Lecture 14

Strings

CSE115: Computing Concepts

Some String Functions from **`<string.h>`**

Function	**Purpose**
`strlen`	Returns the number of characters in a string
`strcpy`	Makes a copy of a string
`strncpy`	Makes a copy of a string
`strcat`	Appends a string to the end of another string
`strncat`	Appends a string to the end of another string
`strcmp`	Compare two strings alphabetically
`strncmp`	Compare two strings alphabetically

strlen example

```c
#include <stdio.h>
#include <string.h>

int main()
{
     char someStr[100] = "I love Bangladesh";
     int n;

     n = strlen("Hello world");
     printf("Length of Hello world = %d\n",n);
     n = strlen(someStr);
     printf("Length of %s = %d\n",someStr, n);
     gets(someStr);
     n = strlen(someStr);
     printf("Length of %s = %d\n",someStr, n);
     return 0;
}
```

String Assignment

- Strings can not be assigned using the assignment operator ‘=’.

```
    char str[20];
    str = “Test String”; 
```
not valid.

- String copy

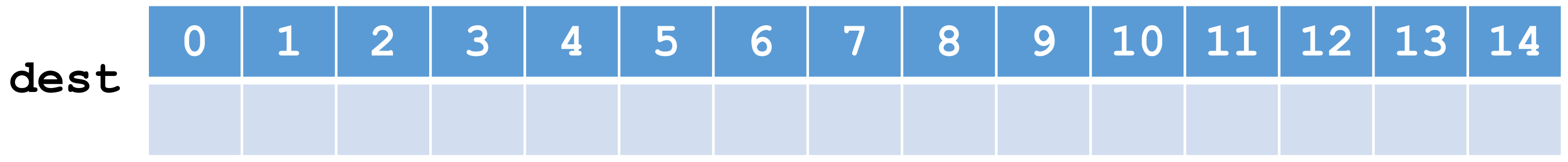
```
    strcpy(destination, source)
```

Function **`strcpy`**

- Function `strcpy` copies source string into the destination string.

  ```
  char dest[15];
  ```

- The **null character** is appended at the end automatically.
- If source string is longer than the destination string, the overflow characters may occupy the memory space used by other variables.



Function **`strcpy`**

- Function `strcpy` copies source string into the destination string.

```
char dest[15];
strcpy(dest, “test string”);
```

- The **null character** is appended at the end automatically.
- If source string is longer than the destination string, the overflow characters may occupy the memory space used by other variables.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	t	e	s	t		s	t	r	i	n	g	\0			

Function **`strncpy`**

- Function `strncpy` copies source string into the destination string by specifying the number of characters to copy.

```
    char dest[15];
```

  - If source string is longer than the destination string, the overflow characters are discarded automatically.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest															

Function **`strncpy`**

- Function `strncpy` copies source string into the destination string by specifying the number of characters to copy.

```
char dest[15];
strncpy(dest, “test string”, 6);
```

  - If source string is longer than the destination string, the overflow characters are discarded automatically.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	t	e	s	t		s									

Function **`strncpy`**

- Function `strncpy` copies source string into the destination string by specifying the number of characters to copy.

```
char dest[15];
strncpy(dest, "test string", 6);
```

`dest[6] = '\0';`

  - You have to place the null character manually.
  - If source string is longer than the destination string, the overflow characters are discarded automatically.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	t	e	s	t		s	\0								

strcpy and **strncpy** example

```
#include <stdio.h>
#include <string.h>

int main( )
{
   char source[ ] = "fresh2refresh";
   char target[20]= "";
   printf("source = %s\n", source);
   printf("target = %s\n", target);
   strcpy(target, source);
   printf("target after 1st strcpy( ) = %s\n", target);
   strcpy(target, "***************");
   printf("target after 2st strcpy( ) = %s\n", target);
   strncpy(target, source, 6);
   printf("target after strncpy( ) = %s\n", target);
   target[6] = '\0';
   printf("target after target[6] = '\\0' = %s\n",
target);
   return 0;
}
```

String Appending

- Strings can not be appended using the addition operator '+'.

  ```
  str = "Test" + "String";
  ```
  not valid.

- String concatenation

  ```
  strcat(destination, source)
  ```

Function **strcat**

- Function `strcat` concatenates the destination string with the source string.

```
char dest[15] = "Yin";
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	Y	i	n	\0											

Function **strcat**

- Function `strcat` concatenates the destination string with the source string.

```
char dest[15] = “Yin”;
strcat(dest, “ Yang”);
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	Y	i	n		Y	a	n	g	\0						

Function **`strncat`**

- Function `strcat` concatenates the destination string with the source string. By the specified number of characters to append

```
        char dest[15] = "Quest";
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	q	u	e	s	t	\0									

Function **`strncat`**

- Function `strncat` concatenates the destination string with the source string. By the specified number of characters to append

```
char dest[15] = “Quest”;
strncat(dest, “ionized”, 3);
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
dest	q	u	e	s	t	i	o	n	\0						

`strcat` and **`strncat`** example

```
#include <stdio.h>
#include <string.h>

int main ()
{
    char str1[50] = "", str2[50];
    strcpy (str2,"The");
    strncat (str1, str2, 2);
    strcpy(str2, "underdog");
    strncat (str1, str2, 5);
    strncat(str1, "catches up", 3);
    strncat(str1, "swiftly", 1);
    printf("%s", str1);
    return 0;
}
```

String Comparison

- The comparison between two strings is done by comparing each corresponding character in them (in terms of ASCII code).
  - “thrill” < “throw”
  - “joy” < joyous“
  - “Hi” < “hi”
- Strings can not be compared using the relational operators like ‘<’ or ‘==’.

```
char str1[20]=“joy”, str2[20]=“joyous”;
if (str1 < str2) not valid.
```

- String comparison

```
strcmp(string1, string2)
```

String Comparison using strcmp

Relationship	**Returned Value**	**Example**
`string1 < string2`	Negative	`strcmp(“Hello”, “Hi”)`
`string1 = string2`	0	`strcmp(“Hi”, “Hi”)`
`string1 > string2`	Positive	`strcmp(“joyous”, “joy”)`

`strcmp` Example

```
#include <stdio.h>
#include <string.h>
int main( )
{
    char str1[20] = "fresh‟, str2[20] = "refresh";
    int result;
    result = strcmp(str1, "fresh");
    printf("%s and fresh: %d\n", str1, result);
    result = strcmp(str1, "Fresh");
    printf("%s and Fresh: %d\n", str1, result);
    result = strcmp(str1, str2);
    printf("%s and %s: %d\n", str1, str2, result);
    result = strcmp(str1, "f");
    printf("%s and f: %d\n", str1, result);
    gets(str1); gets(str2);
    if(strcmp(str1, str2) == 0)
        printf("%s == %s", str1, str2);
    else if(strcmp(str1, str2) < 0)
        printf("%s < %s", str1, str2);
    else printf("%s > %s", str1, str2);
    return 0;
}
```

strncmp Example

```
#include <stdio.h>
#include <string.h>

int main ()
{
    char str[3][5] = { "R2D2" , "C3PO" , "R2A6" };
    int n;
    printf("Looking for R2xx...\n");
    for (n=0 ; n<3 ; n++)
    if (strncmp (str[n],"R2xx",2) == 0)
    {
        printf ("found %s\n",str[n]);
    }
    return 0;
}
```