ReuseDistance

0.01

Generated by Doxygen 1.6.3

Sun Oct 21 13:06:00 2012

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 node234_Tag Struct Reference

Public Attributes

- node234 * parent
- node234 * kids [4]
- int counts [4]
- ReuseEntry * elems [3]

4.1.1 Detailed Description

Definition at line 58 of file tree234.c.

4.1.2 Member Data Documentation

4.1.2.1 int node234_Tag::counts[4]

Definition at line 61 of file tree234.c.

4.1.2.2 ReuseEntry* node234_Tag::elems[3]

Definition at line 62 of file tree234.c.

4.1.2.3 node234* node234_Tag::kids[4]

Definition at line 60 of file tree234.c.

4.1.2.4 node234* node234_Tag::parent

Definition at line 59 of file tree234.c.

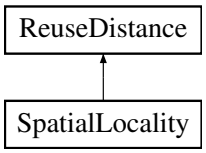
The documentation for this struct was generated from the following file:

- /home/michaell/software/ReuseDistance/tree234.c

4.2 ReuseDistance Class Reference

`#include <ReuseDistance.hpp>`

Inheritance diagram for ReuseDistance:



Public Member Functions

- ReuseDistance (uint64_t w, uint64_t b)
- ReuseDistance (uint64_t w)
- virtual ~ReuseDistance ()
- virtual void Print (std::ostream &f, bool annotate=false)
- virtual void Print (bool annotate=false)
- void PrintFormat (std::ostream &f)
- virtual void Process (ReuseEntry &addr)
- void Process (ReuseEntry *addrs, uint64_t count)
- void Process (std::vector< ReuseEntry > rs)
- void Process (std::vector< ReuseEntry * > addrs)
- ReuseStats * GetStats (uint64_t id)
- void GetIndices (std::vector< uint64_t > &ids)
- virtual void GetActiveAddresses (std::vector< uint64_t > &addrs)

Static Public Attributes

- static const uint64_t DefaultBinIndividual = 32
- static const uint64_t Infinity = INFINITY_REUSE

Protected Member Functions

- void Init (uint64_t w, uint64_t b)
- virtual ReuseStats * GetStats (uint64_t id, bool gen)
- virtual const std::string Describe ()

Protected Attributes

- reuse_map_type< uint64_t, ReuseStats * > stats
- uint64_t capacity
- uint64_t sequence
- uint64_t binindividual
- uint64_t maxtracking

4.2.1 Detailed Description

Tracks reuse distances for a memory address stream. Keep track of the addresses within a specific window of history, whose size can be finite or infinite. For basic usage, see the documentation at http://bit.ly/ScqZVj for the constructors, the Process methods and the Print methods. Also see the simple test file test/test.cpp included in this source package.

Definition at line 86 of file ReuseDistance.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ReuseDistance::ReuseDistance (uint64_t *w*, uint64_t *b*)

Contructs a ReuseDistance object.

Parameters

- ***w*** The maximum window size, or alternatively the maximum possible reuse distance that this tool will find. No window/distance limit is imposed if ReuseDistance::Infinity is used, though you could easily run of of memory.
- ***b*** All distances not greater than b will be tracked individually. All distances are tracked individually if b == ReuseDistance::Infinity. Beyond individual tracking, distances are tracked in bins whose boundaries are the powers of two greater than b (and not exeeding w, of course).

Definition at line 56 of file ReuseDistance.cpp.

4.2.2.2 ReuseDistance::ReuseDistance (uint64_t *w*)

Contructs a ReuseDistance object. Equivalent to calling the other constructor with b == ReuseDistance::DefaultBinIndividual

Definition at line 60 of file ReuseDistance.cpp.

4.2.2.3 ReuseDistance::~ReuseDistance () `[virtual]`

Destroys a ReuseDistance object.

Definition at line 64 of file ReuseDistance.cpp.

