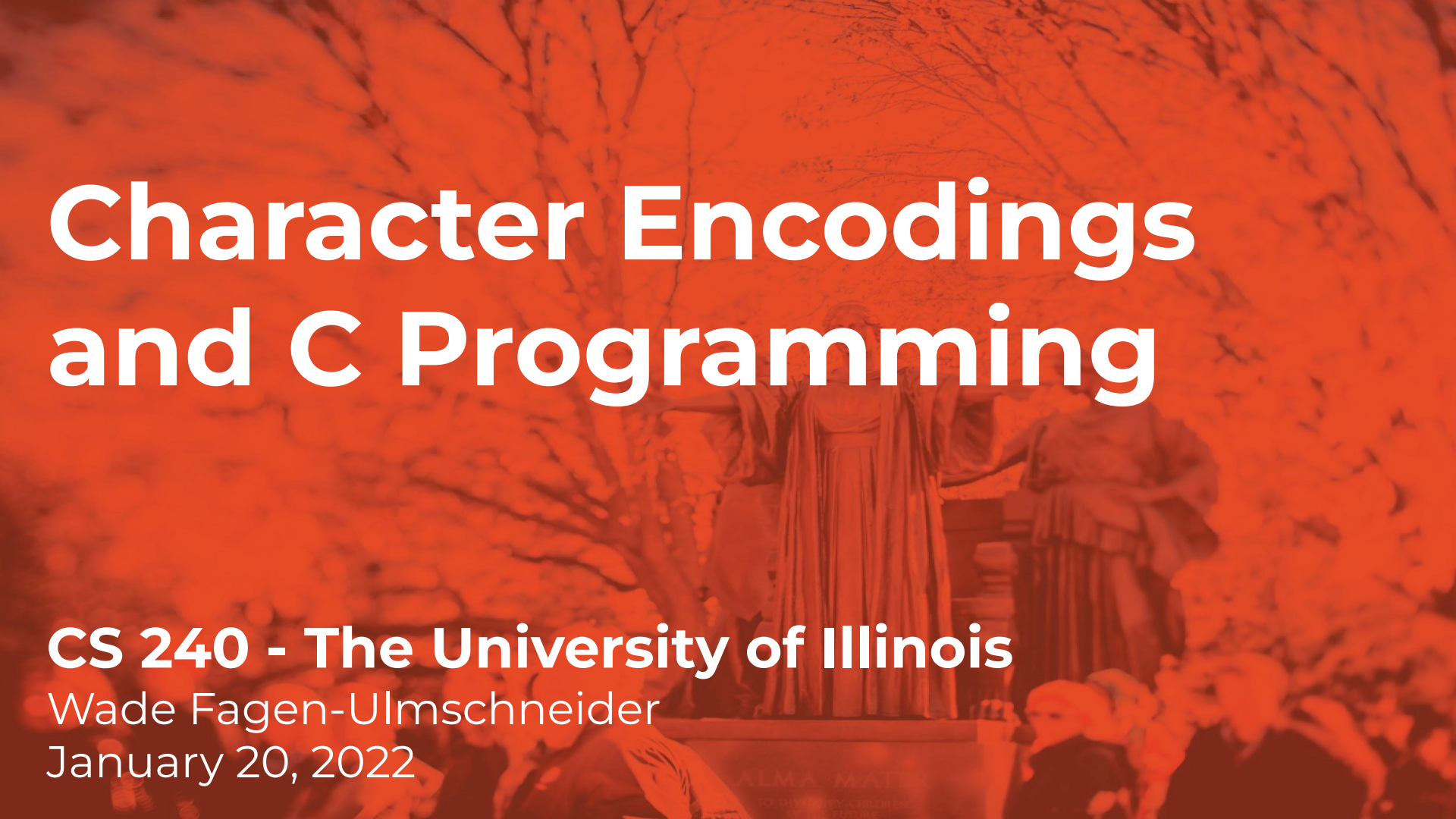


Character Encodings and C Programming

CS 240 - The University of Illinois

Wade Fagen-Ulmschneider

January 20, 2022

Character Encodings

Representing numbers is great -- but what about words? Can we make sentences with binary data?

