UTF8gbsn

Network Scanner

制作者 Doxygen 1.13.2

Chapter 1

网络扫描器（NetworkScanner）

一个基于 Qt 的多功能局域网扫描工具，提供 ** 网络拓扑可视化 **、** 设备分析 **、** 端口扫描 **、** 安全报告生成 ** 等能力，适用于网络运维、校园网管理与安全排查等场景。

1.1 主要功能

- **自动识别本地网络接口**
- ** 扫描局域网所有可达主机 **，识别：
  - IP 地址、主机名、MAC 地址
  - 可达状态、厂商信息、设备类型
- ** 网络拓扑可视化 **：自动绘制网络结构图
- ** 智能设备识别 **：分类展示路由器、服务器、PC、移动设备等
- ** 图表统计分析 **：
  - 设备类型分布（饼图）
  - 厂商分布（饼图）
  - 开放端口分布（柱状图）
- ** 端口扫描与风险评估 **：
  - 支持自定义端口列表
  - 高危端口识别与安全加固建议
- ** 扫描历史与会话比较 **：快速查看网络变化
- ** 计划任务扫描 **：支持定时执行、结果保存、通知提醒
- ** 明暗主题切换 **：支持夜间模式
- ** 多线程并发扫描 **：大幅提升扫描速度
- ** 结果导出与过滤 **：支持搜索、筛选与导出功能

1.2 最新更新（v2.3.0）

- 扫描引擎全面优化：采用QScopedPointer 管理资源，彻底解决内存泄漏
- UI 响应性大幅提升：扫描时页面切换更流畅，操作更顺畅
- 分批处理扫描任务：更智能地分配资源，避免系统过载
- 改进超时机制：自动识别并恢复卡住的扫描，避免程序无响应
- 优化进程管理：彻底解决扫描卡在 98 的问题
- 改进端口扫描算法：重点检测最常用端口，降低系统资源占用
- 增加事件处理：在关键操作中保持UI 响应性
- 降低资源占用：优化并发数量，减少系统负载
- 完善错误恢复：自动侦测和处理异常情况

1.3 构建要求

- Qt 6.0+（支持 QtCharts 模块）或 Qt 5 + 手动安装 QtCharts
- C++17 标准
- CMake 3.16+

1.4 构建步骤

克隆项目：

```
git clone git@github.com:Dictatora0/NetworkScanner.git
```

运行：

```
cd /Users/lifulin/Desktop/NetworkScanner
rm -rf build
mkdir build
cd build
cmake ..
make -j8
./NetScanner
```

Chapter 2

命名空间索引

2.1 命名空间列表

这里列出了所有命名空间定义，附带简要说明:

Chapter 3

继承关系索引

3.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

Chapter 4

类索引

4.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明:

Chapter 5

文件索引

5.1 文件列表

这里列出了所有文件，并附带简要说明:

Chapter 6

命名空间文档

6.1 QT_WARNING_DISABLE_DEPRECATED 命名空间参考

类

- struct qt_meta_tag_ZN10MainWindowE_t
- struct qt_meta_tag_ZN11ScanHistoryE_t
- struct qt_meta_tag_ZN14DeviceAnalyzerE_t
- struct qt_meta_tag_ZN14NetworkScannerE_t
- struct qt_meta_tag_ZN19NetworkTopologyViewE_t

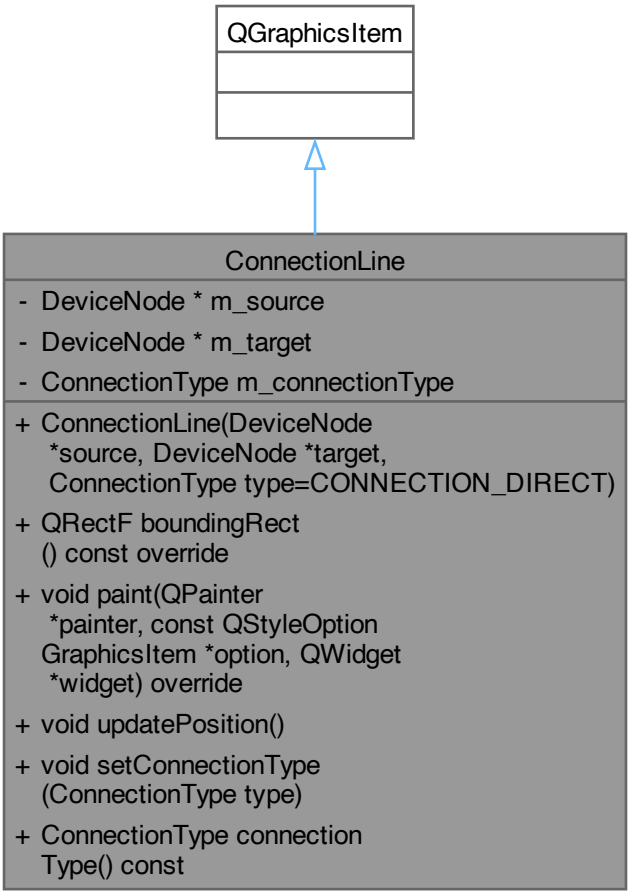
6.2 Ui 命名空间参考

Chapter 7

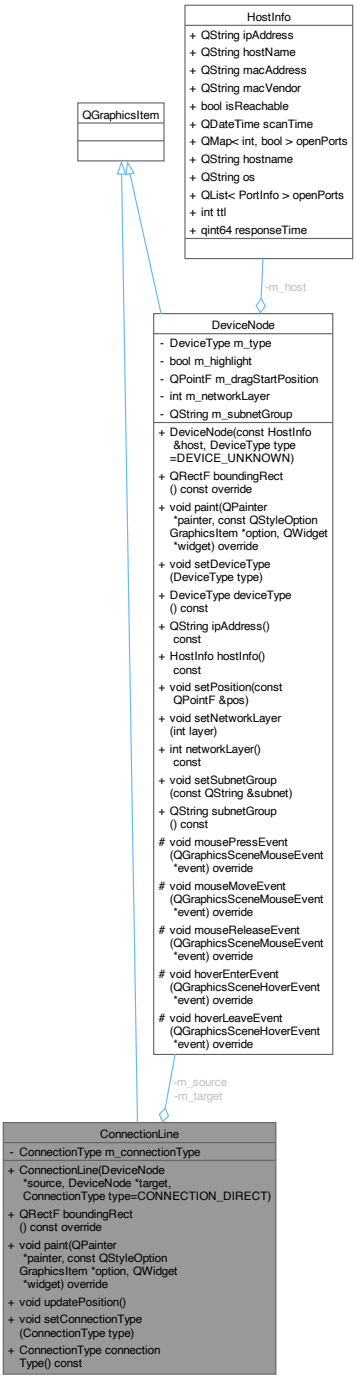
类说明

7.1 ConnectionLine 类参考

代表网络拓扑图中两个设备节点之间的连接线，继承自 QGraphicsItem。
#include <networktopology.h>
类 ConnectionLine 继承关系图:



ConnectionLine 的协作图:



Public 成员函数

- ConnectionLine (DeviceNode *source, DeviceNode *target, ConnectionType type=CONNECTION_DIRECT)

  ConnectionLine 构造函数。
- QRectF boundingRect () const override

  返回连接线的边界矩形。
- void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override

绘制连接线。
- void updatePosition ()
  更新连接线的位置 (当连接的节点移动时调用)。
- void setConnectionType (ConnectionType type)
  设置连接线的类型。
- ConnectionType connectionType () const
  获取连接线的类型。

Private 属性

- DeviceNode * m_source
  源设备节点
- DeviceNode * m_target
  目标设备节点
- ConnectionType m_connectionType
  连接类型

7.1.1 详细描述

代表网络拓扑图中两个设备节点之间的连接线，继承自 QGraphicsItem。
负责绘制连接线，并根据连接类型 (直接、无线、VPN、路由) 显示不同样式。

7.1.2 构造及析构函数说明

7.1.2.1 ConnectionLine()

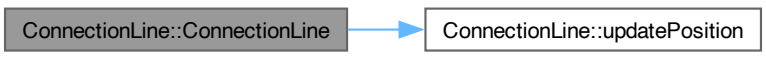
```
ConnectionLine::ConnectionLine (
            DeviceNode * source,
            DeviceNode * target,
            ConnectionType type = CONNECTION_DIRECT)
```

ConnectionLine 构造函数。

参数

source	源设备节点指针。
target	目标设备节点指针。
type	连接类型，默认为 CONNECTION_DIRECT。

函数调用图:



7.1.3 成员函数说明

7.1.3.1 boundingRect()

```
QRectF ConnectionLine::boundingRect () const    [override]
```

返回连接线的边界矩形。

返回

QRectF 边界矩形。

7.1.3.2 connectionType()

ConnectionType ConnectionLine::connectionType () const [inline]
获取连接线的类型。

返回

ConnectionType 连接类型。

7.1.3.3 paint()

void ConnectionLine::paint (
QPainter * painter,
const QStyleOptionGraphicsItem * option,
QWidget * widget) [override]
绘制连接线。

参数

painter	QPainter 指针。
option	QStyleOptionGraphicsItem 指针。
widget	QWidget 指针。

根据连接类型绘制不同样式的线条，并在线条末端绘制箭头。线条会连接到节点的边缘而不是中心。

7.1.3.4 setConnectionType()

void ConnectionLine::setConnectionType (
ConnectionType type)
设置连接线的类型。

参数

type	新的连接类型。

7.1.3.5 updatePosition()

void ConnectionLine::updatePosition ()
更新连接线的位置 (当连接的节点移动时调用)。
更新连接线的位置。
当连接的节点移动时，此方法被调用以更新连接线的几何形状。这是这个函数的调用关系图:

7.1.4 类成员变量说明

7.1.4.1 m_connectionType

ConnectionType ConnectionLine::m_connectionType [private]
连接类型

7.1.4.2 m_source

DeviceNode* ConnectionLine::m_source [private]
源设备节点

7.1.4.3 m_target

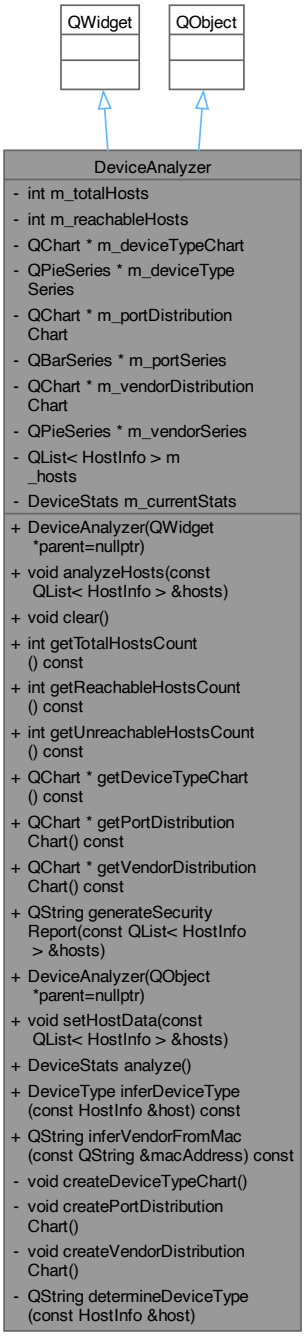
DeviceNode* ConnectionLine::m_target [private]
目标设备节点
该类的文档由以下文件生成:

- networktopology.h
- networktopology.cpp

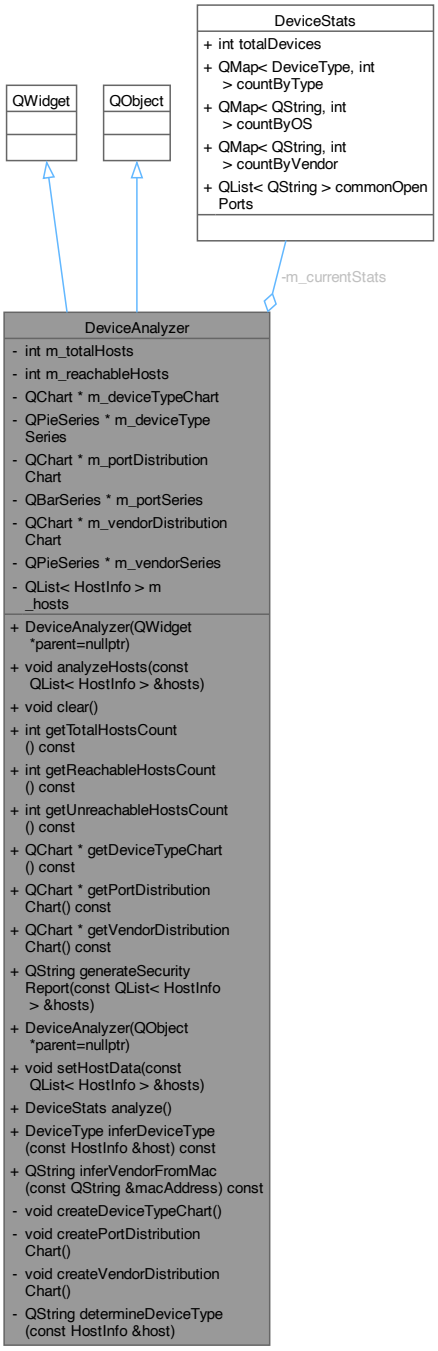
7.2 DeviceAnalyzer 类参考

设备分析器类 - 提供对扫描结果的统计分析
#include <deviceanalyzer.h>

类 DeviceAnalyzer 继承关系图:



DeviceAnalyzer 的协作图:



信号

- void analysisCompleted ()

  分析完成信号
- void analysisComplete (const DeviceStats &stats)

  当分析完成时发射此信号（如果分析是异步的）。

Public 成员函数

- DeviceAnalyzer (QWidget *parent=nullptr)
  DeviceAnalyzer 构造函数
- void analyzeHosts (const QList< HostInfo > &hosts)
  分析扫描结果并更新统计图表
- void clear ()
  清除分析结果和图表数据
- int getTotalHostsCount () const
  获取发现的总设备数
- int getReachableHostsCount () const
  获取可访问设备数
- int getUnreachableHostsCount () const
  获取不可访问设备数
- QChart * getDeviceTypeChart () const
  获取设备类型分布图表
- QChart * getPortDistributionChart () const
  获取端口分布图表
- QChart * getVendorDistributionChart () const
  获取设备厂商分布图表
- QString generateSecurityReport (const QList< HostInfo > &hosts)
  创建安全风险报告
- DeviceAnalyzer (QObject *parent=nullptr)
  DeviceAnalyzer 构造函数。
- void setHostData (const QList< HostInfo > &hosts)
  设置用于分析的主机数据。
- DeviceStats analyze ()
  执行分析操作。
- DeviceType inferDeviceType (const HostInfo &host) const
  根据主机信息推断设备类型。
- QString inferVendorFromMac (const QString &macAddress) const
  从MAC 地址推断设备制造商。

Private 成员函数

- void createDeviceTypeChart ()
  创建设备类型图表
- void createPortDistributionChart ()
  创建端口分布图表
- void createVendorDistributionChart ()
  创建设备厂商分布图表
- QString determineDeviceType (const HostInfo &host)
  根据主机信息判断设备类型

Private 属性

- int m_totalHosts
  发现总设备数
- int m_reachableHosts
  可访问设备数
- QChart * m_deviceTypeChart
  设备类型图表对象
- QPieSeries * m_deviceTypeSeries

设备类型饼图系列

- QChart * m_portDistributionChart
  端口分布图表对象
- QBarSeries * m_portSeries
  端口柱状图系列
- QChart * m_vendorDistributionChart
  设备厂商图表对象
- QPieSeries * m_vendorSeries
  设备厂商饼图系列
- QList< HostInfo > m_hosts
  当前用于分析的主机数据。
- DeviceStats m_currentStats
  当前的统计结果。

7.2.1 详细描述

设备分析器类 - 提供对扫描结果的统计分析
负责对扫描到的主机列表进行深入分析和统计。
此类接收一批 HostInfo 对象，并从中提取统计数据，例如设备类型分布、操作系统分布、常见开放端口等。

7.2.2 构造及析构函数说明

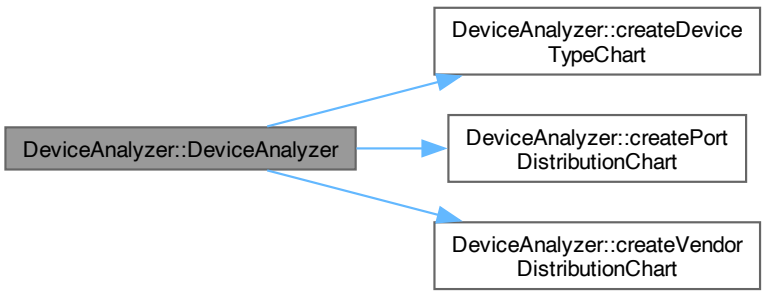
7.2.2.1 DeviceAnalyzer() [1/2]

DeviceAnalyzer::DeviceAnalyzer (
QWidget * parent = nullptr)
DeviceAnalyzer 构造函数

参数

parent	父控件指针

函数调用图:



7.2.2.2 DeviceAnalyzer() [2/2]

DeviceAnalyzer::DeviceAnalyzer (
QObject * parent = nullptr) [explicit]
DeviceAnalyzer 构造函数。

参数

parent	父QObject，默认为 nullptr。

7.2.3 成员函数说明

7.2.3.1 analysisComplete

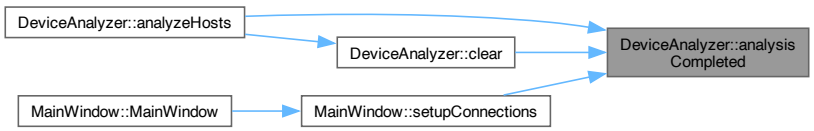
void DeviceAnalyzer::analysisComplete (
const DeviceStats & stats) [signal]
当分析完成时发射此信号（如果分析是异步的）。

参数

stats	分析得到的统计数据。

7.2.3.2 analysisCompleted

void DeviceAnalyzer::analysisCompleted () [signal]
分析完成信号
这是这个函数的调用关系图:



7.2.3.3 analyze()

DeviceStats DeviceAnalyzer::analyze ()
执行分析操作。
分析结果可以通过信号或直接返回一个包含统计数据的结构体。对于耗时较长的分析，可以考虑异步执行。

返回

DeviceStats 结构体，包含分析结果。

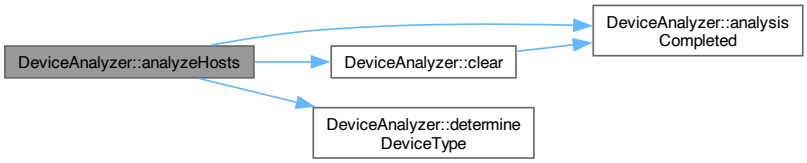
7.2.3.4 analyzeHosts()

void DeviceAnalyzer::analyzeHosts (
const QList< HostInfo > & hosts)
分析扫描结果并更新统计图表

参数

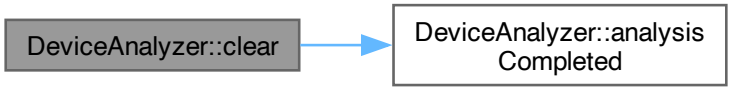
hosts	主机信息列表

函数调用图:

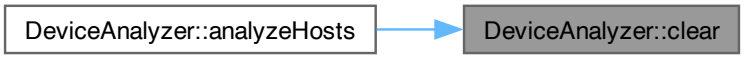


7.2.3.5 clear()

void DeviceAnalyzer::clear ()
清除分析结果和图表数据
函数调用图:



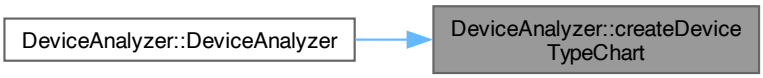
这是这个函数的调用关系图:



7.2.3.6 createDeviceTypeChart()

void DeviceAnalyzer::createDeviceTypeChart () [private]
创建设备类型图表

这是这个函数的调用关系图:

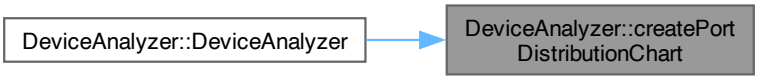


7.2.3.7 createPortDistributionChart()

void DeviceAnalyzer::createPortDistributionChart () [private]

创建端口分布图表

这是这个函数的调用关系图:

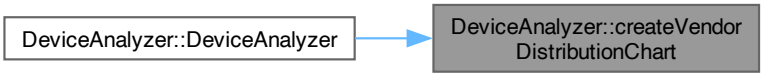


7.2.3.8 createVendorDistributionChart()

void DeviceAnalyzer::createVendorDistributionChart () [private]

创建设备厂商分布图表

这是这个函数的调用关系图:



7.2.3.9 determineDeviceType()

QString DeviceAnalyzer::determineDeviceType (
const HostInfo & host) [private]

根据主机信息判断设备类型

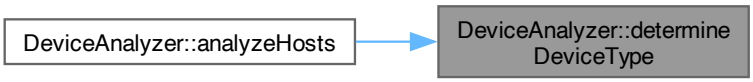
参数

host	主机信息

返回

设备类型字符串

这是这个函数的调用关系图:



7.2.3.10 generateSecurityReport()

QString DeviceAnalyzer::generateSecurityReport (
const QList< HostInfo > & hosts)

创建安全风险报告

参数

hosts	主机信息列表

返回

安全报告字符串

7.2.3.11 getDeviceTypeChart()

QChart * DeviceAnalyzer::getDeviceTypeChart () const [inline]

获取设备类型分布图表

返回

设备类型图表指针

7.2.3.12 getPortDistributionChart()

QChart * DeviceAnalyzer::getPortDistributionChart () const [inline]

获取端口分布图表

返回

端口分布图表指针

7.2.3.13 getReachableHostsCount()

int DeviceAnalyzer::getReachableHostsCount () const [inline]

获取可访问设备数

返回

可访问设备数

7.2.3.14 getTotalHostsCount()

int DeviceAnalyzer::getTotalHostsCount () const [inline]
获取发现的总设备数

返回

总设备数

7.2.3.15 getUnreachableHostsCount()

int DeviceAnalyzer::getUnreachableHostsCount () const [inline]
获取不可访问设备数

返回

不可访问设备数

7.2.3.16 getVendorDistributionChart()

QChart * DeviceAnalyzer::getVendorDistributionChart () const [inline]
获取设备厂商分布图表

返回

设备厂商图表指针

7.2.3.17 inferDeviceType()

DeviceType DeviceAnalyzer::inferDeviceType (
const HostInfo & host) const
根据主机信息推断设备类型。

参数

host	单个主机的信息。

返回

推断出的 DeviceType。

注解

这个方法也可以是 TopologyAnalyzer 的一部分，或者是一个共享的工具函数。如果在这里实现，它可能基于开放端口、OS 信息等进行推断。

7.2.3.18 inferVendorFromMac()

QString DeviceAnalyzer::inferVendorFromMac (
const QString & macAddress) const
从MAC 地址推断设备制造商。

参数

macAddress	主机的MAC 地址。

返回

设备制造商的名称，如果无法确定则返回空字符串或"Unknown"。

注解

此功能通常需要一个MAC 地址到制造商的数据库（OUI 数据库）。

7.2.3.19 setHostData()

void DeviceAnalyzer::setHostData (
const QList< HostInfo > & hosts)
设置用于分析的主机数据。

参数

hosts	从网络扫描获取的主机信息列表。

7.2.4 类成员变量说明

7.2.4.1 m_currentStats

DeviceStats DeviceAnalyzer::m_currentStats [private]
当前的统计结果。

7.2.4.2 m_deviceTypeChart

QChart* DeviceAnalyzer::m_deviceTypeChart [private]
设备类型图表对象

7.2.4.3 m_deviceTypeSeries

QPieSeries* DeviceAnalyzer::m_deviceTypeSeries [private]
设备类型饼图系列

7.2.4.4 m_hosts

QList<HostInfo> DeviceAnalyzer::m_hosts [private]
当前用于分析的主机数据。

7.2.4.5 m_portDistributionChart

QChart* DeviceAnalyzer::m_portDistributionChart [private]
端口分布图表对象

7.2.4.6 m_portSeries

QBarSeries* DeviceAnalyzer::m_portSeries [private]
端口柱状图系列

7.2.4.7 m_reachableHosts

int DeviceAnalyzer::m_reachableHosts [private]
可访问设备数

7.2.4.8 m_totalHosts

int DeviceAnalyzer::m_totalHosts [private]
发现总设备数

7.2.4.9 m_vendorDistributionChart

QChart* DeviceAnalyzer::m_vendorDistributionChart [private]
设备厂商图表对象

7.2.4.10 m_vendorSeries

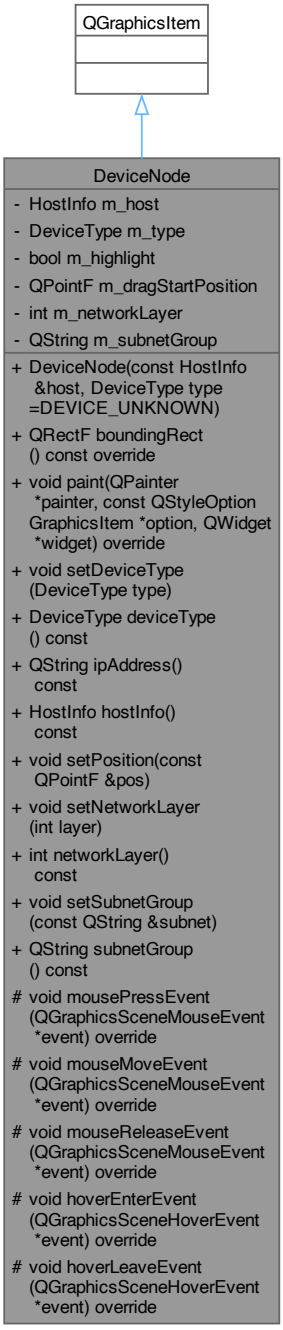
QPieSeries* DeviceAnalyzer::m_vendorSeries [private]
设备厂商饼图系列
该类的文档由以下文件生成:

- deviceanalyzer.h
- src/core/deviceanalyzer.h
- deviceanalyzer.cpp
- NetScanner_autogen/EWIEGA46WW/moc_deviceanalyzer.cpp
- NetScanner_autogen/JRIAJ772TK/moc_deviceanalyzer.cpp

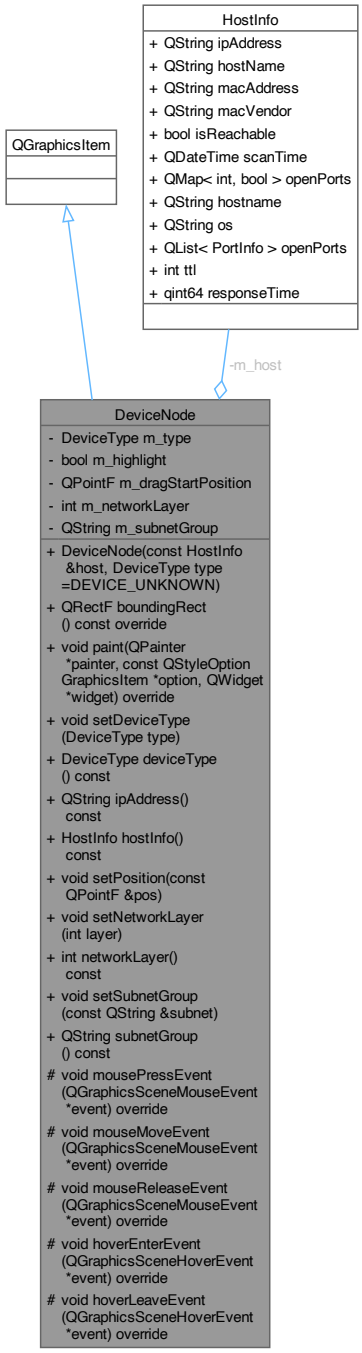
7.3 DeviceNode 类参考

代表网络拓扑图中的一个设备节点，继承自 QGraphicsItem。
#include <networktopology.h>

类 DeviceNode 继承关系图:



DeviceNode 的协作图:



Public 成员函数

- DeviceNode (const HostInfo &host, DeviceType type=DEVICE_UNKNOWN)

  DeviceNode 构造函数。
- QRectF boundingRect () const override

  返回节点的边界矩形。
- void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override

绘制设备节点。
- void setDeviceType (DeviceType type)
  设置节点的设备类型。
- DeviceType deviceType () const
  获取节点的设备类型。
- QString ipAddress () const
  获取节点的IP 地址。
- HostInfo hostInfo () const
  获取节点关联的完整主机信息。
- void setPosition (const QPointF &pos)
  设置节点的位置 (已废弃，QGraphicsItem 自带 setPos)。
- void setNetworkLayer (int layer)
  设置节点的网络层级 (例如，基于TTL)。
- int networkLayer () const
  获取节点的网络层级。
- void setSubnetGroup (const QString &subnet)
  设置节点所属的子网组。
- QString subnetGroup () const
  获取节点所属的子网组。

Protected **成员函数**

- void mousePressEvent (QGraphicsSceneMouseEvent *event) override
  处理鼠标按下事件。
- void mouseMoveEvent (QGraphicsSceneMouseEvent *event) override
  处理鼠标移动事件。
- void mouseReleaseEvent (QGraphicsSceneMouseEvent *event) override
  处理鼠标释放事件。
- void hoverEnterEvent (QGraphicsSceneHoverEvent *event) override
  处理鼠标悬停进入事件。
- void hoverLeaveEvent (QGraphicsSceneHoverEvent *event) override
  处理鼠标悬停离开事件。

Private **属性**

- HostInfo m_host
  节点关联的主机信息
- DeviceType m_type
  节点的设备类型
- bool m_highlight
  节点是否高亮 (鼠标悬停)
- QPointF m_dragStartPosition
  拖动开始时的位置
- int m_networkLayer
  节点的网络层级 (用于布局)
- QString m_subnetGroup
  节点所属的子网组 (用于布局)

7.3.1 详细描述

代表网络拓扑图中的一个设备节点，继承自 QGraphicsItem。
负责绘制设备图标、显示IP 地址，并处理用户交互 (如拖动、悬停提示)。

7.3.2 构造及析构函数说明

7.3.2.1 DeviceNode()

DeviceNode::DeviceNode (
const HostInfo & host,
DeviceType type = DEVICE_UNKNOWN)

DeviceNode 构造函数。

参数

host	该节点代表的主机信息。
type	该节点的设备类型，默认为 DEVICE_UNKNOWN。

7.3.3 成员函数说明

7.3.3.1 boundingRect()

QRectF DeviceNode::boundingRect () const [override]

返回节点的边界矩形。

返回

QRectF 边界矩形。

7.3.3.2 deviceType()

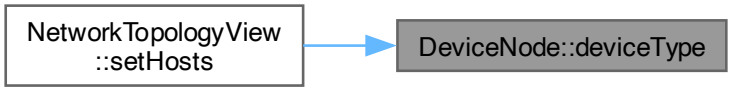
DeviceType DeviceNode::deviceType () const [inline]

获取节点的设备类型。

返回

DeviceType 设备类型。

这是这个函数的调用关系图:



7.3.3.3 hostInfo()

HostInfo DeviceNode::hostInfo () const [inline]

获取节点关联的完整主机信息。

返回

HostInfo 主机信息结构体。

7.3.3.4 hoverEnterEvent()

void DeviceNode::hoverEnterEvent (
QGraphicsSceneHoverEvent * event) [override], [protected]

处理鼠标悬停进入事件。

参数

event	悬停事件指针。

高亮节点并显示包含设备信息的工具提示。

7.3.3.5 hoverLeaveEvent()

void DeviceNode::hoverLeaveEvent (
QGraphicsSceneHoverEvent * event) [override], [protected]
处理鼠标悬停离开事件。

参数

event	悬停事件指针。

取消节点高亮。

7.3.3.6 ipAddress()

QString DeviceNode::ipAddress () const [inline]
获取节点的IP 地址。

返回

QString IP 地址字符串。

7.3.3.7 mouseMoveEvent()

void DeviceNode::mouseMoveEvent (
QGraphicsSceneMouseEvent * event) [override], [protected]
处理鼠标移动事件。

参数

event	鼠标事件指针。

更新场景以重绘连接线。

7.3.3.8 mousePressEvent()

void DeviceNode::mousePressEvent (
QGraphicsSceneMouseEvent * event) [override], [protected]
处理鼠标按下事件。

参数

event	鼠标事件指针。

7.3.3.9 mouseReleaseEvent()

void DeviceNode::mouseReleaseEvent (
QGraphicsSceneMouseEvent * event) [override], [protected]
处理鼠标释放事件。

参数

event	鼠标事件指针。

7.3.3.10 networkLayer()

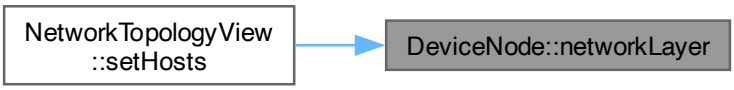
int DeviceNode::networkLayer () const [inline]

获取节点的网络层级。

返回

int 网络层级。

这是这个函数的调用关系图:



7.3.3.11 paint()

void DeviceNode::paint (
QPainter * painter,
const QStyleOptionGraphicsItem * option,
QWidget * widget) [override]

绘制设备节点。

参数

painter	QPainter 指针。
option	QStyleOptionGraphicsItem 指针。
widget	QWidget 指针 (关联的视图)。

根据设备类型绘制不同颜色和图标的圆形节点，并显示IP 地址。选中或高亮时颜色会变化。

7.3.3.12 setDeviceType()

void DeviceNode::setDeviceType (
DeviceType type)

设置节点的设备类型。

参数

type	新的设备类型。

7.3.3.13 setNetworkLayer()

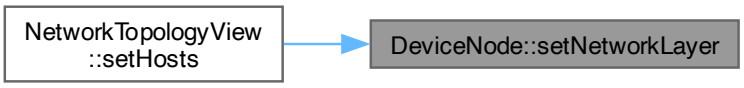
void DeviceNode::setNetworkLayer (
int layer) [inline]

设置节点的网络层级 (例如，基于TTL)。

参数

layer	网络层级整数。

这是这个函数的调用关系图:



7.3.3.14 setPosition()

void DeviceNode::setPosition (
const QPointF & pos)
设置节点的位置 (已废弃，QGraphicsItem 自带 setPos)。

参数

pos	新的位置坐标。

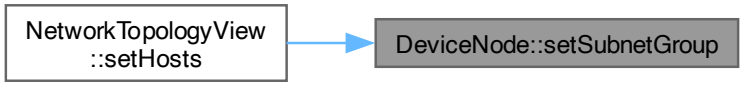
7.3.3.15 setSubnetGroup()

void DeviceNode::setSubnetGroup (
const QString & subnet) [inline]
设置节点所属的子网组。

参数

subnet	子网地址字符串。

这是这个函数的调用关系图:



7.3.3.16 subnetGroup()

QString DeviceNode::subnetGroup () const [inline]
获取节点所属的子网组。

返回

QString 子网地址字符串。

7.3.4 类成员变量说明

7.3.4.1 m_dragStartPosition

QPointF DeviceNode::m_dragStartPosition [private]
拖动开始时的位置

7.3.4.2 m_highlight

bool DeviceNode::m_highlight [private]
节点是否高亮 (鼠标悬停)

7.3.4.3 m_host

HostInfo DeviceNode::m_host [private]
节点关联的主机信息

7.3.4.4 m_networkLayer

int DeviceNode::m_networkLayer [private]
节点的网络层级 (用于布局)

7.3.4.5 m_subnetGroup

QString DeviceNode::m_subnetGroup [private]
节点所属的子网组 (用于布局)

7.3.4.6 m_type

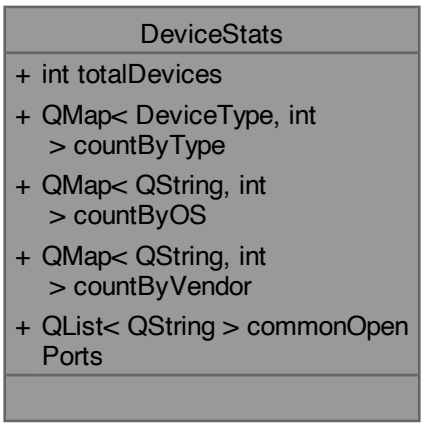
DeviceType DeviceNode::m_type [private]
节点的设备类型
该类的文档由以下文件生成:

- networktopology.h
- networktopology.cpp

7.4 DeviceStats 结构体参考

存储设备统计信息。
#include <deviceanalyzer.h>

DeviceStats 的协作图:



Public 属性

- int totalDevices
  设备总数。
- QMap< DeviceType, int > countByType
  按设备类型统计的数量。
- QMap< QString, int > countByOS
  按操作系统统计的数量。
- QMap< QString, int > countByVendor
  按设备制造商统计的数量 (基于MAC 地址)。
- QList< QString > commonOpenPorts
  常见的开放端口列表。

7.4.1 详细描述

存储设备统计信息。

7.4.2 类成员变量说明

7.4.2.1 commonOpenPorts

QList<QString> DeviceStats::commonOpenPorts
常见的开放端口列表。

7.4.2.2 countByOS

QMap<QString, int> DeviceStats::countByOS
按操作系统统计的数量。

7.4.2.3 countByType

QMap<DeviceType, int> DeviceStats::countByType
按设备类型统计的数量。

7.4.2.4 countByVendor

QMap<QString, int> DeviceStats::countByVendor
按设备制造商统计的数量 (基于MAC 地址)。

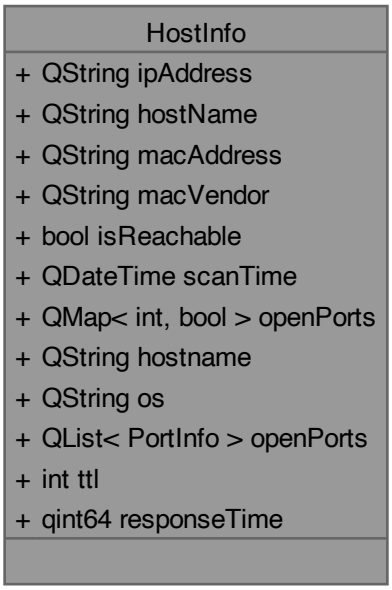
7.4.2.5 totalDevices

int DeviceStats::totalDevices
设备总数。
该结构体的文档由以下文件生成:

- src/core/deviceanalyzer.h

7.5 HostInfo 结构体参考

存储主机信息的结构体
#include <networkscanner.h>
HostInfo 的协作图:



Public 属性

- QString ipAddress
  主机IP 地址
- QString hostName
  主机名称
- QString macAddress
  MAC 物理地址
- QString macVendor
  MAC 地址对应的厂商
- bool isReachable
  主机是否可达

- QDateTime scanTime
  扫描时间
- QMap< int, bool > openPorts
  开放的端口及状态 (端口号 -> 是否开放)
- QString hostname
  主机的主机名 (如果可解析)。
- QString os
  推测的主机操作系统。
- QList< PortInfo > openPorts
  主机上开放的端口列表。
- int ttl
  从扫描器到主机的初始TTL 值。
- qint64 responseTime
  主机的响应时间 (例如 ping 延迟，单位毫秒)。

7.5.1 详细描述

存储主机信息的结构体
存储扫描到的单个主机的详细信息。
包含IP 地址、主机名、MAC 地址、厂商信息等扫描结果

7.5.2 类成员变量说明

7.5.2.1 hostName

QString HostInfo::hostName
主机名称

7.5.2.2 hostname

QString HostInfo::hostname
主机的主机名 (如果可解析)。

7.5.2.3 ipAddress

QString HostInfo::ipAddress
主机IP 地址
主机的IP 地址。

7.5.2.4 isReachable

bool HostInfo::isReachable
主机是否可达

7.5.2.5 macAddress

QString HostInfo::macAddress
MAC 物理地址
主机的MAC 地址 (如果可获取)。

7.5.2.6 macVendor

QString HostInfo::macVendor
MAC 地址对应的厂商

7.5.2.7 openPorts [1/2]

QMap<int, bool> HostInfo::openPorts
开放的端口及状态 (端口号 -> 是否开放)

7.5.2.8 openPorts [2/2]

QList<PortInfo> HostInfo::openPorts
主机上开放的端口列表。

7.5.2.9 os

QString HostInfo::os
推测的主机操作系统。

7.5.2.10 responseTime

qint64 HostInfo::responseTime
主机的响应时间 (例如 ping 延迟，单位毫秒)。

7.5.2.11 scanTime

QDateTime HostInfo::scanTime
扫描时间

7.5.2.12 ttl

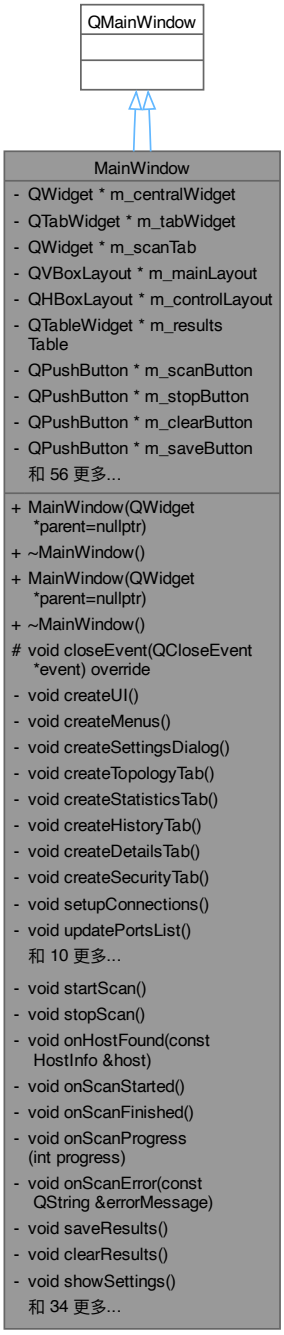
int HostInfo::ttl
从扫描器到主机的初始TTL 值。
该结构体的文档由以下文件生成:

- networkscanner.h
- src/data/hostinfo.h

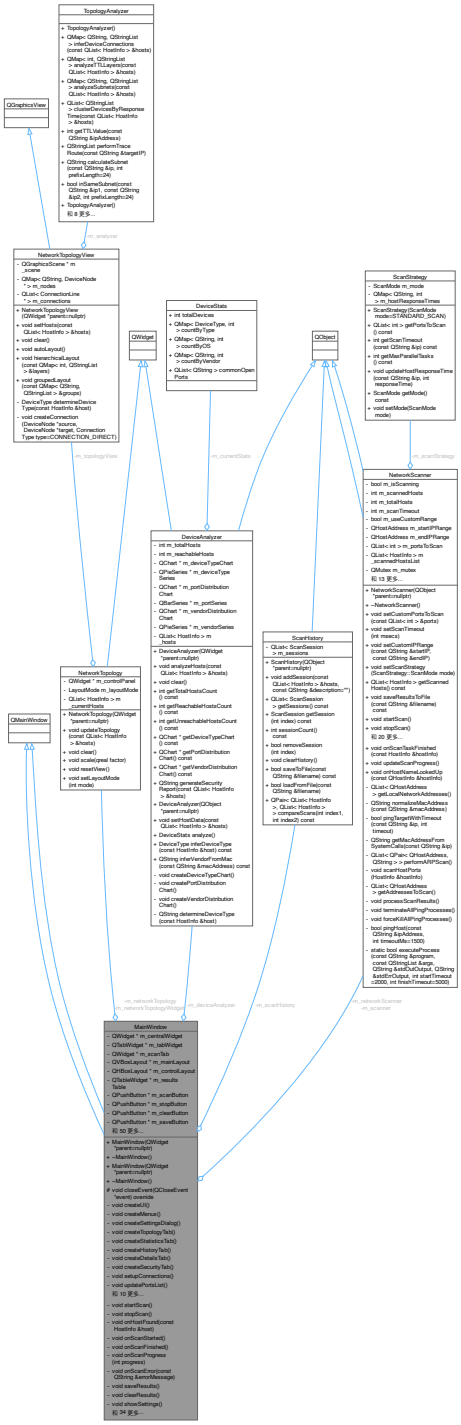
7.6 MainWindow 类参考

主窗口类，负责程序的主要界面和交互逻辑。
#include <mainwindow.h>

类 MainWindow 继承关系图:



MainWindow 的协作图:



Public 成员函数

- MainWindow (QWidget *parent=nullptr)

  MainWindow 构造函数

- ~MainWindow ()

  MainWindow 析构函数

- MainWindow (QWidget *parent=nullptr)

  MainWindow 构造函数。

- ~MainWindow ()
  MainWindow 析构函数。

Protected 成员函数

- void closeEvent (QCloseEvent *event) override
  关闭事件处理程序。

Private 槽

- void startScan ()
  开始扫描
- void stopScan ()
  停止扫描
- void onHostFound (const HostInfo &host)
  当发现一个主机时调用
- void onScanStarted ()
  当扫描开始时调用
- void onScanFinished ()
  当扫描结束时调用
- void onScanProgress (int progress)
  当扫描进度更新时调用
- void onScanError (const QString &errorMessage)
  当扫描发生错误时调用
- void saveResults ()
  保存扫描结果
- void clearResults ()
  清除扫描结果
- void showSettings ()
  显示设置界面
- void applySettings ()
  应用设置
- void showAbout ()
  显示关于对话框
- void showHostDetails (int row, int column)
  显示主机详细信息
- void exportToCSV ()
  导出结果为CSV 文件
- void togglePortScanOptions (bool checked)
  切换自定义端口扫描选项的可用状态
- void toggleRangeOptions (bool checked)
  切换自定义IP 范围选项的可用状态
- void showTopologyView ()
  显示网络拓扑视图
- void showStatisticsView ()
  显示统计分析视图
- void showHistoryView ()
  显示扫描历史视图
- void generateSecurityReport ()
  生成安全报告
- void saveTopologyImage ()
  保存网络拓扑图为图片

- void toggleDarkMode (bool enable)
  切换暗色模式
- void compareScanResults ()
  比较扫描结果
- void scheduleScan ()
  计划扫描
- void saveHistoryToFile ()
  保存扫描历史到文件
- void loadHistoryFromFile ()
  从文件加载扫描历史
- void updateNetworkTopology ()
  更新网络拓扑图
- void refreshTopology ()
  刷新网络拓扑图
- void filterResults ()
  过滤扫描结果
- void clearFilters ()
  清除过滤器
- void onThemeChanged ()
  当主题改变时调用
- void on_actionStartScan_triggered ()
- void on_actionStopScan_triggered ()
- void on_actionConfigureScan_triggered ()
- void on_actionViewTopology_triggered ()
- void on_actionViewAnalysis_triggered ()
- void on_actionViewHistory_triggered ()
- void on_actionExit_triggered ()
- void on_actionAbout_triggered ()
- void handleHostFound (const HostInfo &hostInfo)
- void handleScanProgress (int percentage, const QString &message)
- void handleScanFinished (const QList< HostInfo > &results)
- void handleScanError (const QString &errorMessage)
- void handleDeviceSelectedFromTopology (const HostInfo &hostInfo)

Private **成员函数**

- void createUI ()
  创建用户界面
- void createMenus ()
  创建菜单
- void createSettingsDialog ()
  创建设置对话框
- void createTopologyTab ()
  创建网络拓扑标签页
- void createStatisticsTab ()
  创建统计分析标签页
- void createHistoryTab ()
  创建扫描历史标签页
- void createDetailsTab ()
  创建主机详情标签页
- void createSecurityTab ()
  创建安全相关标签页 (如果需要)

- void setupConnections ()
  设置信号和槽的连接
- void updatePortsList ()
  更新端口列表 (如果需要)
- void loadSettings ()
  加载程序设置
- void saveSettings ()
  保存程序设置
- void updateStatistics ()
  更新统计数据
- void applyTheme (bool darkMode)
  应用主题（暗色/浅色）
- void createMenus ()
  创建菜单栏。
- void createToolbars ()
  创建工具栏。
- void createStatusbar ()
  创建状态栏。
- void setupDockWidgets ()
  创建并设置中央区域的停靠窗口或堆叠窗口。
- void readSettings ()
  读取应用程序设置。
- void writeSettings ()
  保存应用程序设置。

Private 属性

- QWidget * m_centralWidget
  中央控件
- QTabWidget * m_tabWidget
  标签页控件
- QWidget * m_scanTab
  扫描结果标签页
- QVBoxLayout * m_mainLayout
  主布局（扫描结果页）
- QHBoxLayout * m_controlLayout
  控制按钮布局
- QTableWidget * m_resultsTable
  扫描结果表格
- QPushButton * m_scanButton
  开始扫描按钮
- QPushButton * m_stopButton
  停止扫描按钮
- QPushButton * m_clearButton
  清除结果按钮
- QPushButton * m_saveButton
  保存结果按钮
- QProgressBar * m_progressBar
  扫描进度条
- QLabel * m_statusLabel
  状态标签

- QStatusBar * m_statusBar
  状态栏
- QWidget * m_settingsTab
  扫描设置标签页
- QVBoxLayout * m_settingsLayout
  设置标签页布局
- QGroupBox * m_portsGroupBox
  端口设置组
- QCheckBox * m_customPortsCheckBox
  自定义端口复选框
- QLineEdit * m_portsLineEdit
  端口输入框
- QSpinBox * m_timeoutSpinBox
  超时时间设置框
- QGroupBox * m_rangeGroupBox
  IP 范围设置组
- QCheckBox * m_customRangeCheckBox
  自定义IP 范围复选框
- QLineEdit * m_startIPLineEdit
  起始IP 输入框
- QLineEdit * m_endIPLineEdit
  结束IP 输入框
- QWidget * m_detailsTab
  主机详情标签页
- QVBoxLayout * m_detailsLayout
  主机详情页布局
- QTextEdit * m_detailsTextEdit
  主机详情文本框
- QWidget * m_topologyTab
  网络拓扑标签页
- NetworkTopology * m_networkTopology
  网络拓扑控件
- QWidget * m_statisticsTab
  统计分析标签页
- DeviceAnalyzer * m_deviceAnalyzer
  设备分析器
- QChartView * m_deviceTypeChartView
  设备类型图表视图
- QChartView * m_vendorChartView
  厂商分布图表视图
- QChartView * m_portDistributionChartView
  端口分布图表视图
- QTextEdit * m_securityReportText
  安全报告文本框
- QWidget * m_historyTab
  扫描历史标签页
- ScanHistory * m_scanHistory
  扫描历史管理器
- QComboBox * m_sessionComboBox
  扫描会话选择框
- QTableWidget * m_historyTable

扫描历史表格
- QMenu * m_fileMenu
  文件菜单
- QMenu * m_viewMenu
  视图菜单
- QMenu * m_toolsMenu
  工具菜单
- QMenu * m_helpMenu
  帮助菜单
- QAction * m_exportAction
  导出结果动作
- QAction * m_saveHistoryAction
  保存扫描历史动作
- QAction * m_loadHistoryAction
  加载扫描历史动作
- QAction * m_saveTopologyAction
  保存网络拓扑图动作
- QAction * m_exitAction
  退出动作
- QAction * m_settingsAction
  设置动作
- QAction * m_darkModeAction
  暗色模式动作
- QAction * m_scheduleScanAction
  计划扫描动作
- QAction * m_aboutAction
  关于动作
- QWidget * m_filterWidget
  过滤控件容器
- QLineEdit * m_filterIPLineEdit
  IP 过滤输入框
- QComboBox * m_filterVendorComboBox
  厂商过滤选择框
- QComboBox * m_filterTypeComboBox
  设备类型过滤选择框
- QPushButton * m_filterButton
  应用过滤按钮
- QPushButton * m_clearFilterButton
  清除过滤按钮
- NetworkScanner * m_scanner
  网络扫描器实例
- int m_hostsFound
  已发现的主机数量
- int m_currentHostIndex
  当前结果表格中选中的主机索引 (可能已废弃)
- bool m_darkModeEnabled
  是否启用暗色模式
- Ui::MainWindow * ui
  Qt Designer 生成的UI 类实例 (如果使用.ui 文件)。
- NetworkScanner * m_networkScanner
  网络扫描器实例。

- NetworkTopology * m_networkTopologyWidget
  网络拓扑图显示组件。
- QList< HostInfo > m_currentScanResults
  当前扫描操作的结果缓存。
- QString m_currentScanTarget
  当前扫描目标。

7.6.1 详细描述

主窗口类，负责程序的主要界面和交互逻辑。
应用程序的主窗口，集成各个功能模块。
MainWindow 负责组织用户界面，包括启动扫描、显示拓扑图、查看设备分析结果和管理扫描历史等。

7.6.2 构造及析构函数说明

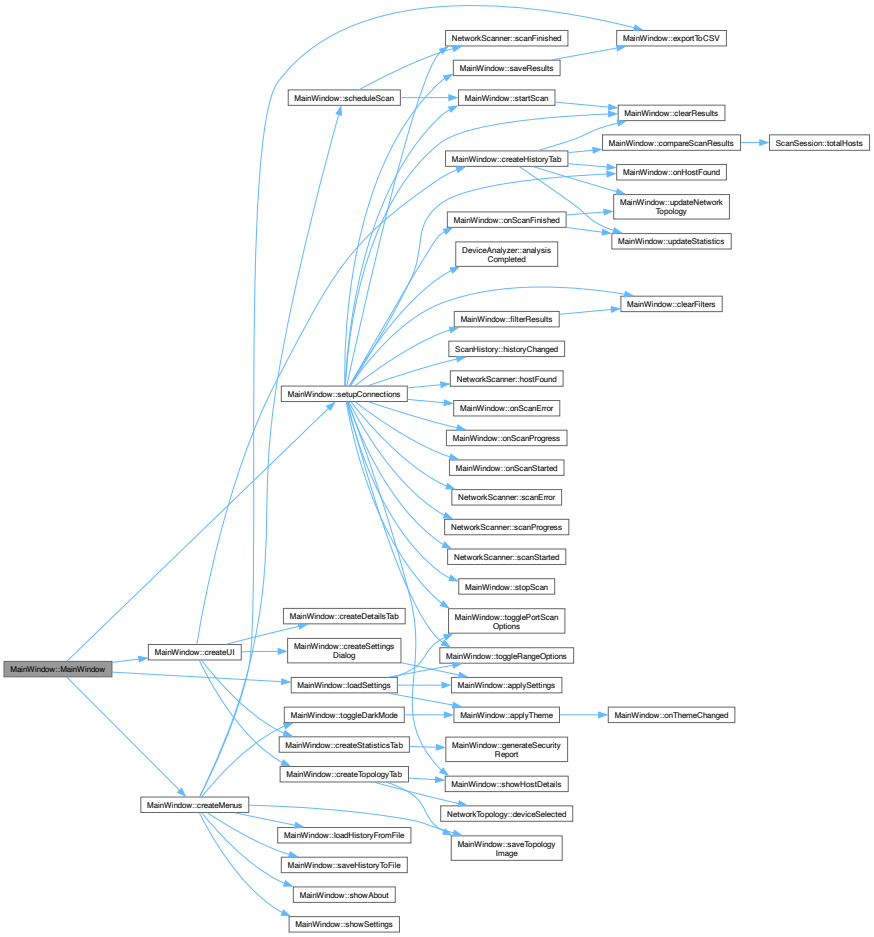
7.6.2.1 MainWindow() [1/2]

MainWindow::MainWindow (
QWidget * parent = nullptr)
MainWindow 构造函数

参数

parent	父控件指针

函数调用图:



7.6.2.2 ~MainWindow() [1/2]

MainWindow::~MainWindow ()
MainWindow 析构函数
函数调用图:

7.6.2.3 MainWindow() [2/2]

MainWindow::MainWindow (
QWidget * parent = nullptr)
MainWindow 构造函数。

参数

parent	父QWidget，默认为 nullptr。

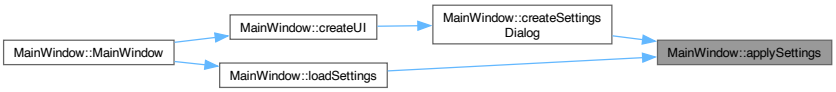
7.6.2.4 ~MainWindow() [2/2]

MainWindow::~MainWindow ()
MainWindow 析构函数。

7.6.3 成员函数说明

7.6.3.1 applySettings

void MainWindow::applySettings () [private], [slot]
应用设置
这是这个函数的调用关系图:



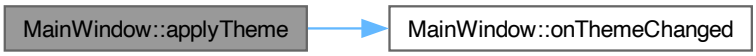
7.6.3.2 applyTheme()

void MainWindow::applyTheme (
bool darkMode) [private]
应用主题（暗色/浅色）
应用暗色/浅色主题

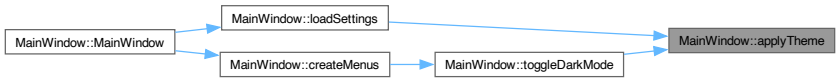
参数

darkMode	是否应用暗色模式

函数调用图:



这是这个函数的调用关系图:

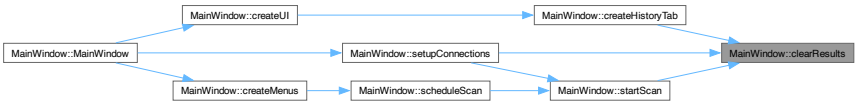


7.6.3.3 clearFilters

void MainWindow::clearFilters () [private], [slot]
清除过滤器
清除所有过滤器并显示所有扫描结果这是这个函数的调用关系图:

7.6.3.4 clearResults

void MainWindow::clearResults () [private], [slot]
清除扫描结果
这是这个函数的调用关系图:



7.6.3.5 closeEvent()

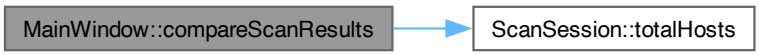
void MainWindow::closeEvent (
QCloseEvent * event) [override], [protected]
关闭事件处理程序。

参数

event	关闭事件。

7.6.3.6 compareScanResults

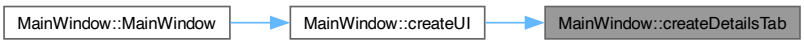
void MainWindow::compareScanResults () [private], [slot]
比较扫描结果
比较两次扫描会话的结果函数调用图:



这是这个函数的调用关系图:

7.6.3.7 createDetailsTab()

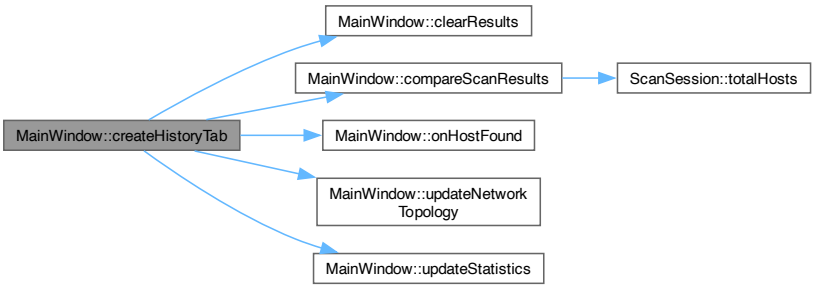
void MainWindow::createDetailsTab () [private]
创建主机详情标签页
这是这个函数的调用关系图:



7.6.3.8 createHistoryTab()

void MainWindow::createHistoryTab () [private]
创建扫描历史标签页

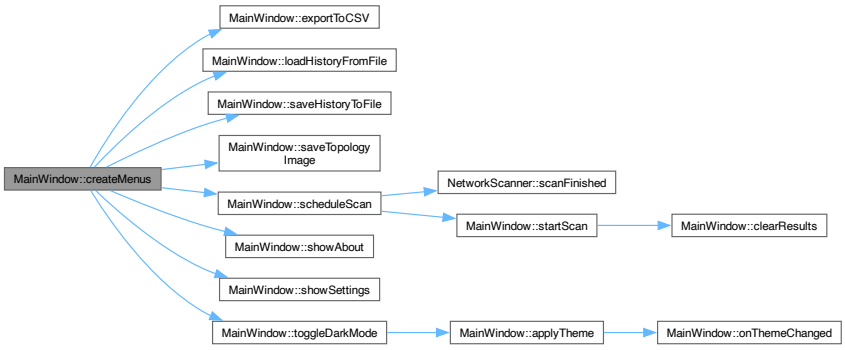
函数调用图:



这是这个函数的调用关系图:

7.6.3.9 createMenus() [1/2]

void MainWindow::createMenus () [private]
创建菜单
函数调用图:



这是这个函数的调用关系图:

7.6.3.10 createMenus() [2/2]

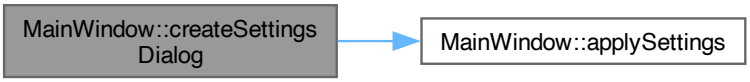
void MainWindow::createMenus () [private]
创建菜单栏。

7.6.3.11 createSecurityTab()

void MainWindow::createSecurityTab () [private]
创建安全相关标签页 (如果需要)

7.6.3.12 createSettingsDialog()

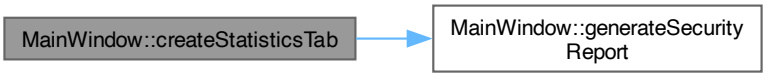
void MainWindow::createSettingsDialog () [private]
创建设置对话框
创建设置对话框（实际为设置标签页）函数调用图:



这是这个函数的调用关系图:

7.6.3.13 createStatisticsTab()

void MainWindow::createStatisticsTab () [private]
创建统计分析标签页
函数调用图:



这是这个函数的调用关系图:

7.6.3.14 createStatusbar()

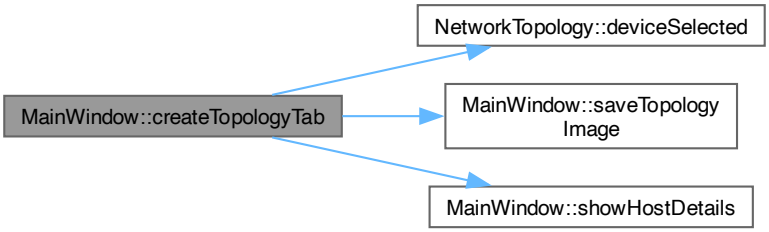
void MainWindow::createStatusbar () [private]
创建状态栏。

7.6.3.15 createToolbars()

void MainWindow::createToolbars () [private]
创建工具栏。

7.6.3.16 createTopologyTab()

void MainWindow::createTopologyTab () [private]
创建网络拓扑标签页
函数调用图:

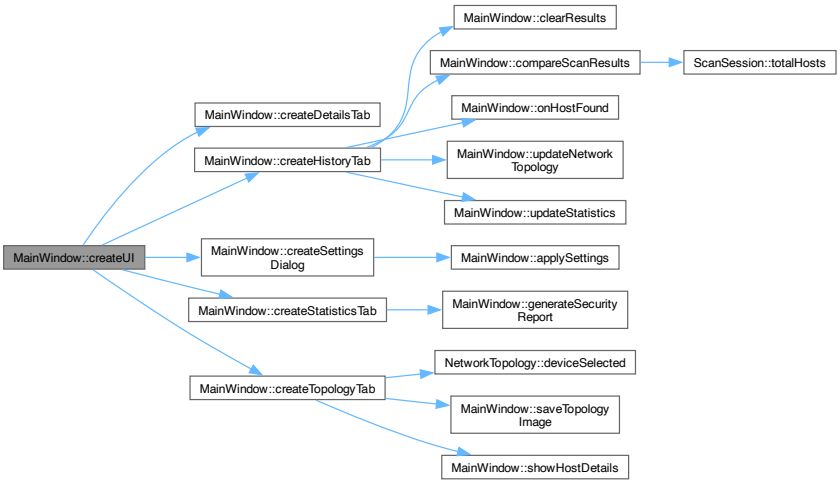


这是这个函数的调用关系图:

7.6.3.17 createUI()

void MainWindow::createUI () [private]
创建用户界面

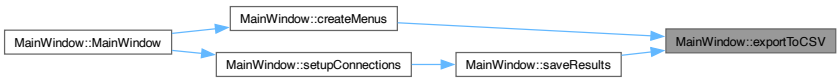
函数调用图:



这是这个函数的调用关系图:

7.6.3.18 exportToCSV

void MainWindow::exportToCSV () [private], [slot]
导出结果为CSV 文件
这是这个函数的调用关系图:



7.6.3.19 filterResults

void MainWindow::filterResults () [private], [slot]
过滤扫描结果

根据用户输入的条件过滤扫描结果表格函数调用图:

这是这个函数的调用关系图:

7.6.3.20 generateSecurityReport

void MainWindow::generateSecurityReport () [private], [slot]
生成安全报告
生成并显示安全报告这是这个函数的调用关系图:

7.6.3.21 handleDeviceSelectedFromTopology

void MainWindow::handleDeviceSelectedFromTopology (
const HostInfo & hostInfo) [private], [slot]

7.6.3.22 handleHostFound

void MainWindow::handleHostFound (
const HostInfo & hostInfo) [private], [slot]

7.6.3.23 handleScanError

void MainWindow::handleScanError (
const QString & errorMessage) [private], [slot]

7.6.3.24 handleScanFinished

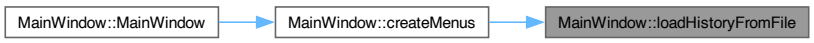
void MainWindow::handleScanFinished (
const QList< HostInfo > & results) [private], [slot]

7.6.3.25 handleScanProgress

void MainWindow::handleScanProgress (
int percentage,
const QString & message) [private], [slot]

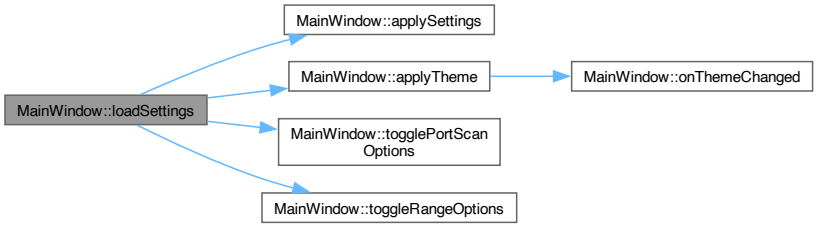
7.6.3.26 loadHistoryFromFile

void MainWindow::loadHistoryFromFile () [private], [slot]
从文件加载扫描历史
这是这个函数的调用关系图:

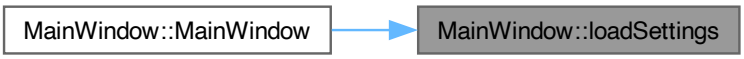


7.6.3.27 loadSettings()

void MainWindow::loadSettings () [private]
加载程序设置
函数调用图:



这是这个函数的调用关系图:



7.6.3.28 on_actionAbout_triggered

void MainWindow::on_actionAbout_triggered () [private], [slot]

7.6.3.29 on_actionConfigureScan_triggered

void MainWindow::on_actionConfigureScan_triggered () [private], [slot]

7.6.3.30 on_actionExit_triggered

void MainWindow::on_actionExit_triggered () [private], [slot]

7.6.3.31 on_actionStartScan_triggered

void MainWindow::on_actionStartScan_triggered () [private], [slot]

7.6.3.32 on_actionStopScan_triggered

void MainWindow::on_actionStopScan_triggered () [private], [slot]

7.6.3.33 on_actionViewAnalysis_triggered

void MainWindow::on_actionViewAnalysis_triggered () [private], [slot]

7.6.3.34 on_actionViewHistory_triggered

void MainWindow::on_actionViewHistory_triggered () [private], [slot]

7.6.3.35 on_actionViewTopology_triggered

void MainWindow::on_actionViewTopology_triggered () [private], [slot]

7.6.3.36 onHostFound

void MainWindow::onHostFound (
const HostInfo & host) [private], [slot]

当发现一个主机时调用
当发现一个主机时调用，更新UI 显示

参数

host	发现的主机信息

这是这个函数的调用关系图:

7.6.3.37 onScanError

void MainWindow::onScanError (
const QString & errorMessage) [private], [slot]

当扫描发生错误时调用

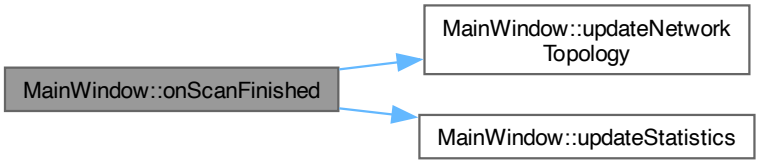
参数

errorMessage	错误信息

这是这个函数的调用关系图:

7.6.3.38 onScanFinished

void MainWindow::onScanFinished () [private], [slot]
当扫描结束时调用
当扫描结束时调用，更新UI 状态并处理结果函数调用图:



这是这个函数的调用关系图:

7.6.3.39 onScanProgress

void MainWindow::onScanProgress (
int progress) [private], [slot]
当扫描进度更新时调用

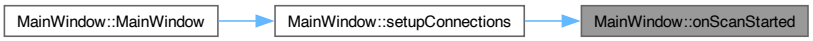
参数

progress	扫描进度 (0-100)

这是这个函数的调用关系图:

7.6.3.40 onScanStarted

void MainWindow::onScanStarted () [private], [slot]
当扫描开始时调用
当扫描开始时调用，更新UI 状态这是这个函数的调用关系图:



7.6.3.41 onThemeChanged

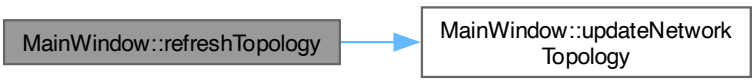
void MainWindow::onThemeChanged () [private], [slot]
当主题改变时调用
当主题（暗色/浅色模式）改变时调用，刷新UI 元素这是这个函数的调用关系图:

7.6.3.42 readSettings()

void MainWindow::readSettings () [private]
读取应用程序设置。

7.6.3.43 refreshTopology

void MainWindow::refreshTopology () [private], [slot]
刷新网络拓扑图
刷新网络拓扑图 (重新计算并显示) 函数调用图:



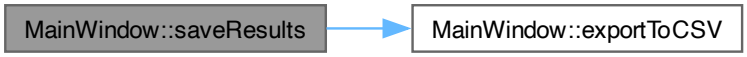
7.6.3.44 saveHistoryToFile

void MainWindow::saveHistoryToFile () [private], [slot]
保存扫描历史到文件
这是这个函数的调用关系图:

7.6.3.45 saveResults

void MainWindow::saveResults () [private], [slot]
保存扫描结果

保存扫描结果 (实际调用导出CSV) 函数调用图:



这是这个函数的调用关系图:

7.6.3.46 saveSettings()

void MainWindow::saveSettings () [private]
保存程序设置
这是这个函数的调用关系图:

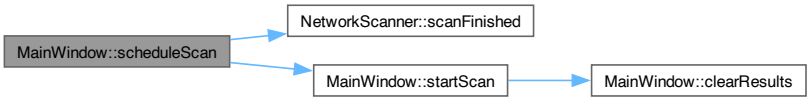
7.6.3.47 saveTopologyImage

void MainWindow::saveTopologyImage () [private], [slot]
保存网络拓扑图为图片
保存网络拓扑图为图片文件这是这个函数的调用关系图:

7.6.3.48 scheduleScan

void MainWindow::scheduleScan () [private], [slot]
计划扫描

计划一次未来的网络扫描函数调用图:

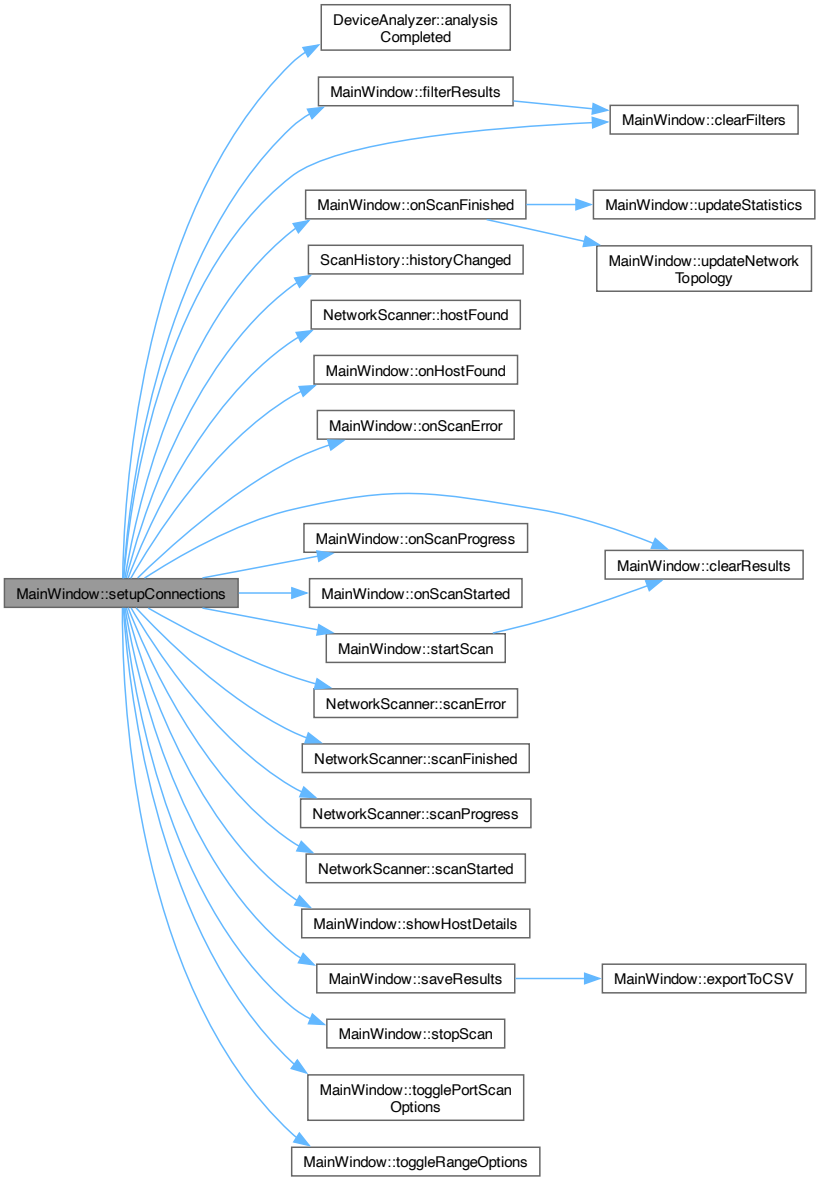


这是这个函数的调用关系图:

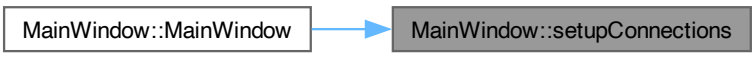
7.6.3.49 setupConnections()

void MainWindow::setupConnections () [private]
设置信号和槽的连接

函数调用图:



这是这个函数的调用关系图:



7.6.3.50 setupDockWidgets()

void MainWindow::setupDockWidgets () [private]

创建并设置中央区域的停靠窗口或堆叠窗口。

7.6.3.51 showAbout

void MainWindow::showAbout () [private], [slot]
显示关于对话框
这是这个函数的调用关系图:

7.6.3.52 showHistoryView

void MainWindow::showHistoryView () [private], [slot]
显示扫描历史视图

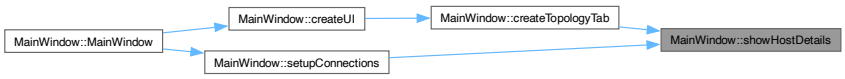
7.6.3.53 showHostDetails

void MainWindow::showHostDetails (
int row,
int column) [private], [slot]
显示主机详细信息

参数

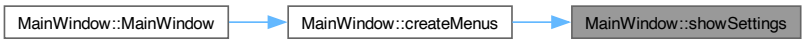
row	表格中的行号
column	表格中的列号
row	表格中的行号
column	表格中的列号 (未使用)

这是这个函数的调用关系图:



7.6.3.54 showSettings

void MainWindow::showSettings () [private], [slot]
显示设置界面
这是这个函数的调用关系图:



7.6.3.55 showStatisticsView

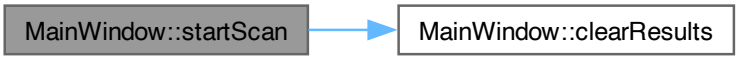
void MainWindow::showStatisticsView () [private], [slot]
显示统计分析视图

7.6.3.56 showTopologyView

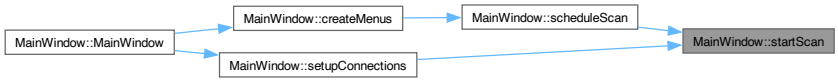
void MainWindow::showTopologyView () [private], [slot]
显示网络拓扑视图

7.6.3.57 startScan

void MainWindow::startScan () [private], [slot]
开始扫描
函数调用图:

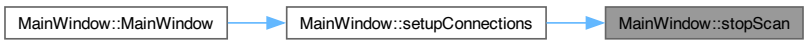


这是这个函数的调用关系图:



7.6.3.58 stopScan

void MainWindow::stopScan () [private], [slot]
停止扫描
这是这个函数的调用关系图:



7.6.3.59 toggleDarkMode

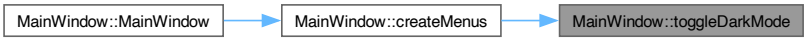
void MainWindow::toggleDarkMode (
bool enable) [private], [slot]
切换暗色模式
切换暗色/浅色主题

参数

enable	是否启用暗色模式

函数调用图:

这是这个函数的调用关系图:



7.6.3.60 togglePortScanOptions

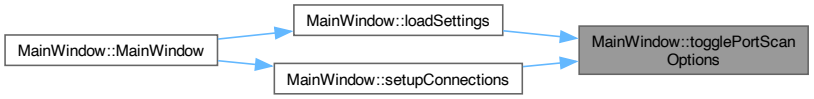
void MainWindow::togglePortScanOptions (
bool checked) [private], [slot]

切换自定义端口扫描选项的可用状态

参数

checked	是否选中

这是这个函数的调用关系图:



7.6.3.61 toggleRangeOptions

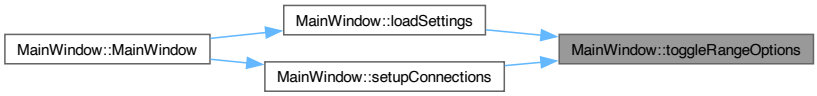
void MainWindow::toggleRangeOptions (
bool checked) [private], [slot]

切换自定义IP 范围选项的可用状态

参数

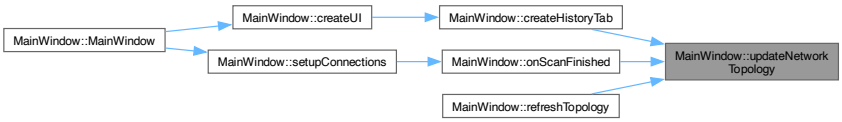
checked	是否选中

这是这个函数的调用关系图:



7.6.3.62 updateNetworkTopology

void MainWindow::updateNetworkTopology () [private], [slot]
更新网络拓扑图
更新网络拓扑图显示这是这个函数的调用关系图:

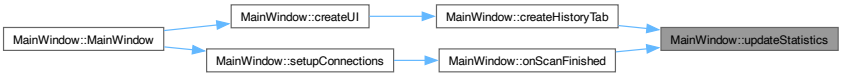


7.6.3.63 updatePortsList()

void MainWindow::updatePortsList () [private]
更新端口列表 (如果需要)

7.6.3.64 updateStatistics()

void MainWindow::updateStatistics () [private]
更新统计数据
更新统计数据分析和图表这是这个函数的调用关系图:



7.6.3.65 writeSettings()

void MainWindow::writeSettings () [private]
保存应用程序设置。

7.6.4 类成员变量说明

7.6.4.1 m_aboutAction

QAction* MainWindow::m_aboutAction [private]
关于动作

7.6.4.2 m_centralWidget

QWidget* MainWindow::m_centralWidget [private]
中央控件

7.6.4.3 m_clearButton

QPushButton* MainWindow::m_clearButton [private]
清除结果按钮

7.6.4.4 m_clearFilterButton

QPushButton* MainWindow::m_clearFilterButton [private]
清除过滤按钮

7.6.4.5 m_controlLayout

QHBoxLayout* MainWindow::m_controlLayout [private]
控制按钮布局

7.6.4.6 m_currentHostIndex

int MainWindow::m_currentHostIndex [private]
当前结果表格中选中的主机索引 (可能已废弃)

