Str255 data type

#include <Types.h>

typedef unsigned char **Str255**[256] ; address of 256 unsigned bytes

Variations of the Str255 types are available in different lengths. These are
typedef unsigned char **Str63**[64];
typedef unsigned char **Str32**[33];
typedef unsigned char **Str31**[32];
typedef unsigned char **Str27**[28];
typedef unsigned char **Str15**[16];

typedef const unsigned char ***ConstStr255Param**; address of constant unsigned byte. Used primarily as a parameter type in function declarations

Variations of the ConstStr255Param type are available in different lengths. These are
typedef unsigned char ***ConstStr63Param**;
typedef unsigned char ***ConstStr32Param**;
typedef unsigned char ***ConstStr31Param**;
typedef unsigned char ***ConstStr27Param**;
typedef unsigned char ***ConstStr15Param**;

typedef unsigned char ***StringPtr**; address of an unsigned byte
typedef unsigned char ****StringHandle**; handle leading to an unsigned byte

---

Notes: **Str255** is the C-structure name for a length-prefixed Pascal-style string. The first byte of an **Str255** is assumed to be the length of the text and the following bytes (up to 255 of them) are considered the text of the string.

The following functions are directly related to the use of **Str255s**:

DrawString	**IUCompString**	**NumToString**	**StringToNum**
EqualString	**IUEqualString**	**RelString**	**StringWidth**
GetIndString	**NewString**	**SetString**	**UprString**

These functions also use a **Str255** as a required parameter:

AppendMenu	**GetFNum**	**OpenDriver**	**SetFLock**
Create	**GetFontName**	**OpenResFile**	**SetIText**
CreateResFile	**GetNamedResource**	**OpenRF**	**SetWTitle**
DIZero	**GetWTitle**	**OpenRFPerm**	**SFGetFile**
FSDelete	**InsMenuItem**	**ParamText**	**SFPGetFile**
FSOpen	**NewDialog**	**Rename**	**SFPPutFile**
Get1NamedResource	**NewMenu**	**RstFLock**	**SFPutFile**
GetCTitle	**NewWindow**	**SetCTitle**	**StuffHex**
GetFInfo	**OpenDeskAcc**	**SetFInfo**	

The **Str255** data type uses up 256 bytes of storage. Note that this is

inefficient in many cases. For instance, even though **FSOpen** wants an Str255, filenames are never larger than 64 bytes (length + 63 characters). An array of filenames declared as either of:

```
Str255      nameArray[20];      /* takes 5120 bytes of storage */
Str63       nameArray[20];      /* takes only 1280 bytes */
```

The latter is clearly preferable ... and **FSOpen**(nameArray[j],...) works perfectly well either way.

Note that if a toolbox call returns **Str255** as a variable parameter (eg, **GetWTitle**), you better be sure that the pointer you pass really points to fully 256 bytes of available storage!

In some cases is it more convenient to use ASCIIZ strings and call standard C library functions such as strlen, strcpy, and so forth.