Chapter - 17 Debugging and Optimization

Debugging Techniques

- Divide and conquer
- Debug only code
- Debug Command Line Switch

  Note: Use I/O redirection and the editor to browse large volumes of debug output.
- Interactive Debuggers

Gnu Debugger (gdb) commands:

`run` Start execution of a program.

`break` *`line-number`*

Insert a breakpoint at the given line number.

`break` *`function-name`*

Insert a breakpoint at the first line of the named function.

`cont` Continue execution after a breakpoint.

`print` *`expression`*

Display the value of an expression.

`step` Execute a single line in the program.

`next` Execute a single line in the program, skip over function calls.

`list` List the source program.

`where` Print the list of currently active functions.

`status` Print out a list of breakpoints

`delete` Remove a break point.

A program to debug

```
 1 #include <iostream>
 2 int seven_count;    /* number of seven's in the data */
 3 int data[5];        /* the data to count 3 and 7 in */
 4 int three_count;    /* the number of threes in the data */
 5
 6 main() {
 7     int index;  /* index into the data */
 8     void get_data(int data[]);
 9
10     seven_count = 0;
11     three_count = 0;
12     get_data(data);
13
14     for (index = 1; index <= 5; ++index) {
15         if (data[index] == 3)
16             ++three_count;
17         if (data[index] == 7)
18          ++seven_count;
19     }
20     std::cout << "Three's " << three_count <<
21          " Seven's " << seven_count << '\n';
22     return (0);
23 }
24 /*******************************************************
25  * get_data -- get 5 numbers from the command line   *
26  *******************************************************/
27 void get_data(int data[])
28 {
29     std::cout << "Enter 5 numbers\n";
30     std::cin >> data[1] >>data[2] >>data[3] >>data[4] >>data[5];
31 }
```

First Debugging Session

```
When we run this program with the data 3 7 3 0 2 the results are:
Threes 3 Sevens 3
GDB run:.
 % gdb count
GDB is free software and you are welcome to distribute
 copies of it
 under certain conditions; type "show copying" to see
the conditions.
There is absolutely no warranty for GDB; type
"show warranty" for details.
GDB 4.12 (m68k-sun-sunos4.0.3),
Copyright 1994 Free Software Foundation, Inc...
 (gdb) break main
Breakpoint 1 at 0x22c2: file count.cc, line 10.
 (gdb) run
Starting program: /usr/sdo/count/count


Breakpoint 1, main () at count.cc:10
10          seven_count = 0;
 (gdb) next
11          three_count = 0;
(gdb) print seven_count
$1 = 0
 (gdb) next
12          get_data(data);
 (gdb) print seven_count
$2 = 0
 (gdb) next
Enter 5 numbers
3 7 3 0 2
14          for (index = 1; index <= 5; ++index) {
 (gdb) print seven_count
$3 = 2


 All right how did seven_count get to 2?
```

Figuring out what happened

The value got change in get_data. Let’s look through it.

```
 (gdb) break get_data
Breakpoint 2 at 0x23b2: file count.cc, line 29.
 (gdb) info breakpoints
Num Type           Disp Enb Address    What
1   breakpoint     keep y   0x000022c2 in main at count.cc:10
2   breakpoint     keep y   0x000023b2 in get_data(int *) at count.cc:29
 (gdb) delete 1
 (gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /usr/sdo/count/count
Breakpoint 2, get_data (data=0x208f8) at count.cc:29
29          std::cout << "Enter 5 numbers\n";
 (gdb) print seven_count
$5 = 0
 (gdb) next
30          std::cin >> data[1] >> data[2] >> data[3] >> data[4] >> data[5];
 (gdb) print seven_count
$6 = 0
 (gdb) next
Enter 5 numbers
3 7 3 0 2
31      }
 (gdb) print seven_count
$7 = 2
 (gdb) list 30
27      void get_data(int data[])
28      {
29         std::cout << "Enter 5 numbers\n";
30         std::cin >>data[1] >>data[2]>>data[3]>>data[4]>>data[5];
31      }
```

