FCString

Inherits From: Object

Declared In: FCString.h

Class Description

FCString is a convenient class for managing traditional null-terminated C strings of characters. An FCString may contain a string of any length, and may hold any character except the null character (character value 0).

Methods in FCString often take as arguments either C strings or FCStrings. Methods taking C string arguments have the word "string" in their selectors, whereas methods taking FCString arguments don't.

Some methods below refer to the UNIX manual documentation (**sprintf:...** , for instance). To see this documentation, type "man *functionName* " at a Terminal command line ("man sprintf", for example). You can also use the Digital Librarian to search the documentation in /NextLibrary/Documentation/Unix/ManPages.

A variety of useful macros for working with single characters are available in /usr/include/ctype.h, which is imported by FCString.h. For more on these, see the UNIX manual page for *ctype.*

Instance Variables

Inherited from Object
None declared in this class.

Declared in FCString
char *__fc_string__ ;
int __fc_bufferLength__ ;

_fc_string	the string contents
_fc_bufferLength	the length of the string buffer

Method Types

Initializing a new FCString object	-init -initWithString:
Copying and freeing an FCString	-copyFromZone: -free
Manipulating an FCString	-prepend: -prependString: -append: -appendCharacter: -appendString: -replaceAll:with: -replaceAllStrings:with: -replaceFirst:with: -replaceFirstString:with: -replaceCharacterAt:with: -reverse -setToAllCapitals

-setToInitialCapitals
-setToLowerCase
-trimBlanks
-empty
-sprintf:...
-vsprintf:vaList:

Querying an FCString

-compare:
-compareString:
-characterAt:
-getSubstring:start:length:
-occurrencesOf:
-indexOf:
-indexOfString:
-indexOfCharacter:
-indexOfLastCharacter:
-isNumber
-isIdentifier
+packCollection:withSeparator:
-unpackAt:
-stringLength
-textLength

Reading and Writing an FCString

-writeText:
-readText:

Setting and Retrieving the FCString's Value

-stringValue
-setStringValue:
-intValue
-setIntValue:
-floatValue
-setFloatValue:
-doubleValue
-setDoubleValue:

Archiving	-write: -read:

Class Methods

packCollection:withSeparator:

+ **packCollection:stringCollection withSeparator:**(char)*aChar;*

Creates and returns an FCString containing the concatenation of the string values of the objects in *stringCollection* , separated by *aChar* .

stringCollection must be a kind of FCCollection. The objects it contains are sent the - **stringValue** message to get their string values. Objects that don't implement - **stringValue** are skipped. If none of the objects respond to - **stringValue** or if *stringCollection* is **nil** , returns **nil** .

See also: **- unpackAt:**

Instance Methods

append:

- **append:newFCString;**

Copies and appends *newFCString* to the end of the receiver. If *newFCString* is **nil** , returns **nil** .

See also: **- appendString:, - appendCharacter:, - prepend:**

appendCharacter:

- **appendCharacter:**(char)*newChar;*

Appends *newChar* to the end of the receiver. If *newChar* is the null character (0), returns **nil** .

See also: **- appendString:, - prepend:**

appendString:

- **appendString:**(const char *)*newString;*

Copies and appends *newString* to the end of the receiver. If *newString* is NULL, returns **nil** .

See also: **- append:, - appendCharacter:, - prependString:**

characterAt:

- (char)**characterAt:**(unsigned)*index;*

Returns the character at position *index* . If *index* is greater than the length of the receiver, returns 0.

See also: **- getSubstring:start:length:, - replaceCharacterAt:with:**

compare:

- (FCCompareType)**compare:anObject;**

Compares the receiver with *anObject* . Returns FC_COMPARE_GREATER_THAN, FC_COMPARE_EQUAL_TO, or FC_COMPARE_LESS_THAN depending on whether [self stringValue] is lexicographically greater than, equal to, or less than [*anObject* stringValue].

If *anObject* isn't an FCString or subclass of FCString, returns FC_CANT_COMPARE.