ASCII

Organization

To begin to create words:

- A letter is _____ binary bits.
  _____ hex digits!

(We call this unit a _____.)

Organization

Global standard called the **American Standard Code for Information Interchange (ASCII)** is a _____ for translating numbers to characters.

ASCII

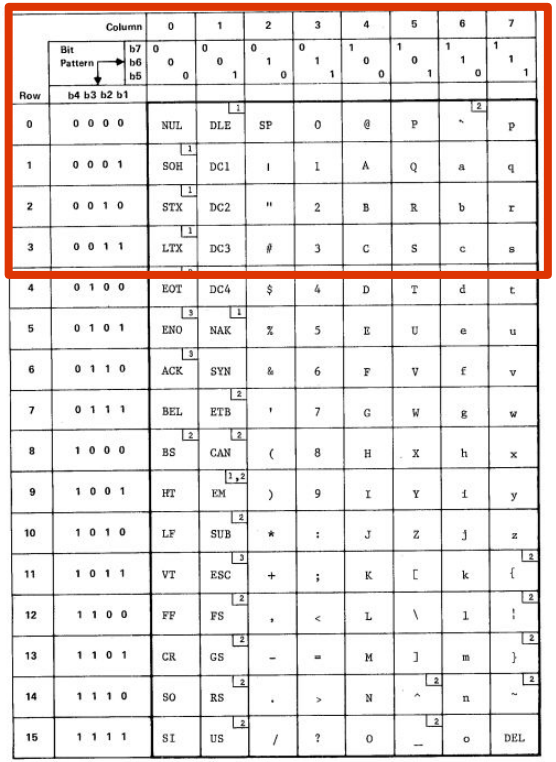
Row	b4 b3 b2 b1	Column 0 (b7 b6 b5 = 0 0 0)	Column 1 (0 0 1)	Column 2 (0 1 0)	Column 3 (0 1 1)	Column 4 (1 0 0)	Column 5 (1 0 1)	Column 6 (1 1 0)	Column 7 (1 1 1)
0	0 0 0 0	NUL	[1] DLE	SP	0	@	P	[2] `	p
1	0 0 0 1	[1] SOH	DC1	!	1	A	Q	a	q
2	0 0 1 0	[1] STX	DC2	"	2	B	R	b	r
3	0 0 1 1	[1] LTX	DC3	#	3	C	S	c	s
4	0 1 0 0	[2] EOT	DC4	$	4	D	T	d	t
5	0 1 0 1	[3] ENO	[1] NAK	%	5	E	U	e	u
6	0 1 1 0	[3] ACK	SYN	&	6	F	V	f	v
7	0 1 1 1	BEL	[2] ETB	'	7	G	W	g	w
8	1 0 0 0	[2] BS	[2] CAN	(	8	H	X	h	x
9	1 0 0 1	HT	[1,2] EM	)	9	I	Y	i	y
10	1 0 1 0	LF	[2] SUB	*	:	J	Z	j	z
11	1 0 1 1	VT	[3] ESC	+	;	K	[	k	[2] {
12	1 1 0 0	FF	[2] FS	,	<	L	\	l	[2] ¦
13	1 1 0 1	CR	[2] GS	-	=	M	]	m	[2] }
14	1 1 1 0	SO	[2] RS	.	>	N	[2] ^	n	[2] ~
15	1 1 1 1	SI	[2] US	/	?	O	[2] _	o	DEL

[1] Change of name
[2] New character
[3] Moved character

Fig. 14.12 ASCII, 1967 and 1968

Coded Character Set, History, and Development (Charles MacKenzie)
https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf

ASCII



Row	b4 b3 b2 b1	Column 0 (b7 b6 b5 = 0 0 0)	Column 1 (0 0 1)	Column 2 (0 1 0)	Column 3 (0 1 1)	Column 4 (1 0 0)	Column 5 (1 0 1)	Column 6 (1 1 0)	Column 7 (1 1 1)
0	0 0 0 0	NUL	[1] DLE	SP	0	@	P	[2] `	p
1	0 0 0 1	[1] SOH	DC1	!	1	A	Q	a	q
2	0 0 1 0	[1] STX	DC2	"	2	B	R	b	r
3	0 0 1 1	[1] LTX	DC3	#	3	C	S	c	s
4	0 1 0 0	EOT	DC4	$	4	D	T	d	t
5	0 1 0 1	[3] ENO	[1] NAK	%	5	E	U	e	u
6	0 1 1 0	[3] ACK	SYN	&	6	F	V	f	v
7	0 1 1 1	BEL	[2] ETB	'	7	G	W	g	w
8	1 0 0 0	[2] BS	[2] CAN	(	8	H	X	h	x
9	1 0 0 1	HT	[1,2] EM	)	9	I	Y	i	y
10	1 0 1 0	LF	[2] SUB	*	:	J	Z	j	z
11	1 0 1 1	VT	[3] ESC	+	;	K	[	k	[2] {
12	1 1 0 0	FF	[2] FS	,	<	L	\	l	[2] ¦
13	1 1 0 1	CR	[2] GS	-	=	M	]	m	[2] }
14	1 1 1 0	SO	[2] RS	.	>	N	[2] ^	n	[2] ~
15	1 1 1 1	SI	[2] US	/	?	O	[2] _	o	DEL

[1] Change of name
[2] New character
[3] Moved character

Fig. 14.12 ASCII, 1967 and 1968

Coded Character Set, History, and Development (Charles MacKenzie)
https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf

Column		0	1	2	3	4	5	6	7
Bit Pattern →	b7	0	0	0	0	1	1	1	1
	b6	0	0	1	1	0	0	1	1
	b5	0	1	0	1	0	1	0	1
Row	b4 b3 b2 b1								
0	0 0 0 0	NUL	DLE [1]	SP	0	@	P	` [2]	p
1	0 0 0 1	SOH [1]	DC1	!	1	A	Q	a	q
2	0 0 1 0	STX [1]	DC2	"	2	B	R	b	r
3	0 0 1 1	LTX [1]	DC3	#	3	C	S	c	s

Coded Character Set, History, and Development (Charles MacKenzie)
https://textfiles.meulie.net/bitsaved/Books/Mackenzie_CodedCharSets.pdf

