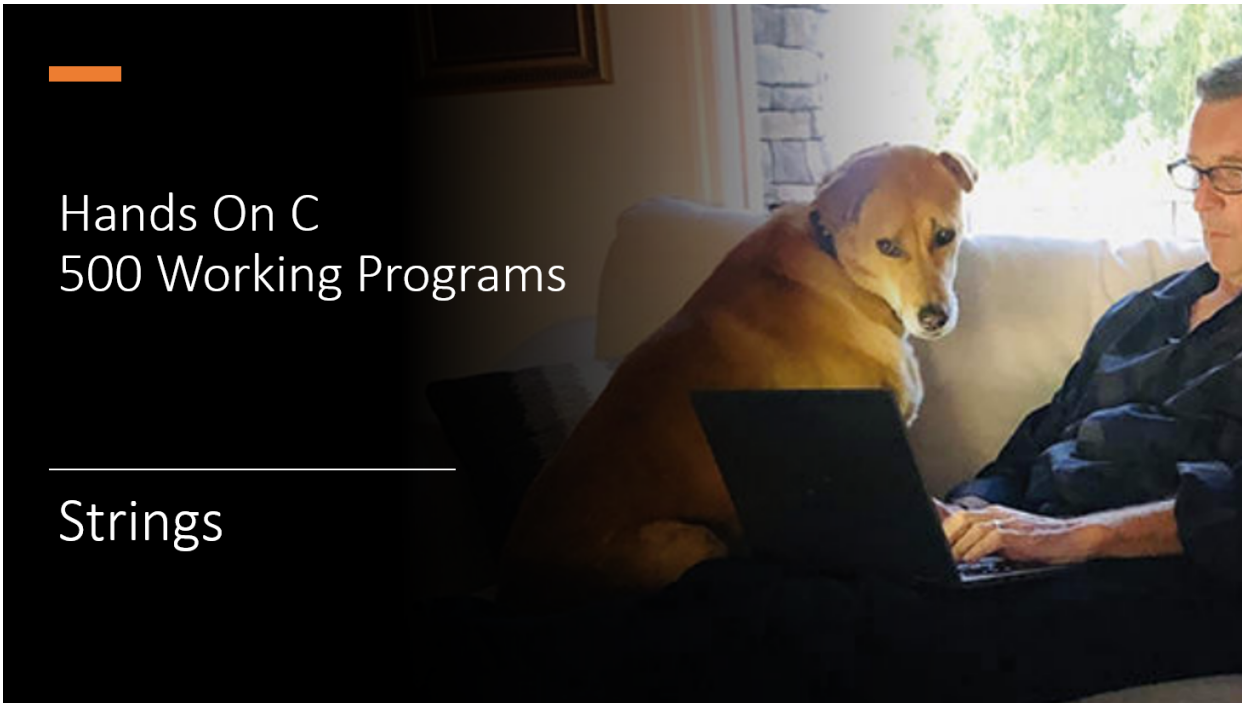


C Stores Strings as an Array of Characters in Consectutive Memory Locations

H	e	l	l	o		W	o	r	l	d	\0

In [1]:
```c
#include <stdio.h>

int main(void)
{
   printf("Hello, world");
}
```

```
Hello, world
```

In [2]:
```c
#include <stdio.h>

int main(void)
{
  char string[64] = "Hello, world";

  printf("%s", string);
}
```

```
Hello, world
```

In [3]:
```c
#include <stdio.h>

int main(void)
{
    char string[] = "Hello, world";

    printf("%s", string);
}
```

```
Hello, world
```

In [4]:
```c
#include <stdio.h>

int main(void)
{
   char *string = "Hello, world";

   printf("%s", string);
}
```

```
Hello, world
```

In [5]:
```c
#include <stdio.h>

int main(void)
{
   char string[64] = "Hello, world";

   for (int i = 0; string[i] != '\0'; ++i)
     printf("%c", string[i]);
}
```

```
Hello, world
```