See also: **- compareString:**

compareString:

- (FCCompareType)**compareString:**(const char *)*string;*

Compares the receiver with *anObject* . Returns FC_COMPARE_GREATER_THAN, FC_COMPARE_EQUAL_TO, or FC_COMPARE_LESS_THAN depending on whether [self stringValue] is lexicographically greater than, equal to, or less than *string* .

If *string* is NULL, returns FC_CANT_COMPARE.

See also: **- compareString:**

copyFromZone:

- **copyFromZone:**(NXZone *)*zone;*

Creates an FCString in *zone* and returns an FCString containing a copy of the receiver's text. If the method fails, returns **nil** .

doubleValue

- (double)**doubleValue;**

Returns a double derived from the receiver. The receiver may contain an optional string of spaces, then an optional sign character, then a string of digits optionally containing a decimal point, then an optional "e" or "E" followed by an optionally signed integer. The first unrecognized character ends the conversion.

See also: **- setDoubleValue:, - stringValue, - intValue, - floatValue**

empty

- **empty;**

Empties the receiver. This is equivalent to setting the string value of the receiver to "".

See also: **- free**

floatValue

- (float)**floatValue;**

Returns a float derived from the receiver. The receiver may contain an optional string of spaces, then an optional sign character, then a string of digits optionally containing a decimal point, then an optional "e" or "E" followed by an optionally signed integer. The first unrecognized character ends the conversion.

See also: **- setFloatValue:, - stringValue, - intValue, - doubleValue**

free

- **free;**

Deallocates the receiver and the storage used by the string it contains. Returns **nil** .

getSubstring:start:length:

- (int)**getSubstring:**(char *)*buf* **start:**(int)*startPos* **length:**(int)*numChars;*

Copies a substring of length *numChars* starting at *startPos* into *buf* . *buf* must be allocated by the client. If the substring runs past the end of the FCString, only the number of characters available are copied. If the substring does not run to the end of the FCString, no terminating null will be appended to *buf* . Returns the number of characters copied. If *startPos* is greater than the length of the receiver, if *startPos* is negative, or if *buf* is NULL, returns -1.

See also: **- characterAt:**

indexOf:

- (unsigned)**indexOf:otherFCString;**

Returns the position of the first character of the first occurrence of *otherFCString* in the receiver. If *otherFCString* is **nil** or not present in the receiver, returns FC_UNSIGNED_INVALID. In addition, if the receiver or argument is empty or if *otherFCString* is longer than the receiver, returns FC_UNSIGNED_INVALID.

See also: - indexOfString:, - indexOfCharacter:, - indexOfLastCharacter:, - occurrencesOf:

indexOfCharacter:

- (unsigned)**indexOfCharacter:**(char)*letter;*

Returns the index of the first occurrence of *letter* in the receiver. If *letter* does not occur in the receiver, returns FC_UNSIGNED_INVALID.

See also: - indexOfLastCharacter:, - indexOf:, - indexOfString:, - occurrencesOf:

indexOfLastCharacter:

- (unsigned)**indexOfLastCharacter:**(char)*letter;*

Returns the index of the last occurrence of *letter* in the receiver. If *letter* does not occur in the receiver, returns FC_UNSIGNED_INVALID.

See also: - indexOfCharacter:, - indexOf:, - indexOfString:, - occurrencesOf:

indexOfString:

- (unsigned)**indexOfString:**(const char *)*otherString;*

Returns the position of the first character of the first occurrence of *otherString* in the receiver. If *otherString* is NULL or not present in the receiver, returns FC_UNSIGNED_INVALID.

See also: **- indexOf:, - indexOfCharacter:, - indexOfLastCharacter:, - occurrencesOf:**

init

- **init;**

Initializes and returns the receiver, a new FCString instance, and sets its string value to the empty string. This method is the designated initializer for FCStrings that start empty.

See also: **- initWithString:**

initWithString:

- **initWithString:**(const char *)*theString;*

Initializes and returns the receiver, a new FCString instance, and sets its string value to *theString*. This method is the designated initializer for FCStrings that start with a value.

