphs in the font are examined and any blank glyphs not in 'blanks' are not placed in the returned FcCharSet.

Version

Fontconfig version 2.4.2

FcFreeTypeQuery

Name

FcFreeTypeQuery — compute pattern from font file (and index)

Synopsis

```
#include <fontconfig.h>
#include <fcfreetype.h>

FcPattern * FcFreeTypeQuery(const char *file, int id, FcBlanks
*blanks, int *count);
```

Description

Constructs a pattern representing the 'id'th font in 'file'. The number of fonts in 'file' is returned in 'count'.

Version

Fontconfig version 2.4.2

FcFreeTypeQueryFace

Name

FcFreeTypeQueryFace — compute pattern from FT_Face

Synopsis

```
#include <fontconfig.h>
#include <fcfreetype.h>

FcPattern * FcFreeTypeQueryFace(const FT_Face face, const char *file,
int id, FcBlanks *blanks);
```

Description

Constructs a pattern representing 'face'. 'file' and 'id' are used solely as data for pattern elements (FC_FILE, FC_INDEX and sometimes FC_FAMILY).

Version

Fontconfig version 2.4.2

FcValue

FcValue is a structure containing a type tag and a union of all possible datatypes. The tag is an enum of type *FcType* and is intended to provide a measure of run-time typechecking, although that depends on careful programming.

FcValueDestroy

Name

`FcValueDestroy` — Free a value

Synopsis

```
#include <fontconfig.h>

void FcValueDestroy(FcValue v);
```

Description

Frees any memory referenced by `v`. Values of type FcTypeString, FcTypeMatrix and FcTypeCharSet reference memory, the other types do not.

Version

Fontconfig version 2.4.2

FcValueSave

Name

`FcValueSave` — Copy a value

Synopsis

```
#include <fontconfig.h>

FcValue FcValueSave(FcValue v);
```

Description

Returns a copy of *v* duplicating any object referenced by it so that *v* may be safely destroyed without harming the new value.

Version

Fontconfig version 2.4.2

FcCharSet

An FcCharSet is a boolean array indicating a set of unicode chars. Those associated with a font are marked constant and cannot be edited. FcCharSets may be reference counted internally to reduce memory consumption; this may be visible to applications as the result of FcCharSetCopy may return it's argument, and that CharSet may remain unmodifiable.

FcCharSetCreate

Name

`FcCharSetCreate` — Create an empty character set

Synopsis

```
#include <fontconfig.h>

FcCharSet * FcCharSetCreate(void);
```

Description

`FcCharSetCreate` allocates and initializes a new empty character set object.

Version

Fontconfig version 2.4.2

FcCharSetDestroy

Name

FcCharSetDestroy — Destroy a character set

Synopsis

```
#include <fontconfig.h>

void FcCharSetDestroy(FcCharSet *fcs);
```

Description

FcCharSetDestroy decrements the reference count *fcs*. If the reference count becomes zero, all memory referenced is freed.

Version

Fontconfig version 2.4.2

FcCharSetAddChar

Name

FcCharSetAddChar — Add a character to a charset

Synopsis

```
#include <fontconfig.h>

FcBool FcCharSetAddChar(FcCharSet *fcs, FcChar32 ucs4);
```

Description

FcCharSetAddChar adds a single unicode char to the set, returning FcFalse on failure, either as a result of a constant set or from running out of memory.

Version

Fontconfig version 2.4.2

FcCharSetCopy

Name

FcCharSetCopy — Copy a charset

Synopsis

```
#include <fontconfig.h>

FcCharSet * FcCharSetCopy(FcCharSet *src);
```

Description

Makes a copy of *src*; note that this may not actually do anything more than increment the reference count on *src*.

Version

Fontconfig version 2.4.2

FcCharSetEqual

Name

FcCharSetEqual — Compare two charsets

Synopsis

```
#include <fontconfig.h>

FcBool FcCharSetEqual(const FcCharSet *a, const FcCharSet *b);
```

Description

Returns whether *a* and *b* contain the same set of unicode chars.

Version

Fontconfig version 2.4.2

FcCharSetIntersect

Name

`FcCharSetIntersect` — Intersect charsets

Synopsis

```
#include <fontconfig.h>

FcCharSet * FcCharSetIntersect(const FcCharSet *a, const FcCharSet
*b);
```

Description

Returns a set including only those chars found in both *a* and *b*.

Version

Fontconfig version 2.4.2

FcCharSetUnion

Name

`FcCharSetUnion` — Add charsets

Synopsis

```
#include <fontconfig.h>

FcCharSet * FcCharSetUnion(const FcCharSet *a, const FcCharSet *b);
```

Description

Returns a set including only those chars found in either *a* or *b*.

