

COMP26020 Programming Languages and Paradigms Part 1: C Programming

The C Standard Library Part 1

String & Memory Manipulation, Console Input, Math and Time

Standard Library

- Already saw: `printf`, `malloc`, `atoi`, etc.
- `man <function name>` in a Linux terminal:
 - Function description, prototype, required headers, return value

String Manipulation

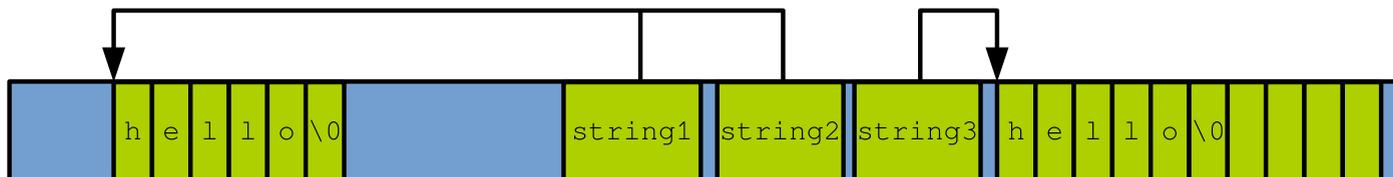
String Copy

```
char *strcpy(char *dest, const char *src);  
char *strncpy(char *dest, const char *src, size_t n);
```

[man](#)

```
#include <string.h>  
/* ... */  
  
char *string1 = "hello";  
char *string2 = string1; // this is not a string copy!  
char string3[10]; // allocated space of 10 bytes, it's called a buffer  
  
/* not super safe, what happen if string1 > string3? */  
strcpy(string3, string1);  
  
/* better */  
strncpy(string3, string1, 10);  
  
printf("string1 @%p: %s\n", string1, string1); // string1 @1234: hello  
printf("string2 @%p: %s\n", string2, string2); // string2 @1234: hello  
printf("string3 @%p: %s\n", string3, string3); // string3 @5678: hello
```

[12-standard-library-1/strcpy.c](#)



String Concatenation

```
char *strcat(char *dest, const char *src);  
char *strncat(char *dest, const char *src, size_t n);
```

[man](#)

```
#include <string.h>  
  
/* ... */  
  
char world[6] = "world";  
char s1[32];  
char s2[32];  
  
strcpy(s1, "hello ");  
strcpy(s2, "hello ");  
  
strcat(s1, world); // not very safe  
strncat(s2, world, 32); // better  
  
printf("s1: %s\n", s1); // hello world  
printf("s2: %s\n", s2); // hello world
```

[12-standard-library-1/strcat.c](#) 

Format-based String Creation

```
int sprintf(char *str, const char *format, ...);
```

[man](#)

- Fill a string with `printf`-like specifiers

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

```
int main(int argc, char **argv) {  
    int a = 12;  
    float b = 4.5;  
    char *s = "hello";  
    char string[64];  
  
    sprintf(string, "a is %d, b is %f, s is %s\n", a, b, s);  
    printf("%s", string); // a is 12, b is 4.500000, s is hello  
  
    return 0;  
}
```

[12-standard-library-1/sprintf.c](#)  

String Length and Comparison

- Get string length with `strlen`
- Compare two strings with `strcmp` -- *warning: returns 0 if they match!*

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

[man](#)

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

[man](#)

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

```
/* ... */
```

```
char *s1 = "hello"; char *s2 = "hello";
```

```
char *s3 = "not hello";
```

```
printf("strcmp(s1, s2) returns: %d\n", strcmp(s1, s2)); // 0
```

```
printf("strcmp(s1, s3) returns: %d\n", strcmp(s1, s3)); // -6
```

```
printf("length of s1: %d\n", strlen(s1)); // 5
```

```
printf("length of s2: %d\n", strlen(s2)); // 5
```

```
printf("length of s3: %d\n", strlen(s3)); // 9
```

[12-standard-library-1/strlen-strcmp.c](https://en.cppreference.com/w/cpp/string/basic/string_constants) 

Console Input

Console Input

```
char *fgets(char *s, int size, FILE *stream);m
```

```
int scanf(const char *format, ...);
```