4.2.3 Member Function Documentation

4.2.3.1 virtual const std::string ReuseDistance::Describe () `[inline, protected, virtual]`

Definition at line 108 of file ReuseDistance.hpp.

4.2.3.2 void ReuseDistance::GetActiveAddresses (std::vector< uint64_t > & *addrs*) `[virtual]`

Get a std::vector containing all of the addresses currently in this ReuseDistance object's active window.

Parameters

addrs A std::vector which will contain the addresses. It is an error to pass this vector non-empty (that is addrs.size() == 0 is enforced at runtime).

Returns

none

Reimplemented in SpatialLocality.

Definition at line 90 of file ReuseDistance.cpp.

4.2.3.3 void ReuseDistance::GetIndices (std::vector< uint64_t > & *ids*)

Get a std::vector containing all of the unique indices processed by this ReuseDistance object.

Parameters

ids A std::vector which will contain the ids. It is an error to pass this vector non-empty (that is addrs.size() == 0 is enforced at runtime).

Returns

none

Definition at line 82 of file ReuseDistance.cpp.

4.2.3.4 ReuseStats * ReuseDistance::GetStats (uint64_t *id*)

Get the ReuseStats object associated with some unique id.

Parameters

id The unique id.

Returns

The ReuseStats object associated with parameter id, or NULL if no ReuseStats is associate with id.

Definition at line 286 of file ReuseDistance.cpp.

4.2.3.5 ReuseStats * ReuseDistance::GetStats (uint64_t *id*, bool *gen*) [protected, virtual]

Definition at line 245 of file ReuseDistance.cpp.

4.2.3.6 void ReuseDistance::Init (uint64_t *w*, uint64_t *b*) [protected]

Definition at line 40 of file ReuseDistance.cpp.

4.2.3.7 void ReuseDistance::Print (bool *annotate* = `false`) `[virtual]`

Print statistics for this ReuseDistance to std::cout. See the other version of ReuseDistance::Print for information about output format.

Parameters

annotate Also print annotations describing the meaning of output fields, preceded by a '#'.

Returns

none

Definition at line 100 of file ReuseDistance.cpp.

4.2.3.8 virtual void ReuseDistance::Print (std::ostream & *f*, bool *annotate* = `false`) `[virtual]`

Print statistics for this ReuseDistance to an output stream. The first line of the output is 7 tokens: [1] a string identifier for the class (REUSESTATS or SPATIALSTATS), [2] the capacity or window size (0 == unlimited), [3] the maximum individual value being tracked, above which values are tracked by bins whose boundaries are powers of 2, [4] the maximum value to track, above which any value is considered a miss. For ReuseDistance, this is equal to the capacity, for subclasses this can be different. [6] the number of ids that will be printed, [6] the total number of accesses made (the number of ReuseEntry elements that were Process'ed) and [7] the number of accesses that cold-misses or were outside the window range. The stats for individual ids are printed on subsequent lines. The printing of each id begins with a line which is comprised of 4 tokens: [1] a string identifier (REUSEID or SPATIALID), [2] the id, [3] the number of accesses to that id and [4] the number of accesses for that id that were cold-misses or were outside the window range. Each subsequent line contains information about a single bin for that id. These lines have 3 tokens: [1] and [2] the lower and upper boundaries (both inclusive) of the bin and [3] the number of accesses falling into that bin. See also ReuseDistance::PrintFormat

Parameters

f The output stream to print results to.

annotate Also print annotations describing the meaning of output fields, preceded by a '#'.

Returns

none

4.2.3.9 void ReuseDistance::PrintFormat (std::ostream & *f*)

Print information about the output format of ReuseDistance or one of its subclasses

Parameters

f The stream to receive the output.

Returns

none

4.2.3.10 void ReuseDistance::Process (std::vector< ReuseEntry * > *addrs*)

Process multiple memory addresses. Equivalent to calling Process on each element of the input vector.