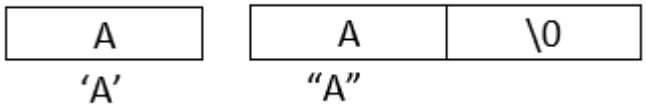
In [6]:
```c
#include <stdio.h>

int main(void)
{
   char string[64] = "Hello, world";

   for (int i = 0; string[i]; ++i)  // null is an ASCII 0 which C considers false
     printf("%c", string[i]);
}
```

```
Hello, world
```

Revisiting How 'A' Differs from "A"



In [7]:
```c
#include <stdio.h>

int main(void)
{
   char letter = 'A';
   char string[] = "A";

   printf("sizeof letter %ld\n", sizeof(letter));
   printf("sizeof string %ld\n", sizeof(string));
}
```

```
sizeof letter 1
sizeof string 2
```

Understanding Buffer Overflow

In [8]:
```c
#include <stdio.h>

int main(void)
{
   char firstName[] = "Kris";
   char lastName[64] = "Jamsa";
}
```

In [9]:
```c
#include <stdio.h>

int main(void)
{
   char firstName[] = "Kris";
   char lastName[64] = "Jamsa";

   firstName[5] = 'B';

   printf("%s %s\n", firstName, lastName);
}
```

```
Kris Bamsa
```

Understanding Depricated Functions

In [10]:
```
#include <stdio.h>

int main(void)
{
  char string[64];

  printf("Enter a string: ");
  gets(string);                 // See compiler warning
}
```

```
/tmp/tmpj8knaow_.c: In function ‘main’:
/tmp/tmpj8knaow_.c:8:3: warning: implicit declaration of function ‘gets’; did y
ou mean ‘fgets’? [-Wimplicit-function-declaration]
   gets(string);                 // See compiler warning
   ^~~~
   fgets
/tmp/ccaIypOD.o: In function `main':
tmpj8knaow_.c:(.text+0x35): warning: the `gets' function is dangerous and shoul
d not be used.

Enter a string:
```

Representing Single and Double Quotes within Strings

In [11]:
```c
#include <stdio.h>

int main(void)
{
   printf("That\'s incredible\n");
   printf("He said \"C is cool!\"\n");
}
```

```
That's incredible
He said "C is cool!"
```

Determining the Number of Characters in a String

In [12]:
```c
#include <stdio.h>

int main(void)
{
   char string[] = "Hello, world!";
   int length;

   for (length = 0; string[length] != '\0'; length++)
     ;

   printf("The string %s has %d characters\n", string, length);
}
```

```
The string Hello, world! has 13 characters
```

In [13]:
```c
#include <stdio.h>

int main(void)
{
   char string[] = "Hello, world!";
   int length;

   for (length = 0; string[length]; length++)
     ;

   printf("The string %s has %d characters\n", string, length);
}
```

```
The string Hello, world! has 13 characters
```

Using the C strlen Function

In [14]:
```
#include <stdio.h>
#include <string.h>

int main(void)
{
   char string[64] = "Hello, world!";

   printf("The string %s has %ld letters\n", string, strlen(string));
}
```

```
The string Hello, world! has 13 letters
```

Copying a String to Another Variable

In [15]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
   char string[64] = "Hello, world!";
   char newString[64];

   strcpy(newString, string);
   printf("Old string %s\nNew string %s\n", string, newString);
}
```

```
Old string Hello, world!
New string Hello, world!
```

Remember Buffer Overflow

In [16]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
   char thirdString[64] = "Kris Jamsa";
   char newString[5];
   char string[64] = "Hello, world!";

   strcpy(newString, string);
   printf("Old string %s\nNew string %s\n", string, newString);
   printf("Third string %s\n", thirdString);
}
```

```
Old string Hello, world!
New string Hello, world!
Third string , world!
```