Version

Fontconfig version 2.4.2

FcCharSetSubtract

Name

`FcCharSetSubtract` — Subtract charsets

Synopsis

```
#include <fontconfig.h>

FcCharSet * FcCharSetSubtract(const FcCharSet *a, const FcCharSet *b);
```

Description

Returns a set including only those chars found in *a* but not *b*.

Version

Fontconfig version 2.4.2

FcCharSetHasChar

Name

`FcCharSetHasChar` — Check a charset for a char

Synopsis

```
#include <fontconfig.h>

FcBool FcCharSetHasChar(const FcCharSet *fcs, FcChar32 ucs4);
```

Description

Returns whether *fcs* contains the char *ucs4*.

Version

Fontconfig version 2.4.2

FcCharSetCount

Name

FcCharSetCount — Count entries in a charset

Synopsis

```
#include <fontconfig.h>

FcChar32 FcCharSetCount(const FcCharSet *a);
```

Description

Returns the total number of unicode chars in *a*.

Version

Fontconfig version 2.4.2

FcCharSetIntersectCount

Name

FcCharSetIntersectCount — Intersect and count charsets

Synopsis

```
#include <fontconfig.h>

FcChar32 FcCharSetIntersectCount(const FcCharSet *a, const FcCharSet
*b);
```

Description

Returns the number of chars that are in both *a* and *b*.

Version

Fontconfig version 2.4.2

FcCharSetSubtractCount

Name

FcCharSetSubtractCount — Subtract and count charsets

Synopsis

```
#include <fontconfig.h>

FcChar32 FcCharSetSubtractCount(const FcCharSet *a, const FcCharSet
*b);
```

Description

Returns the number of chars that are in *a* but not in *b*.

Version

Fontconfig version 2.4.2

FcCharSetIsSubset

Name

FcCharSetIsSubset — Test for charset inclusion

Synopsis

```
#include <fontconfig.h>

FcBool FcCharSetIsSubset(const FcCharSet *a, const FcCharSet *b);
```

Description

Returns whether *a* is a subset of *b*.

Version

Fontconfig version 2.4.2

FcCharSetFirstPage

Name

FcCharSetFirstPage — Start enumerating charset contents

Synopsis

```
#include <fontconfig.h>

FcChar32 FcCharSetFirstPage(const FcCharSet *a,
FcChar32[FC_CHARSET_MAP_SIZE] map, FcChar32 *next);
```

Description

Builds an array of bits marking the first page of Unicode coverage of *a*. Returns the base of the array. *next* contains the next page in the font.

Version

Fontconfig version 2.4.2

FcCharSetNextPage

Name

FcCharSetNextPage — Continue enumerating charset contents

Synopsis

```
#include <fontconfig.h>

FcChar32 FcCharSetNextPage(const FcCharSet *a,
FcChar32[FC_CHARSET_MAP_SIZE] map, FcChar32 *next);
```

Description

Builds an array of bits marking the Unicode coverage of *a* for page **next*. Returns the base of the array. *next* contains the next page in the font.

Version

Fontconfig version 2.4.2

FcMatrix

FcMatrix structures hold an affine transformation in matrix form.

FcMatrixInit

Name

FcMatrixInit — initialize an FcMatrix structure

Synopsis

```
#include <fontconfig.h>

void FcMatrixInit(FcMatrix *matrix);
```

Description

FcMatrixInit initializes *matrix* to the identity matrix.

Version

Fontconfig version 2.4.2

FcMatrixCopy

Name

FcMatrixCopy — Copy a matrix

Synopsis

```
#include <fontconfig.h>

void FcMatrixCopy(const FcMatrix *matrix);
```

Description

FcMatrixCopy allocates a new FcMatrix and copies *mat* into it.

Version

Fontconfig version 2.4.2

FcMatrixEqual

Name

FcMatrixEqual — Compare two matrices

Synopsis

```
#include <fontconfig.h>

void FcMatrixEqual(const FcMatrix *matrix1, const FcMatrix *matrix2);
```

Description

FcMatrixEqual compares *matrix1* and *matrix2* returning FcTrue when they are equal and FcFalse when they are not.

Version

Fontconfig version 2.4.2

FcMatrixMultiply

Name

FcMatrixMultiply — Multiply matrices

Synopsis

```
#include <fontconfig.h>

void FcMatrixMultiply(FcMatrix *result, const FcMatrix *matrix1, const
FcMatrix *matrix2);
```

Description

FcMatrixMultiply multiplies *matrix1* and *matrix2* storing the result in *result*.