Parameters

addrs A std::vector of memory addresses to process.

Returns

none

4.2.3.11 void ReuseDistance::Process (std::vector< ReuseEntry > *rs*)

Process multiple memory addresses. Equivalent to calling Process on each element of the input vector.

Parameters

addrs A std::vector of memory addresses to process.

Returns

none

4.2.3.12 void ReuseDistance::Process (ReuseEntry * *addrs*, uint64_t *count*)

Process multiple memory addresses. Equivalent to calling Process on each element of the input array.

Parameters

addrs An array of structures describing memory addresses to process.

count The number of elements in addrs.

Returns

none

Definition at line 104 of file ReuseDistance.cpp.

4.2.3.13 void ReuseDistance::Process (ReuseEntry & *addr*) `[virtual]`

Process a single memory address.

Parameters

addr The structure describing the memory address to process.

Returns

none

Reimplemented in SpatialLocality.

Definition at line 124 of file ReuseDistance.cpp.

4.2.4 Member Data Documentation

4.2.4.1 uint64_t ReuseDistance::binindividual [protected]

Definition at line 103 of file ReuseDistance.hpp.

4.2.4.2 uint64_t ReuseDistance::capacity [protected]

Definition at line 101 of file ReuseDistance.hpp.

4.2.4.3 const uint64_t ReuseDistance::DefaultBinIndividual = 32 [static]

Definition at line 112 of file ReuseDistance.hpp.

4.2.4.4 const uint64_t ReuseDistance::Infinity = INFINITY_REUSE [static]

Definition at line 113 of file ReuseDistance.hpp.

4.2.4.5 uint64_t ReuseDistance::maxtracking [protected]

Definition at line 104 of file ReuseDistance.hpp.

4.2.4.6 uint64_t ReuseDistance::sequence [protected]

Definition at line 102 of file ReuseDistance.hpp.

4.2.4.7 reuse_map_type<uint64_t, ReuseStats*> ReuseDistance::stats [protected]

Definition at line 99 of file ReuseDistance.hpp.

The documentation for this class was generated from the following files:

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp
- /home/michaell/software/ReuseDistance/ReuseDistance.cpp

4.3 ReuseEntry Struct Reference

`#include <ReuseDistance.hpp>`

Public Attributes

- uint64_t id
- uint64_t address

4.3.1 Detailed Description

ReuseEntry is used to pass memory addresses into a ReuseDistance.

id The unique id of the entity which generated the memory address. Statistics are tracked seperately for each unique id. address A memory address.

Definition at line 69 of file ReuseDistance.hpp.

4.3.2 Member Data Documentation

4.3.2.1 uint64_t ReuseEntry::address

Definition at line 71 of file ReuseDistance.hpp.

4.3.2.2 uint64_t ReuseEntry::id

Definition at line 70 of file ReuseDistance.hpp.

The documentation for this struct was generated from the following file:

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp

4.4 ReuseStats Class Reference

`#include <ReuseDistance.hpp>`

Public Member Functions

- ReuseStats (uint64_t idx, uint64_t bin, uint64_t num, uint64_t inv)
- ~ReuseStats ()
- void Update (uint64_t dist)
- void Miss ()
- virtual uint64_t GetMissCount ()
- virtual void Print (std::ostream &f, bool annotate=false)
- void GetSortedDistances (std::vector< uint64_t > &dists)
- uint64_t GetMaximumDistance ()
- uint64_t CountDistance (uint64_t dist)
- uint64_t GetAccessCount ()

Static Public Member Functions

- static void PrintFormat (std::ostream &f)

4.4.1 Detailed Description

ReuseStats holds count of observed reuse distances.

Definition at line 260 of file ReuseDistance.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ReuseStats::ReuseStats (uint64_t *idx*, uint64_t *bin*, uint64_t *num*, uint64_t *inv*) [inline]

Contructs a ReuseStats object.

