Union…Casting…Which to Choose?

Casting

When do you use union and when do you use casting?

Casting

- Casting is used to convert a variable’s type into another type.
- Sometimes the compiler will do the “casting” but it is simple to make it explicit.

```
int int_Num;
float float_Num;
int_Num = 32;
float_Num = (float) int_Num;
printf ( “%d %f\n”, int_Num, float_Num );
```

```
32 32.000000
```

```
#include <stdio.h>

int main ()
{
   char character;
   int int_Num;
   float float_Num;

   /*  Convert character to integer */
   character = 'a';
   int_Num = (int) character;
   printf ( "%c %d\n", character, int_Num );

   character = '2';
   int_Num = (int) character;
   printf ( "%c %d\n", character, int_Num );

   /*  Convert float to integer */
   float_Num = 32.125;
   int_Num = (int) float_Num;
   printf ( "%f %d\n", float_Num, int_Num );
}
```

```
$ ./cast2
a 97
2 50
32.125000 32
```

```
#include <stdio.h>

int main ()
{
   int int_Num;
   short little_Num;

   /*  Convert integer to short */
   int_Num = 1000;
   little_Num = (short) int_Num;
   printf ( "%d %d\n", int_Num, little_Num );

   int_Num = 1000000;
   little_Num = (short) int_Num;
   printf ( "%d %d\n", int_Num, little_Num );

}
```

Integer can "fit" into the short.

Integer cannot "fit" into the short.

```
$ ./cast3
1000 1000
1000000 16960
```

```
1000000 = 00000000 00001111 01000010 01000000
  16960 =                   01000010 01000000
```

```c
#include <stdio.h>
int main ()
{
   union {
      long long_element;
      float float_element;
   } u;

   long long_var;
   float float_var;

   long_var = u.long_element = 10;
   printf ( "The value of long_var cast to a float is: %f\n",
      (float) long_var );
   printf ( "The value of float_element is: %f\n", u.float_element );

   float_var = u.float_element = 3.555;
   printf ( "The value of float_var cast to a long is: %ld\n",
      (long) float_var );
   printf ( "The value of long_element is: %ld\n", u.long_element );

}
```

```
$ ./unionCast
The value of long_var cast to a float is: 10.000000
The value of float_element is: 0.000000
The value of float_var cast to a long is: 3
The value of long_element is: 1080263967
```

```c
#include <stdio.h>
int main ()
{
   union {
      long long_element;
      float float_element;
   } u;

   long long_var;
   float float_var;

   long_var = u.long_element = 10;
   printf ( "The value of long_var cast to a float is: %f\n",
      (float) long_var );
   printf ( "The value of float_element is: %f\n", u.float_element );

   float_var = u.float_element = 3.555;
   printf ( "The value of float_var cast to a long is: %ld\n",
      (long) float_var );
   printf ( "The value of long_element is: %ld\n", u.long_element );

}
```

Casting with Pointers

The values of the characters are treated as an integer.

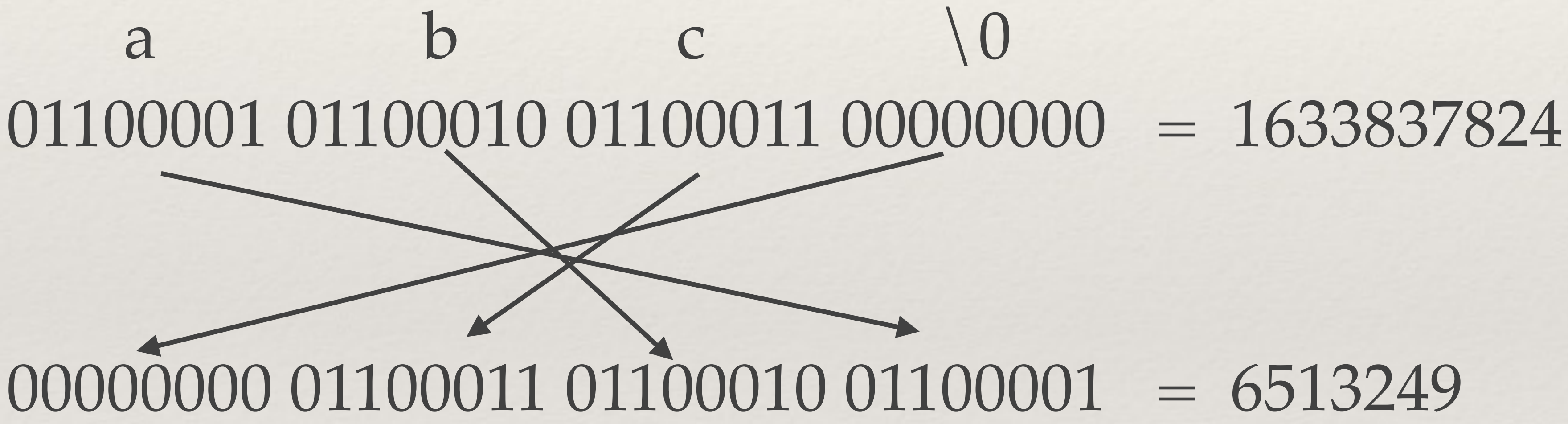
	a	b	c	\0
char_ptr =	01100001	01100010	01100011	00000000
*int_ptr =	1633837824			

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

int main ()
{
   char *char_ptr;
   char pstring[] = { "abc" };
   int *int_ptr;

   char_ptr = pstring;
   int_ptr = (int *) char_ptr;

   printf ( "%s = %d\n", pstring, *int_ptr );

}
```

But there is a problem…

```
$ ./cast4a
abc = 6513249
```

Not 1633837824 - why?



```
       a          b          c         \0
01100001 01100010 01100011 00000000  =  1633837824

00000000 01100011 01100010 01100001  =  6513249
```

The bytes are reversed!!!

The Endian is Near…

Byte Ordering

Not just size is important - so is order!!!

Byte Ordering

- Microprocessor architectures commonly use two different methods to store the individual bytes of multibyte numerical data in memory.   The operating system is not a factor.
- This difference is referred to as *byte ordering* or *endian nature*.
- **Little-Endian Byte Ordering**
  - Least significant bytes first
- **Big-Endian Byte Ordering (or Network Byte Order)**
  - Most significant bytes first

Terminology

- The terms big-endian and little-endian come from Jonathan Swift's eighteenth-century satire Gulliver's Travels. The subjects of the empire of Blefuscu were divided into two factions: those who ate eggs starting from the big end and those who ate eggs starting from the little end

https://developer.apple.com/library/content/documentation/CoreFoundation/Conceptual/CFMemoryMgmt/Concepts/ByteOrdering.html

Representation in Memory

- Decimal: 1025
- 32 bit representation
- Big Endian
  - Binary: 00000000 00000000 00000100 00000001
- Little Endian
  - Binary: 00000001 00000100 00000000 00000000

http://www.yolinux.com/TUTORIALS/Endian-Byte-Order.html

Casting with Pointers

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

int main ()
{
   char *char_ptr;
   int *int_ptr;
   char pstring[] = { "abc" };
   int littleE;
   int bigE;

   char_ptr = pstring;
   int_ptr = (int *) char_ptr;
   printf ( "%s = %d = ", pstring, *int_ptr );
   littleE = *int_ptr;

   bigE = htobe32(littleE);
   printf ( "%d\n", bigE );
}
```

```
abc = 6513249 = 1633837824
```