Type Specifier & Type Qualifier (1A)

Young Won Lim
10/4/17

Type Specifier

- void
- char
- short
- int
- long
- float
- double

- signed
- unsigned

- *struct-or-union-specifier*
- *enum-specifier*
- *typedef-name*

- const

- volatile

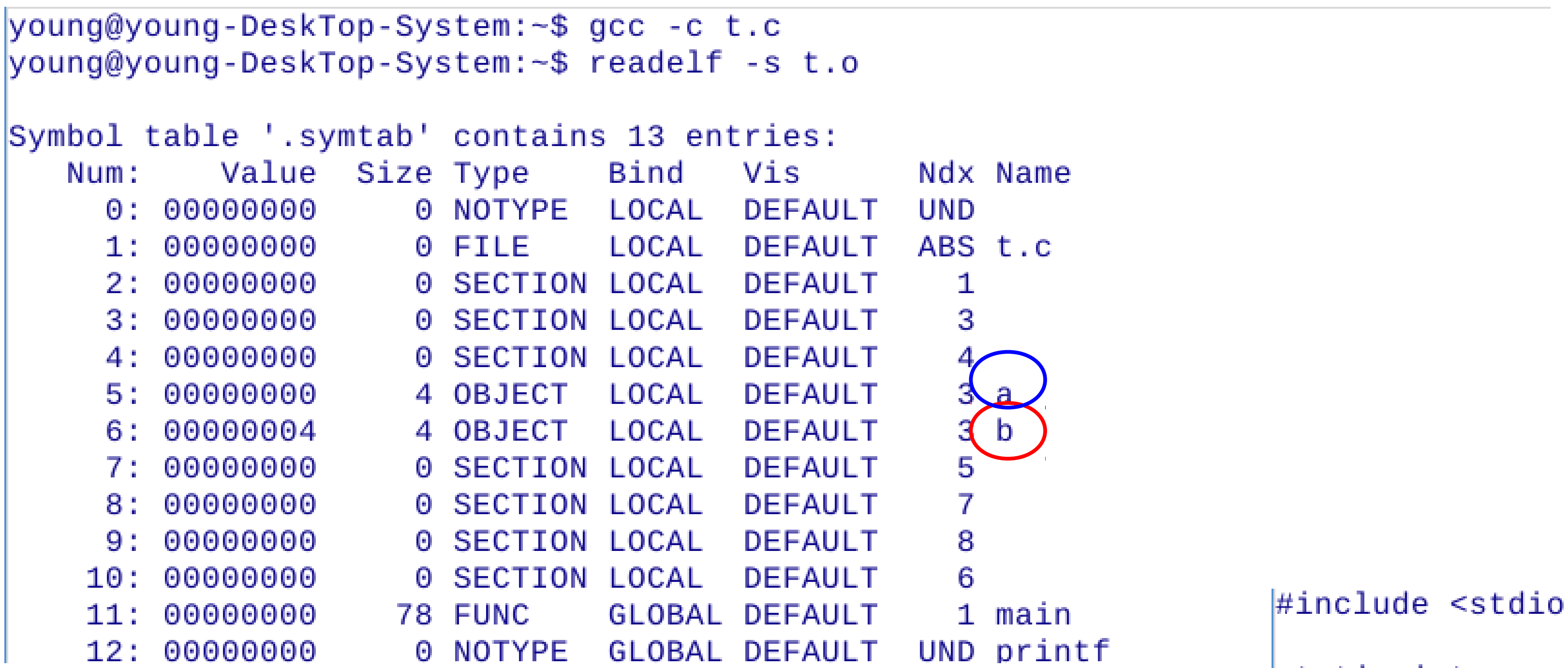
volatile

- do not optimize
- provide a reliable access
  to special memory location used
  by computer hardware or
  by asynchronous process
  such as interrupt handlers