Appending One String to Another

In [17]:
```c
#include <stdio.h>

int main(void)
{
   char string[64] = "Hello";
   char secondString[64] = "World!";

   int i, j;

   // find end of first string
   for (i = 0; string[i]; i++)
     ;

   // append second string
   for (j = 0; string[i] = secondString[j]; ++i, ++j)
       ;

   printf("%s\n", string);
}
```

```
HelloWorld!
```

Using the C strcat Function to Append One String to Another

In [18]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char string[64] = "Hello";
    char secondString[64] = "World";

    strcat(string, secondString);
    printf("%s\n", string);
}
```

```
HelloWorld
```

In [19]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char string[64] = "Hello";
    char secondString[64] = "World";

    strcat(string, ", ");
    strcat(string, secondString);
    printf("%s\n", string);
}
```

```
Hello, World
```

Using C's strncat Function to Append n Characters of One String to Another

In [20]:
```
#include <stdio.h>
#include <string.h>

int main(void)

{
    char string[64] = "Hello";
    char secondString[64] = "World Peace";

    strncat(string, secondString, 5);
    printf("%s\n", string);
}
```

```
HelloWorld
```

Using C's strxfrm Function to Copy n Characters from One String to Another

In [21]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char string[64] = "Hello";
    char secondString[64] = "World Peace";

    strxfrm(string, secondString, 5);
    printf("%s\n", string);
}
```

```
World
```

Determining if Two Strings are the Same

In [22]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
  char stringOne[] = "Hello, World";
  char stringTwo[] = "Hello, World";
  char stringThree[] = "Hello, World!";

  int i;

  for (i = 0; stringOne[i] == stringTwo[i] && stringOne[i]; i++)
    ;
  if (stringOne[i] == '\0' && stringTwo[i] == '\0')
    printf("%s and %s are the same\n", stringOne, stringTwo);

  for (i = 0; stringTwo[i] == stringThree[i] && stringTwo[i]; i++)
    ;
  if (stringTwo[i] == '\0' && stringThree[i] == '\0')
    printf("%s and %s are the same\n", stringTwo, stringThree);
}
```

```
Hello, World and Hello, World are the same
```

Using C's strcmp Function to Determine if Two Strings are Equal

```
returns 0 if equal
```

```
returns -1 if string1 < string2
returns 1 if string > string2
```

In [23]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
  char stringOne[] = "Hello, World";
  char stringTwo[] = "Hello, World";
  char stringThree[] = "Hello, World!";

if (strcmp(stringOne, stringTwo) == 0)
    printf("%s and %s are the same\n", stringOne, stringTwo);

  if (strcmp(stringTwo, stringThree) == 0)
    printf("%s and %s are the same\n", stringTwo, stringThree);
}
```

```
Hello, World and Hello, World are the same
```

Using C's strncmp to Compare the First n Characters of Two Strings

In [24]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
  char stringOne[] = "Hello, World";
  char stringTwo[] = "Hello, World";
  char stringThree[] = "Hello, Mundo!";

  if (strncmp(stringOne, stringTwo, 5) == 0)
    printf("First 5 characters of %s and %s are the same\n", stringOne, stringTwo

  if (strncmp(stringTwo, stringThree, 5) == 0)
    printf("First 5 characters %s and %s are the same\n", stringTwo, stringThree)
}
```

```
First 5 characters of Hello, World and Hello, World are the same
First 5 characters Hello, World and Hello, Mundo! are the same
```

Ignoring Case When Checking if Two Strings are the Same

In [25]:
```c
#include <stdio.h>
#include <ctype.h>

int main(void)
{
  char stringOne[] = "Hello, World";
  char stringTwo[] = "hello, world";
  char stringThree[] = "Hello, World!";

  int i;

  for (i = 0; toupper(stringOne[i]) == toupper(stringTwo[i]) && stringOne[i]; i++
    ;

  if (stringOne[i] == '\0' && stringTwo[i] == '\0')
    printf("%s and %s are the same\n", stringOne, stringTwo);

  for (i = 0; toupper(stringTwo[i]) == toupper(stringThree[i]) && stringTwo[i]; i
    ;

  if (stringTwo[i] == '\0' && stringThree[i] == '\0')
    printf("%s and %s are the same\n", stringTwo, stringThree);

}
```

```
Hello, World and hello, world are the same
```

