Preprocessor Directives in C

Computing Lab

https://www.isical.ac.in/~dfslab

Indian Statistical Institute

General rules

- Can be used *anywhere* in a program
- Affect the remaining part of the program
- Syntax: # must be the *first non-whitespace character* in the line

Including files

- Syntax:

```
#include <stdio.h>   /* for include files provided by the system */
#include "common.h"  /* for your "own" header files */
```

- Effect $\equiv$ copy-paste (replace the line by the entire contents of the file)
- Typically contain `#define`s, `typedef`s, variable and function declarations
- Additional directories to search for included files specified as follows

```
gcc -I/path/to/directory1 -I/path/to/directory2 ...
```

Defining constants

- Syntax

```
/* From stdio.h */
# define EOF (-1)

/* From stdint.h: minimum of signed integral types.  */
# define INT8_MIN               (-128)
# define INT16_MIN              (-32767-1)
# define INT32_MIN              (-2147483647-1)
# define INT64_MIN              (-__INT64_C(9223372036854775807)-1)
/* Maximum of signed integral types.  */
# define INT8_MAX               (127)
# define INT16_MAX              (32767)
# define INT32_MAX              (2147483647)
# define INT64_MAX              (__INT64_C(9223372036854775807))
```

- Effect $\equiv$ find-and-replace
  - replace the first string whenever it occurs as a complete word by the second string

Defining macros

- Syntax

```
#define MAX(a,b) (((a)>(b))?(a):(b))
#define MIN(a,b) (((a)<(b))?(a):(b))
#define SQR(x) ((x) * (x))

#define ERR_MESG(x) { perror(x); return -1; }

#define Malloc(n,type) (type *) malloc( (unsigned) ((n)*sizeof(type)))
#define Realloc(loc,n,type) (type *) realloc((char *)(loc), \
                                              (unsigned) ((n)*sizeof(type)))
```

- Effect
  - replace call to a macro by body of the macro
  - *string-replace* each parameter by the corresponding argument

See review question 2.

Conditional compilation: `ifdef`, `ifndef`

- Syntax

```
#ifdef DEBUG /* if defined */
    fprintf(stderr, "Some error message\n");
#endif

#ifndef EOF  /* if NOT defined */
# define EOF (-1)
#endif
```

- Alternative to commenting and uncommenting large portions of code
- To define a constant at compile time (without changing source file)
  `gcc -DDEBUG ...`
- See review question 1.

Conditional compilation: `if`

- Syntax

```
#if (DEBUG_LEVEL >= DETAILED) && defined(NAME) && undefined_name
    fprintf(stderr, "Some low level error message\n");
#endif
```

- Can use any *constant expressions* of integer type
- Expression can contain
  - integer / character constants
  - standard arithmetic / logical operators
  - macros
  - `defined` operator
  - identifiers (zero value)
- **DO NOT USE** `sizeof` operator

NOTE: All calculations use widest integer type known to the compiler (typically 64 bits)

Review questions I

1. What happens if a file a.h includes b.h and vice versa?
2. Why does the following not work as expected?

```
#define SQR(x) (x * x)

int x = 5, y;
y = SQR(x+1);
printf("%d\n", y);
```