Version

Fontconfig version 2.4.2

FcMatrixRotate

Name

FcMatrixRotate — Rotate a matrix

Synopsis

```
#include <fontconfig.h>

void FcMatrixRotate(FcMatrix *matrix, double cos, double sin);
```

Description

FcMatrixRotate rotates *matrix* by the angle who's sine is *sin* and cosine is *cos*. This is done by multiplying by the matrix:

```
cos -sin
sin  cos
```

Version

Fontconfig version 2.4.2

FcMatrixScale

Name

FcMatrixScale — Scale a matrix

Synopsis

```
#include <fontconfig.h>

void FcMatrixScale(FcMatrix *matrix, double sx, double dy);
```

Description

FcMatrixScale multiplies *matrix* x values by *sx* and y values by *sy*. This is done by multiplying by the matrix:

```
sx  0
0   sy
```

Version

Fontconfig version 2.4.2

FcMatrixShear

Name

FcMatrixShear — Shear a matrix

Synopsis

```
#include <fontconfig.h>

void FcMatrixShear(FcMatrix *matrix, double sh, double sv);
```

Description

FcMatrixShare shears *matrix* horizontally by *sh* and vertically by *sv*. This is done by multiplying by the matrix:

```
1  sh
sv  1
```

Version

Fontconfig version 2.4.2

FcConfig

An FcConfig object holds the internal representation of a configuration. There is a default configuration which applications may use by passing 0 to any function using the data within an FcConfig.

FcConfigCreate

Name

`FcConfigCreate` — Create a configuration

Synopsis

```
#include <fontconfig.h>

FcConfig * FcConfigCreate(void);
```

Description

Creates an empty configuration.

Version

Fontconfig version 2.4.2

FcConfigDestroy

Name

`FcConfigDestroy` — Destroy a configuration

Synopsis

```
#include <fontconfig.h>

void FcConfigDestroy(FcConfig *config);
```

Description

Destroys a configuration and any data associated with it. Note that calling this function with the return from FcConfigGetCurrent will place the library in an indeterminate state.

Version

Fontconfig version 2.4.2

FcConfigSetCurrent

Name

`FcConfigSetCurrent` — Set configuration as default

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigSetCurrent(FcConfig *config);
```

Description

Sets the current default configuration to *`config`*. Implicitly calls FcConfigBuildFonts if necessary, returning FcFalse if that call fails.

Version

Fontconfig version 2.4.2

FcConfigGetCurrent

Name

FcConfigGetCurrent — Return current configuration

Synopsis

```
#include <fontconfig.h>

FcConfig * FcConfigGetCurrent(void);
```

Description

Returns the current default configuration.

Version

Fontconfig version 2.4.2

FcConfigUptoDate

Name

FcConfigUptoDate — Check timestamps on config files

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigUptoDate(FcConfig *config);
```

Description

Checks all of the files related to *config* and returns whether the in-memory version is in sync with the disk version.

Version

Fontconfig version 2.4.2

FcConfigBuildFonts

Name

FcConfigBuildFonts — Build font database

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigBuildFonts(FcConfig *config);
```

Description

Builds the set of available fonts for the given configuration. Note that any changes to the configuration after this call have indeterminate effects. Returns FcFalse if this operation runs out of memory.

Version

Fontconfig version 2.4.2

FcConfigGetConfigDirs

Name

FcConfigGetConfigDirs — Get config directories

Synopsis

```
#include <fontconfig.h>

FcStrList * FcConfigGetConfigDirs(FcConfig *config);
```

Description

Returns the list of font directories specified in the configuration files for *config*. Does not include any subdirectories.

Version

Fontconfig version 2.4.2

FcConfigGetFontDirs

Name

FcConfigGetFontDirs — Get font directories

Synopsis

```
#include <fontconfig.h>

FcStrList * FcConfigGetFontDirs(FcConfig *config);
```

Description

Returns the list of font directories in *config*. This includes the configured font directories along with any directories below those in the filesystem.

Version

Fontconfig version 2.4.2

FcConfigGetConfigFiles

Name

FcConfigGetConfigFiles — Get config files

Synopsis

```
#include <fontconfig.h>

FcStrList * FcConfigGetConfigFiles(FcConfig *config);
```

Description

Returns the list of known configuration files used to generate *config*. Note that this will not include any configuration done with FcConfigParse.

Version

Fontconfig version 2.4.2

FcConfigGetCache

Name

FcConfigGetCache — Get cache filename

Synopsis

```
#include <fontconfig.h>

char * FcConfigGetCache(FcConfig *config);
```

Description

Returns the name of the file used to store per-user font information.

Version

Fontconfig version 2.4.2

FcConfigGetFonts

Name

FcConfigGetFonts — Get config font set

Synopsis

