zergmap

Generated by Doxygen 1.8.11

Contents

Chapter 1

README

Initial Commit

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 _item Struct Reference

Data Fields

- void * data
- double priority

4.1.1 Field Documentation

4.1.1.1 void * _item::data

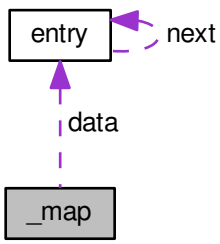
4.1.1.2 double _item::priority

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

- src/dpqueue/heap.c
- src/dpqueue/pqueue.c

4.2 _map Struct Reference

Collaboration diagram for _map:



Data Fields

- struct entry ∗ data
- size_t size
- size_t capacity

4.2.1 Field Documentation

4.2.1.1 size_t _map::capacity

4.2.1.2 struct entry ∗ _map::data

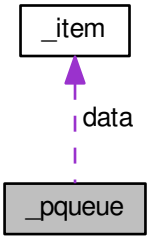
4.2.1.3 size_t _map::size

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

- src/map/hashtable.c
- src/map/map.c

4.3 _pqueue Struct Reference

Collaboration diagram for _pqueue:



Data Fields

- struct _item ∗ data
- int(∗ cmp)(void ∗, void ∗)
- size_t size
- size_t capacity

4.3.1 Field Documentation

4.3.1.1 size_t _pqueue::capacity

4.3.1.2 int(∗ _pqueue::cmp)(void ∗, void ∗)

4.3.1.3 struct _item ∗ _pqueue::data

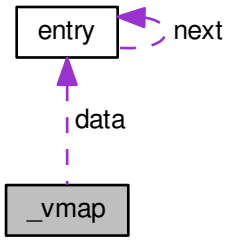
4.3.1.4 size_t _pqueue::size

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

- src/dpqueue/heap.c
- src/dpqueue/pqueue.c

4.4 _vmap Struct Reference

Collaboration diagram for _vmap:



Data Fields

- struct entry ∗ data
- size_t size
- size_t capacity

4.4.1 Field Documentation

4.4.1.1 size_t _vmap::capacity

4.4.1.2 struct entry∗ _vmap::data

4.4.1.3 size_t _vmap::size

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

- src/map/vmap.c

4.5 gpsPayload::accuracy Union Reference

`#include <zergStructs.h>`

Data Fields

- float fAccuracy
- unsigned int iAccuracy

4.5.1 Field Documentation

4.5.1.1 float gpsPayload::accuracy::fAccuracy

4.5.1.2 unsigned int gpsPayload::accuracy::iAccuracy

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

- src/zerg/zergStructs.h

4.6 gpsPayload::altitude Union Reference

`#include <zergStructs.h>`

Data Fields

- float fAltitude
- unsigned int iAltitude

4.6.1 Field Documentation

4.6.1.1 float gpsPayload::altitude::fAltitude

4.6.1.2 unsigned int gpsPayload::altitude::iAltitude

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

- src/zerg/zergStructs.h

4.7 gpsPayload::bearing Union Reference

`#include <zergStructs.h>`

Data Fields

- float fBearing
- unsigned int iBearing

4.7.1 Field Documentation

4.7.1.1 float gpsPayload::bearing::fBearing

4.7.1.2 unsigned int gpsPayload::bearing::iBearing

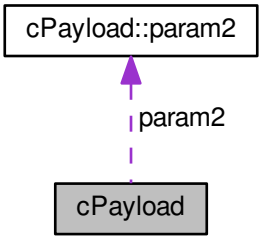
The documentation for this union was generated from the following file:

- src/zerg/zergStructs.h

4.8 cPayload Struct Reference

```
#include <zergStructs.h>
```

Collaboration diagram for cPayload:



Data Structures

- union param2

Data Fields

- unsigned short command
- unsigned short param1
- union cPayload::param2 param2

4.8.1 Field Documentation

4.8.1.1 unsigned short cPayload::command

4.8.1.2 unsigned short cPayload::param1

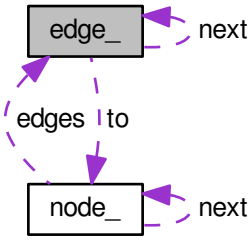
4.8.1.3 union cPayload::param2 cPayload::param2

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

- src/zerg/zergStructs.h

4.9 edge_ Struct Reference

Collaboration diagram for edge_:



Data Fields

- double weight
- struct node_ * to
- struct edge_ * next

4.9.1 Field Documentation

4.9.1.1 struct edge_ * edge_::next

4.9.1.2 struct node_ * edge_::to

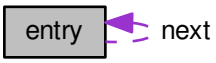
4.9.1.3 double edge_::weight

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

- src/graph/adjlist.c
- src/graph/Graph.c

4.10 entry Struct Reference

Collaboration diagram for entry:



Data Fields

- char * key
- double value
- struct entry * next
- void * value

4.10.1 Field Documentation

4.10.1.1 char * entry::key

4.10.1.2 struct entry * entry::next

4.10.1.3 void* entry::value

4.10.1.4 double entry::value

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

- src/map/hashtable.c
- src/map/map.c
- src/map/vmap.c

4.11 ethernetHeader Struct Reference

```
#include <zergStructs.h>
```

Data Fields

- unsigned char destMac [6]
- unsigned char sourceMac [6]
- unsigned short int etherType

4.11.1 Field Documentation

4.11.1.1 unsigned char ethernetHeader::destMac[6]

4.11.1.2 unsigned short int ethernetHeader::etherType

4.11.1.3 unsigned char ethernetHeader::sourceMac[6]

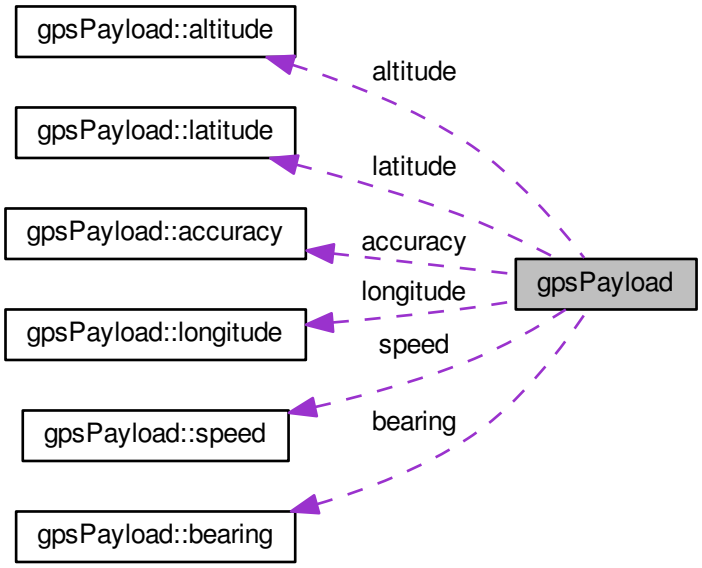
The documentation for this struct was generated from the following file:

- src/zerg/zergStructs.h

4.12 gpsPayload Struct Reference

`#include <zergStructs.h>`

Collaboration diagram for gpsPayload:



Data Structures

- union accuracy
- union altitude
- union bearing
- union latitude
- union longitude
- union speed

Data Fields

- union gpsPayload::longitude longitude
- union gpsPayload::latitude latitude
- union gpsPayload::altitude altitude
- union gpsPayload::bearing bearing
- union gpsPayload::speed speed
- union gpsPayload::accuracy accuracy

4.12.1 Field Documentation

4.12.1.1 union gpsPayload::accuracy gpsPayload::accuracy

4.12.1.2 union gpsPayload::altitude gpsPayload::altitude

4.12.1.3 union gpsPayload::bearing gpsPayload::bearing

4.12.1.4 union gpsPayload::latitude gpsPayload::latitude

4.12.1.5 union gpsPayload::longitude gpsPayload::longitude

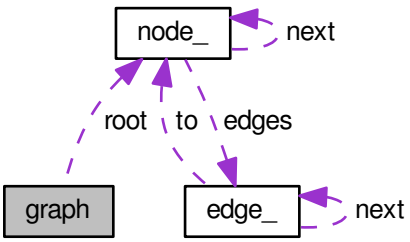
4.12.1.6 union gpsPayload::speed gpsPayload::speed

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

- src/zerg/zergStructs.h

4.13 graph Struct Reference

Collaboration diagram for graph:



Data Fields

- struct node_ * root

4.13.1 Field Documentation

4.13.1.1 struct node_ * graph::root

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

- src/graph/adjlist.c
- src/graph/Graph.c

4.14 ipv4Header Struct Reference

```
#include <zergStructs.h>
```

Data Fields

- unsigned char ipHeaderLength:4
- unsigned char version:4
- unsigned char dscp
- unsigned short ipLength
- unsigned short id
- unsigned short flags
- unsigned char ttl
- unsigned char protocol
- unsigned short checksum
- unsigned int sourceIp
- unsigned int destIp

4.14.1 Field Documentation

4.14.1.1 unsigned short ipv4Header::checksum

4.14.1.2 unsigned int ipv4Header::destIp

4.14.1.3 unsigned char ipv4Header::dscp

4.14.1.4 unsigned short ipv4Header::flags

4.14.1.5 unsigned short ipv4Header::id

4.14.1.6 unsigned char ipv4Header::ipHeaderLength

4.14.1.7 unsigned short ipv4Header::ipLength

4.14.1.8 unsigned char ipv4Header::protocol

4.14.1.9 unsigned int ipv4Header::sourceIp

4.14.1.10 unsigned char ipv4Header::ttl

4.14.1.11 unsigned char ipv4Header::version

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

- src/zerg/zergStructs.h

4.15 ipv6Header Struct Reference

`#include <zergStructs.h>`

Data Fields

- unsigned char version:4
- unsigned char trafficClass
- unsigned int flowLabel:20
- unsigned short payloadLength
- unsigned char nextHeader
- unsigned char hopLimit
- unsigned char source [16]
- unsigned char destination [16]

