



# Powerful tools for Linux C/C++ developers in Eclipse

FOSDEM 2012

Andrew Overholt  
Red Hat



# C/C++ Development

- Editing
- Building
- Debugging
- Profiling



C/C++ - FOSDEM/src/FOSDEM.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Includes
- src
  - FOSDEM.c
- Debug
- gprofTest
- helloC++
  - Binaries
  - Includes
  - src
  - autom4te.cache
  - aclocal.m4
  - AUTHORS
  - ChangeLog
  - config.guess
  - config.log

FOSDEM.c

```
#include <stdio.h>
#include <stdlib.h>

int square(int i)
{
    int square = i * i;
    return square;
}

int main(void) {
    int i = 2;

    i = square(i);

    return EXIT_SUCCESS;
}
```

Outline

- stdio.h
- stdlib.h
- square(int) : int
- main(void) : int

Problems Tasks Console Properties

CDT Build Console [vignere\_cipher]

```
make all
make: Nothing to be done for `all'.

**** Build Finished ****
```

Writable Smart Insert 7:18

# Code Completion

```
int main() {  
    int match = strcmp("fosdem", "fosdem");  
    printf  
    return  
}
```

- printf (const char \*template, ...) int
- printf\_size (FILE \*fp, const struct printf\_info \*info, const void \*con
- printf\_size\_info (const struct printf\_info \*info, size\_t n, int \*argtyp
- printf(const char \* \_\_format,...) : int

The printf function  
the template string  
number of characters  
output error.

## Ctrl-**<Spacebar>**

# API documentation

```
int main() {  
    int match = strcmp("fosdem", "fosdem");  
    if (match == 0) {  
        printf("Strings match\n");  
    } else {  
        printf("Strings don't match\n");  
    }  
    return 0;  
}
```

**Name:** printf

**Prototype:** int printf (const char \*template, ...)

**Description:**

The printf function prints the optional arguments under the control of the template string template to the stream stdout. It returns the number of characters printed, or a negative value if there was an output error.

**Header files:**

stdio.h

Press 'F2' for focus

# Add #include

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int match = strcmp("a", "a");
    return EXIT_
}
```

Toggle Comment	Ctrl+/_
Add Block Comment	Shift+Ctrl+/_
Remove Block Comment	Shift+Ctrl+\
<hr/>	
Shift Right	
Shift Left	Shift+Tab
Correct Indentation	Ctrl+I
Format	Shift+Ctrl+F
<b>Add Include</b>	<b>Shift+Ctrl+N</b>
<hr/>	
Generate Getters and Setters...	
Implement Method...	

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void) {
    int match = strcmp("a", "a");
    return EXIT_SUCCESS;
}
```

# Refactoring

```
int main(void) {  
    int i = 2;  
    int square = i * i;  
    return EXIT_SUCCESS;  
}
```

Quick Fix	Ctrl+1	
Source	Shift+Alt+S	>
Surround With	Shift+Alt+Z	>
Refactor		>
Declarations		>
References		>
Search Text		>
Make Targets		>
Run As		>
	Rename...	Shift+Alt+R
	Extract Constant...	Alt+C
	Extract Local Variable...	Shift+Alt+L
	Extract Function...	Shift+Alt+M
	Toggle Function Definition	Shift+Alt+T
	Hide Method...	

# Refactoring

The following changes are necessary to perform the refactoring.

