

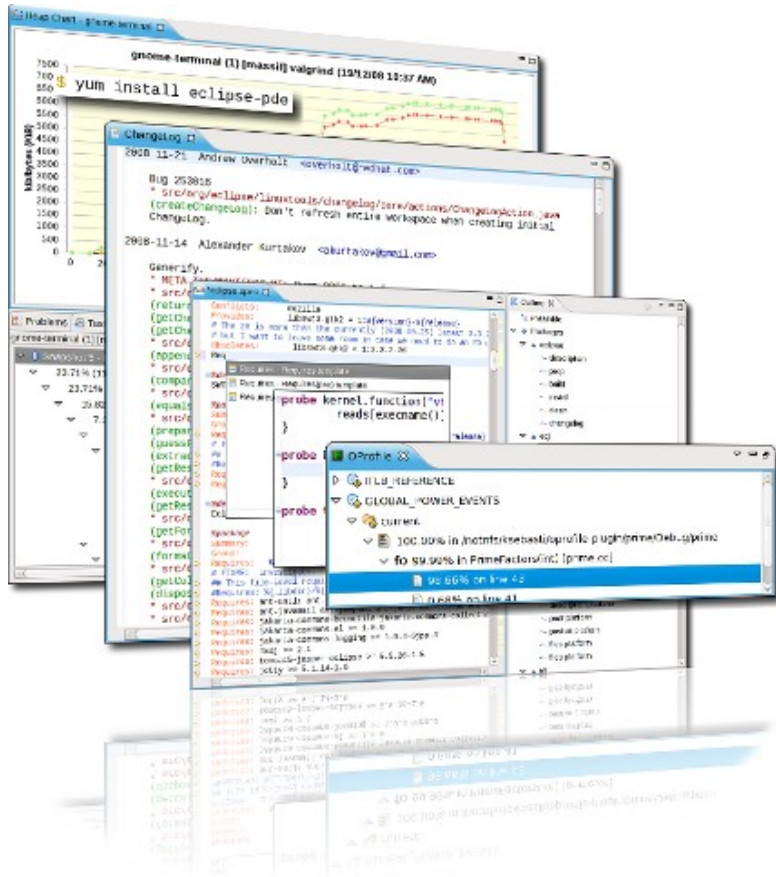


Eclipse Linux Tools Project

Linux Symposium Tracing Summit 2009

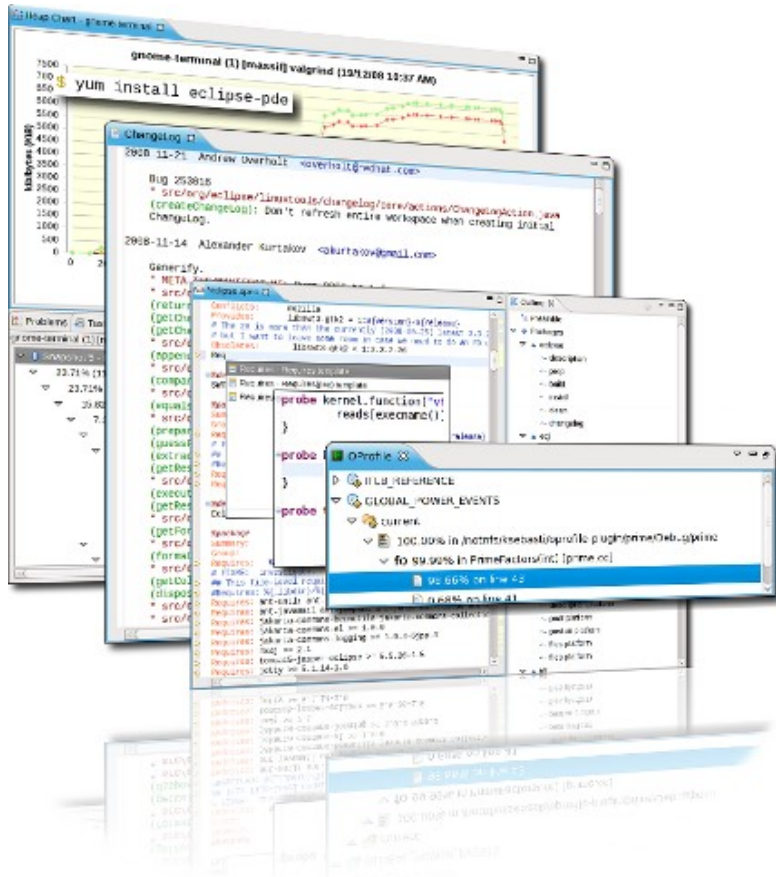
Andrew Overholt
Red Hat

Agenda



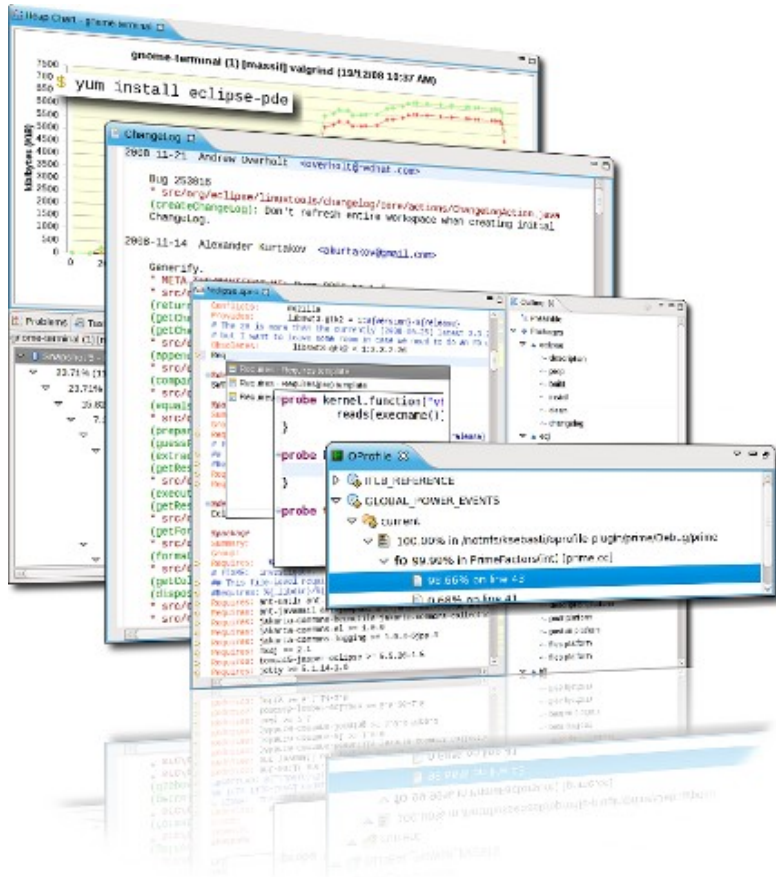
- Background
- Today's focus: tools for C/C++ developers built on CDT
- Memory profiling
- Call profiling
- Work areas
- Future plans and how to get involved