```
#include <fontconfig.h>

FcFontSet * FcConfigGetFonts(FcConfig *config, FcSetName set);
```

Description

Returns one of the two sets of fonts from the configuration as specified by *set*.

Version

Fontconfig version 2.4.2

FcConfigGetBlanks

Name

FcConfigGetBlanks — Get config blanks

Synopsis

```
#include <fontconfig.h>

FcBlanks * FcConfigGetBlanks(FcConfig *config);
```

Description

Returns the FcBlanks object associated with the given configuration, if no blanks were present in the configuration, this function will return 0.

Version

Fontconfig version 2.4.2

FcConfigGetRescanInverval

Name

FcConfigGetRescanInverval — Get config rescan interval

Synopsis

```
#include <fontconfig.h>

int FcConfigGetRescanInverval(FcConfig *config);
```

Description

Returns the interval between automatic checks of the configuration (in seconds) specified in *config*. The configuration is checked during a call to FcFontList when this interval has passed since the last check.

Version

Fontconfig version 2.4.2

FcConfigSetRescanInverval

Name

FcConfigSetRescanInverval — Set config rescan interval

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigSetRescanInverval(FcConfig *config, int rescanInterval);
```

Description

Sets the rescan interval; returns FcFalse if an error occurred.

Version

Fontconfig version 2.4.2

FcConfigAppFontAddFile

Name

FcConfigAppFontAddFile — Add font file to font database

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigAppFontAddFile(FcConfig *config, const char *file);
```

Description

Adds an application-specific font to the configuration.

Version

Fontconfig version 2.4.2

FcConfigAppFontAddDir

Name

`FcConfigAppFontAddDir` — Add fonts from directory to font database

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigAppFontAddDir(FcConfig *config, const char *);
```

Description

Scans the specified directory for fonts, adding each one found to the application-specific set of fonts.

Version

Fontconfig version 2.4.2

FcConfigAppFontClear

Name

`FcConfigAppFontClear` — Remove all app fonts from font database

Synopsis

```
#include <fontconfig.h>

void FcConfigAppFontClear(FcConfig *config);
```

Description

Clears the set of application-specific fonts.

Version

Fontconfig version 2.4.2

FcConfigSubstituteWithPat

Name

`FcConfigSubstituteWithPat` — Execute substitutions

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigSubstituteWithPat(FcConfig *config, FcPattern *p,
FcPattern *p_pat, FcMatchKind kind);
```

Description

Performs the sequence of pattern modification operations, if *`kind`* is FcMatchPattern, then those tagged as pattern operations are applied, else if *`kind`* is FcMatchFont, those tagged as font operations are applied and p_pat is used for <test> elements with target=pattern.

Version

Fontconfig version 2.4.2

FcConfigSubstitute

Name

`FcConfigSubstitute` — Execute substitutions

Synopsis

```
#include <fontconfig.h>
```

```
FcBool FcConfigSubstitute(FcConfig *config, FcPattern *p, FcMatchKind
kind);
```

Description

Calls FcConfigSubstituteWithPat setting p_pat to NULL.

Version

Fontconfig version 2.4.2

FcFontMatch

Name

FcFontMatch — Return best font

Synopsis

```
#include <fontconfig.h>

FcPattern * FcFontMatch(FcConfig *config, FcPattern *p, FcResult
*result);
```

Description

Returns the font in *config* most close matching *p*. This function should be called only after FcConfigSubstitute and FcDefaultSubstitute have been called for *p*; otherwise the results will not be correct.

Version

Fontconfig version 2.4.2

FcFontSort

Name

FcFontSort — Return list of matching fonts

Synopsis

```
#include <fontconfig.h>

FcFontSet * FcFontSort(FcConfig *config, FcPattern *p, FcBool trim,
FcCharSet **csp, FcResult *result);
```

Description

Returns the list of fonts sorted by closeness to *p*. If *trim* is FcTrue, elements in the list which don't include Unicode coverage not provided by earlier elements in the list are elided. The union of Unicode coverage of all of the fonts is returned in *csp*, if *csp* is not NULL. This function should be called only after FcConfigSubstitute and FcDefaultSubstitute have been called for *p*; otherwise the results will not be correct.

The returned FcFontSet references FcPattern structures which may be shared by the return value from multiple FcFontSort calls, applications must not modify these patterns. Instead, they should be passed, along with *p* to FcFontRenderPrepare which combines them into a complete pattern.

The FcFontSet returned by FcFontSort is destroyed by caling FcFontSetDestroy.

Version

Fontconfig version 2.4.2

FcFontRenderPrepare

Name

FcFontRenderPrepare — Prepare pattern for loading font file

Synopsis