7.6.4.7 m_currentScanResults

QList<HostInfo> MainWindow::m_currentScanResults [private]
当前扫描操作的结果缓存。

7.6.4.8 m_currentScanTarget

QString MainWindow::m_currentScanTarget [private]
当前扫描目标。

7.6.4.9 m_customPortsCheckBox

QCheckBox* MainWindow::m_customPortsCheckBox [private]
自定义端口复选框

7.6.4.10 m_customRangeCheckBox

QCheckBox* MainWindow::m_customRangeCheckBox [private]
自定义IP 范围复选框

7.6.4.11 m_darkModeAction

QAction* MainWindow::m_darkModeAction [private]
暗色模式动作

7.6.4.12 m_darkModeEnabled

bool MainWindow::m_darkModeEnabled [private]
是否启用暗色模式

7.6.4.13 m_detailsLayout

QVBoxLayout* MainWindow::m_detailsLayout [private]
主机详情页布局

7.6.4.14 m_detailsTab

QWidget* MainWindow::m_detailsTab [private]
主机详情标签页

7.6.4.15 m_detailsTextEdit

QTextEdit* MainWindow::m_detailsTextEdit [private]
主机详情文本框

7.6.4.16 m_deviceAnalyzer

DeviceAnalyzer* MainWindow::m_deviceAnalyzer [private]
设备分析器

7.6.4.17 m_deviceTypeChartView

QChartView* MainWindow::m_deviceTypeChartView [private]
设备类型图表视图

7.6.4.18 m_endIPLineEdit

QLineEdit* MainWindow::m_endIPLineEdit [private]
结束IP 输入框

7.6.4.19 m_exitAction

QAction* MainWindow::m_exitAction [private]
退出动作

7.6.4.20 m_exportAction

QAction* MainWindow::m_exportAction [private]
导出结果动作

7.6.4.21 m_fileMenu

QMenu* MainWindow::m_fileMenu [private]
文件菜单

7.6.4.22 m_filterButton

QPushButton* MainWindow::m_filterButton [private]
应用过滤按钮

7.6.4.23 m_filterIPLineEdit

QLineEdit* MainWindow::m_filterIPLineEdit [private]
IP 过滤输入框

7.6.4.24 m_filterTypeComboBox

QComboBox* MainWindow::m_filterTypeComboBox [private]
设备类型过滤选择框

7.6.4.25 m_filterVendorComboBox

QComboBox* MainWindow::m_filterVendorComboBox [private]
厂商过滤选择框

7.6.4.26 m_filterWidget

QWidget* MainWindow::m_filterWidget [private]
过滤控件容器

7.6.4.27 m_helpMenu

QMenu* MainWindow::m_helpMenu [private]
帮助菜单

7.6.4.28 m_historyTab

QWidget* MainWindow::m_historyTab [private]
扫描历史标签页

7.6.4.29 m_historyTable

QTableWidget* MainWindow::m_historyTable [private]
扫描历史表格

7.6.4.30 m_hostsFound

int MainWindow::m_hostsFound [private]
已发现的主机数量

7.6.4.31 m_loadHistoryAction

QAction* MainWindow::m_loadHistoryAction [private]
加载扫描历史动作

7.6.4.32 m_mainLayout

QVBoxLayout* MainWindow::m_mainLayout [private]
主布局（扫描结果页）

7.6.4.33 m_networkScanner

NetworkScanner* MainWindow::m_networkScanner [private]
网络扫描器实例。

7.6.4.34 m_networkTopology

NetworkTopology* MainWindow::m_networkTopology [private]
网络拓扑控件

7.6.4.35 m_networkTopologyWidget

NetworkTopology* MainWindow::m_networkTopologyWidget [private]
网络拓扑图显示组件。

7.6.4.36 m_portDistributionChartView

QChartView* MainWindow::m_portDistributionChartView [private]
端口分布图表视图

7.6.4.37 m_portsGroupBox

QGroupBox* MainWindow::m_portsGroupBox [private]
端口设置组

7.6.4.38 m_portsLineEdit

QLineEdit* MainWindow::m_portsLineEdit [private]
端口输入框

7.6.4.39 m_progressBar

QProgressBar* MainWindow::m_progressBar [private]
扫描进度条

7.6.4.40 m_rangeGroupBox

QGroupBox* MainWindow::m_rangeGroupBox [private]
IP 范围设置组

7.6.4.41 m_resultsTable

QTableWidget* MainWindow::m_resultsTable [private]
扫描结果表格

7.6.4.42 m_saveButton

QPushButton* MainWindow::m_saveButton [private]
保存结果按钮

7.6.4.43 m_saveHistoryAction

QAction* MainWindow::m_saveHistoryAction [private]
保存扫描历史动作

7.6.4.44 m_saveTopologyAction

QAction* MainWindow::m_saveTopologyAction [private]
保存网络拓扑图动作

7.6.4.45 m_scanButton

QPushButton* MainWindow::m_scanButton [private]
开始扫描按钮

7.6.4.46 m_scanHistory

ScanHistory* MainWindow::m_scanHistory [private]
扫描历史管理器

7.6.4.47 m_scanner

NetworkScanner* MainWindow::m_scanner [private]
网络扫描器实例

7.6.4.48 m_scanTab

QWidget* MainWindow::m_scanTab [private]
扫描结果标签页

7.6.4.49 m_scheduleScanAction

QAction* MainWindow::m_scheduleScanAction [private]
计划扫描动作

7.6.4.50 m_securityReportText

QTextEdit* MainWindow::m_securityReportText [private]
安全报告文本框

7.6.4.51 m_sessionComboBox

QComboBox* MainWindow::m_sessionComboBox [private]
扫描会话选择框

7.6.4.52 m_settingsAction

QAction* MainWindow::m_settingsAction [private]
设置动作

7.6.4.53 m_settingsLayout

QVBoxLayout* MainWindow::m_settingsLayout [private]
设置标签页布局

7.6.4.54 m_settingsTab

QWidget* MainWindow::m_settingsTab [private]
扫描设置标签页

7.6.4.55 m_startIPLineEdit

QLineEdit* MainWindow::m_startIPLineEdit [private]
起始IP 输入框

7.6.4.56 m_statisticsTab

QWidget* MainWindow::m_statisticsTab [private]
统计分析标签页

7.6.4.57 m_statusBar

QStatusBar* MainWindow::m_statusBar [private]
状态栏

7.6.4.58 m_statusLabel

QLabel* MainWindow::m_statusLabel [private]
状态标签

7.6.4.59 m_stopButton

QPushButton* MainWindow::m_stopButton [private]
停止扫描按钮

7.6.4.60 m_tabWidget

QTabWidget* MainWindow::m_tabWidget [private]
标签页控件

7.6.4.61 m_timeoutSpinBox

QSpinBox* MainWindow::m_timeoutSpinBox [private]
超时时间设置框

7.6.4.62 m_toolsMenu

QMenu* MainWindow::m_toolsMenu [private]
工具菜单

7.6.4.63 m_topologyTab

QWidget* MainWindow::m_topologyTab [private]
网络拓扑标签页

7.6.4.64 m_vendorChartView

QChartView* MainWindow::m_vendorChartView [private]
厂商分布图表视图

7.6.4.65 m_viewMenu

QMenu* MainWindow::m_viewMenu [private]
视图菜单

7.6.4.66 ui

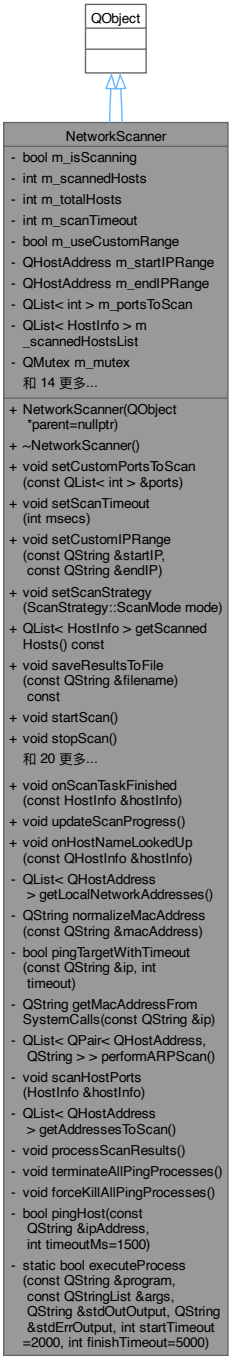
Ui::MainWindow* MainWindow::ui [private]
Qt Designer 生成的UI 类实例 (如果使用.ui 文件)。
该类的文档由以下文件生成:

- mainwindow.h
- src/gui/mainwindow.h
- mainwindow.cpp

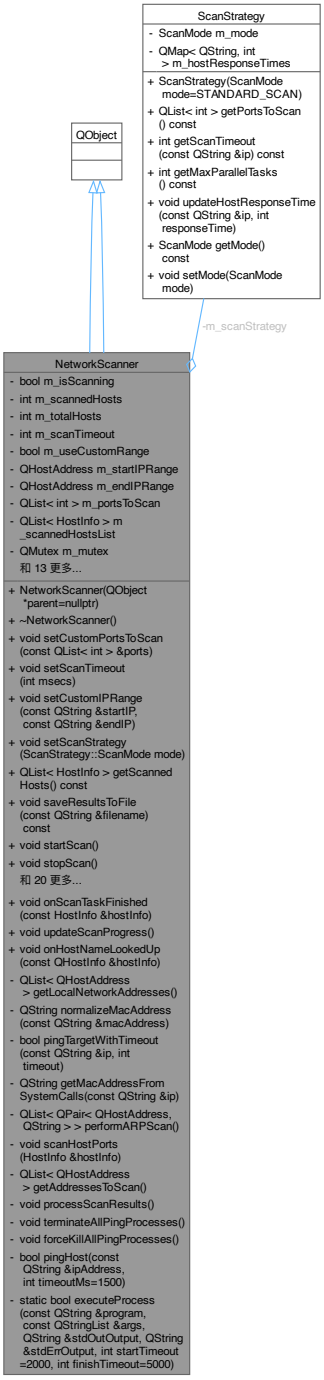
7.7 NetworkScanner 类参考

网络扫描器类
#include <networkscanner.h>

类 NetworkScanner 继承关系图:



NetworkScanner 的协作图:



Public 槽

- void onScanTaskFinished (const HostInfo &hostInfo)

  处理一个扫描任务完成的槽函数 (由 ScanTask 调用)
- void updateScanProgress ()

  更新整体扫描进度的槽函数
- void onHostNameLookedUp (const QHostInfo &hostInfo)

  当异步主机名查询完成后调用的槽函数

信号

- void hostFound (const HostInfo &host)
  当发现一个可达或不可达的主机时发出此信号
- void scanStarted ()
  当扫描过程开始时发出此信号
- void scanFinished ()
  当扫描过程（正常或异常）结束时发出此信号
- void scanProgress (int progress)
  扫描进度更新信号
- void scanError (const QString &errorMessage)
  当扫描过程中发生错误时发出此信号
- void hostFound (const HostInfo &hostInfo)
  当发现一个活动主机时发射此信号。
- void scanProgress (int percentage, const QString &message)
  报告扫描进度。
- void scanFinished (const QList< HostInfo > &results)
  当整个扫描过程完成时发射此信号。
- void scanError (const QString &errorMessage)
  当扫描过程中发生错误时发射此信号。

Public 成员函数

- NetworkScanner (QObject *parent=nullptr)
  构造函数
- ~NetworkScanner ()
  析构函数
- void setCustomPortsToScan (const QList< int > &ports)
  设置自定义端口扫描列表
- void setScanTimeout (int msecs)
  设置扫描超时时间
- void setCustomIPRange (const QString &startIP, const QString &endIP)
  设置自定义IP 范围进行扫描
- void setScanStrategy (ScanStrategy::ScanMode mode)
  设置扫描策略
- QList< HostInfo > getScannedHosts () const
  获取当前扫描到的所有主机信息
- void saveResultsToFile (const QString &filename) const
  将当前扫描结果保存到CSV 文件
- void startScan ()
  开始网络扫描
- void stopScan ()
  停止当前正在进行的扫描
- bool isScanning () const
  检查扫描器当前是否正在扫描
- QList< QHostAddress > quickPingScan (const QList< QHostAddress > &addresses)
  快速Ping 扫描方法 (可能已废弃或内部使用)
- bool isHostReachable (const QHostAddress &address, int timeout)
  检查单个主机是否可达 (可能已废弃或内部使用)
- bool isReachableOnPorts (const QHostAddress &address, const QList< int > &ports, int timeout)
  检查主机在多个端口上是否可达 (可能已废弃或内部使用)
- void scanHost (const QHostAddress &address)

扫描单个主机 (主要内部扫描逻辑调用)

- QString lookupHostName (const QHostAddress &address)

  查询指定IP 地址的主机名
- QString lookupMacAddress (const QHostAddress &address)

  查询指定IP 地址的MAC 地址
- QString lookupMacVendor (const QString &macAddress)

  根据MAC 地址查询对应的硬件厂商
- QString generatePseudoMACFromIP (const QString &ip)

  根据IP 地址生成一个伪MAC 地址 (用于无法获取真实MAC 的情况)
- void setScanMode (ScanMode mode)

  设置扫描模式 (旧的，建议使用 setScanStrategy)
- void setDebugMode (bool debug)

  设置是否启用调试模式
- void setRandomizeScan (bool randomize)

  设置是否启用随机化扫描IP 顺序
- bool checkHostReachable (const QHostAddress &address, int timeout)

  检查主机是否可达 (具体实现，可能调用 pingHost)
- NetworkScanner (QObject *parent=nullptr)

  NetworkScanner 构造函数。
- ~NetworkScanner ()

  NetworkScanner 析构函数。
- void setScanTargets (const QStringList &targets)

  设置要扫描的目标IP 范围或子网。
- void startScan ()

  设置扫描参数/配置。
- void stopScan ()

  停止当前正在进行的扫描。
- bool isScanning () const

  获取当前扫描状态。
- QList< QNetworkInterface > getLocalInterfaces ()

  获取本地网络接口列表。

Private **成员函数**

- QList< QHostAddress > getLocalNetworkAddresses ()

  获取本机所有活动的、非环回的网络接口的IP 地址列表
- QString normalizeMacAddress (const QString &macAddress)

  将MAC 地址规范化为标准格式 (XX:XX:XX:XX:XX:XX)
- bool pingTargetWithTimeout (const QString &ip, int timeout)

  使用 ping 命令检测目标主机是否可达，并指定超时
- QString getMacAddressFromSystemCalls (const QString &ip)

  通过系统调用 (如 arp -a) 获取指定IP 的MAC 地址 (可能平台相关且不可靠)
- QList< QPair< QHostAddress, QString > > performARPScan ()

  执行ARP 扫描以获取本地网络设备的MAC 地址 (可能平台相关且不可靠)
- void scanHostPorts (HostInfo &hostInfo)

  扫描指定主机的端口列表
- QList< QHostAddress > getAddressesToScan ()

  根据当前配置获取将要扫描的IP 地址列表
- void processScanResults ()

  定期处理和汇总扫描结果，更新进度，判断扫描是否完成
- void terminateAllPingProcesses ()

终止所有正在运行的 ping 进程 (尝试优雅终止)

- void forceKillAllPingProcesses ()

  强制终止所有正在运行的 ping 进程 (kill -9)

- bool pingHost (const QString &ipAddress, int timeoutMs=1500)

  使用系统 ping 命令检查主机是否可达，具有超时控制。

静态 Private 成员函数

- static bool executeProcess (const QString &program, const QStringList &args, QString &stdOut↩ Output, QString &stdErrOutput, int startTimeout=2000, int finishTimeout=5000)

  辅助函数：执行外部进程并获取其输出 (已标记为静态，但可能不应为静态如果需要访问成员)

Private 属性

- bool m_isScanning

  标记当前是否正在扫描

- int m_scannedHosts

  当前已扫描的主机数量 (用于进度计算)

- int m_totalHosts

  本次扫描总共需要扫描的主机数量

- int m_scanTimeout

  单个端口或 ping 操作的超时时间 (毫秒)

- bool m_useCustomRange

  是否使用用户自定义的IP 范围

- QHostAddress m_startIPRange

  自定义扫描的起始IP 地址

- QHostAddress m_endIPRange

  自定义扫描的结束IP 地址

- QList< int > m_portsToScan

  当前配置的待扫描端口列表

- QList< HostInfo > m_scannedHostsList

  存储所有扫描到的主机信息 (包括可达和不可达)

- QMutex m_mutex

  用于保护 m_scannedHostsList 和 m_scannedHosts 等共享数据的互斥锁

- QList< QFuture< void > > m_scanFutures

  存储QtConcurrent::run 返回的QFuture 对象，用于跟踪异步任务

- QThreadPool m_threadPool

  自定义线程池，用于执行扫描任务

- QMap< QString, QString > m_macAddressCache

  MAC 地址缓存 (IP -> MAC)，避免重复查询

- ScanStrategy m_scanStrategy

  当前使用的扫描策略对象

- QList< QHostAddress > m_activeHosts

  存储快速Ping 扫描发现的活跃主机 (可能已废弃)

- ScanMode m_scanMode

  当前扫描模式 (旧的，应优先使用 m_scanStrategy.getMode())

- bool m_debugMode

  是否启用调试模式，控制详细日志输出

- bool m_randomizeScan

  是否随机化IP 地址扫描顺序

- QElapsedTimer m_scanStartTime

  记录本次扫描开始的时间，用于计算总耗时

- QDateTime m_lastProgressUpdate

  上一次进度更新的时间，用于控制进度更新频率
- QSemaphore * m_pingSemaphore

  控制并发 ping 进程数量的信号量
- const int m_maxConcurrentPings = 8

  最大并发 ping 任务数量 (已从 15 减少)
- QStringList m_targets

  当前要扫描的目标列表。
- QList< HostInfo > m_foundHosts

  本次扫描已发现的主机列表。

7.7.1 详细描述

网络扫描器类
核心网络扫描类。
提供网络设备发现和端口扫描功能，支持多种扫描策略和并发处理。
此类负责发现本地网络中的活动主机，并收集每个主机的详细信息，如IP 地址、MAC 地址、开放端口、操作系统等。它可能会使用多种扫描技术（如ARP 扫描、ICMP ping、端口扫描等）。

注解

此类可能需要以提升的权限运行以执行某些扫描操作（如原始套接字访问）。

7.7.2 构造及析构函数说明

7.7.2.1 NetworkScanner() [1/2]

NetworkScanner::NetworkScanner (
QObject * parent = nullptr) [explicit]
构造函数
NetworkScanner 类构造函数

参数

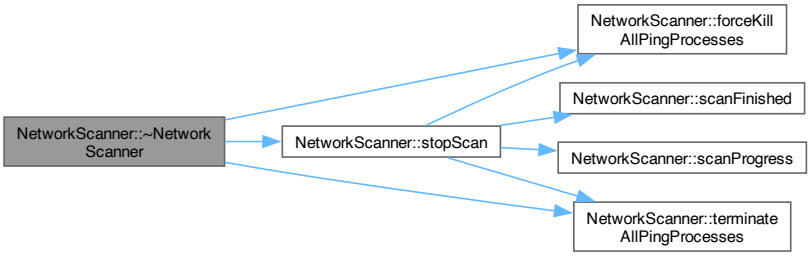
parent	父对象指针
parent	父对象

初始化扫描器参数和线程池。默认扫描常用端口，并设置线程池最大线程数。初始化 ping 信号量以控制并发 ping 操作。

7.7.2.2 ~NetworkScanner() [1/2]

NetworkScanner::~NetworkScanner ()
析构函数
NetworkScanner 类析构函数
停止任何正在进行的扫描并清理资源。
确保停止所有正在进行的扫描，终止所有相关进程 (特别是 ping 进程)，并清理所有分配的资源，包括线

程池、扫描结果列表和信号量。函数调用图:



7.7.2.3 NetworkScanner() [2/2]

NetworkScanner::NetworkScanner (
QObject * parent = nullptr) [explicit]

NetworkScanner 构造函数。

参数

parent	父QObject，默认为 nullptr。

7.7.2.4 ~NetworkScanner() [2/2]

NetworkScanner::~NetworkScanner ()

NetworkScanner 析构函数。

7.7.3 成员函数说明

7.7.3.1 checkHostReachable()

bool NetworkScanner::checkHostReachable (
const QHostAddress & address,
int timeout)

检查主机是否可达 (具体实现，可能调用 pingHost)

参数

address	主机地址
timeout	超时时间 (毫秒)

返回

主机是否可达

7.7.3.2 executeProcess()

bool NetworkScanner::executeProcess (
const QString & program,
const QStringList & args,
QString & stdOutOutput,
QString & stdErrOutput,

int startTimeout = 2000,
int finishTimeout = 5000) [static], [private]

辅助函数：执行外部进程并获取其输出 (已标记为静态，但可能不应为静态如果需要访问成员)
执行外部进程并获取其输出 (已废弃)。

参数

program	要执行的程序名或路径
args	传递给程序的参数列表
stdOutOutput	用于接收标准输出的字符串引用
stdErrOutput	用于接收标准错误的字符串引用
startTimeout	等待进程启动的超时时间 (毫秒)
finishTimeout	等待进程完成的超时时间 (毫秒)

返回

如果进程成功执行并退出则返回 true，否则返回 false

注解

此方法已被重构为不再实际执行外部进程，以提高稳定性。

参数

program	程序路径。
args	参数列表。
stdOutOutput	标准输出结果 (引用)。
stdErrOutput	标准错误结果 (引用)。
startTimeout	启动超时 (毫秒)。
finishTimeout	完成超时 (毫秒)。

返回

始终返回 false，因为外部进程调用已被禁用。

注解

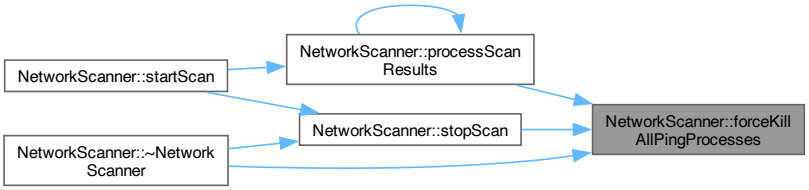
为了提高稳定性和跨平台兼容性，此函数已被修改为不执行任何外部进程，并会设置 stdErrOutput 为” 禁用了外部进程”。

7.7.3.3 forceKillAllPingProcesses()

void NetworkScanner::forceKillAllPingProcesses () [private]

强制终止所有正在运行的 ping 进程 (kill -9)
强制终止所有正在运行的 ping 进程 (通常使用 kill -9 或 taskkill /F)。
此方法会根据操作系统使用特定的系统命令来强制结束所有名为”ping” 或”ping.exe” 的进程。在 macOS 和Linux 上，会多次尝试并验证清理效果。也会输出警告如果仍有进程未能终止。这是这个函数的调用关

系图:



7.7.3.4 generatePseudoMACFromIP()

QString NetworkScanner::generatePseudoMACFromIP (
const QString & ip)

根据IP 地址生成一个伪MAC 地址 (用于无法获取真实MAC 的情况)
根据IP 地址生成一个格式化的伪MAC 地址。

参数

ip	IP 地址

返回

生成的伪MAC 地址，格式如”SM:XX:XX:XX:XX”

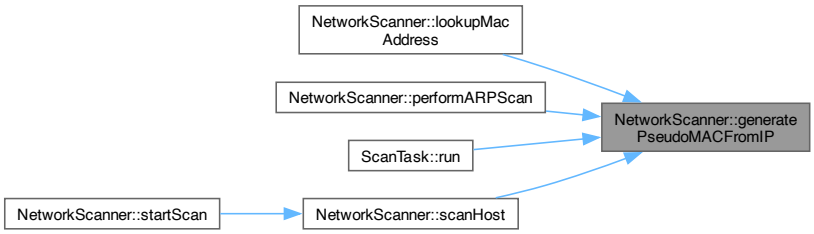
参数

ip	IP 地址字符串 (例如”192.168.1.10”)。

返回

伪MAC 地址字符串，格式为”SM:XX:XX:XX:XX”，其中XX 是IP 地址各部分的十六进制表示。如果IP 格式无效，则返回一个固定的默认伪MAC ”SM:00:00:00:01”。

”SM” 前缀用于标识这是一个模拟的MAC 地址。这是这个函数的调用关系图:



7.7.3.5 getAddressesToScan()

QList< QHostAddress > NetworkScanner::getAddressesToScan () [private]
根据当前配置获取将要扫描的IP 地址列表

返回

IP 地址列表

7.7.3.6 getLocalInterfaces()

QList< QNetworkInterface > NetworkScanner::getLocalInterfaces ()
获取本地网络接口列表。

返回

QNetworkInterface 对象列表。

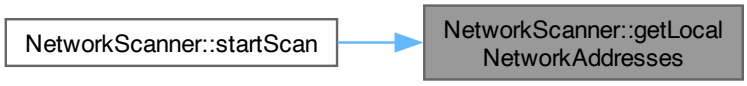
7.7.3.7 getLocalNetworkAddresses()

QList< QHostAddress > NetworkScanner::getLocalNetworkAddresses () [private]
获取本机所有活动的、非环回的网络接口的IP 地址列表

返回

IP 地址列表 (QHostAddress)

这是这个函数的调用关系图:



7.7.3.8 getMacAddressFromSystemCalls()

QString NetworkScanner::getMacAddressFromSystemCalls (
const QString & ip) [private]
通过系统调用 (如 arp -a) 获取指定IP 的MAC 地址 (可能平台相关且不可靠)

参数

ip	目标IP 地址字符串

返回

MAC 地址字符串，如果无法获取则为空

7.7.3.9 getScannedHosts()

QList< HostInfo > NetworkScanner::getScannedHosts () const
获取当前扫描到的所有主机信息
获取扫描结果

返回

包含所有已发现主机信息的列表

扫描到的主机信息列表

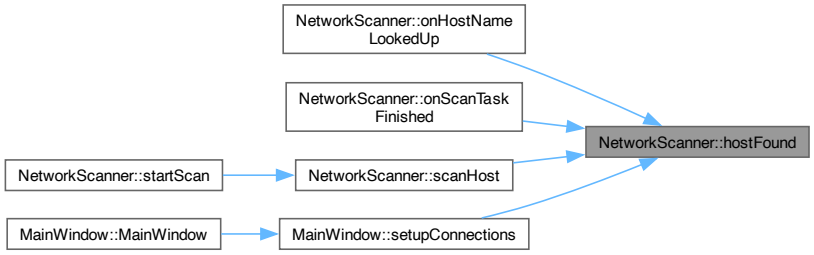
7.7.3.10 hostFound [1/2]

void NetworkScanner::hostFound (
const HostInfo & host) [signal]
当发现一个可达或不可达的主机时发出此信号

参数

host	包含主机信息的 HostInfo 结构体

这是这个函数的调用关系图:



7.7.3.11 hostFound [2/2]

void NetworkScanner::hostFound (
const HostInfo & hostInfo) [signal]
当发现一个活动主机时发射此信号。

参数

hostInfo	发现的主机的详细信息。

7.7.3.12 isHostReachable()

bool NetworkScanner::isHostReachable (
const QHostAddress & address,
int timeout)
检查单个主机是否可达 (可能已废弃或内部使用)

参数

address	主机地址

timeout	超时时间 (毫秒)

返回

主机是否可达

7.7.3.13 isReachableOnPorts()

bool NetworkScanner::isReachableOnPorts (
const QHostAddress & address,
const QList< int > & ports,
int timeout)

检查主机在多个端口上是否可达 (可能已废弃或内部使用)

参数

address	主机地址
ports	端口列表
timeout	超时时间 (毫秒)

返回

是否至少有一个端口可达

7.7.3.14 isScanning() [1/2]

bool NetworkScanner::isScanning () const

检查扫描器当前是否正在扫描

返回

如果正在扫描则返回 true，否则返回 false

7.7.3.15 isScanning() [2/2]

bool NetworkScanner::isScanning () const

获取当前扫描状态。

返回

如果正在扫描则返回 true，否则返回 false。

7.7.3.16 lookupHostName()

QString NetworkScanner::lookupHostName (
const QHostAddress & address)

查询指定IP 地址的主机名

参数

address	主机IP 地址

返回

主机名，如果无法解析则返回IP 地址或” 未知主机”

这是这个函数的调用关系图:

7.7.3.17 lookupMacAddress()

QString NetworkScanner::lookupMacAddress (
const QHostAddress & address)

查询指定IP 地址的MAC 地址

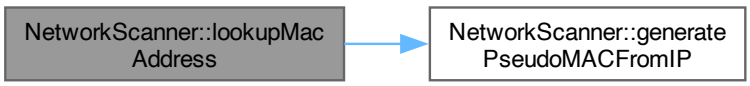
参数

address	主机IP 地址

返回

MAC 地址，如果无法获取则返回” 未知” 或伪MAC 地址

函数调用图:



7.7.3.18 lookupMacVendor()

QString NetworkScanner::lookupMacVendor (
const QString & macAddress)

根据MAC 地址查询对应的硬件厂商

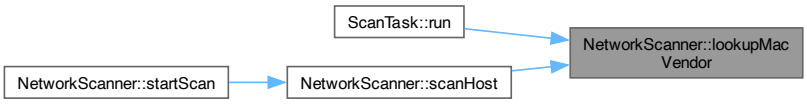
参数

macAddress	MAC 地址 (通常为前 6 位OUI)

返回

厂商名称，如果未知则返回” 未知厂商”

这是这个函数的调用关系图:



7.7.3.19 normalizeMacAddress()

QString NetworkScanner::normalizeMacAddress (
const QString & macAddress) [private]

将MAC 地址规范化为标准格式 (XX:XX:XX:XX:XX:XX)
将MAC 地址字符串规范化为大写的”XX:XX:XX:XX:XX:XX” 格式。

参数

macAddress	原始MAC 地址字符串

返回

规范化后的MAC 地址，如果无法规范化则返回原始或” 未知”

参数

macAddress	原始MAC 地址字符串，可以包含”:” 或”-” 作为分隔符，或无分隔符。

返回

规范化后的MAC 地址。如果输入无效或无法规范化，则返回” 未知”。

这是这个函数的调用关系图:

7.7.3.20 onHostNameLookedUp

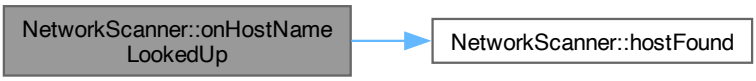
void NetworkScanner::onHostNameLookedUp (
const QHostInfo & hostInfo) [slot]

当异步主机名查询完成后调用的槽函数
当异步主机名查询 (QHostInfo::lookupHost()) 完成后调用的槽函数。

参数

hostInfo	QHostInfo 对象，包含查询结果
hostInfo	QHostInfo 对象，包含查询结果 (IP 地址、主机名、错误信息等)。

如果查询成功且获取到主机名，则会更新 m_scannedHostsList 中对应IP 地址条目的主机名，并重新发出 hostFound() 信号以通知UI 更新。函数调用图:



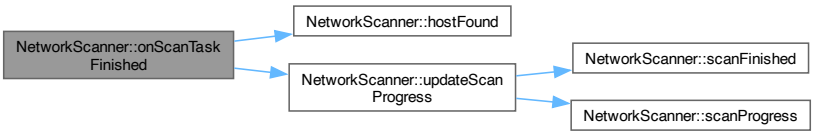
7.7.3.21 onScanTaskFinished

void NetworkScanner::onScanTaskFinished (
const HostInfo & hostInfo) [slot]
处理一个扫描任务完成的槽函数 (由 ScanTask 调用)
处理一个扫描任务完成的槽函数 (由 ScanTask 通过信号槽机制异步调用)。

参数

hostInfo	扫描到的主机信息
hostInfo	包含扫描结果的 HostInfo 结构体。

此方法在主线程中执行。它会将接收到的 hostInfo 添加到 m_scannedHostsList (受互斥锁保护)，发出 hostFound() 信号，并调用 updateScanProgress() 更新整体扫描进度。函数调用图:



7.7.3.22 performARPScan()

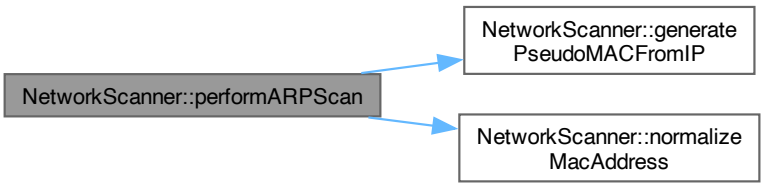
QList< QPair< QHostAddress, QString > > NetworkScanner::performARPScan () [private]
执行ARP 扫描以获取本地网络设备的MAC 地址 (可能平台相关且不可靠)
执行ARP 扫描以获取本地网络接口的MAC 地址及其子网内一些IP 的伪MAC。

返回

IP 地址和MAC 地址对的列表

IP 地址 (QHostAddress) 和 MAC 地址 (QString) 对的列表。

此方法不再执行外部 arp 命令。它会遍历所有活动的网络接口，获取接口本身的MAC 地址，并为其IP 地址（如果尚未在缓存中）创建条目。同时，会为每个接口所在子网内的一些IP 地址生成伪MAC 地址并添加到结果中。所有获取或生成的MAC 地址都会存入 m_macAddressCache。函数调用图:



7.7.3.23 pingHost()

bool NetworkScanner::pingHost (
const QString & ipAddress,
int timeoutMs = 1500) [private]

使用系统 ping 命令检查主机是否可达，具有超时控制。

参数

ipAddress	目标主机的IP 地址字符串。
timeoutMs	ping 操作的超时时间（毫秒），默认为 1500ms。

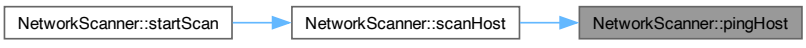
返回

如果主机在超时时间内响应 ping，则返回 true；否则返回 false。

注解

此方法会尝试获取 m_pingSemaphore 信号量以控制并发 ping 数量。

这是这个函数的调用关系图:



7.7.3.24 pingTargetWithTimeout()

bool NetworkScanner::pingTargetWithTimeout (
const QString & ip,
int timeout) [private]

使用 ping 命令检测目标主机是否可达，并指定超时

参数

ip	目标IP 地址字符串

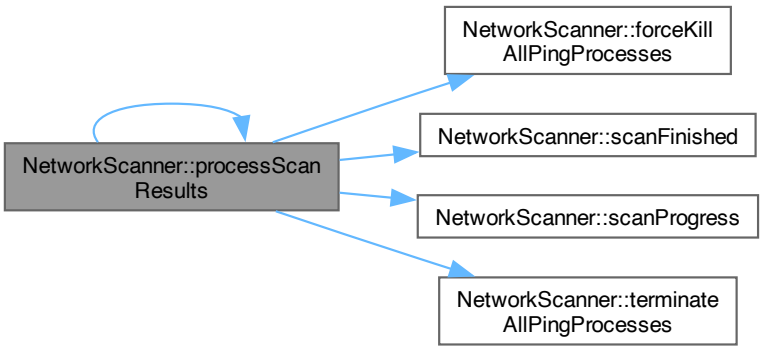
timeout	ping 命令的超时时间 (毫秒)

返回

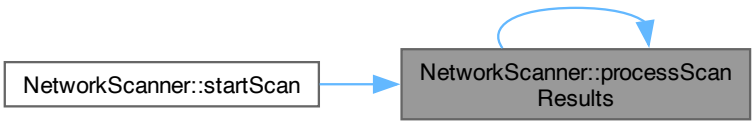
如果主机可达则返回 true，否则返回 false

7.7.3.25 processScanResults()

void NetworkScanner::processScanResults () [private]
定期处理和汇总扫描结果，更新进度，判断扫描是否完成
函数调用图:



这是这个函数的调用关系图:



7.7.3.26 quickPingScan()

QList< QHostAddress > NetworkScanner::quickPingScan (
const QList< QHostAddress > & addresses)
快速Ping 扫描方法 (可能已废弃或内部使用)

参数

addresses	要扫描的地址列表

返回

活跃的主机地址列表

7.7.3.27 saveResultsToFile()

void NetworkScanner::saveResultsToFile (
const QString & filename) const

将当前扫描结果保存到CSV 文件
保存结果到文件

参数

filename	要保存的文件名
filename	文件名

7.7.3.28 scanError [1/2]

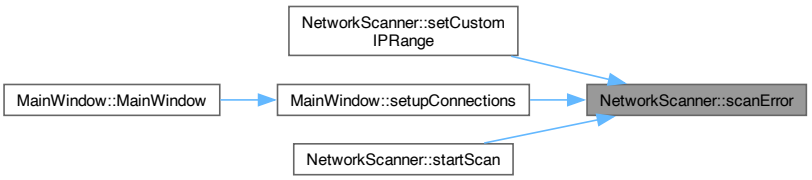
void NetworkScanner::scanError (
const QString & errorMessage) [signal]

当扫描过程中发生错误时发出此信号

参数

errorMessage	描述错误的字符串

这是这个函数的调用关系图:



7.7.3.29 scanError [2/2]

void NetworkScanner::scanError (
const QString & errorMessage) [signal]

当扫描过程中发生错误时发射此信号。

参数

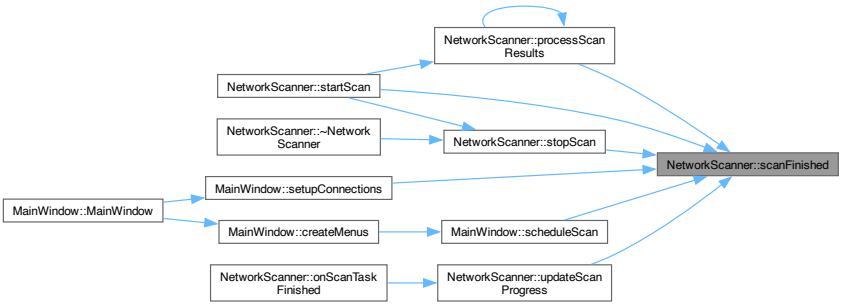
errorMessage	错误信息描述。

7.7.3.30 scanFinished [1/2]

void NetworkScanner::scanFinished () [signal]

当扫描过程（正常或异常）结束时发出此信号

这是这个函数的调用关系图:



7.7.3.31 scanFinished [2/2]

void NetworkScanner::scanFinished (
const QList< HostInfo > & results) [signal]

当整个扫描过程完成时发射此信号。

参数

results	所有发现的主机列表。

7.7.3.32 scanHost()

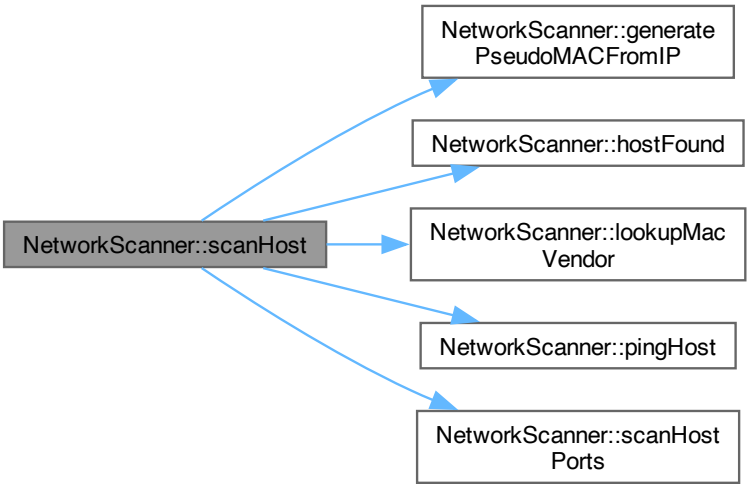
void NetworkScanner::scanHost (
const QHostAddress & address)

扫描单个主机 (主要内部扫描逻辑调用)

参数

address	要扫描的主机IP 地址

函数调用图:



这是这个函数的调用关系图:

7.7.3.33 scanHostPorts()

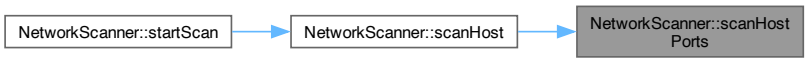
void NetworkScanner::scanHostPorts (
HostInfo & hostInfo) [private]

扫描指定主机的端口列表

参数

hostInfo	要更新端口信息的主机结构体 (会被修改)

这是这个函数的调用关系图:



7.7.3.34 scanProgress [1/2]

void NetworkScanner::scanProgress (
int percentage,
const QString & message) [signal]

报告扫描进度。

参数

percentage	完成百分比 (0-100)。
message	当前扫描状态的描述性消息。

7.7.3.35 scanProgress [2/2]

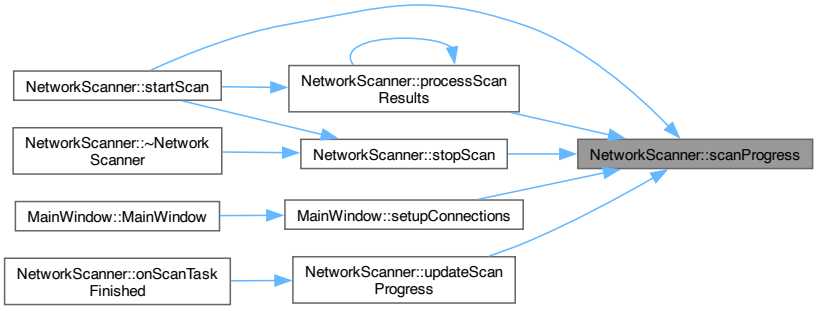
void NetworkScanner::scanProgress (
int progress) [signal]

扫描进度更新信号

参数

progress	当前扫描进度百分比 (0-100)

这是这个函数的调用关系图:

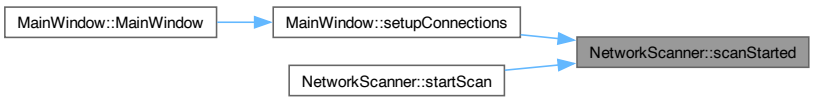


7.7.3.36 scanStarted

void NetworkScanner::scanStarted () [signal]

当扫描过程开始时发出此信号

这是这个函数的调用关系图:



7.7.3.37 setCustomIPRange()

void NetworkScanner::setCustomIPRange (
const QString & startIP,
const QString & endIP)

设置自定义IP 范围进行扫描
设置自定义IP 范围

参数

startIP	起始IP 地址 (例如”192.168.1.1”)
endIP	结束IP 地址 (例如”192.168.1.254”)

如果扫描正在进行，此设置无效。

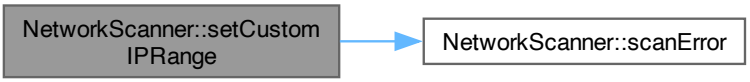
参数

startIP	起始IP 地址
endIP	结束IP 地址

注解

如果扫描正在进行，此设置将无效。如果IP 范围无效，会发出 scanError 信号。

函数调用图:



7.7.3.38 setCustomPortsToScan()

void NetworkScanner::setCustomPortsToScan (
const QList< int > & ports)

设置自定义端口扫描列表

参数

ports	要扫描的端口列表。如果扫描正在进行，此设置无效。
ports	要扫描的端口列表

注解

如果扫描正在进行，此设置将无效。

7.7.3.39 setDebugMode()

void NetworkScanner::setDebugMode (
bool debug) [inline]

设置是否启用调试模式

参数

debug	如果为 true，则启用调试输出

7.7.3.40 setRandomizeScan()

void NetworkScanner::setRandomizeScan (
bool randomize) [inline]

设置是否启用随机化扫描IP 顺序

参数

randomize	如果为 true，则打乱IP 扫描顺序

7.7.3.41 setScanMode()

void NetworkScanner::setScanMode (
ScanMode mode) [inline]

设置扫描模式 (旧的，建议使用 setScanStrategy)

参数

mode	要设置的扫描模式

7.7.3.42 setScanStrategy()

void NetworkScanner::setScanStrategy (
ScanStrategy::ScanMode mode)

设置扫描策略

参数

mode	要使用的扫描模式 (快速, 标准, 深度)

7.7.3.43 setScanTargets()

void NetworkScanner::setScanTargets (
const QStringList & targets)

设置要扫描的目标IP 范围或子网。

参数

targets	IP 地址列表、IP 范围 (例如”192.168.1.1-192.168.1.254”) 或子网掩码 (例如”192.168.1.0/24”)。

7.7.3.44 setScanTimeout()

void NetworkScanner::setScanTimeout (
int msecs)