Agenda

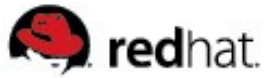


- **Background**
- Today's focus: tools for C/C++ developers built on CDT
- Memory profiling
- Call profiling
- Work areas
- Future plans and how to get involved

Agenda

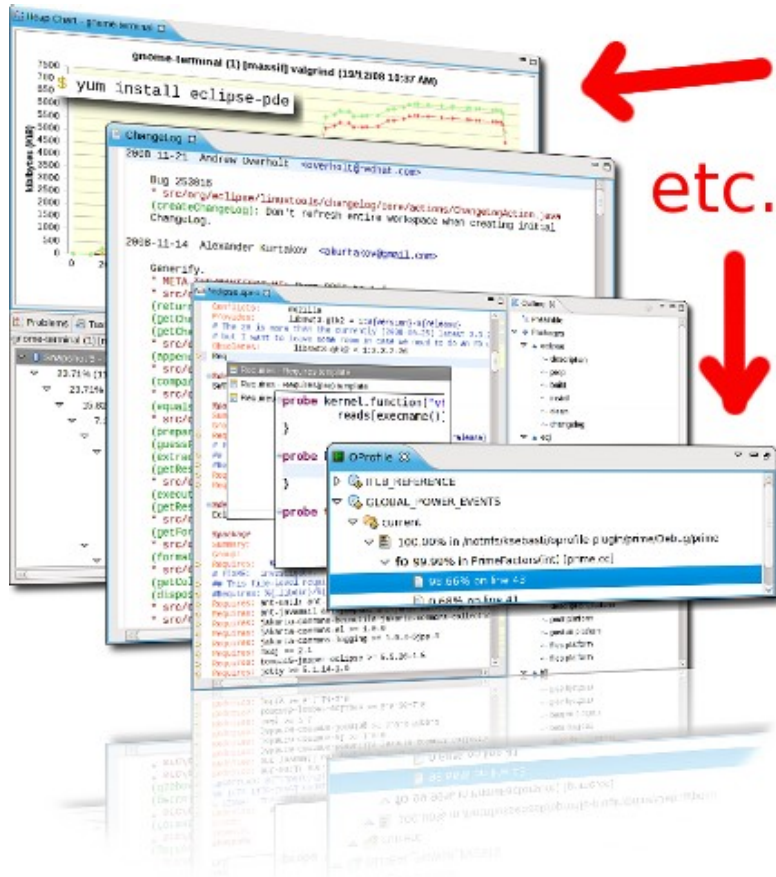


- Background
- Today's focus: **tools for C/C++ developers** built on CDT
- Memory profiling
- Call profiling
- Work areas
- Future plans and how to get involved

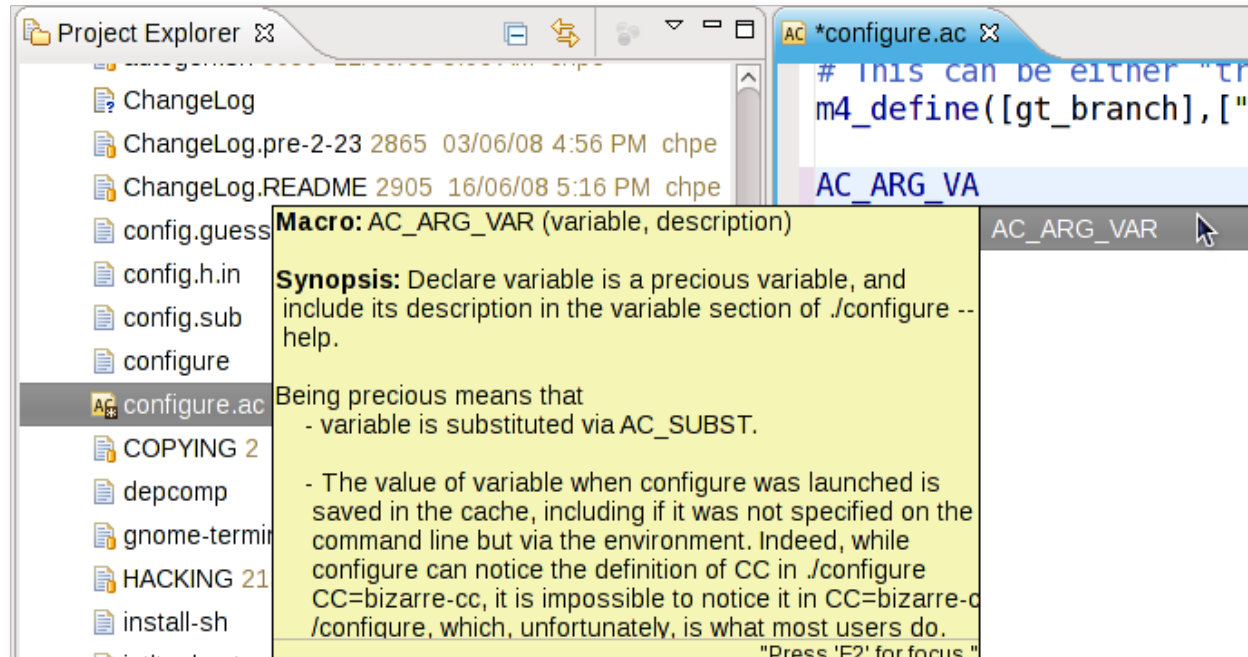


C/C++ Tools

- Developer-focused
- Sane defaults
- Integrate with CDT functionality



GNU Autotools



The screenshot shows an IDE interface with a Project Explorer on the left and a code editor on the right. The Project Explorer lists files such as ChangeLog, config.guess, and configure.ac. The code editor displays the contents of configure.ac, showing a comment and a macro definition. A tooltip is visible over the macro name AC_ARG_VAR, providing a synopsis and usage information.