0b 0100 0001 = 0x41 = A

0b 0100 0001 = 0x41 = A

0b 0100 0010 = 0x42 = B

0b 0100 0001 = 0x41 = A

0b 0100 0010 = 0x42 = B

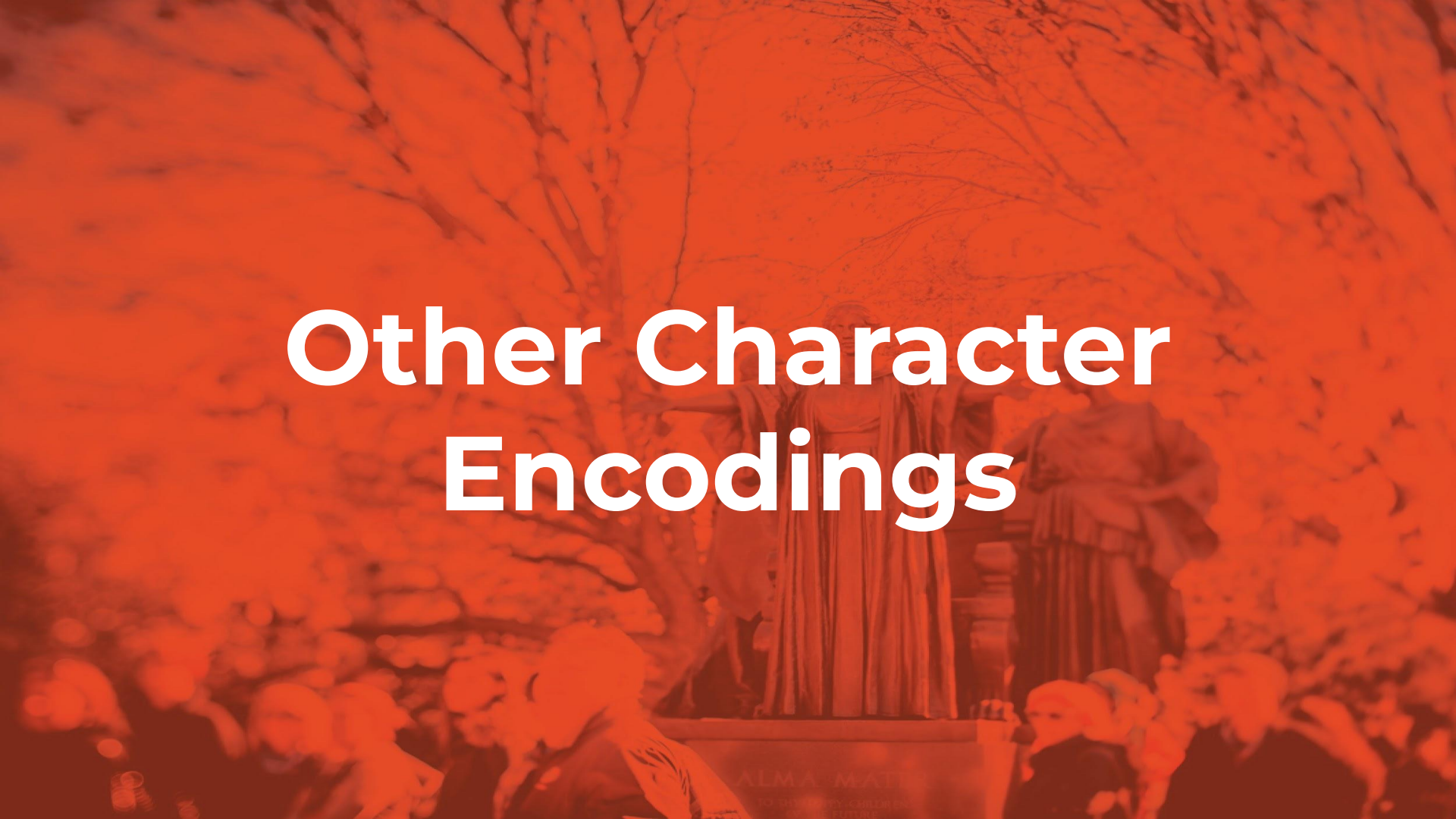
= 0x43 =

= 0x44 =

0b 0100 0001 = 0x41 = A

0b 0110 0001 = 0x61 = a

Shortcomings with ASCII



Other Character Encodings

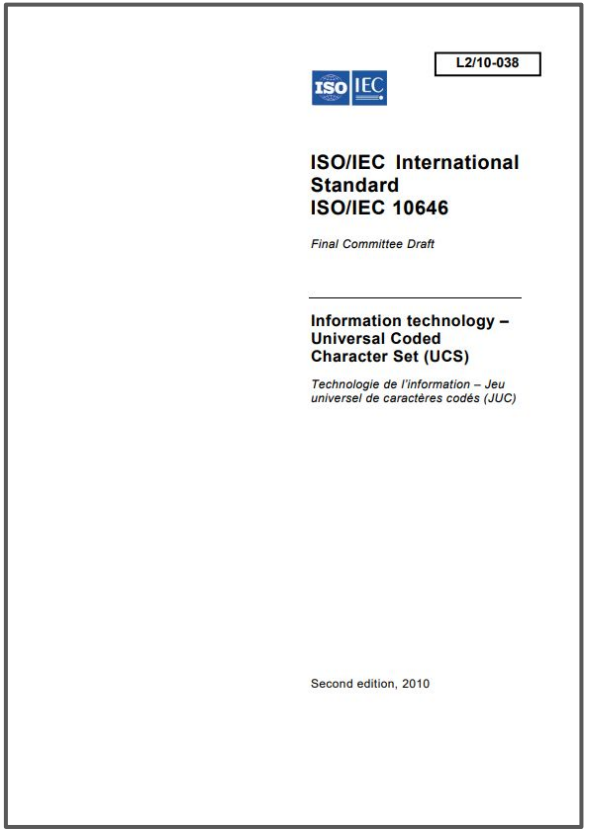
Character Encodings

There are **many** other character encodings beyond ASCII.