4.15.1 Field Documentation

4.15.1.1 unsigned char ipv6Header::destination[16]

4.15.1.2 unsigned int ipv6Header::flowLabel

4.15.1.3 unsigned char ipv6Header::hopLimit

4.15.1.4 unsigned char ipv6Header::nextHeader

4.15.1.5 unsigned short ipv6Header::payloadLength

4.15.1.6 unsigned char ipv6Header::source[16]

4.15.1.7 unsigned char ipv6Header::trafficClass

4.15.1.8 unsigned char ipv6Header::version

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

- src/zerg/zergStructs.h

4.16 gpsPayload::latitude Union Reference

`#include <zergStructs.h>`

Data Fields

- double dLat
- unsigned long long iLat

4.16.1 Field Documentation

4.16.1.1 double gpsPayload::latitude::dLat

4.16.1.2 unsigned long long gpsPayload::latitude::iLat

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

- src/zerg/zergStructs.h

4.17 gpsPayload::longitude Union Reference

`#include <zergStructs.h>`

Data Fields

- double dLong
- unsigned long long iLong

4.17.1 Field Documentation

4.17.1.1 double gpsPayload::longitude::dLong

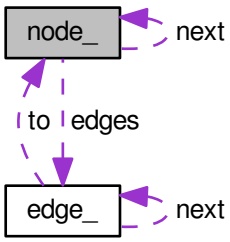
4.17.1.2 unsigned long long gpsPayload::longitude::iLong

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

- src/zerg/zergStructs.h

4.18 node_ Struct Reference

Collaboration diagram for node_:



Data Fields

- char * name
- struct node_ * next
- struct edge_ * edges

4.18.1 Field Documentation

4.18.1.1 struct edge_ * node_::edges

4.18.1.2 char * node_::name

4.18.1.3 struct node_ * node_::next

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

- src/graph/adjlist.c
- src/graph/Graph.c

4.19 cPayload::param2 Union Reference

```
#include <zergStructs.h>
```

Data Fields

- float fParam2
- int iParam2
- unsigned int uiParam2

4.19.1 Field Documentation

4.19.1.1 float cPayload::param2::fParam2

4.19.1.2 int cPayload::param2::iParam2

4.19.1.3 unsigned int cPayload::param2::uiParam2

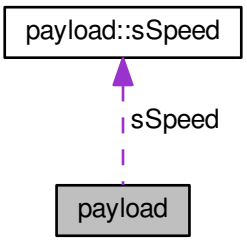
The documentation for this union was generated from the following file:

- src/zerg/zergStructs.h

4.20 payload Struct Reference

`#include <zergStructs.h>`

Collaboration diagram for payload:



Data Structures

- union sSpeed

Data Fields

- unsigned int currHitPoints:24
- unsigned char armor
- unsigned int maxHitPoints:24
- unsigned char type
- union payload::sSpeed sSpeed

4.20.1 Field Documentation

4.20.1.1 unsigned char payload::armor

4.20.1.2 unsigned int payload::currHitPoints

4.20.1.3 unsigned int payload::maxHitPoints

4.20.1.4 union payload::sSpeed payload::sSpeed

4.20.1.5 unsigned char payload::type

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

- src/zerg/zergStructs.h

4.21 pcapFileHeader Struct Reference

```
#include <zergStructs.h>
```

Data Fields

- unsigned int fileTypeId
- unsigned short majorVersion
- unsigned short minorVersion
- unsigned int gmtOffset
- unsigned int accDelta
- unsigned int maxLength
- unsigned int linkLayerType

4.21.1 Field Documentation

4.21.1.1 unsigned int pcapFileHeader::accDelta

4.21.1.2 unsigned int pcapFileHeader::fileTypeId

4.21.1.3 unsigned int pcapFileHeader::gmtOffset

4.21.1.4 unsigned int pcapFileHeader::linkLayerType

4.21.1.5 unsigned short pcapFileHeader::majorVersion

4.21.1.6 unsigned int pcapFileHeader::maxLength

4.21.1.7 unsigned short pcapFileHeader::minorVersion

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

- src/zerg/zergStructs.h

4.22 pcapPacketHeader Struct Reference

`#include <zergStructs.h>`

Data Fields

- unsigned int unixEpoch
- unsigned int microEpoch
- unsigned int lengthOfData
- unsigned int fullLength

4.22.1 Field Documentation

4.22.1.1 unsigned int pcapPacketHeader::fullLength

4.22.1.2 unsigned int pcapPacketHeader::lengthOfData

4.22.1.3 unsigned int pcapPacketHeader::microEpoch

4.22.1.4 unsigned int pcapPacketHeader::unixEpoch

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

- src/zerg/zergStructs.h

4.23 gpsPayload::speed Union Reference

`#include <zergStructs.h>`

Data Fields

- float fSpeed
- unsigned int iSpeed

4.23.1 Field Documentation

4.23.1.1 float gpsPayload::speed::fSpeed

4.23.1.2 unsigned int gpsPayload::speed::iSpeed

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

- src/zerg/zergStructs.h

4.24 payload::sSpeed Union Reference

`#include <zergStructs.h>`

Data Fields

- float fSpeed
- unsigned int iSpeed

4.24.1 Field Documentation

4.24.1.1 float payload::sSpeed::fSpeed

4.24.1.2 unsigned int payload::sSpeed::iSpeed

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

- src/zerg/zergStructs.h

4.25 udpHeader Struct Reference

`#include <zergStructs.h>`

Data Fields

- unsigned short sourcePort
- unsigned short destPort
- unsigned short length
- unsigned short checksum

4.25.1 Field Documentation

4.25.1.1 unsigned short udpHeader::checksum

4.25.1.2 unsigned short udpHeader::destPort

4.25.1.3 unsigned short udpHeader::length

4.25.1.4 unsigned short udpHeader::sourcePort

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

- src/zerg/zergStructs.h

4.26 zergPacket Struct Reference

`#include <zergStructs.h>`

Data Fields

- char type:4
- char version:4
- unsigned int totalLength:24
- unsigned short sourceId
- unsigned short destinationId
- unsigned int sequenceId

4.26.1 Field Documentation

4.26.1.1 unsigned short zergPacket::destinationId

4.26.1.2 unsigned int zergPacket::sequenceId

4.26.1.3 unsigned short zergPacket::sourceId

4.26.1.4 unsigned int zergPacket::totalLength

4.26.1.5 char zergPacket::type

4.26.1.6 char zergPacket::version

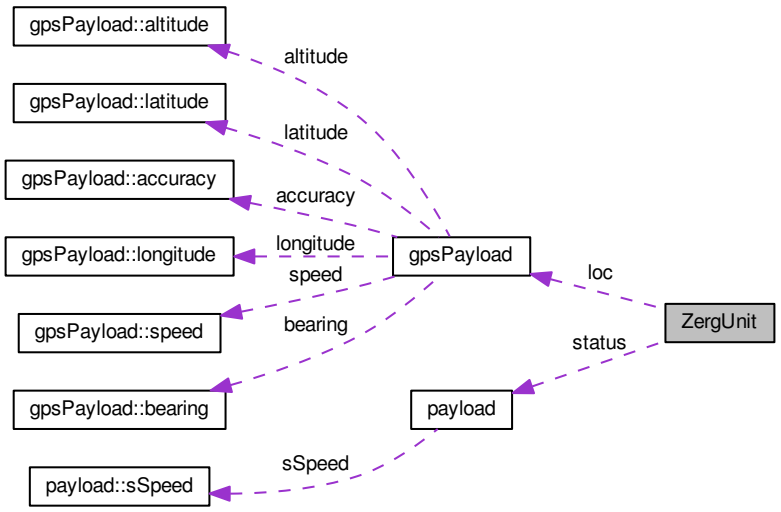
The documentation for this struct was generated from the following file:

- src/zerg/zergStructs.h

4.27 ZergUnit Struct Reference

`#include <zergStructs.h>`

Collaboration diagram for ZergUnit:



Data Fields

- unsigned short id
- bool dupe
- payload * status
- gpsPayload * loc
- int seen

4.27.1 Field Documentation

4.27.1.1 bool ZergUnit::dupe

4.27.1.2 unsigned short ZergUnit::id

4.27.1.3 gpsPayload* ZergUnit::loc

4.27.1.4 int ZergUnit::seen

4.27.1.5 payload* ZergUnit::status

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

- src/zerg/zergStructs.h

Chapter 5

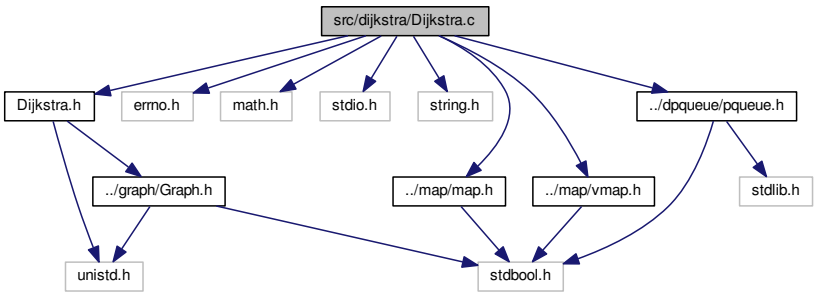
File Documentation

5.1 README.md File Reference

5.2 src/dijkstra/Dijkstra.c File Reference

```
#include "Dijkstra.h"
#include <errno.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include "../map/map.h"
#include "../map/vmap.h"
#include "../dpqueue/pqueue.h"
```

Include dependency graph for Dijkstra.c:



Functions

- ssize_t Dijkstra_path (const Graph *g, const char *start, const char *end, char ***path)

  Find path between start and end.
- void Dijkstra_solveMaze (char **mazeFromFile, char **route, size_t hops)

  Changes 2d array of the maze and changes the route to show the solved maze.

