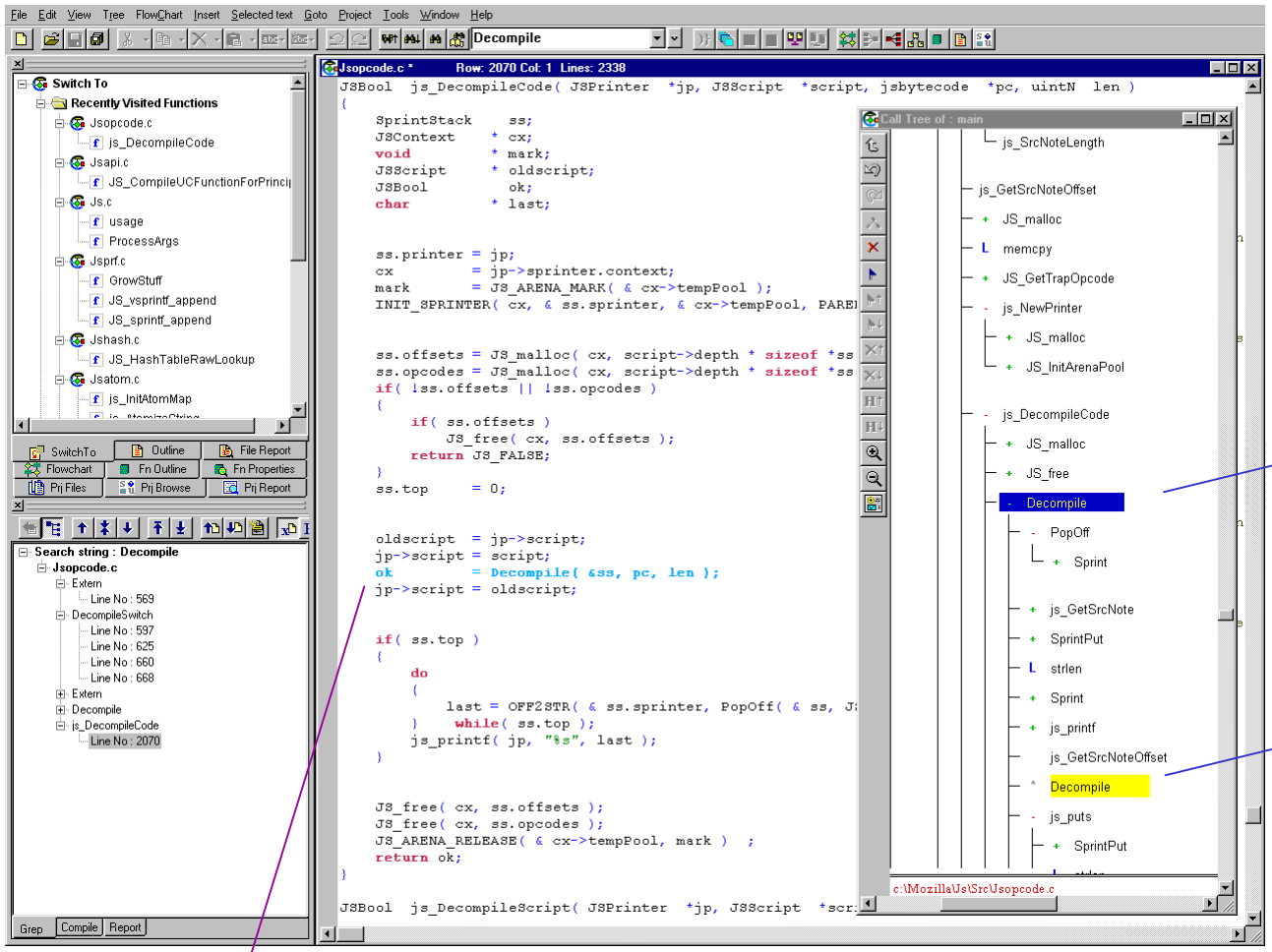
Viewing Trees in Crystal C/C++

1. A Call-Tree
2. Create a Call-Tree, Caller-Tree, or File-Tree
3. Toolbar and Related Operations
4. Search by Function-name in the Tree
5. Call-Path

A Call-Tree


Switch to Other Files or Functions
The List of all occurrences
Occurrence in Code
Current node
Its next Occurrence
Switch To
Recently Visited Functions
Search string : Decompile
Call Tree of : main
Jsopcode.c * Row: 2070 Col: 1 Lines: 2338

Call-tree, Caller-tree, and File-tree

1. To create a Call-tree:

   Place the cursor within a function; click the Call-tree icon  .