See also: **- init**

intValue

- (int)**intValue;**

Returns an integer derived from the receiver. The receiver may contain a string of spaces, then an optional sign character, then a string of digits. The first unrecognized character ends the conversion.

See also: **- setIntValue:, - stringValue, - floatValue, - doubleValue**

isIdentifier

- (BOOL)**isIdentifier;**

Returns YES if the receiver is a valid C identifier (e.g., a variable name). Otherwise, returns NO.

See also: **- isNumber**

isNumber

- (BOOL)**isNumber;**

Returns YES if the FCString is a number. A number is defined as any sequence of digit characters optionally preceded by a plus or minus sign and optionally containing a single decimal point. Otherwise, returns NO.

See also: **- isIdentifier, - intValue, - floatValue, - doubleValue**

occurrencesOf:

- (unsigned)**occurrencesOf:anFCString;**

Returns the number of times *anFCString* occurs in the receiver. If *anFCString* is **nil** , returns 0.

See also: **- indexOf:**

prepend:

- **prepend:newFCString;**

Copies and prepends *newFCString* to the beginning of the receiver. If *newFCString* is **nil** , returns **nil** .

See also: **- prependString:, - append:**

prependString:

- **prependString:**(const char *)*newString;*

Copies and prepends *newString* to the beginning of the receiver. If *newString* is NULL, returns **nil** .

See also: **- prepend:, - appendString:**

read:

- **read:**(NXTypedStream *)*stream;*

Reads the FCString from the typed stream *stream* . Used by the Objective-C archiving functions. You should not use this method directly.

See also: **- write:**

readText:

- **readText:**(NXStream *)*stream;*

Sets the receiver to be a copy of the text read from *stream* . Null characters in the stream are ignored. If *stream* is NULL, returns **nil** .

See also: **- writeText:**

replaceAll:with:

- **replaceAll:oldFCString with:newFCString;**

Replaces all occurrences of *oldFCString* in the receiver with *newFCString* . If either argument is **nil** , returns **nil** . In addition, if *oldFCString* does not occur in the receiver or if *oldFCString* is the same object as *newFCString* , returns **nil** .

See also: **- replaceAllStrings:with:, - replaceFirst:with:**

replaceAllStrings:with:

- **replaceAllStrings:**(const char *)*oldText* **with:**(const char *)*newText;*

Replaces all occurrences of *oldText* in the receiver with *newText* . If either argument is NULL, returns **nil** . In addition, if *oldText* does not occur in the receiver or if *oldText* is the same string as *newText* , returns **nil** .

See also: **- replaceAll:with:, - replaceFirstString:with:**

replaceCharacterAt:with:

- **replaceCharacterAt:**(unsigned)*index* **with:**(char)*newCharacter;*

Replaces the character at *index* in the receiver with *newCharacter* . If the receiver is empty or is shorter than *index* , returns **nil** . If *newCharacter* is ASCII NUL (that is, (char)0), the receiver is shortened appropriately.

See also: **- replaceAllStrings:with:, - characterAt:**

replaceFirst:with:

- **replaceFirst:oldFCString with:newFCString;**

Replaces the first occurrence of *oldFCString* in the receiver with *newFCString* . If either argument is **nil** or if the arguments are the same FCString, returns **nil** .

See also: **- replaceFirstString:with:, - replaceAll:with:**

replaceFirstString:with:

- **replaceFirstString:**(const char *)*oldText* **with:**(const char *)*newText;*

Replaces the first occurrence of *oldText* in the receiver with *newText* . If either argument is NULL, returns **nil** . In addition, if the arguments are the same string or if *oldText* does not occur in the receiver, returns **nil** .

See also: **- replaceFirst:with:, - replaceAllStrings:with:**

reverse

- **reverse;**

Reverses the order of characters in the receiver.

setDoubleValue:

- **setDoubleValue:**(double)*aDouble;*

Sets the receiver to the string equivalent of *aDouble* . Note that this method will only store a limited amount of precision. If more precision is needed, **sprintf:...** should be used with an appropriate format string.