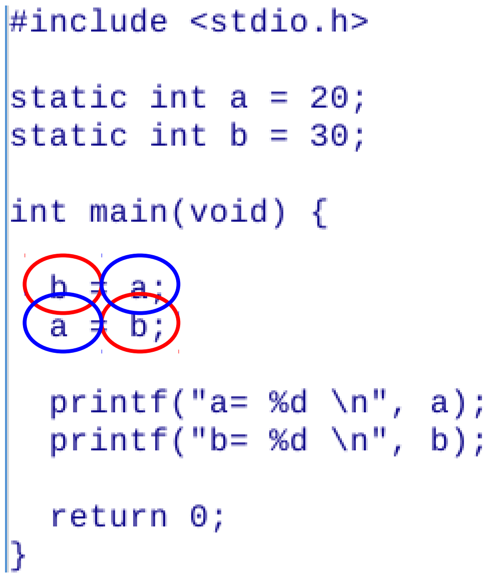
```
a = *port_address;        // read access
…
some computation
...
*port_address = a;     // write access
```

compiler optimize to
remove
port_address

Compile without optimization

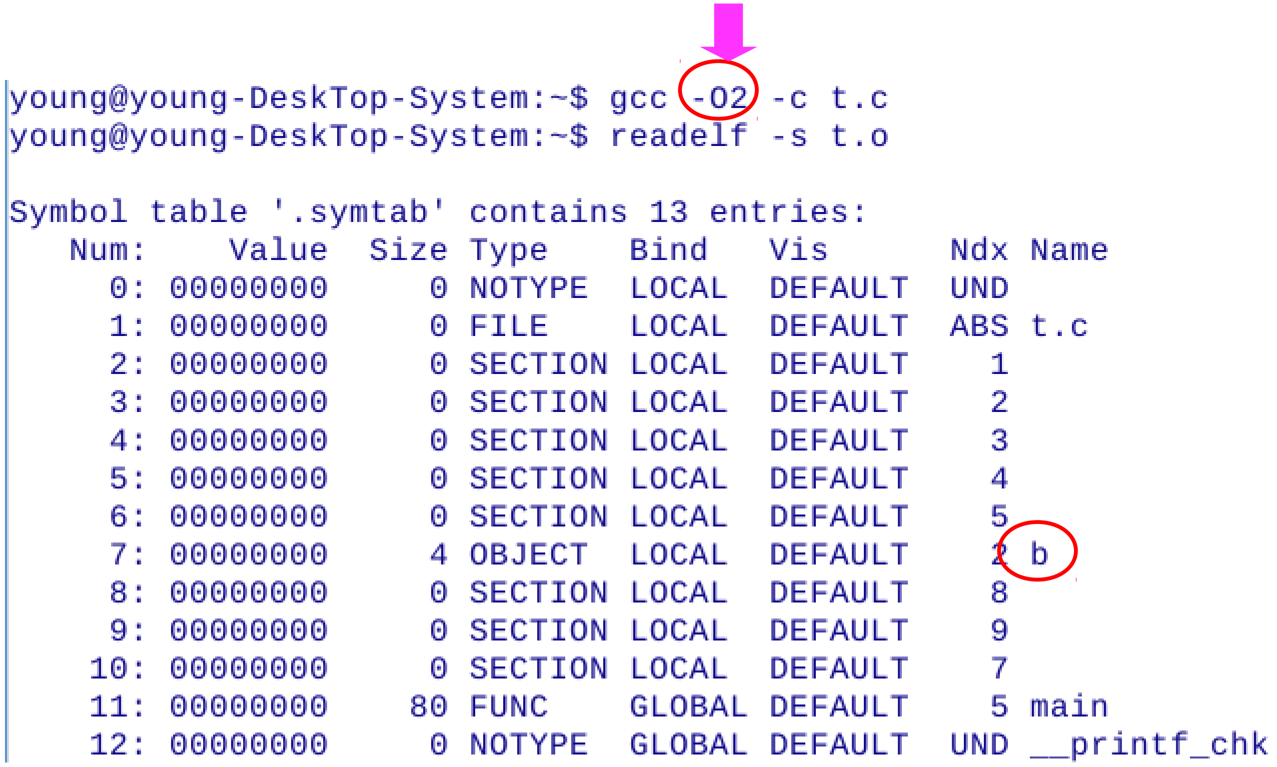


```
young@young-DeskTop-System:~$ gcc -c t.c
young@young-DeskTop-System:~$ readelf -s t.o

Symbol table '.symtab' contains 13 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FILE    LOCAL  DEFAULT  ABS t.c
     2: 00000000     0 SECTION LOCAL  DEFAULT    1
     3: 00000000     0 SECTION LOCAL  DEFAULT    3
     4: 00000000     0 SECTION LOCAL  DEFAULT    4
     5: 00000000     4 OBJECT  LOCAL  DEFAULT    3 a
     6: 00000004     4 OBJECT  LOCAL  DEFAULT    3 b
     7: 00000000     0 SECTION LOCAL  DEFAULT    5
     8: 00000000     0 SECTION LOCAL  DEFAULT    7
     9: 00000000     0 SECTION LOCAL  DEFAULT    8
    10: 00000000     0 SECTION LOCAL  DEFAULT    6
    11: 00000000    78 FUNC    GLOBAL DEFAULT    1 main
    12: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND printf
```

