C Preprocessor

- C preprocessor manipulates ***text***

  file inclusion

  macros

  conditional compilation

- Invoked automatically by the C compiler

  try "**`lcc -E foo.c`**"

- File inclusion

  header files contain declarations for one or more C program files

  names of header file should end in "**`.h`**"

  specify system header filename with **`<...>`** — ***location independent***

```
#include <stdio.h>

#include "defs.h"
```

Macros

- Macros provide parameterized text substitution
- Macro ***definition***

```
#define MAXLINE 120


#define lower(c) ((c)-'A'+'a')
```

- Macro ***replacement***

```
char buf[MAXLINE+1];   becomes   char buf[120+1];
c = lower(buf[i]);               c = ((buf[i])-'A'+'a');
```

More on Macros

```
#define plusone(x) x+1
```

- Good idea?

```
i = 3*plusone(2);        becomes     what?
```

  - use parentheses liberally:

```
      #define plusone(x) ((x)+1)
      #define max(a,b) ((a)>(b)?(a):(b))
   y = max(i++,j++);        becomes    y = ((i++)>(j++)?(i++):(j++));
```

  - avoid macros that use an argument more than ***once***

Conditional Compilation

- Removing macro definitions

```
#undef plusone
```

- Conditional compilation

```
#ifdef name
#ifndef name
#if expr
#elif
#else
#endif
```

  - header file **`defs.h`**:

```
   #ifndef DEFS_INCLUDED
   #define DEFS_INCLUDED

   #define MAXLINE 120
...
   #endif
```

- Use conditional compilation ***sparingly***

Formatted Output

- **`int printf(const char *format, ...)`**
- Writes **`format`** to stdout with the values of argument 2, 3, ..., ***n***
  `printf("error at line %d: %s\n", lineno, msg);`
- ***<u>Beware</u>***:

  <u>no</u> type checking
  <u>no</u> type conversion
  <u>no</u> checking for the correct number of arguments
- Conversion specifications

`%d`	decimal	plus other formatting control, e.g. field widths
`%o`	octal	
`%x`	hexadecimal	
`%c`	char	
`%u`	unsigned	
`%e %f %g`	float & double	
`%s`	string	
`%%`	%	

- See **`sprintf`**, **`fprintf`**, **`vfprintf`**; and see **`scanf`**, **`fscanf`**, **`sscanf`** for ***<u>input</u>***

Basic Input/Output Functions

- C language has no I/O statements; use standard C ***<u>library</u>***
  - "**`#include <stdio.h>`**" before using the functions
  - **`int getchar(void)`** returns the next character in "stdin" or **`EOF`**
  - **`getchar()`** returns an **`int`**, not a **`char`**, to accommodate **`EOF`**
  - **`int putchar(int c)`** writes character **`c`** to the "stdout" and returns **`c`**
- A basic copy program:

```
#include <stdio.h>

int main(void) {
    int c;

    while ((c = getchar()) != EOF)
            putchar(c);
    return 0;
}

% copy < foo.c > bar.c
% copy < foo.c
% copy > bar.c
```