Preprocessor

Preprocessor Topics

- C Preprocessor
- Constant Expressions
- Miscellaneous Directives
- Macros Expansion

C Preprocessor

- Traditionally, a separate process that executes before the C compiler
- Usually, output from the preprocessor automatically feeds the compiler

C Preprocessor

- Sometimes, it is helpful to capture and examine the output the C preprocessor
  - Consult your compiler reference
- In some environments, utilities other than the C compiler use the C preprocessor
  - Consider portability issues prior to incorporating the C preprocessor into you own applications
  - Consider obtaining the GNU C preprocessor

C Preprocessor

- Preprocessor Responsibilities
  - Comment Suppression
    - *Each comment is replaced by a single space*
  - Trigraph substitution

#	??=	[	??(	{	??<
\	??/	]	??)	}	??>
^	??'	|	??!	~	??-

C Preprocessor

- Preprocessor Responsibilities
  - Preprocessor directive execution
    - #if, #include, etc
  - Macro expansion

Constant Expressions

- #if arg
  - arg must be a constant expression
  - constant expression may include the defined preprocessor
  - Example

```
#if  CHECK_LEVEL  ==  2
  if  (  target  <  0  )
    abort_prog(  FILE,  LINE  );
#elif  CHECK_LEVEL  ==  1
  if  (  target  <  0  )
    printf("Error: %s %d\n", __FILE__,__LINE__ );
#else
  result  =  sqrt(  target  );
#endif
```

Constant Expressions

- #if arg
  - Example

```
#if  defined(VAX) || defined(ALPHA)
#define  FILE_TYPE         (0)
#elif  defined(  MSDOS  )
#define  FILE_TYPE         (1)
#else
#define  FILE_TYPE         (2)
#endif
```

Miscellaneous Directives

- #line
  - #line *num*
  - #line *num "filenname"*
  - *Sets the value of __LINE__ and __FILE__*
  - *Mainly useful if you're writing a C code generator*
  - #line 99 "proj1.c"
- #error
  - *Prints a user-defined diagnostic*
  - #if defined(VAX) && OPTION2 == 1
  - #error "OPTION2 not valid for VAX"
  - #endif
- #pragma
  - *Used for environment-dependent features*
  - #ifdef VAX
  - #pragma builtins
  - #endif
- #(null directive)
  - *Directly analogous to the C null statement*

Miscellaneous Directives

- **# preprocessor operator**
  - *A # preceding a macro argument* stringizes *the argument in the macro expansion*
  - Example

```
#define PRINT_VAL(a)      \
            (printf( #a " = %d\n", (a)))
  . . .
 inx  =  42;
 PRINT_VAL(inx);
 PRINT_VAL(inx+1);
Expands to ...
 (printf( "inx"  "  =  %d\n",  (inx)));
 (printf( "inx+1"  “ = %d\n", (inx+1)));
```

Miscellaneous Directives

- **## preprocessor operator**
  - *A ## between two tokens in a macro definition causes the tokens to be concatenated in the macro expansion*
  - Example

```
#define MENU_DEF(  n,  l,  s  )       \
  char  *n  ##  _name  =  #n;         \
  char  *n  ##  _label  =  #l;        \
  int  n  ##  _state  =  s
   . .
MENU_DEF(  circle,  Circle,  TRUE  );
Expands to ...
char  *circle_name  =  "circle";
char  *circle_label  =  "Circle";
int     circle_state  =  TRUE;
```

Macro Expansion

- **The expansion of a macro replaces the macro in the source code**
- **The expansion of a macro is rescanned for more macros**

```
#define PI                 (3.14159)
#define AREA_CIRC( r )     ((r)*(r)*PI)
    . . .
  area = AREA_CIRC( new_rad );
Preprocessing step 1...
  area = ((new_rad)*(new_rad)*PI);
Preprocessing step 2...
  area = ((new_rad)*(new_rad)*(3.14159) );
```

Macro Expansion

- **A macro expansion is not treated as a preprocessor directive even if it resembles one**
- **The name of a macro in the expansion of the macro is not subject to replacement**

```
#define malloc(  a  )                                  \
  (  log(  "malloc",  __FILE__,  __LINE__  ),          \
     malloc(  (a)  )                                   \
  )
   . . .
  inx  =  malloc(  10  *  sizeof(  int  )  );
```

Expands to ...

```
  inx  =  (  log(  "malloc",  "proj1.c",  157  ),
              malloc(  10  *  sizeof(int)  )
           );
```