[man](#)

```
int main(int argc, char **argv) {
    int int1, int2;
    double double1;
    float float1;
    char s[128];

    printf("Please input a string:\n");
    fgets(s, 128, stdin);

    printf("Please input an integer:\n");
    scanf("%d", &int1);

    printf("Please input a float:\n");
    scanf("%lf", &double1); /* make sure to use %lf for double and %f for float */

    printf("Please enter an integer and a float separated by a space\n");
    scanf("%d %f", &int2, &float1);

    printf("You have entered: %d, %d, %lf, %f, and %s\n", int1, int2, double1,
           float1, s);
    return 0;
}
```

Memory Manipulation

Writing bytes to memory, Copying memory

```
void *memset(void *s, int c, size_t n);
```

[man](#)

```
void *memcpy(void *dest, void *src, size_t n);
```

[m](#)

```
#include <stdio.h>
#include <string.h> // needed for memcpy and memset
#include <stdlib.h>

int main(int argc, char **argv) {
    int buffer_size = 10;

    char *ptr1 = malloc(buffer_size);
    char *ptr2 = malloc(buffer_size);

    if(ptr1 && ptr2) {
        memset(ptr1, 0x40, buffer_size); // 0x40 is ascii code for @
        memcpy(ptr2, ptr1, buffer_size);

        for(int i=0; i<buffer_size; i++) {
            printf("ptr1[%d] = %c\n", i, ptr1[i]);
            printf("ptr2[%d] = %c\n", i, ptr2[i]);
        }
    }

    return 0;
}
```

Math

Math functions

```
double ceil(double x);  
float ceilf(float x);
```

[man](#)

```
double sqrt(double x);
```

[man](#)

```
double pow(double x);
```

[man](#)

```
double cos(double x);
```

```
//
```

[man](#)

```
// When compiling a program using math.h,  
// use -lm on the command line:  
// gcc program.c -o program -lm
```

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

```
#include <math.h> // needed for math functions
```

```
int main(int argc, char **argv) {  
    printf("ceil 2.5: %f\n", ceil(2.5));  
    printf("floor 2.5: %f\n", floor(2.5));  
    printf("2^5: %f\n", pow(2, 5));  
    printf("sqrt(4): %f\n", sqrt(4));
```

```
    return 0;
```

```
}
```

[12-standard-library-1/math.c](#)  

- And many more, see here:

<http://www.cplusplus.com/reference/cmath/>

Time

Sleeping

`unsigned int sleep(unsigned int seconds);` [man](#)

`int usleep(useconds_t usec);` [man](#)

```
#include <stdio.h>
#include <unistd.h> // needed for sleep and usleep

int main(int argc, char **argv) {
    printf("hello!\n");
    printf("Sleeping for 2 seconds ...\n");

    sleep(2);

    printf("Now sleeping for .5 seconds ...\n");

    usleep(500000);

    return 0;
}
```

[12-standard-library-1/sleep.c](#)  

Current Time

```
time_t time(time_t *tloc); // time_t is generally a long long unsigned int
```

[man](#)

```
#include <stdio.h>
#include <time.h> // needed for time()

int main(int argc, char **argv) {
    unsigned long long t = time(NULL);

    printf("Number of seconds elapsed since the epoch (01/01/1970):\n");
    printf("%llu\n", t);
    return 0;
}
```

[12-standard-library-1/time.c](#)  

Measuring Execution Time

```
int gettimeofday(struct timeval *tv, struct timezone *tz);  
// struct timeval {  
//     time_t      tv_sec;      /* seconds (type: generally long unsigned) */  
//     suseconds_t tv_usec;     /* microseconds (type: generally long unsigned) */  
// };
```

[man](#)

```
#include <stdio.h>  
#include <sys/time.h> // needed for gettimeofday  
  
int main(int argc, char **argv) {  
    struct timeval tv, start, stop, elapsed;  
  
    gettimeofday(&tv, NULL);  
    printf("Seconds since the epoch: %lu.%06lu\n", tv.tv_sec, tv.tv_usec);  
  
    gettimeofday(&start, NULL);  
    for(int i=0; i<1000000000; i++);  
    gettimeofday(&stop, NULL);  
  
    timersub(&stop, &start, &elapsed);  
  
    printf("Busy loop took %lu.%06lu seconds\n", elapsed.tv_sec,  
          elapsed.tv_usec);  
  
    return 0;  
}
```



Summary

- C standard library offers various low level functions for
 - String and memory operations
 - Console I/O
 - Math
 - Time
 - etc.
-

Feedback form: <https://bit.ly/2VFzbb2>