5.2.1 Function Documentation

5.2.1.1 ssize_t Dijkstra_path (const Graph * g, const char * start, const char * end, char *** path)

Find path between start and end.

Parameters

g	Graph to traverse
start	starting node in graph
end	ending node in graph
path	output parameter which is filled with shortest start-to-end path

Returns

number of nodes in path (negative if error or disconnected)

5.2.1.2 void Dijkstra_solveMaze (char ** mazeFromFile, char ** route, size_t hops)

Changes 2d array of the maze and changes the route to show the solved maze.

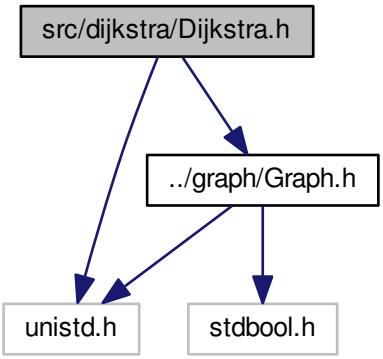
Parameters

mazeFromFile	2d array of maze
route	Nodes in graph that lead to end
hops	Number of hops for tracking in loop

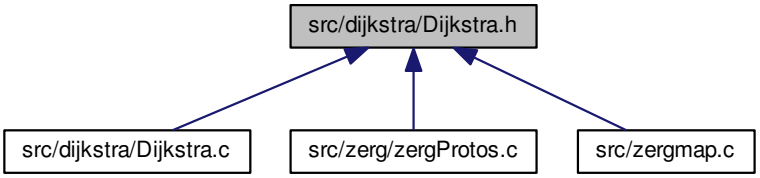
5.3 src/dijkstra/Dijkstra.h File Reference

```
#include <unistd.h>
#include "../graph/Graph.h"
```

Include dependency graph for Dijkstra.h:



This graph shows which files directly or indirectly include this file:



Functions

- ssize_t Dijkstra_path (const Graph *g, const char *start, const char *end, char ***path)

  Find path between start and end.
- void Dijkstra_solveMaze (char **mazeFromFile, char **route, size_t hops)

  Changes 2d array of the maze and changes the route to show the solved maze.

5.3.1 Function Documentation

5.3.1.1 ssize_t Dijkstra_path (const Graph * *g*, const char * *start*, const char * *end*, char *** *path*)

Find path between start and end.

Parameters

g	Graph to traverse
start	starting node in graph
end	ending node in graph
path	output parameter which is filled with shortest start-to-end path

Returns

number of nodes in path (negative if error or disconnected)

5.3.1.2 void Dijkstra_solveMaze (char ** *mazeFromFile,* char ** *route,* size_t *hops*)

Changes 2d array of the maze and changes the route to show the solved maze.

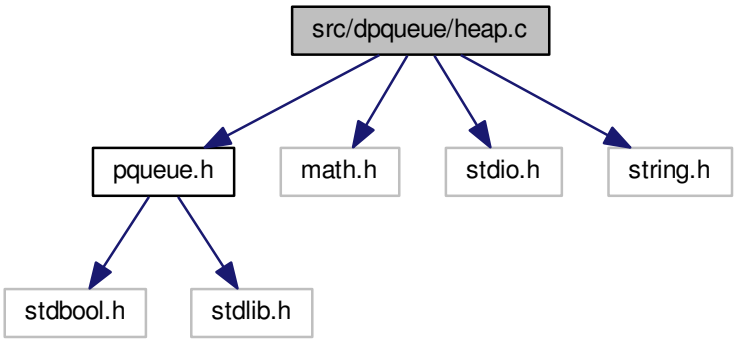
Parameters

mazeFromFile	2d array of maze
route	Nodes in graph that lead to end
hops	Number of hops for tracking in loop

5.4 src/dpqueue/heap.c File Reference

```
#include "pqueue.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
```

Include dependency graph for heap.c:



Data Structures

- struct _item
- struct _pqueue

Functions

- pqueue * pqueue_create (int(*cmp)(void *, void *))
- size_t pqueue_size (const pqueue *pq)
- bool pqueue_enqueue (pqueue *pq, void *item, double priority)
- bool pqueue_reprioritize (pqueue *pq, void *item, double priority)
- double pqueue_dequeue (pqueue *pq, void **item)
- void * pqueue_search (const pqueue *pq, void *item)
- void pqueue_destroy (pqueue *pq)
- void heap_print (const pqueue *pq)

5.4.1 Function Documentation

5.4.1.1 void heap_print (const pqueue * *pq*)

5.4.1.2 pqueue* pqueue_create (int(*)(void *, void *) *cmp*)

5.4.1.3 double pqueue_dequeue (pqueue * *pq*, void ** *item*)

5.4.1.4 void pqueue_destroy (pqueue * *pq*)

5.4.1.5 bool pqueue_enqueue (pqueue * *pq*, void * *item*, double *priority*)

5.4.1.6 bool pqueue_reprioritize (pqueue * *pq*, void * *item*, double *priority*)

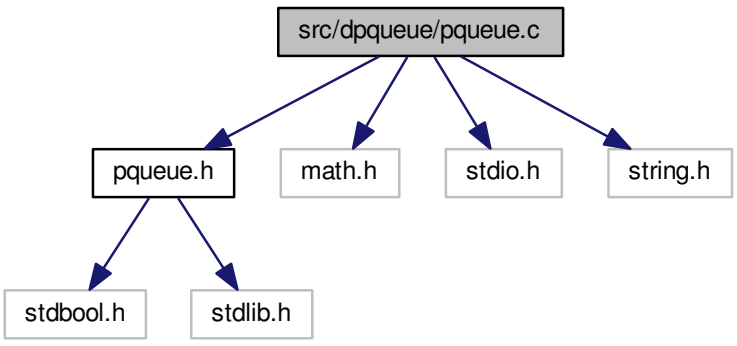
5.4.1.7 void* pqueue_search (const pqueue * *pq*, void * *item*)

5.4.1.8 size_t pqueue_size (const pqueue * *pq*)

5.5 src/dpqueue/pqueue.c File Reference

```
#include "pqueue.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
```

Include dependency graph for pqueue.c:



Data Structures

- struct _item
- struct _pqueue

Functions

- pqueue * pqueue_create (int(*cmp)(void *, void *))
- size_t pqueue_size (const pqueue *pq)
- bool pqueue_enqueue (pqueue *pq, void *item, double priority)
- bool pqueue_reprioritize (pqueue *pq, void *item, double priority)
- double pqueue_dequeue (pqueue *pq, void **item)
- void * pqueue_search (const pqueue *pq, void *item)
- void pqueue_destroy (pqueue *pq)
- void heap_print (const pqueue *pq)

5.5.1 Function Documentation

5.5.1.1 void heap_print (const pqueue * *pq*)

5.5.1.2 pqueue* pqueue_create (int(*)(void *, void *) *cmp*)

5.5.1.3 double pqueue_dequeue (pqueue * *pq*, void ** *item*)

5.5.1.4 void pqueue_destroy (pqueue * *pq*)

5.5.1.5 bool pqueue_enqueue (pqueue * *pq*, void * *item*, double *priority*)

5.5.1.6 bool pqueue_reprioritize (pqueue * *pq*, void * *item*, double *priority*)

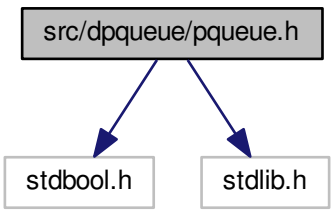
5.5.1.7 void* pqueue_search (const pqueue * *pq*, void * *item*)

5.5.1.8 size_t pqueue_size (const pqueue * *pq*)

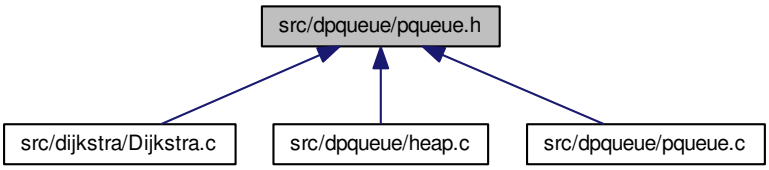
5.6 src/dpqueue/pqueue.h File Reference

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

Include dependency graph for pqueue.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct _pqueue pqueue

Functions

- pqueue * pqueue_create (int(*cmp)(void *, void *))
- size_t pqueue_size (const pqueue *pq)
- bool pqueue_enqueue (pqueue *pq, void *item, double priority)
- bool pqueue_reprioritize (pqueue *pq, void *item, double priority)
- double pqueue_dequeue (pqueue *pq, void **item)
- void * pqueue_search (const pqueue *pq, void *item)
- void pqueue_destroy (pqueue *pq)

5.6.1 Typedef Documentation

5.6.1.1 typedef struct _pqueue pqueue

5.6.2 Function Documentation

5.6.2.1 pqueue* pqueue_create (int(*)(void *, void *) *cmp*)

5.6.2.2 double pqueue_dequeue (pqueue * *pq*, void ** *item*)

5.6.2.3 void pqueue_destroy (pqueue * *pq*)