2. To create a Caller-tree:

   Place the cursor on a function-name and click the Caller-tree icon  .

3. To create a File-tree:

   Click View → Current File's Includes or click the File-tree icon  .

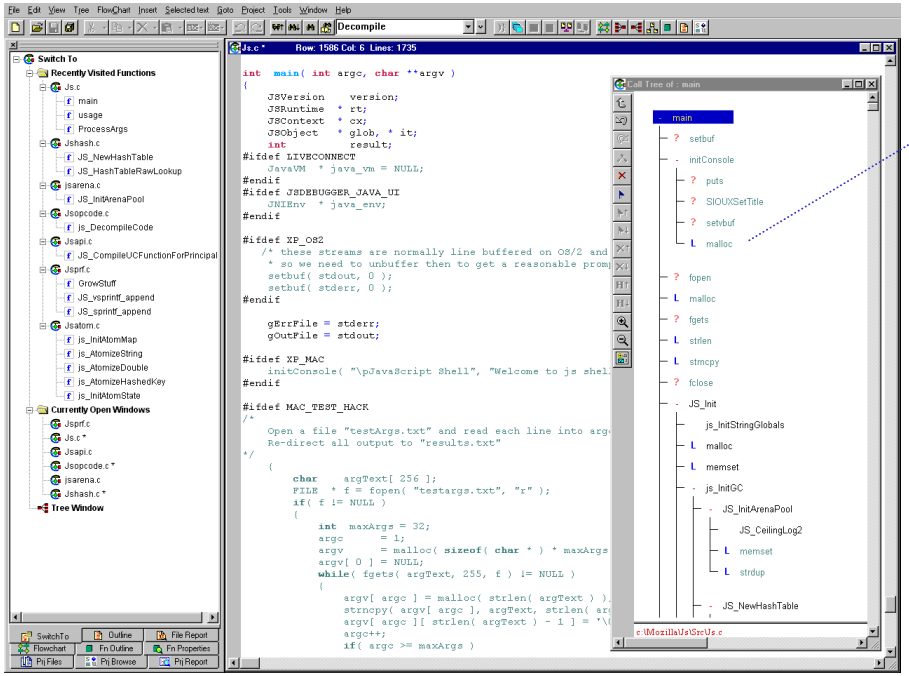
- To export a tree as a bitmap file:

  Use the "Tree" pull-down menu.

  Click Tree->Export Tree Image -> Whole

  Or drag-and select a part of the tree,
  then click Tree->Export Tree Image -> Selected

Call-Tree of main( )



L indicates a Library function.

? indicates - the function is undeclared.

***** indicates - the function is recursive.

P indicates - the tree node is not a function; rather it is a pointer to a function.

The green-colored nodes indicate that those function-calls were found in ignored lines of code. (the result of #if, #ifdef, etc.)

Right-click in the Tree-window for operations on the current-node, e.g.

- Expand Full
- Show its Call-path
- Show all Occurrences

Toolbar and Related Operations

- To make the tree concise by hiding nodes that you are not interested in:

  Click the node you wish to hide;

  Click  , or use right-click menu.

  To view hidden nodes, click Show All Nodes.

- Mark the nodes that are of interest to you.