Changes to be performed

- FOSDEM.c - FOSDEM/src

FOSDEM.c

Original Source	Refactored Source
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  int main(void) {     int i = 2;      int square = i * i;      return EXIT_SUCCESS; }</pre>	<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  int square(int i) {     int square = i * i;     return i; }  int main(void) {     int i = 2;      i = square(i);      return EXIT_SUCCESS; }</pre>



# Refactoring

```
int main(void) {  
    int i = 2;  
    int square = i * i;  
    return EXIT_SUCCESS;  
}
```

```
int square(int i)  
{  
    int square = i * i;  
    return i;  
}
```

```
int main(void) {  
    int i = 2;  
    i = square(i);  
    return EXIT_SUCCESS;  
}
```

# Error Highlighting

```
int main(void) {  
    int i = 2;  
    int square = i * j;  
    return EXIT_SUCCESS;  
}
```

 Symbol 'j' could not be resolved

Press 'F2' for focus

# Static Analysis

```
int main(void) {
    int a = 3;
    if (a = 2) {
        if (a || a + 2 && a + 3)
            return;
    }
    puts("!!!Hello World!!!");
    return EXIT_SUCCESS;
}
```

⚠ Suggested parenthesis around expression 'a + 2 && a + 3'

# Outline

```
int square(int i)
{
    int square = i * i;
    return i;
}

int main(void) {
    int i = 2;

    i = square(i);

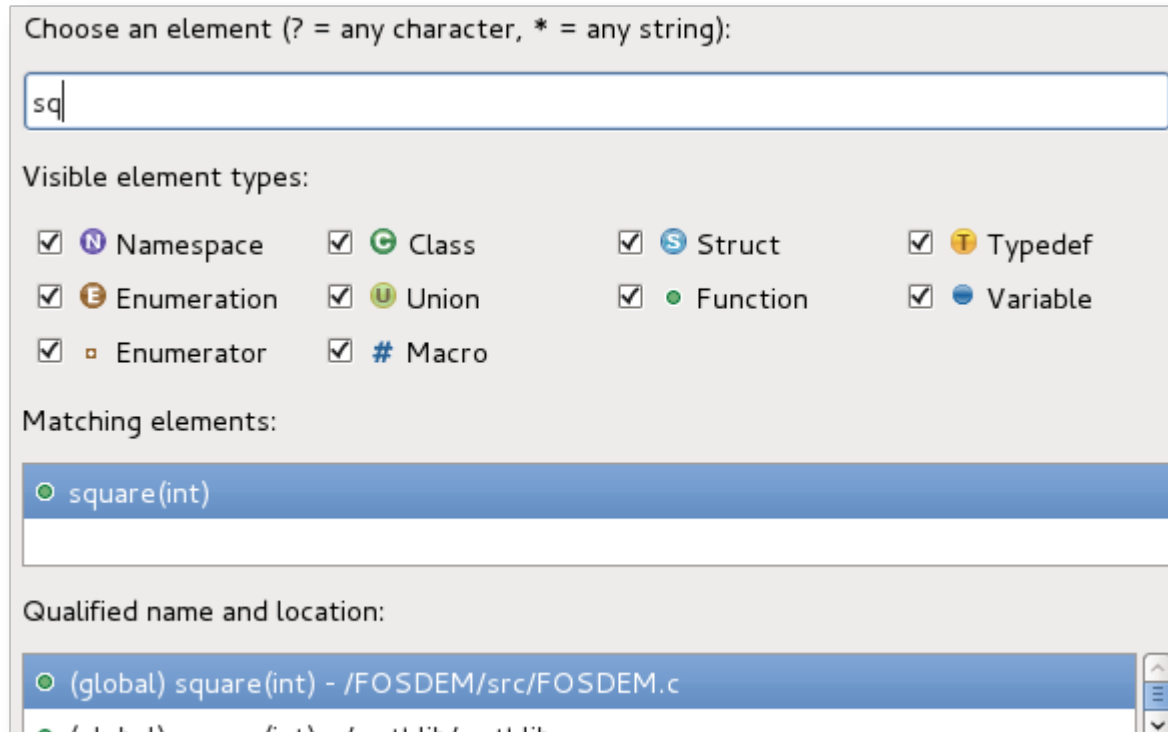
    return EXIT_SUCCESS;
}
```