See also: **- stringValue, - sprintf:..., - setStringValue:, - setIntValue:, - setFloatValue:**

setFloatValue:

- **setFloatValue:**(float)*aFloat;*

Sets the receiver to the string equivalent of *aFloat* . Note that this method will only store a limited amount of precision. If more precision is needed, **sprintf:...** should be used with an appropriate format string.

See also: **- stringValue, - sprintf:..., - setStringValue:, - setIntValue:, - setDoubleValue:**

setIntValue:

- **setIntValue:**(int)*anInt;*

Sets the receiver to the string equivalent of *anInt* .

See also: **- stringValue, - setStringValue:, - setFloatValue:, - setDoubleValue:**

setStringValue:

- **setStringValue:**(const char *)*aString;*

Sets the receiver to be a copy of *aString* . If *aString* is NULL, returns **nil** .

See also: **- stringValue, - setIntValue:, - setFloatValue:, - setDoubleValue:**

setToAllCapitals

- **setToAllCapitals;**

Capitalizes all the lower-case characters in the receiver.

See also: **- setToInitialCapitals, - setToLowerCase**

setToInitialCapitals

- **setToInitialCapitals;**

Capitalizes the first character of every word in the receiver. A word is delimited by any combination and number of spaces, tabs, carriage returns, new lines, vertical tabs, or form feeds.

See also: **- setToAllCapitals, - setToLowerCase**

setToLowerCase

- **setToLowerCase;**

Changes all upper-case characters in the receiver to lower-case.

See also: **- setToAllCapitals, - setToInitialCapitals**

sprintf:...

- **sprintf:**(const char *)*format,* **...;**

Sets the receiver to contain the results of the corresponding sprintf() function call. For more information on the arguments to sprintf(), see the appropriate UNIX manual page. If *format* is NULL, returns **nil** .

See also: **- vsprintf:vaList:**

stringLength

- (unsigned)**stringLength;**

Returns the number of characters in the receiver. The returned value does not include the terminating null character.

See also: **- textLength**

stringValue

- (const char *)**stringValue;**

Returns a pointer to the receiver's _fc_string instance variable. Do not modify this string directly.

See also: **- setStringValue:, - intValue, - floatValue, - doubleValue**

textLength

- (int)**textLength;**

Returns the number of characters in the receiver. The return value does not include the terminating null character. This method is provided in imitation of the Application Kit's Text object.

See also: **- stringLength**

trimBlanks

- **trimBlanks;**

Removes all whitespace characters from the beginning and end of the receiver. Whitespace characters are defined as spaces, tabs, carriage returns, new lines, vertical tabs, and form feeds.

unpackAt:

- **unpackAt:**(const char *)*chars;*

Divides the receiver into individual FCStrings at every occurrence of any of the characters in *chars* . Returns an FCOrderedSet containing the FCStrings. All occurrences of *chars* are discarded in the returned FCOrderedSet. The receiver is not modified. If the receiver is empty or if *chars* is NULL, returns **nil** .

NOTE: The behavior of this method is unpredictable if there are immediately adjacent occurrences of *chars* in the receiver.

See also: **+ packCollection:withSeparator:**

vsprintf:vaList:

- **vsprintf:**(const char *)*format* **vaList:**(va_list)*ap;*

Sets the receiver to contain the results of the corresponding vsprintf() function call. For more information on the arguments to vsprintf(), see the appropriate UNIX manual page. If *format* is NULL, returns **nil** .

See also: **- sprintf:**

write:

- **write:**(NXTypedStream *)*stream;*

Writes the receiving FCString to the typed stream *stream* . Used by the Objective-C archiving functions. You should not use this method directly.

See also: **- read:**

writeText:

- **writeText:**(NXStream *)*stream;*

Writes the receiver's text to the *stream* . The terminating null is not written. If *stream* is NULL, returns **nil** .

See also: **- readText:**