Converting a String to Upper or Lowercase

In [26]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
  char stringOne[] = "Hello, World";

  // Convert to lowercase
  for (int i = 0; stringOne[i]; ++i)
      if ((stringOne[i] >= 'A') && (stringOne[i] <= 'Z'))
          stringOne[i] = stringOne[i] + ('a' - 'A');

  printf("%s\n", stringOne);

  // Convert to uppercase
  for (int i = 0; stringOne[i]; ++i)
      if ((stringOne[i] >= 'a') && (stringOne[i] <= 'z'))
          stringOne[i] = stringOne[i] - ('a' - 'A');

  printf("%s\n", stringOne);
}
```

```
hello, world
HELLO, WORLD
```

Finding the First Occurrence of a Character within a String

In [27]:
```
#include <stdio.h>
#include <string.h>

int main(void)
{
  char string[64] = "Hello";
  char *ptr;

  ptr = strchr(string, 'l');

  if (*ptr)
    printf("The letter %c appears in %s at offset %ld\n", 'l', string, ptr-string
}
```

```
The letter l appears in Hello at offset 2
```

Finding the First Occurrence of a Character within a String

In [28]:
```c
#include <stdio.h>
#include <string.h>

int main(void)
{
  char string[64] = "Hello";
  char *ptr;

  ptr = strrchr(string, 'l');

  if (*ptr)
    printf("The letter %c appears rightmost in %s at offset %ld\n", 'l', string,
}
```

```
The letter l appears rightmost in Hello at offset 3
```

Converting ASCII Representations of a Number to a Number

```
atoi     ASCII to integer
atof     ASCII to float
atol     ASCII to long
strtod   String to double
strtol   String to long
```

In [29]:
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void)
{
   int a = atoi("1234");
   float b = atof("12345.6789");
   long c = atol("123456789L");

   printf("%d %f %ld\n", a, b, c);

   char *result;
   double d = strtod("987.654321", &result);

   if (*result == '\0')
     printf("%g\n", d);

   long e = strtol("987654321", &result, 10);  // base 10

   if (*result == '\0')
     printf("%ld\n", e);
}
```

```
1234 12345.678711 123456789
987.654
987654321
```

Locating the First Occurrence of a Substring witin

a String

In [30]:
```
#include <stdio.h>
#include <string.h>

int main(void)
{
   char string[] = "AbcABCAbc";

   char *ptr;

   ptr = strstr(string, "ABC");
   if (ptr != NULL)
     printf("Substring ABC found");
   else
     printf("Substring ABC not found");
}
```

```
Substring ABC found
```

Writing Formatted Output to a String Variable

In [31]:
```
#include <stdio.h>

int main(void)
{
  char buffer[256];

  sprintf(buffer, "%d %f %s", 1, 3.14, "Hello, world");

  printf("%s", buffer);
}
```

```
1 3.140000 Hello, world
```

Reading Formatted Input from a String Variable

In [32]:
```
#include <stdio.h>

int main(void)
{
  char buffer[256];
  int age;
  float pi;
  char string[256];

  sprintf(buffer, "%d %f %s", 1, 3.14, "Hello, world");

  printf("%s\n", buffer);

  sscanf(buffer, "%d %f %s", &age, &pi, string);
  printf("%d %f %s\n", age, pi, string);
}
```

```
1 3.140000 Hello, world
1 3.140000 Hello,
```

What You will Learn Next

```
To help C programmers manipulate strings, the C library provides a collection
of functions your programs can use.

for (int i = 0; string[i]; ++)
  if (islower(string[i])
    string[i] = toupper(string[i]);
```