Object-Oriented Programming: Debugging

Lectured by Ming-Te Chi 紀明德

First Semester, 2023

Computer Science Department
National Chengchi University

Slides credited from 李蔡彥 and 廖峻鋒

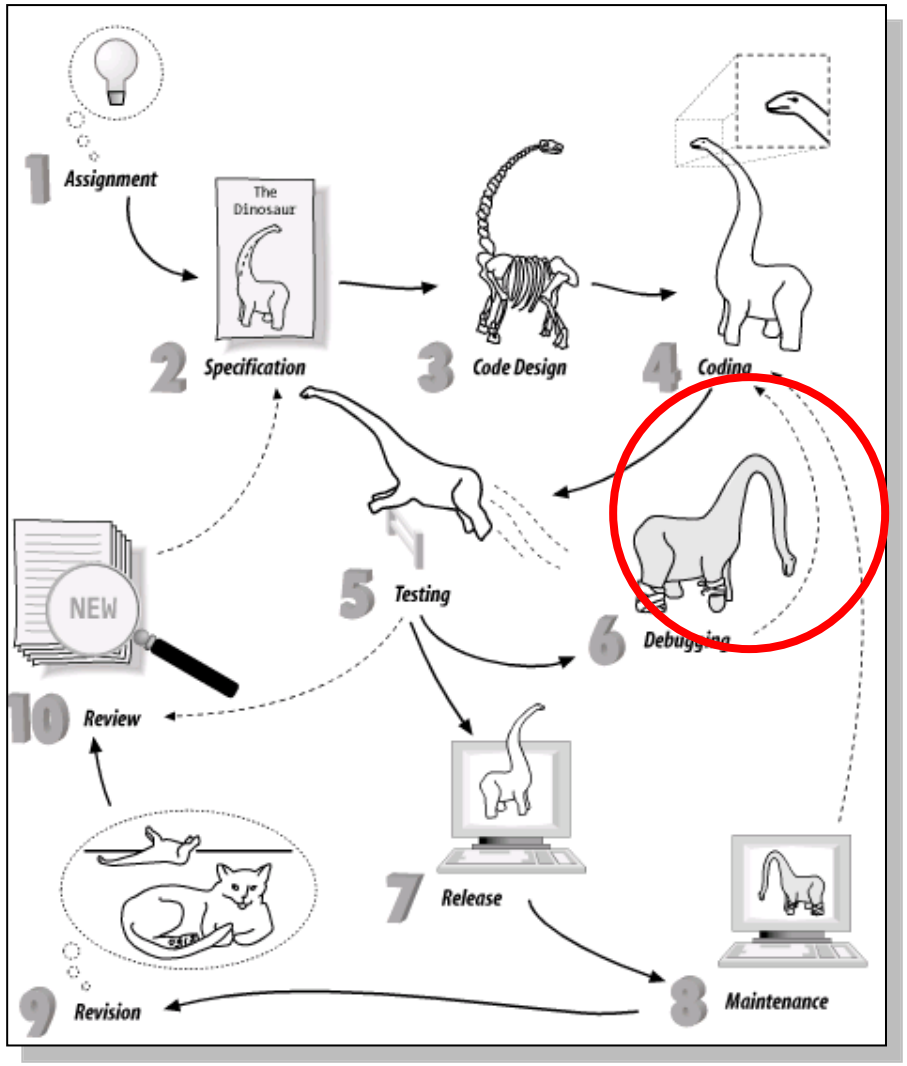
The Process of Language Translation

- Preprocessor
  - 引入檔案、處理巨集，例如: #include <iostream>
- Compiler
  - 將個別檔案的原始碼轉換為object code (和物件導向的object無關)
    - 1st pass: building syntax tree
      - Global optimization
    - 2nd pass: code generator
      - Peephole optimization: looking for redundant statements
- Linker
  - 將個別模組的object code與函式庫連結起來，產生機器碼(可執行檔)
    - #include <iostream> : 代表要連結iostream這個函式庫
  - Dynamic linking : 如dll或Java，執行時才連結