Macro: AC_ARG_VAR (variable, description)

Synopsis: Declare variable is a precious variable, and include its description in the variable section of ./configure --help.

Being precious means that

- variable is substituted via AC_SUBST.
- The value of variable when configure was launched is saved in the cache, including if it was not specified on the command line but via the environment. Indeed, while configure can notice the definition of CC in ./configure CC=bizarre-cc, it is impossible to notice it in CC=bizarre-c

./configure, which, unfortunately, is what most users do.

"Press 'E?' for focus."

Libhover

```
void *blah = mall;  
  
if (display_name  
    display = gdk_d  
else  
    {  
        GSList *displ  
        const char *p  
  
        period = strr  
        if (period)  
        {  
            gulong n;
```

glib.h
terminal-intl.h

- mallinfo (void) struct mallinfo
- mallinfo (void) struct mallinfo
- malloc (size_t nbytes) void *
- malloc (size_t size) void *
- malloc_stats (void) void
- malloc_usable_size (void *aptr) size_t
- mallopt (int param, int value) int
- mallopt (int parameter, value) int

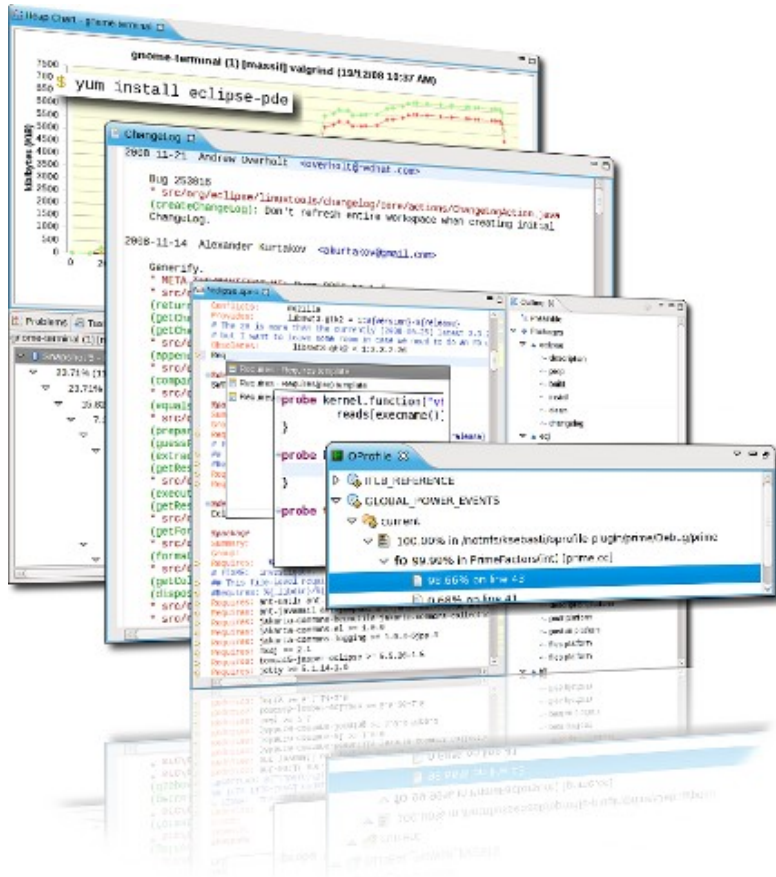
This function returns a pointer to a newly allocated block size bytes long, or a null pointer if the block could not be allocated.