Character Encodings

One of the most common is the **Unicode Transformation Format (8-bit)**, commonly called:

ISO/IEC 10646



L2/10-038

ISO/IEC International Standard
ISO/IEC 10646

Final Committee Draft

Information technology – Universal Coded Character Set (UCS)

Technologie de l'information – Jeu universel de caractères codés (JUC)

Second edition, 2010

https://www.iso.org/standard/76835.html

Length	Byte #1	Byte #2	Byte #3	Byte #4
1-byte:	`0___ ____`			
2-bytes:	`110_ ____`	`10__ ____`		
3-bytes:	`1110 ____`	`10__ ____`	`10__ ____`	
4-bytes:	`1111 0___`	`10__ ____`	`10__ ____`	`10__ ____`

Characters in UTF-8

a

Characters in UTF-8

ε

U+03b5

0100 1000 0110 1001 1111 0000

1001 1111 1000 1110 1000 1001

0100 1000|0110 1001|1111 0000|

1001 1111|1000 1110|1000 1001

Length	Byte #1	Byte #2	Byte #3	Byte #4
1-byte:	`0___ ____`			
2-bytes:	`110_ ____`	`10__ ____`		
3-bytes:	`1110 ____`	`10__ ____`	`10__ ____`	
4-bytes:	`1111 0___`	`10__ ____`	`10__ ____`	`10__ ____`

0100 1000|0110 1001|1111 0000|

1001 1111|1000 1110|1000 1001

0100 1000|0110 1001|1111 0000|

1001 1111|1000 1110|1000 1001

Hi 

Programming in C

You already know C++!

You already know C++!

Programming in C **is a simplification of C++**.

1. Program Starting Point:

2. Printing to `stdout`

02/printf.c

```c
#include <stdio.h>

int main() {
  int i = 42;
  char *s = "Hello, world!";
  float f = 3.14;

  printf("%d  %s  %f\n", i, s, f);
  printf("%d\n", s[0]);
  printf("%d\n", s);
  printf("%d\n", f);

  return 0;
}
```

3. Pointers

4. Heap Memory Allocation

02/malloc.c

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

int main() {
  char *s = malloc(10);
  int *num = malloc( sizeof(int) );