```
#include <fontconfig.h>

FcPattern * FcFontRenderPrepare(FcConfig *config, FcPattern *pat,
FcPattern *font);
```

Description

Creates a new pattern consisting of elements of *font* not appearing in *pat*, elements of *pat* not appearing in *font* and the best matching value from *pat* for elements appearing in both. The result is passed to FcConfigSubstitute with *kind* FcMatchFont and then returned.

Version

Fontconfig version 2.4.2

FcFontList

Name

FcFontList — List fonts

Synopsis

```
#include <fontconfig.h>

FcFontSet * FcFontList(FcConfig *config, FcPattern *p, FcObjectSet
*os);
```

Description

Selects fonts matching *p*, creates patterns from those fonts containing only the objects in *os* and returns the set of unique such patterns.

Version

Fontconfig version 2.4.2

FcConfigFilename

Name

FcConfigFilename — Find a config file

Synopsis

```
#include <fontconfig.h>

char * FcConfigFilename(const char *name);
```

Description

Given the specified external entity name, return the associated filename. This provides applications a way to convert various configuration file references into filename form.

A null or empty *name* indicates that the default configuration file should be used; which file this references can be overridden with the FC_CONFIG_FILE environment variable. Next, if the name starts with ~, it refers to a file in the current users home directory. Otherwise if the name doesn't start with '/', it refers to a file in the default configuration directory; the built-in default directory can be overridden with the FC_CONFIG_DIR environment variable.

Version

Fontconfig version 2.4.2

FcConfigParseAndLoad

Name

`FcConfigParseAndLoad` — load a configuration file

Synopsis

```
#include <fontconfig.h>

FcBool FcConfigParseAndLoad(FcConfig *config, const FcChar8 *file);
```

Description

Walks the configuration in 'file' and constructs the internal representation in 'config'. Any include files referenced from within 'file' will be loaded with FcConfigLoad and also parsed. If 'complain' is FcFalse, no warning will be displayed if 'file' does not exist.

Version

Fontconfig version 2.4.2

FcObjectType

Provides for applcation-specified font name object types so that new pattern elements can be generated from font names.

FcNameRegisterObjectTypes

Name

FcNameRegisterObjectTypes — Register object types

Synopsis

```
#include <fontconfig.h>

FcBool FcNameRegisterObjectTypes(const FcObjectType *types, int ntype);
```

Description

Register *ntype* new object types.

Version

Fontconfig version 2.4.2

FcNameUnregisterObjectTypes

Name

FcNameUnregisterObjectTypes — Unregister object types

Synopsis

```
#include <fontconfig.h>

FcBool FcNameUnregisterObjectTypes(const FcObjectType *types, int
ntype);
```

Description

Unregister *ntype* object types.

Version

Fontconfig version 2.4.2

FcNameGetObjectType

Name

FcNameGetObjectType — Lookup an object type

Synopsis

```
#include <fontconfig.h>

const FcObjectType * FcNameGetObjectType(const char *object);
```

Description

Return the object type for the pattern element named *object*.

Version

Fontconfig version 2.4.2

FcConstant

Provides for application-specified symbolic constants for font names.

FcNameRegisterConstants

Name

FcNameRegisterConstants — Register symbolic constants

Synopsis

```
#include <fontconfig.h>

FcBool FcNameRegisterConstants(const FcConstant *consts, int nconsts);
```

Description

Register *nconsts* new symbolic constants.

Version

Fontconfig version 2.4.2

FcNameUnregisterConstants

Name

FcNameUnregisterConstants — Unregister symbolic constants

Synopsis

```
#include <fontconfig.h>

FcBool FcNameUnregisterConstants(const FcConstant *consts, int
nconsts);
```

Description

Unregister *nconsts* symbolic constants.

Version

Fontconfig version 2.4.2

FcNameGetConstant

Name

FcNameGetConstant — Lookup symbolic constant

Synopsis

```
#include <fontconfig.h>

const FcConstant * FcNameGetConstant(FcChar8 *string);
```

Description

Return the FcConstant structure related to symbolic constant *string*.

Version

Fontconfig version 2.4.2

FcNameConstant

Name

FcNameConstant — Get the value for a symbolic constant

Synopsis

```
#include <fontconfig.h>

FcBool FcNameConstant(FcChar8 *string, int *result);
```

Description

Returns whether a symbolic constant with name *string* is registered, placing the value of the constant in *result* if present.

Version

Fontconfig version 2.4.2

FcBlanks

An FcBlanks object holds a list of Unicode chars which are expected to be blank when drawn. When scanning new fonts, any glyphs which are empty and not in this list will be assumed to be broken and not placed in the FcCharSet associated with the font. This provides a significantly more accurate CharSet for applications.

FcBlanksCreate

Name

FcBlanksCreate — Create an FcBlanks

Synopsis