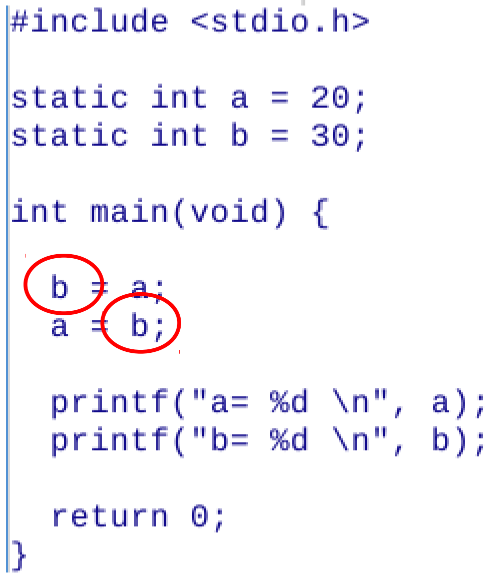


```
#include <stdio.h>

static int a = 20;
static int b = 30;

int main(void) {

  b = a;
  a = b;

  printf("a= %d \n", a);
  printf("b= %d \n", b);

  return 0;
}
```

Compile with optimization



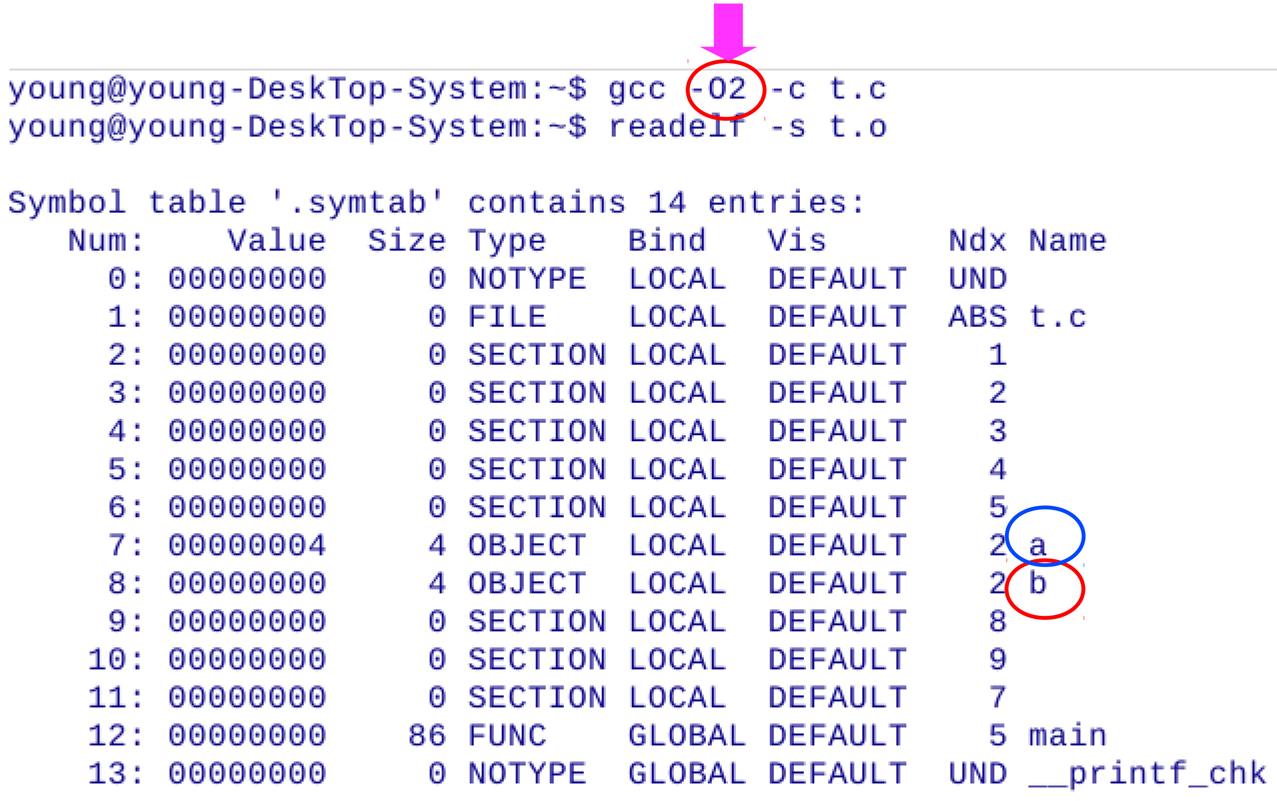
```
young@young-DeskTop-System:~$ gcc -O2 -c t.c
young@young-DeskTop-System:~$ readelf -s t.o

Symbol table '.symtab' contains 13 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FILE    LOCAL  DEFAULT  ABS t.c
     2: 00000000     0 SECTION LOCAL  DEFAULT    1
     3: 00000000     0 SECTION LOCAL  DEFAULT    2
     4: 00000000     0 SECTION LOCAL  DEFAULT    3
