Functions to manipulate C-string

Comp Sci 1570 Introduction to C++

MISSOURI S&T | Computer Science

Outline

Some basic C-string functions

```
// Returns the length of string s1.
strlen(s1);

// Copies string s2 into string s1.
strcpy(s1, s2);

// Concatenates string s2 onto
// the end of string s1.
strcat(s1, s2);

// Returns 0 if s1 and s2 are the same;
// less than 0 if s1<s2;
// greater than 0 if s1>s2.
strcmp(s1, s2);
```

Outline

strlen()

- The first of the functions we will look at is the simplest, the string length function.
- This function returns an int and has a ntca as a parameter.
- It returns the length of the data contained in the array, returning an integer equal to the number of characters in the array before reaching the first null character.
- The pitfall of the function is that there is nothing to keep it from walking off the array if there is no null character in the array!
- As a programmer, you must keep an eye on the data in your null-terminated character arrays.

```
char a_string[10] = "Bob";

// will assign 3 to length
int length = strlen(a_string);
```

Outline

strcpy()

- Returns nothing but has two ntcas as parameters.
- First parameter is the target, a non-const ntca, and the second as the source, a const ntca.
- strcpy() will copy the contents of source into target, copying every character up to the first null character of the source
- The pitfall is that this function can quite easily walk off the array!
- There is nothing to stop the function from trying to copy a ntca of length 10 into an array of length 5.
- Even if the source ntca that you pass to the function indeed has a null character marking the data, the function can still fail if you pass too short an array.

```
char source[20] = "Goodbye";
char target[20] = "Hello there";
strcpy(target, source);  // source unchanged,
```

Outline

strcat()

- Similar to strcpy(), strcat() will return nothing, but takes two ntcas, the first non-const and the second const.
- It will concatenate the second (source) onto the first (target).
- Pitfall: even worse than that of strcpy(), walking off the array is very easy to do.

```
char source[20] = "There";
char target[20] = "Hello";

// leaves source unchanged, but target is modi
strcat(target, source);
```

strcmp()

- strcmp stands for "string compare" and it performs the functionality that you would normally get out of the == operator.
- strcmp() returns an integer that gives information about two ntcas passed to it.
- Each character of the first parameter is compared to each character of the second parameter until a difference is encountered or a null character is reached. Thus, the function first compares ntca1[0] to ntca2[0].
- If they have the same ASCII value, it goes on.
- It compares ntca1[1] to ntca2[1]. If they have the same ASCII value, it goes on.
- This continues until a difference if found. Suppose that a difference is found in the nth index. If ntca1[n]'s ASCII value is less than ntca2[n]'s ASCII value, then -1 is returned. If the opposite is the case, then 1 is returned.
- 0 is returned if both ntcas are identical.

Outline

strcmp()

```
char ntca1[20] = "bob";
char ntca2[20] = "bob";
char ntca3[20] = "Bob";

cout<<strcmp(ntca1, ntca2);    // outputs 0
cout<<strcmp(ntca1, ntca3);    // outputs 1
cout<<strcmp(ntca3, ntca1);    // outputs -1
cout<<strcmp(ntca1, "bobby");  // outputs -1

if (!(strcmp(ntca1, ntca2))
    cout<<"these␣strings␣are␣identical"<<endl;
else
    cout<<"these␣strings␣are␣different"<<endl;
```

Outline

n versions of str functions

- It is easy to write insecure or buggy code with C-string input
- Using strncpy(), strncmp(), and strncat() can help
- Check out examples

Unsafe functions

- Range checking with strncopy, strncat, cin.getline, etc is often suggested

Classic overflow fix?

```
#include <iostream>
#include <cstring>

int main( void )
{
  char strDest[3]="hi";
  char strSrc[]="Welcome";
  char anotherCstring[]="Hello";

  strncpy(strDest, strSrc, 5);

  std::cout << strDest;

  return 0;
}
```

strncopy can cause another overflow too (no NULL check)

Outline

Strings

- The standard C++ library provides a string class type that supports all the operations mentioned for C-strings, but additionally with much more functionality.
- Used to require #include < string >, but now it does not.
- Brief introduction today, much more next time