```
#include <fontconfig.h>

FcBlanks * FcBlanksCreate(void);
```

Description

Creates an empty FcBlanks object.

Version

Fontconfig version 2.4.2

FcBlanksDestroy

Name

FcBlanksDestroy — Destroy and FcBlanks

Synopsis

```
#include <fontconfig.h>

void FcBlanksDestroy(FcBlanks *b);
```

Description

Destroys an FcBlanks object, freeing any associated memory.

Version

Fontconfig version 2.4.2

FcBlanksAdd

Name

FcBlanksAdd — Add a character to an FcBlanks

Synopsis

```
#include <fontconfig.h>

FcBool FcBlanksAdd(FcBlanks *b, FcChar32 ucs4);
```

Description

Adds a single character to an FcBlanks object, returning FcFalse if this process ran out of memory.

Version

Fontconfig version 2.4.2

FcBlanksIsMember

Name

`FcBlanksIsMember` — Query membership in an FcBlanks

Synopsis

```
#include <fontconfig.h>

FcBool FcBlanksIsMember(FcBlanks *b, FcChar32 ucs4);
```

Description

Returns whether the specified FcBlanks object contains the indicated Unicode value.

Version

Fontconfig version 2.4.2

FcAtomic

These functions provide a safe way to update config files, allowing ongoing reading of the old config file while locked for writing and ensuring that a consistent and complete version of the config file is always available.

FcAtomicCreate

Name

`FcAtomicCreate` — create an FcAtomic object

Synopsis

```
#include <fontconfig.h>

FcAtomic * FcAtomicCreate(const FcChar8 *file);
```

Description

Creates a data structure containing data needed to control access to *file*. Writing is done to a separate file. Once that file is complete, the original configuration file is atomically replaced so that reading process always see a consistent and complete file without the need to lock for reading.

Version

Fontconfig version 2.4.2

FcAtomicLock

Name

FcAtomicLock — lock a file

Synopsis

```
#include <fontconfig.h>

FcBool FcAtomicLock(FcAtomic *atomic);
```

Description

Attempts to lock the file referenced by *atomic*. Returns FcFalse if the file is locked by another process, else returns FcTrue and leaves the file locked.

Version

Fontconfig version 2.4.2

FcAtomicNewFile

Name

FcAtomicNewFile — return new temporary file name

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcAtomicNewFile(FcAtomic *atomic);
```

Description

Returns the filename for writing a new version of the file referenced by *atomic*.

Version

Fontconfig version 2.4.2

FcAtomicOrigFile

Name

FcAtomicOrigFile — return original file name

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcAtomicOrigFile(FcAtomic *atomic);
```

Description

Returns the file refernced by *atomic*.

Version

Fontconfig version 2.4.2

FcAtomicReplaceOrig

Name

FcAtomicReplaceOrig — replace original with new

Synopsis

```
#include <fontconfig.h>

FcBool FcAtomicReplaceOrig(FcAtomic *atomic);
```

Description

Replaces the original file referenced by *atomic* with the new file.

Version

Fontconfig version 2.4.2

FcAtomicDeleteNew

Name

FcAtomicDeleteNew — delete new file

Synopsis

```
#include <fontconfig.h>

void FcAtomicDeleteNew(FcAtomic *atomic);
```

Description

Deletes the new file. Used in error recovery to back out changes.

Version

Fontconfig version 2.4.2

FcAtomicUnlock

Name

FcAtomicUnlock — unlock a file

Synopsis

```
#include <fontconfig.h>
void FcAtomicUnlock(FcAtomic *atomic);
```

Description

Unlocks the file.

Version

Fontconfig version 2.4.2

FcAtomicDestroy

Name

FcAtomicDestroy — destroy an FcAtomic object

Synopsis

```
#include <fontconfig.h>
void FcAtomicDestroy(FcAtomic *atomic);
```

Description

Destroys *atomic*.

Version

Fontconfig version 2.4.2

File and Directory routines

These routines work with font files and directories, including font directory cache files.

FcFileScan

Name

FcFileScan — scan a font file

Synopsis

```
#include <fontconfig.h>

FcBool FcFileScan(FcFontSet *set, FcStrSet *dirs, FcFileCache *cache,
FcBlanks *blanks, const char *file, FcBool force);
```

Description

Scans a single file and adds all fonts found to *set*. If *force* is FcTrue, then the file is scanned even if associated information is found in *cache*. If *file* is a directory, it is added to *dirs*.

Version

Fontconfig version 2.4.2

FcDirScan

Name

FcDirScan — scan a font directory

Synopsis