Press 'Ctrl+Space' to show Template Proposals

SystemTap

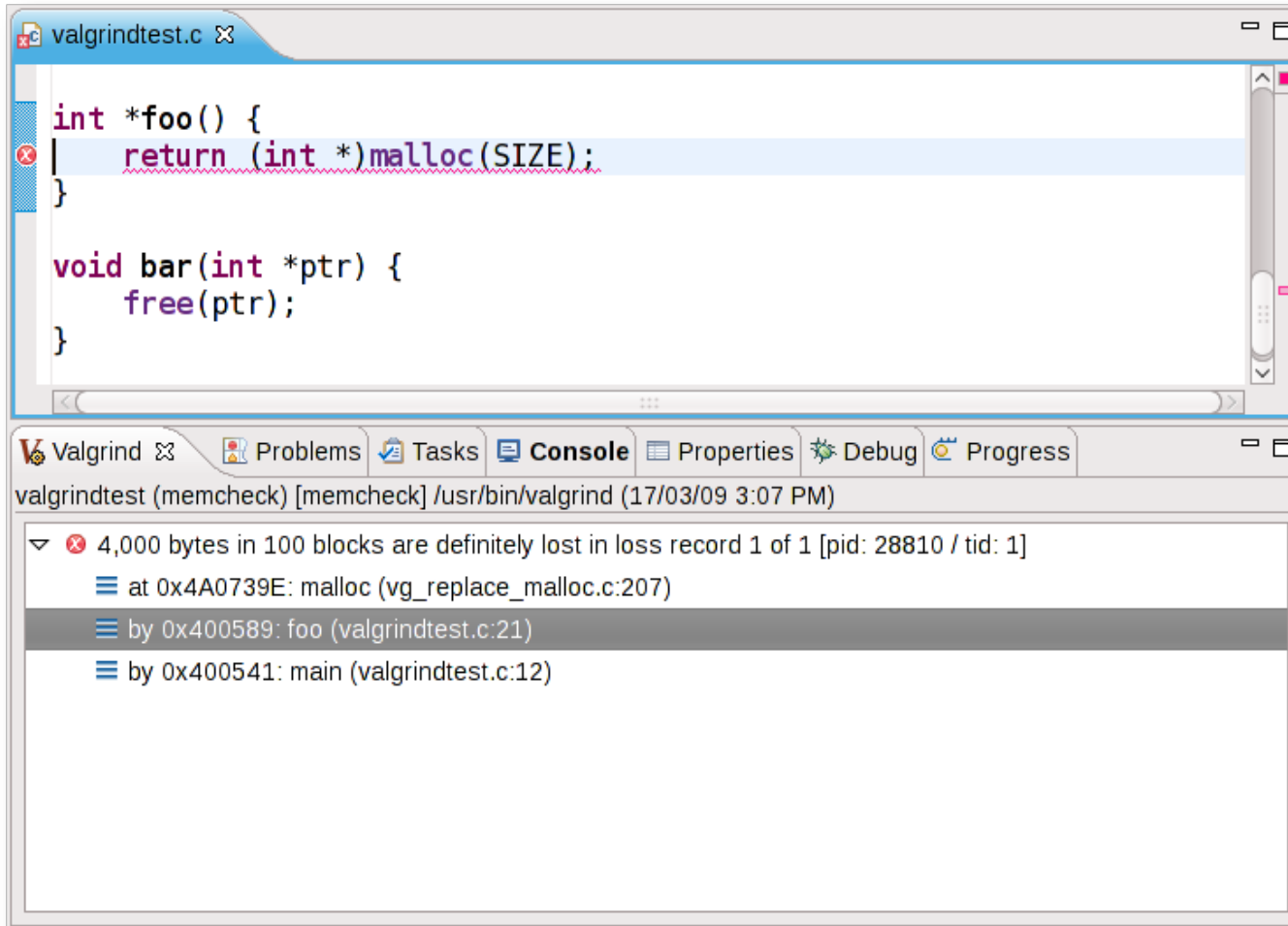
```
*test.stp ✕  
  
eprobe kernel.function("vfs_read").return {  
    reads[execname()] += $return  
}  
  
eprobe kernel.function("vfs_write").return {  
    writes[execname()] += $return  
}  
  
eprobe timer.s(1) {  
    foreach (p in reads)  
        total_io[p] += reads[p]  
    foreach (p in writes)  
        total_io[p] += writes[p]  
    foreach (p in total_io- limit 10)  
        printf("%15s r: %8d KiB w: %8d KiB\n",  
            p, reads[p]/1024,  
            writes[p]/1024)  
        printf("\n")  
    # Note we don't zero out reads, writes and total_io,  
}
```


Agenda



- Background
- Today's focus: tools for C/C++ developers built on CDT
- **Memory profiling**
- Call profiling
- Work areas
- Future plans and how to get involved

Valgrind



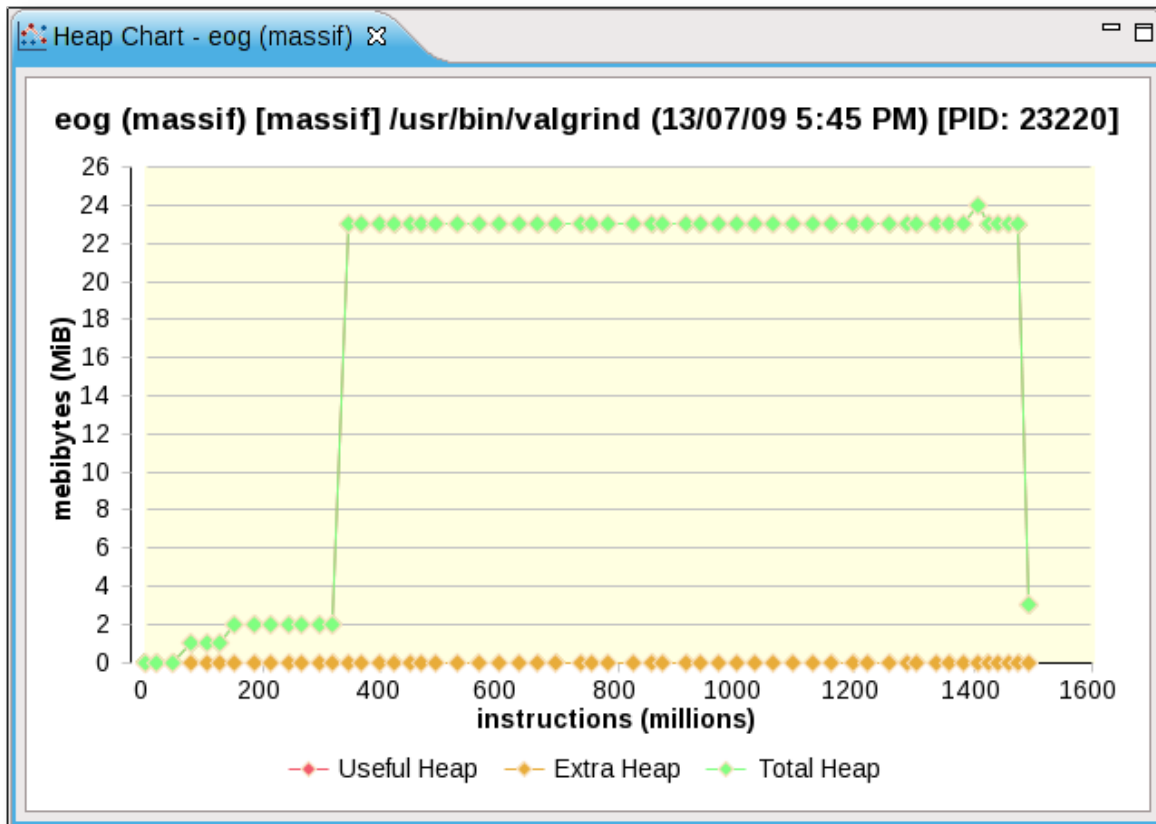
The screenshot shows a code editor window titled 'valgrindtest.c' with the following C code:

```
int *foo() {  
    return (int *)malloc(SIZE);  
}  
  
void bar(int *ptr) {  
    free(ptr);  
}
```