5.6.2.4 bool pqueue_enqueue (pqueue * *pq*, void * *item*, double *priority*)

5.6.2.5 bool pqueue_reprioritize (pqueue * *pq*, void * *item*, double *priority*)

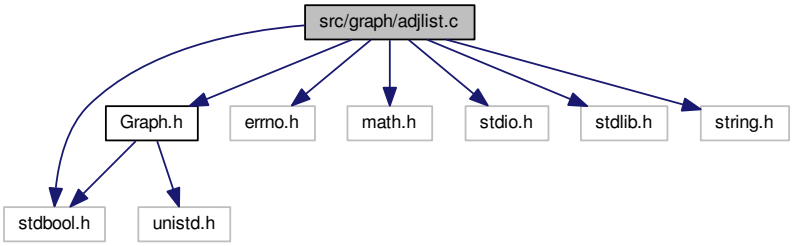
5.6.2.6 void* pqueue_search (const pqueue * *pq*, void * *item*)

5.6.2.7 size_t pqueue_size (const pqueue * *pq*)

5.7 src/graph/adjlist.c File Reference

```
#include "Graph.h"
#include <errno.h>
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for adjlist.c:



Data Structures

- struct edge_
- struct node_
- struct graph

Macros

- #define _XOPEN_SOURCE 500

Typedefs

- typedef struct edge_ edge_
- typedef struct node_ node_

Functions

- Graph * Graph_create (void)

  Create an empty graph structure.
- bool Graph_addNode (Graph *g, const char *name)

  Adds a node to the graph (does not add duplicates)
- bool Graph_addEdge (Graph *g, const char *from, const char *to, double weight)

  Adds an edge to the graph (does not add duplicates)
- bool Graph_isAdjacent (const Graph *g, const char *from, const char *to)

  Checks if two nodes are adjacent.
- ssize_t Graph_getNodes (const Graph *g, char ***nodes)

provide list of nodes of a graph

- ssize_t Graph_getNeighbors (const Graph *g, const char *name, char ***neighbors)

  provide list of neighbor's names for a given node
- double Graph_getEdgeWeight (const Graph *g, const char *from, const char *to)

  Provide edge weight between two nodes.
- bool Graph_updateEdgeWeight (const Graph *g, const char *from, const char *to, double weight)

  Update edge weight between two nodes.
- void Graph_deleteNode (Graph *g, const char *name)

  Remove a node from the graph.
- void Graph_deleteEdge (Graph *g, const char *from, const char *to)

  Remove an edge from the graph.
- void Graph_print (const Graph *g)

  Prints graph to stdout.
- void Graph_disassemble (Graph *g)

  Destroy the graph scaffolding without affecting the underlying data.

5.7.1 Macro Definition Documentation

5.7.1.1 #define _XOPEN_SOURCE 500

5.7.2 Typedef Documentation

5.7.2.1 typedef struct edge_ edge_

5.7.2.2 typedef struct node_ node_

5.7.3 Function Documentation

5.7.3.1 bool Graph_addEdge (Graph * *g*, const char * *from*, const char * *to*, double *weight*)

Adds an edge to the graph (does not add duplicates)

Parameters

g	Graph to modify
from	Name of source node
to	Name of destination node
weight	Cost of the edge

Returns

true for successful add

5.7.3.2 bool Graph_addNode (Graph * *g*, const char * *name*)

Adds a node to the graph (does not add duplicates)

Parameters

g	Graph to modify
name	Name of new node

Returns

true for successful add

5.7.3.3 Graph* Graph_create (void)

Create an empty graph structure.

Returns

the graph structure, or NULL on error

5.7.3.4 void Graph_deleteEdge (Graph * *g*, const char * *from*, const char * *to*)

Remove an edge from the graph.

Parameters

g	Graph to alter
from	Starting node of edge
to	Ending node of edge

5.7.3.5 void Graph_deleteNode (Graph * *g*, const char * *name*)

Remove a node from the graph.

Parameters

g	Graph to alter
name	Name of node to remove

5.7.3.6 void Graph_disassemble (Graph * *g*)

Destroy the graph scaffolding without affecting the underlying data.

Parameters

g	Graph to disassemble

5.7.3.7 double Graph_getEdgeWeight (const Graph * g, const char * from, const char * to)

Provide edge weight between two nodes.

Parameters

g	Graph to inspect
from	Starting node
to	neighbor node

Returns

weight of edge (NAN if edge does not exist)

5.7.3.8 ssize_t Graph_getNeighbors (const Graph * g, const char * name, char *** neighbors)

provide list of neighbor's names for a given node

Parameters

g	Graph to inspect
name	node's name to find neighbors for
neighbors	input parameter to store array of neighbors' names

Returns

number of neighbors found (-1 for error)

5.7.3.9 ssize_t Graph_getNodes (const Graph * g, char *** nodes)

provide list of nodes of a graph

Parameters

g	Graph to inspect
nodes	input parameter to store array of nodes' names

Returns

number of nodes found (-1 for error)

5.7.3.10 bool Graph_isAdjacent (const Graph * g, const char * from, const char * to)

Checks if two nodes are adjacent.

Parameters

g	Graph to inspect
from	Name of source node
to	Name of destination node

Returns

True if nodes are adjacent, false otherwise

5.7.3.11 void Graph_print (const Graph * g)

Prints graph to stdout.

Parameters

g	Graph to print

5.7.3.12 bool Graph_updateEdgeWeight (const Graph * g, const char * from, const char * to, double weight)

Update edge weight between two nodes.

Parameters

g	Graph to inspect
from	Starting node
to	neighbor node
weight	new weight of edge

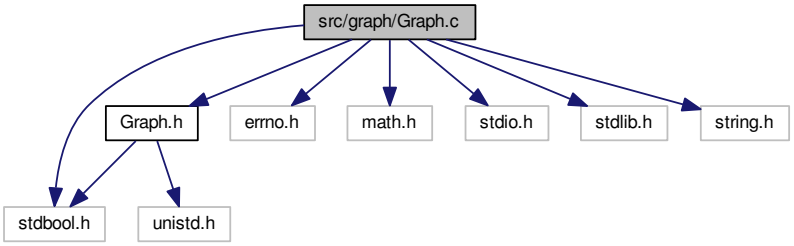
Returns

True if successfuly updated, False if failed to update

5.8 src/graph/Graph.c File Reference

```
#include "Graph.h"
#include <errno.h>
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for Graph.c:



Data Structures

- struct edge_
- struct node_
- struct graph

Macros

- #define _XOPEN_SOURCE 500

Typedefs

- typedef struct edge_ edge_
- typedef struct node_ node_

Functions

- Graph * Graph_create (void)

  Create an empty graph structure.
- bool Graph_addNode (Graph *g, const char *name)

  Adds a node to the graph (does not add duplicates)
- bool Graph_addEdge (Graph *g, const char *from, const char *to, double weight)

  Adds an edge to the graph (does not add duplicates)
- bool Graph_isAdjacent (const Graph *g, const char *from, const char *to)

  Checks if two nodes are adjacent.
- ssize_t Graph_getNodes (const Graph *g, char ***nodes)

  provide list of nodes of a graph
- ssize_t Graph_getNeighbors (const Graph *g, const char *name, char ***neighbors)

  provide list of neighbor's names for a given node
- double Graph_getEdgeWeight (const Graph *g, const char *from, const char *to)

  Provide edge weight between two nodes.
- bool Graph_updateEdgeWeight (const Graph *g, const char *from, const char *to, double weight)

  Update edge weight between two nodes.
- void Graph_deleteNode (Graph *g, const char *name)

Remove a node from the graph.

- void Graph_deleteEdge (Graph *g, const char *from, const char *to)

  Remove an edge from the graph.

- void Graph_print (const Graph *g)

  Prints graph to stdout.

- void Graph_disassemble (Graph *g)

  Destroy the graph scaffolding without affecting the underlying data.

5.8.1 Macro Definition Documentation

5.8.1.1 #define _XOPEN_SOURCE 500

5.8.2 Typedef Documentation

5.8.2.1 typedef struct edge_ edge_

5.8.2.2 typedef struct node_ node_

5.8.3 Function Documentation

5.8.3.1 bool Graph_addEdge (Graph * *g*, const char * *from*, const char * *to*, double *weight*)

Adds an edge to the graph (does not add duplicates)

Parameters

g	Graph to modify
from	Name of source node
to	Name of destination node
weight	Cost of the edge

Returns

true for successful add

5.8.3.2 bool Graph_addNode (Graph * *g*, const char * *name*)

Adds a node to the graph (does not add duplicates)

Parameters

g	Graph to modify
name	Name of new node

Returns

true for successful add

5.8.3.3 Graph* Graph_create (void)

Create an empty graph structure.

Returns

the graph structure, or NULL on error

5.8.3.4 void Graph_deleteEdge (Graph * *g*, const char * *from*, const char * *to*)

Remove an edge from the graph.

Parameters

g	Graph to alter
from	Starting node of edge
to	Ending node of edge

5.8.3.5 void Graph_deleteNode (Graph * *g*, const char * *name*)

Remove a node from the graph.

Parameters

g	Graph to alter
name	Name of node to remove

5.8.3.6 void Graph_disassemble (Graph * *g*)

Destroy the graph scaffolding without affecting the underlying data.

Parameters

g	Graph to disassemble

5.8.3.7 double Graph_getEdgeWeight (const Graph * *g*, const char * *from*, const char * *to*)

Provide edge weight between two nodes.