     5: 00000000     0 SECTION LOCAL  DEFAULT    4
     6: 00000000     0 SECTION LOCAL  DEFAULT    5
     7: 00000000     4 OBJECT  LOCAL  DEFAULT    2 b
     8: 00000000     0 SECTION LOCAL  DEFAULT    8
     9: 00000000     0 SECTION LOCAL  DEFAULT    9
    10: 00000000     0 SECTION LOCAL  DEFAULT    7
    11: 00000000    80 FUNC    GLOBAL DEFAULT    5 main
    12: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND __printf_chk
```



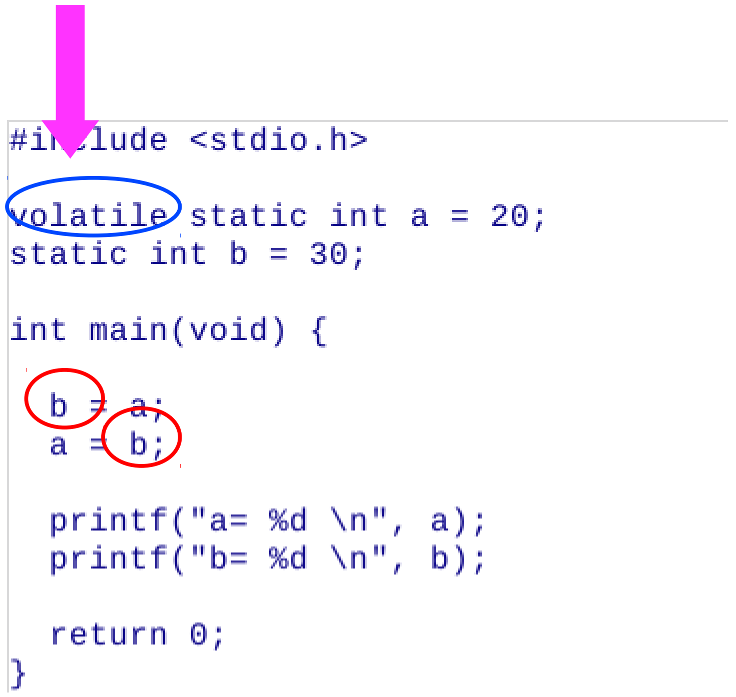
```
#include <stdio.h>

static int a = 20;
static int b = 30;

int main(void) {

  b = a;
  a = b;

  printf("a= %d \n", a);
  printf("b= %d \n", b);

  return 0;
}
```