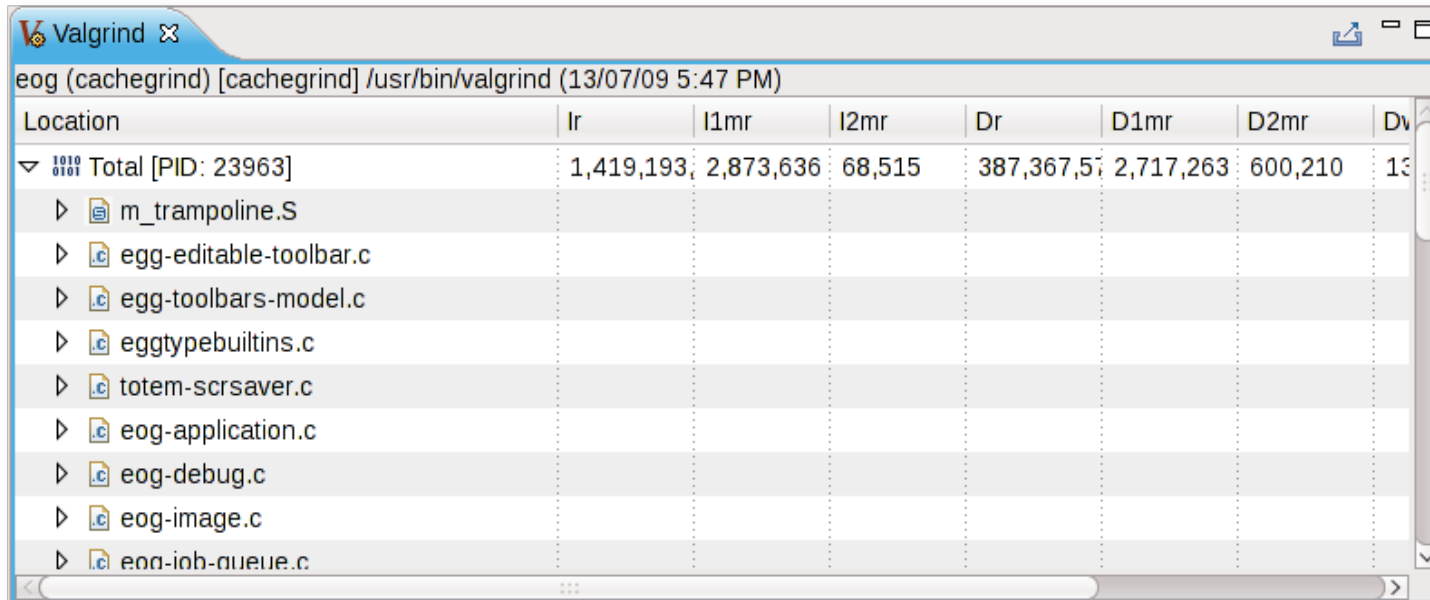
Below the code editor is a Valgrind output window. The title bar shows 'Valgrind' and several icons. The main text in the window reads: 'valgrindtest (memcheck) [memcheck] /usr/bin/valgrind (17/03/09 3:07 PM)'. A red 'x' icon indicates an error: '4,000 bytes in 100 blocks are definitely lost in loss record 1 of 1 [pid: 28810 / tid: 1]'. The stack trace below the error message is:

- at 0x4A0739E: malloc (vg_replace_malloc.c:207)
- by 0x400589: foo (valgrindtest.c:21)
- by 0x400541: main (valgrindtest.c:12)

