Esta é uma versão muito simplificada e enfeitada do arquivo string.h (não confunda com strings.h)

```
/* ////////////////////////////////////////////////////////
// Funções de manipulação de cadeias de caracteres (= strings)
//////////////////////////////////////////////////////// */


/* Uso típico: k = strlen( x);
// A função devolve o comprimento da string x. Em outras
// palavras, devolve o número de caracteres de x (sem contar
// o '\0' final).  O código da função tem o mesmo efeito que
//       for (i = 0; x[i] != 0; ++i) ;
//       return i;
// que por sua vez equivale a
//       y = x;
//       while (*y++) ;
//       return y-x-1;
*/
unsigned int strlen( char *x);


/* Uso típico: if (strcmp( x, y) == 0) ... ;
// Compara lexicograficamente as strings apontadas por x e y.
// Devolve um número negativo se x vem antes de y, devolve 0
// se x é igual a y e devolve um número positivo se x vem
// depois de y. O código da função equivale a
//       for (i = 0; x[i] == y[i]; ++i)
//          if (x[i] == 0) return 0;
//       return x[i] - y[i];
// que por sua vez equivale a
//       while (*x++ == *y++)
//          if (*(x-1) == 0) return 0;
//       return *(x-1) - *(y-1);
*/
int strcmp( char *x, char *y);


/* Uso típico: strcpy( y, x);
// Copia a string x (inclusive o '\0' final) no espaço
// alocado para a string y. Antes de chamar a função,
// certifique-se de que o espaço alocado a y tem pelo menos
// strlen( x) + 1 bytes. A função devolve y. Exemplo:
//       char y[4];
//       strcpy( y, "ABC");
// O código da função equivale a
//       for (i = 0; (y[i] = x[i]) != 0; ++i) ;
// que por sua vez equivale a
//       while (*y++ = *x++) ;
*/
char *strcpy( char *y, char *x);
```

```
/* Uso típico: strcat( x, y);
// Concatena as strings x e y, isto é, acrescenta y ao final
// de x. Devolve o endereço da string resultante, ou seja,
// devolve x. Antes de chamar a função, certifique-se de que
// o espaço alocado a x é suficiente para comportar
// strlen( y) bytes adicionais (após o '\0' que marca o fim
// de x). Exemplo:
//       char x[7];
//       strcpy( x, "ABC");
//       strcat( x, "DEF");
// O código da função equivale a
//       strcpy( x + strlen( x), y);
*/
char *strcat( char *, char *);
```

Extrato da man page

```
NAME
     string,  strcat,  strncat,  strchr,
     strrchr,  strcmp, strncmp, strcpy, strncpy, strcspn, strspn,
     strdup, strlen, strpbrk, strstr, strtok, strtok_r,  - string
     operations

SYNOPSIS

     #include <string.h>

     char *strcat(char *s1, const char *s2);

     char *strncat(char *s1, const char *s2, size_t n);

     char *strchr(const char *s, int c);

     char *strrchr(const char *s, int c);

     int strcmp(const char *s1, const char *s2);

     int strncmp(const char *s1, const char *s2, size_t n);

     char *strcpy(char *s1, const char *s2);

     char *strncpy(char *s1, const char *s2, size_t n);

     size_t strcspn(const char *s1, const char *s2);

     size_t strspn(const char *s1, const char *s2);

     char *strdup(const char *s1);
```

```
size_t strlen(const char *s);

char *strpbrk(const char *s1, const char *s2);

char *strstr(const char *s1, const char *s2);

char *strtok(char *s1, const char *s2);

char *strtok_r(char *s1, const char *s2, char **lasts);
```

DESCRIPTION

The arguments s, s1, and s2 point to strings (arrays of characters terminated by a null character). The strcat(), strncat(), strcpy(), strncpy(), strtok(), and strtok_r() functions all alter their first argument. These functions do not check for overflow of the array pointed to by the first argument.

strcat(), strncat()

The strcat() function appends a copy of string s2, including the terminating null character, to the end of string s1. The strncat() function appends at most n characters. Each returns a pointer to the null-terminated result. The initial character of s2 overrides the null character at the end of s1.

strchr(), strrchr()

The strchr() function returns a pointer to the first occurrence of c (converted to a char) in string s, or a null pointer if c does not occur in the string. The strrchr() function returns a pointer to the last occurrence of c. The null character terminating a string is considered to be part of the string.

strcmp(), strncmp()

The strcmp() function compares two strings byte-by-byte, according to the ordering of your machine's character set. The function returns an integer greater than, equal to, or less than 0, if the string pointed to by s1 is greater than, equal to, or less than the string pointed to by s2 respectively. The sign of a non-zero return value is determined by the sign of the difference between the values of the first pair of bytes that differ in the strings being compared. The strncmp() function makes the same comparison but looks at a maximum of n bytes. Bytes following a null byte are not compared.

strcpy(), strncpy()

The strcpy() function copies string s2 to s1, including the terminating null character, stopping after the null character has been copied. The strncpy() function copies exactly n bytes, truncating s2 or adding null characters to s1 if necessary. The result will not be null-terminated if the length of s2 is n or more. Each function returns s1.

strcspn(), strspn()

The strcspn() function returns the length of the initial

segment of string s1 that consists entirely of characters not from string s2. The strspn() function returns the length of the initial segment of string s1 that consists entirely of characters from string s2.

strdup()
The strdup() function returns a pointer to a new string that is a duplicate of the string pointed to by s1. The space for the new string is obtained using malloc(3C). If the new string cannot be created, a null pointer is returned.

strlen()
The strlen() function returns the number of bytes in s, not including the terminating null character.

strpbrk()
The strpbrk() function returns a pointer to the first occurrence in string s1 of any character from string s2, or a null pointer if no character from s2 exists in s1.

strstr(), strstr()
The strstr() function locates the first occurrence of the string s2 (excluding the terminating null character) in string s1. The strstr() function returns a pointer to the located string, or a null pointer if the string is not found. If s2 points to a string with zero length (that is, the string ""), the function returns s1.

strtok()
The strtok() function can be used to break the string pointed to by s1 into a sequence of tokens, each of which is delimited by one or more characters from the string pointed to by s2. The strtok() function considers the string s1 to consist of a sequence of zero or more text tokens separated by spans of one or more characters from the separator string s2. The first call (with pointer s1 specified) returns a pointer to the first character of the first token, and will have written a null character into s1 immediately following the returned token. The function keeps track of its position in the string between separate calls, so that subsequent calls (which must be made with the first argument being a null pointer) will work through the string s1 immediately following that token. In this way subsequent calls will work through the string s1 until no tokens remain. The separator string s2 may be different from call to call. When no token remains in s1, a null pointer is returned.

strtok_r()
The strtok_r() function has the same functionality as strtok() except that a pointer to a string placeholder lasts must be supplied by the caller. The lasts pointer is to keep track of the next substring in which to search for the next token.

---

http://www.ime.usp.br/~pf/algoritmos/