Parameters

idx The unique id for this ReuseStats

bin Stop collecting individual bins above this value

num Any value above this is considered a miss

inv The value which represents a miss

Definition at line 282 of file ReuseDistance.hpp.

4.4.2.2 ReuseStats::~ReuseStats () [inline]

Destroys a ReuseStats object.

Definition at line 288 of file ReuseDistance.hpp.

4.4.3 Member Function Documentation

4.4.3.1 uint64_t ReuseStats::CountDistance (uint64_t *dist*)

Count the number of times some distance has been observed.

Parameters

dist The distance to count.

Returns

The number of times d has been observed.

Definition at line 310 of file ReuseDistance.cpp.

4.4.3.2 uint64_t ReuseStats::GetAccessCount ()

Count the total number of distances observed.

Returns

The total number of distances observed.

Definition at line 290 of file ReuseDistance.cpp.

4.4.3.3 uint64_t ReuseStats::GetMaximumDistance ()

Get the maximum distance observed.

Returns

The maximum distance observed.

Definition at line 294 of file ReuseDistance.cpp.

4.4.3.4 uint64_t ReuseStats::GetMissCount () `[virtual]`

Get the number of misses. This is equal to the number of times Update(ReuseDistance::Infinity) is called.

Returns

The number of misses to this ReuseDistance object

Definition at line 78 of file ReuseDistance.cpp.

4.4.3.5 void ReuseStats::GetSortedDistances (std::vector< uint64_t > & *dists*)

Get a std::vector containing the distances observed, sorted in ascending order.

Parameters

dists The vector which will hold the sorted distance values. It is an error for dists to be passed in non-empty (that is, dists.size() == 0 is enforced).

Returns

none

4.4.3.6 void ReuseStats::Miss ()

Increment the number of misses. That is, addresses which were not found inside the active address window. This is equivalent Update(0), but is faster.

Returns

none

4.4.3.7 virtual void ReuseStats::Print (std::ostream & *f*, bool *annotate* = `false`) `[virtual]`

Print a summary of the current reuse distances and counts for some id.

Parameters

f The stream to receive the output.

annotate Also print annotations describing the meaning of output fields, preceded by a '#'.

Returns

none

4.4.3.8 static void ReuseStats::PrintFormat (std::ostream & *f*) `[static]`

Print information about the output format of ReuseStats

Parameters

f The stream to receive the output.

Returns

none

4.4.3.9 void ReuseStats::Update (uint64_t *dist*)

Increment the counter for some distance.

Parameters

dist A reuse distance observed in the memory address stream.

Returns

none

Definition at line 305 of file ReuseDistance.cpp.

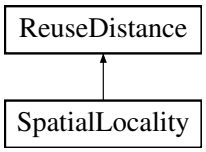
The documentation for this class was generated from the following files:

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp
- /home/michaell/software/ReuseDistance/ReuseDistance.cpp

4.5 SpatialLocality Class Reference

`#include <ReuseDistance.hpp>`

Inheritance diagram for SpatialLocality:



Public Member Functions

- SpatialLocality (uint64_t w, uint64_t b, uint64_t n)
- SpatialLocality (uint64_t w, uint64_t b)
- SpatialLocality (uint64_t w)
- SpatialLocality ()
- virtual ~SpatialLocality ()
- virtual void GetActiveAddresses (std::vector< uint64_t > &addrs)
- virtual void Process (ReuseEntry &addr)

Static Public Attributes

- static const uint64_t DefaultWindowSize = 64

4.5.1 Detailed Description

Finds and tracks spatial locality within a memory address stream. Spatial locality is defined as the minimum distance between the current address and any of the previous N addresses, as in http://www.sdsc.edu/~allans/sc05_locality.pdf. This class allows that window size N to be customized. For basic usage, see the documentation at http://bit.ly/ScqZVj for the constructors, the Process methods and the Print methods. Also see the simple test file test/test.cpp included in this source package.