设置扫描超时时间

参数

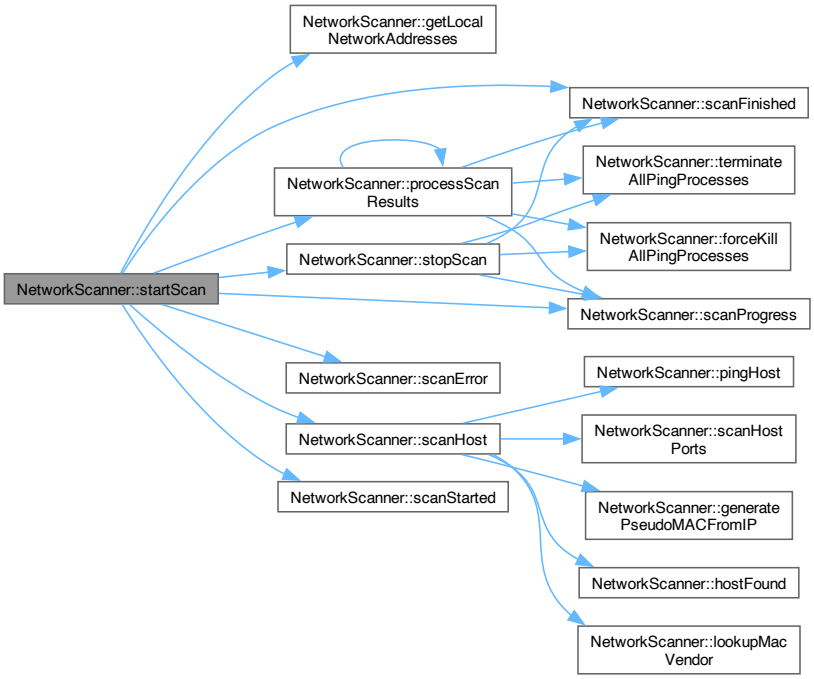
msecs	超时时间 (毫秒)。如果扫描正在进行，此设置无效。
msecs	超时时间（毫秒）

注解

如果扫描正在进行，此设置将无效。超时值必须大于 0。

7.7.3.45 startScan() [1/2]

void NetworkScanner::startScan ()
开始网络扫描
根据当前设置 (IP 范围、端口、策略等) 启动扫描过程。会触发 scanStarted() 信号。函数调用图:



7.7.3.46 startScan() [2/2]

void NetworkScanner::startScan ()
设置扫描参数/配置。
例如，可以设置端口扫描范围、超时时间、使用的扫描方法等。

参数

config	一个包含配置选项的结构体或QVariantMap。

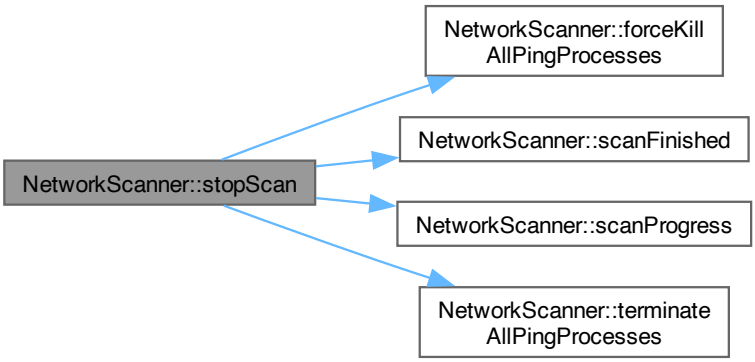
开始网络扫描。
这是一个异步操作。扫描进度和结果通过信号报告。

参见

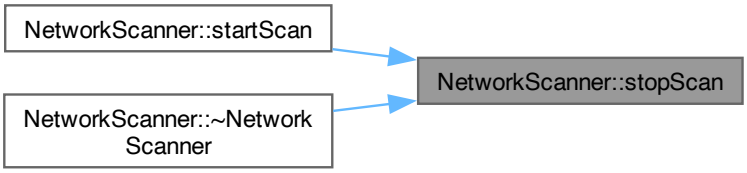
scanProgress, hostFound, scanFinished

7.7.3.47 stopScan() [1/2]

void NetworkScanner::stopScan ()
停止当前正在进行的扫描
会尝试终止所有扫描任务和相关进程，并触发 scanFinished() 信号。函数调用图:



这是这个函数的调用关系图:



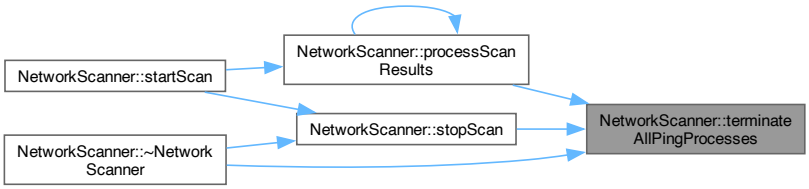
7.7.3.48 stopScan() [2/2]

void NetworkScanner::stopScan ()
停止当前正在进行的扫描。

7.7.3.49 terminateAllPingProcesses()

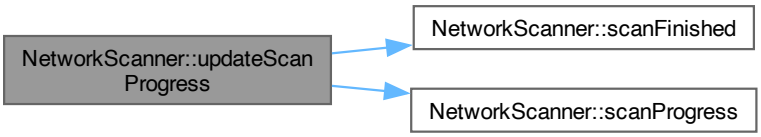
void NetworkScanner::terminateAllPingProcesses () [private]
终止所有正在运行的 ping 进程 (尝试优雅终止)
终止所有正在运行的 ping 进程 (尝试优雅终止)。
此方法会根据操作系统使用不同的命令 (pkill, taskkill) 来终止名为”ping” 的进程。在 macOS 上，会多

次尝试并验证终止效果。也会尝试释放所有 ping 信号量。这是这个函数的调用关系图:

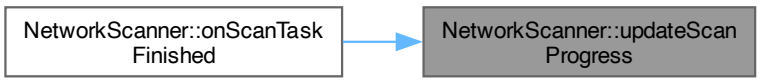


7.7.3.50 updateScanProgress

void NetworkScanner::updateScanProgress () [slot]
更新整体扫描进度的槽函数
更新整体扫描进度。
此方法计算当前扫描进度百分比 ((m_scannedHosts * 100) / m_totalHosts)，并发出 scanProgress() 信号。
如果已扫描主机数达到总主机数 (m_scannedHosts >= m_totalHosts) 且扫描仍在进行 (m_isScanning)，则将 m_isScanning 设置为 false 并发出 scanFinished() 信号。函数调用图:



这是这个函数的调用关系图:



7.7.4 类成员变量说明

7.7.4.1 m_activeHosts

QList<QHostAddress> NetworkScanner::m_activeHosts [private]
存储快速Ping 扫描发现的活跃主机 (可能已废弃)

7.7.4.2 m_debugMode

bool NetworkScanner::m_debugMode [private]
是否启用调试模式，控制详细日志输出

7.7.4.3 m_endIPRange

QHostAddress NetworkScanner::m_endIPRange [private]
自定义扫描的结束IP 地址

7.7.4.4 m_foundHosts

QList<HostInfo> NetworkScanner::m_foundHosts [private]
本次扫描已发现的主机列表。

7.7.4.5 m_isScanning

bool NetworkScanner::m_isScanning [private]
标记当前是否正在扫描
标记当前是否正在进行扫描。

7.7.4.6 m_lastProgressUpdate

QDateTime NetworkScanner::m_lastProgressUpdate [private]
上一次进度更新的时间，用于控制进度更新频率

7.7.4.7 m_macAddressCache

QMap<QString, QString> NetworkScanner::m_macAddressCache [private]
MAC 地址缓存 (IP -> MAC)，避免重复查询

7.7.4.8 m_maxConcurrentPings

const int NetworkScanner::m_maxConcurrentPings = 8 [private]
最大并发 ping 任务数量 (已从 15 减少)

7.7.4.9 m_mutex

QMutex NetworkScanner::m_mutex [private]
用于保护 m_scannedHostsList 和 m_scannedHosts 等共享数据的互斥锁

7.7.4.10 m_pingSemaphore

QSemaphore* NetworkScanner::m_pingSemaphore [private]
控制并发 ping 进程数量的信号量

7.7.4.11 m_portsToScan

QList<int> NetworkScanner::m_portsToScan [private]
当前配置的待扫描端口列表

7.7.4.12 m_randomizeScan

bool NetworkScanner::m_randomizeScan [private]
是否随机化IP 地址扫描顺序

7.7.4.13 m_scanFutures

QList<QFuture<void> > NetworkScanner::m_scanFutures [private]
存储QtConcurrent::run 返回的QFuture 对象，用于跟踪异步任务

7.7.4.14 m_scanMode

ScanMode NetworkScanner::m_scanMode [private]
当前扫描模式 (旧的，应优先使用 m_scanStrategy.getMode())

7.7.4.15 m_scannedHosts

int NetworkScanner::m_scannedHosts [private]
当前已扫描的主机数量 (用于进度计算)

7.7.4.16 m_scannedHostsList

QList<HostInfo> NetworkScanner::m_scannedHostsList [private]
存储所有扫描到的主机信息 (包括可达和不可达)

7.7.4.17 m_scanStartTime

QElapsedTimer NetworkScanner::m_scanStartTime [private]
记录本次扫描开始的时间，用于计算总耗时

7.7.4.18 m_scanStrategy

ScanStrategy NetworkScanner::m_scanStrategy [private]
当前使用的扫描策略对象

7.7.4.19 m_scanTimeout

int NetworkScanner::m_scanTimeout [private]
单个端口或 ping 操作的超时时间 (毫秒)

7.7.4.20 m_startIPRange

QHostAddress NetworkScanner::m_startIPRange [private]
自定义扫描的起始IP 地址

7.7.4.21 m_targets

QStringList NetworkScanner::m_targets [private]
当前要扫描的目标列表。

7.7.4.22 m_threadPool

QThreadPool NetworkScanner::m_threadPool [private]
自定义线程池，用于执行扫描任务

7.7.4.23 m_totalHosts

int NetworkScanner::m_totalHosts [private]
本次扫描总共需要扫描的主机数量

7.7.4.24 m_useCustomRange

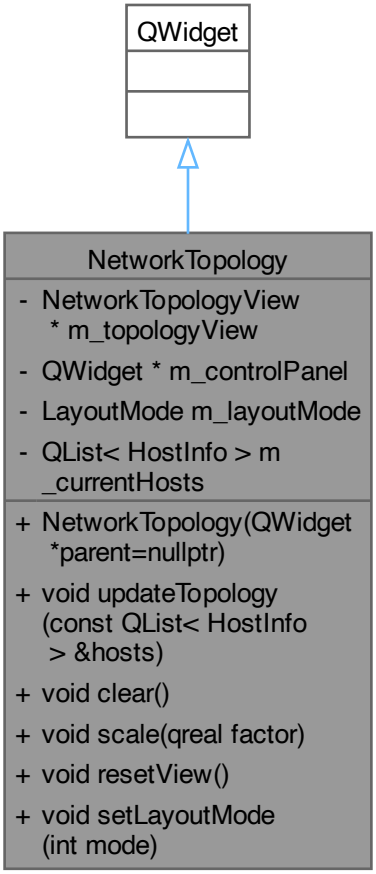
bool NetworkScanner::m_useCustomRange [private]
是否使用用户自定义的IP 范围
该类的文档由以下文件生成:

- networkscanner.h
- src/core/networkscanner.h
- NetScanner_autogen/JRIAJ772TK/moc_networkscanner.cpp
- networkscanner.cpp

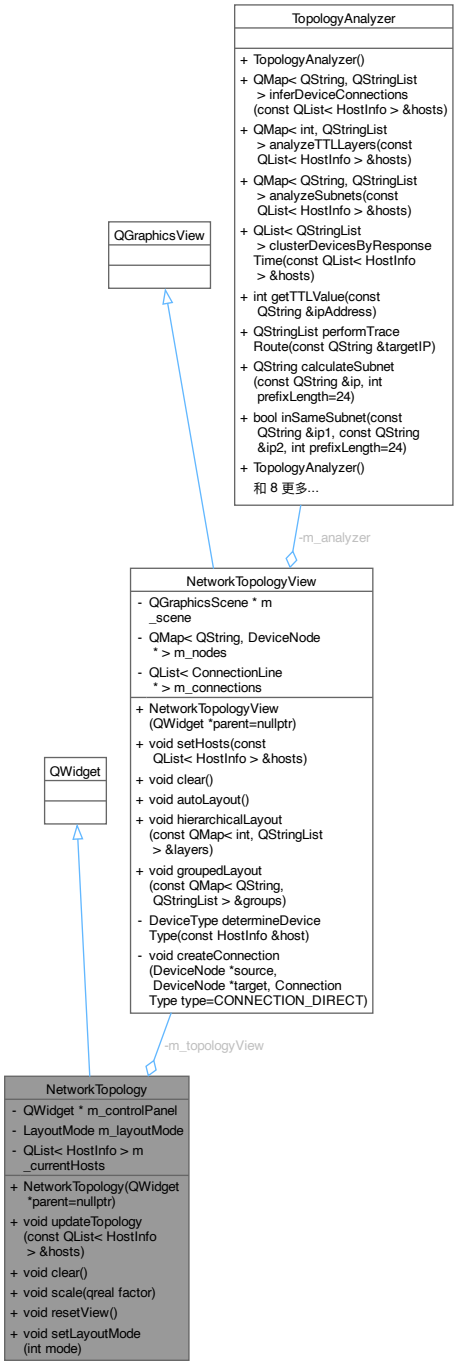
7.8 NetworkTopology 类参考

网络拓扑组件的主控件，继承自 QWidget。
#include <networktopology.h>

类 NetworkTopology 继承关系图:



NetworkTopology 的协作图:



信号

- void deviceSelected (const HostInfo &host)

  当用户在拓扑图中选择一个设备时发出此信号 (从 NetworkTopologyView 转发)。

Public 成员函数

- NetworkTopology (QWidget *parent=nullptr)

  NetworkTopology 构造函数。

- void updateTopology (const QList< HostInfo > &hosts)
  更新拓扑图显示的主机列表。
- void clear ()
  清除当前显示的拓扑图。
- void scale (qreal factor)
  缩放拓扑图视图。
- void resetView ()
  重置拓扑图视图，使其适应所有节点。
- void setLayoutMode (int mode)
  设置拓扑图的布局模式。

Private 类型

- enum LayoutMode { LAYOUT_AUTO , LAYOUT_HIERARCHICAL , LAYOUT_GROUPED }
  定义支持的布局模式。

Private 属性

- NetworkTopologyView * m_topologyView
  拓扑图视图控件
- QWidget * m_controlPanel
  控制面板控件
- LayoutMode m_layoutMode
  当前选定的布局模式
- QList< HostInfo > m_currentHosts
  缓存当前主机列表，用于布局切换时重新应用

7.8.1 详细描述

网络拓扑组件的主控件，继承自 QWidget。
包含 NetworkTopologyView (用于显示拓扑图) 和一个控制面板 (用于布局切换、缩放等操作)。

7.8.2 成员枚举类型说明

7.8.2.1 LayoutMode

enum NetworkTopology::LayoutMode [private]
定义支持的布局模式。

枚举值

LAYOUT_AUTO	自动布局
LAYOUT_HIERARCHICAL	层次化布局
LAYOUT_GROUPED	分组布局

7.8.3 构造及析构函数说明

7.8.3.1 NetworkTopology()

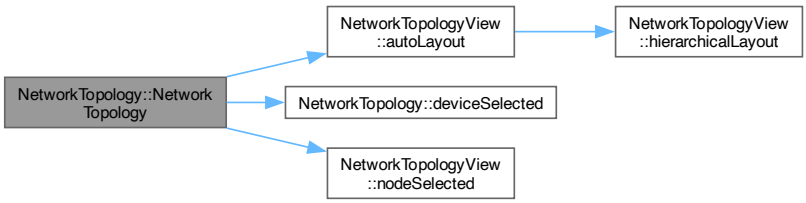
NetworkTopology::NetworkTopology (
QWidget * parent = nullptr)
NetworkTopology 构造函数。

参数

parent	父控件指针。

parent	父控件指针。

初始化拓扑图视图和控制面板，并连接相关的信号槽。函数调用图:



7.8.4 成员函数说明

7.8.4.1 clear()

void NetworkTopology::clear ()
清除当前显示的拓扑图。

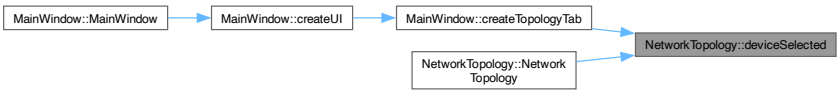
7.8.4.2 deviceSelected

void NetworkTopology::deviceSelected (
const HostInfo & host) [signal]
当用户在拓扑图中选择一个设备时发出此信号 (从 NetworkTopologyView 转发)。

参数

host	被选设备的 HostInfo。

这是这个函数的调用关系图:



7.8.4.3 resetView()

void NetworkTopology::resetView ()
重置拓扑图视图，使其适应所有节点。

7.8.4.4 scale()

void NetworkTopology::scale (
qreal factor)
缩放拓扑图视图。

参数

factor	缩放因子 (例如，1.2 表示放大，0.8 表示缩小)。

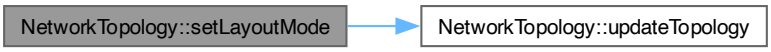
7.8.4.5 setLayoutMode()

void NetworkTopology::setLayoutMode (
int mode)
设置拓扑图的布局模式。

参数

mode	布局模式枚举值 (LayoutMode)。
mode	布局模式的整数表示 (LayoutMode 枚举值)。

如果当前已有主机数据，会调用 updateTopology 重新应用布局。函数调用图:



7.8.4.6 updateTopology()

void NetworkTopology::updateTopology (
const QList< HostInfo > & hosts)
更新拓扑图显示的主机列表。

参数

hosts	新的主机信息列表。
hosts	新的主机信息列表。

缓存主机列表，并调用 NetworkTopologyView::setHosts 更新视图。这是这个函数的调用关系图:

7.8.5 类成员变量说明

7.8.5.1 m_controlPanel

QWidget* NetworkTopology::m_controlPanel [private]
控制面板控件

7.8.5.2 m_currentHosts

QList<HostInfo> NetworkTopology::m_currentHosts [private]
缓存当前主机列表，用于布局切换时重新应用

7.8.5.3 m_layoutMode

LayoutMode NetworkTopology::m_layoutMode [private]
当前选定的布局模式

7.8.5.4 m_topologyView

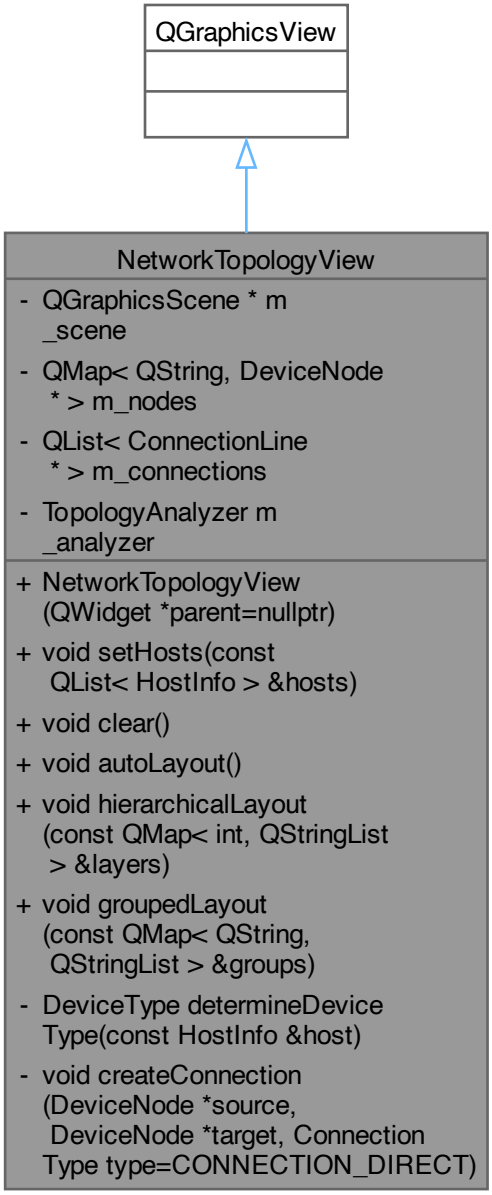
NetworkTopologyView* NetworkTopology::m_topologyView [private]
拓扑图视图控件
该类的文档由以下文件生成:

- networktopology.h
- NetScanner_autogen/JRIAJ772TK/moc_networktopology.cpp
- networktopology.cpp

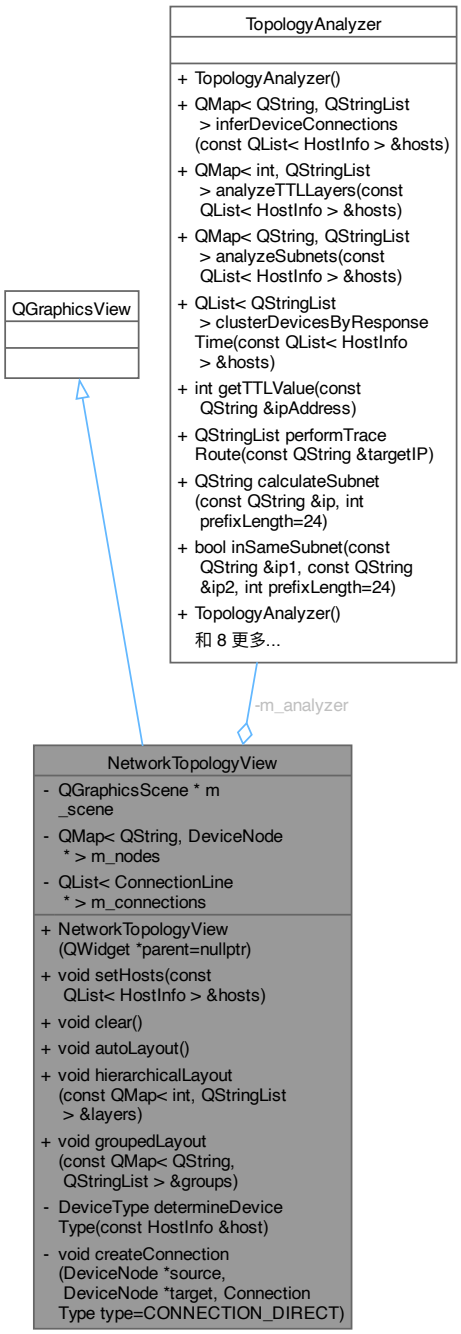
7.9 NetworkTopologyView 类参考

网络拓扑图的视图类，继承自 QGraphicsView。
#include <networktopology.h>

类 NetworkTopologyView 继承关系图:



NetworkTopologyView 的协作图:



信号

- void nodeSelected (const HostInfo &host)
  当用户在拓扑图中选择一个设备节点时发出此信号。

Public 成员函数

- NetworkTopologyView (QWidget *parent=nullptr)
  NetworkTopologyView 构造函数。

- void setHosts (const QList< HostInfo > &hosts)
  设置要在拓扑图中显示的主机列表。
- void clear ()
  清除拓扑图中的所有节点和连接。
- void autoLayout ()
  应用自动布局算法重新排列节点。
- void hierarchicalLayout (const QMap< int, QStringList > &layers)
  应用层次化布局算法。
- void groupedLayout (const QMap< QString, QStringList > &groups)
  应用分组布局算法。

Private **成员函数**

- DeviceType determineDeviceType (const HostInfo &host)
  根据主机信息推断设备类型。
- void createConnection (DeviceNode *source, DeviceNode *target, ConnectionType type=CONNECTION_DIRECT
  创建两个设备节点之间的连接线。

Private **属性**

- QGraphicsScene * m_scene
  QGraphicsScene 用于管理图形项
- QMap< QString, DeviceNode * > m_nodes
  IP 地址到DeviceNode 指针的映射
- QList< ConnectionLine * > m_connections
  存储所有连接线的列表
- TopologyAnalyzer m_analyzer
  拓扑分析器实例

7.9.1 详细描述

网络拓扑图的视图类，继承自 QGraphicsView。
负责显示 DeviceNode 和 ConnectionLine 对象，提供缩放、拖动等交互功能，并实现不同的布局算法 (如层次布局、分组布局)。

7.9.2 构造及析构函数说明

7.9.2.1 NetworkTopologyView()

NetworkTopologyView::NetworkTopologyView (
QWidget * parent = nullptr)
NetworkTopologyView 构造函数。

参数

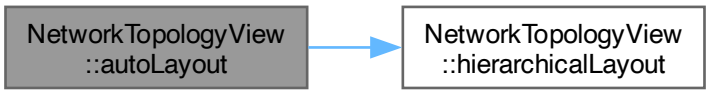
parent	父控件指针。

7.9.3 成员函数说明

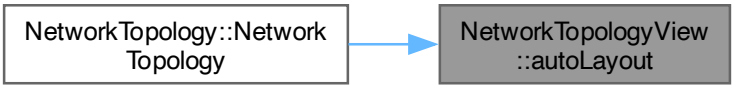
7.9.3.1 autoLayout()

void NetworkTopologyView::autoLayout ()
应用自动布局算法重新排列节点。
应用自动布局算法重新排列拓扑图中的节点。

当前实现是简化的层次布局: 查找路由器作为第 0 层，其他设备作为第 1 层，然后调用 hierarchicalLayout。如果节点数少于 2，则不执行布局。函数调用图:

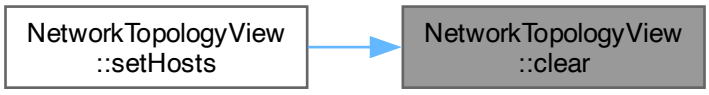


这是这个函数的调用关系图:



7.9.3.2 clear()

void NetworkTopologyView::clear ()
清除拓扑图中的所有节点和连接。
这是这个函数的调用关系图:



7.9.3.3 createConnection()

void NetworkTopologyView::createConnection (
DeviceNode * source,
DeviceNode * target,
ConnectionType type = CONNECTION_DIRECT) [private]
创建两个设备节点之间的连接线。
创建两个设备节点之间的连接线，并添加到场景中。

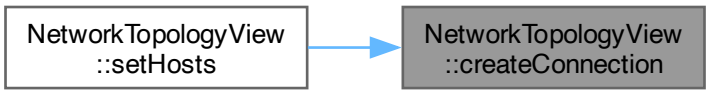
参数

source	源设备节点指针。

参数

target	目标设备节点指针。
type	连接类型，默认为 CONNECTION_DIRECT。

这是这个函数的调用关系图:



7.9.3.4 determineDeviceType()

DeviceType NetworkTopologyView::determineDeviceType (
const HostInfo & host) [private]

根据主机信息推断设备类型。

参数

host	主机信息。

返回

DeviceType 推断出的设备类型。

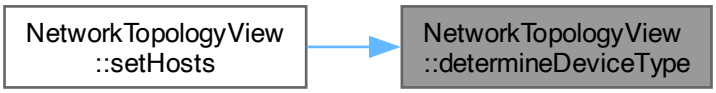
参数

host	主机信息。

返回

DeviceType 推断出的设备类型。

基于IP 地址后缀、主机名和MAC 厂商中的关键词进行推断。这是这个函数的调用关系图:



7.9.3.5 groupedLayout()

void NetworkTopologyView::groupedLayout (
const QMap< QString, QStringList > & groups)
应用分组布局算法。
应用分组布局算法到拓扑图视图。

参数

groups	一个映射表，键为组名/ID，值为该组下的设备IP 列表。
groups	一个映射表，键为组名/ID (例如子网地址)，值为该组下的设备IP 列表。

将不同组的节点分别放置在不同的区域，组内节点采用环形布局。节点放置时带有动画效果。

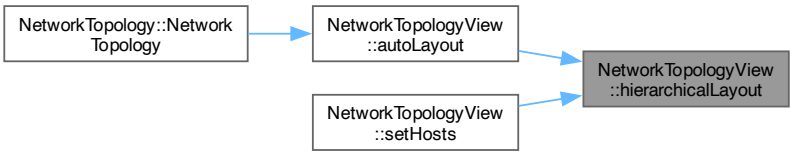
7.9.3.6 hierarchicalLayout()

void NetworkTopologyView::hierarchicalLayout (
const QMap< int, QStringList > & layers)
应用层次化布局算法。

参数

layers	一个映射表，键为层级，值为该层级下的设备IP 列表。
layers	一个映射表，键为层级，值为该层级下的设备IP 列表。

根据层级信息，将节点在垂直方向上分层排列，同层节点水平排列。节点放置时带有动画效果。这是这个函数的调用关系图:



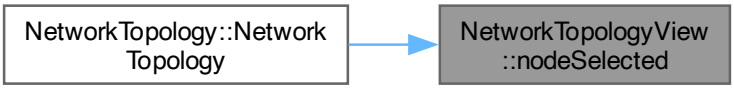
7.9.3.7 nodeSelected

void NetworkTopologyView::nodeSelected (
const HostInfo & host) [signal]
当用户在拓扑图中选择一个设备节点时发出此信号。

参数

host 被选节点的	HostInfo。

这是这个函数的调用关系图:



7.9.3.8 setHosts()

void NetworkTopologyView::setHosts (
const QList< HostInfo > & hosts)

设置要在拓扑图中显示的主机列表。

参数

hosts	主机信息列表。

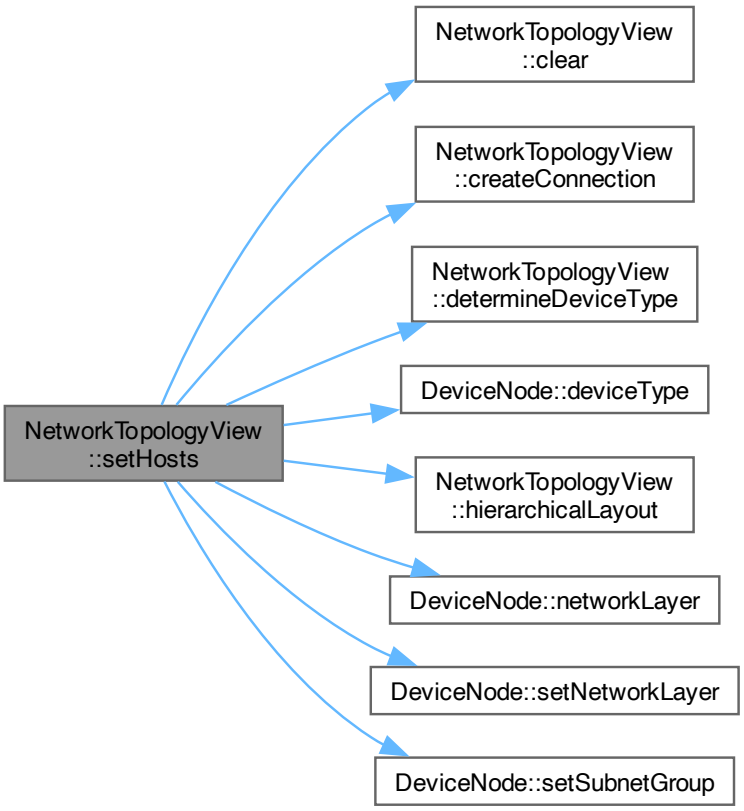
此方法会清除现有节点和连接，然后根据新的主机列表创建拓扑图。

参数

hosts	主机信息列表。

此方法会清除现有节点和连接，然后根据新的主机列表创建拓扑图。使用 TopologyAnalyzer 推断连接关

系和网络层次，并应用层次化布局。函数调用图:



7.9.4 类成员变量说明

7.9.4.1 m_analyzer

TopologyAnalyzer NetworkTopologyView::m_analyzer [private]
拓扑分析器实例

7.9.4.2 m_connections

QList<ConnectionLine*> NetworkTopologyView::m_connections [private]
存储所有连接线的列表

7.9.4.3 m_nodes

QMap<QString, DeviceNode*> NetworkTopologyView::m_nodes [private]
IP 地址到DeviceNode 指针的映射

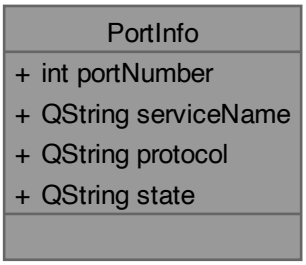
7.9.4.4 m_scene

QGraphicsScene* NetworkTopologyView::m_scene [private]
QGraphicsScene 用于管理图形项
该类的文档由以下文件生成:

- networktopology.h
- NetScanner_autogen/JRIAJ772TK/moc_networktopology.cpp
- networktopology.cpp

7.10 PortInfo 结构体参考

存储单个端口的信息。
#include <hostinfo.h>
PortInfo 的协作图:



Public 属性

- int portNumber
  端口号。
- QString serviceName
  端口上运行的服务名称 (例如 http, ftp)。
- QString protocol
  端口使用的协议 (例如 TCP, UDP)。
- QString state
  端口状态 (例如 open, closed, filtered)。

7.10.1 详细描述

存储单个端口的信息。

7.10.2 类成员变量说明

7.10.2.1 portNumber

int PortInfo::portNumber
端口号。

7.10.2.2 protocol

QString PortInfo::protocol
端口使用的协议 (例如 TCP, UDP)。

7.10.2.3 serviceName

QString PortInfo::serviceName
端口上运行的服务名称 (例如 http, ftp)。

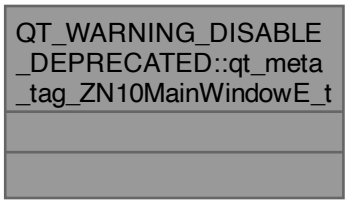
7.10.2.4 state

QString PortInfo::state
端口状态 (例如 open, closed, filtered)。
该结构体的文档由以下文件生成:

- src/data/hostinfo.h

7.11 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN10MainWindowE_t 结构体参考

QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN10MainWindowE_t 的协作图:

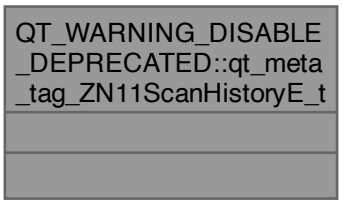


该结构体的文档由以下文件生成:

- NetScanner_autogen/EWIEGA46WW/moc_mainwindow.cpp

7.12 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN11ScanHistoryE_t 结构体参考

QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN11ScanHistoryE_t 的协作图:

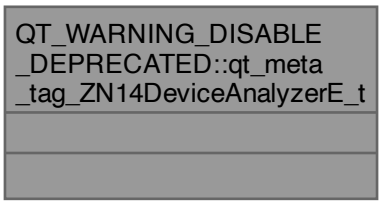


该结构体的文档由以下文件生成:

- NetScanner_autogen/JRIAJ772TK/moc_scanhistory.cpp

7.13 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14DeviceAnalyzerE_t 结构体参考

QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14DeviceAnalyzerE_t 的协作图:

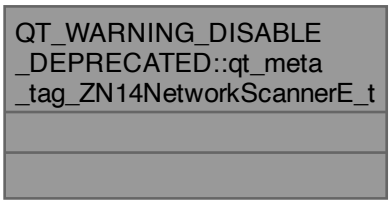


该结构体的文档由以下文件生成:

- NetScanner_autogen/EWIEGA46WW/moc_deviceanalyzer.cpp

7.14 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14NetworkScannerE_t 结构体参考

QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14NetworkScannerE_t 的协作图:

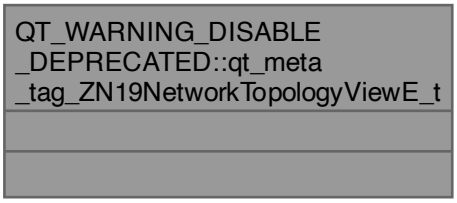


该结构体的文档由以下文件生成:

- NetScanner_autogen/JRIAJ772TK/moc_networkscanner.cpp

7.15 QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN19NetworkTopologyViewE_t 结构体参考

QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN19NetworkTopologyViewE_t 的协作图:



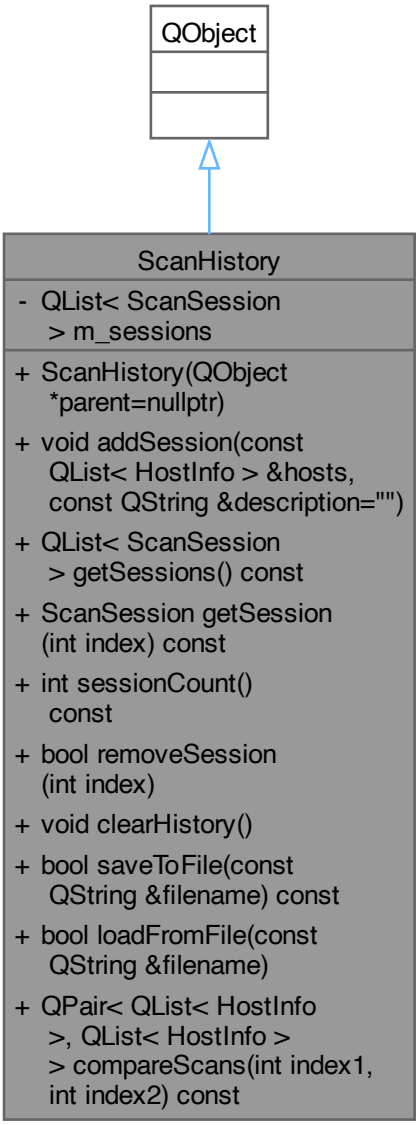
该结构体的文档由以下文件生成:

- NetScanner_autogen/JRIAJ772TK/moc_networktopology.cpp

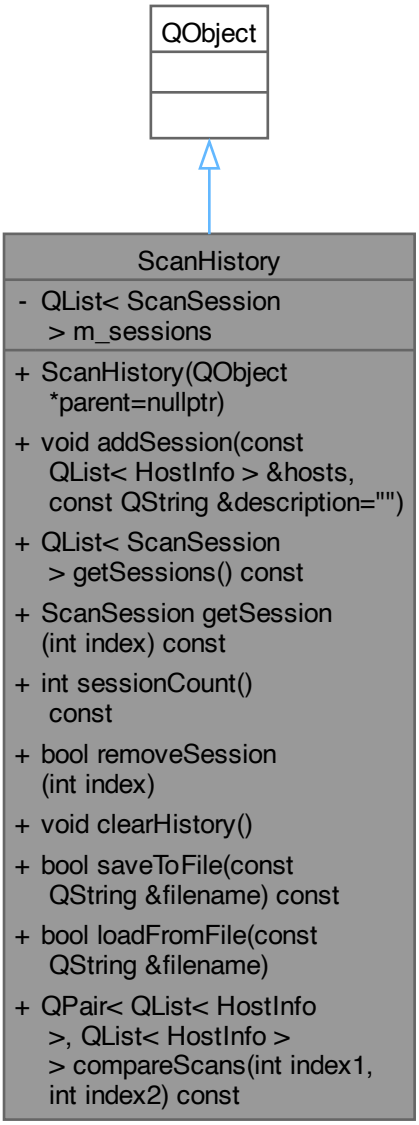
7.16 ScanHistory 类参考

扫描历史管理器类。
#include <scanhistory.h>

类 ScanHistory 继承关系图:



ScanHistory 的协作图:



信号

- void historyChanged ()

  当扫描历史记录发生变化时（例如添加、删除会话或加载历史）发出此信号。

Public 成员函数

- ScanHistory (QObject *parent=nullptr)

  ScanHistory 构造函数。
- void addSession (const QList< HostInfo > &hosts, const QString &description="")

  添加一次新的扫描会话到历史记录。
- QList< ScanSession > getSessions () const

获取所有存储的扫描会话。

- ScanSession getSession (int index) const
  根据索引获取指定的扫描会话。
- int sessionCount () const
  获取存储的扫描会话总数。
- bool removeSession (int index)
  根据索引移除一个扫描会话。
- void clearHistory ()
  清除所有扫描历史记录。
- bool saveToFile (const QString &filename) const
  将扫描历史保存到文件 (例如JSON 格式)。
- bool loadFromFile (const QString &filename)
  从文件加载扫描历史 (例如JSON 格式)。
- QPair< QList< HostInfo >, QList< HostInfo > > compareScans (int index1, int index2) const
  比较两次扫描会话的结果，找出新增和消失的主机。

Private 属性

- QList< ScanSession > m_sessions
  存储所有扫描会话的列表

7.16.1 详细描述

扫描历史管理器类。
负责存储、加载、管理和比较多次扫描会话。提供会话的增删改查功能，支持从文件导入/导出历史记录 (例如JSON 格式)，并能比较两次会话的差异。

7.16.2 构造及析构函数说明

7.16.2.1 ScanHistory()

```
ScanHistory::ScanHistory (
        QObject * parent = nullptr)   [explicit]
```

ScanHistory 构造函数。

参数

parent	父对象指针。

7.16.3 成员函数说明

7.16.3.1 addSession()

```
void ScanHistory::addSession (
        const QList< HostInfo > & hosts,
        const QString & description = "")
```

添加一次新的扫描会话到历史记录。

参数

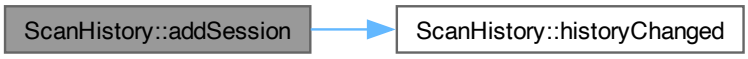
hosts	本次扫描发现的主机列表。
description	(可选) 会话的描述，默认为当前日期时间。

创建一个新的 ScanSession 对象并添加到 m_sessions 列表中。会触发 historyChanged() 信号。

参数

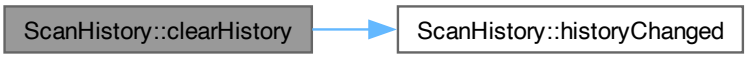
hosts	本次扫描发现的主机列表。
description	(可选) 会话的描述，默认为当前日期时间。

创建一个新的 ScanSession 对象并添加到 m_sessions 列表中。如果描述为空，则使用当前日期时间作为描述。会触发 historyChanged() 信号。函数调用图:



7.16.3.2 clearHistory()

void ScanHistory::clearHistory ()
清除所有扫描历史记录。
会触发 historyChanged() 信号。函数调用图:



7.16.3.3 compareScans()

QPair< QList< HostInfo >, QList< HostInfo > > ScanHistory::compareScans (
int index1,
int index2) const
比较两次扫描会话的结果，找出新增和消失的主机。

参数

index1	第一个会话的索引 (通常是较早的会话)。
index2	第二个会话的索引 (通常是较新的会话)。

返回

QPair<QList<HostInfo>, QList<HostInfo>>

- first: 在会话 2 中出现但未在会话 1 中出现的主机 (新增主机)。
- second: 在会话 1 中出现但未在会话 2 中出现的主机 (消失主机)。

异常

std::out_of_range	如果索引无效。

7.16.3.4 getSession()

ScanSession ScanHistory::getSession (
int index) const
根据索引获取指定的扫描会话。

参数

index	会话在列表中的索引。

返回

ScanSession 指定的扫描会话。如果索引无效，则返回一个空的默认会话。

7.16.3.5 getSessions()

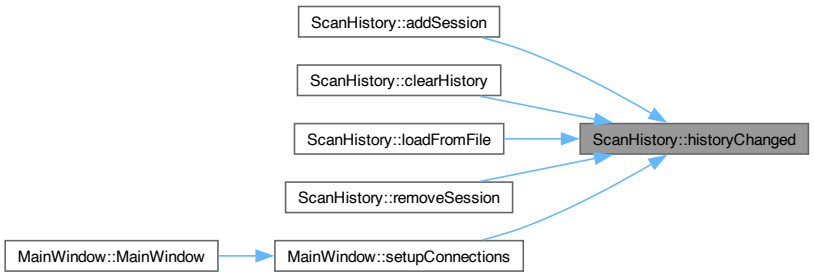
QList< ScanSession > ScanHistory::getSessions () const
获取所有存储的扫描会话。

返回

QList<ScanSession> 包含所有扫描会话的列表。

7.16.3.6 historyChanged

void ScanHistory::historyChanged () [signal]
当扫描历史记录发生变化时（例如添加、删除会话或加载历史）发出此信号。
这是这个函数的调用关系图:



7.16.3.7 loadFromFile()

bool ScanHistory::loadFromFile (
const QString & filename)
从文件加载扫描历史（例如JSON 格式）。
从文件加载扫描历史（JSON 格式）。

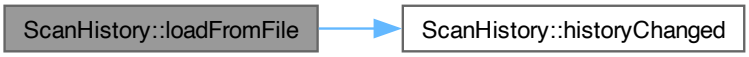
参数

filename	要加载的文件名。

返回

如果加载成功则返回 true，否则返回 false。

加载成功后会触发 historyChanged() 信号。函数调用图:



7.16.3.8 removeSession()

bool ScanHistory::removeSession (
int index)

根据索引移除一个扫描会话。

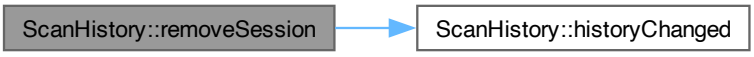
参数

index	要移除的会话的索引。

返回

如果成功移除则返回 true，否则 (例如索引无效) 返回 false。

会触发 historyChanged() 信号。函数调用图:



7.16.3.9 saveToFile()

bool ScanHistory::saveToFile (
const QString & filename) const

将扫描历史保存到文件 (例如JSON 格式)。
将扫描历史保存到文件 (JSON 格式)。

参数

filename	要保存的文件名。

返回

如果保存成功则返回 true，否则返回 false。

7.16.3.10 sessionCount()

int ScanHistory::sessionCount () const
获取存储的扫描会话总数。

返回

int 会话数量。

7.16.4 类成员变量说明

7.16.4.1 m_sessions

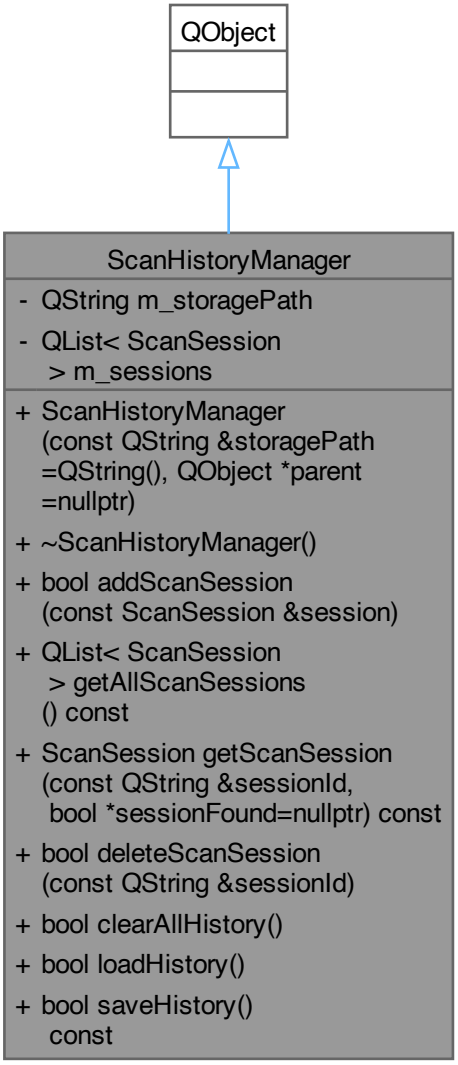
QList<ScanSession> ScanHistory::m_sessions [private]
存储所有扫描会话的列表
该类的文档由以下文件生成:

- scanhistory.h
- NetScanner_autogen/JRIAJ772TK/moc_scanhistory.cpp
- scanhistory.cpp

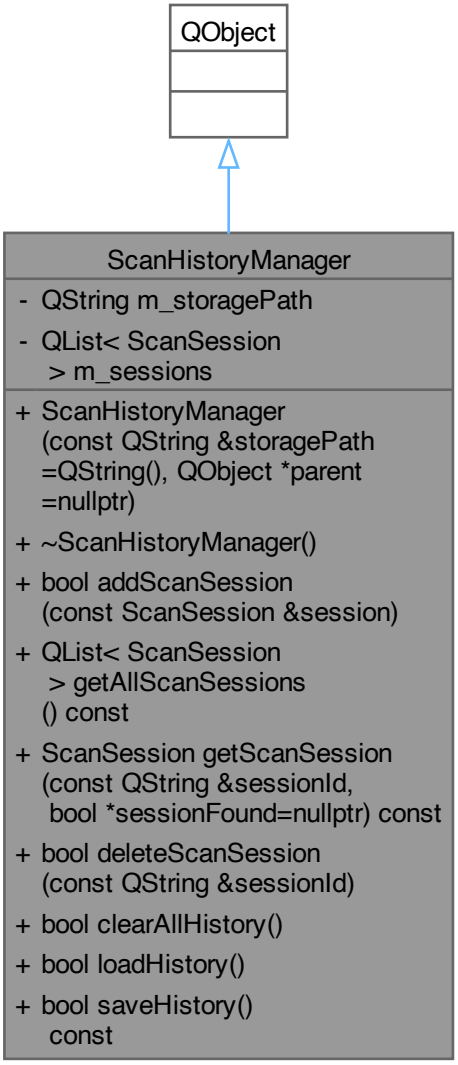
7.17 ScanHistoryManager 类参考

管理扫描历史记录的类。
#include <scanhistory.h>

类 ScanHistoryManager 继承关系图:



ScanHistoryManager 的协作图:



信号

- void historyChanged ()

  当历史记录发生更改时（例如添加或删除会话）发射此信号。

Public 成员函数

- ScanHistoryManager (const QString &storagePath=QString(), QObject *parent=nullptr)

  ScanHistoryManager 构造函数。
- ~ScanHistoryManager ()

  ScanHistoryManager 析构函数。
- bool addScanSession (const ScanSession &session)

  添加一次新的扫描会话到历史记录中。
- QList< ScanSession > getAllScanSessions () const

获取所有已保存的扫描会话。

- ScanSession getScanSession (const QString &sessionId, bool *sessionFound=nullptr) const

  根据会话ID 获取特定的扫描会话。
- bool deleteScanSession (const QString &sessionId)

  删除指定ID 的扫描会话。
- bool clearAllHistory ()

  清除所有的扫描历史记录。
- bool loadHistory ()

  加载历史记录。
- bool saveHistory () const

  保存当前内存中的历史记录到持久化存储。

Private 属性

- QString m_storagePath

  历史记录文件的路径或数据库连接信息。
- QList< ScanSession > m_sessions

  内存中缓存的扫描会话列表。

7.17.1 详细描述

管理扫描历史记录的类。
负责加载、保存、查询和删除扫描会话。历史记录可以存储在文件（如JSON, XML, SQLite）或数据库中。

7.17.2 构造及析构函数说明

7.17.2.1 ScanHistoryManager()

```
ScanHistoryManager::ScanHistoryManager (
            const QString & storagePath = QString(),
            QObject * parent = nullptr)   [explicit]
```

ScanHistoryManager 构造函数。

参数

storagePath	历史记录的存储路径或数据库连接字符串。默认为空，可能使用默认位置。
parent	父QObject，默认为 nullptr。

7.17.2.2 ~ScanHistoryManager()

```
ScanHistoryManager::~ScanHistoryManager ()
```

ScanHistoryManager 析构函数。
确保在对象销毁前所有挂起的历史记录更改都已保存。

7.17.3 成员函数说明

7.17.3.1 addScanSession()

```
bool ScanHistoryManager::addScanSession (
            const ScanSession & session)
```

添加一次新的扫描会话到历史记录中。

参数

session	要添加的扫描会话数据。

返回

如果成功添加则返回 true，否则返回 false (例如，存储失败)。

7.17.3.2 clearAllHistory()

bool ScanHistoryManager::clearAllHistory ()
清除所有的扫描历史记录。

返回

如果成功清除所有记录则返回 true，否则返回 false。

7.17.3.3 deleteScanSession()

bool ScanHistoryManager::deleteScanSession (
const QString & sessionId)
删除指定ID 的扫描会话。

参数

sessionId	要删除的扫描会话的ID。

返回

如果成功删除则返回 true，否则返回 false。

7.17.3.4 getAllScanSessions()

QList< ScanSession > ScanHistoryManager::getAllScanSessions () const
获取所有已保存的扫描会话。

返回

ScanSession 对象的列表，按时间倒序排列 (最新的在前)。

7.17.3.5 getScanSession()

ScanSession ScanHistoryManager::getScanSession (
const QString & sessionId,
bool * sessionFound = nullptr) const
根据会话ID 获取特定的扫描会话。

参数

sessionId	要检索的扫描会话的ID。
sessionFound	[out] 如果找到会话，此引用将被设置为 true。

返回

如果找到，则返回 ScanSession 对象；否则返回一个空的或无效的 ScanSession。

7.17.3.6 historyChanged

void ScanHistoryManager::historyChanged () [signal]
当历史记录发生更改时（例如添加或删除会话）发射此信号。

7.17.3.7 loadHistory()

bool ScanHistoryManager::loadHistory ()
加载历史记录。
从指定的存储位置（文件或数据库）加载历史记录到内存中。通常在构造时或显式调用时执行。

返回

如果成功加载则返回 true，否则返回 false。

7.17.3.8 saveHistory()

bool ScanHistoryManager::saveHistory () const
保存当前内存中的历史记录到持久化存储。
可以在每次更改后自动调用，或在程序退出前显式调用。

返回

如果成功保存则返回 true，否则返回 false。

7.17.4 类成员变量说明

7.17.4.1 m_sessions

QList<ScanSession> ScanHistoryManager::m_sessions [private]
内存中缓存的扫描会话列表。

7.17.4.2 m_storagePath

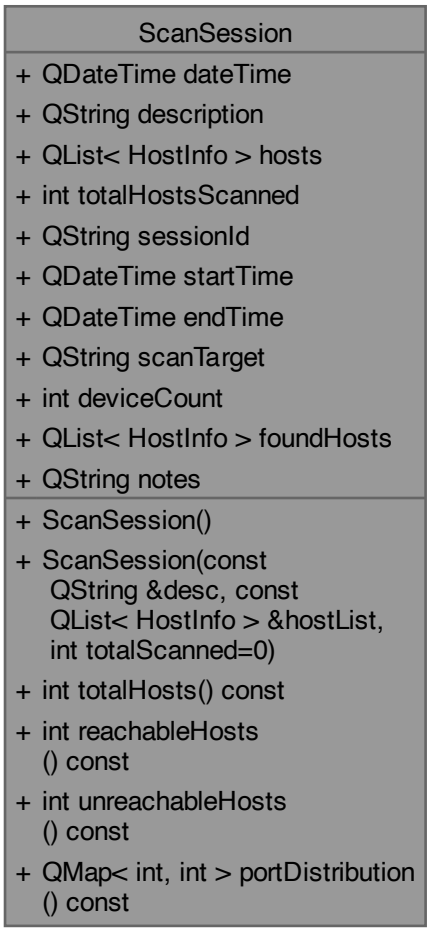
QString ScanHistoryManager::m_storagePath [private]
历史记录文件的路径或数据库连接信息。
该类的文档由以下文件生成:

- src/data/scanhistory.h

7.18 ScanSession 结构体参考

存储单次扫描会话的信息。
#include <scanhistory.h>

ScanSession 的协作图:



Public 成员函数

- ScanSession ()

  ScanSession 默认构造函数。
- ScanSession (const QString &desc, const QList< HostInfo > &hostList, int totalScanned=0)

  ScanSession 构造函数。
- int totalHosts () const

  获取本次会话中发现的主机总数。
- int reachableHosts () const

  计算并返回会话中可达主机的数量。
- int unreachableHosts () const
- QMap< int, int > portDistribution () const

  计算并返回会话中各开放端口的分布情况。

Public 属性

- QDateTime dateTime

  扫描会话的日期和时间

- QString description
  会话描述（例如，”家庭网络扫描”）
- QList< HostInfo > hosts
  本次扫描发现的主机列表
- int totalHostsScanned
  本次扫描尝试扫描的主机总数（可选）
- QString sessionId
  本次扫描会话的唯一ID（例如 UUID 或时间戳字符串）。
- QDateTime startTime
  扫描开始时间。
- QDateTime endTime
  扫描结束时间。
- QString scanTarget
  本次扫描的目标（例如”192.168.1.0/24”）。
- int deviceCount
  本次扫描发现的设备数量。
- QList< HostInfo > foundHosts
  本次扫描发现的所有主机的详细信息。
- QString notes
  用户为本次扫描添加的备注（可选）。

7.18.1 详细描述

存储单次扫描会话的信息。
存储一次完整的扫描会话信息。
包含扫描的日期时间、描述、主机总数以及该会话中发现的所有主机信息。

7.18.2 构造及析构函数说明

7.18.2.1 ScanSession() [1/2]

ScanSession::ScanSession () [inline]
ScanSession 默认构造函数。
初始化 dateTime 为当前时间，description 为空，totalHostsScanned 为 0。

7.18.2.2 ScanSession() [2/2]

```
ScanSession::ScanSession (
        const QString & desc,
        const QList< HostInfo > & hostList,
        int totalScanned = 0)  [inline]
```

ScanSession 构造函数。

参数

desc	会话描述。
hostList	本次扫描发现的主机列表。
totalScanned	（可选）本次扫描尝试扫描的总主机数。

7.18.3 成员函数说明

7.18.3.1 portDistribution()

QMap< int, int > ScanSession::portDistribution () const
计算并返回会话中各开放端口的分布情况。

返回

QMap<int, int> 键为端口号，值为使用该端口的主机数量。

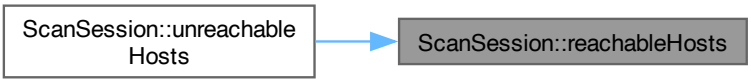
7.18.3.2 reachableHosts()

int ScanSession::reachableHosts () const
计算并返回会话中可达主机的数量。

返回

int 可达主机的数量。

这是这个函数的调用关系图:



7.18.3.3 totalHosts()

int ScanSession::totalHosts () const [inline]
获取本次会话中发现的主机总数。

返回

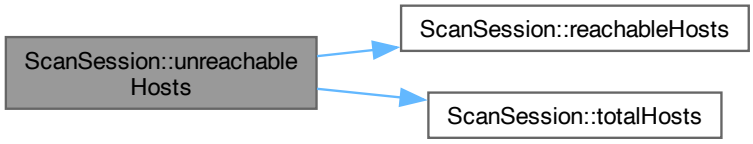
int 主机数量。

这是这个函数的调用关系图:

7.18.3.4 unreachableHosts()

int ScanSession::unreachableHosts () const [inline]

函数调用图:



7.18.4 类成员变量说明

7.18.4.1 dateTime

QDateTime ScanSession::dateTime
扫描会话的日期和时间

7.18.4.2 description

QString ScanSession::description
会话描述 (例如，”家庭网络扫描”)

7.18.4.3 deviceCount

int ScanSession::deviceCount
本次扫描发现的设备数量。

7.18.4.4 endTime

QDateTime ScanSession::endTime
扫描结束时间。

7.18.4.5 foundHosts

QList<HostInfo> ScanSession::foundHosts
本次扫描发现的所有主机的详细信息。

7.18.4.6 hosts

QList<HostInfo> ScanSession::hosts
本次扫描发现的主机列表

7.18.4.7 notes

QString ScanSession::notes
用户为本次扫描添加的备注 (可选)。

7.18.4.8 scanTarget

QString ScanSession::scanTarget
本次扫描的目标 (例如”192.168.1.0/24”)。

7.18.4.9 sessionId

QString ScanSession::sessionId
本次扫描会话的唯一ID (例如 UUID 或时间戳字符串)。

7.18.4.10 startTime

QDateTime ScanSession::startTime
扫描开始时间。

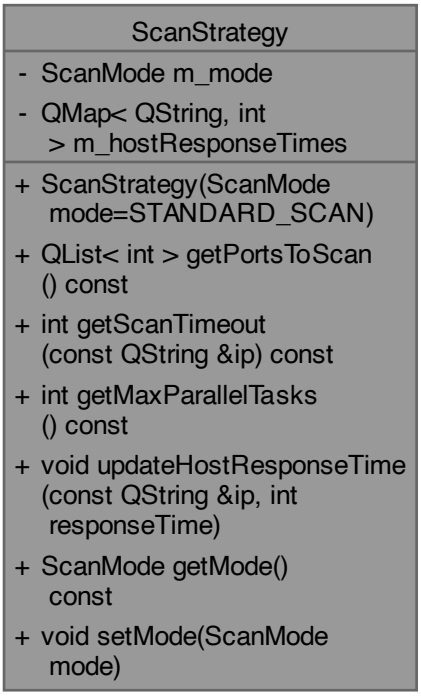
7.18.4.11 totalHostsScanned

int ScanSession::totalHostsScanned
本次扫描尝试扫描的主机总数 (可选)
该结构体的文档由以下文件生成:

- scanhistory.h
- src/data/scanhistory.h
- scanhistory.cpp

7.19 ScanStrategy 类参考

扫描策略类
#include <networkscanner.h>
ScanStrategy 的协作图:



Public 类型

- enum ScanMode { QUICK_SCAN , STANDARD_SCAN , DEEP_SCAN }
  扫描模式枚举

Public 成员函数

- ScanStrategy (ScanMode mode=STANDARD_SCAN)

构造函数

- QList< int > getPortsToScan () const
  获取要扫描的端口列表
- int getScanTimeout (const QString &ip) const
  获取扫描超时时间
- int getMaxParallelTasks () const
  获取最大并行任务数
- void updateHostResponseTime (const QString &ip, int responseTime)
  更新主机响应时间记录
- ScanMode getMode () const
  获取当前扫描模式
- void setMode (ScanMode mode)
  设置扫描模式

Private 属性

- ScanMode m_mode
  当前扫描模式
- QMap< QString, int > m_hostResponseTimes
  IP 地址 -> 响应时间映射

7.19.1 详细描述

扫描策略类
定义不同的扫描模式和参数，如快速扫描、标准扫描和深度扫描

7.19.2 成员枚举类型说明

7.19.2.1 ScanMode

enum ScanStrategy::ScanMode
扫描模式枚举

枚举值

QUICK_SCAN	仅检测主机存活
STANDARD_SCAN	扫描常用端口
DEEP_SCAN	全面端口扫描

7.19.3 构造及析构函数说明

7.19.3.1 ScanStrategy()

ScanStrategy::ScanStrategy (
ScanMode mode = STANDARD_SCAN)
构造函数
ScanStrategy 构造函数。

参数

mode	扫描模式，默认为标准扫描
mode	扫描模式，默认为 STANDARD_SCAN。

7.19.4 成员函数说明

7.19.4.1 getMaxParallelTasks()

int ScanStrategy::getMaxParallelTasks () const
获取最大并行任务数
根据当前扫描模式获取推荐的最大并行任务数。

返回

并行任务数量
并行任务数量。

基于 QThread::idealThreadCount() (CPU 核心数) 计算:

- QUICK_SCAN: 返回核心数的 2 倍。
- STANDARD_SCAN: 返回核心数。
- DEEP_SCAN: 返回核心数的一半 (至少为 1)。

7.19.4.2 getMode()

ScanMode ScanStrategy::getMode () const [inline]
获取当前扫描模式

返回

扫描模式

7.19.4.3 getPortsToScan()

QList< int > ScanStrategy::getPortsToScan () const
获取要扫描的端口列表
根据当前扫描模式获取推荐扫描的端口列表。

返回

端口号列表
QList<int> 包含端口号的列表。

- QUICK_SCAN: 返回少量最常用端口 (如 80, 443, 22, 3389)。
  - STANDARD_SCAN: 返回一组常用服务端口。
  - DEEP_SCAN: 返回更广泛的端口列表，包括不常用端口和特定服务端口。

7.19.4.4 getScanTimeout()

int ScanStrategy::getScanTimeout (
const QString & ip) const
获取扫描超时时间
根据IP 地址和当前扫描模式获取推荐的扫描超时时间。

参数

ip	目标IP 地址

返回

超时时间（毫秒）

参数

ip	目标IP 地址字符串。

返回

超时时间（毫秒）。

如果该IP 有历史响应时间记录 (m_hostResponseTimes)，则基于历史时间计算超时 (乘以 2，上限 5000ms)。否则，根据扫描模式返回默认超时：QUICK_SCAN (200ms), STANDARD_SCAN (500ms), DEEP_SCAN (1000ms)。

7.19.4.5 setMode()

void ScanStrategy::setMode (
ScanMode mode) [inline]

设置扫描模式

参数

mode	要设置的扫描模式

7.19.4.6 updateHostResponseTime()

void ScanStrategy::updateHostResponseTime (
const QString & ip,
int responseTime)

更新主机响应时间记录
更新指定IP 地址的历史响应时间记录。

参数

ip	主机IP 地址
responseTime	响应时间（毫秒）
ip	主机IP 地址字符串。
responseTime	新的响应时间（毫秒）。

如果该IP 已有历史记录，则使用加权平均 (旧时间权重 0.8，新时间权重 0.2) 平滑更新。如果响应时间小于等于 0，则不更新。

7.19.5 类成员变量说明

7.19.5.1 m_hostResponseTimes

QMap<QString, int> ScanStrategy::m_hostResponseTimes [private]

IP 地址 -> 响应时间映射

7.19.5.2 m_mode

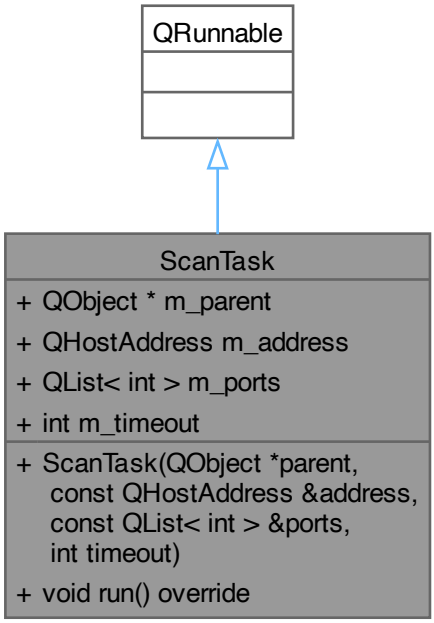
ScanMode ScanStrategy::m_mode [private]

当前扫描模式
该类的文档由以下文件生成:

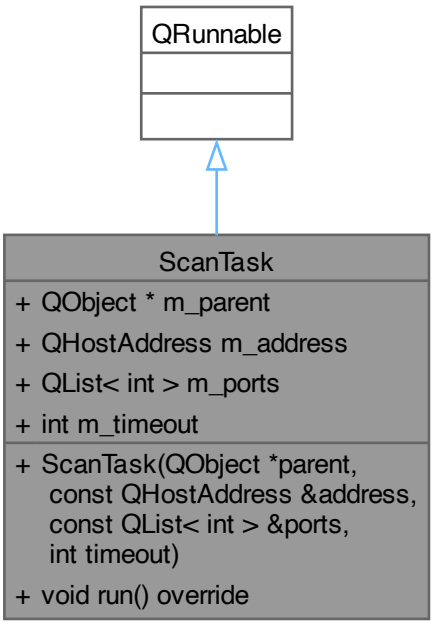
- networkscanner.h
- networkscanner.cpp

7.20 ScanTask 类参考

扫描任务类
#include <networkscanner.h>
类 ScanTask 继承关系图:



ScanTask 的协作图:



Public 成员函数

- ScanTask (QObject *parent, const QHostAddress &address, const QList< int > &ports, int timeout)

  构造函数
- void run () override

  执行扫描任务

Public 属性

- QObject * m_parent

  父对象指针 (NetworkScanner)
- QHostAddress m_address

  扫描目标IP 地址
- QList< int > m_ports

  待扫描的端口列表
- int m_timeout

  单个端口连接超时时间 (毫秒)

7.20.1 详细描述

扫描任务类
用于并行执行的单个主机扫描任务, 继承自 QRunnable 以便在线程池中运行。

7.20.2 构造及析构函数说明

7.20.2.1 ScanTask()

```
ScanTask::ScanTask (
                QObject * parent,
                const QHostAddress & address,
                const QList< int > & ports,
                int timeout)
```

构造函数
ScanTask 构造函数。

参数

parent	父对象（通常是 NetworkScanner 实例）
address	要扫描的地址
ports	要扫描的端口列表
timeout	连接超时时间（毫秒）
parent	父对象指针，通常是 NetworkScanner 实例。
address	要扫描的 QHostAddress。
ports	要扫描的端口列表。
timeout	单个端口连接的超时时间（毫秒）。

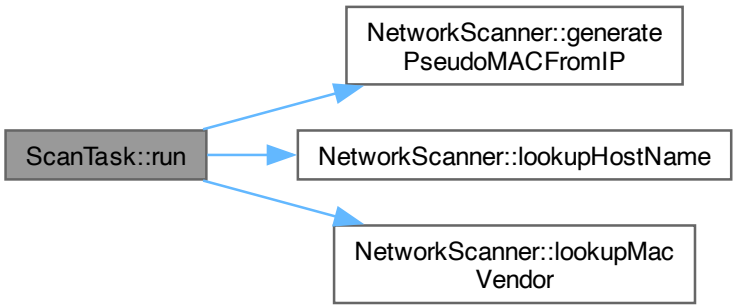
初始化扫描任务，并设置 QRunnable::setAutoDelete(true) 以便在任务完成后自动删除。

7.20.3 成员函数说明

7.20.3.1 run()

```
void ScanTask::run ()  [override]
```

执行扫描任务
执行单个主机的扫描任务，在线程池中运行。
实现QRunnable 的抽象方法，在此方法中执行实际的扫描逻辑。
此方法是 QRunnable::run() 的实现。它会创建一个 HostInfo 结构体，然后尝试通过连接指定端口（列表中的前 3 个）来判断主机是否可达。如果主机可达，会进一步尝试获取主机名和生成伪MAC 地址。最后，通过 Qt::QueuedConnection 调用父对象（NetworkScanner）的 onScanTaskFinished() 槽函数来报告结果。函数调用图:



7.20.4 类成员变量说明

7.20.4.1 m_address

QHostAddress ScanTask::m_address
扫描目标IP 地址

7.20.4.2 m_parent

QObject* ScanTask::m_parent
父对象指针 (NetworkScanner)

7.20.4.3 m_ports

QList<int> ScanTask::m_ports
待扫描的端口列表

7.20.4.4 m_timeout

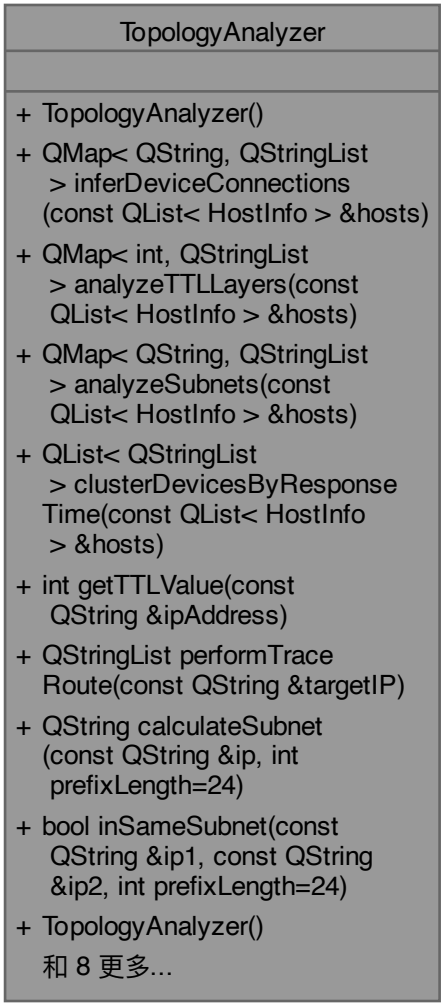
int ScanTask::m_timeout
单个端口连接超时时间 (毫秒)
该类的文档由以下文件生成:

- networkscanner.h
- networkscanner.cpp

7.21 TopologyAnalyzer 类参考

网络拓扑分析器类。
#include <networktopology.h>

TopologyAnalyzer 的协作图:



Public 成员函数

- TopologyAnalyzer ()
  TopologyAnalyzer 构造函数。
- QMap< QString, QStringList > inferDeviceConnections (const QList< HostInfo > &hosts)
  根据主机信息推断设备之间的连接关系。
- QMap< int, QStringList > analyzeTTLLayers (const QList< HostInfo > &hosts)
  分析网络中设备的TTL（Time To Live）层次结构。
- QMap< QString, QStringList > analyzeSubnets (const QList< HostInfo > &hosts)
  分析网络中的子网结构。
- QList< QStringList > clusterDevicesByResponseTime (const QList< HostInfo > &hosts)
  基于设备响应时间对设备进行聚类。
- int getTTLValue (const QString &ipAddress)
  获取特定IP 地址的TTL 值。
- QStringList performTraceRoute (const QString &targetIP)

执行 traceroute 命令获取到目标IP 的路径信息。
- QString calculateSubnet (const QString &ip, int prefixLength=24)
  计算给定IP 地址所属的子网地址。
- bool inSameSubnet (const QString &ip1, const QString &ip2, int prefixLength=24)
  判断两个IP 地址是否在同一个子网中。
- TopologyAnalyzer ()
  TopologyAnalyzer 构造函数。
- QMap< QString, QStringList > inferDeviceConnections (const QList< HostInfo > &hosts)
  推断设备之间的连接关系。
- QMap< int, QStringList > analyzeTTLLayers (const QList< HostInfo > &hosts)
  分析网络的层次结构，通常基于TTL 值。
- QMap< QString, QStringList > analyzeSubnets (const QList< HostInfo > &hosts)
  分析网络中的子网结构。
- QList< QStringList > clusterDevicesByResponseTime (const QList< HostInfo > &hosts)
  基于设备响应时间对设备进行聚类。
- int getTTLValue (const QString &ipAddress)
  获取特定IP 地址的TTL（Time To Live）值。
- QStringList performTraceRoute (const QString &targetIP)
  执行 traceroute 命令以获取到目标IP 的路径信息。
- QString calculateSubnet (const QString &ip, int prefixLength=24)
  根据IP 地址和前缀长度计算子网掩码和网络地址。
- bool inSameSubnet (const QString &ip1, const QString &ip2, int prefixLength=24)
  判断两个IP 地址是否位于同一个子网中。

7.21.1 详细描述

网络拓扑分析器类。
负责分析原始主机数据，推断设备连接、网络层次和子网结构。
提供一系列方法来分析主机列表，推断设备间的连接关系、网络层次结构、子网结构，并进行设备聚类。
此类提供了一系列方法来处理HostInfo 列表，并从中提取网络拓扑相关的结构化信息。

7.21.2 构造及析构函数说明

7.21.2.1 TopologyAnalyzer() [1/2]

TopologyAnalyzer::TopologyAnalyzer ()
TopologyAnalyzer 构造函数。

7.21.2.2 TopologyAnalyzer() [2/2]

TopologyAnalyzer::TopologyAnalyzer ()
TopologyAnalyzer 构造函数。

7.21.3 成员函数说明

7.21.3.1 analyzeSubnets() [1/2]

QMap< QString, QStringList > TopologyAnalyzer::analyzeSubnets (
const QList< HostInfo > & hosts)
分析网络中的子网结构。

参数

hosts	主机信息列表。

返回

一个映射表，键为子网地址 (如”192.168.1.0”)，值为该子网下的设备IP 列表。

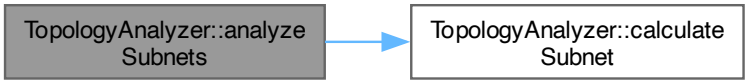
参数

hosts	主机信息列表。

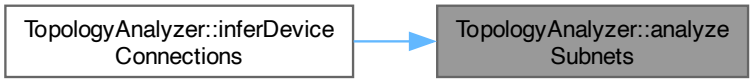
返回

一个映射表，键为子网地址 (如”192.168.1.0”)，值为该子网下的设备IP 列表。

使用 calculateSubnet 方法为每个主机计算其所属子网。函数调用图:



这是这个函数的调用关系图:



7.21.3.2 analyzeSubnets() [2/2]

QMap< QString, QStringList > TopologyAnalyzer::analyzeSubnets (
const QList< HostInfo > & hosts)

分析网络中的子网结构。

参数

hosts	扫描到的主机信息列表。

返回

返回一个映射，键是子网ID（例如”192.168.1.0/24”），值是该子网下的设备IP 列表。

7.21.3.3 analyzeTTLLayers() [1/2]

QMap< int, QStringList > TopologyAnalyzer::analyzeTTLLayers (
const QList< HostInfo > & hosts)

分析网络中设备的TTL（Time To Live）层次结构。

参数

hosts	主机信息列表。

返回

一个映射表，键为TTL 层级（整数），值为该层级下的设备IP 列表。

通常网关在第 0 层，直连设备在第 1 层，依此类推。TTL 值通过 ping 获取。

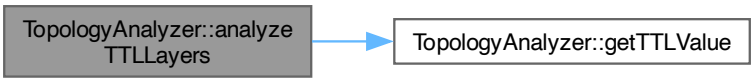
参数

hosts	主机信息列表。

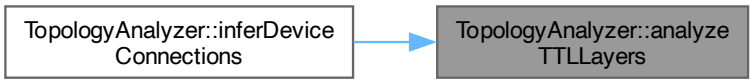
返回

一个映射表，键为TTL 层级（整数），值为该层级下的设备IP 列表。

通常网关在第 0 层，直连设备在第 1 层，依此类推。TTL 值通过 ping 获取。如果无法获取TTL，设备默认放在第 1 层。函数调用图:



这是这个函数的调用关系图:



7.21.3.4 analyzeTTLLayers() [2/2]

QMap< int, QStringList > TopologyAnalyzer::analyzeTTLLayers (
const QList< HostInfo > & hosts)

分析网络的层次结构，通常基于TTL 值。

参数

hosts	扫描到的主机信息列表。

返回

返回一个映射，键是网络层级（例如，基于TTL 的跳数），值是该层级下的设备IP 列表。

7.21.3.5 calculateSubnet() [1/2]

QString TopologyAnalyzer::calculateSubnet (
const QString & ip,
int prefixLength = 24)

计算给定IP 地址所属的子网地址。

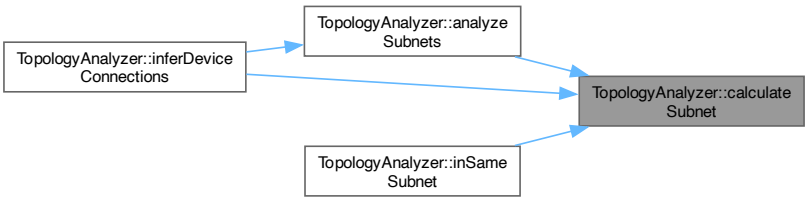
参数

ip	IP 地址字符串。
prefixLength	子网前缀长度 (例如，对于 255.255.255.0，为 24)。默认为 24。

返回

子网地址字符串 (例如”192.168.1.0”)。如果IP 格式无效则返回空字符串。

这是这个函数的调用关系图:



7.21.3.6 calculateSubnet() [2/2]

QString TopologyAnalyzer::calculateSubnet (
const QString & ip,
int prefixLength = 24)

根据IP 地址和前缀长度计算子网掩码和网络地址。

参数

ip	IP 地址字符串 (例如”192.168.1.10”)。
prefixLength	子网前缀长度 (例如 24 for /24)。

返回

返回子网的网络地址字符串 (例如”192.168.1.0”)。

7.21.3.7 clusterDevicesByResponseTime() [1/2]

QList< QStringList > TopologyAnalyzer::clusterDevicesByResponseTime (
const QList< HostInfo > & hosts)

基于设备响应时间对设备进行聚类。

参数

hosts	主机信息列表。

返回

一个包含多个设备IP 列表的列表，每个内部列表代表一个聚类。

例如，可以分为本地设备、近端设备、远端设备等。

参数

hosts	主机信息列表。

返回

一个包含多个设备IP 列表的列表，每个内部列表代表一个聚类。

例如，可以分为本地设备 (0-10ms), 近端设备 (10-50ms), 远端设备 (>50ms)。响应时间通过执行 ping 命令获取。

7.21.3.8 clusterDevicesByResponseTime() [2/2]

```
QList< QStringList > TopologyAnalyzer::clusterDevicesByResponseTime (
        const QList< HostInfo > & hosts)
```

基于设备响应时间对设备进行聚类。

参数

hosts	扫描到的主机信息列表。

返回

返回一个列表，其中每个子列表代表一个设备集群（集群内设备响应时间相近）。

7.21.3.9 getTTLValue() [1/2]

```
int TopologyAnalyzer::getTTLValue (
        const QString & ipAddress)
```

获取特定IP 地址的TTL 值。

参数

ipAddress	目标IP 地址字符串。

返回

TTL 值。如果无法获取，则返回-1。根据操作系统执行 ping 命令解析TTL。

参数

ipAddress	目标IP 地址字符串。

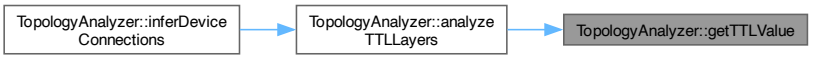
返回

TTL 值。如果无法获取，则返回-1。根据操作系统执行 ping 命令解析TTL。

注解

根据不同操作系统的 ping 命令输出格式解析TTL。

这是这个函数的调用关系图:



7.21.3.10 getTTLValue() [2/2]

int TopologyAnalyzer::getTTLValue (
const QString & ipAddress)

获取特定IP 地址的TTL（Time To Live）值。
此方法可能通过 ping 命令或预存的HostInfo 数据获取TTL。

参数

ipAddress	目标设备的IP 地址。

返回

成功则返回TTL 值，失败或无法获取则返回一个特殊值（如-1）。

7.21.3.11 inferDeviceConnections() [1/2]

QMap< QString, QStringList > TopologyAnalyzer::inferDeviceConnections (
const QList< HostInfo > & hosts)

根据主机信息推断设备之间的连接关系。

参数

hosts	包含网络中所有主机信息的列表。

返回

一个映射表，键为源设备IP，值为目标设备IP 列表，表示它们之间的连接。

连接关系可能基于网关信息、TTL 层级和子网分析。

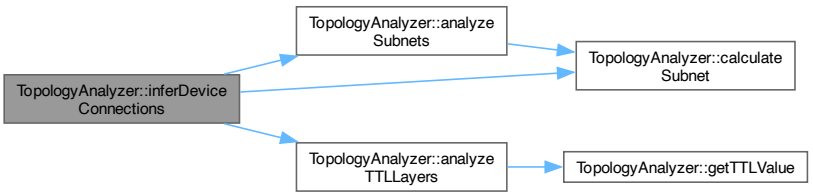
参数

hosts	包含网络中所有主机信息的列表。

返回

一个映射表，键为源设备IP，值为目标设备IP 列表，表示它们之间的连接。

连接关系可能基于网关信息、TTL 层级和子网分析。优先查找网关设备作为连接中心，然后分析TTL 层次和子网结构来构建连接。函数调用图:



7.21.3.12 inferDeviceConnections() [2/2]

QMap< QString, QStringList > TopologyAnalyzer::inferDeviceConnections (
const QList< HostInfo > & hosts)

推断设备之间的连接关系。

参数

hosts	扫描到的主机信息列表。

返回

返回一个映射，键是设备IP，值是与其有直接或间接连接的设备IP 列表。连接关系可能基于 traceroute 结果、TTL 差异等。

7.21.3.13 inSameSubnet() [1/2]

bool TopologyAnalyzer::inSameSubnet (
const QString & ip1,
const QString & ip2,
int prefixLength = 24)