Binary Search

```
/*****************************************************
 * search -- Search a set of numbers.                *
 *                                                   *
 * Usage:                                            *
 *      search                                       *
 *               you will be asked numbers to lookup *
 *                                                   *
 * Files:                                            *
 *      numbers.dat -- numbers 1 per line to search  *
 *                     (Numbers must be ordered)     *
 *****************************************************/
#include <iostream>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>


const int MAX_NUMBERS = 1000;   // Max numbers in file
const char *const DATA_FILE = "numbers.dat";// File with numbers


int data[MAX_NUMBERS];  // Array of numbers to search
int max_count;          // Number of valid elements in data
```

```
main()
{
    ifstream in_file;   // Input file
    int middle;         // Middle of our search range
    int low, high;      // Upper/lower bound
    int search;         // number to search for

    in_file.open(DATA_FILE, ios::in);
    if (in_file.bad()) {
        cerr << "Error:Unable to open " << DATA_FILE << '\n';
        exit (8);
    }

    /*
     * Read in data
     */

    max_count = 0;
    while (1) {
        char line[30];  // Line from the input file

        if (in_file.eof())
            break;

        in_file.getline(line, sizeof(line));

        sscanf(line, "%d", data[max_count]);
        if (data[max_count] == -1)
            break;

        ++max_count;
    }
```

```
    while (1) {
        std::cout << "Enter number to search for or -1 to quit:" ;
        std::cin >> search;

        if (search == -1)
            break;

        low = 0;
        high = max_count;

        while (1) {
            middle = (low + high) / 2;

            if (data[middle] == search) {
                std::cout << "Found at index " << middle << '\n';
            }

            if (low == high) {
                std::cout << "Not found\n";
                break;
            }

            if (data[middle] < search)
                low = middle;
            else
                high = middle;
        }
    }
    return (0);
}
```

Data

```
4
6
14
16
17
-1
```

When we run this program on UNIX, the results are:

```
% search
Segmentation fault (core dumped)
```

Debugging Session

```
 % gdb search
GDB is free software and ....
GDB 4.12 (m68k-sun-sunos4.0.3),
Copyright 1994 Free Software Foundation, Inc...
 (gdb) run
Starting program: /usr/sdo/search/search

Program received signal SIGSEGV, Segmentation fault.
0xec46320 in number ()
(gdb)
 (gdb) where
#0  0xec46320 in number ()
#1  0xec45cc2 in _doscan ()
#2  0xec45b34 in sscanf ()
#3  0x2400 in main () at search.cc:48
(gdb)
 (gdb) list 48
43              if (in_file.eof())
44                  break;
45
46              in_file.getline(line, sizeof(line));
47
48              sscanf(line, "%d", data[max_count]);
49              if (data[max_count] == -1)
50                  break;
51
52              ++max_count;
 (gdb) quit
The program is running.  Quit anyway (and kill it)? (y or n) y
```

```
% gdb search
GDB is free software ...
GDB 4.12 (m68k-sun-sunos4.0.3),
Copyright 1994 Free Software Foundation, Inc...
 (gdb) list main
18      const char *const DATA_FILE = "numbers.dat";// File with numbers
19
20      int data[MAX_NUMBERS];  // Array of numbers to search
21      int max_count;          // Number of valid elements in data
22      main()
23      {
24          ifstream in_file;   // Input file
25          int middle;         // Middle of our search range
26          int low, high;      // Upper/lower bound
27          int search;         // number to search for
 (gdb) break main
Breakpoint 1 at 0x2318: file search.cc, line 24.
 (gdb) run
Starting program: /usr/sdo/search/search
Breakpoint 1, main () at search.cc:24
24          ifstream in_file;   // Input file
 (gdb) step
29          in_file.open(DATA_FILE, ios::in);
 (gdb) step
30          if (in_file.bad()) {
 (gdb) step
39          max_count = 0;
 (gdb) step
43              if (in_file.eof())
 (gdb) step
46              in_file.getline(line, sizeof(line));
 (gdb) step
48              sscanf(line, "%d", data[max_count]);
 (gdb) step
Program received signal SIGSEGV, Segmentation fault.
0xec46320 in number ()
 (gdb) quit
 The program is running.  Quit anyway (and kill it)? (y or n) y
```

If at first you don't succeed, play second base.