```
#include <fontconfig.h>

FcBool FcDirScan(FcFontSet *set, FcStrSet *dirs, FcFileCache *cache,
FcBlanks *blanks, const char *dir, FcBool force);
```

Description

Scans an entire directory and adds all fonts found to *set*. If *force* is FcTrue, then the directory and all files within it are scanned even if information is present in the per-directory cache file or *cache*. Any subdirectories found are added to *dirs*.

Version

Fontconfig version 2.4.2

FcDirSave

Name

FcDirSave — save a directory cache

Synopsis

```
#include <fontconfig.h>

FcBool FcDirSave(FcFontSet *set, FcStrSet *dirs, const char *dir);
```

Description

Creates the per-directory cache file for *dir* and populates it with the fonts in *set* and subdirectories in *dirs*.

Version

Fontconfig version 2.4.2

FcDirCacheValid

Name

FcDirCacheValid — check directory cache timestamp

Synopsis

```
#include <fontconfig.h>

FcBool FcDirCacheValid(const FcChar8 *cache_file);
```

Description

Returns FcTrue if *cache_file* is no older than the directory containing it, else FcFalse.

Version

Fontconfig version 2.4.2

FcStrSet and FcStrList

A data structure for enumerating strings, used to list directories while scanning the configuration as directories are added while scanning.

FcStrSetCreate

Name

FcStrSetCreate — create a string set

Synopsis

```
#include <fontconfig.h>

FcStrSet * FcStrSetCreate(void);
```

Description

Create an empty set.

Version

Fontconfig version 2.4.2

FcStrSetMember

Name

FcStrSetMember — check set for membership

Synopsis

```
#include <fontconfig.h>
FcBool FcStrSetMember(FcStrSet *set, const FcChar8 *s);
```

Description

Returns whether *s* is a member of *set*.

Version

Fontconfig version 2.4.2

FcStrSetAdd

Name

FcStrSetAdd — add to a string set

Synopsis

```
#include <fontconfig.h>
FcBool FcStrSetAdd(FcStrSet *set, const FcChar8 *s);
```

Description

Adds a copy of *s* to *set*.

Version

Fontconfig version 2.4.2

FcStrSetAddFilename

Name

FcStrSetAddFilename — add a filename to a string set

Synopsis

```
#include <fontconfig.h>

FcBool FcStrSetAddFilename(FcStrSet *set, const FcChar8 *s);
```

Description

Adds a copy *s* to *set*, The copy is created with FcStrCopyFilename so that leading '~' values are replaced with the value of the HOME environment variable.

Version

Fontconfig version 2.4.2

FcStrSetDel

Name

FcStrSetDel — delete from a string set

Synopsis

```
#include <fontconfig.h>

FcBool FcStrSetDel(FcStrSet *set, const FcChar8 *s);
```

Description

Removes *s* from *set*, returning FcTrue if *s* was a member else FcFalse.

Version

Fontconfig version 2.4.2

FcStrSetDestroy

Name

FcStrSetDestroy — destroy a string set

Synopsis

```
#include <fontconfig.h>
void FcStrSetDestroy(FcStrSet *set);
```

Description

Destroys *set*.

Version

Fontconfig version 2.4.2

FcStrListCreate

Name

FcStrListCreate — create a string iterator

Synopsis

```
#include <fontconfig.h>
FcStrList * FcStrListCreate(FcStrSet *set);
```

Description

Creates an iterator to list the strings in *set*.

Version

Fontconfig version 2.4.2

FcStrListNext

Name

FcStrListNext — get next string in iteration

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrListNext(FcStrList *list);
```

Description

Returns the next string in *set*.

Version

Fontconfig version 2.4.2

FcStrListDone

Name

FcStrListDone — destroy a string iterator

Synopsis

```
#include <fontconfig.h>

void FcStrListDone(FcStrList *list);
```

Description

Destroys the enumerator *list*.

Version

Fontconfig version 2.4.2

String utilities

Fontconfig manipulates many UTF-8 strings represented with the FcChar8 type. These functions are exposed to help applications deal with these UTF-8 strings in a locale-insensitive manner.

FcUtf8ToUcs4

Name

FcUtf8ToUcs4 — convert UTF-8 to UCS4

Synopsis

```
#include <fontconfig.h>

int FcUtf8ToUcs4(FcChar8 *src, FcChar32 *dst, int len);
```

Description

Converts the next Unicode char from *src* into *dst* and returns the number of bytes containing the char. *src* nust be at least *len* bytes long.

Version

Fontconfig version 2.4.2

FcUcs4ToUtf8

Name

FcUcs4ToUtf8 — convert UCS4 to UTF-8

Synopsis

```
#include <fontconfig.h>

int FcUcs4ToUtf8(FcChar32 src, FcChar8 dst[FC_UTF8_MAX_LEN]);
```

Description

Converts the Unicode char from *src* into *dst* and returns the number of bytes needed to encode the char.