判断两个IP 地址是否在同一个子网中。

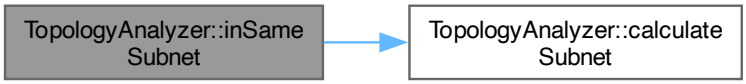
参数

ip1	第一个IP 地址字符串。
ip2	第二个IP 地址字符串。
prefixLength	子网前缀长度。默认为 24。

返回

如果在同一子网则返回 true，否则返回 false。

函数调用图:



7.21.3.14 inSameSubnet() [2/2]

```
bool TopologyAnalyzer::inSameSubnet (
                const QString & ip1,
                const QString & ip2,
                int prefixLength = 24)
```

判断两个IP 地址是否位于同一个子网中。

参数

ip1	第一个IP 地址。
ip2	第二个IP 地址。
prefixLength	子网前缀长度，默认为 24。

返回

如果两个IP 在同一个指定前缀长度的子网中，则返回 true；否则返回 false。

7.21.3.15 performTraceRoute() [1/2]

```
QStringList TopologyAnalyzer::performTraceRoute (
                const QString & targetIP)
```

执行 traceroute 命令获取到目标IP 的路径信息。

参数

targetIP	目标IP 地址字符串。

返回

一个包含路径上各跳IP 地址的字符串列表。

根据操作系统执行 traceroute 或 tracert 命令。

参数

targetIP	目标IP 地址字符串。

返回

一个包含路径上各跳IP 地址的字符串列表。

根据操作系统执行 traceroute 或 tracert 命令，并解析输出。

7.21.3.16 performTraceRoute() [2/2]

QStringList TopologyAnalyzer::performTraceRoute (
const QString & targetIP)

执行 traceroute 命令以获取到目标IP 的路径信息。
注意：执行外部命令可能需要适当的权限，并且耗时较长。

参数

targetIP	目标设备的IP 地址。

返回

返回一个字符串列表，包含 traceroute 路径上的每一跳IP 地址。如果 traceroute 失败或目标不可达，可能返回空列表或包含错误信息。

该类的文档由以下文件生成:

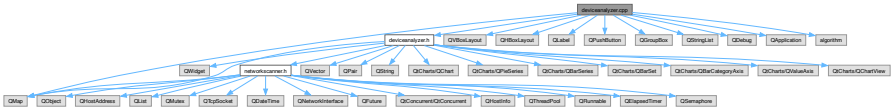
- networktopology.h
- src/core/topologyanalyzer.h
- networktopology.cpp

Chapter 8

文件说明

8.1 deviceanalyzer.cpp 文件参考

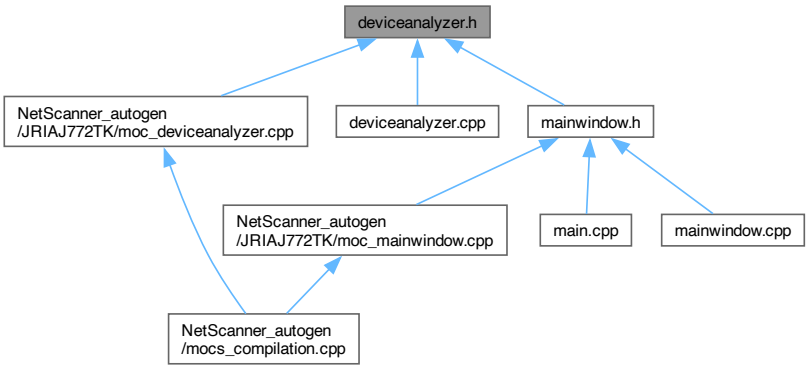
#include ”deviceanalyzer.h”
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QLabel>
#include <QPushButton>
#include <QGroupBox>
#include <QMap>
#include <QStringList>
#include <QDebug>
#include <QApplication>
#include <algorithm>
deviceanalyzer.cpp 的引用 (Include) 关系图:



8.2 deviceanalyzer.h 文件参考

#include <QWidget>
#include <QMap>
#include <QVector>
#include <QPair>
#include <QString>
#include <QtCharts/QChart>
#include <QtCharts/QPieSeries>
#include <QtCharts/QBarSeries>
#include <QtCharts/QBarSet>
#include <QtCharts/QBarCategoryAxis>
#include <QtCharts/QValueAxis>
#include <QtCharts/QChartView>
#include ”networkscanner.h”
deviceanalyzer.h 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- class DeviceAnalyzer

  设备分析器类 - 提供对扫描结果的统计分析

8.3 deviceanalyzer.h

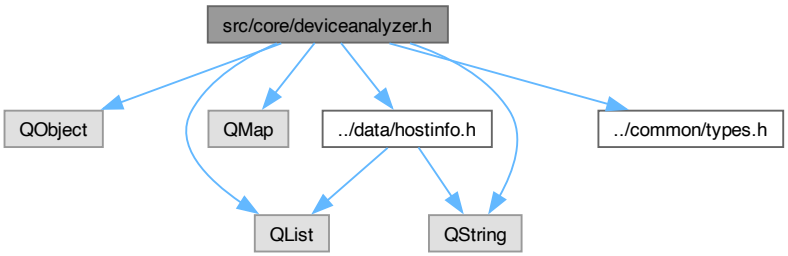
浏览该文件的文档.

```
00001 #ifndef DEVICEANALYZER_H
00002 #define DEVICEANALYZER_H
00003
00004 #include <QWidget>
00005 #include <QMap>
00006 #include <QVector>
00007 #include <QPair>
00008 #include <QString>
00009 #include <QtCharts/QChart>
00010 #include <QtCharts/QPieSeries>
00011 #include <QtCharts/QBarSeries>
00012 #include <QtCharts/QBarSet>
00013 #include <QtCharts/QBarCategoryAxis>
00014 #include <QtCharts/QValueAxis>
00015 #include <QtCharts/QChartView>
00016 #include "networkscanner.h"
00017
00018 // 设备分析器类 - 提供对扫描结果的统计分析
00023 class DeviceAnalyzer : public QWidget
00024 {
00025     Q_OBJECT
00026
00027 public:
00032     DeviceAnalyzer(QWidget *parent = nullptr);
00033
00038     void analyzeHosts(const QList<HostInfo> &hosts);
00042     void clear();
00043
00044     // 获取各种统计信息
00049     int getTotalHostsCount() const { return m_totalHosts; }
00054     int getReachableHostsCount() const { return m_reachableHosts; }
00059     int getUnreachableHostsCount() const { return m_totalHosts - m_reachableHosts; }
00060
00061     // 获取各种图表
00066     QChart* getDeviceTypeChart() const { return m_deviceTypeChart; }
00071     QChart* getPortDistributionChart() const { return m_portDistributionChart; }
00076     QChart* getVendorDistributionChart() const { return m_vendorDistributionChart; }
00077
00083     QString generateSecurityReport(const QList<HostInfo> &hosts);
00084
00085 signals:
00089     void analysisCompleted();
00090
00091 private:
00092     // 扫描统计
```

```
00093     int m_totalHosts;
00094     int m_reachableHosts;
00095
00096     // 设备类型分布图表
00097     QChart *m_deviceTypeChart;
00098     QPieSeries *m_deviceTypeSeries;
00099
00100     // 端口分布图表
00101     QChart *m_portDistributionChart;
00102     QBarSeries *m_portSeries;
00103
00104     // 厂商分布图表
00105     QChart *m_vendorDistributionChart;
00106     QPieSeries *m_vendorSeries;
00107
00108     // 创建各类图表
00112     void createDeviceTypeChart();
00116     void createPortDistributionChart();
00120     void createVendorDistributionChart();
00121
00122     // 设备类型判断
00128     QString determineDeviceType(const HostInfo &host);
00129 };
00130
00131 #endif // DEVICEANALYZER_H
```

8.4 src/core/deviceanalyzer.h 文件参考

定义设备分析类，用于统计和分析扫描到的设备信息。
#include <QObject>
#include <QList>
#include <QMap>
#include <QString>
#include "../data/hostinfo.h"
#include "../common/types.h"
deviceanalyzer.h 的引用 (Include) 关系图:



类

- struct DeviceStats

  存储设备统计信息。
- class DeviceAnalyzer

  设备分析器类 - 提供对扫描结果的统计分析

8.4.1 详细描述

定义设备分析类，用于统计和分析扫描到的设备信息。

版权所有

Copyright (c) 2024

8.5 deviceanalyzer.h

浏览该文件的文档.

```
00001
00007
00008 #ifndef DEVICEANALYZER_H
00009 #define DEVICEANALYZER_H
00010
00011 #include <QObject> // 可能需要 QObject 以支持异步分析或信号
00012 #include <QList>
00013 #include <QMap>
00014 #include <QString>
00015 #include "../data/hostinfo.h"
00016 #include "../common/types.h" // For DeviceType enum
00017
00022 struct DeviceStats {
00023     int totalDevices;
00024     QMap<DeviceType, int> countByType;
00025     QMap<QString, int> countByOS;
00026     QMap<QString, int> countByVendor;
00027     QList<QString> commonOpenPorts;
00028     // 可以添加更多统计维度，例如平均响应时间、存活主机百分比等
00029 };
00030
00038 class DeviceAnalyzer : public QObject
00039 {
00040     Q_OBJECT
00041
00042 public:
00047     explicit DeviceAnalyzer(QObject *parent = nullptr);
00048
00053     void setHostData(const QList<HostInfo> &hosts);
00054
00062     DeviceStats analyze();
00063
00071     DeviceType inferDeviceType(const HostInfo &host) const;
00072
00079     QString inferVendorFromMac(const QString &macAddress) const;
00080
00081     // 可以添加更多特定的分析方法，例如：
00082     // QMap<QString, int> getOperatingSystemDistribution() const;
00083     // QMap<DeviceType, int> getDeviceTypeDistribution() const;
00084     // QList<PortInfo> getMostCommonOpenPorts(int topN = 10) const;
00085
00086 signals:
00091     void analysisComplete(const DeviceStats &stats);
00092
00093 private:
00094     QList<HostInfo> m_hosts;
00095     DeviceStats m_currentStats;
00096
00097     // 内部可能需要 OUI 数据库的引用或加载逻辑
00098     // QMap<QString, QString> m_ouiDatabase;
00099 };
00100
00101 #endif // DEVICEANALYZER_H
```

8.6 main.cpp 文件参考

#include "mainwindow.h"
#include <QApplication>
#include <QFontDatabase>
#include <QSplashScreen>
#include <QPixmap>
#include <QTimer>
#include <QFont>
#include <QStyleFactory>
main.cpp 的引用 (Include) 关系图:

函数

- int main (int argc, char ∗argv[])
  主函数，应用程序入口点。

8.6.1 函数说明

8.6.1.1 main()

int main (
int argc,
char ∗ argv[])
主函数，应用程序入口点。

参数

argc	命令行参数数量。
argv	命令行参数数组。

返回

应用程序退出代码。

8.7 mainwindow.cpp 文件参考

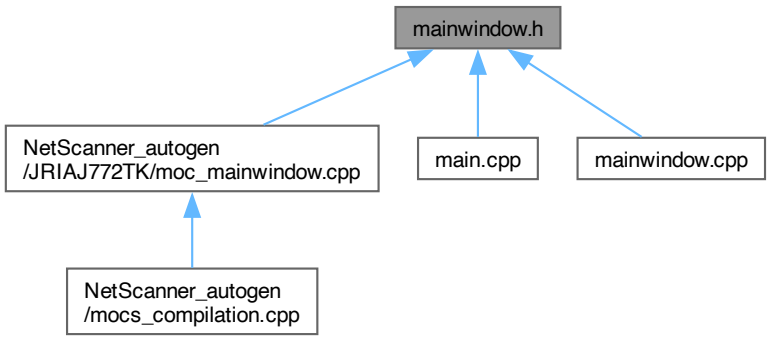
#include ”mainwindow.h”
#include <QColor>
#include <QDateTime>
#include <QStandardPaths>
#include <QTimer>
#include <QPalette>
#include <QPixmap>
#include <QPainter>
#include <QStyleFactory>
#include <QInputDialog>
#include <QCalendarWidget>
#include <QScrollArea>
#include <QTimeEdit>
mainwindow.cpp 的引用 (Include) 关系图:

8.8 mainwindow.h 文件参考

#include <QMainWindow>
#include <QTableWidget>
#include <QPushButton>
#include <QProgressBar>
#include <QLabel>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QStatusBar>
#include <QHeaderView>
#include <QMessageBox>
#include <QLineEdit>

#include <QComboBox>
#include <QSpinBox>
#include <QCheckBox>
#include <QGroupBox>
#include <QFileDialog>
#include <QAction>
#include <QMenu>
#include <QMenuBar>
#include <QTabWidget>
#include <QTextEdit>
#include <QSettings>
#include <QSplitter>
#include <QStyledItemDelegate>
#include <QApplication>
#include <QtCharts/QChartView>
#include <QtCharts/QChart>
#include <QGraphicsView>
#include <QGraphicsScene>
#include "networkscanner.h"
#include "networktopology.h"
#include "deviceanalyzer.h"
#include "scanhistory.h"
mainwindow.h 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- class MainWindow

  主窗口类，负责程序的主要界面和交互逻辑。

8.9 mainwindow.h

浏览该文件的文档.

```
00001 #ifndef MAINWINDOW_H
00002 #define MAINWINDOW_H
00003
```

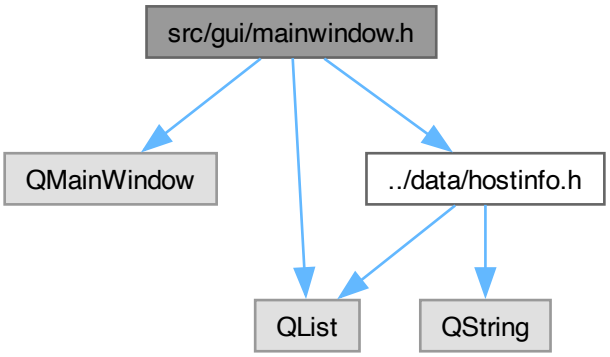
```
00004 #include <QMainWindow>
00005 #include <QTableWidget>
00006 #include <QPushButton>
00007 #include <QProgressBar>
00008 #include <QLabel>
00009 #include <QVBoxLayout>
00010 #include <QHBoxLayout>
00011 #include <QStatusBar>
00012 #include <QHeaderView>
00013 #include <QMessageBox>
00014 #include <QLineEdit>
00015 #include <QComboBox>
00016 #include <QSpinBox>
00017 #include <QCheckBox>
00018 #include <QGroupBox>
00019 #include <QFileDialog>
00020 #include <QAction>
00021 #include <QMenu>
00022 #include <QMenuBar>
00023 #include <QTabWidget>
00024 #include <QTextEdit>
00025 #include <QSettings>
00026 #include <QSplitter>
00027 #include <QStyledItemDelegate>
00028 #include <QApplication>
00029 #include <QtCharts/QChartView>
00030 #include <QtCharts/QChart>
00031 #include <QGraphicsView>
00032 #include <QGraphicsScene>
00033
00034 #include "networkscanner.h"
00035 #include "networktopology.h"
00036 #include "deviceanalyzer.h"
00037 #include "scanhistory.h"
00038
00039 // QtCharts 命名空间已经在 deviceanalyzer.h 中引入
00040 // using namespace QtCharts;
00041
00046 class MainWindow : public QMainWindow
00047 {
00048     Q_OBJECT
00049
00050 public:
00055     MainWindow(QWidget *parent = nullptr);
00059     ~MainWindow();
00060
00061 private slots:
00063     void startScan();
00065     void stopScan();
00070     void onHostFound(const HostInfo &host);
00072     void onScanStarted();
00074     void onScanFinished();
00079     void onScanProgress(int progress);
00084     void onScanError(const QString &errorMessage);
00085
00086     // 现有功能槽
00088     void saveResults();
00090     void clearResults();
00092     void showSettings();
00094     void applySettings();
00096     void showAbout();
00102     void showHostDetails(int row, int column);
00104     void exportToCSV();
00109     void togglePortScanOptions(bool checked);
00114     void toggleRangeOptions(bool checked);
00115
00116     // 新增功能槽
00118     void showTopologyView();
00120     void showStatisticsView();
00122     void showHistoryView();
00124     void generateSecurityReport();
00126     void saveTopologyImage();
00131     void toggleDarkMode(bool enable);
00133     void compareScanResults();
00135     void scheduleScan();
00137     void saveHistoryToFile();
00139     void loadHistoryFromFile();
00141     void updateNetworkTopology();
00143     void refreshTopology();
00145     void filterResults();
00147     void clearFilters();
00149     void onThemeChanged();
00150
00151 private:
00153     void createUI();
00155     void createMenus();
00157     void createSettingsDialog();
```

```
00159     void createTopologyTab();
00161     void createStatisticsTab();
00163     void createHistoryTab();
00165     void createDetailsTab();
00167     void createSecurityTab(); // TODO: 根据实际需求决定是否实现
00169     void setupConnections();
00171     void updatePortsList(); // TODO: 根据实际需求决定是否实现
00173     void loadSettings();
00175     void saveSettings();
00177     void updateStatistics();
00182     void applyTheme(bool darkMode);
00183
00184     // UI 元素
00185     QWidget *m_centralWidget;
00186     QTabWidget *m_tabWidget;
00187
00188     // 扫描结果标签页
00189     QWidget *m_scanTab;
00190     QVBoxLayout *m_mainLayout;
00191     QHBoxLayout *m_controlLayout;
00192     QTableWidget *m_resultsTable;
00193     QPushButton *m_scanButton;
00194     QPushButton *m_stopButton;
00195     QPushButton *m_clearButton;
00196     QPushButton *m_saveButton;
00197     QProgressBar *m_progressBar;
00198     QLabel *m_statusLabel;
00199     QStatusBar *m_statusBar;
00200
00201     // 扫描设置标签页
00202     QWidget *m_settingsTab;
00203     QVBoxLayout *m_settingsLayout;
00204
00205     // 端口设置区域
00206     QGroupBox *m_portsGroupBox;
00207     QCheckBox *m_customPortsCheckBox;
00208     QLineEdit *m_portsLineEdit;
00209     QSpinBox *m_timeoutSpinBox;
00210
00211     // IP 范围设置区域
00212     QGroupBox *m_rangeGroupBox;
00213     QCheckBox *m_customRangeCheckBox;
00214     QLineEdit *m_startIPLineEdit;
00215     QLineEdit *m_endIPLineEdit;
00216
00217     // 主机详情标签页
00218     QWidget *m_detailsTab;
00219     QVBoxLayout *m_detailsLayout;
00220     QTextEdit *m_detailsTextEdit;
00221
00222     // 网络拓扑标签页
00223     QWidget *m_topologyTab;
00224     NetworkTopology *m_networkTopology;
00225
00226     // 统计分析标签页
00227     QWidget *m_statisticsTab;
00228     DeviceAnalyzer *m_deviceAnalyzer;
00229     QChartView *m_deviceTypeChartView;
00230     QChartView *m_vendorChartView;
00231     QChartView *m_portDistributionChartView;
00232     QTextEdit *m_securityReportText;
00233
00234     // 扫描历史标签页
00235     QWidget *m_historyTab;
00236     ScanHistory *m_scanHistory;
00237     QComboBox *m_sessionComboBox;
00238     QTableWidget *m_historyTable;
00239
00240     // 菜单项
00241     QMenu *m_fileMenu;
00242     QMenu *m_viewMenu;
00243     QMenu *m_toolsMenu;
00244     QMenu *m_helpMenu;
00245     QAction *m_exportAction;
00246     QAction *m_saveHistoryAction;
00247     QAction *m_loadHistoryAction;
00248     QAction *m_saveTopologyAction;
00249     QAction *m_exitAction;
00250     QAction *m_settingsAction;
00251     QAction *m_darkModeAction;
00252     QAction *m_scheduleScanAction;
00253     QAction *m_aboutAction;
00254
00255     // 过滤控件
00256     QWidget *m_filterWidget;
00257     QLineEdit *m_filterIPLineEdit;
00258     QComboBox *m_filterVendorComboBox;
```

```
00259     QComboBox *m_filterTypeComboBox;
00260     QPushButton *m_filterButton;
00261     QPushButton *m_clearFilterButton;
00262
00263     // 网络扫描器
00264     NetworkScanner *m_scanner;
00265
00266     // 扫描的主机数量
00267     int m_hostsFound;
00268
00269     // 当前查看的主机索引
00270     int m_currentHostIndex;
00271
00272     // 主题设置
00273     bool m_darkModeEnabled;
00274 };
00275
00276 #endif // MAINWINDOW_H
```

8.10 src/gui/mainwindow.h 文件参考

定义应用程序的主窗口类。
#include <QMainWindow>
#include <QList>
#include "../data/hostinfo.h"
mainwindow.h 的引用 (Include) 关系图:



类

- class MainWindow

  主窗口类，负责程序的主要界面和交互逻辑。

命名空间

- namespace Ui

8.10.1 详细描述

定义应用程序的主窗口类。

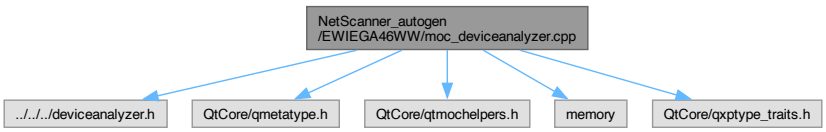
8.11 mainwindow.h

浏览该文件的文档.

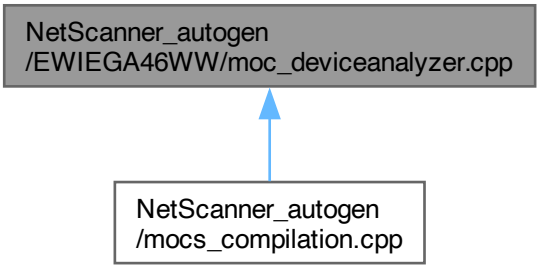
```
00001
00008
00009 #ifndef MAINWINDOW_H
00010 #define MAINWINDOW_H
00011
00012 #include <QMainWindow>
00013 #include <QList>
00014 #include "../data/hostinfo.h" // For HostInfo struct
00015
00016 // 前向声明 ui 类 (如果使用 .ui 文件)
00017 QT_BEGIN_NAMESPACE
00018 namespace Ui { class MainWindow; }
00019 QT_END_NAMESPACE
00020
00021 // 前向声明自定义组件
00022 class NetworkScanner;       // 在 core 模块
00023 class NetworkTopology;      // 在 gui 模块
00024 class DeviceAnalyzerView;   // 假设的设备分析视图类 (可能在 gui 或新模块)
00025 class ScanHistoryView;      // 假设的扫描历史视图类 (可能在 gui 或新模块)
00026
00034 class MainWindow : public QMainWindow
00035 {
00036     Q_OBJECT
00037
00038 public:
00043     MainWindow(QWidget *parent = nullptr);
00044
00048     ~MainWindow();
00049
00050 protected:
00055     void closeEvent(QCloseEvent *event) override;
00056
00057 private slots:
00058     // UI 动作槽函数
00059     void on_actionStartScan_triggered();
00060     void on_actionStopScan_triggered();
00061     void on_actionConfigureScan_triggered();
00062     void on_actionViewTopology_triggered();
00063     void on_actionViewAnalysis_triggered();
00064     void on_actionViewHistory_triggered();
00065     void on_actionExit_triggered();
00066     void on_actionAbout_triggered();
00067
00068     // NetworkScanner 信号的槽函数
00069     void handleHostFound(const HostInfo &hostInfo);
00070     void handleScanProgress(int percentage, const QString &message);
00071     void handleScanFinished(const QList<HostInfo> &results);
00072     void handleScanError(const QString &errorMessage);
00073
00074     // NetworkTopology 信号的槽函数
00075     void handleDeviceSelectedFromTopology(const HostInfo &hostInfo);
00076
00077 private:
00081     void createMenus();
00082
00086     void createToolbars();
00087
00091     void createStatusbar();
00092
00096     void setupDockWidgets(); // 或者 setupCentralWidget()
00097
00101     void readSettings();
00102
00106     void writeSettings();
00107
00108     Ui::MainWindow *ui;
00109
00110     // 功能模块实例
00111     NetworkScanner *m_networkScanner;
00112     NetworkTopology *m_networkTopologyWidget;
00113     // DeviceAnalyzerView *m_deviceAnalyzerView; ///< @brief 设备分析视图组件。
00114     // ScanHistoryView *m_scanHistoryView;       ///< @brief 扫描历史视图组件。
00115
00116     // 其他成员变量
00117     QList<HostInfo> m_currentScanResults;
00118     QString m_currentScanTarget;
00119 };
00120
00121 #endif // MAINWINDOW_H
```

8.12 NetScanner_autogen/EWIEGA46WW/moc_deviceanalyzer.cpp 文件参考

#include "../../../deviceanalyzer.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
moc_deviceanalyzer.cpp 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14DeviceAnalyzerE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.12.1 宏定义说明

8.12.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.13 moc_deviceanalyzer.cpp

浏览该文件的文档.

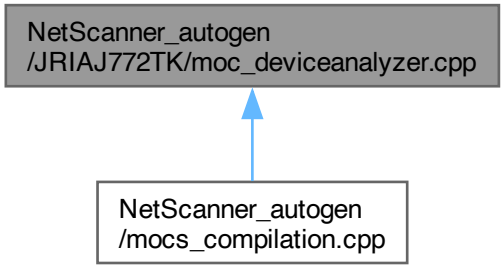
```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'deviceanalyzer.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 ****************************************************************************/
00008
00009 #include "../../../deviceanalyzer.h"
00010 #include <QtCore/qmetatype.h>
00011
00012 #include <QtCore/qtmochelpers.h>
00013
00014 #include <memory>
00015
00016
00017 #include <QtCore/qxptype_traits.h>
00018 #if !defined(Q_MOC_OUTPUT_REVISION)
00019 #error "The header file 'deviceanalyzer.h' doesn't include <QObject>."
00020 #elif Q_MOC_OUTPUT_REVISION != 69
00021 #error "This file was generated using the moc from 6.9.0. It"
00022 #error "cannot be used with the include files from this version of Qt."
00023 #error "(The moc has changed too much.)"
00024 #endif
00025
00026 #ifndef Q_CONSTINIT
00027 #define Q_CONSTINIT
00028 #endif
00029
00030 QT_WARNING_PUSH
00031 QT_WARNING_DISABLE_DEPRECATED
00032 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00033 namespace {
00034 struct qt_meta_tag_ZN14DeviceAnalyzerE_t {};
00035 } // unnamed namespace
00036
00037 template <> constexpr inline auto
      DeviceAnalyzer::qt_create_metaobjectdata<qt_meta_tag_ZN14DeviceAnalyzerE_t>()
00038 {
00039     namespace QMC = QtMocConstants;
00040     QtMocHelpers::StringRefStorage qt_stringData {
00041         "DeviceAnalyzer",
00042         "analysisCompleted",
00043         ""
00044     };
00045
00046     QtMocHelpers::UintData qt_methods {
00047         // Signal 'analysisCompleted'
00048         QtMocHelpers::SignalData<void()>(1, 2, QMC::AccessPublic, QMetaType::Void),
00049     };
00050     QtMocHelpers::UintData qt_properties {
00051     };
00052     QtMocHelpers::UintData qt_enums {
00053     };
00054     return QtMocHelpers::metaObjectData<DeviceAnalyzer,
      qt_meta_tag_ZN14DeviceAnalyzerE_t>(QMC::MetaObjectFlag{}, qt_stringData,
00055             qt_methods, qt_properties, qt_enums);
00056 }
00057 Q_CONSTINIT const QMetaObject DeviceAnalyzer::staticMetaObject = { {
00058     QMetaObject::SuperData::link<QWidget::staticMetaObject>(),
00059     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.stringdata,
00060     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.data,
00061     qt_static_metacall,
00062     nullptr,
00063     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.metaTypes,
00064     nullptr
00065 } };
00066
00067 void DeviceAnalyzer::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00068 {
00069     auto *_t = static_cast<DeviceAnalyzer *>(_o);
00070     if (_c == QMetaObject::InvokeMetaMethod) {
00071         switch (_id) {
00072         case 0: _t->analysisCompleted(); break;
00073         default: ;
00074         }
00075     }
00076     if (_c == QMetaObject::IndexOfMethod) {
00077         if (QtMocHelpers::indexOfMethod<void (DeviceAnalyzer::*)()>(_a, &DeviceAnalyzer::analysisCompleted, 0))
00078             return;
00079     }
00080 }
00081
00082 const QMetaObject *DeviceAnalyzer::metaObject() const
00083 {
00084     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00085 }
```

```
00086
00087 void *DeviceAnalyzer::qt_metacast(const char *_clname)
00088 {
00089     if (!_clname) return nullptr;
00090     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.strings))
00091         return static_cast<void*>(this);
00092     return QWidget::qt_metacast(_clname);
00093 }
00094
00095 int DeviceAnalyzer::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00096 {
00097     _id = QWidget::qt_metacall(_c, _id, _a);
00098     if (_id < 0)
00099         return _id;
00100     if (_c == QMetaObject::InvokeMetaMethod) {
00101         if (_id < 1)
00102             qt_static_metacall(this, _c, _id, _a);
00103         _id -= 1;
00104     }
00105     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00106         if (_id < 1)
00107             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00108         _id -= 1;
00109     }
00110     return _id;
00111 }
00112
00113 // SIGNAL 0
00114 void DeviceAnalyzer::analysisCompleted()
00115 {
00116     QMetaObject::activate(this, &staticMetaObject, 0, nullptr);
00117 }
00118 QT_WARNING_POP
```

8.14 NetScanner_autogen/JRIAJ772TK/moc_deviceanalyzer.cpp 文件参考

#include "../../deviceanalyzer.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
moc_deviceanalyzer.cpp 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14DeviceAnalyzerE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.14.1 宏定义说明

8.14.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.15 moc_deviceanalyzer.cpp

浏览该文件的文档.

```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'deviceanalyzer.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 ****************************************************************************/
00008
00009 #include "../../deviceanalyzer.h"
00010 #include <QtCore/qmetatype.h>
00011
00012 #include <QtCore/qtmochelpers.h>
00013
00014 #include <memory>
00015
00016
00017 #include <QtCore/qxptype_traits.h>
00018 #if !defined(Q_MOC_OUTPUT_REVISION)
00019 #error "The header file 'deviceanalyzer.h' doesn't include <QObject>."
00020 #elif Q_MOC_OUTPUT_REVISION != 69
00021 #error "This file was generated using the moc from 6.9.0. It"
00022 #error "cannot be used with the include files from this version of Qt."
00023 #error "(The moc has changed too much.)"
00024 #endif
00025
00026 #ifndef Q_CONSTINIT
00027 #define Q_CONSTINIT
00028 #endif
00029
00030 QT_WARNING_PUSH
00031 QT_WARNING_DISABLE_DEPRECATED
00032 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00033 namespace {
00034 struct qt_meta_tag_ZN14DeviceAnalyzerE_t {};
00035 } // unnamed namespace
00036
00037 template <> constexpr inline auto
      DeviceAnalyzer::qt_create_metaobjectdata<qt_meta_tag_ZN14DeviceAnalyzerE_t>()
00038 {
00039     namespace QMC = QtMocConstants;
00040     QtMocHelpers::StringRefStorage qt_stringData {
00041         "DeviceAnalyzer",
00042         "analysisCompleted",
00043         ""
00044     };
00045
00046     QtMocHelpers::UintData qt_methods {
00047         // Signal 'analysisCompleted'
00048         QtMocHelpers::SignalData<void()>(1, 2, QMC::AccessPublic, QMetaType::Void),
00049     };
00050     QtMocHelpers::UintData qt_properties {
00051     };
00052     QtMocHelpers::UintData qt_enums {
00053     };
00054     return QtMocHelpers::metaObjectData<DeviceAnalyzer,
      qt_meta_tag_ZN14DeviceAnalyzerE_t>(QMC::MetaObjectFlag{}, qt_stringData,
00055             qt_methods, qt_properties, qt_enums);
```

```
00056 }
00057 Q_CONSTINIT const QMetaObject DeviceAnalyzer::staticMetaObject = { {
00058     QMetaObject::SuperData::link<QWidget::staticMetaObject>(),
00059     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.stringdata,
00060     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.data,
00061     qt_static_metacall,
00062     nullptr,
00063     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.metaTypes,
00064     nullptr
00065 } };
00066
00067 void DeviceAnalyzer::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00068 {
00069     auto *_t = static_cast<DeviceAnalyzer *>(_o);
00070     if (_c == QMetaObject::InvokeMetaMethod) {
00071         switch (_id) {
00072         case 0: _t->analysisCompleted(); break;
00073         default: ;
00074         }
00075     }
00076     if (_c == QMetaObject::IndexOfMethod) {
00077         if (QtMocHelpers::indexOfMethod<void (DeviceAnalyzer::*)()>(_a, &DeviceAnalyzer::analysisCompleted, 0))
00078             return;
00079     }
00080 }
00081
00082 const QMetaObject *DeviceAnalyzer::metaObject() const
00083 {
00084     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00085 }
00086
00087 void *DeviceAnalyzer::qt_metacast(const char *_clname)
00088 {
00089     if (!_clname) return nullptr;
00090     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14DeviceAnalyzerE_t>.strings))
00091         return static_cast<void*>(this);
00092     return QWidget::qt_metacast(_clname);
00093 }
00094
00095 int DeviceAnalyzer::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00096 {
00097     _id = QWidget::qt_metacall(_c, _id, _a);
00098     if (_id < 0)
00099         return _id;
00100     if (_c == QMetaObject::InvokeMetaMethod) {
00101         if (_id < 1)
00102             qt_static_metacall(this, _c, _id, _a);
00103         _id -= 1;
00104     }
00105     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00106         if (_id < 1)
00107             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00108         _id -= 1;
00109     }
00110     return _id;
00111 }
00112
00113 // SIGNAL 0
00114 void DeviceAnalyzer::analysisCompleted()
00115 {
00116     QMetaObject::activate(this, &staticMetaObject, 0, nullptr);
00117 }
00118 QT_WARNING_POP
```

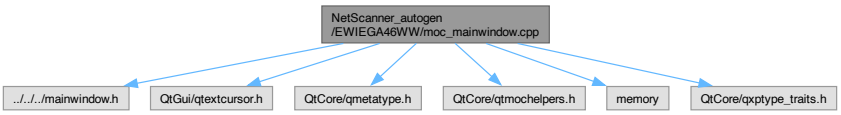
8.16 NetScanner_autogen/EWIEGA46WW/moc_deviceanalyzer.cpp.d 文件参考

8.17 NetScanner_autogen/JRIAJ772TK/moc_deviceanalyzer.cpp.d 文件参考

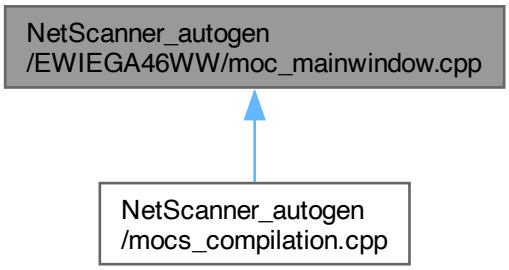
8.18 NetScanner_autogen/EWIEGA46WW/moc_mainwindow.cpp 文件参考

```
#include "../../../mainwindow.h"
#include <QtGui/qtextcursor.h>
#include <QtCore/qmetatype.h>
```

#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
moc_mainwindow.cpp 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN10MainWindowE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.18.1 宏定义说明

8.18.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.19 moc_mainwindow.cpp

浏览该文件的文档.

```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'mainwindow.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 ****************************************************************************/
00008
00009 #include "../../../mainwindow.h"
```

```
00010 #include <QtGui/qtextcursor.h>
00011 #include <QtCore/qmetatype.h>
00012
00013 #include <QtCore/qtmochelpers.h>
00014
00015 #include <memory>
00016
00017
00018 #include <QtCore/qxptype_traits.h>
00019 #if !defined(Q_MOC_OUTPUT_REVISION)
00020 #error "The header file 'mainwindow.h' doesn't include <QObject>."
00021 #elif Q_MOC_OUTPUT_REVISION != 69
00022 #error "This file was generated using the moc from 6.9.0. It"
00023 #error "cannot be used with the include files from this version of Qt."
00024 #error "(The moc has changed too much.)"
00025 #endif
00026
00027 #ifndef Q_CONSTINIT
00028 #define Q_CONSTINIT
00029 #endif
00030
00031 QT_WARNING_PUSH
00032 QT_WARNING_DISABLE_DEPRECATED
00033 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00034 namespace {
00035 struct qt_meta_tag_ZN10MainWindowE_t {};
00036 } // unnamed namespace
00037
00038 template <> constexpr inline auto MainWindow::qt_create_metaobjectdata<qt_meta_tag_ZN10MainWindowE_t>()
00039 {
00040     namespace QMC = QtMocConstants;
00041     QtMocHelpers::StringRefStorage qt_stringData {
00042         "MainWindow",
00043         "startScan",
00044         "",
00045         "stopScan",
00046         "onHostFound",
00047         "HostInfo",
00048         "host",
00049         "onScanStarted",
00050         "onScanFinished",
00051         "onScanProgress",
00052         "progress",
00053         "onScanError",
00054         "errorMessage",
00055         "saveResults",
00056         "clearResults",
00057         "showSettings",
00058         "applySettings",
00059         "showAbout",
00060         "showHostDetails",
00061         "row",
00062         "column",
00063         "exportToCSV",
00064         "togglePortScanOptions",
00065         "checked",
00066         "toggleRangeOptions",
00067         "showTopologyView",
00068         "showStatisticsView",
00069         "showHistoryView",
00070         "generateSecurityReport",
00071         "saveTopologyImage",
00072         "toggleDarkMode",
00073         "enable",
00074         "compareScanResults",
00075         "scheduleScan",
00076         "saveHistoryToFile",
00077         "loadHistoryFromFile",
00078         "updateNetworkTopology",
00079         "refreshTopology",
00080         "filterResults",
00081         "clearFilters",
00082         "onThemeChanged"
00083     };
00084
00085     QtMocHelpers::UintData qt_methods {
00086         // Slot 'startScan'
00087         QtMocHelpers::SlotData<void()>(1, 2, QMC::AccessPrivate, QMetaType::Void),
00088         // Slot 'stopScan'
00089         QtMocHelpers::SlotData<void()>(3, 2, QMC::AccessPrivate, QMetaType::Void),
00090         // Slot 'onHostFound'
00091         QtMocHelpers::SlotData<void(const HostInfo &)>(4, 2, QMC::AccessPrivate, QMetaType::Void, {{
00092             { 0x80000000 | 5, 6 },
00093         }}),
00094         // Slot 'onScanStarted'
00095         QtMocHelpers::SlotData<void()>(7, 2, QMC::AccessPrivate, QMetaType::Void),
00096         // Slot 'onScanFinished'
```

```
00097         QtMocHelpers::SlotData<void()>(8, 2, QMC::AccessPrivate, QMetaType::Void),
00098         // Slot 'onScanProgress'
00099         QtMocHelpers::SlotData<void(int)>(9, 2, QMC::AccessPrivate, QMetaType::Void, {{
00100             { QMetaType::Int, 10 },
00101         }}),
00102         // Slot 'onScanError'
00103         QtMocHelpers::SlotData<void(const QString &)>(11, 2, QMC::AccessPrivate, QMetaType::Void, {{
00104             { QMetaType::QString, 12 },
00105         }}),
00106         // Slot 'saveResults'
00107         QtMocHelpers::SlotData<void()>(13, 2, QMC::AccessPrivate, QMetaType::Void),
00108         // Slot 'clearResults'
00109         QtMocHelpers::SlotData<void()>(14, 2, QMC::AccessPrivate, QMetaType::Void),
00110         // Slot 'showSettings'
00111         QtMocHelpers::SlotData<void()>(15, 2, QMC::AccessPrivate, QMetaType::Void),
00112         // Slot 'applySettings'
00113         QtMocHelpers::SlotData<void()>(16, 2, QMC::AccessPrivate, QMetaType::Void),
00114         // Slot 'showAbout'
00115         QtMocHelpers::SlotData<void()>(17, 2, QMC::AccessPrivate, QMetaType::Void),
00116         // Slot 'showHostDetails'
00117         QtMocHelpers::SlotData<void(int, int)>(18, 2, QMC::AccessPrivate, QMetaType::Void, {{
00118             { QMetaType::Int, 19 }, { QMetaType::Int, 20 },
00119         }}),
00120         // Slot 'exportToCSV'
00121         QtMocHelpers::SlotData<void()>(21, 2, QMC::AccessPrivate, QMetaType::Void),
00122         // Slot 'togglePortScanOptions'
00123         QtMocHelpers::SlotData<void(bool)>(22, 2, QMC::AccessPrivate, QMetaType::Void, {{
00124             { QMetaType::Bool, 23 },
00125         }}),
00126         // Slot 'toggleRangeOptions'
00127         QtMocHelpers::SlotData<void(bool)>(24, 2, QMC::AccessPrivate, QMetaType::Void, {{
00128             { QMetaType::Bool, 23 },
00129         }}),
00130         // Slot 'showTopologyView'
00131         QtMocHelpers::SlotData<void()>(25, 2, QMC::AccessPrivate, QMetaType::Void),
00132         // Slot 'showStatisticsView'
00133         QtMocHelpers::SlotData<void()>(26, 2, QMC::AccessPrivate, QMetaType::Void),
00134         // Slot 'showHistoryView'
00135         QtMocHelpers::SlotData<void()>(27, 2, QMC::AccessPrivate, QMetaType::Void),
00136         // Slot 'generateSecurityReport'
00137         QtMocHelpers::SlotData<void()>(28, 2, QMC::AccessPrivate, QMetaType::Void),
00138         // Slot 'saveTopologyImage'
00139         QtMocHelpers::SlotData<void()>(29, 2, QMC::AccessPrivate, QMetaType::Void),
00140         // Slot 'toggleDarkMode'
00141         QtMocHelpers::SlotData<void(bool)>(30, 2, QMC::AccessPrivate, QMetaType::Void, {{
00142             { QMetaType::Bool, 31 },
00143         }}),
00144         // Slot 'compareScanResults'
00145         QtMocHelpers::SlotData<void()>(32, 2, QMC::AccessPrivate, QMetaType::Void),
00146         // Slot 'scheduleScan'
00147         QtMocHelpers::SlotData<void()>(33, 2, QMC::AccessPrivate, QMetaType::Void),
00148         // Slot 'saveHistoryToFile'
00149         QtMocHelpers::SlotData<void()>(34, 2, QMC::AccessPrivate, QMetaType::Void),
00150         // Slot 'loadHistoryFromFile'
00151         QtMocHelpers::SlotData<void()>(35, 2, QMC::AccessPrivate, QMetaType::Void),
00152         // Slot 'updateNetworkTopology'
00153         QtMocHelpers::SlotData<void()>(36, 2, QMC::AccessPrivate, QMetaType::Void),
00154         // Slot 'refreshTopology'
00155         QtMocHelpers::SlotData<void()>(37, 2, QMC::AccessPrivate, QMetaType::Void),
00156         // Slot 'filterResults'
00157         QtMocHelpers::SlotData<void()>(38, 2, QMC::AccessPrivate, QMetaType::Void),
00158         // Slot 'clearFilters'
00159         QtMocHelpers::SlotData<void()>(39, 2, QMC::AccessPrivate, QMetaType::Void),
00160         // Slot 'onThemeChanged'
00161         QtMocHelpers::SlotData<void()>(40, 2, QMC::AccessPrivate, QMetaType::Void),
00162     };
00163     QtMocHelpers::UintData qt__properties {
00164     };
00165     QtMocHelpers::UintData qt__enums {
00166     };
00167     return QtMocHelpers::metaObjectData<MainWindow,
      qt__meta__tag__ZN10MainWindowE__t>(QMC::MetaObjectFlag{}, qt__stringData,
00168             qt__methods, qt__properties, qt__enums);
00169 }
00170 Q_CONSTINIT const QMetaObject MainWindow::staticMetaObject = { {
00171     QMetaObject::SuperData::link<QMainWindow::staticMetaObject>(),
00172     qt__staticMetaObjectStaticContent<qt__meta__tag__ZN10MainWindowE__t>.stringdata,
00173     qt__staticMetaObjectStaticContent<qt__meta__tag__ZN10MainWindowE__t>.data,
00174     qt__static__metacall,
00175     nullptr,
00176     qt__staticMetaObjectRelocatingContent<qt__meta__tag__ZN10MainWindowE__t>.metaTypes,
00177     nullptr
00178 } };
00179
00180 void MainWindow::qt__static__metacall(QObject *__o, QMetaObject::Call __c, int __id, void **__a)
00181 {
00182     auto *__t = static__cast<MainWindow *>(__o);
```