sq|

● square(int) : int

## Ctrl-o – Quick Outline

# Navigation



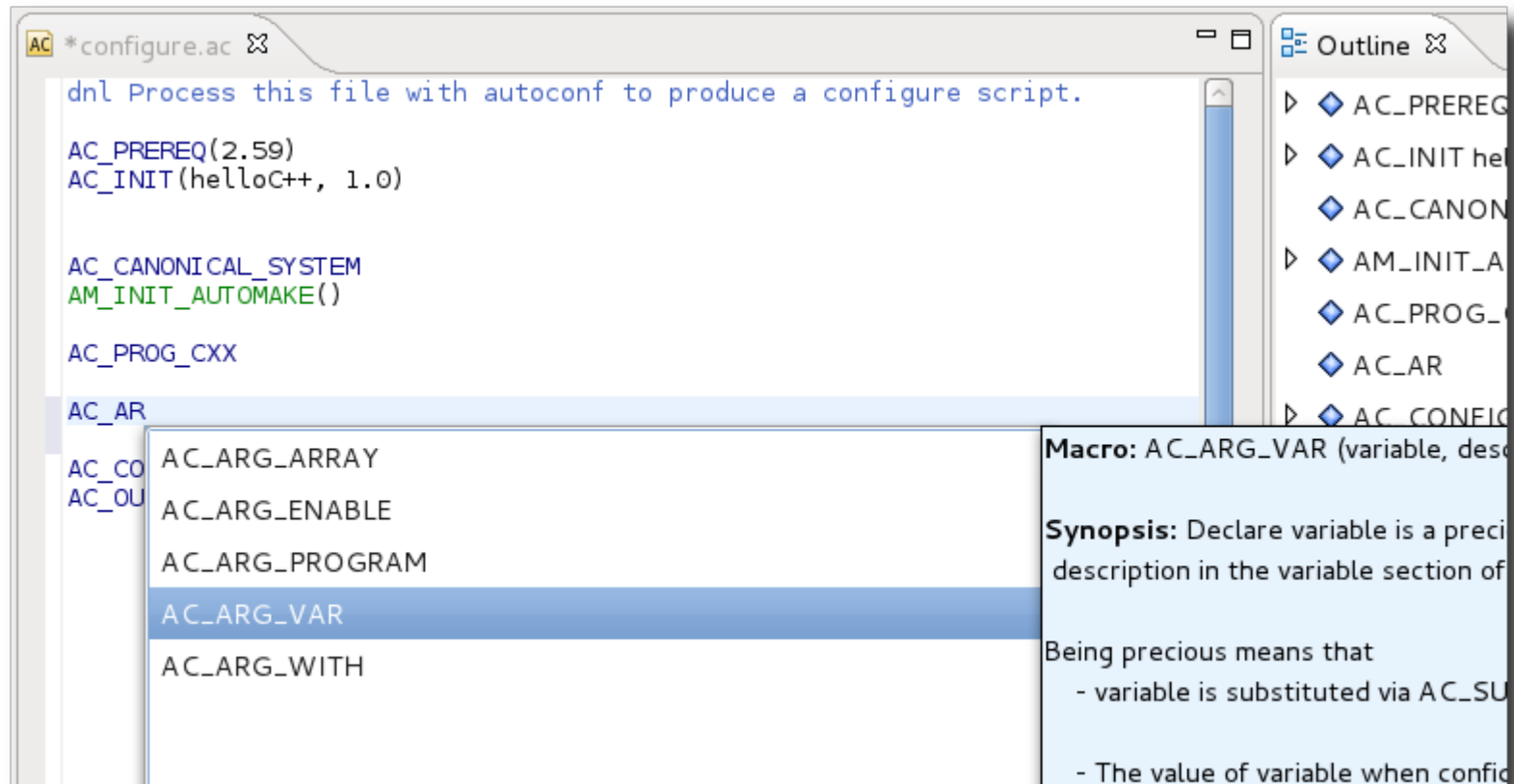
## Ctrl-Shift-t – Open Element



# Building

- gcc
- Makefiles
- GNU Autotools

# autoconf completion



The screenshot shows an IDE window titled `*configure.ac` with the following content:

```
dn! Process this file with autoconf to produce a configure script.  
  
AC_PREREQ(2.59)  
AC_INIT(helloC++, 1.0)  
  
AC_CANONICAL_SYSTEM  
AM_INIT_AUTOMAKE()  
  
AC_PROG_CXX  
  
AC_AR  
  
AC_CO  
AC_OU
```