寫完程式碼之後

- 先用肉眼檢查程式碼
- 編譯
- 測試
  - 黑箱測試
  - 白箱測試
- 藉由測試案例找出Bug

Software life cycle



Bugs

- Logical Errors
- Runtime Errors
  - Segmentation violation:
    dereferencing a pointer containing a bad value
  - Stack overflow:
    using too many local variables
  - Divide by zero:
    Arithmetic overflow

當測試發現有Bug時

- Segmentation fault or core dump
  - 一行一行印出訊息以找出是哪行造成程式被中斷
- 程式結果不正確
  - 猜測可能有Bug的地方, 印出變數的值並檢查是否符合預期
- 原始碼中充滿一堆印訊息的程式碼
- 每次新增印訊息的程式碼都要重新編譯和執行

使用Debugger的好處

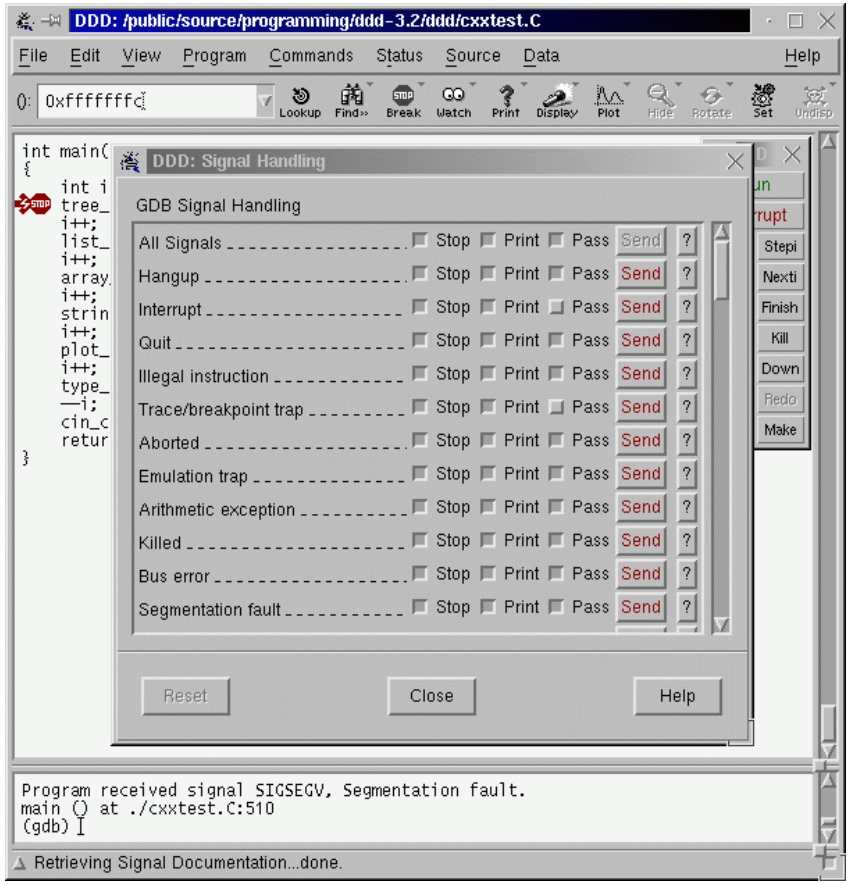
- 直接在除錯器中執行程式以監控
  - 可以設定中斷點
  - 可以在除錯器中印出程式目前的狀態
  - 逐行執行
- 不用加入一堆印除錯訊息的程式碼

How to debug?

- Use `printf`

```
#ifdef DEBUG
    printf("i = %d\n", i);
#endif
```

- Use interactive Debugger
  - GDB: GNU Debugger
    - Most popular
  - DDD: Data Display Debugger
    - GUI version of GDB
    - Easier to use, but slow



GDB overview

- GDB, the GNU Project debugger
- The most popular debugger for UNIX systems
- Program crashes!!
  - Leaves a “core” file
    - Memory dump of the program
- Documentation
  - Debugging with GDB

  http://www.gnu.org/software/gdb/documentation/
  - `$ man gdb`

Starting GDB

- Symbol table for GDB
  - `$ gcc -g filename.c`
- Common usage
  - `$ gdb binary`
  - `$ gdb binary core`

GDB: Essential Commands

- `run` *`[arglist]`*
  - run your program (with arglist)