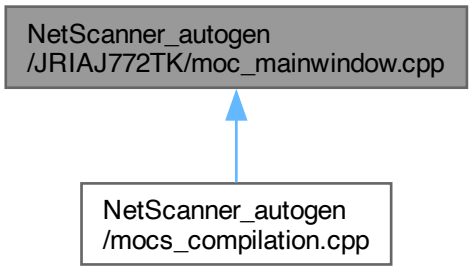
```
00183     if (_c == QMetaObject::InvokeMetaMethod) {
00184         switch (_id) {
00185         case 0: _t->startScan(); break;
00186         case 1: _t->stopScan(); break;
00187         case 2: _t->onHostFound((*reinterpret_cast< std::add_pointer_t<HostInfo>>(_a[1]))); break;
00188         case 3: _t->onScanStarted(); break;
00189         case 4: _t->onScanFinished(); break;
00190         case 5: _t->onScanProgress((*reinterpret_cast< std::add_pointer_t<int>>(_a[1]))); break;
00191         case 6: _t->onScanError((*reinterpret_cast< std::add_pointer_t<QString>>(_a[1]))); break;
00192         case 7: _t->saveResults(); break;
00193         case 8: _t->clearResults(); break;
00194         case 9: _t->showSettings(); break;
00195         case 10: _t->applySettings(); break;
00196         case 11: _t->showAbout(); break;
00197         case 12: _t->showHostDetails((*reinterpret_cast< std::add_pointer_t<int>>(_a[1])),(*reinterpret_cast<
      std::add_pointer_t<int>>(_a[2]))); break;
00198         case 13: _t->exportToCSV(); break;
00199         case 14: _t->togglePortScanOptions((*reinterpret_cast< std::add_pointer_t<bool>>(_a[1]))); break;
00200         case 15: _t->toggleRangeOptions((*reinterpret_cast< std::add_pointer_t<bool>>(_a[1]))); break;
00201         case 16: _t->showTopologyView(); break;
00202         case 17: _t->showStatisticsView(); break;
00203         case 18: _t->showHistoryView(); break;
00204         case 19: _t->generateSecurityReport(); break;
00205         case 20: _t->saveTopologyImage(); break;
00206         case 21: _t->toggleDarkMode((*reinterpret_cast< std::add_pointer_t<bool>>(_a[1]))); break;
00207         case 22: _t->compareScanResults(); break;
00208         case 23: _t->scheduleScan(); break;
00209         case 24: _t->saveHistoryToFile(); break;
00210         case 25: _t->loadHistoryFromFile(); break;
00211         case 26: _t->updateNetworkTopology(); break;
00212         case 27: _t->refreshTopology(); break;
00213         case 28: _t->filterResults(); break;
00214         case 29: _t->clearFilters(); break;
00215         case 30: _t->onThemeChanged(); break;
00216         default: ;
00217         }
00218     }
00219 }
00220
00221 const QMetaObject *MainWindow::metaObject() const
00222 {
00223     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00224 }
00225
00226 void *MainWindow::qt_metacast(const char *_clname)
00227 {
00228     if (!_clname) return nullptr;
00229     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN10MainWindowE_t>.strings))
00230         return static_cast<void*>(this);
00231     return QMainWindow::qt_metacast(_clname);
00232 }
00233
00234 int MainWindow::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00235 {
00236     _id = QMainWindow::qt_metacall(_c, _id, _a);
00237     if (_id < 0)
00238         return _id;
00239     if (_c == QMetaObject::InvokeMetaMethod) {
00240         if (_id < 31)
00241             qt_static_metacall(this, _c, _id, _a);
00242         _id -= 31;
00243     }
00244     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00245         if (_id < 31)
00246             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00247         _id -= 31;
00248     }
00249     return _id;
00250 }
00251 QT_WARNING_POP
```

8.20 NetScanner_autogen/JRIAJ772TK/moc_mainwindow.cpp 文件参考

```
#include "../../mainwindow.h"
#include <QtGui/qtextcursor.h>
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
```

moc_mainwindow.cpp 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN10MainWindowE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.20.1 宏定义说明

8.20.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.21 moc_mainwindow.cpp

浏览该文件的文档.

```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'mainwindow.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 ****************************************************************************/
00008
00009 #include "../../mainwindow.h"
00010 #include <QtGui/qtextcursor.h>
00011 #include <QtCore/qmetatype.h>
00012
00013 #include <QtCore/qtmochelpers.h>
00014
00015 #include <memory>
00016
00017
00018 #include <QtCore/qxptype_traits.h>
00019 #if !defined(Q_MOC_OUTPUT_REVISION)
```

```
00020 #error "The header file 'mainwindow.h' doesn't include <QObject>."
00021 #elif Q_MOC_OUTPUT_REVISION != 69
00022 #error "This file was generated using the moc from 6.9.0. It"
00023 #error "cannot be used with the include files from this version of Qt."
00024 #error "(The moc has changed too much.)"
00025 #endif
00026
00027 #ifndef Q_CONSTINIT
00028 #define Q_CONSTINIT
00029 #endif
00030
00031 QT_WARNING_PUSH
00032 QT_WARNING_DISABLE_DEPRECATED
00033 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00034 namespace {
00035 struct qt_meta_tag_ZN10MainWindowE_t {};
00036 } // unnamed namespace
00037
00038 template <> constexpr inline auto MainWindow::qt_create_metaobjectdata<qt_meta_tag_ZN10MainWindowE_t>()
00039 {
00040     namespace QMC = QtMocConstants;
00041     QtMocHelpers::StringRefStorage qt_stringData {
00042         "MainWindow",
00043         "startScan",
00044         "",
00045         "stopScan",
00046         "onHostFound",
00047         "HostInfo",
00048         "host",
00049         "onScanStarted",
00050         "onScanFinished",
00051         "onScanProgress",
00052         "progress",
00053         "onScanError",
00054         "errorMessage",
00055         "saveResults",
00056         "clearResults",
00057         "showSettings",
00058         "applySettings",
00059         "showAbout",
00060         "showHostDetails",
00061         "row",
00062         "column",
00063         "exportToCSV",
00064         "togglePortScanOptions",
00065         "checked",
00066         "toggleRangeOptions",
00067         "showTopologyView",
00068         "showStatisticsView",
00069         "showHistoryView",
00070         "generateSecurityReport",
00071         "saveTopologyImage",
00072         "toggleDarkMode",
00073         "enable",
00074         "compareScanResults",
00075         "scheduleScan",
00076         "saveHistoryToFile",
00077         "loadHistoryFromFile",
00078         "updateNetworkTopology",
00079         "refreshTopology",
00080         "filterResults",
00081         "clearFilters",
00082         "onThemeChanged"
00083     };
00084
00085     QtMocHelpers::UintData qt_methods {
00086         // Slot 'startScan'
00087         QtMocHelpers::SlotData<void()>(1, 2, QMC::AccessPrivate, QMetaType::Void),
00088         // Slot 'stopScan'
00089         QtMocHelpers::SlotData<void()>(3, 2, QMC::AccessPrivate, QMetaType::Void),
00090         // Slot 'onHostFound'
00091         QtMocHelpers::SlotData<void(const HostInfo &)>(4, 2, QMC::AccessPrivate, QMetaType::Void, {{
00092             { 0x80000000 | 5, 6 },
00093         }}),
00094         // Slot 'onScanStarted'
00095         QtMocHelpers::SlotData<void()>(7, 2, QMC::AccessPrivate, QMetaType::Void),
00096         // Slot 'onScanFinished'
00097         QtMocHelpers::SlotData<void()>(8, 2, QMC::AccessPrivate, QMetaType::Void),
00098         // Slot 'onScanProgress'
00099         QtMocHelpers::SlotData<void(int)>(9, 2, QMC::AccessPrivate, QMetaType::Void, {{
00100             { QMetaType::Int, 10 },
00101         }}),
00102         // Slot 'onScanError'
00103         QtMocHelpers::SlotData<void(const QString &)>(11, 2, QMC::AccessPrivate, QMetaType::Void, {{
00104             { QMetaType::QString, 12 },
00105         }}),
00106         // Slot 'saveResults'
```

```
00107         QtMocHelpers::SlotData<void()>(13, 2, QMC::AccessPrivate, QMetaType::Void),
00108         // Slot 'clearResults'
00109         QtMocHelpers::SlotData<void()>(14, 2, QMC::AccessPrivate, QMetaType::Void),
00110         // Slot 'showSettings'
00111         QtMocHelpers::SlotData<void()>(15, 2, QMC::AccessPrivate, QMetaType::Void),
00112         // Slot 'applySettings'
00113         QtMocHelpers::SlotData<void()>(16, 2, QMC::AccessPrivate, QMetaType::Void),
00114         // Slot 'showAbout'
00115         QtMocHelpers::SlotData<void()>(17, 2, QMC::AccessPrivate, QMetaType::Void),
00116         // Slot 'showHostDetails'
00117         QtMocHelpers::SlotData<void(int, int)>(18, 2, QMC::AccessPrivate, QMetaType::Void, {{
00118             { QMetaType::Int, 19 }, { QMetaType::Int, 20 },
00119         }}),
00120         // Slot 'exportToCSV'
00121         QtMocHelpers::SlotData<void()>(21, 2, QMC::AccessPrivate, QMetaType::Void),
00122         // Slot 'togglePortScanOptions'
00123         QtMocHelpers::SlotData<void(bool)>(22, 2, QMC::AccessPrivate, QMetaType::Void, {{
00124             { QMetaType::Bool, 23 },
00125         }}),
00126         // Slot 'toggleRangeOptions'
00127         QtMocHelpers::SlotData<void(bool)>(24, 2, QMC::AccessPrivate, QMetaType::Void, {{
00128             { QMetaType::Bool, 23 },
00129         }}),
00130         // Slot 'showTopologyView'
00131         QtMocHelpers::SlotData<void()>(25, 2, QMC::AccessPrivate, QMetaType::Void),
00132         // Slot 'showStatisticsView'
00133         QtMocHelpers::SlotData<void()>(26, 2, QMC::AccessPrivate, QMetaType::Void),
00134         // Slot 'showHistoryView'
00135         QtMocHelpers::SlotData<void()>(27, 2, QMC::AccessPrivate, QMetaType::Void),
00136         // Slot 'generateSecurityReport'
00137         QtMocHelpers::SlotData<void()>(28, 2, QMC::AccessPrivate, QMetaType::Void),
00138         // Slot 'saveTopologyImage'
00139         QtMocHelpers::SlotData<void()>(29, 2, QMC::AccessPrivate, QMetaType::Void),
00140         // Slot 'toggleDarkMode'
00141         QtMocHelpers::SlotData<void(bool)>(30, 2, QMC::AccessPrivate, QMetaType::Void, {{
00142             { QMetaType::Bool, 31 },
00143         }}),
00144         // Slot 'compareScanResults'
00145         QtMocHelpers::SlotData<void()>(32, 2, QMC::AccessPrivate, QMetaType::Void),
00146         // Slot 'scheduleScan'
00147         QtMocHelpers::SlotData<void()>(33, 2, QMC::AccessPrivate, QMetaType::Void),
00148         // Slot 'saveHistoryToFile'
00149         QtMocHelpers::SlotData<void()>(34, 2, QMC::AccessPrivate, QMetaType::Void),
00150         // Slot 'loadHistoryFromFile'
00151         QtMocHelpers::SlotData<void()>(35, 2, QMC::AccessPrivate, QMetaType::Void),
00152         // Slot 'updateNetworkTopology'
00153         QtMocHelpers::SlotData<void()>(36, 2, QMC::AccessPrivate, QMetaType::Void),
00154         // Slot 'refreshTopology'
00155         QtMocHelpers::SlotData<void()>(37, 2, QMC::AccessPrivate, QMetaType::Void),
00156         // Slot 'filterResults'
00157         QtMocHelpers::SlotData<void()>(38, 2, QMC::AccessPrivate, QMetaType::Void),
00158         // Slot 'clearFilters'
00159         QtMocHelpers::SlotData<void()>(39, 2, QMC::AccessPrivate, QMetaType::Void),
00160         // Slot 'onThemeChanged'
00161         QtMocHelpers::SlotData<void()>(40, 2, QMC::AccessPrivate, QMetaType::Void),
00162     };
00163     QtMocHelpers::UintData qt_properties {
00164     };
00165     QtMocHelpers::UintData qt_enums {
00166     };
00167     return QtMocHelpers::metaObjectData<MainWindow,
    qt_meta_tag_ZN10MainWindowE_t>(QMC::MetaObjectFlag{}, qt_stringData,
00168             qt_methods, qt_properties, qt_enums);
00169 }
00170 Q_CONSTINIT const QMetaObject MainWindow::staticMetaObject = { {
00171     QMetaObject::SuperData::link<QMainWindow::staticMetaObject>(),
00172     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN10MainWindowE_t>.stringdata,
00173     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN10MainWindowE_t>.data,
00174     qt_static_metacall,
00175     nullptr,
00176     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN10MainWindowE_t>.metaTypes,
00177     nullptr
00178 } };
00179
00180 void MainWindow::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00181 {
00182     auto *_t = static_cast<MainWindow *>(_o);
00183     if (_c == QMetaObject::InvokeMetaMethod) {
00184         switch (_id) {
00185         case 0: _t->startScan(); break;
00186         case 1: _t->stopScan(); break;
00187         case 2: _t->onHostFound((*reinterpret_cast< std::add_pointer_t<HostInfo>>(_a[1]))); break;
00188         case 3: _t->onScanStarted(); break;
00189         case 4: _t->onScanFinished(); break;
00190         case 5: _t->onScanProgress((*reinterpret_cast< std::add_pointer_t<int>>(_a[1]))); break;
00191         case 6: _t->onScanError((*reinterpret_cast< std::add_pointer_t<QString>>(_a[1]))); break;
00192         case 7: _t->saveResults(); break;
```

```
00193         case 8: _t->clearResults(); break;
00194         case 9: _t->showSettings(); break;
00195         case 10: _t->applySettings(); break;
00196         case 11: _t->showAbout(); break;
00197         case 12: _t->showHostDetails((*reinterpret_cast< std::add_pointer_t<int>>(_a[1])),(*reinterpret_cast<
      std::add_pointer_t<int>>(_a[2]))); break;
00198         case 13: _t->exportToCSV(); break;
00199         case 14: _t->togglePortScanOptions((*reinterpret_cast< std::add_pointer_t<bool>>(_a[1]))); break;
00200         case 15: _t->toggleRangeOptions((*reinterpret_cast< std::add_pointer_t<bool>>(_a[1]))); break;
00201         case 16: _t->showTopologyView(); break;
00202         case 17: _t->showStatisticsView(); break;
00203         case 18: _t->showHistoryView(); break;
00204         case 19: _t->generateSecurityReport(); break;
00205         case 20: _t->saveTopologyImage(); break;
00206         case 21: _t->toggleDarkMode((*reinterpret_cast< std::add_pointer_t<bool>>(_a[1]))); break;
00207         case 22: _t->compareScanResults(); break;
00208         case 23: _t->scheduleScan(); break;
00209         case 24: _t->saveHistoryToFile(); break;
00210         case 25: _t->loadHistoryFromFile(); break;
00211         case 26: _t->updateNetworkTopology(); break;
00212         case 27: _t->refreshTopology(); break;
00213         case 28: _t->filterResults(); break;
00214         case 29: _t->clearFilters(); break;
00215         case 30: _t->onThemeChanged(); break;
00216         default: ;
00217         }
00218     }
00219 }
00220
00221 const QMetaObject *MainWindow::metaObject() const
00222 {
00223     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00224 }
00225
00226 void *MainWindow::qt_metacast(const char *_clname)
00227 {
00228     if (!_clname) return nullptr;
00229     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN10MainWindowE_t>.strings))
00230         return static_cast<void*>(this);
00231     return QMainWindow::qt_metacast(_clname);
00232 }
00233
00234 int MainWindow::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00235 {
00236     _id = QMainWindow::qt_metacall(_c, _id, _a);
00237     if (_id < 0)
00238         return _id;
00239     if (_c == QMetaObject::InvokeMetaMethod) {
00240         if (_id < 31)
00241             qt_static_metacall(this, _c, _id, _a);
00242         _id -= 31;
00243     }
00244     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00245         if (_id < 31)
00246             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00247         _id -= 31;
00248     }
00249     return _id;
00250 }
00251 QT_WARNING_POP
```

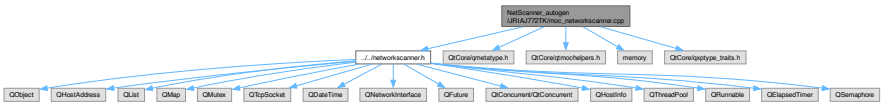
8.22 NetScanner_autogen/EWIEGA46WW/moc_mainwindow.cpp.d 文件参考

8.23 NetScanner_autogen/JRIAJ772TK/moc_mainwindow.cpp.d 文件参考

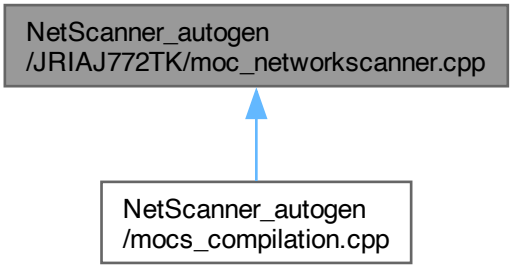
8.24 NetScanner_autogen/JRIAJ772TK/moc_networkscanner.cpp 文件参考

```
#include "../../networkscanner.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
```

moc_networkscanner.cpp 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN14NetworkScannerE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.24.1 宏定义说明

8.24.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.25 moc_networkscanner.cpp

浏览该文件的文档.

```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'networkscanner.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****************************************************************************/
00008
00009 #include "../../networkscanner.h"
00010 #include <QtCore/qmetatype.h>
00011
00012 #include <QtCore/qtmochelpers.h>
00013
00014 #include <memory>
00015
00016
```

```
00017 #include <QtCore/qxptype_traits.h>
00018 #if !defined(Q_MOC_OUTPUT_REVISION)
00019 #error "The header file 'networkscanner.h' doesn't include <QObject>."
00020 #elif Q_MOC_OUTPUT_REVISION != 69
00021 #error "This file was generated using the moc from 6.9.0. It"
00022 #error "cannot be used with the include files from this version of Qt."
00023 #error "(The moc has changed too much.)"
00024 #endif
00025
00026 #ifndef Q_CONSTINIT
00027 #define Q_CONSTINIT
00028 #endif
00029
00030 QT_WARNING_PUSH
00031 QT_WARNING_DISABLE_DEPRECATED
00032 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00033 namespace {
00034 struct qt_meta_tag_ZN14NetworkScannerE_t {};
00035 } // unnamed namespace
00036
00037 template <> constexpr inline auto
      NetworkScanner::qt_create_metaobjectdata<qt_meta_tag_ZN14NetworkScannerE_t>()
00038 {
00039     namespace QMC = QtMocConstants;
00040     QtMocHelpers::StringRefStorage qt_stringData {
00041         "NetworkScanner",
00042         "scanStarted",
00043         "",
00044         "scanFinished",
00045         "hostFound",
00046         "HostInfo",
00047         "host",
00048         "scanProgress",
00049         "progress",
00050         "scanError",
00051         "errorMessage",
00052         "processScanResults"
00053     };
00054
00055     QtMocHelpers::UintData qt_methods {
00056         // Signal 'scanStarted'
00057         QtMocHelpers::SignalData<void()>(1, 2, QMC::AccessPublic, QMetaType::Void),
00058         // Signal 'scanFinished'
00059         QtMocHelpers::SignalData<void()>(3, 2, QMC::AccessPublic, QMetaType::Void),
00060         // Signal 'hostFound'
00061         QtMocHelpers::SignalData<void(const HostInfo &)>(4, 2, QMC::AccessPublic, QMetaType::Void, {{
00062             { 0x80000000 | 5, 6 },
00063         }}),
00064         // Signal 'scanProgress'
00065         QtMocHelpers::SignalData<void(int)>(7, 2, QMC::AccessPublic, QMetaType::Void, {{
00066             { QMetaType::Int, 8 },
00067         }}),
00068         // Signal 'scanError'
00069         QtMocHelpers::SignalData<void(const QString &)>(9, 2, QMC::AccessPublic, QMetaType::Void, {{
00070             { QMetaType::QString, 10 },
00071         }}),
00072         // Slot 'processScanResults'
00073         QtMocHelpers::SlotData<void()>(11, 2, QMC::AccessPrivate, QMetaType::Void),
00074     };
00075     QtMocHelpers::UintData qt_properties {
00076     };
00077     QtMocHelpers::UintData qt_enums {
00078     };
00079     return QtMocHelpers::metaObjectData<NetworkScanner,
      qt_meta_tag_ZN14NetworkScannerE_t>(QMC::MetaObjectFlag{}, qt_stringData,
00080             qt_methods, qt_properties, qt_enums);
00081 }
00082 Q_CONSTINIT const QMetaObject NetworkScanner::staticMetaObject = { {
00083     QMetaObject::SuperData::link<QObject::staticMetaObject>(),
00084     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14NetworkScannerE_t>.stringdata,
00085     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14NetworkScannerE_t>.data,
00086     qt_static_metacall,
00087     nullptr,
00088     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN14NetworkScannerE_t>.metaTypes,
00089     nullptr
00090 } };
00091
00092 void NetworkScanner::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00093 {
00094     auto *_t = static_cast<NetworkScanner *>(_o);
00095     if (_c == QMetaObject::InvokeMetaMethod) {
00096         switch (_id) {
00097         case 0: _t->scanStarted(); break;
00098         case 1: _t->scanFinished(); break;
00099         case 2: _t->hostFound((*reinterpret_cast< std::add_pointer_t<HostInfo>>(_a[1]))); break;
00100         case 3: _t->scanProgress((*reinterpret_cast< std::add_pointer_t<int>>(_a[1]))); break;
00101         case 4: _t->scanError((*reinterpret_cast< std::add_pointer_t<QString>>(_a[1]))); break;
```

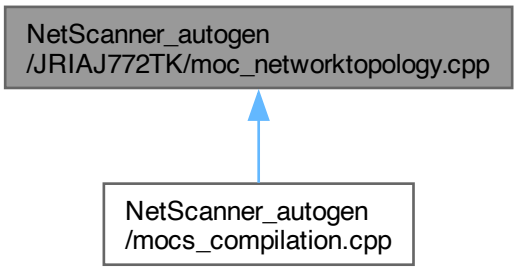
```
00102         case 5: _t->processScanResults(); break;
00103         default: ;
00104         }
00105     }
00106     if (_c == QMetaObject::IndexOfMethod) {
00107         if (QtMocHelpers::indexOfMethod<void (NetworkScanner::*)()>(_a, &NetworkScanner::scanStarted, 0))
00108             return;
00109         if (QtMocHelpers::indexOfMethod<void (NetworkScanner::*)()>(_a, &NetworkScanner::scanFinished, 1))
00110             return;
00111         if (QtMocHelpers::indexOfMethod<void (NetworkScanner::*)(const HostInfo & )>(_a,
    &NetworkScanner::hostFound, 2))
00112             return;
00113         if (QtMocHelpers::indexOfMethod<void (NetworkScanner::*)(int )>(_a, &NetworkScanner::scanProgress, 3))
00114             return;
00115         if (QtMocHelpers::indexOfMethod<void (NetworkScanner::*)(const QString & )>(_a, &NetworkScanner::scanError,
    4))
00116             return;
00117     }
00118 }
00119
00120 const QMetaObject *NetworkScanner::metaObject() const
00121 {
00122     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00123 }
00124
00125 void *NetworkScanner::qt_metacast(const char *_clname)
00126 {
00127     if (!_clname) return nullptr;
00128     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN14NetworkScannerE_t>.strings))
00129         return static_cast<void*>(this);
00130     return QObject::qt_metacast(_clname);
00131 }
00132
00133 int NetworkScanner::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00134 {
00135     _id = QObject::qt_metacall(_c, _id, _a);
00136     if (_id < 0)
00137         return _id;
00138     if (_c == QMetaObject::InvokeMetaMethod) {
00139         if (_id < 6)
00140             qt_static_metacall(this, _c, _id, _a);
00141         _id -= 6;
00142     }
00143     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00144         if (_id < 6)
00145             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00146         _id -= 6;
00147     }
00148     return _id;
00149 }
00150
00151 // SIGNAL 0
00152 void NetworkScanner::scanStarted()
00153 {
00154     QMetaObject::activate(this, &staticMetaObject, 0, nullptr);
00155 }
00156
00157 // SIGNAL 1
00158 void NetworkScanner::scanFinished()
00159 {
00160     QMetaObject::activate(this, &staticMetaObject, 1, nullptr);
00161 }
00162
00163 // SIGNAL 2
00164 void NetworkScanner::hostFound(const HostInfo & _t1)
00165 {
00166     QMetaObject::activate<void>(this, &staticMetaObject, 2, nullptr, _t1);
00167 }
00168
00169 // SIGNAL 3
00170 void NetworkScanner::scanProgress(int _t1)
00171 {
00172     QMetaObject::activate<void>(this, &staticMetaObject, 3, nullptr, _t1);
00173 }
00174
00175 // SIGNAL 4
00176 void NetworkScanner::scanError(const QString & _t1)
00177 {
00178     QMetaObject::activate<void>(this, &staticMetaObject, 4, nullptr, _t1);
00179 }
00180 QT_WARNING_POP
```

8.26 NetScanner_autogen/JRIAJ772TK/moc_networkscanner.cpp.d 文件参考

8.27 NetScanner_autogen/JRIAJ772TK/moc_networktopology.cpp 文件参考

#include "../../networktopology.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
moc_networktopology.cpp 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN19NetworkTopologyViewE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.27.1 宏定义说明

8.27.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.28 moc_networktopology.cpp

浏览该文件的文档.

```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'networktopology.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****************************************************************************/
00008
00009 #include "../../networktopology.h"
00010 #include <QtCore/qmetatype.h>
00011
00012 #include <QtCore/qtmochelpers.h>
00013
00014 #include <memory>
00015
00016
00017 #include <QtCore/qxptype_traits.h>
00018 #if !defined(Q_MOC_OUTPUT_REVISION)
00019 #error "The header file 'networktopology.h' doesn't include <QObject>."
00020 #elif Q_MOC_OUTPUT_REVISION != 69
00021 #error "This file was generated using the moc from 6.9.0. It"
00022 #error "cannot be used with the include files from this version of Qt."
00023 #error "(The moc has changed too much.)"
00024 #endif
00025
00026 #ifndef Q_CONSTINIT
00027 #define Q_CONSTINIT
00028 #endif
00029
00030 QT_WARNING_PUSH
00031 QT_WARNING_DISABLE_DEPRECATED
00032 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00033 namespace {
00034 struct qt_meta_tag_ZN19NetworkTopologyViewE_t {};
00035 } // unnamed namespace
00036
00037 template <> constexpr inline auto
      NetworkTopologyView::qt_create_metaobjectdata<qt_meta_tag_ZN19NetworkTopologyViewE_t>()
00038 {
00039     namespace QMC = QtMocConstants;
00040     QtMocHelpers::StringRefStorage qt_stringData {
00041         "NetworkTopologyView",
00042         "nodeSelected",
00043         "",
00044         "HostInfo",
00045         "host"
00046     };
00047
00048     QtMocHelpers::UintData qt_methods {
00049         // Signal 'nodeSelected'
00050         QtMocHelpers::SignalData<void(const HostInfo &)>(1, 2, QMC::AccessPublic, QMetaType::Void, {{
00051             { 0x80000000 | 3, 4 },
00052         }}),
00053     };
00054     QtMocHelpers::UintData qt_properties {
00055     };
00056     QtMocHelpers::UintData qt_enums {
00057     };
00058     return QtMocHelpers::metaObjectData<NetworkTopologyView,
      qt_meta_tag_ZN19NetworkTopologyViewE_t>(QMC::MetaObjectFlag{}, qt_stringData,
00059             qt_methods, qt_properties, qt_enums);
00060 }
00061 Q_CONSTINIT const QMetaObject NetworkTopologyView::staticMetaObject = { {
00062     QMetaObject::SuperData::link<QGraphicsView::staticMetaObject>(),
00063     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN19NetworkTopologyViewE_t>.stringdata,
00064     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN19NetworkTopologyViewE_t>.data,
00065     qt_static_metacall,
00066     nullptr,
00067     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN19NetworkTopologyViewE_t>.metaTypes,
00068     nullptr
00069 } };
00070
00071 void NetworkTopologyView::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00072 {
00073     auto *_t = static_cast<NetworkTopologyView *>(_o);
00074     if (_c == QMetaObject::InvokeMetaMethod) {
00075         switch (_id) {
00076         case 0: _t->nodeSelected((*reinterpret_cast< std::add_pointer_t<HostInfo>>(_a[1]))); break;
00077         default: ;
00078         }
00079     }
00080     if (_c == QMetaObject::IndexOfMethod) {
00081         if (QtMocHelpers::indexOfMethod<void (NetworkTopologyView::*)(const HostInfo & )>(_a,
```

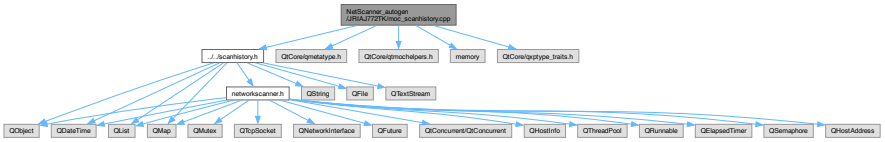
```
    &NetworkTopologyView::nodeSelected, 0))
00082         return;
00083     }
00084 }
00085
00086 const QMetaObject *NetworkTopologyView::metaObject() const
00087 {
00088     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00089 }
00090
00091 void *NetworkTopologyView::qt_metacast(const char *_clname)
00092 {
00093     if (!_clname) return nullptr;
00094     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN19NetworkTopologyViewE_t>.strings))
00095         return static_cast<void*>(this);
00096     return QGraphicsView::qt_metacast(_clname);
00097 }
00098
00099 int NetworkTopologyView::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00100 {
00101     _id = QGraphicsView::qt_metacall(_c, _id, _a);
00102     if (_id < 0)
00103         return _id;
00104     if (_c == QMetaObject::InvokeMetaMethod) {
00105         if (_id < 1)
00106             qt_static_metacall(this, _c, _id, _a);
00107         _id -= 1;
00108     }
00109     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00110         if (_id < 1)
00111             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00112         _id -= 1;
00113     }
00114     return _id;
00115 }
00116
00117 // SIGNAL 0
00118 void NetworkTopologyView::nodeSelected(const HostInfo & _t1)
00119 {
00120     QMetaObject::activate<void>(this, &staticMetaObject, 0, nullptr, _t1);
00121 }
00122 namespace {
00123 struct qt_meta_tag_ZN15NetworkTopologyE_t {};
00124 } // unnamed namespace
00125
00126 template <> constexpr inline auto
    NetworkTopology::qt_create_metaobjectdata<qt_meta_tag_ZN15NetworkTopologyE_t>()
00127 {
00128     namespace QMC = QtMocConstants;
00129     QtMocHelpers::StringRefStorage qt_stringData {
00130         "NetworkTopology",
00131         "deviceSelected",
00132         "",
00133         "HostInfo",
00134         "host"
00135     };
00136
00137     QtMocHelpers::UintData qt_methods {
00138         // Signal 'deviceSelected'
00139         QtMocHelpers::SignalData<void(const HostInfo &)>(1, 2, QMC::AccessPublic, QMetaType::Void, {{
00140             { 0x80000000 | 3, 4 },
00141         }}),
00142     };
00143     QtMocHelpers::UintData qt_properties {
00144     };
00145     QtMocHelpers::UintData qt_enums {
00146     };
00147     return QtMocHelpers::metaObjectData<NetworkTopology,
    qt_meta_tag_ZN15NetworkTopologyE_t>(QMC::MetaObjectFlag{}, qt_stringData,
00148             qt_methods, qt_properties, qt_enums);
00149 }
00150 Q_CONSTINIT const QMetaObject NetworkTopology::staticMetaObject = { {
00151     QMetaObject::SuperData::link<QWidget::staticMetaObject>(),
00152     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN15NetworkTopologyE_t>.stringdata,
00153     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN15NetworkTopologyE_t>.data,
00154     qt_static_metacall,
00155     nullptr,
00156     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN15NetworkTopologyE_t>.metaTypes,
00157     nullptr
00158 } };
00159
00160 void NetworkTopology::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00161 {
00162     auto *_t = static_cast<NetworkTopology *>(_o);
00163     if (_c == QMetaObject::InvokeMetaMethod) {
00164         switch (_id) {
00165         case 0: _t->deviceSelected((*reinterpret_cast< std::add_pointer_t<HostInfo>>(_a[1]))); break;
```

```
00166         default: ;
00167         }
00168     }
00169     if (_c == QMetaObject::IndexOfMethod) {
00170         if (QtMocHelpers::indexOfMethod<void (NetworkTopology::*)(const HostInfo & )>(_a,
      &NetworkTopology::deviceSelected, 0))
00171             return;
00172     }
00173 }
00174
00175 const QMetaObject *NetworkTopology::metaObject() const
00176 {
00177     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00178 }
00179
00180 void *NetworkTopology::qt_metacast(const char *_clname)
00181 {
00182     if (!_clname) return nullptr;
00183     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN15NetworkTopologyE_t>.strings))
00184         return static_cast<void*>(this);
00185     return QWidget::qt_metacast(_clname);
00186 }
00187
00188 int NetworkTopology::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00189 {
00190     _id = QWidget::qt_metacall(_c, _id, _a);
00191     if (_id < 0)
00192         return _id;
00193     if (_c == QMetaObject::InvokeMetaMethod) {
00194         if (_id < 1)
00195             qt_static_metacall(this, _c, _id, _a);
00196         _id -= 1;
00197     }
00198     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00199         if (_id < 1)
00200             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00201         _id -= 1;
00202     }
00203     return _id;
00204 }
00205
00206 // SIGNAL 0
00207 void NetworkTopology::deviceSelected(const HostInfo & _t1)
00208 {
00209     QMetaObject::activate<void>(this, &staticMetaObject, 0, nullptr, _t1);
00210 }
00211 QT_WARNING_POP
```

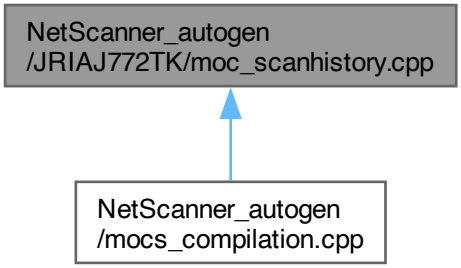
8.29 NetScanner_autogen/JRIAJ772TK/moc_networktopology.cpp.d 文件参考

8.30 NetScanner_autogen/JRIAJ772TK/moc_scanhistory.cpp 文件参考

#include "../../scanhistory.h"
#include <QtCore/qmetatype.h>
#include <QtCore/qtmochelpers.h>
#include <memory>
#include <QtCore/qxptype_traits.h>
moc_scanhistory.cpp 的引用 (Include) 关系图:



此图展示该文件被哪些文件直接或间接地引用了:



类

- struct QT_WARNING_DISABLE_DEPRECATED::qt_meta_tag_ZN11ScanHistoryE_t

命名空间

- namespace QT_WARNING_DISABLE_DEPRECATED

宏定义

- #define Q_CONSTINIT

8.30.1 宏定义说明

8.30.1.1 Q_CONSTINIT

#define Q_CONSTINIT

8.31 moc_scanhistory.cpp

浏览该文件的文档.

```
00001 /****************************************************************************
00002 ** Meta object code from reading C++ file 'scanhistory.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 69 (Qt 6.9.0)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****************************************************************************/
00008
00009 #include "../../scanhistory.h"
00010 #include <QtCore/qmetatype.h>
00011
00012 #include <QtCore/qtmochelpers.h>
00013
00014 #include <memory>
00015
00016
00017 #include <QtCore/qxptype_traits.h>
00018 #if !defined(Q_MOC_OUTPUT_REVISION)
00019 #error "The header file 'scanhistory.h' doesn't include <QObject>."
00020 #elif Q_MOC_OUTPUT_REVISION != 69
00021 #error "This file was generated using the moc from 6.9.0. It"
00022 #error "cannot be used with the include files from this version of Qt."
00023 #error "(The moc has changed too much.)"
00024 #endif
00025
00026 #ifndef Q_CONSTINIT
00027 #define Q_CONSTINIT
00028 #endif
00029
```

```
00030 QT_WARNING_PUSH
00031 QT_WARNING_DISABLE_DEPRECATED
00032 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00033 namespace {
00034 struct qt_meta_tag_ZN11ScanHistoryE_t {};
00035 } // unnamed namespace
00036
00037 template <> constexpr inline auto ScanHistory::qt_create_metaobjectdata<qt_meta_tag_ZN11ScanHistoryE_t>()
00038 {
00039     namespace QMC = QtMocConstants;
00040     QtMocHelpers::StringRefStorage qt_stringData {
00041         "ScanHistory",
00042         "historyChanged",
00043         ""
00044     };
00045
00046     QtMocHelpers::UintData qt_methods {
00047         // Signal 'historyChanged'
00048         QtMocHelpers::SignalData<void()>(1, 2, QMC::AccessPublic, QMetaType::Void),
00049     };
00050     QtMocHelpers::UintData qt_properties {
00051     };
00052     QtMocHelpers::UintData qt_enums {
00053     };
00054     return QtMocHelpers::metaObjectData<ScanHistory, qt_meta_tag_ZN11ScanHistoryE_t>(QMC::MetaObjectFlag{},
    qt_stringData,
00055             qt_methods, qt_properties, qt_enums);
00056 }
00057 Q_CONSTINIT const QMetaObject ScanHistory::staticMetaObject = { {
00058     QMetaObject::SuperData::link<QObject::staticMetaObject>(),
00059     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN11ScanHistoryE_t>.stringdata,
00060     qt_staticMetaObjectStaticContent<qt_meta_tag_ZN11ScanHistoryE_t>.data,
00061     qt_static_metacall,
00062     nullptr,
00063     qt_staticMetaObjectRelocatingContent<qt_meta_tag_ZN11ScanHistoryE_t>.metaTypes,
00064     nullptr
00065 } };
00066
00067 void ScanHistory::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00068 {
00069     auto *_t = static_cast<ScanHistory *>(_o);
00070     if (_c == QMetaObject::InvokeMetaMethod) {
00071         switch (_id) {
00072         case 0: _t->historyChanged(); break;
00073         default: ;
00074         }
00075     }
00076     if (_c == QMetaObject::IndexOfMethod) {
00077         if (QtMocHelpers::indexOfMethod<void (ScanHistory::*)()>(_a, &ScanHistory::historyChanged, 0))
00078             return;
00079     }
00080 }
00081
00082 const QMetaObject *ScanHistory::metaObject() const
00083 {
00084     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00085 }
00086
00087 void *ScanHistory::qt_metacast(const char *_clname)
00088 {
00089     if (!_clname) return nullptr;
00090     if (!strcmp(_clname, qt_staticMetaObjectStaticContent<qt_meta_tag_ZN11ScanHistoryE_t>.strings))
00091         return static_cast<void*>(this);
00092     return QObject::qt_metacast(_clname);
00093 }
00094
00095 int ScanHistory::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00096 {
00097     _id = QObject::qt_metacall(_c, _id, _a);
00098     if (_id < 0)
00099         return _id;
00100     if (_c == QMetaObject::InvokeMetaMethod) {
00101         if (_id < 1)
00102             qt_static_metacall(this, _c, _id, _a);
00103         _id -= 1;
00104     }
00105     if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00106         if (_id < 1)
00107             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00108         _id -= 1;
00109     }
00110     return _id;
00111 }
00112
00113 // SIGNAL 0
00114 void ScanHistory::historyChanged()
00115 {
```

```
00116     QMetaObject::activate(this, &staticMetaObject, 0, nullptr);
00117 }
00118 QT_WARNING_POP
```

