Representing and Manipulating Integers

(2/3)

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Type Casting in C (1)

- Constants
  - By default, considered to be signed integers
  - Unsigned if have “U” or “u” as suffix
    - 0U, 12345U, 0x1A2Bu

- Type casting
  - Explicit casting
    ```
    int tx, ty;
    unsigned ux, uy;
    tx = (int) ux;
    uy = (unsigned) ty;
    ```
  - Implicit casting via
    - Assignments
    - Procedure calls
    ```
    int f(unsigned);
    tx = ux;
    f(ty);
    ```
Type Casting in C (2)

- Expression evaluation
  - If mix unsigned and signed in single expression, signed values implicitly cast to **unsigned**.
  - Including comparison operations `<`, `>`, `==`, `<=`, `>=`

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 == 0U</td>
<td>unsigned</td>
<td>True</td>
</tr>
<tr>
<td>-1 &lt; 0</td>
<td>signed</td>
<td>True</td>
</tr>
<tr>
<td>-1 &lt; 0U</td>
<td>unsigned</td>
<td>False</td>
</tr>
<tr>
<td>-1 &gt; -2</td>
<td>signed</td>
<td>True</td>
</tr>
<tr>
<td>(unsigned) -1 &gt; -2</td>
<td>unsigned</td>
<td>True</td>
</tr>
<tr>
<td>2147483647 &gt; -2147483647-1</td>
<td>signed</td>
<td>True</td>
</tr>
<tr>
<td>2147483647U &gt; -2147483647-1</td>
<td>unsigned</td>
<td>False</td>
</tr>
<tr>
<td>2147483647 &gt; (int) 2147483648U</td>
<td>signed</td>
<td>True</td>
</tr>
</tbody>
</table>
Type Casting in C (3)

- **Example 1-1**

```c
int main ()
{
    unsigned i;
    for (i = 10; i > 0; i--)
        printf ("%u\n", i);
}
```

- **Example 1-2**

```c
int main ()
{
    unsigned i;
    for (i = 10; i >= 0; i--)
        printf ("%u\n", i);
}
```
### Example 2

```c
int sum_array (int a[], unsigned len)
{
    int i;
    int result = 0;

    for (i = 0; i <= len - 1; i++)
        result += a[i];

    return result;
}
```
Type Casting in C (5)

- Example 3-1

```c
void copy_mem1 (char *src, char *dest, unsigned len)
{
    unsigned i;
    for (i = 0; i < len; i++)
        *dest++ = *src++;
}
```

- Example 3-2

```c
void copy_mem2 (char *src, char *dest, unsigned len)
{
    int i;
    for (i = 0; i < len; i++)
        *dest++ = *src++;
}
```
Type Casting in C (6)

- Example 3-3

```c
void copy_mem3 (char *src, char *dest, unsigned len)
{
    for (; len > 0; len--)
        *dest++ = *src++;
}
```

- Example 3-4

```c
void copy_mem4 (char *src, char *dest, unsigned len)
{
    for (; (int) len > 0; len--)
        *dest++ = *src++;
}
```
Example 4

```c
#include <stdio.h>

int main ()
{
    unsigned char c;

    while ((c = getchar()) != EOF)
        putchar (c);
}
```
Lessons

- There are many tricky situations when you use unsigned integers – hard to debug
- Do not use just because numbers are nonnegative
- Use only when you need collections of bits with no numeric interpretation (“flags”)
- Few languages other than C support unsigned integers
Addition (1)

- Integer addition example
  - 4-bit integers $u$, $v$
  - Compute true sum
  - True sum requires one more bit ("carry")
  - Values increase linearly with $u$ and $v$
  - Forms planar surface
Addition (2)

- **Unsigned addition**
  - Ignores carry output
  - Wraps around
    - If true sum $\geq 2^w$
    - At most once

---

![Graph showing unsigned addition](image)

*True Sum*

$2^{w+1}$

Overflow

$2^w$

$0$

*Unsigned addition*
Addition (3)

- **Signed addition**
  - Drop off MSB
  - Treat remaining bits as 2’s comp. integer

![Graph showing two's complement addition](image)

**True Sum**

- Positive overflow
- Negative overflow

- Signed addition

- Positive overflow
- Negative overflow

- Two’s complement addition (4-bit word)