An auto-completion menu is open over the `AC_AR` line, listing the following options:

- AC\_ARG\_ARRAY
- AC\_ARG\_ENABLE
- AC\_ARG\_PROGRAM
- AC\_ARG\_VAR** (highlighted)
- AC\_ARG\_WITH

To the right, an **Outline** pane shows a tree view of the file's structure, with `AC_ARG_VAR` expanded to show its details:

**Macro:** AC\_ARG\_VAR (variable, desc

**Synopsis:** Declare variable is a preci  
description in the variable section of

Being precious means that

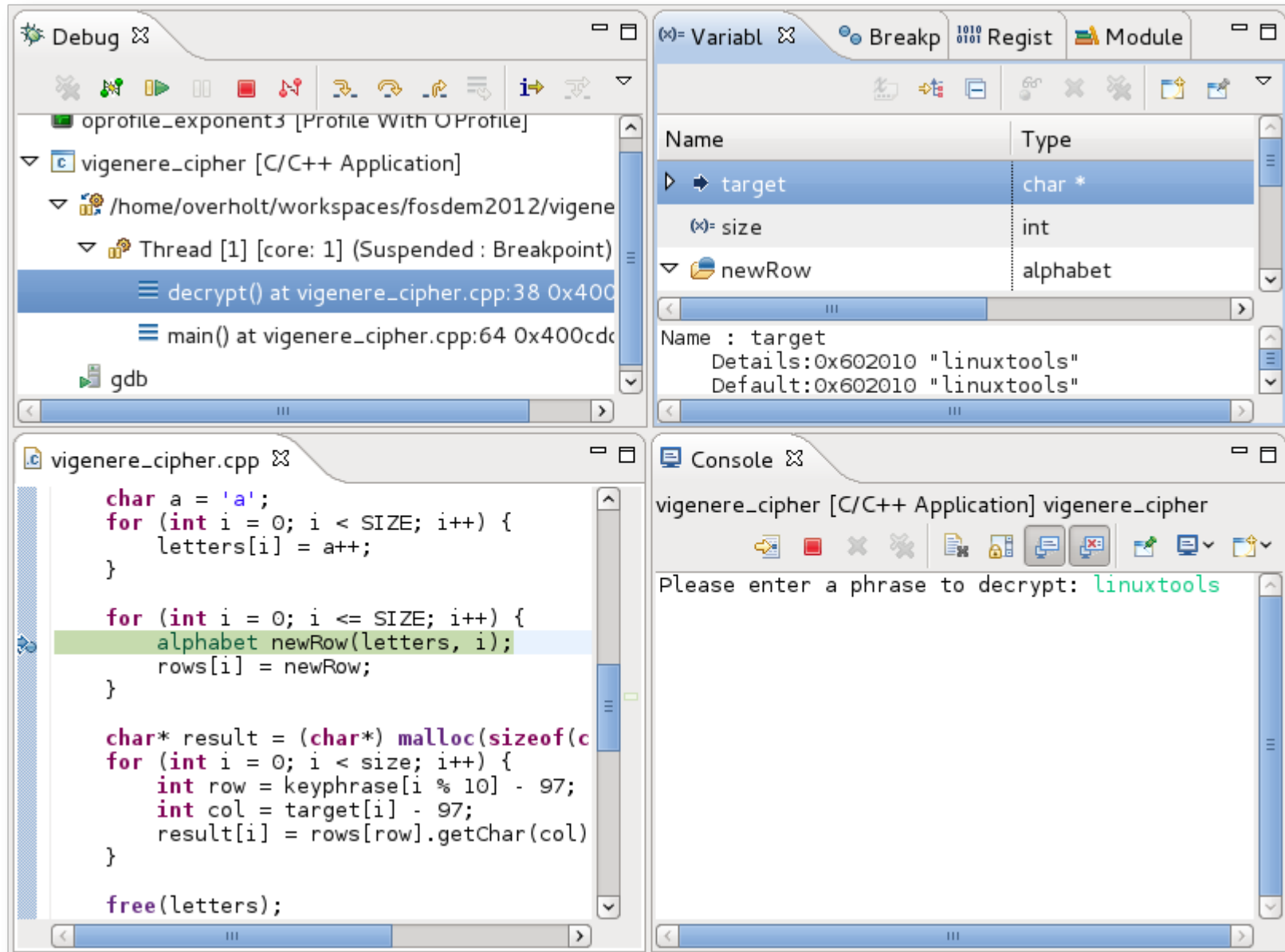
- variable is substituted via AC\_SU
- The value of variable when confic