Definition at line 378 of file ReuseDistance.hpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 SpatialLocality::SpatialLocality (uint64_t *w*, uint64_t *b*, uint64_t *n*) [inline]

Contructs a ReuseDistance object.

Parameters

- ***w*** The maximum window size, which is the maximum number of addresses that will be searched for spatial locality.
- ***b*** All distances not greater than b will be tracked individually. All distances are tracked individually if b == ReuseDistance::Infinity. Beyond individual tracking, distances are tracked in bins whose boundaries are the powers of two greater than b and not greater than n.

n All distances greater than n will be counted as infinite. Use n == ReuseDistance::Infinity for no limit. n >= b is enforced at runtime.

Definition at line 410 of file ReuseDistance.hpp.

4.5.2.2 SpatialLocality::SpatialLocality (uint64_t *w*, uint64_t *b*) [inline]

Constructs a SpatialLocality object. Equivalent to calling the other 3-argument constructor with n == ReuseDistance::Infinity

Definition at line 416 of file ReuseDistance.hpp.

4.5.2.3 SpatialLocality::SpatialLocality (uint64_t *w*) [inline]

Constructs a SpatialLocality object. Equivalent to calling the other 3-argument constructor with w == b and n == ReuseDistance::Infinity

Definition at line 422 of file ReuseDistance.hpp.

4.5.2.4 SpatialLocality::SpatialLocality () [inline]

Constructs a SpatialLocality object. Equivalent to calling the other 3-argument constructor with w == b == SpatialLocality::DefaultWindowSize and n == ReuseDistance::Infinity

Definition at line 428 of file ReuseDistance.hpp.

4.5.2.5 virtual SpatialLocality::~SpatialLocality () [inline, virtual]

Destroys a SpatialLocality object.

Definition at line 433 of file ReuseDistance.hpp.

4.5.3 Member Function Documentation

4.5.3.1 void SpatialLocality::GetActiveAddresses (std::vector< uint64_t > & *addrs*) [virtual]

Get a std::vector containing all of the addresses currently in this SpatialLocality object's active window.

Parameters

addrs A std::vector which will contain the addresses. It is an error to pass this vector non-empty (that is addrs.size() == 0 is enforced at runtime).

Returns

none

Reimplemented from ReuseDistance.

Definition at line 429 of file ReuseDistance.cpp.

4.5.3.2 void SpatialLocality::Process (ReuseEntry & *addr*) [virtual]

Process a single memory address.

Parameters

addr The structure describing the memory address to process.

Returns

none

Reimplemented from ReuseDistance.

Definition at line 377 of file ReuseDistance.cpp.

4.5.4 Member Data Documentation

4.5.4.1 const uint64_t SpatialLocality::DefaultWindowSize = 64 [static]

Definition at line 397 of file ReuseDistance.hpp.

The documentation for this class was generated from the following files:

- /home/michaell/software/ReuseDistance/ReuseDistance.hpp
- /home/michaell/software/ReuseDistance/ReuseDistance.cpp

4.6 tree234_Tag Struct Reference

Public Attributes

- node234 * root

4.6.1 Detailed Description

Definition at line 54 of file tree234.c.

4.6.2 Member Data Documentation

4.6.2.1 node234* tree234_Tag::root

Definition at line 55 of file tree234.c.

The documentation for this struct was generated from the following file:

- /home/michaell/software/ReuseDistance/tree234.c

Chapter 5

File Documentation

5.1 /home/michaell/software/ReuseDistance/foo.cpp File Reference

```
#include <map>
#include <stdint.h>
```

Functions

- int main (int argc, char **argv)

5.1.1 Function Documentation

5.1.1.1 int main (int *argc*, char ** *argv*)

Definition at line 6 of file foo.cpp.