Parameters

g	Graph to inspect
from	Starting node
to	neighbor node

Returns

weight of edge (NAN if edge does not exist)

5.8.3.8 ssize_t Graph_getNeighbors (const Graph * g, const char * name, char *** neighbors)

provide list of neighbor's names for a given node

Parameters

g	Graph to inspect
name	node's name to find neighbors for
neighbors	input parameter to store array of neighbors' names

Returns

number of neighbors found (-1 for error)

5.8.3.9 ssize_t Graph_getNodes (const Graph * g, char *** nodes)

provide list of nodes of a graph

Parameters

g	Graph to inspect
nodes	input parameter to store array of nodes' names

Returns

number of nodes found (-1 for error)

5.8.3.10 bool Graph_isAdjacent (const Graph * g, const char * from, const char * to)

Checks if two nodes are adjacent.

Parameters

g	Graph to inspect
from	Name of source node
to	Name of destination node

Returns

True if nodes are adjacent, false otherwise

5.8.3.11 void Graph_print (const Graph * g)

Prints graph to stdout.

Parameters

g	Graph to print

5.8.3.12 bool Graph_updateEdgeWeight (const Graph * g, const char * from, const char * to, double weight)

Update edge weight between two nodes.

Parameters

g	Graph to inspect
from	Starting node
to	neighbor node
weight	new weight of edge

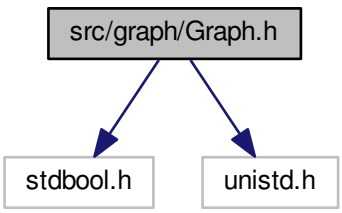
Returns

True if successfuly updated, False if failed to update

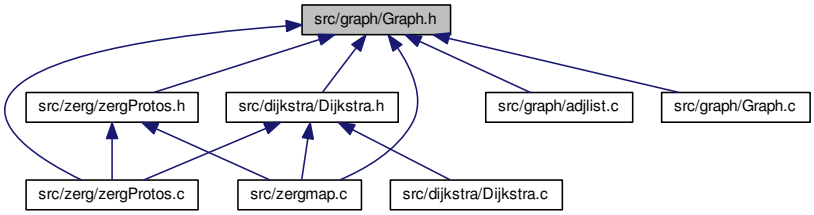
5.9 src/graph/Graph.h File Reference

```
#include <stdbool.h>
#include <unistd.h>
```

Include dependency graph for Graph.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct graph Graph

Functions

- Graph * Graph_create (void)
  Create an empty graph structure.
- bool Graph_addNode (Graph *g, const char *name)
  Adds a node to the graph (does not add duplicates)
- bool Graph_addEdge (Graph *g, const char *from, const char *to, double weight)
  Adds an edge to the graph (does not add duplicates)
- bool Graph_isAdjacent (const Graph *g, const char *from, const char *to)
  Checks if two nodes are adjacent.
- ssize_t Graph_getNodes (const Graph *g, char ***nodes)
  provide list of nodes of a graph
- ssize_t Graph_getNeighbors (const Graph *g, const char *name, char ***neighbors)
  provide list of neighbor's names for a given node
- double Graph_getEdgeWeight (const Graph *g, const char *from, const char *to)
  Provide edge weight between two nodes.
- bool Graph_updateEdgeWeight (const Graph *g, const char *from, const char *to, double weight)
  Update edge weight between two nodes.
- void Graph_deleteNode (Graph *g, const char *name)
  Remove a node from the graph.
- void Graph_deleteEdge (Graph *g, const char *from, const char *to)
  Remove an edge from the graph.
- void Graph_print (const Graph *g)
  Prints graph to stdout.
- void Graph_disassemble (Graph *g)
  Destroy the graph scaffolding without affecting the underlying data.

5.9.1 Typedef Documentation

5.9.1.1 typedef struct graph Graph

5.9.2 Function Documentation

5.9.2.1 bool Graph_addEdge (Graph * *g*, const char * *from*, const char * *to*, double *weight*)

Adds an edge to the graph (does not add duplicates)

Parameters

g	Graph to modify
from	Name of source node
to	Name of destination node
weight	Cost of the edge

Returns

true for successful add

5.9.2.2 bool Graph_addNode (Graph * g, const char * name)

Adds a node to the graph (does not add duplicates)

Parameters

g	Graph to modify
name	Name of new node

Returns

true for successful add

5.9.2.3 Graph* Graph_create (void)

Create an empty graph structure.

Returns

the graph structure, or NULL on error

5.9.2.4 void Graph_deleteEdge (Graph * g, const char * from, const char * to)

Remove an edge from the graph.

Parameters

g	Graph to alter
from	Starting node of edge
to	Ending node of edge

5.9.2.5 void Graph_deleteNode (Graph * g, const char * name)

Remove a node from the graph.

Parameters

g	Graph to alter
name	Name of node to remove

5.9.2.6 void Graph_disassemble (Graph * g)

Destroy the graph scaffolding without affecting the underlying data.

Parameters

g	Graph to disassemble

5.9.2.7 double Graph_getEdgeWeight (const Graph * g, const char * from, const char * to)

Provide edge weight between two nodes.

Parameters

g	Graph to inspect
from	Starting node
to	neighbor node

Returns

weight of edge (NAN if edge does not exist)

5.9.2.8 ssize_t Graph_getNeighbors (const Graph * g, const char * name, char *** neighbors)

provide list of neighbor's names for a given node

Parameters

g	Graph to inspect
name	node's name to find neighbors for
neighbors	input parameter to store array of neighbors' names

Returns

number of neighbors found (-1 for error)

5.9.2.9 ssize_t Graph_getNodes (const Graph * g, char *** nodes)

provide list of nodes of a graph

Parameters

g	Graph to inspect
nodes	input parameter to store array of nodes' names

Returns

number of nodes found (-1 for error)

5.9.2.10 bool Graph_isAdjacent (const Graph * g, const char * from, const char * to)

Checks if two nodes are adjacent.

Parameters

g	Graph to inspect
from	Name of source node
to	Name of destination node

Returns

True if nodes are adjacent, false otherwise

5.9.2.11 void Graph_print (const Graph * g)

Prints graph to stdout.

Parameters

g	Graph to print

5.9.2.12 bool Graph_updateEdgeWeight (const Graph * g, const char * from, const char * to, double weight)

Update edge weight between two nodes.

Parameters

g	Graph to inspect
from	Starting node
to	neighbor node
weight	new weight of edge

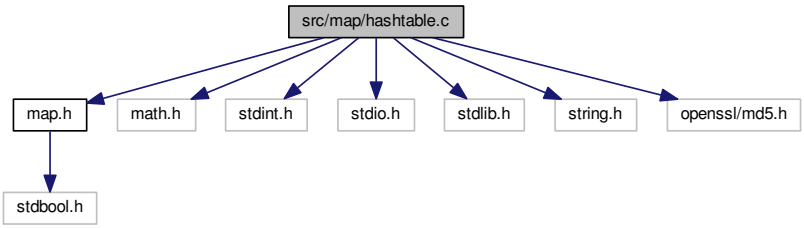
Returns

True if successfuly updated, False if failed to update

5.10 src/map/hashtable.c File Reference

```
#include "map.h"
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/md5.h>
```

Include dependency graph for hashtable.c:



Data Structures

- struct entry
- struct _map

Macros

- #define _XOPEN_SOURCE 500

Functions

- map ∗ map_create (void)

  Creates an empty map structure.
- bool map_insert (map ∗m, const char ∗key, double value)

  Inserts new key and value into map.
- bool map_exists (map ∗m, const char ∗key)

  Checks if key exists in map.
- double map_lookup (map ∗m, const char ∗key)

  Returns Value of specified key.
- void map_destroy (map ∗m)

  Breaks down map and frees memory.
- void hashtable_print (map ∗m)

5.10.1 Macro Definition Documentation

5.10.1.1 #define _XOPEN_SOURCE 500

5.10.2 Function Documentation

5.10.2.1 void hashtable_print (map * m)

5.10.2.2 map* map_create (void)

Creates an empty map structure.

Returns

Pointer to new map in memory

5.10.2.3 void map_destroy (map * m)

Breaks down map and frees memory.

Parameters

m	Map to destroy

5.10.2.4 bool map_exists (map * m, const char * key)

Checks if key exists in map.

Parameters

m	Map to inspect
key	Key value to look for in map

Returns

True if found successfully, False if not found

5.10.2.5 bool map_insert (map * m, const char * key, double value)

Inserts new key and value into map.

Parameters

m	Map to insert into
key	Key value to add to map
value	Value for specified key

Returns

True if added successfully, False if failed

5.10.2.6 double map_lookup (map * *m*, const char * *key*)

Returns Value of specified key.

Parameters

m	Map to inspect
key	Key value to look for in map

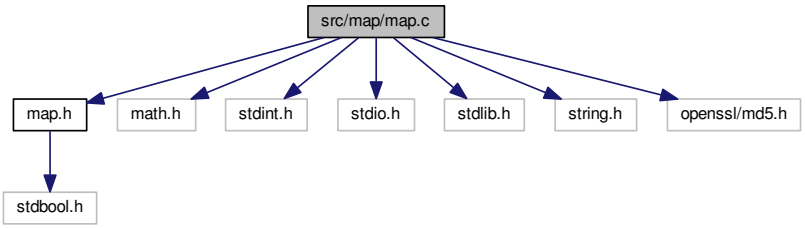
Returns

Value at key specified

5.11 src/map/map.c File Reference

```
#include "map.h"
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/md5.h>
```

Include dependency graph for map.c:



Data Structures

- struct entry
- struct _map

Macros

- #define _XOPEN_SOURCE 500

Functions

- map ∗ map_create (void)
  Creates an empty map structure.
- bool map_insert (map ∗m, const char ∗key, double value)
  Inserts new key and value into map.
- bool map_exists (map ∗m, const char ∗key)
  Checks if key exists in map.
- double map_lookup (map ∗m, const char ∗key)
  Returns Value of specified key.
- void map_destroy (map ∗m)
  Breaks down map and frees memory.
- void hashtable_print (map ∗m)

5.11.1 Macro Definition Documentation

5.11.1.1 #define _XOPEN_SOURCE 500

5.11.2 Function Documentation

5.11.2.1 void hashtable_print (map ∗ *m*)

5.11.2.2 map∗ map_create (void)

Creates an empty map structure.