We try again:

```
Enter number to search for or -1 to quit:4
Found at index 0
Found at index 0
Not found
Enter number to search for or -1 to quit:^C
```

```
 % gdb search
GDB is free software ...
 (gdb) list 66,77
66              while (1) {
67                  middle = (low + high) / 2;
68
69                  if (data[middle] == search) {
70                      std::cout << "Found at index " << middle << '\n';
71                  }
72
73                  if (low == high) {
74                      std::cout << "Not found\n";
75                      break;
76                  }
77
 (gdb) break 70
Breakpoint 1 at 0x249e: file search.cc, line 70.
 (gdb) run
Starting program: /usr/sdo/search/search
Enter number to search for or -1 to quit:4
Breakpoint 1, main () at search.cc:70
70                      std::cout << "Found at index " << middle << '\n';
 (gdb) step
Found at index 0
73                  if (low == high) {
 (gdb) step
78                  if (data[middle] < search)
 (gdb) step
81                      high = middle;
 (gdb) step
67                  middle = (low + high) / 2;
 (gdb) step
69                  if (data[middle] == search) {
 (gdb) step
70                      std::cout << "Found at index " << middle << '\n';
 (gdb) step
Found at index 0
73                  if (low == high) {
```


The Mistake

```
if (data[middle] == search) {
        std::cout << "Found at index " << middle << '\n';
    }
```

Changes to:

```
    if (data[middle] == search) {
        std::cout << "Found at index " << middle << '\n';
        break;
    }
```

Try again:

```
 % search
 Enter number to search for or -1 to quit:4
Found at index 0
 Enter number to search for or -1 to quit:6
Found at index 1
 Enter number to search for or -1 to quit:3
Not found
 Enter number to search for or -1 to quit:5
program runs forever (or until we abort it)
```

Debug Session III

```
 % gdb search
GDB is free software and ...
 (gdb) run
Starting program: /usr/sdo/search/search
 Enter number to search for or -1 to quit:5
^C
Program received signal SIGINT, Interrupt.
0x2500 in main () at search.cc:79
79                  if (data[middle] < search)
79                  if (data[middle] < search)
 (gdb) print middle
$1 = 0
 (gdb) print data[middle]
$2 = 4
 (gdb) print search
$3 = 5
 (gdb) step
80                      low = middle;
 (gdb) step
67                  middle = (low + high) / 2;
 (gdb) step
69                  if (data[middle] == search) {
 (gdb) step
74                  if (low == high) {
 (gdb) step
79                  if (data[middle] < search)
 (gdb) step
80                      low = middle;
 (gdb) step
67                  middle = (low + high) / 2;
 (gdb) step
69                  if (data[middle] == search) {
 (gdb) step
74                  if (low == high) {
 (gdb) step
79                  if (data[middle] < search)
 (gdb) step
80                      low = middle;
 (gdb) step
67                  middle = (low + high) / 2;
```

Debugging Cont.

```
 (gdb) step
69                  if (data[middle] == search) {
 (gdb) step
74                  if (low == high) {
 (gdb) step
79                  if (data[middle] < search)
 (gdb) step
80                      low = middle;
 (gdb) step
67                  middle = (low + high) / 2;
 (gdb) step
69                  if (data[middle] == search) {
 (gdb) print low
$5 = 0
 (gdb) print middle
$6 = 0
 (gdb) print high
$7 = 1
 (gdb) print search
$8 = 5
 (gdb) print data[0]
$9 = 4
 (gdb) print data[1]
$10 = 6
 (gdb) quit
 The program is running.  Quit anyway (and kill it)? (y or n) y
```

Final Fix

```
if (data[middle] < search)
    low = middle;
else
    high = middle;
```

Should be:

```
if (data[middle] < search)
    low = middle +1;
else
    high = middle -1;
```

Tricking Interactive Debuggers to Stop when you want them to

```
float point_color(int point_number)
{
    float correction;           // color correction factor
    extern float red,green,blue;// current colors


    // Lookup color correction
    extern lookup(int point_number);


   if (point_number == 735)  /* ### Temp code ### */
       point_number = point_number;   /* ### Line to stop on ### */


    correction = lookup(point_number);
    return (red*correction * 100.0 +
            blue*correction * 10.0 +
            green*correction);
}
```

Runtime Errors

Segmentation Violation

- Bad pointer
- Indexing off the end of an array

Stack Overflow

- Too many local variables (big problem in DOS/Windows).
- Infinite recursion

Divide by 0

Floating exception (core dumped)

- On UNIX this is caused by floating point and *integer* divides.