5.2 /home/michaell/software/ReuseDistance/ReuseDistance.cpp File Reference

```
#include <ReuseDistance.hpp>
```

Defines

- #define REUSE_DEBUG
- #define debug_assert(...) assert(__VA_ARGS__)

Functions

- uint64_t uint64abs (uint64_t a)
- uint64_t ShaveBitsPwr2 (uint64_t val)

5.2.1 Define Documentation

5.2.1.1 #define debug_assert(...) assert(__VA_ARGS__)

Definition at line 27 of file ReuseDistance.cpp.

5.2.1.2 #define REUSE_DEBUG

Definition at line 25 of file ReuseDistance.cpp.

5.2.2 Function Documentation

5.2.2.1 uint64_t ShaveBitsPwr2 (uint64_t *val*) [inline]

Definition at line 257 of file ReuseDistance.cpp.

5.2.2.2 uint64_t uint64abs (uint64_t *a*) [inline]

Definition at line 32 of file ReuseDistance.cpp.

5.3 /home/michaell/software/ReuseDistance/ReuseDistance.hpp File Reference

```
#include <assert.h>
#include <stdlib.h>
#include <tree234.h>
#include <algorithm>
#include <iostream>
#include <ostream>
#include <list>
#include <map>
#include <vector>
```

Classes

- struct ReuseEntry
- class ReuseDistance
- class ReuseStats
- class SpatialLocality

Defines

- #define reuse_map_type std::map
- #define TAB "\t"
- #define ENDL "\n"
- #define __seq id
- #define INFINITY_REUSE (0)
- #define INVALID_SPATIAL (0xFFFFFFFFFFFFFFFFL)

Functions

- int reusecmp (void *va, void *vb)

5.3.1 Detailed Description

Author

Michael Laurenzano <michaell@sdsc.edu>

Version

0.01

5.3.2 LICENSE

This file is part of the ReuseDistance tool.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

5.3.3 DESCRIPTION

The ReuseDistanceHandler class allows for calculation and statistic tracking for finding memory reuse distances given a stream of memory addresses and ids.

Definition in file ReuseDistance.hpp.

5.3.4 Define Documentation

5.3.4.1 #define __seq id

Definition at line 54 of file ReuseDistance.hpp.

5.3.4.2 #define ENDL "\n"

Definition at line 52 of file ReuseDistance.hpp.

5.3.4.3 #define INFINITY_REUSE (0)

Definition at line 57 of file ReuseDistance.hpp.

5.3.4.4 #define INVALID_SPATIAL (0xFFFFFFFFFFFFFFFFL)

Definition at line 58 of file ReuseDistance.hpp.

5.3.4.5 #define reuse_map_type std::map

Definition at line 48 of file ReuseDistance.hpp.

5.3.4.6 #define TAB "\t"

Definition at line 51 of file ReuseDistance.hpp.

5.3.5 Function Documentation

5.3.5.1 int reusecmp (void * *va*, void * *vb*)