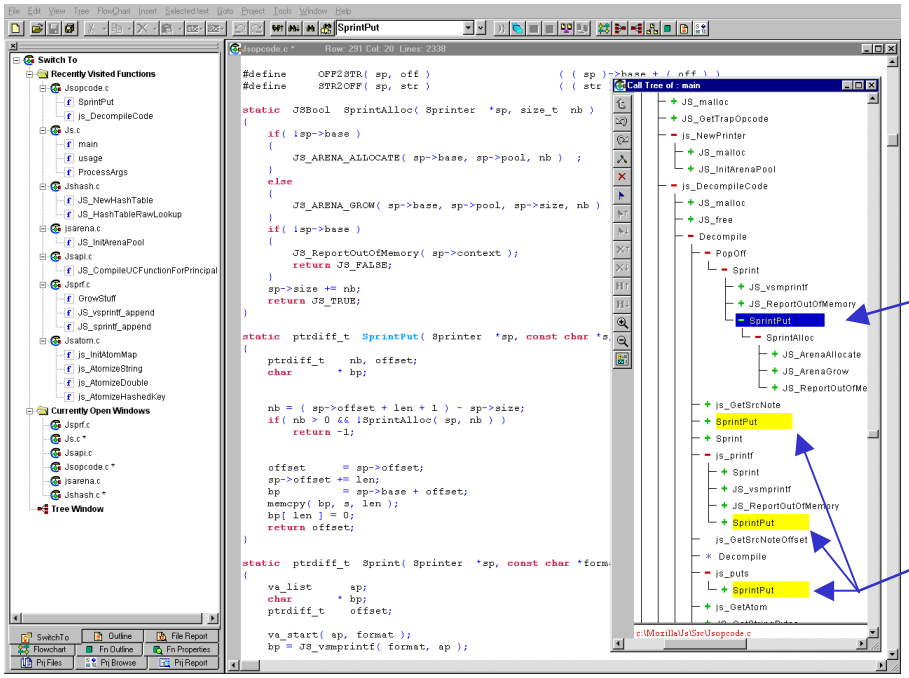
  Click  or  to revisit marked nodes;

- A file-tree shows headers that are #included (directly or nested) in the current file.

  right-click to highlight header files that were not found or could not be opened.

  Click  or  to visit highlighted nodes;

Search by function-name in the Expanded Tree



Result of Searching
for **SprintPut():**

First occurrence of **SprintPut()** in the tree

Other occurrences are highlighted.

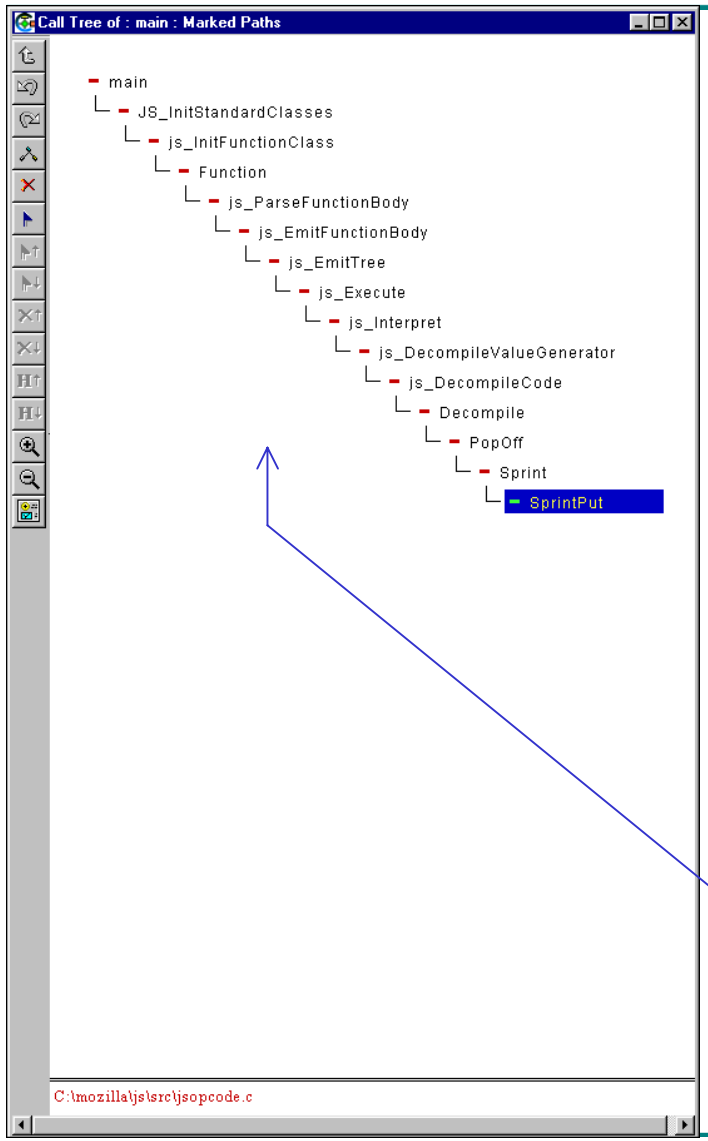
To Search:

- Enter the desired function name in the Search Panel; press the Enter key.