Valgrind

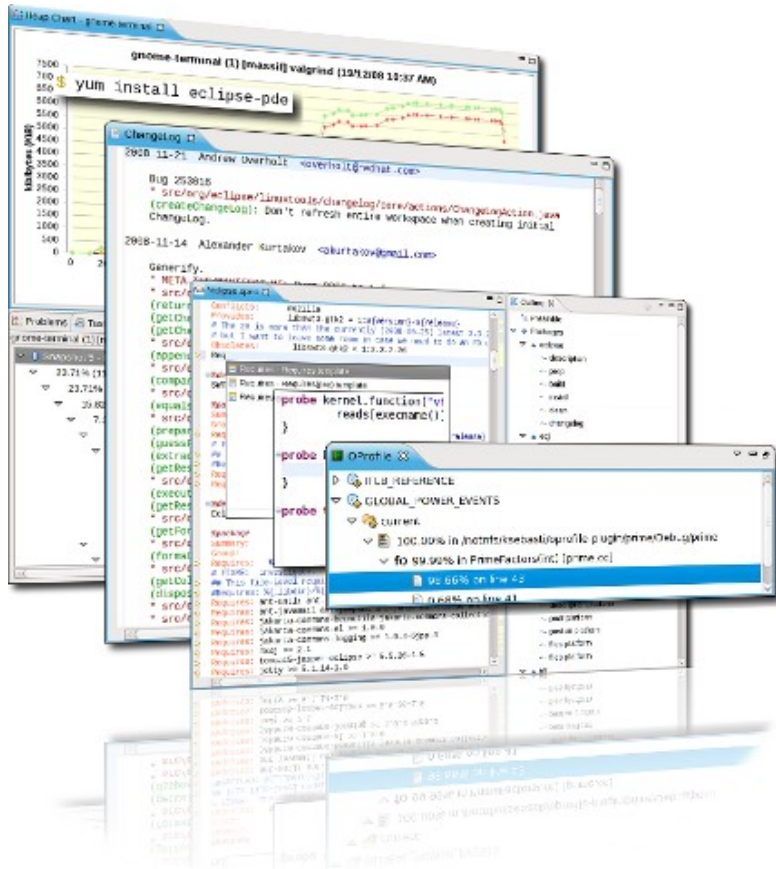


Valgrind



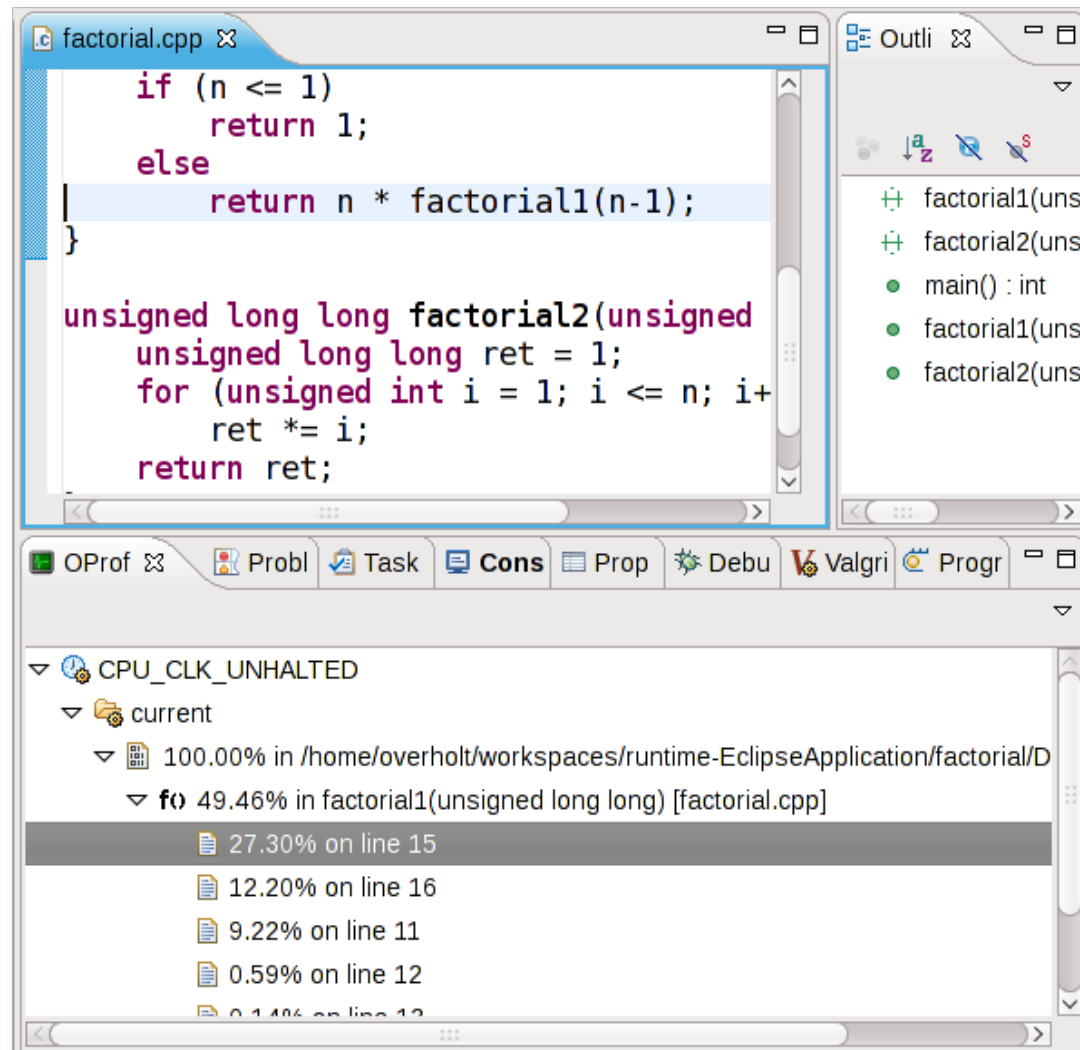
Location	Ir	l1mr	l2mr	Dr	D1mr	D2mr	Dv
▼ Total [PID: 23963]	1,419,193	2,873,636	68,515	387,367,571	2,717,263	600,210	13
▶ m_trampoline.S							
▶ egg-editable-toolbar.c							
▶ egg-toolbars-model.c							
▶ eggtypebuiltins.c							
▶ totem-scrcsaver.c							
▶ eog-application.c							
▶ eog-debug.c							
▶ eog-image.c							
▶ eog-iob-queue.c							

Agenda



- Background
- Today's focus: tools for C/C++ developers built on CDT
- Memory profiling
- **Call profiling**
- Work areas
- Future plans and how to get involved

OProfile



The screenshot displays the Eclipse IDE interface. The main editor window shows the source code for `factorial.cpp`. The code includes a recursive `factorial1` function and an iterative `factorial2` function. The `factorial1` function is highlighted in blue.

```
if (n <= 1)
    return 1;
else
    return n * factorial1(n-1);
}

unsigned long long factorial2(unsigned
unsigned long long ret = 1;
for (unsigned int i = 1; i <= n; i+
    ret *= i;
return ret;
```