- `list [filename:]line_number`
  `list function`
  `list from,[ to]`
  - display source code

- `print expr`
  - print the result of an expression

GDB: Essential Commands

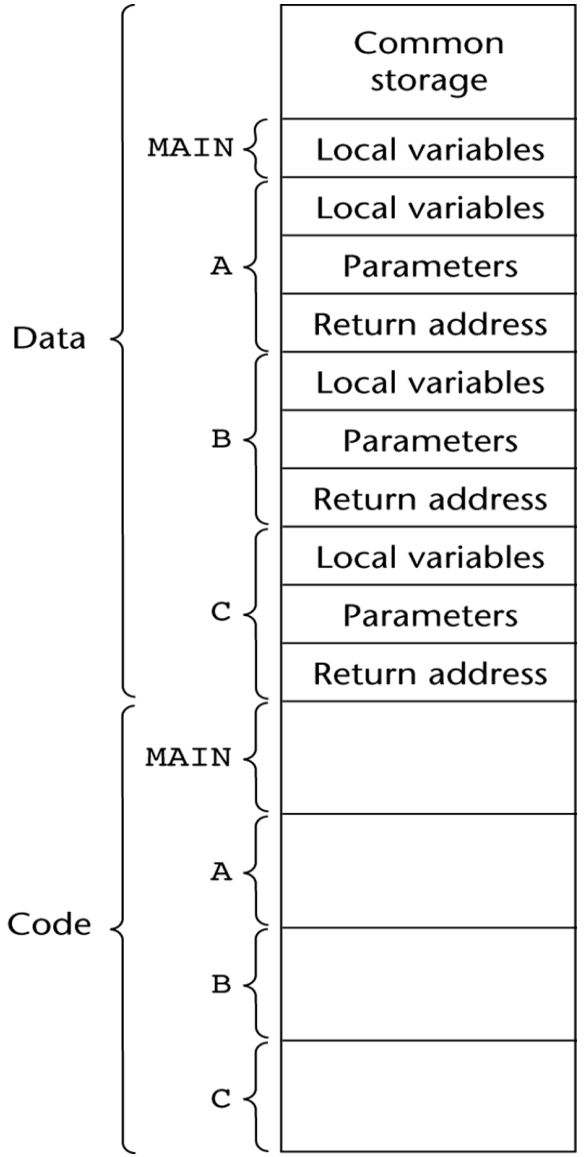
- `break [file:]line_number`
  `break class::member_fn`
  `break function`
  - set breakpoint

- `delete [bp_number|range]`
  - delete breakpoints

- `disable [bp_number|range]`
- `enalbe [bp_number|range]`
- `ignore bp_number iterations`

GDB: Essential Commands

- `next, step`
  - next instruction/line
- `c`
  - continue execution
- `help`
  - built-in help
- `quit`
  - exit from GDB