(You may enter leading chars and use the drop-down list.)

Click  in main toolbar to go to other occurrences.

Call-Path of SprintPut( )



- On the previous page, you found SprintPut( ) in the expanded Call-Tree.

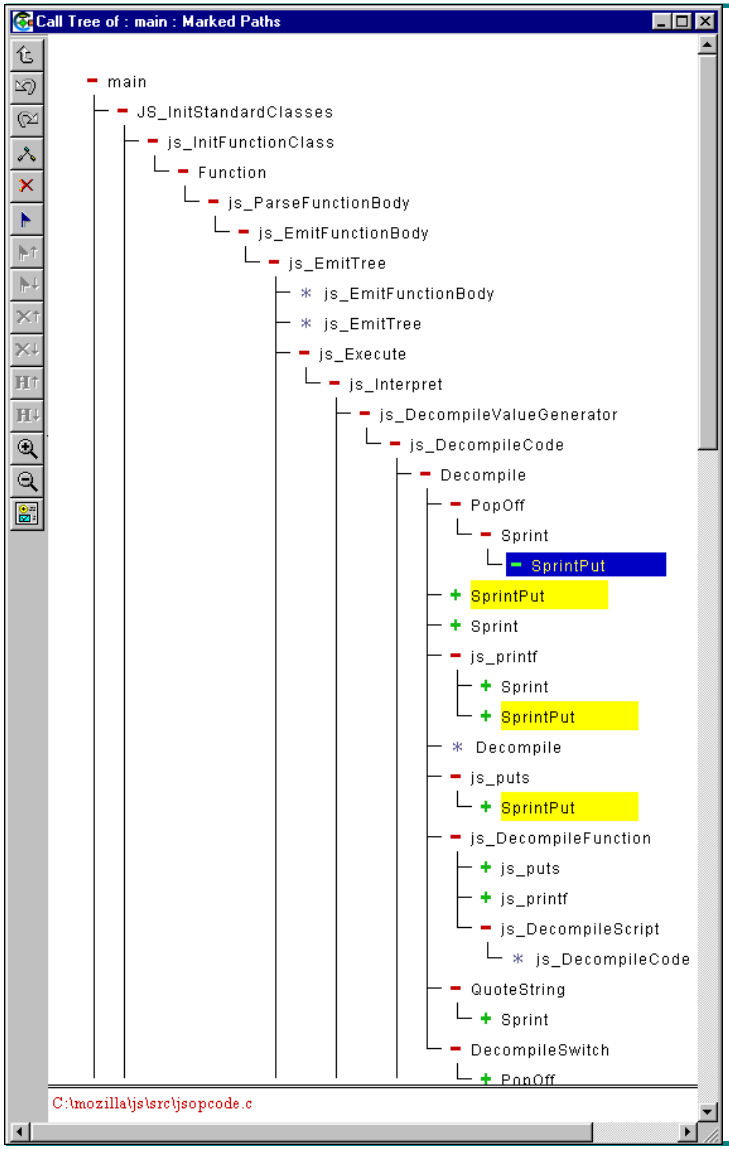
  The Call-Tree is about 30 pages long.

- **How do you extract the call-sequence** from the tree?

  1. Right-click anywhere in the Tree-window; then click <Show Path>.

Now you can see the call-path clearly.

All Call-Paths of SprintPut( )



- Earlier we saw the call-path to only the current-node.

- To extract call-paths to all occurrences of SprintPut():

  1. Right-click in the Tree-window; then click <Show All Paths>.

← Call-paths to All Occurrences of SprintPut():

The leaf-nodes are either

- SprintPut() itself
- or a caller that is already expanded once.

SGV Sarc 907 Broad Oaks Drive, Herndon, Virginia 20170
Phone: 703-904-0678 Fax: 703-904-0155 www.sgvsarc.com