Buffering problem

```
#include <iostream>
main()
{
    int i,j;    /* two random integers */


    i = 1;
    j = 0;
    std::cout << "Starting\n";
    std::cout << "Before divide...";
    i = i / j;  // divide by zero error
    std::cout << "After\n";
    return(0);
}
```

When run, this program outputs:

```
Starting
Floating exception (core dumped)
```

Problem Solved

```
#include <iostream>
int main()
{
    int i,j;    /* two random integers */

    i = 1;
    j = 0;
    std::cout << "Starting\n";
    std::cout.flush();
    std::cout << "Before divide...";
    std::cout.flush();
    i = i / j;  // divide by zero error
    std::cout << "After\n";
    std::cout.flush();
    return(0);
}
```

Confessional Method of Debugging

Programmer explains his program to an interested party, an uninterested part, a wall. The programmer just explains his program.

Typical session:

“Hey Bill, could you take a look at this. My program has a bug in it. The output should be 8.0 and I’m getting -8.0. The output is computed using this formula and I’ve checked out the payment value and rate and the date must be correct unless there is something wrong with the leap year code which — Thank you Bill, you’ve found my problem.”

Bill says nothing.

Optimization

And now a word about optimization:

Don't!

Getting a faster machine is the most economical way to speed up your program.

Unoptimized Program

```
const int X_SIZE = 60;
const int Y_SIZE = 30;

int matrix[X_SIZE][Y_SIZE];

void init_matrix(void)
{
    int x,y;    // current element to initialize

    for (x = 0; x < X_SIZE; ++x) {
        for (y = 0; y < Y_SIZE; ++y) {
            matrix[x][y] = -1;
        }
    }
}
```

Register Variables

```
const int X_SIZE = 60;
const int Y_SIZE = 30;

int matrix[X_SIZE][Y_SIZE];

void init_matrix(void)
{
    register int x,y;    // current element to initialize


    for (x = 0; x < X_SIZE; ++x) {
        for (y = 0; y < Y_SIZE; ++y) {
            matrix[x][y] = -1;
        }
    }
}
```

The **register** keword is a hint that tells the compiter to put a frequently used variable in a machine register (which is faster than stack memory).

But most modern compilers ignore this hit because they can do register allocation better than an human anyway.

With loop ordering

```
const int X_SIZE = 60;
const int Y_SIZE = 30;

int matrix[X_SIZE][Y_SIZE];

void init_matrix(void)
{
    register int x,y;    // current element to init
ialize

    for (y = 0; y < Y_SIZE; ++y) {
        for (x = 0; x < X_SIZE; ++x) {
            matrix[x][y] = -1;
        }
    }
}
```

Powers of 2

Indexing an array requires a multiply. For example to execute the line:

```
matrix[x][y] = -1;
```

the program must compute the location for placing the -1. To do this the program must perform the following steps:

1) Get the address of the `matrix`.
2) Compute `x * Y_SIZE`.
3) Compute `y`.
4) Add up all three parts to form the address. In C++ this code looks like:

```
*(matrix + (x * Y_SIZE) + y) = -1;
```

If we change Y_SIZE from 30 to 32, we waste space but speed up the computation.

Using Pointers

```
const int X_SIZE = 60;
const int Y_SIZE = 30;

int matrix[X_SIZE][Y_SIZE];

void init_matrix(void)
{
    register int index;          // element counter
    register int *matrix_ptr;    // Current element


    matrix_ptr = &matrix[0][0];
    for (index = 0; index < X_SIZE * Y_SIZE; ++index) {
        *matrix_ptr = -1;
        ++matrix_ptr;
    }
}
```

Can the loop counter and the `matrix_ptr` be combined?

Using the library function

```
#include <string.h>

const int X_SIZE = 60;
const int Y_SIZE = 30;

int matrix[X_SIZE][Y_SIZE];

void init_matrix(void)
{
    memset(matrix, -1, sizeof(matrix));
}
```

Our function is one line long. We might want to make it an **inline** function.

Optimizing techniques

- Remove invariant code from loops
- Loop ordering
- Reduction in strength
- Use reference parameters
- Powers of 2
- Pointers
- **inline** functions

Optimization Costs

Operation	Relative Cost
file input and out (`<<` and `>>`) Also includes the C functions `printf` and `scanf`.	1000
new and **delete**	800
trigonometric functions (sin, cos...)	500
floating point (any operation)	100
integer divide	30
integer multiple	20
function call	10
simple array index	6
shifts	5
add/subtract	5
pointer dereference	2
bitwise and, or, not	1
logical and, or, not	1