Returns

Pointer to new map in memory

5.11.2.3 void map_destroy (map ∗ *m*)

Breaks down map and frees memory.

Parameters

m	Map to destroy

5.11.2.4 bool map_exists (map ∗ *m,* const char ∗ *key*)

Checks if key exists in map.

Parameters

m	Map to inspect
key	Key value to look for in map

Returns

True if found successfully, False if not found

5.11.2.5 bool map_insert (map * *m*, const char * *key*, double *value*)

Inserts new key and value into map.

Parameters

m	Map to insert into
key	Key value to add to map
value	Value for specified key

Returns

True if added successfully, False if failed

5.11.2.6 double map_lookup (map * *m*, const char * *key*)

Returns Value of specified key.

Parameters

m	Map to inspect
key	Key value to look for in map

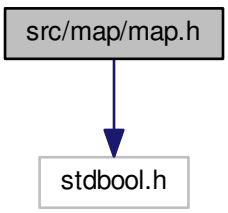
Returns

Value at key specified

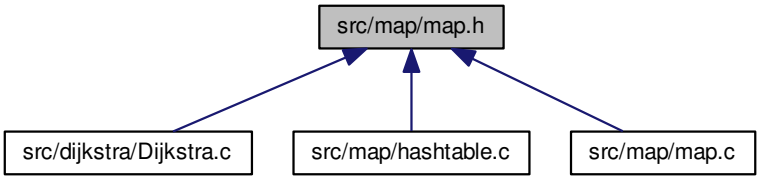
5.12 src/map/map.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for map.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct _map map

Functions

- map ∗ map_create (void)
  Creates an empty map structure.
- bool map_insert (map ∗m, const char ∗key, double value)
  Inserts new key and value into map.
- bool map_exists (map ∗m, const char ∗key)
  Checks if key exists in map.
- double map_lookup (map ∗m, const char ∗key)
  Returns Value of specified key.
- void map_destroy (map ∗m)
  Breaks down map and frees memory.

5.12.1 Typedef Documentation

5.12.1.1 typedef struct _map map

5.12.2 Function Documentation

5.12.2.1 map∗ map_create (void)

Creates an empty map structure.

Returns

Pointer to new map in memory

5.12.2.2 void map_destroy (map ∗ *m*)

Breaks down map and frees memory.

Parameters

m	Map to destroy

5.12.2.3 bool map_exists (map * *m*, const char * *key*)

Checks if key exists in map.

Parameters

m	Map to inspect
key	Key value to look for in map

Returns

True if found successfully, False if not found

5.12.2.4 bool map_insert (map * *m*, const char * *key*, double *value*)

Inserts new key and value into map.

Parameters

m	Map to insert into
key	Key value to add to map
value	Value for specified key

Returns

True if added successfully, False if failed

5.12.2.5 double map_lookup (map * *m*, const char * *key*)

Returns Value of specified key.

Parameters

m	Map to inspect
key	Key value to look for in map

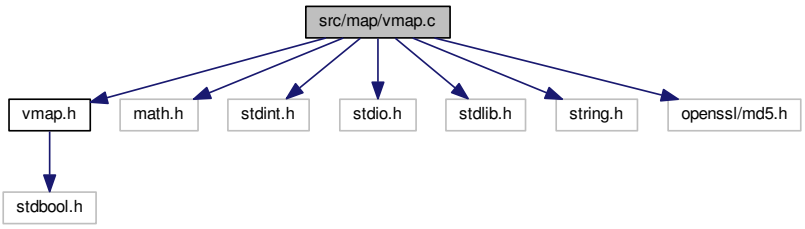
Returns

Value at key specified

5.13 src/map/vmap.c File Reference

```
#include "vmap.h"
#include <math.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/md5.h>
```

Include dependency graph for vmap.c:



Data Structures

- struct entry
- struct _vmap

Macros

- #define _XOPEN_SOURCE 500

Functions

- vmap * vmap_create (void)

  Creates an empty vmap structure.
- bool vmap_insert (vmap *m, const char *key, void *value)

  Inserts new key and value into vmap.
- bool vmap_exists (vmap *m, const char *key)

  Checks if key exists in vmap.
- void * vmap_lookup (vmap *m, const char *key)

  Returns Value of specified key.
- void vmap_destroy (vmap *m)

  Breaks down vmap and frees memory.

5.13.1 Macro Definition Documentation

5.13.1.1 #define _XOPEN_SOURCE 500

5.13.2 Function Documentation

5.13.2.1 vmap* vmap_create (void)

Creates an empty vmap structure.

Returns

Pointer to new vmap in memory

5.13.2.2 void vmap_destroy (vmap * m)

Breaks down vmap and frees memory.

Parameters

m	VMap to destroy

5.13.2.3 bool vmap_exists (vmap * m, const char * key)

Checks if key exists in vmap.

Parameters

m	VMap to inspect
key	Key value to look for in vmap

Returns

True if found successfully, False if not found

5.13.2.4 bool vmap_insert (vmap * m, const char * key, void * value)

Inserts new key and value into vmap.

Parameters

m	VMap to insert into
key	Key value to add to vmap
value	Value for specified key

Returns

True if added successfully, False if failed

5.13.2.5 void* vmap_lookup (vmap * *m*, const char * *key*)

Returns Value of specified key.

Parameters

m	VMap to inspect
key	Key value to look for in vmap

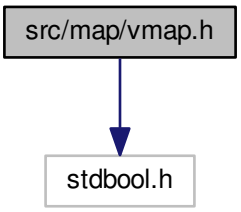
Returns

Value at key specified

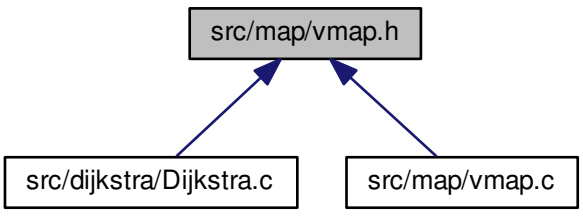
5.14 src/map/vmap.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for vmap.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef struct _vmap vmap

Functions

- vmap * vmap_create (void)

  Creates an empty vmap structure.
- bool vmap_insert (vmap *m, const char *key, void *value)

  Inserts new key and value into vmap.
- bool vmap_exists (vmap *m, const char *key)

  Checks if key exists in vmap.
- void * vmap_lookup (vmap *m, const char *key)

  Returns Value of specified key.
- void vmap_destroy (vmap *m)

  Breaks down vmap and frees memory.

5.14.1 Typedef Documentation

5.14.1.1 typedef struct _vmap vmap

5.14.2 Function Documentation

5.14.2.1 vmap* vmap_create (void)

Creates an empty vmap structure.

Returns

Pointer to new vmap in memory

5.14.2.2 void vmap_destroy (vmap * *m*)

Breaks down vmap and frees memory.

Parameters

m	VMap to destroy

5.14.2.3 bool vmap_exists (vmap * *m*, const char * *key*)

Checks if key exists in vmap.

Parameters

m	VMap to inspect
key	Key value to look for in vmap

Returns

True if found successfully, False if not found

5.14.2.4 bool vmap_insert (vmap * m, const char * key, void * value)

Inserts new key and value into vmap.

Parameters

m	VMap to insert into
key	Key value to add to vmap
value	Value for specified key

Returns

True if added successfully, False if failed

5.14.2.5 void* vmap_lookup (vmap * m, const char * key)

Returns Value of specified key.