Version

Fontconfig version 2.4.2

FcUtf8Len

Name

FcUtf8Len — count UTF-8 encoded chars

Synopsis

```
#include <fontconfig.h>

FcBool FcUtf8Len(FcChar8 *src, int len, int *nchar, int *wchar);
```

Description

Counts the number of Unicode chars in *len* bytes of *src*. Places that count in *nchar*. *wchar* contains 1, 2 or 4 depending on the number of bytes needed to hold the largest unicode char counted. The return value indicates whether *src* is a well-formed UTF8 string.

Version

Fontconfig version 2.4.2

FcUtf16ToUcs4

Name

FcUtf16ToUcs4 — convert UTF-16 to UCS4

Synopsis

```
#include <fontconfig.h>

int FcUtf16ToUcs4(FcChar8 *src, FcEndian endian, FcChar32 *dst, int
len);
```

Description

Converts the next Unicode char from *src* into *dst* and returns the number of bytes containing the char. *src* must be at least *len* bytes long. Bytes of *src* are combined into 16-bit units according to *endian*.

Version

Fontconfig version 2.4.2

FcUtf16Len

Name

FcUtf16Len — count UTF-16 encoded chars

Synopsis

```
#include <fontconfig.h>

FcBool FcUtf16Len(FcChar8 *src, FcEndian endian, int len, int *nchar,
int *wchar);
```

Description

Counts the number of Unicode chars in *len* bytes of *src*. Bytes of *src* are combined into 16-bit units according to *endian*. Places that count in *nchar*. *wchar* contains 1, 2 or 4 depending on the number of bytes needed to hold the largest unicode char counted. The return value indicates whether *string* is a well-formed UTF16 string.

Version

Fontconfig version 2.4.2

FcStrCopy

Name

FcStrCopy — duplicate a string

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrCopy(const FcChar8 *s);
```

Description

Allocates memory, copies *s* and returns the resulting buffer. Yes, this is `strdup`, but that function isn't available on every platform.

Version

Fontconfig version 2.4.2

FcStrDowncase

Name

`FcStrDowncase` — create a lower case translation of a string

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrDowncase(const FcChar8 *s);
```

Description

Allocates memory, copies *s*, converting upper case letters to lower case and returns the allocated buffer.

Version

Fontconfig version 2.4.2

FcStrCopyFilename

Name

`FcStrCopyFilename` — copy a string, expanding '~'

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrCopyFilename(const FcChar8 *s);
```

Description

Just like FcStrCopy except that it converts any leading '~' characters in *s* to the value of the HOME environment variable. Returns NULL if '~' is present in *s* and HOME is unset.

Version

Fontconfig version 2.4.2

FcStrCmpIgnoreCase

Name

`FcStrCmpIgnoreCase` — compare UTF-8 strings ignoring ASCII case

Synopsis

```
#include <fontconfig.h>

int FcStrCmpIgnoreCase(const FcChar8 *s1, const FcChar8 *s2);
```

Description

Returns the usual <0, 0, >0 result of comparing *s1* and *s2*. This test is case-insensitive in the ASCII range and will operate properly with UTF8 encoded strings, although it does not check for well formed strings.

Version

Fontconfig version 2.4.2

FcStrStr

Name

FcStrStr — locate UTF-8 substring

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrStr(const char *s1, const char *s2);
```

Description

Returns the location of *s2* in *s1*. Returns NULL if *s2* is not present in *s1*. This test will operate properly with UTF8 encoded strings, although it does not check for well formed strings.

Version

Fontconfig version 2.4.2

FcStrStrIgnoreCase

Name

FcStrStrIgnoreCase — locate UTF-8 substring ignoring ASCII case

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrStrIgnoreCase(const char *s1, const char *s2);
```

Description

Returns the location of *s2* in *s1*, ignoring ASCII case. Returns NULL if *s2* is not present in *s1*. This test is case-insensitive in the ASCII range and will operate properly with UTF8 encoded strings, although it does not check for well formed strings.

Version

Fontconfig version 2.4.2

FcStrDirname

Name

FcStrDirname — directory part of filename

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrDirname(const FcChar8 *file);
```

Description

Returns the directory containing *file*. This is returned in newly allocated storage which should be freed when no longer needed.

Version

Fontconfig version 2.4.2

FcStrBasename

Name

FcStrBasename — last component of filename

Synopsis

```
#include <fontconfig.h>

FcChar8 * FcStrBasename(const FcChar8 *file);
```

Description

Returns the filename of *file* stripped of any leading directory names. This is returned in newly allocated storage which should be freed when no longer needed.

Version

Fontconfig version 2.4.2