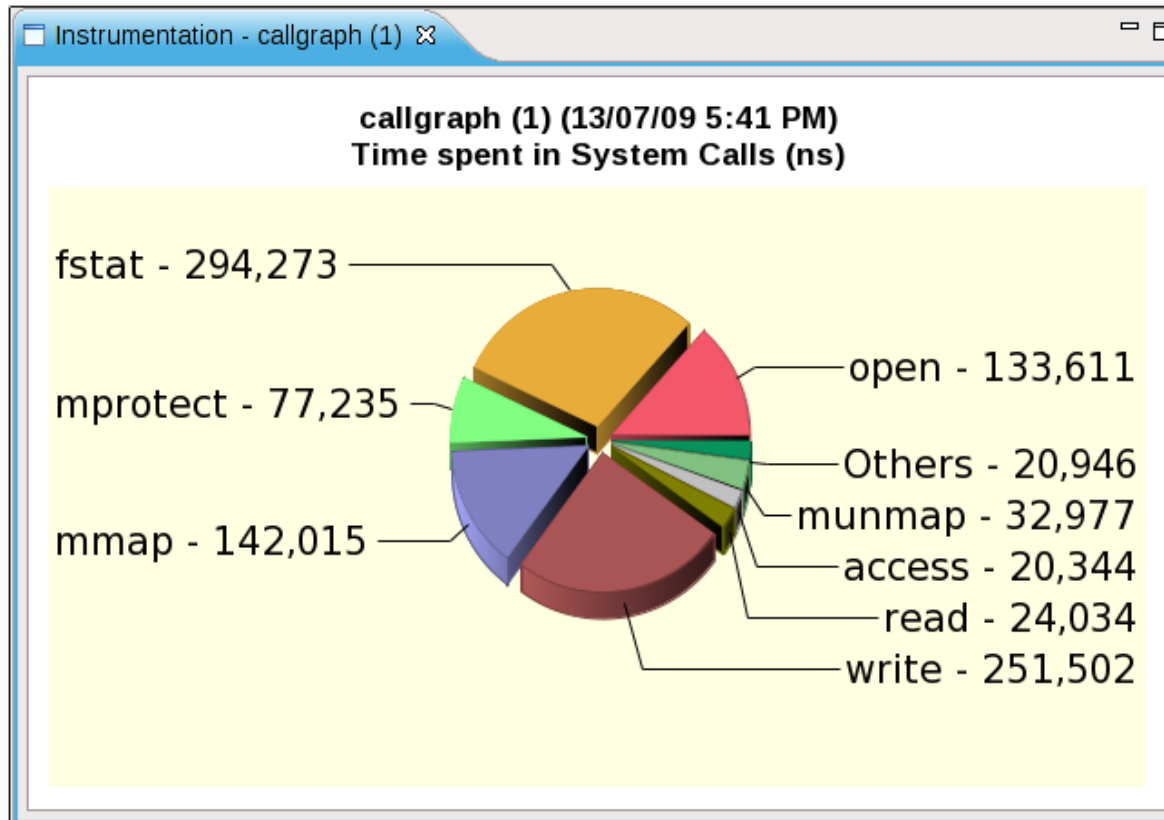
The Outliner on the right shows the project structure with the following items:

- factorial1(uns
- factorial2(uns
- main() : int
- factorial1(uns
- factorial2(uns

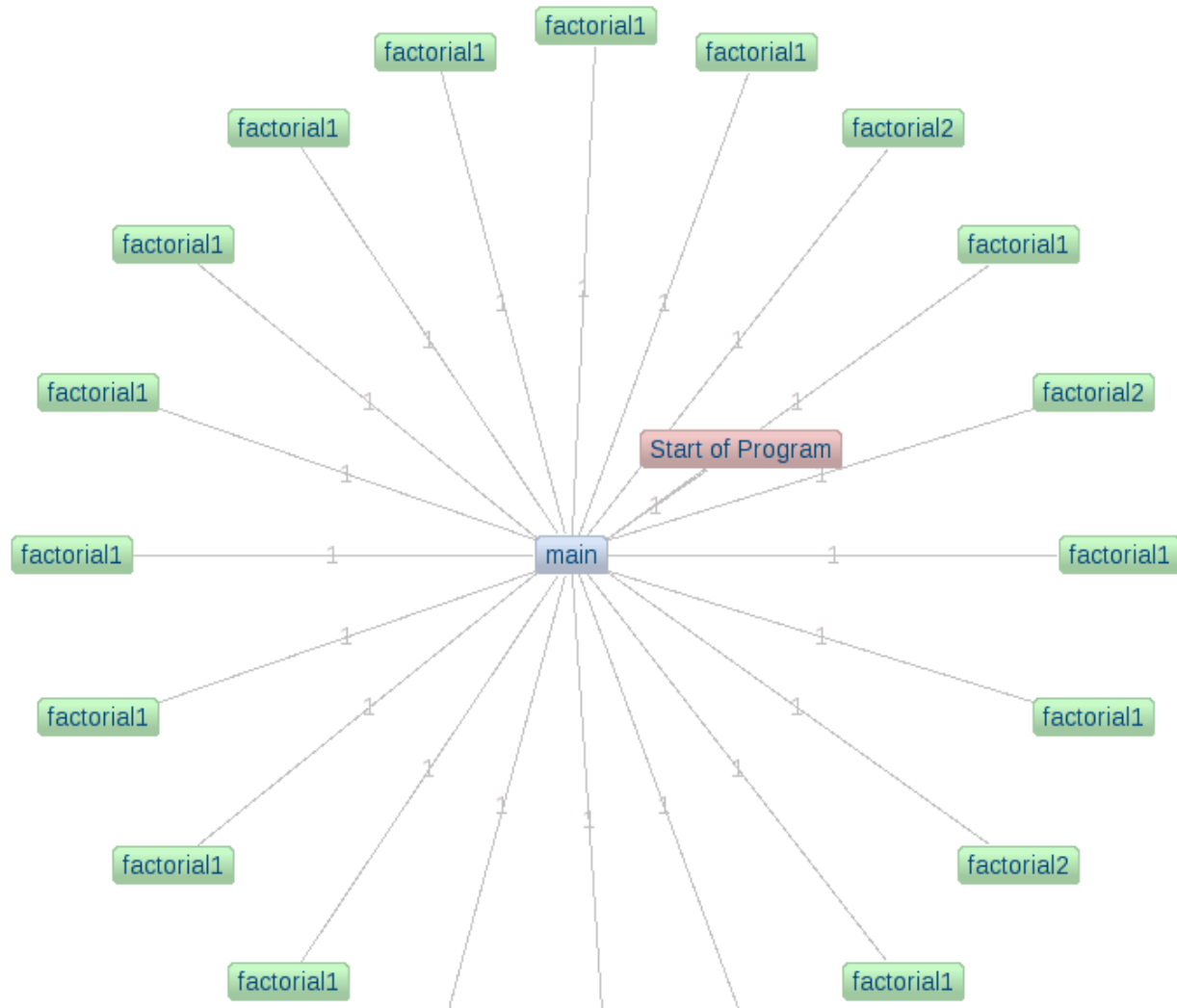
The OProfile view at the bottom shows the following performance data:

- CPU_CLK_UNHALTED
- current
- 100.00% in `/home/overholt/workspaces/runtime-EclipseApplication/factorial/D`
- 49.46% in `factorial1(unsigned long long) [factorial.cpp]`
- 27.30% on line 15
- 12.20% on line 16
- 9.22% on line 11
- 0.59% on line 12
- 0.14% on line 13

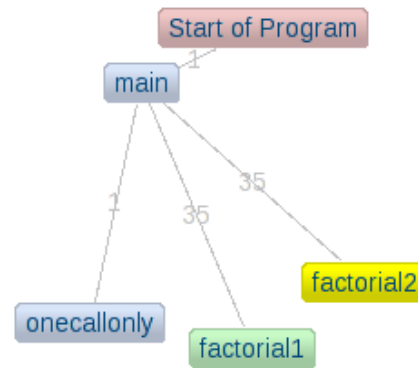
Custom



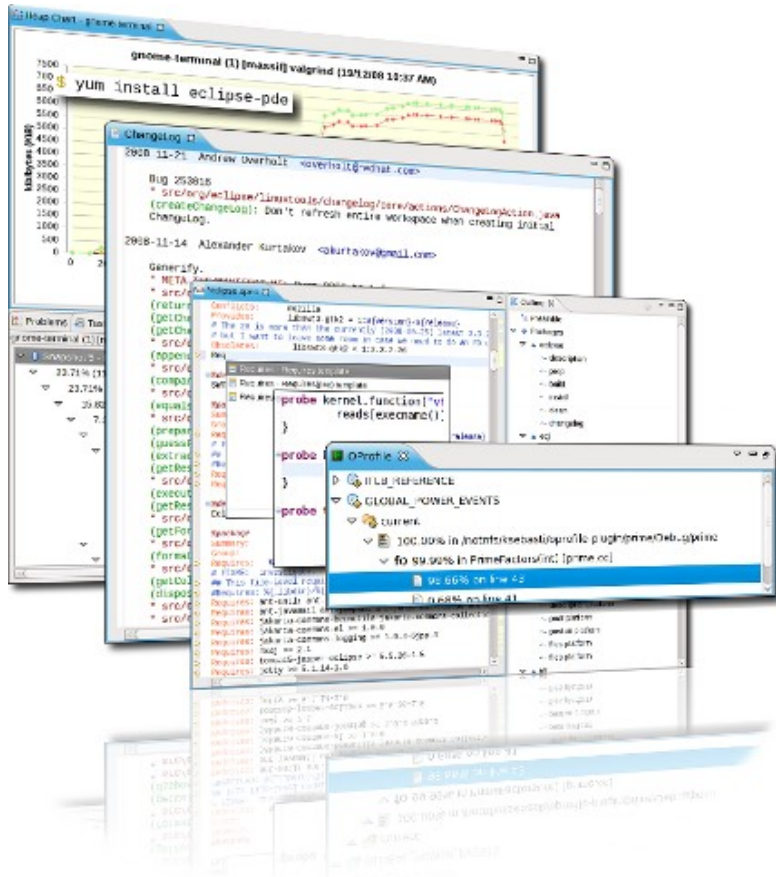
Custom



Custom



Agenda

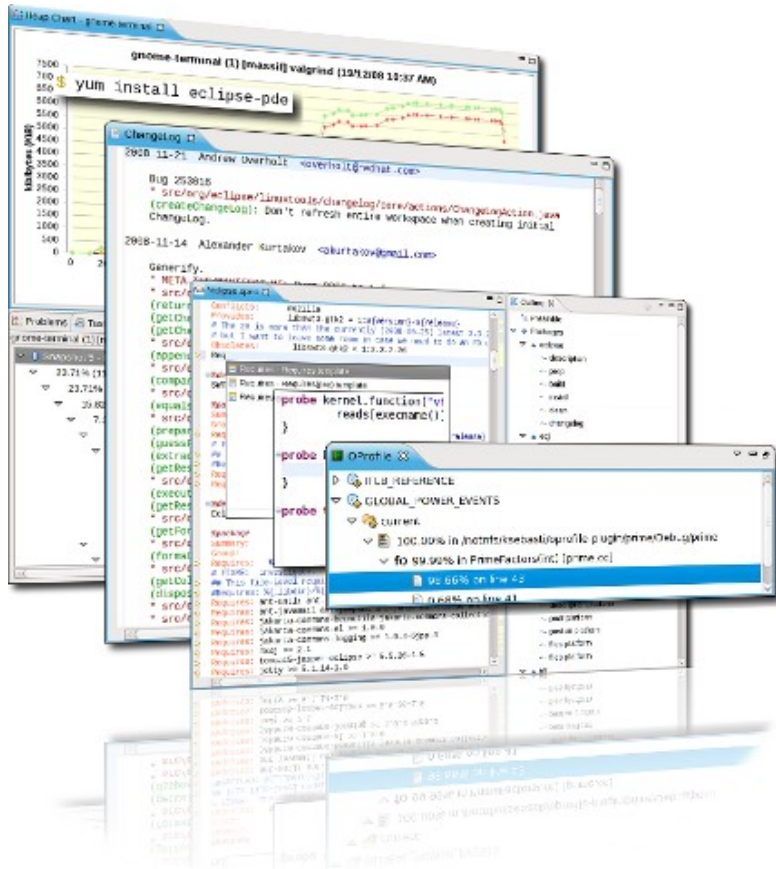


- Background
- Today's focus: tools for C/C++ developers built on CDT
- Memory profiling
- Call profiling
- **Work areas**
- Future plans and how to get involved

Work areas

- User interaction
 - Profile As menu?
- APIs
- Data presentation and visualization
- Functionality
- Integration with source code

Agenda



- Background
- Today's focus: tools for C/C++ developers built on CDT
- Memory profiling
- Call profiling
- Work areas
- **Future plans and how to get involved**



Future (near / distant)

- SystemTapGui integration
- gprof integration
- LTTng tooling
- Languages other than C/C++



Join us

- We welcome contributors of all forms
 - Plug-in testers
 - Plug-in developers
 - Web designers
 - Documentation authors
 - Graphic designers
 - Commercial adopters



<http://www.eclipse.org/linuxtools>