Parameters

m	VMap to inspect
key	Key value to look for in vmap

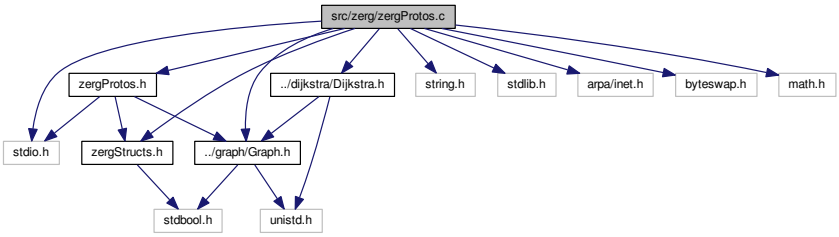
Returns

Value at key specified

5.15 src/zerg/zergProtos.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <byteswap.h>
#include <math.h>
#include "zergProtos.h"
#include "zergStructs.h"
#include "../graph/Graph.h"
#include "../dijkstra/Dijkstra.h"
```

Include dependency graph for zergProtos.c:



Macros

- #define _GNU_SOURCE

Functions

- void readZerg (FILE *source, FILE *dest)
- void checkEntry (char string[16], unsigned int input, zergPacket *packet)
- void pickPacketType (FILE *source, FILE *dest, zergPacket *packet)
- void writeMessage (FILE *source, FILE *dest)
- void writeStatus (FILE *source, FILE *dest)
- void writeCommand (FILE *source, FILE *dest)
- void writeGPS (FILE *source, FILE *dest)
- void writePcapHeader (FILE *dest)
- void writePcapPacket (FILE *dest, int zergLength)
- void writeEtherHeader (FILE *dest)
- void writeIpv4Header (FILE *dest, int zergLength)
- void writeUdpHeader (FILE *dest, int zergLength)
- void writeZergHeader (FILE *dest, zergPacket *packet)
- int rotate3ByteInt (int swap)
- int rotateBack (int swap)
- void fileCorruption (void)
- int validateHeader (zergPacket *packet)
- void parseCapture (FILE *psychicCapture, ZergUnit **unit, int *zergCount)
- void readPcapHeader (FILE *psychicCapture)
- void readPcapPacket (FILE *psychicCapture)
- void readEthernetPacket (FILE *psychicCapture)
- void hexToInt (unsigned int *myInt, unsigned char hex)
- void hexToShort (unsigned short *myShort, unsigned char hex)
- void hexToDouble (unsigned long long *myLong, unsigned char hex)
- void readIpv4Packet (FILE *psychicCapture, unsigned int *ipTotalLength)
- void readIpv6Packet (FILE *psychicCapture, unsigned int *ipTotalLength)
- void readUdpPacket (FILE *psychicCapture, unsigned int *udpTotalLength)
- void readZergPacket (FILE *psychicCapture, unsigned int *udpTotalLength, ZergUnit **unit, int *zergCount)
- void readMessage (FILE *psychicCapture, unsigned int payloadLength)
- void readStatus (FILE *psychicCapture, unsigned int payloadLength, ZergUnit *unit)
- void readCommand (FILE *psychicCapture)
- void readGPS (FILE *psychicCapture, ZergUnit *unit)
- ZergUnit * create_unit (void)

Creates and initializes a new ZergUnit.

- void print_zergUnit (ZergUnit *z)

  Prints id and GPS information for ZergUnit.

- double zergUnit_distance (ZergUnit *z1, ZergUnit *z2)

  Calculates the difference in location between two zerg units.

- char ** Zerg_twoPaths (Graph *zergGraph, ZergUnit **unitList, int *zergCount, int changeLimit)

  My attempt to find disjointed node paths between two nodes.

- void deleteRoute (ZergUnit **route, char *node, int *count)

  Used to delete and shift zergUnit list after deletion.

Variables

- int zergPayloadSize = 0
- int fscanNum = 0

5.15.1 Macro Definition Documentation

5.15.1.1 #define _GNU_SOURCE

5.15.2 Function Documentation

5.15.2.1 void checkEntry (char *string[16],* unsigned int *input,* **zergPacket** * *packet*)

5.15.2.2 **ZergUnit*** create_unit (void)

Creates and initializes a new ZergUnit.

Returns

Pointer to new ZergUnit

5.15.2.3 void deleteRoute (**ZergUnit** ** *route,* char * *node,* int * *count*)

Used to delete and shift zergUnit list after deletion.

Parameters

route	zergUnit list that had been traversed
node	Zerg id as a string to be removed
count	Pointer to count to keep track of number of zergUnits

5.15.2.4 void fileCorruption (void)

5.15.2.5 void hexToDouble (unsigned long long * *myLong,* unsigned char *hex*)

5.15.2.6 void hexToInt (unsigned int * *myInt,* unsigned char *hex*)

5.15.2.7 void hexToShort (unsigned short * *myShort,* unsigned char *hex*)

5.15.2.8 void parseCapture (FILE * *psychicCapture,* **ZergUnit** ** *unit,* int * *zergCount*)

5.15.2.9 void pickPacketType (FILE * *source,* FILE * *dest,* **zergPacket** * *packet*)

5.15.2.10 void print_zergUnit (**ZergUnit** * *z*)

Prints id and GPS information for ZergUnit.

5.15.2.11 void readCommand (FILE * *psychicCapture*)

5.15.2.12 void readEthernetPacket (FILE * *psychicCapture*)

5.15.2.13 void readGPS (FILE * *psychicCapture,* **ZergUnit** * *unit*)

5.15.2.14 void readIpv4Packet (FILE * *psychicCapture,* unsigned int * *ipTotalLength*)

5.15.2.15 void readIpv6Packet (FILE * *psychicCapture,* unsigned int * *ipTotalLength*)

5.15.2.16 void readMessage (FILE * *psychicCapture,* unsigned int *payloadLength*)

5.15.2.17 void readPcapHeader (FILE * *psychicCapture*)

5.15.2.18 void readPcapPacket (FILE * *psychicCapture*)

5.15.2.19 void readStatus (FILE * *psychicCapture,* unsigned int *payloadLength,* **ZergUnit** * *unit*)

5.15.2.20 void readUdpPacket (FILE * *psychicCapture,* unsigned int * *udpTotalLength*)

5.15.2.21 void readZerg (FILE * *source,* FILE * *dest*)

5.15.2.22 void readZergPacket (FILE * *psychicCapture,* unsigned int * *udpTotalLength,* **ZergUnit** ** *unit,* int * *zergCount*)

5.15.2.23 int rotate3ByteInt (int *swap*)

5.15.2.24 int rotateBack (int *swap*)

5.15.2.25 int validateHeader (**zergPacket** * *packet*)

5.15.2.26 void writeCommand (FILE * *source,* FILE * *dest*)

5.15.2.27 void writeEtherHeader (FILE * *dest*)

5.15.2.28 void writeGPS (FILE * *source,* FILE * *dest*)

5.15.2.29 void writeIpv4Header (FILE * *dest,* int *zergLength*)

5.15.2.30 void writeMessage (FILE * *source,* FILE * *dest*)

5.15.2.31 void writePcapHeader (FILE * *dest*)

5.15.2.32 void writePcapPacket (FILE * *dest,* int *zergLength*)

5.15.2.33 void writeStatus (FILE * *source,* FILE * *dest*)

5.15.2.34 void writeUdpHeader (FILE * *dest,* int *zergLength*)

5.15.2.35 void writeZergHeader (FILE * *dest,* zergPacket * *packet*)

5.15.2.36 char** Zerg_twoPaths (Graph * *zergGraph,* ZergUnit ** *unitList,* int * *zergCount,* int *changeLimit*)

My attempt to find disjointed node paths between two nodes.

Parameters

zergGraph	Graph of ZergUnits to check
unitList	Array of ZergUnit pointers
zergCount	Pointer to number of ZergUnits in unitList
changeLimit	Used to set how many Zergs can be deleted

Returns

List of node id's that were deleted

5.15.2.37 double zergUnit_distance (ZergUnit * *z1,* ZergUnit * *z2*)

Calculates the difference in location between two zerg units.

Parameters

z1	From Zerg
z2	To Zerg

Returns

Distance as Double

5.15.3 Variable Documentation

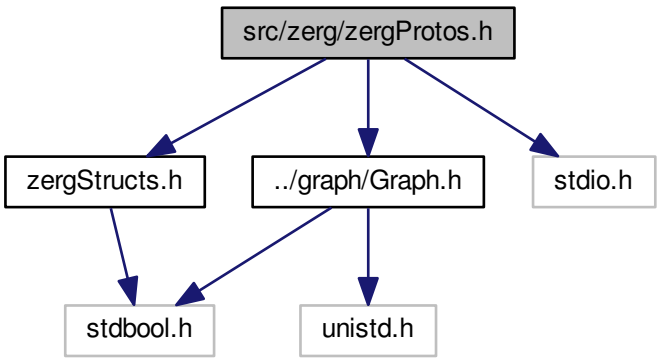
5.15.3.1 int fscanNum = 0

5.15.3.2 int zergPayloadSize = 0

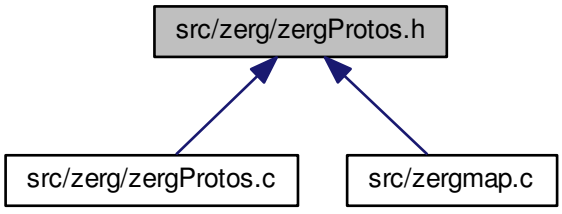
5.16 src/zerg/zergProtos.h File Reference