5.4 /home/michaell/software/ReuseDistance/tree234.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "tree234.h"
#include <tree234.h>
#include <algorithm>
#include <iostream>
#include <ostream>
#include <list>
#include <map>
#include <vector>
```

Classes

- struct tree234_Tag
- struct node234_Tag

Defines

- #define smalloc malloc
- #define sfree free
- #define mknew(typ) ((typ *) smalloc (sizeof (typ)))
- #define LOG(x)
- #define reusecmp(va, vb) (va->__seq - vb->__seq)

Typedefs

- typedef struct node234_Tag node234

Functions

- tree234 * newtree234 ()
- void freetree234 (tree234 *t)
- int count234 (tree234 *t)
- ReuseEntry * add234 (tree234 *t, ReuseEntry *e)
- ReuseEntry * index234 (tree234 *t, int index)
- ReuseEntry * findrelpos234 (tree234 *t, ReuseEntry *e, int *index)
- ReuseEntry * delpos234 (tree234 *t, int index)
- ReuseEntry * del234 (tree234 *t, ReuseEntry *e)

5.4.1 Define Documentation

5.4.1.1 #define LOG(x)

Definition at line 47 of file tree234.c.

5.4.1.2 #define mknew(typ) ((typ *) smalloc (sizeof (typ)))

Definition at line 42 of file tree234.c.

5.4.1.3 #define reusecmp(va, vb) (va->__seq - vb->__seq)

Definition at line 50 of file tree234.c.

5.4.1.4 #define sfree free

Definition at line 40 of file tree234.c.

5.4.1.5 #define smalloc malloc

Definition at line 39 of file tree234.c.

5.4.2 Typedef Documentation

5.4.2.1 typedef struct node234_Tag node234

Definition at line 52 of file tree234.c.

5.4.3 Function Documentation

5.4.3.1 ReuseEntry* add234 (tree234 * *t*, ReuseEntry * *e*)

Definition at line 394 of file tree234.c.

5.4.3.2 int count234 (tree234 * *t*)

Definition at line 111 of file tree234.c.

5.4.3.3 ReuseEntry* del234 (tree234 * *t*, ReuseEntry * *e*)

Definition at line 836 of file tree234.c.

5.4.3.4 ReuseEntry* delpos234 (tree234 * *t*, int *index*)

Definition at line 830 of file tree234.c.

5.4.3.5 ReuseEntry* findrelpos234 (tree234 * *t*, ReuseEntry * *e*, int * *index*)

Definition at line 441 of file tree234.c.

5.4.3.6 void freetree234 (tree234 * *t*)

Definition at line 87 of file tree234.c.

5.4.3.7 ReuseEntry* index234 (tree234 * *t*, int *index*)

Definition at line 402 of file tree234.c.

5.4.3.8 tree234* newtree234 ()

Definition at line 68 of file tree234.c.

5.5 /home/michaell/software/ReuseDistance/tree234.h File Reference

Typedefs

- typedef struct tree234_Tag tree234

Functions

- tree234 * newtree234 ()
- void freetree234 (tree234 *t)
- ReuseEntry * add234 (tree234 *t, ReuseEntry *e)
- ReuseEntry * addpos234 (tree234 *t, ReuseEntry *e, int index)
- ReuseEntry * index234 (tree234 *t, int index)
- ReuseEntry * findrelpos234 (tree234 *t, ReuseEntry *e, int *index)
- ReuseEntry * del234 (tree234 *t, ReuseEntry *e)
- ReuseEntry * delpos234 (tree234 *t, int index)
- int count234 (tree234 *t)

5.5.1 Typedef Documentation

5.5.1.1 typedef struct tree234_Tag tree234

Definition at line 36 of file tree234.h.

5.5.2 Function Documentation

5.5.2.1 ReuseEntry* add234 (tree234 * *t*, ReuseEntry * *e*)

Definition at line 394 of file tree234.c.

5.5.2.2 ReuseEntry* addpos234 (tree234 * *t*, ReuseEntry * *e*, int *index*)

5.5.2.3 int count234 (tree234 * *t*)

Definition at line 111 of file tree234.c.

5.5.2.4 ReuseEntry* del234 (tree234 * *t*, ReuseEntry * *e*)

Definition at line 836 of file tree234.c.

5.5.2.5 ReuseEntry* delpos234 (tree234 * *t*, int *index*)

Definition at line 830 of file tree234.c.

5.5.2.6 ReuseEntry* findrelpos234 (tree234 * *t*, ReuseEntry * *e*, int * *index*)

Definition at line 441 of file tree234.c.

5.5.2.7 void freetree234 (tree234 * *t*)

Definition at line 87 of file tree234.c.

5.5.2.8 ReuseEntry* index234 (tree234 * *t*, int *index*)

Definition at line 402 of file tree234.c.

5.5.2.9 tree234* newtree234 ()

Definition at line 68 of file tree234.c.