// API for Reading and Rendering Fonts
// Version 1.0
// Copyright 2016, MERL, 201 Broadway, Cambridge, MA, 02139
// All rights reserved
// Ronald Perry

#ifndef GFX_FONTMGR_
define GFX_FONTMGR_
#endif

// Required include files for this header file
#include "Nitro.h"

#ifdef __cplusplus
extern "C" {
#endif

// Create a thread-safe instantiation of a font manager. A font manager instance
// enables high quality rendering of fonts in a variety of formats such as TrueType
// and OpenType. This function returns an opaque pointer to a font manager instance
// upon success; a NULL pointer is returned if the request cannot be satisfied.
void *gfxCreateFontMgr (void);

void gfxDestroyFontMgr (void *fontMgr);

int gfxLoadFont (void *fontMgr, char *filename, int slot);

void gfxUnloadFont (void *fontMgr, int slot);

#ifdef __cplusplus
}  // extern "C"
#endif
}  // ifndef GFX_FONTMGR_
typedef struct {
    int numGlyphs;
    int unitsPerEM;
    int isFixedPitch;
    int underlinePosition;
    int underlineThickness;
    int ascender;
    int descender;
    int lineGap;
    int caretSlopeRise;
    int caretSlopeRun;
    int caretOffset;
    int height;
    int bbox[4];
} GFXFontData;

void gfxGetFontData (void *fontMgr, int slot, GFXFontData *fontData);

void gfxSetHintLevel (void *fontMgr, int hintLevel);

void gfxSetRenderer (void *fontMgr, int renderer);
void gfxSetRenderingMode (void *fontMgr, int renderMode);

void gfxSetKerning (void *fontMgr, int enableKerning);

void *gfxCreateCSMTable (void *fontMgr, NTO_I1616 outsideCutoff, NTO_I1616 insideCutoff);

void gfxDestroyCSMTable (void *fontMgr, void *csmTable);

typedef struct {
    int numChars; // Number of character codes in charCodes[]
    int *charCodes; // Unicode character codes packed in slots 0..numChars-1
} GFXString; // GFXString type definition

void *gfxCreateRenderString (void *fontMgr, int slot, GFXString *s, int pixelSize,
    char rgb[3], void *csmTable);

Draw the specified render string renderString associated with the given font
Applications can use render strings to perform their own typesetting using the following steps:

1. Turn off kerning via `gfxSetKerning()`.
2. Create a render string `S` via `gfxCreateRenderString()` at the desired size with all of the necessary character codes.
3. Use `gfxGetRenderStringAttrs()` to derive the advance width for each glyph in `S`. The advance width for a glyph at index `i` in `S` is simply the difference in the position (i.e., `(penX, penY)`) between the glyph at `i+1` and the glyph at `i`.
4. Under control of a typesetting algorithm that uses the advance width data determined in step 3, use the single glyph drawing feature of `gfxDrawRenderString()` to draw a desired glyph in `S` at the computed typeset position.
5. Space bands are not included in render strings; they simply adjust the position for each glyph immediately following their occurrence. To determine the default advance width for a space band (note: typesetting algorithms typically adjust the advance width for space bands and therefore this may be unnecessary), you can perform the following steps: create a render string for a GFXString consisting of a space band and a single character (e.g., "\W"), use the pen position for the first glyph in the render string (which is "\W" since space bands are not included in render strings) to derive the default advance width for the space band -- it equals the `(penX, penY)` position for the first glyph.

See `gfxGetAdvanceWidths()` for an alternative to the steps outlined above - it's simpler and more direct, it doesn't require precomputed render strings, it directly handles space bands, and it supports the computation of advance widths with and without kerning.

```c
void gfxDrawRenderString (void *fontMgr, void *renderString, int x, int y, int useTextureMaps, int singleGlyph);
```

Return the number of glyphs in the specified render string `renderString` associated with the given font manager instance.

```c
int gfxGetRenderStringLength (void *fontMgr, void *renderString);
```

Return the size in bytes of the specified render string `renderString` associated with the given font manager instance.

```c
int gfxGetRenderStringSize (void *fontMgr, void *renderString);
```

Get the attributes of the `i`-th glyph in the specified render string `renderString` associated with the given font manager instance. Attributes include:

- `penX`: Base x coordinate for drawing pixmap
- `penY`: Base y coordinate for drawing pixmap
- `xOffset`: Add to `penX` to determine drawing (e.g., BLT) position for pixmap
- `yOffset`: Add to `penY` to determine drawing (e.g., BLT) position for pixmap
- `w`: Width of pixmap in pixels
- `h`: Height of pixmap in pixels
typedef struct
{
int sizeInBytes; // Total size in bytes of this structure and its parts
int slot; // Input slot
int charCode; // Input character code
int pixelSize; // Input pixel size
int width; // Width in pixels for bitmap when rendered
int height; // Height in pixels for bitmap when rendered
int xOffset; // Add to pen x to position bitmap properly on display
int yOffset; // Add to pen y to position bitmap properly on display
NTO_I1616 xAdvance; // x component of advance width
NTO_I1616 yAdvance; // y component of advance width
NTOPath path; // Output path; pen cmds immediately follow GFXOutline
} GFXOutline;

GFXOutline *gfxCreateOutline (void *fontMgr, int slot, int charCode, int pixelSize);

// Using the same conventions as gfxDrawRenderString(), draw the specified outline // associated with the given font manager instance at the pen position (x, y). rgb[] // and csmTable define the color and CSM table to use during drawing as described in // gfxCreateRenderString().

void gfxDrawOutline (void *fontMgr, GFXOutline *outline, int x, int y, char rgb[3], void *csmTable);
Identical to gfxDrawOutline() with one exception - the rendered outline is not drawn to the display and instead the resulting 8-bit per channel RGBA rendered image is returned to the caller. A NULL pointer is returned if the request cannot be satisfied. To release the memory used to represent the returned image, use the C standard library function free().

```c
char *gfxGetOutlineBitmap (void *fontMgr, GFXOutline *outline, char rgb[3], void *csmTable);
```

Destroy the specified outline associated with the given font manager instance

```c
void gfxDestroyOutline (void *fontMgr, GFXOutline *outline);
```

Return the Nitro renderer instance associated with the specified font manager instance

```c
void *gfxGetNitroInstance (void *fontMgr);
```

End of C++ wrapper

```c
#endif
```

End of _GFX_FONTMGR_

```c
#endif
```