

*Intro to Programming II*  
*Compiling and debugging*

Chris Brooks

Department of Computer Science  
University of San Francisco

## 24-2: Stages of Compilation

---

- What are the stages of the compilation of a program?

## 24-3: Stages of Compilation

---

- What are the stages of the compilation of a program?
- Lexing - separate source code into tokens.
- Parsing - evaluate statements and generate assembly code
- Assembly - transform assembly code into binary code.
- Linking - Multiple object files are merged into a single executable.

## 24-4: Example

---

- Assume we start with two source files: file1.c and file2.c
- First, each file is parsed and compiled separately.
- The linker then builds a single executable
  - Resolves all function calls and variable references to a single address space.
  - Builds an executable with a single “main” entry point.

## 24-5: Stages of Compilation

---

- The C compiler adds another step at the beginning of the process.
- This step is called preprocessing.
- The preprocessor does three things:
  - Strips out comments.
  - “Pastes in” included headers.
  - Expands macros.

## 24-6: Included headers

---

- In C, headers are included directly in the source code at the very first stage of compilation.
- This can sometimes lead to strange compiler errors.
  - File A includes file B, and file B includes file A.

## 24-7: Macros

---

- The C preprocessor also uses a `#define` directive.
- This is used to:
  - Define constants
  - Define macros: small chunks of code.

## 24-8: Defining constants

---

```
#define ARRAYSIZE 100
```

```
int arr[ARRAYSIZE];
```

- The preprocessor replaces ARRAYSIZE with 100
- Note: this is NOT a variable - the preprocessor just replaces the string 'ARRAYSIZE' with the string '100'

## 24-9: Macros

---

- We can also do this syntactic replacement with blocks of code.

```
#define SWAP(a,b) {int temp = a; a = b; b = temp;}
int main(void) {
    int x = 10;
    int y = 20;

    printf("%d %d", x,y);
    SWAP(x,y);
    printf("%d %d", x,y);
}
```

- All occurrences of SWAP are replaced with the corresponding code.

## 24-10: Macros

---

- Note that SWAP is not a function.
  - Code is just cut-and-pasted.
  - No checking of argument types
  - No return types
- Advantages: useful for 'quick and dirty' operations, no overhead from a function call.
- Disadvantages: Can be difficult to debug.

## 24-11: The Symbol table

---

- Normally, gcc constructs a symbol table during the compilation process.
- This is used to map variable names to addresses.
- Typically, the symbol table is discarded after object files are generated.
- Compiling with the -g option retains the symbol table.
- This allows us to run the debugger.

## 24-12: GDB

---

- GDB is the GNU debugger.
- This is very helpful for debugging C programs.
- Same features as the eclipse debugger.
  - Break, run, list, print variables, step.

## 24-13: GDB

---

- break - stop at:
  - a particular line
  - a particular function.

## 24-14: GDB

---

- run - start a program
- continue - continue execution from the breakpoint.
- step - move to the next line of the program

## 24-15: GDB

---

- `print` - show the current value of a variable
- `list` - show the current lines in the program.