- the variable a vanishes

With **volatile** and optimization



```
young@young-DeskTop-System:~$ gcc -O2 -c t.c
young@young-DeskTop-System:~$ readelf -s t.o

Symbol table '.symtab' contains 14 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FILE    LOCAL  DEFAULT  ABS t.c
     2: 00000000     0 SECTION LOCAL  DEFAULT    1
     3: 00000000     0 SECTION LOCAL  DEFAULT    2
     4: 00000000     0 SECTION LOCAL  DEFAULT    3
     5: 00000000     0 SECTION LOCAL  DEFAULT    4
     6: 00000000     0 SECTION LOCAL  DEFAULT    5
     7: 00000004     4 OBJECT  LOCAL  DEFAULT    2 a
     8: 00000000     4 OBJECT  LOCAL  DEFAULT    2 b
     9: 00000000     0 SECTION LOCAL  DEFAULT    8
    10: 00000000     0 SECTION LOCAL  DEFAULT    9
    11: 00000000     0 SECTION LOCAL  DEFAULT    7
    12: 00000000    86 FUNC    GLOBAL DEFAULT    5 main
    13: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND __printf_chk
```



```
#include <stdio.h>

volatile static int a = 20;
static int b = 30;

int main(void) {

  b = a;
  a = b;

  printf("a= %d \n", a);
  printf("b= %d \n", b);

  return 0;
}
```

- the variable a remains

const int

```
#include <stdio.h>


int main(void) {
  const int a = 10;

  a++;

  printf("a=%d\n", a);

}
```

```
gcc -Wall t.c
t.c: In function ‘main’:
t.c:7:4: error: increment of read-only
variable ‘a’
   a++;
    ^
```

```
#include <stdio.h>

void func(const int *p) {
  printf("a=%d\n", *p);
  ++*p;
}

int main(void) {
  int a = 10;

  func( &a );
}
```

```
gcc -Wall t.c
t.c: In function ‘func’:
t.c:7:3: error: increment of read-only
location ‘*p’
   ++*p;
   ^
```

const and pointers

```
int *   const      const_pointer;
```

a constant pointer

```
const      int      *pointer_to_const;
```

a constant integer

const type, const pointer type (1)

```
const int *p;
int * const q ;
const int * const r ;
```

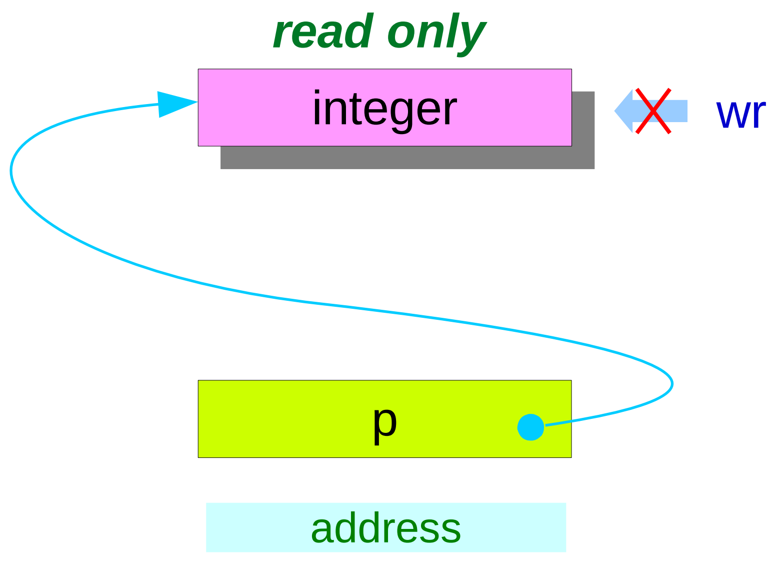
a constant integer

a constant pointer

const type, const pointer type (2)