# Build System Future

- install dependencies listed in pkg-config information (configure.ac) and build logs
- management of changes required in multiple files (ex. configure.ac and Makefile.am)



# Debugging



The screenshot displays a debugger window with the following components:

- Top-Left Pane (Debug Console):** Shows the current session as 'oprofile\_exponent3 [Profile With OProfile]'. The application being debugged is 'vigenere\_cipher [C/C++ Application]'. The current thread is 'Thread [1] [core: 1] (Suspended : Breakpoint)'. The execution is paused at the 'decrypt()' function in 'vigenere\_cipher.cpp:38' at memory address '0x400c...'. The 'main()' function is also visible at 'vigenere\_cipher.cpp:64'.
- Top-Right Pane (Variables):** Displays a table of variables:

Name	Type
target	char *
size	int
newRow	alphabet

Below the table, details for the 'target' variable are shown: 'Name : target', 'Details: 0x602010 "linuxtools"', and 'Default: 0x602010 "linuxtools"'. The variable 'target' is highlighted in blue.
- Bottom-Left Pane (Source Code):** Shows the source code for 'vigenere\_cipher.cpp'. A breakpoint is set at the line 'alphabet newRow(letters, i);'. The code includes a loop for processing letters, a loop for generating rows, and a final loop for constructing the result string.