  printf("%p %p\n", s, num);
  return 0;
}
```

5. Strings

5. Strings

There is no “data type” in C known as a string. Instead, we refer to “C Strings” as a sequence of characters:

- A “C string” is just a character pointer: _____.

- The string continues until it reaches a _____ byte.

02/string.c

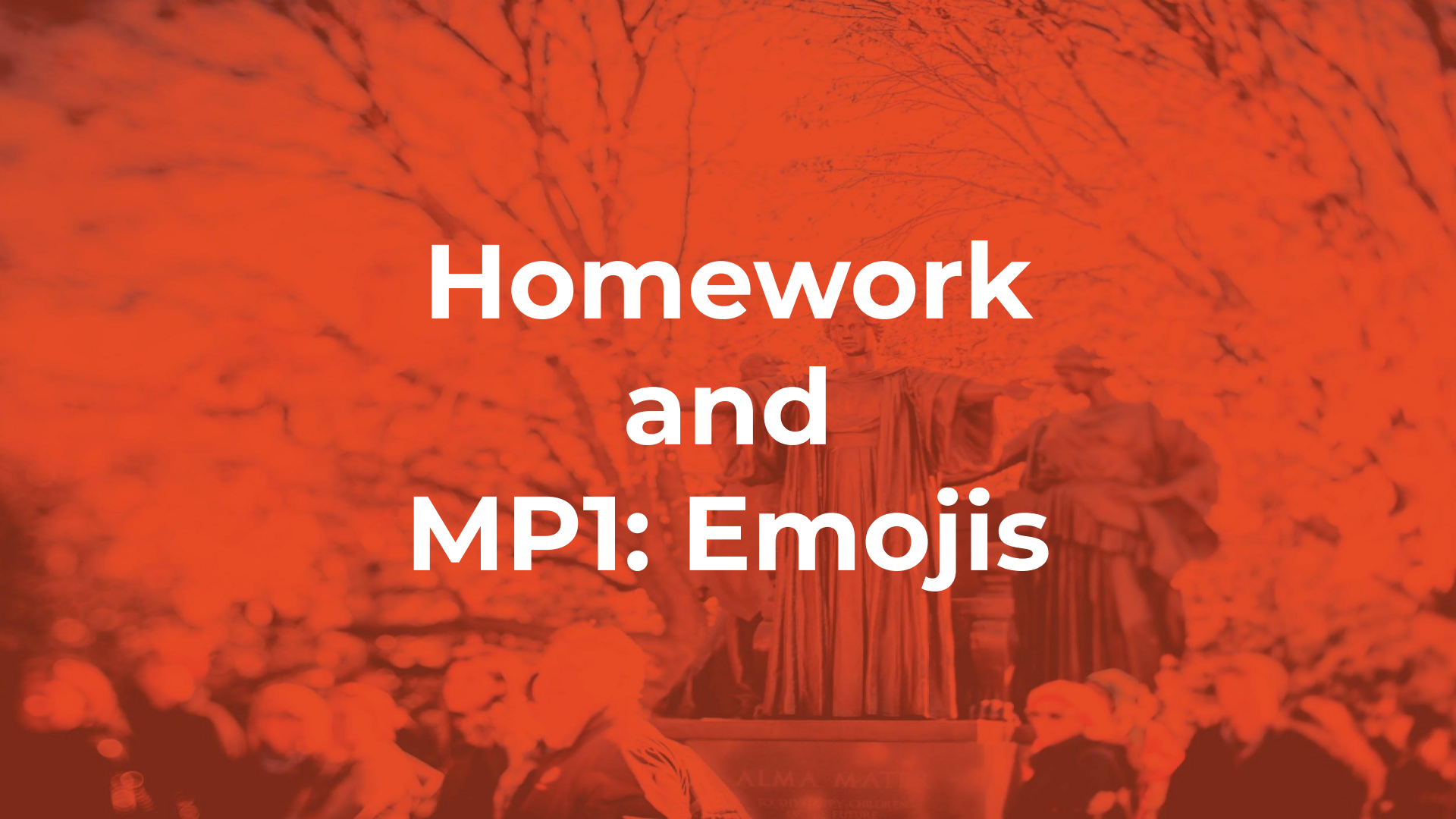
```c
char *s = malloc(6);
strcpy(s, "cs240");
printf("s[0]: 0x%x == %d == %c\n", s[0], s[0], s[0]);
printf("s[4]: 0x%x == %d == %c\n", s[4], s[4], s[4]);
printf("s[5]: 0x%x == %d == %c\n", s[5], s[5], s[5]);
printf("s == \"%s\", strlen(s): %ld\n\n", s, strlen(s));

char *s2 = s + 2;
printf("s2[0]: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == \"%s\", strlen(s2): %ld\n\n", s2, strlen(s2));

*s2 = 0;
printf("s2[0]: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == \"%s\", strlen(s2): %ld\n\n", s2, strlen(s2));

printf("s == \"%s\", strlen(s): %ld\n", s, strlen(s));
```

02/utf8.c

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

int main() {
  char *s = malloc(5);
  s[0]=0xF0; s[1]=0x9F; s[2]=0x8E; s[3]=0x89; s[4]=0x00;

  char *s1 = "\xF0\x9F\x8E\x89";
  char *s2 = "🎉";
  char *s3 = "\U0001f389";    // \U - must be 8 bytes

  printf("%s %s %s %s\n", s, s1, s2, s3);
  printf("strlen(): %ld %ld %ld %ld\n", strlen(s),
                        strlen(s1), strlen(s2), strlen(s3));
  return 0;
}
```

Some extremely useful built in string functions:

- **`strcmp(char *s1, char *s2)`** -- Compares two strings
- **`strcat(char *dest, char *src)`** -- Concatenate two strings
- **`strcpy(char *dest, char *src)`** -- Copies a string
- **`strlen(char *s)`** -- Returns the length of the string



Homework and MP1: Emojis