THE PREPROCESSOR

Preprocessor

- C lang. uses the preprocessor to expend its power and notation
- *preprocessor directives*
  - lines beginning with a #
  - communicates with the preprocessor

#include

#define

Use of #include

#include <stdio.h>

#include <stdlib.h>

- The preprocessor looks for the file only in other places and not in the current directory.
- Where the standard header files are stored is system-dependent.

#include "filename"

- Search is made first in the current directory,
- and then in other system-dependent places

Use of #define

#define ***identifier token_string***$_{\text{opt}}$

- The preprocessor replaces every occurrence of ***identifier*** by ***token_string*** in the remainder of the file, except in quoted string.

```
#define SECONDS_PER-DAY  (60*60*24)
#define PI               3.14159
#define C                299792.458     /*speed of light in km/sec */
#define EOF              (-1)           /*typical end-of-file value */
#define MAXINT           2147483647     /*largest 4-byte integer */
```

- The use of simple **#define** can improve
  - program clarity
  - Program portability

Use of #define

- Syntactic sugar
  - alter the syntax of C toward users' preference

```
#define  EQ  ==

while (i EQ 1) {
  ...
```

⇒

```
while (i == 1) {
  ...
```

Macros with Arguments

- #define can be used to write macro definitions with parameters

  #define *identifier*(*identifier*, …, *identifier*) *token_string*$_{opt}$

  ```
  #define SQ(x)  ((x) * (x))
  SQ(7 + w)          ⇒   ((7 + w) * (7 + w))
  SQ(SQ(*p))         ⇒   ((((*p) * (*p))) * (((*p) * (*p))))

  #define SQ(x)  x * x
  SQ(a + b)          ⇒   a + b * a + b   ≠   ((a + b) * (a + b))

  #define SQ(x)  (x) * (x)
  4 / SQ(2)          ⇒   4 / (2) * (2)   ≠ 4 / ((2) * (2))
  ```

Macros with Arguments

```
#define SQ (x)  ((x) * (x))
SQ(7)          ⇒    (x)  ((x) * (x)) (7)


#define SQ(x)  ((x) * (x));   /* a common error */
x = SQ(y);     ⇒    x = ((y) * (y));;


if (x == 2)
   x = SQ(y);  ⇒    x = ((y) * (y));;
else
   ++x;
```

Macros with Arguments

- Macros are frequently used to replace function calls by inline code
  - **More Efficient !!!**

```
#define min(x, y)  (((x) < (y)) ? (x) : (y))
m = min(u, v);    ⇒    m = (((u) < (v)) ? (u) : (v));

#define min4(a, b, c, d)  min(min(a, b), min(c, d))
```

- A macro definition can use both functions and macros in its body.

```
#define SQ(x)       ((x) * (x))
#define CUBE(x)     (SQ(x) * (x))
#define F_POW(x)    sqrt(sqrt(CUBE(x))) /* fractional power: 3/4 */
```

Macros in stdio.h and ctype.h

[stdio.h]

```
#define getchar()   getc(stdin)
#define putchar(c)   putc((c), stdout)
```

[ctype.h]

Macro	Nonzero (true) is returned if:
isalpha(c)	c is a letter
isupper(c)	c is an uppercase letter
islower(c)	c is an lowercase letter
isdigit(c)	c is a digit
isalnum(c)	c is a letter or digit
isxdigit(c)	c is a hexadecimal digit
isspace(c)	c is a white space character
ispunct(c)	c is a punctuation character
isprint(c)	c is a printable character
isgraph(c)	c is printable, but not a space
iscntrl(c)	c is a control character
isascii(c)	c is an ASCII code

Macros in **stdio.h** and **ctype.h**

[ctype.h]

Call to the function or macro	Value returned
toupper(c)	corresponding uppercase value or c
tolower(c)	corresponding lowercase value or c
toascii(c)	corresponding ASCII value

- **c** is a variable of integral type, such as **char** or **int**.
- The value of c stored in memory does not get changed

Conditional Compilation

```
#define DEBUG 1
#if DEBUG
      printf("debug: a="%d"\n", a);
#endif

#define DEBUG
#ifdef DEBUG
      printf("debug: a="%d"\n", a);
#endif
```

Conditional Compilation

- #if
- #ifdef
- #ifndef
- #endif

- #undef *identifier*

Conditional Compilation

```
#if defined(HP9000)||defined(SUN4) && !defined(VAX)
      /* machine-dependent code */
#endif
```

__linux__	Defined on Linux
__sun	Defined on Solaris
__FreeBSD__	Defined on FreeBSD
__NetBSD__	Defined on NetBSD
__OpenBSD__	Defined on OpenBSD
__APPLE__	Defined on Mac OS X
__hpux	Defined on HP-UX
__osf__	Defined on Tru64 UNIX (formerly DEC OSF1)
__sgi	Defined on Irix
_AIX	Defined on AIX

Predefined Macros (ANSI C)

__DATE__ 매크로가 치환되는 순간의 날짜 (문자열)
__TIME__ 매크로가 치환되는 순간의 시간 (문자열)
__STDC__ ANSI 표준을 따르는지 여부 (0 또는 1)
__FILE__ 매크로가 포함된 source file
__LINE__ source file에서 line number