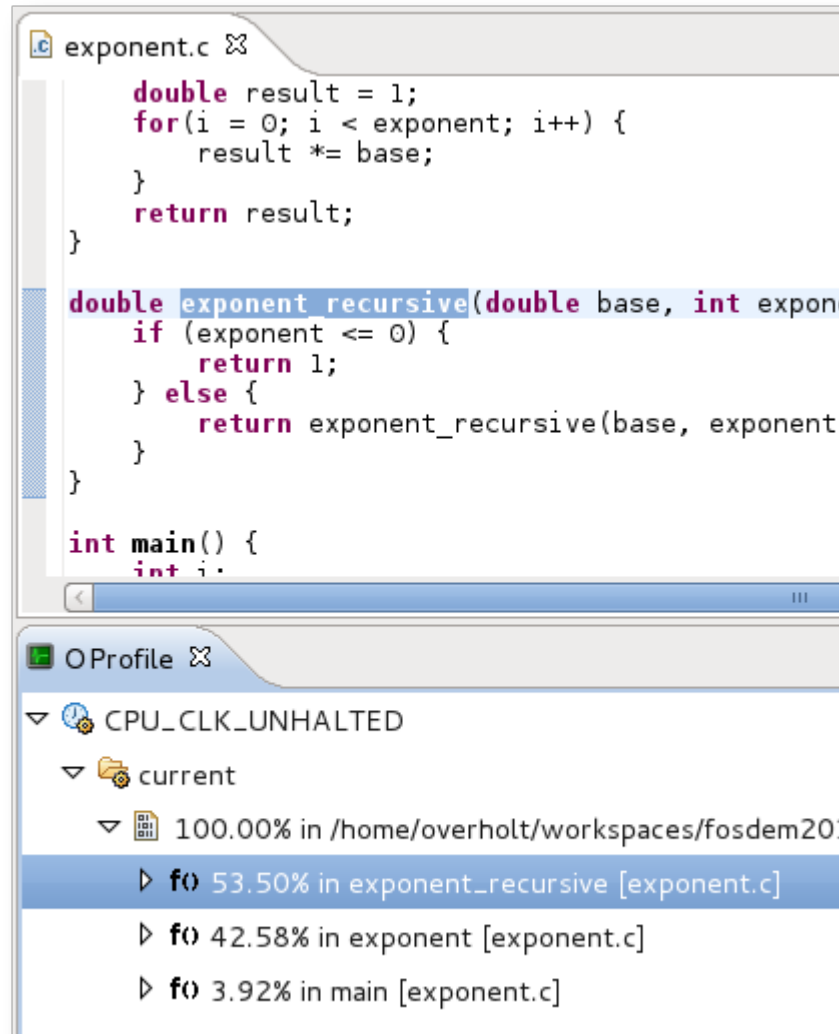
```
char a = 'a';
for (int i = 0; i < SIZE; i++) {
    letters[i] = a++;
}

for (int i = 0; i <= SIZE; i++) {
    alphabet newRow(letters, i);
    rows[i] = newRow;
}

char* result = (char*) malloc(sizeof(c
for (int i = 0; i < size; i++) {
    int row = keyphrase[i % 10] - 97;
    int col = target[i] - 97;
    result[i] = rows[row].getChar(col)
}

free(letters);
```
- Bottom-Right Pane (Console):** Shows the application's output. The prompt 'Please enter a phrase to decrypt:' is followed by the user input 'linuxtools'.

# OProfile



The image shows a code editor window titled "exponent.c" and an OProfile profiler window. The code in the editor is as follows:

```
double result = 1;
for(i = 0; i < exponent; i++) {
    result *= base;
}
return result;
}

double exponent_recursive(double base, int expon
if (exponent <= 0) {
    return 1;
} else {
    return exponent_recursive(base, exponent
}
}

int main() {
    int i;
```

The OProfile window shows the following performance profile:

- ▼ CPU\_CLK\_UNHALTED
  - ▼ current
    - ▼ 100.00% in /home/overholt/workspaces/fosdem2017/...
    - ▶ **f0** 53.50% in exponent\_recursive [exponent.c]
    - ▶ **f0** 42.58% in exponent [exponent.c]
    - ▶ **f0** 3.92% in main [exponent.c]