gdb_example.c

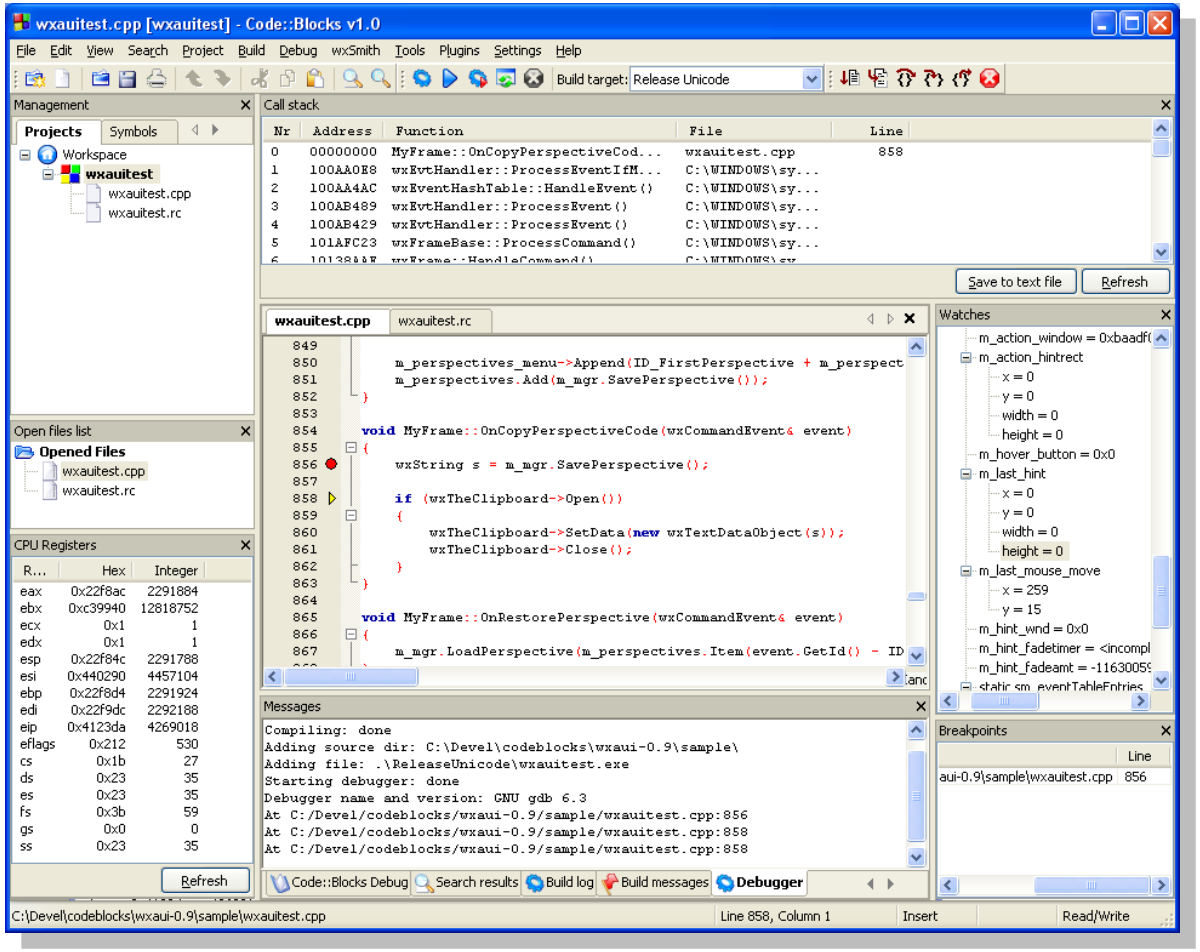
```
// gdb_example.c:
// Test the swap( ) function, which exchanges the contents of two int variables.
// ------------------------------------------------------------
#include <stdio.h>
void swap(int*, int*);

int main() {
  int a = 10, b = 20;
  /* ... */
  printf("The old values: a = %d; b = %d.\n", a, b);
  swap(&a, &b);
  printf("The new values: a = %d; b = %d.\n", a, b);
  /* ... */
  return 0;
}

void swap(int *p1, int *p2) {
  int* tmp = p1;
  p1 = p2;
  p2 = tmp;
}
```

Debugging in IDE with GDB

- Code::Blocks https://www.codeblocks.org/



作業注意事項

- 準時繳交作業
- 遲交 *vs*. 屍體 *vs*. 沒交
- 嚴禁抄襲!! Honor Code
- 評分標準
  - Correctness
  - Robustness
  - Coding Style
  - Comments
  - Etc..
- Problem ? → moodle.nccu.edu.tw

Book Reference

- For Beginner:
  - C++ Primer Plus (Sams)
  - C++ How to Program
  - C++ in a Nutshell (O'Reilly) 2003
  - Tour of C++ (3rd, 2020)
- Bibles:
  - The C++ Programming Language (4th, 2013)
  - C++ Primer (5th, 2012)
- For Advance:
  - Effective C++ (3ed, 2005)
  - More Effective C++ (1995)
  - Effective Modern C++ (1st, 2014)

Book Reference

- C++ In-Depth Series (Addison Wesley)
  1. C++ Template Meta-programming
  2. C++ Coding Standards
  3. Exceptional C++ Style
  4. Applied C++
  5. C++ Network Programming, Volume 2
  6. Boost Graph Library
  7. More Exceptional C++
  8. C++ Network Programming, Volume I
  9. Modern C++ Design
  10. Accelerated C++:
  11. Exceptional C++
  12. Essential C++