```
#include "zergStructs.h"
#include "../graph/Graph.h"
#include <stdio.h>
```

Include dependency graph for zergProtos.h:



This graph shows which files directly or indirectly include this file:



Functions

- void readZerg (FILE *source, FILE *dest)
- void checkEntry (char string[16], unsigned int input, zergPacket *packet)
- void pickPacketType (FILE *source, FILE *dest, zergPacket *packet)
- void writeMessage (FILE *source, FILE *dest)

- void writeStatus (FILE *source, FILE *dest)
- void writeCommand (FILE *source, FILE *dest)
- void writeGPS (FILE *source, FILE *dest)
- void writePcapHeader (FILE *dest)
- void writePcapPacket (FILE *dest, int zergLength)
- void writeEtherHeader (FILE *dest)
- void writeIpv4Header (FILE *dest, int zergLength)
- void writeUdpHeader (FILE *dest, int zergLength)
- void writeZergHeader (FILE *dest, zergPacket *packet)
- int rotate3ByteInt (int swap)
- int rotateBack (int swap)
- void fileCorruption (void)
- int validateHeader (zergPacket *packet)
- ZergUnit * create_unit (void)

  Creates and initializes a new ZergUnit.
- void parseCapture (FILE *psychicCapture, ZergUnit **unit, int *zergCount)
- void readPcapHeader (FILE *psychicCapture)
- void readPcapPacket (FILE *psychicCapture)
- void readEthernetPacket (FILE *psychicCapture)
- void readIpv4Packet (FILE *psychicCapture, unsigned int *ipTotalLength)
- void readIpv6Packet (FILE *psychicCaputre, unsigned int *ipTotalLength)
- void readUdpPacket (FILE *psychicCapture, unsigned int *udpTotalLength)
- void readZergPacket (FILE *psychicCapture, unsigned int *udpTotalLength, ZergUnit **unit, int *zergCount)
- void readMessage (FILE *psychicCapture, unsigned int payloadLength)
- void readStatus (FILE *psychicCapture, unsigned int payloadLength, ZergUnit *unit)
- void readCommand (FILE *psychicCapture)
- void readGPS (FILE *psychicCapture, ZergUnit *unit)
- void hexToInt (unsigned int *myInt, unsigned char hex)
- void hexToShort (unsigned short *myShort, unsigned char hex)
- void hexToDouble (unsigned long long *myLong, unsigned char hex)
- void decimalDegreesToDMS (double coordinate)
- double zergUnit_distance (ZergUnit *z1, ZergUnit *z2)

  Calculates the difference in location between two zerg units.
- void print_zergUnit (ZergUnit *z)

  Prints id and GPS information for ZergUnit.
- char ** Zerg_twoPaths (Graph *zergGraph, ZergUnit **unitList, int *zergCount, int changeLimit)

  My attempt to find disjointed node paths between two nodes.
- void deleteRoute (ZergUnit **route, char *node, int *count)

  Used to delete and shift zergUnit list after deletion.

5.16.1 Function Documentation

5.16.1.1 void checkEntry (char *string[16],* unsigned int *input,* zergPacket * *packet*)

5.16.1.2 ZergUnit* create_unit (void)

Creates and initializes a new ZergUnit.

Returns

Pointer to new ZergUnit

5.16.1.3 void decimalDegreesToDMS (double *coordinate*)

5.16.1.4 void deleteRoute (ZergUnit ** *route,* char * *node,* int * *count*)

Used to delete and shift zergUnit list after deletion.

Parameters

route	zergUnit list that had been traversed
node	Zerg id as a string to be removed
count	Pointer to count to keep track of number of zergUnits

5.16.1.5 void fileCorruption (void)

5.16.1.6 void hexToDouble (unsigned long long * *myLong*, unsigned char *hex*)

5.16.1.7 void hexToInt (unsigned int * *myInt*, unsigned char *hex*)

5.16.1.8 void hexToShort (unsigned short * *myShort*, unsigned char *hex*)

5.16.1.9 void parseCapture (FILE * *psychicCapture*, ZergUnit ** *unit*, int * *zergCount*)

5.16.1.10 void pickPacketType (FILE * *source*, FILE * *dest*, zergPacket * *packet*)

5.16.1.11 void print_zergUnit (ZergUnit * *z*)

Prints id and GPS information for ZergUnit.

5.16.1.12 void readCommand (FILE * *psychicCapture*)

5.16.1.13 void readEthernetPacket (FILE * *psychicCapture*)

5.16.1.14 void readGPS (FILE * *psychicCapture*, ZergUnit * *unit*)

5.16.1.15 void readIpv4Packet (FILE * *psychicCapture*, unsigned int * *ipTotalLength*)

5.16.1.16 void readIpv6Packet (FILE * *psychicCaputre*, unsigned int * *ipTotalLength*)

5.16.1.17 void readMessage (FILE * *psychicCapture*, unsigned int *payloadLength*)

5.16.1.18 void readPcapHeader (FILE * *psychicCapture*)

5.16.1.19 void readPcapPacket (FILE * *psychicCapture*)

5.16.1.20 void readStatus (FILE * *psychicCapture*, unsigned int *payloadLength*, ZergUnit * *unit*)

5.16.1.21 void readUdpPacket (FILE * *psychicCapture*, unsigned int * *udpTotalLength*)

5.16.1.22 void readZerg (FILE * *source*, FILE * *dest*)

5.16.1.23 void readZergPacket (FILE * *psychicCapture,* unsigned int * *udpTotalLength,* ZergUnit ** *unit,* int * *zergCount*)

5.16.1.24 int rotate3ByteInt (int *swap*)

5.16.1.25 int rotateBack (int *swap*)

5.16.1.26 int validateHeader (zergPacket * *packet*)

5.16.1.27 void writeCommand (FILE * *source,* FILE * *dest*)

5.16.1.28 void writeEtherHeader (FILE * *dest*)

5.16.1.29 void writeGPS (FILE * *source,* FILE * *dest*)

5.16.1.30 void writeIpv4Header (FILE * *dest,* int *zergLength*)

5.16.1.31 void writeMessage (FILE * *source,* FILE * *dest*)

5.16.1.32 void writePcapHeader (FILE * *dest*)

5.16.1.33 void writePcapPacket (FILE * *dest,* int *zergLength*)

5.16.1.34 void writeStatus (FILE * *source,* FILE * *dest*)

5.16.1.35 void writeUdpHeader (FILE * *dest,* int *zergLength*)

5.16.1.36 void writeZergHeader (FILE * *dest,* zergPacket * *packet*)

5.16.1.37 char** Zerg_twoPaths (Graph * *zergGraph,* ZergUnit ** *unitList,* int * *zergCount,* int *changeLimit*)

My attempt to find disjointed node paths between two nodes.

Parameters

zergGraph	Graph of ZergUnits to check
unitList	Array of ZergUnit pointers
zergCount	Pointer to number of ZergUnits in unitList
changeLimit	Used to set how many Zergs can be deleted

Returns

List of node id's that were deleted

5.16.1.38 double zergUnit_distance (ZergUnit * *z1,* ZergUnit * *z2*)

Calculates the difference in location between two zerg units.

Parameters

z1	From Zerg
z2	To Zerg

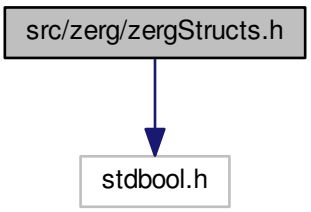
Returns

Distance as Double

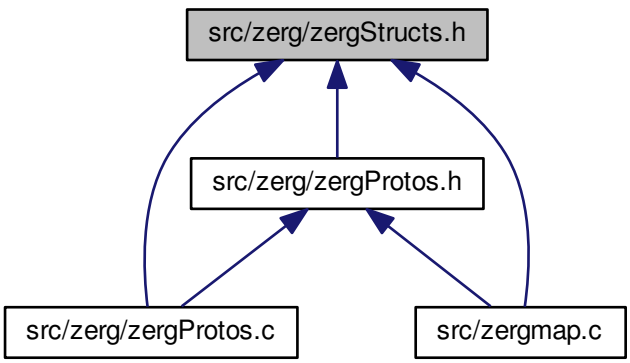
5.17 src/zerg/zergStructs.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for zergStructs.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct zergPacket
- struct pcapFileHeader
- struct pcapPacketHeader
- struct ethernetHeader
- struct ipv4Header
- struct ipv6Header
- struct udpHeader
- struct payload
- union payload::sSpeed
- struct cPayload
- union cPayload::param2
- struct gpsPayload
- union gpsPayload::longitude
- union gpsPayload::latitude
- union gpsPayload::altitude
- union gpsPayload::bearing
- union gpsPayload::speed
- union gpsPayload::accuracy
- struct ZergUnit

Typedefs

- typedef struct zergPacket zergPacket
- typedef struct pcapFileHeader pcapFileHeader
- typedef struct pcapPacketHeader pcapPacketHeader
- typedef struct ethernetHeader ethernetHeader
- typedef struct ipv4Header ipv4Header
- typedef struct ipv6Header ipv6Header
- typedef struct udpHeader udpHeader
- typedef struct payload payload
- typedef struct cPayload cPayload
- typedef struct gpsPayload gpsPayload
- typedef struct ZergUnit ZergUnit

5.17.1 Typedef Documentation

5.17.1.1 typedef struct cPayload cPayload

5.17.1.2 typedef struct ethernetHeader ethernetHeader

5.17.1.3 typedef struct gpsPayload gpsPayload

5.17.1.4 typedef struct ipv4Header ipv4Header

5.17.1.5 typedef struct ipv6Header ipv6Header

5.17.1.6 typedef struct payload payload

5.17.1.7 typedef struct pcapFileHeader pcapFileHeader

5.17.1.8 typedef struct pcapPacketHeader pcapPacketHeader

5.17.1.9 typedef struct udpHeader udpHeader

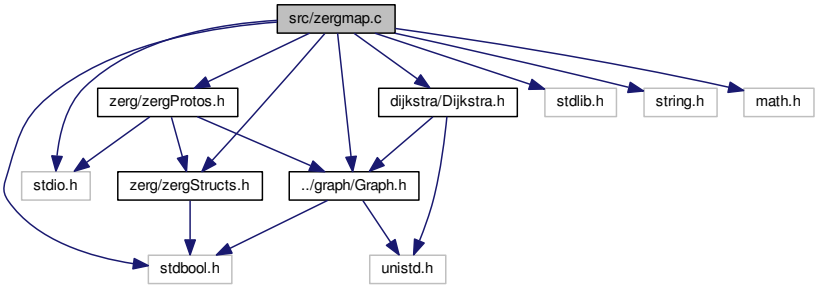
5.17.1.10 typedef struct zergPacket zergPacket

5.17.1.11 typedef struct ZergUnit ZergUnit

5.18 src/zergmap.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include "zerg/zergStructs.h"
#include "zerg/zergProtos.h"
#include "graph/Graph.h"
#include "dijkstra/Dijkstra.h"
```

Include dependency graph for zergmap.c:



Functions

- int main (int argc, char ∗argv[ ])

5.18.1 Function Documentation

5.18.1.1 int main (int *argc*, char ∗ *argv[ ]*)

Index