# Valgrind Memcheck

```
simpleMemcheckTest.c ✖
#include <stdlib.h>
#include <stdio.h>

#define SIZE 10
int main() {
    // free is not called
    char *waste = (char *)malloc(sizeof(char) * SIZE);

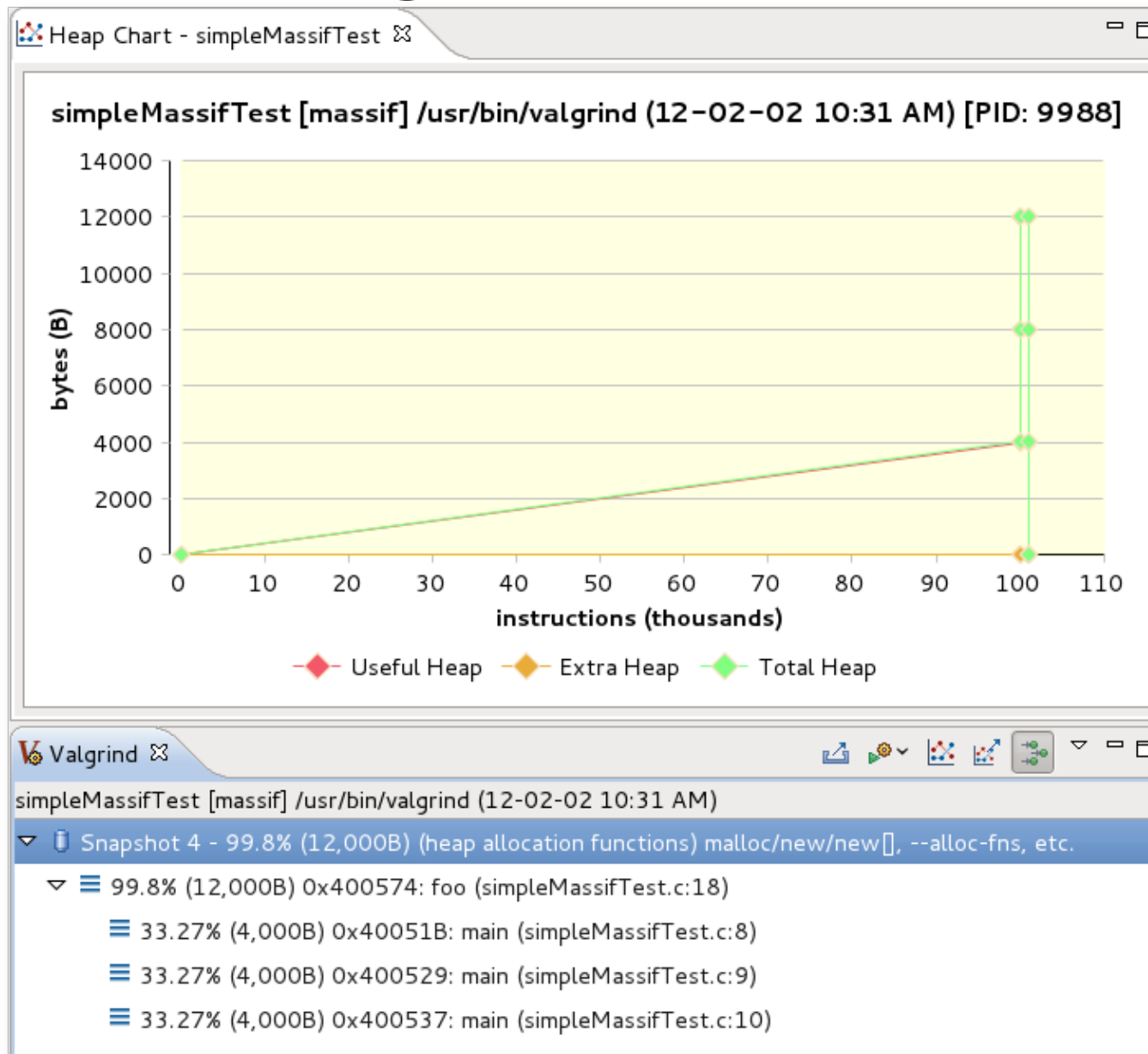
    // uninitialized pointer
    int *a;
    printf("%d\n", *a);

    // write past end of array
    waste[SIZE] = 0;

    return 0;
}

Valgrind ✖
simpleMemcheckTest [memcheck] /usr/bin/valgrind (12-02-02 10:28 AM)
Use of uninitialised value of size 8 [PID: 9839]
    at 0x40051E: main (simpleMemcheckTest.c:11)
Invalid write of size 1 [PID: 9839]
10 bytes in 1 blocks are definitely lost in loss record 1 of 1 [PID: 9839]
```

# Valgrind Massif





# Future

- perf
- remote
- VM integration?
- <your ideas here>

# Other stuff

- git
- Bugzilla/JIRA/Trac
- RPM tools
- a whole lot more



# Join us

- We welcome contributors of all forms!
  - Bug filers and feature requesters
  - Testers
  - Developers
  - Designers
  - Writers

# Contact Information

- Eclipse
  - <http://www.eclipse.org>
- CDT
  - <http://www.eclipse.org/cdt>
- Linux Tools Project
  - <http://www.eclipse.org/linuxtools>