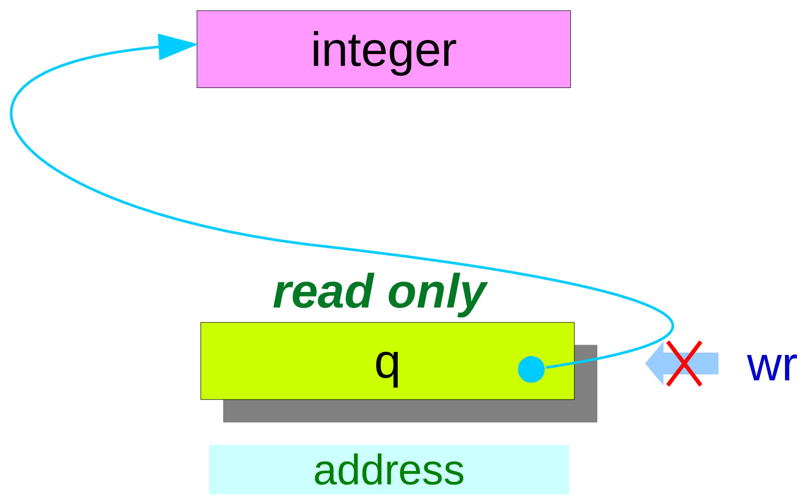
a constant integer

```
const int *p;
```



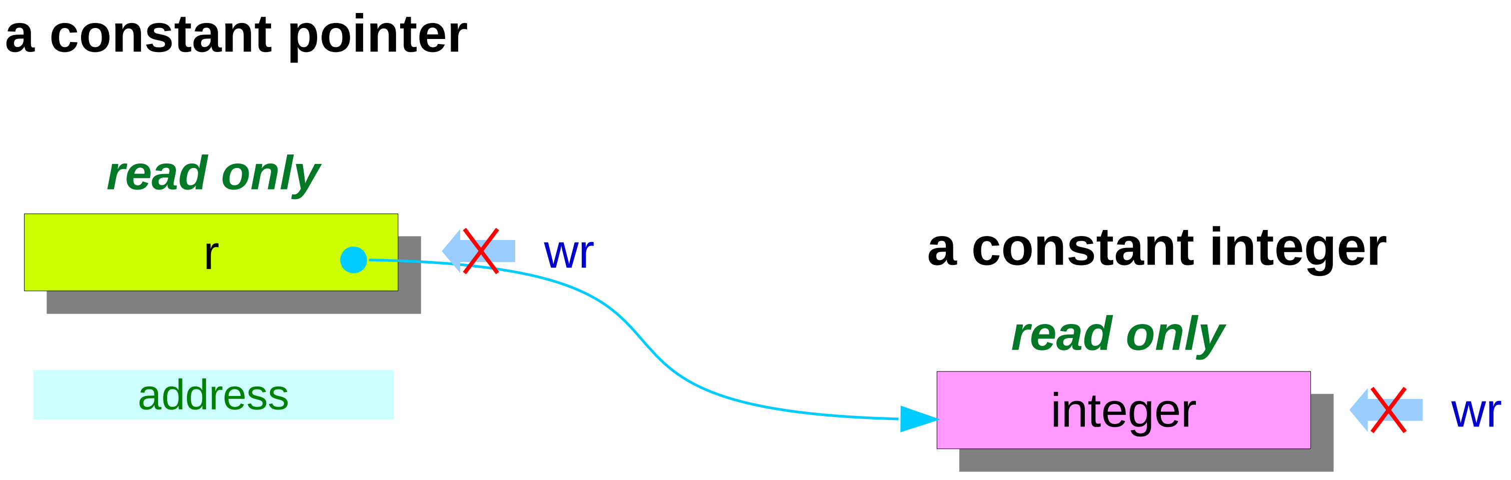
a constant pointer

```
int * const q ;
```



const type, const pointer type (3)

```
const int * const r;
```



References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun