Operating Systems Lab (C+Unix)

Enrico Bini

University of Turin

Outline

Conditional and comma operators

- The conditional "?:" is a ternary operator

  <expr1> ? <expr2> : <expr3>

  returns:
  - <expr2> if <expr1> is non-zero
  - <expr3> if <expr1> is zero

  Typical usage is the maximum

```
#define MAX(x,y)  ( (x) > (y) ? (x) : (y) )
```

- The comma "," operator

  <expr1> , <expr2>

  <expr1> and <expr2> are evaluated in this order and <expr2> is returned
  - sometime used in `for` loops in the first and third expressions, when more assignments or increments are needed

```
int v[VEC_LEN], *p, i, somma = 0;

for (p = v, i = 0; i < VEC_LEN; p++, i++) {
    somma = somma + *p;
}
```

Precedence of operators

- Operators (such as "+" or "*") are used to combine operands and then produce expressions
- When evaluating a complex expression, in what precedence are operators evaluated?

```
a = 2;
b = 3;
c = a+a          *b;
```

- In math we know that multiplications are made before addition
- This is called **precedence** of operators
- C operators have a precedence (to be illustrated later on)

Associativity of operators

- When combining operators of the same precedence, in what order do we proceed?

```
a = 2;
b = 3;
c = 4;
d = a - b - c;
a = b = c = d;
```

- **Associativity** can be "right-to-left" or "left-to-right"

Table Precedence/Associativity 1/3

Available at

`http://en.cppreference.com/w/c/language/operator_precedence`

Starting from **highest** precedence

Prec.	Operator	Description	Associativity
1	`++ --`	Suffix/postfix incr. and decr.	Left-to-right
	`()`	Function call	
	`[]`	Array subscripting	
	`.`	Struct/union access	
	`->`	Struct/union access via pointer	
2	`++ --`	Prefix increment and decrement	Right-to-left
	`+ -`	Unary plus and minus	
	`! ~`	Logical NOT and bitwise NOT	
	`(type)`	Type cast	
	`*`	Indirection (dereference)	
	`&`	Address-of	
	`sizeof`	Size-of	

Table Precedence/Associativity 2/3

Available at

`http://en.cppreference.com/w/c/language/operator_precedence`

Prec.	Operator	Description	Associativity
3	* / %	Mul., div., and remainder	Left-to-right
4	+ -	Addition and subtraction	Left-to-right
5	<< >>	Bitwise left shift and right shift	Left-to-right
6	< <=	Comparison $<$ and $\leq$ respectively	Left-to-right
	> >=	Comparison $>$ and $\geq$ respectively	Left-to-right
7	== !=	For relational $=$ and $\neq$ respectively	Left-to-right
8	&	Bitwise AND	Left-to-right
9	^	Bitwise XOR	Left-to-right
10	|	Bitwise OR	Left-to-right
11	&&	Logical AND	Left-to-right
12	||	Logical OR	Left-to-right

Table Precedence/Associativity 3/3

Available at
`http://en.cppreference.com/w/c/language/operator_precedence`

Prec.	Operator	Description	Associativity
13	`?:`	Ternary conditional	Right-to-Left
14	`=`	Simple assignment	Right-to-Left
	`+= -=`	Assign. by sum/difference	
	`*= /= %=`	Assign. by prod./quot./remainder	
	`<<= >>=`	Assign. by bitwise left/right shift	
	`&= ^= |=`	Assign. by bitwise AND/XOR/OR	
15	`,`	Comma	Left-to-right