8.32 NetScanner_autogen/JRIAJ772TK/moc_scanhistory.cpp.d 文件参考

8.33 NetScanner_autogen/moc_predefs.h 文件参考

宏定义

- #define QT_CHARTS_LIB 1
- #define QT_CORE_LIB 1
- #define QT_GUI_LIB 1
- #define QT_NETWORK_LIB 1
- #define QT_NO_DEBUG 1
- #define QT_OPENGLWIDGETS_LIB 1
- #define QT_OPENGL_LIB 1
- #define QT_WIDGETS_LIB 1
- #define SIZEOF_DPTR (sizeof(void*))
- #define TARGET_IPHONE_SIMULATOR 0
- #define TARGET_OS_ARROW 1
- #define TARGET_OS_BRIDGE 0
- #define TARGET_OS_DRIVERKIT 0
- #define TARGET_OS_EMBEDDED 0
- #define TARGET_OS_IOS 0
- #define TARGET_OS_IOSMAC 0
- #define TARGET_OS_IPHONE 0
- #define TARGET_OS_LINUX 0
- #define TARGET_OS_MAC 1
- #define TARGET_OS_MACCATALYST 0
- #define TARGET_OS_NANO 0
- #define TARGET_OS_OSX 1
- #define TARGET_OS_SIMULATOR 0
- #define TARGET_OS_TV 0
- #define TARGET_OS_UIKITFORMAC 0
- #define TARGET_OS_UNIX 0
- #define TARGET_OS_VISION 0
- #define TARGET_OS_WATCH 0
- #define TARGET_OS_WIN32 0
- #define TARGET_OS_WINDOWS 0
- #define TARGET_OS_XR 0
- #define _LP64 1
- #define __AARCH64EL__ 1
- #define __AARCH64_CMODEL_SMALL__ 1
- #define __AARCH64_SIMD__ 1
- #define __APPLE_CC__ 6000
- #define __APPLE__ 1
- #define __ARM64_ARCH_8__ 1
- #define __ARM_64BIT_STATE 1
- #define __ARM_ACLE 200
- #define __ARM_ALIGN_MAX_STACK_PWR 4
- #define __ARM_ARCH 8
- #define __ARM_ARCH_8_3__ 1
- #define __ARM_ARCH_8_4__ 1

- #define __ARM_ARCH_8_5__ 1
- #define __ARM_ARCH_ISA_A64 1
- #define __ARM_ARCH_PROFILE 'A'
- #define __ARM_FEATURE_AES 1
- #define __ARM_FEATURE_ATOMICS 1
- #define __ARM_FEATURE_BTI 1
- #define __ARM_FEATURE_CLZ 1
- #define __ARM_FEATURE_COMPLEX 1
- #define __ARM_FEATURE_CRC32 1
- #define __ARM_FEATURE_CRYPTO 1
- #define __ARM_FEATURE_DIRECTED_ROUNDING 1
- #define __ARM_FEATURE_DIV 1
- #define __ARM_FEATURE_DOTPROD 1
- #define __ARM_FEATURE_FMA 1
- #define __ARM_FEATURE_FP16_FML 1
- #define __ARM_FEATURE_FP16_SCALAR_ARITHMETIC 1
- #define __ARM_FEATURE_FP16_VECTOR_ARITHMETIC 1
- #define __ARM_FEATURE_FRINT 1
- #define __ARM_FEATURE_IDIV 1
- #define __ARM_FEATURE_JCVT 1
- #define __ARM_FEATURE_LDREX 0xF
- #define __ARM_FEATURE_NUMERIC_MAXMIN 1
- #define __ARM_FEATURE_PAUTH 1
- #define __ARM_FEATURE_QRDMX 1
- #define __ARM_FEATURE_RCPC 1
- #define __ARM_FEATURE_SHA2 1
- #define __ARM_FEATURE_SHA3 1
- #define __ARM_FEATURE_SHA512 1
- #define __ARM_FEATURE_UNALIGNED 1
- #define __ARM_FP 0xE
- #define __ARM_FP16_ARGS 1
- #define __ARM_FP16_FORMAT_IEEE 1
- #define __ARM_NEON 1
- #define __ARM_NEON_FP 0xE
- #define __ARM_NEON__ 1
- #define __ARM_PCS_AAPCS64 1
- #define __ARM_SIZEOF_MINIMAL_ENUM 4
- #define __ARM_SIZEOF_WCHAR_T 4
- #define __ARM_STATE_ZA 1
- #define __ARM_STATE_ZT0 1
- #define __ATOMIC_ACQUIRE 2
- #define __ATOMIC_ACQ_REL 4
- #define __ATOMIC_CONSUME 1
- #define __ATOMIC_RELAXED 0
- #define __ATOMIC_RELEASE 3
- #define __ATOMIC_SEQ_CST 5
- #define __BIGGEST_ALIGNMENT__ 8
- #define __BITINT_MAXWIDTH__ 128
- #define __BLOCKS__ 1
- #define __BOOL_WIDTH__ 8
- #define __BYTE_ORDER__ __ORDER_LITTLE_ENDIAN__
- #define __CHAR16_TYPE__ unsigned short
- #define __CHAR32_TYPE__ unsigned int
- #define __CHAR_BIT__ 8
- #define __CLANG_ATOMIC_BOOL_LOCK_FREE 2

- #define __CLANG_ATOMIC_CHAR16_T_LOCK_FREE 2
- #define __CLANG_ATOMIC_CHAR32_T_LOCK_FREE 2
- #define __CLANG_ATOMIC_CHAR_LOCK_FREE 2
- #define __CLANG_ATOMIC_INT_LOCK_FREE 2
- #define __CLANG_ATOMIC_LLONG_LOCK_FREE 2
- #define __CLANG_ATOMIC_LONG_LOCK_FREE 2
- #define __CLANG_ATOMIC_POINTER_LOCK_FREE 2
- #define __CLANG_ATOMIC_SHORT_LOCK_FREE 2
- #define __CLANG_ATOMIC_WCHAR_T_LOCK_FREE 2
- #define __CONSTANT_CFSTRINGS__ 1
- #define __DBL_DECIMAL_DIG__ 17
- #define __DBL_DENORM_MIN__ 4.9406564584124654e-324
- #define __DBL_DIG__ 15
- #define __DBL_EPSILON__ 2.2204460492503131e-16
- #define __DBL_HAS_DENORM__ 1
- #define __DBL_HAS_INFINITY__ 1
- #define __DBL_HAS_QUIET_NAN__ 1
- #define __DBL_MANT_DIG__ 53
- #define __DBL_MAX_10_EXP__ 308
- #define __DBL_MAX_EXP__ 1024
- #define __DBL_MAX__ 1.7976931348623157e+308
- #define __DBL_MIN_10_EXP__ (-307)
- #define __DBL_MIN_EXP__ (-1021)
- #define __DBL_MIN__ 2.2250738585072014e-308
- #define __DBL_NORM_MAX__ 1.7976931348623157e+308
- #define __DECIMAL_DIG__ __LDBL_DECIMAL_DIG__
- #define __DEPRECATED 1
- #define __DYNAMIC__ 1
- #define __ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ 150000
- #define __ENVIRONMENT_OS_VERSION_MIN_REQUIRED__ 150000
- #define __EXCEPTIONS 1
- #define __FINITE_MATH_ONLY__ 0
- #define __FLT16_DECIMAL_DIG__ 5
- #define __FLT16_DENORM_MIN__ 5.960464477539062e-8F16
- #define __FLT16_DIG__ 3
- #define __FLT16_EPSILON__ 9.765625e-4F16
- #define __FLT16_HAS_DENORM__ 1
- #define __FLT16_HAS_INFINITY__ 1
- #define __FLT16_HAS_QUIET_NAN__ 1
- #define __FLT16_MANT_DIG__ 11
- #define __FLT16_MAX_10_EXP__ 4
- #define __FLT16_MAX_EXP__ 16
- #define __FLT16_MAX__ 6.5504e+4F16
- #define __FLT16_MIN_10_EXP__ (-4)
- #define __FLT16_MIN_EXP__ (-13)
- #define __FLT16_MIN__ 6.103515625e-5F16
- #define __FLT16_NORM_MAX__ 6.5504e+4F16
- #define __FLT_DECIMAL_DIG__ 9
- #define __FLT_DENORM_MIN__ 1.40129846e-45F
- #define __FLT_DIG__ 6
- #define __FLT_EPSILON__ 1.19209290e-7F
- #define __FLT_HAS_DENORM__ 1
- #define __FLT_HAS_INFINITY__ 1
- #define __FLT_HAS_QUIET_NAN__ 1
- #define __FLT_MANT_DIG__ 24

- #define __FLT_MAX_10_EXP__ 38
- #define __FLT_MAX_EXP__ 128
- #define __FLT_MAX__ 3.40282347e+38F
- #define __FLT_MIN_10_EXP__ (-37)
- #define __FLT_MIN_EXP__ (-125)
- #define __FLT_MIN__ 1.17549435e-38F
- #define __FLT_NORM_MAX__ 3.40282347e+38F
- #define __FLT_RADIX__ 2
- #define __FPCLASS_NEGINF 0x0004
- #define __FPCLASS_NEGNORMAL 0x0008
- #define __FPCLASS_NEGSUBNORMAL 0x0010
- #define __FPCLASS_NEGZERO 0x0020
- #define __FPCLASS_POSINF 0x0200
- #define __FPCLASS_POSNORMAL 0x0100
- #define __FPCLASS_POSSUBNORMAL 0x0080
- #define __FPCLASS_POSZERO 0x0040
- #define __FPCLASS_QNAN 0x0002
- #define __FPCLASS_SNAN 0x0001
- #define __FP_FAST_FMA 1
- #define __FP_FAST_FMAF 1
- #define __GCC_ASM_FLAG_OUTPUTS__ 1
- #define __GCC_ATOMIC_BOOL_LOCK_FREE 2
- #define __GCC_ATOMIC_CHAR16_T_LOCK_FREE 2
- #define __GCC_ATOMIC_CHAR32_T_LOCK_FREE 2
- #define __GCC_ATOMIC_CHAR_LOCK_FREE 2
- #define __GCC_ATOMIC_INT_LOCK_FREE 2
- #define __GCC_ATOMIC_LLONG_LOCK_FREE 2
- #define __GCC_ATOMIC_LONG_LOCK_FREE 2
- #define __GCC_ATOMIC_POINTER_LOCK_FREE 2
- #define __GCC_ATOMIC_SHORT_LOCK_FREE 2
- #define __GCC_ATOMIC_TEST_AND_SET_TRUEVAL 1
- #define __GCC_ATOMIC_WCHAR_T_LOCK_FREE 2
- #define __GCC_CONSTRUCTIVE_SIZE 64
- #define __GCC_DESTRUCTIVE_SIZE 64
- #define __GCC_HAVE_DWARF2_CFI_ASM 1
- #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_1 1
- #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_16 1
- #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_2 1
- #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_4 1
- #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_8 1
- #define __GLIBCXX_BITSIZE_INT_N_0 128
- #define __GLIBCXX_TYPE_INT_N_0 __int128
- #define __GNUC_GNU_INLINE__ 1
- #define __GNUC_MINOR__ 2
- #define __GNUC_PATCHLEVEL__ 1
- #define __GNUC__ 4
- #define __GNUG__ 4
- #define __GXX_ABI_VERSION 1002
- #define __GXX_EXPERIMENTAL_CXX0X__ 1
- #define __GXX_RTTI 1
- #define __GXX_WEAK__ 1
- #define __HAVE_FUNCTION_MULTI_VERSIONING 1
- #define __INT16_C_SUFFIX__
- #define __INT16_FMTd__ "hd"
- #define __INT16_FMTi__ "hi"

- #define ___INT16_MAX___ 32767
- #define ___INT16_TYPE___ short
- #define ___INT32_C_SUFFIX___
- #define ___INT32_FMTd___ "d"
- #define ___INT32_FMTi___ "i"
- #define ___INT32_MAX___ 2147483647
- #define ___INT32_TYPE___ int
- #define ___INT64_C_SUFFIX___ LL
- #define ___INT64_FMTd___ "lld"
- #define ___INT64_FMTi___ "lli"
- #define ___INT64_MAX___ 9223372036854775807LL
- #define ___INT64_TYPE___ long long int
- #define ___INT8_C_SUFFIX___
- #define ___INT8_FMTd___ "hhd"
- #define ___INT8_FMTi___ "hhi"
- #define ___INT8_MAX___ 127
- #define ___INT8_TYPE___ signed char
- #define ___INTMAX_C_SUFFIX___ L
- #define ___INTMAX_FMTd___ "ld"
- #define ___INTMAX_FMTi___ "li"
- #define ___INTMAX_MAX___ 9223372036854775807L
- #define ___INTMAX_TYPE___ long int
- #define ___INTMAX_WIDTH___ 64
- #define ___INTPTR_FMTd___ "ld"
- #define ___INTPTR_FMTi___ "li"
- #define ___INTPTR_MAX___ 9223372036854775807L
- #define ___INTPTR_TYPE___ long int
- #define ___INTPTR_WIDTH___ 64
- #define ___INT_FAST16_FMTd___ "hd"
- #define ___INT_FAST16_FMTi___ "hi"
- #define ___INT_FAST16_MAX___ 32767
- #define ___INT_FAST16_TYPE___ short
- #define ___INT_FAST16_WIDTH___ 16
- #define ___INT_FAST32_FMTd___ "d"
- #define ___INT_FAST32_FMTi___ "i"
- #define ___INT_FAST32_MAX___ 2147483647
- #define ___INT_FAST32_TYPE___ int
- #define ___INT_FAST32_WIDTH___ 32
- #define ___INT_FAST64_FMTd___ "lld"
- #define ___INT_FAST64_FMTi___ "lli"
- #define ___INT_FAST64_MAX___ 9223372036854775807LL
- #define ___INT_FAST64_TYPE___ long long int
- #define ___INT_FAST64_WIDTH___ 64
- #define ___INT_FAST8_FMTd___ "hhd"
- #define ___INT_FAST8_FMTi___ "hhi"
- #define ___INT_FAST8_MAX___ 127
- #define ___INT_FAST8_TYPE___ signed char
- #define ___INT_FAST8_WIDTH___ 8
- #define ___INT_LEAST16_FMTd___ "hd"
- #define ___INT_LEAST16_FMTi___ "hi"
- #define ___INT_LEAST16_MAX___ 32767
- #define ___INT_LEAST16_TYPE___ short
- #define ___INT_LEAST16_WIDTH___ 16
- #define ___INT_LEAST32_FMTd___ "d"
- #define ___INT_LEAST32_FMTi___ "i"

- #define __INT_LEAST32_MAX__ 2147483647
- #define __INT_LEAST32_TYPE__ int
- #define __INT_LEAST32_WIDTH__ 32
- #define __INT_LEAST64_FMTd__ "lld"
- #define __INT_LEAST64_FMTi__ "lli"
- #define __INT_LEAST64_MAX__ 9223372036854775807LL
- #define __INT_LEAST64_TYPE__ long long int
- #define __INT_LEAST64_WIDTH__ 64
- #define __INT_LEAST8_FMTd__ "hhd"
- #define __INT_LEAST8_FMTi__ "hhi"
- #define __INT_LEAST8_MAX__ 127
- #define __INT_LEAST8_TYPE__ signed char
- #define __INT_LEAST8_WIDTH__ 8
- #define __INT_MAX__ 2147483647
- #define __INT_WIDTH__ 32
- #define __LDBL_DECIMAL_DIG__ 17
- #define __LDBL_DENORM_MIN__ 4.9406564584124654e-324L
- #define __LDBL_DIG__ 15
- #define __LDBL_EPSILON__ 2.2204460492503131e-16L
- #define __LDBL_HAS_DENORM__ 1
- #define __LDBL_HAS_INFINITY__ 1
- #define __LDBL_HAS_QUIET_NAN__ 1
- #define __LDBL_MANT_DIG__ 53
- #define __LDBL_MAX_10_EXP__ 308
- #define __LDBL_MAX_EXP__ 1024
- #define __LDBL_MAX__ 1.7976931348623157e+308L
- #define __LDBL_MIN_10_EXP__ (-307)
- #define __LDBL_MIN_EXP__ (-1021)
- #define __LDBL_MIN__ 2.2250738585072014e-308L
- #define __LDBL_NORM_MAX__ 1.7976931348623157e+308L
- #define __LITTLE_ENDIAN__ 1
- #define __LLONG_WIDTH__ 64
- #define __LONG_LONG_MAX__ 9223372036854775807LL
- #define __LONG_MAX__ 9223372036854775807L
- #define __LONG_WIDTH__ 64
- #define __LP64__ 1
- #define __MACH__ 1
- #define __MEMORY_SCOPE_DEVICE 1
- #define __MEMORY_SCOPE_SINGLE 4
- #define __MEMORY_SCOPE_SYSTEM 0
- #define __MEMORY_SCOPE_WRKGRP 2
- #define __MEMORY_SCOPE_WVFRNT 3
- #define __NO_INLINE__ 1
- #define __NO_MATH_ERRNO__ 1
- #define __OBJC_BOOL_IS_BOOL 1
- #define __OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES 3
- #define __OPENCL_MEMORY_SCOPE_DEVICE 2
- #define __OPENCL_MEMORY_SCOPE_SUB_GROUP 4
- #define __OPENCL_MEMORY_SCOPE_WORK_GROUP 1
- #define __OPENCL_MEMORY_SCOPE_WORK_ITEM 0
- #define __ORDER_BIG_ENDIAN__ 4321
- #define __ORDER_LITTLE_ENDIAN__ 1234
- #define __ORDER_PDP_ENDIAN__ 3412
- #define __PIC__ 2
- #define __POINTER_WIDTH__ 64

- #define __PRAGMA_REDEFINE_EXTNAME 1
- #define __PTRDIFF_FMTd__ "ld"
- #define __PTRDIFF_FMTi__ "li"
- #define __PTRDIFF_MAX__ 9223372036854775807L
- #define __PTRDIFF_TYPE__ long int
- #define __PTRDIFF_WIDTH__ 64
- #define __REGISTER_PREFIX__
- #define __SCHAR_MAX__ 127
- #define __SHRT_MAX__ 32767
- #define __SHRT_WIDTH__ 16
- #define __SIG_ATOMIC_MAX__ 2147483647
- #define __SIG_ATOMIC_WIDTH__ 32
- #define __SIZEOF_DOUBLE__ 8
- #define __SIZEOF_FLOAT__ 4
- #define __SIZEOF_INT128__ 16
- #define __SIZEOF_INT__ 4
- #define __SIZEOF_LONG_DOUBLE__ 8
- #define __SIZEOF_LONG_LONG__ 8
- #define __SIZEOF_LONG__ 8
- #define __SIZEOF_POINTER__ 8
- #define __SIZEOF_PTRDIFF_T__ 8
- #define __SIZEOF_SHORT__ 2
- #define __SIZEOF_SIZE_T__ 8
- #define __SIZEOF_WCHAR_T__ 4
- #define __SIZEOF_WINT_T__ 4
- #define __SIZE_FMTX__ "lX"
- #define __SIZE_FMTo__ "lo"
- #define __SIZE_FMTu__ "lu"
- #define __SIZE_FMTx__ "lx"
- #define __SIZE_MAX__ 18446744073709551615UL
- #define __SIZE_TYPE__ long unsigned int
- #define __SIZE_WIDTH__ 64
- #define __SSP__ 1
- #define __STDCPP_DEFAULT_NEW_ALIGNMENT__ 16UL
- #define __STDCPP_THREADS__ 1
- #define __STDC_EMBED_EMPTY__ 2
- #define __STDC_EMBED_FOUND__ 1
- #define __STDC_EMBED_NOT_FOUND__ 0
- #define __STDC_HOSTED__ 1
- #define __STDC_NO_THREADS__ 1
- #define __STDC_UTF_16__ 1
- #define __STDC_UTF_32__ 1
- #define __STDC__ 1
- #define __UINT16_C_SUFFIX__
- #define __UINT16_FMTX__ "hX"
- #define __UINT16_FMTo__ "ho"
- #define __UINT16_FMTu__ "hu"
- #define __UINT16_FMTx__ "hx"
- #define __UINT16_MAX__ 65535
- #define __UINT16_TYPE__ unsigned short
- #define __UINT32_C_SUFFIX__ U
- #define __UINT32_FMTX__ "X"
- #define __UINT32_FMTo__ "o"
- #define __UINT32_FMTu__ "u"
- #define __UINT32_FMTx__ "x"

- #define ___UINT32_MAX___ 4294967295U
- #define ___UINT32_TYPE___ unsigned int
- #define ___UINT64_C_SUFFIX___ ULL
- #define ___UINT64_FMTX___ "llX"
- #define ___UINT64_FMTo___ "llo"
- #define ___UINT64_FMTu___ "llu"
- #define ___UINT64_FMTx___ "llx"
- #define ___UINT64_MAX___ 18446744073709551615ULL
- #define ___UINT64_TYPE___ long long unsigned int
- #define ___UINT8_C_SUFFIX___
- #define ___UINT8_FMTX___ "hhX"
- #define ___UINT8_FMTo___ "hho"
- #define ___UINT8_FMTu___ "hhu"
- #define ___UINT8_FMTx___ "hhx"
- #define ___UINT8_MAX___ 255
- #define ___UINT8_TYPE___ unsigned char
- #define ___UINTMAX_C_SUFFIX___ UL
- #define ___UINTMAX_FMTX___ "lX"
- #define ___UINTMAX_FMTo___ "lo"
- #define ___UINTMAX_FMTu___ "lu"
- #define ___UINTMAX_FMTx___ "lx"
- #define ___UINTMAX_MAX___ 18446744073709551615UL
- #define ___UINTMAX_TYPE___ long unsigned int
- #define ___UINTMAX_WIDTH___ 64
- #define ___UINTPTR_FMTX___ "lX"
- #define ___UINTPTR_FMTo___ "lo"
- #define ___UINTPTR_FMTu___ "lu"
- #define ___UINTPTR_FMTx___ "lx"
- #define ___UINTPTR_MAX___ 18446744073709551615UL
- #define ___UINTPTR_TYPE___ long unsigned int
- #define ___UINTPTR_WIDTH___ 64
- #define ___UINT_FAST16_FMTX___ "hX"
- #define ___UINT_FAST16_FMTo___ "ho"
- #define ___UINT_FAST16_FMTu___ "hu"
- #define ___UINT_FAST16_FMTx___ "hx"
- #define ___UINT_FAST16_MAX___ 65535
- #define ___UINT_FAST16_TYPE___ unsigned short
- #define ___UINT_FAST32_FMTX___ "X"
- #define ___UINT_FAST32_FMTo___ "o"
- #define ___UINT_FAST32_FMTu___ "u"
- #define ___UINT_FAST32_FMTx___ "x"
- #define ___UINT_FAST32_MAX___ 4294967295U
- #define ___UINT_FAST32_TYPE___ unsigned int
- #define ___UINT_FAST64_FMTX___ "llX"
- #define ___UINT_FAST64_FMTo___ "llo"
- #define ___UINT_FAST64_FMTu___ "llu"
- #define ___UINT_FAST64_FMTx___ "llx"
- #define ___UINT_FAST64_MAX___ 18446744073709551615ULL
- #define ___UINT_FAST64_TYPE___ long long unsigned int
- #define ___UINT_FAST8_FMTX___ "hhX"
- #define ___UINT_FAST8_FMTo___ "hho"
- #define ___UINT_FAST8_FMTu___ "hhu"
- #define ___UINT_FAST8_FMTx___ "hhx"
- #define ___UINT_FAST8_MAX___ 255
- #define ___UINT_FAST8_TYPE___ unsigned char

- #define __UINT_LEAST16_FMTX__ "hX"
- #define __UINT_LEAST16_FMTo__ "ho"
- #define __UINT_LEAST16_FMTu__ "hu"
- #define __UINT_LEAST16_FMTx__ "hx"
- #define __UINT_LEAST16_MAX__ 65535
- #define __UINT_LEAST16_TYPE__ unsigned short
- #define __UINT_LEAST32_FMTX__ "X"
- #define __UINT_LEAST32_FMTo__ "o"
- #define __UINT_LEAST32_FMTu__ "u"
- #define __UINT_LEAST32_FMTx__ "x"
- #define __UINT_LEAST32_MAX__ 4294967295U
- #define __UINT_LEAST32_TYPE__ unsigned int
- #define __UINT_LEAST64_FMTX__ "llX"
- #define __UINT_LEAST64_FMTo__ "llo"
- #define __UINT_LEAST64_FMTu__ "llu"
- #define __UINT_LEAST64_FMTx__ "llx"
- #define __UINT_LEAST64_MAX__ 18446744073709551615ULL
- #define __UINT_LEAST64_TYPE__ long long unsigned int
- #define __UINT_LEAST8_FMTX__ "hhX"
- #define __UINT_LEAST8_FMTo__ "hho"
- #define __UINT_LEAST8_FMTu__ "hhu"
- #define __UINT_LEAST8_FMTx__ "hhx"
- #define __UINT_LEAST8_MAX__ 255
- #define __UINT_LEAST8_TYPE__ unsigned char
- #define __USER_LABEL_PREFIX__ _
- #define __VERSION__ "Apple LLVM 17.0.0 (clang-1700.0.13.3)"
- #define __WCHAR_MAX__ 2147483647
- #define __WCHAR_TYPE__ int
- #define __WCHAR_WIDTH__ 32
- #define __WINT_MAX__ 2147483647
- #define __WINT_TYPE__ int
- #define __WINT_WIDTH__ 32
- #define __aarch64__ 1
- #define __apple_build_version__ 17000013
- #define __arm64 1
- #define __arm64__ 1
- #define __block __attribute__((__blocks__(byref)))
- #define __clang__ 1
- #define __clang_literal_encoding__ "UTF-8"
- #define __clang_major__ 17
- #define __clang_minor__ 0
- #define __clang_patchlevel__ 0
- #define __clang_version__ "17.0.0 (clang-1700.0.13.3)"
- #define __clang_wide_literal_encoding__ "UTF-32"
- #define __cplusplus 201703L
- #define __cpp_aggregate_bases 201603L
- #define __cpp_aggregate_nsdmi 201304L
- #define __cpp_alias_templates 200704L
- #define __cpp_aligned_new 201606L
- #define __cpp_attributes 200809L
- #define __cpp_binary_literals 201304L
- #define __cpp_capture_star_this 201603L
- #define __cpp_constexpr 201603L
- #define __cpp_constexpr_in_decltype 201711L
- #define __cpp_decltype 200707L

- #define __cpp_decltype_auto 201304L
- #define __cpp_deduction_guides 201703L
- #define __cpp_delegating_constructors 200604L
- #define __cpp_deleted_function 202403L
- #define __cpp_digit_separators 201309L
- #define __cpp_enumerator_attributes 201411L
- #define __cpp_exceptions 199711L
- #define __cpp_fold_expressions 201603L
- #define __cpp_generic_lambdas 201304L
- #define __cpp_guaranteed_copy_elision 201606L
- #define __cpp_hex_float 201603L
- #define __cpp_if_constexpr 201606L
- #define __cpp_impl_destroying_delete 201806L
- #define __cpp_inheriting_constructors 201511L
- #define __cpp_init_captures 201304L
- #define __cpp_initializer_lists 200806L
- #define __cpp_inline_variables 201606L
- #define __cpp_lambdas 200907L
- #define __cpp_named_character_escapes 202207L
- #define __cpp_namespace_attributes 201411L
- #define __cpp_nested_namespace_definitions 201411L
- #define __cpp_noexcept_function_type 201510L
- #define __cpp_nontype_template_args 201411L
- #define __cpp_nontype_template_parameter_auto 201606L
- #define __cpp_nsdmi 200809L
- #define __cpp_pack_indexing 202311L
- #define __cpp_placeholder_variables 202306L
- #define __cpp_range_based_for 201603L
- #define __cpp_raw_strings 200710L
- #define __cpp_ref_qualifiers 200710L
- #define __cpp_return_type_deduction 201304L
- #define __cpp_rtti 199711L
- #define __cpp_rvalue_references 200610L
- #define __cpp_static_assert 201411L
- #define __cpp_static_call_operator 202207L
- #define __cpp_structured_bindings 202403L
- #define __cpp_template_auto 201606L
- #define __cpp_template_template_args 201611L
- #define __cpp_threadsafe_static_init 200806L
- #define __cpp_unicode_characters 200704L
- #define __cpp_unicode_literals 200710L
- #define __cpp_user_defined_literals 200809L
- #define __cpp_variable_templates 201304L
- #define __cpp_variadic_templates 200704L
- #define __cpp_variadic_using 201611L
- #define __llvm__ 1
- #define __nonnull _Nonnull
- #define __null_unspecified _Null_unspecified
- #define __nullable _Nullable
- #define __pic__ 2
- #define __private_extern__ extern
- #define __strong
- #define __unsafe_unretained
- #define __weak __attribute__((objc_gc(weak)))

8.33.1 宏定义说明

8.33.1.1 ___aarch64___

#define ___aarch64___ 1

8.33.1.2 ___AARCH64_CMODEL_SMALL___

#define ___AARCH64_CMODEL_SMALL___ 1

8.33.1.3 ___AARCH64_SIMD___

#define ___AARCH64_SIMD___ 1

8.33.1.4 ___AARCH64EL___

#define ___AARCH64EL___ 1

8.33.1.5 ___APPLE___

#define ___APPLE___ 1

8.33.1.6 ___apple_build_version___

#define ___apple_build_version___ 17000013

8.33.1.7 ___APPLE_CC___

#define ___APPLE_CC___ 6000

8.33.1.8 ___arm64

#define ___arm64 1

8.33.1.9 ___arm64___

#define ___arm64___ 1

8.33.1.10 ___ARM64_ARCH_8___

#define ___ARM64_ARCH_8___ 1

8.33.1.11 ___ARM_64BIT_STATE

#define ___ARM_64BIT_STATE 1

8.33.1.12 ___ARM_ACLE

#define ___ARM_ACLE 200

8.33.1.13 ___ARM_ALIGN_MAX_STACK_PWR

#define ___ARM_ALIGN_MAX_STACK_PWR 4

8.33.1.14 ___ARM_ARCH

#define ___ARM_ARCH 8

8.33.1.15 ___ARM_ARCH_8_3___

#define ___ARM_ARCH_8_3___ 1

8.33.1.16 __ARM_ARCH_8_4__

#define __ARM_ARCH_8_4__ 1

8.33.1.17 __ARM_ARCH_8_5__

#define __ARM_ARCH_8_5__ 1

8.33.1.18 __ARM_ARCH_ISA_A64

#define __ARM_ARCH_ISA_A64 1

8.33.1.19 __ARM_ARCH_PROFILE

#define __ARM_ARCH_PROFILE 'A'

8.33.1.20 __ARM_FEATURE_AES

#define __ARM_FEATURE_AES 1

8.33.1.21 __ARM_FEATURE_ATOMICS

#define __ARM_FEATURE_ATOMICS 1

8.33.1.22 __ARM_FEATURE_BTI

#define __ARM_FEATURE_BTI 1

8.33.1.23 __ARM_FEATURE_CLZ

#define __ARM_FEATURE_CLZ 1

8.33.1.24 __ARM_FEATURE_COMPLEX

#define __ARM_FEATURE_COMPLEX 1

8.33.1.25 __ARM_FEATURE_CRC32

#define __ARM_FEATURE_CRC32 1

8.33.1.26 __ARM_FEATURE_CRYPTO

#define __ARM_FEATURE_CRYPTO 1

8.33.1.27 __ARM_FEATURE_DIRECTED_ROUNDING

#define __ARM_FEATURE_DIRECTED_ROUNDING 1

8.33.1.28 __ARM_FEATURE_DIV

#define __ARM_FEATURE_DIV 1

8.33.1.29 __ARM_FEATURE_DOTPROD

#define __ARM_FEATURE_DOTPROD 1

8.33.1.30 __ARM_FEATURE_FMA

#define __ARM_FEATURE_FMA 1

8.33.1.31 __ARM_FEATURE_FP16_FML

#define __ARM_FEATURE_FP16_FML 1

8.33.1.32 __ARM_FEATURE_FP16_SCALAR_ARITHMETIC

#define __ARM_FEATURE_FP16_SCALAR_ARITHMETIC 1

8.33.1.33 __ARM_FEATURE_FP16_VECTOR_ARITHMETIC

#define __ARM_FEATURE_FP16_VECTOR_ARITHMETIC 1

8.33.1.34 __ARM_FEATURE_FRINT

#define __ARM_FEATURE_FRINT 1

8.33.1.35 __ARM_FEATURE_IDIV

#define __ARM_FEATURE_IDIV 1

8.33.1.36 __ARM_FEATURE_JCVT

#define __ARM_FEATURE_JCVT 1

8.33.1.37 __ARM_FEATURE_LDREX

#define __ARM_FEATURE_LDREX 0xF

8.33.1.38 __ARM_FEATURE_NUMERIC_MAXMIN

#define __ARM_FEATURE_NUMERIC_MAXMIN 1

8.33.1.39 __ARM_FEATURE_PAUTH

#define __ARM_FEATURE_PAUTH 1

8.33.1.40 __ARM_FEATURE_QRDMX

#define __ARM_FEATURE_QRDMX 1

8.33.1.41 __ARM_FEATURE_RCPC

#define __ARM_FEATURE_RCPC 1

8.33.1.42 __ARM_FEATURE_SHA2

#define __ARM_FEATURE_SHA2 1

8.33.1.43 __ARM_FEATURE_SHA3

#define __ARM_FEATURE_SHA3 1

8.33.1.44 __ARM_FEATURE_SHA512

#define __ARM_FEATURE_SHA512 1

8.33.1.45 __ARM_FEATURE_UNALIGNED

#define __ARM_FEATURE_UNALIGNED 1

8.33.1.46 __ARM_FP

#define __ARM_FP 0xE

8.33.1.47 __ARM_FP16_ARGS

#define __ARM_FP16_ARGS 1

8.33.1.48 __ARM_FP16_FORMAT_IEEE

#define __ARM_FP16_FORMAT_IEEE 1

8.33.1.49 __ARM_NEON

#define __ARM_NEON 1

8.33.1.50 __ARM_NEON__

#define __ARM_NEON__ 1

8.33.1.51 __ARM_NEON_FP

#define __ARM_NEON_FP 0xE

8.33.1.52 __ARM_PCS_AAPCS64

#define __ARM_PCS_AAPCS64 1

8.33.1.53 __ARM_SIZEOF_MINIMAL_ENUM

#define __ARM_SIZEOF_MINIMAL_ENUM 4

8.33.1.54 __ARM_SIZEOF_WCHAR_T

#define __ARM_SIZEOF_WCHAR_T 4

8.33.1.55 __ARM_STATE_ZA

#define __ARM_STATE_ZA 1

8.33.1.56 __ARM_STATE_ZT0

#define __ARM_STATE_ZT0 1

8.33.1.57 __ATOMIC_ACQ_REL

#define __ATOMIC_ACQ_REL 4

8.33.1.58 __ATOMIC_ACQUIRE

#define __ATOMIC_ACQUIRE 2

8.33.1.59 __ATOMIC_CONSUME

#define __ATOMIC_CONSUME 1

8.33.1.60 __ATOMIC_RELAXED

#define __ATOMIC_RELAXED 0

8.33.1.61 __ATOMIC_RELEASE

#define __ATOMIC_RELEASE 3

8.33.1.62 __ATOMIC_SEQ_CST

#define __ATOMIC_SEQ_CST 5

8.33.1.63 __BIGGEST_ALIGNMENT__

#define __BIGGEST_ALIGNMENT__ 8

8.33.1.64 ___BITINT_MAXWIDTH___

#define ___BITINT_MAXWIDTH___ 128

8.33.1.65 ___block

#define ___block ___attribute___((___blocks___(byref)))

8.33.1.66 ___BLOCKS___

#define ___BLOCKS___ 1

8.33.1.67 ___BOOL_WIDTH___

#define ___BOOL_WIDTH___ 8

8.33.1.68 ___BYTE_ORDER___

#define ___BYTE_ORDER___ ___ORDER_LITTLE_ENDIAN___

8.33.1.69 ___CHAR16_TYPE___

#define ___CHAR16_TYPE___ unsigned short

8.33.1.70 ___CHAR32_TYPE___

#define ___CHAR32_TYPE___ unsigned int

8.33.1.71 ___CHAR_BIT___

#define ___CHAR_BIT___ 8

8.33.1.72 ___clang___

#define ___clang___ 1

8.33.1.73 ___CLANG_ATOMIC_BOOL_LOCK_FREE

#define ___CLANG_ATOMIC_BOOL_LOCK_FREE 2

8.33.1.74 ___CLANG_ATOMIC_CHAR16_T_LOCK_FREE

#define ___CLANG_ATOMIC_CHAR16_T_LOCK_FREE 2

8.33.1.75 ___CLANG_ATOMIC_CHAR32_T_LOCK_FREE

#define ___CLANG_ATOMIC_CHAR32_T_LOCK_FREE 2

8.33.1.76 ___CLANG_ATOMIC_CHAR_LOCK_FREE

#define ___CLANG_ATOMIC_CHAR_LOCK_FREE 2

8.33.1.77 ___CLANG_ATOMIC_INT_LOCK_FREE

#define ___CLANG_ATOMIC_INT_LOCK_FREE 2

8.33.1.78 ___CLANG_ATOMIC_LLONG_LOCK_FREE

#define ___CLANG_ATOMIC_LLONG_LOCK_FREE 2

8.33.1.79 ___CLANG_ATOMIC_LONG_LOCK_FREE

#define ___CLANG_ATOMIC_LONG_LOCK_FREE 2

8.33.1.80 __CLANG_ATOMIC_POINTER_LOCK_FREE

#define __CLANG_ATOMIC_POINTER_LOCK_FREE 2

8.33.1.81 __CLANG_ATOMIC_SHORT_LOCK_FREE

#define __CLANG_ATOMIC_SHORT_LOCK_FREE 2

8.33.1.82 __CLANG_ATOMIC_WCHAR_T_LOCK_FREE

#define __CLANG_ATOMIC_WCHAR_T_LOCK_FREE 2

8.33.1.83 __clang_literal_encoding__

#define __clang_literal_encoding__ "UTF-8"

8.33.1.84 __clang_major__

#define __clang_major__ 17

8.33.1.85 __clang_minor__

#define __clang_minor__ 0

8.33.1.86 __clang_patchlevel__

#define __clang_patchlevel__ 0

8.33.1.87 __clang_version__

#define __clang_version__ "17.0.0 (clang-1700.0.13.3)"

8.33.1.88 __clang_wide_literal_encoding__

#define __clang_wide_literal_encoding__ "UTF-32"

8.33.1.89 __CONSTANT_CFSTRINGS__

#define __CONSTANT_CFSTRINGS__ 1

8.33.1.90 __cplusplus

#define __cplusplus 201703L

8.33.1.91 __cpp_aggregate_bases

#define __cpp_aggregate_bases 201603L

8.33.1.92 __cpp_aggregate_nsdmi

#define __cpp_aggregate_nsdmi 201304L

8.33.1.93 __cpp_alias_templates

#define __cpp_alias_templates 200704L

8.33.1.94 __cpp_aligned_new

#define __cpp_aligned_new 201606L

8.33.1.95 __cpp_attributes

#define __cpp_attributes 200809L

8.33.1.96 ___cpp_binary_literals

#define ___cpp_binary_literals 201304L

8.33.1.97 ___cpp_capture_star_this

#define ___cpp_capture_star_this 201603L

8.33.1.98 ___cpp_constexpr

#define ___cpp_constexpr 201603L

8.33.1.99 ___cpp_constexpr_in_decltype

#define ___cpp_constexpr_in_decltype 201711L

8.33.1.100 ___cpp_decltype

#define ___cpp_decltype 200707L

8.33.1.101 ___cpp_decltype_auto

#define ___cpp_decltype_auto 201304L

8.33.1.102 ___cpp_deduction_guides

#define ___cpp_deduction_guides 201703L

8.33.1.103 ___cpp_delegating_constructors

#define ___cpp_delegating_constructors 200604L

8.33.1.104 ___cpp_deleted_function

#define ___cpp_deleted_function 202403L

8.33.1.105 ___cpp_digit_separators

#define ___cpp_digit_separators 201309L

8.33.1.106 ___cpp_enumerator_attributes

#define ___cpp_enumerator_attributes 201411L

8.33.1.107 ___cpp_exceptions

#define ___cpp_exceptions 199711L

8.33.1.108 ___cpp_fold_expressions

#define ___cpp_fold_expressions 201603L

8.33.1.109 ___cpp_generic_lambdas

#define ___cpp_generic_lambdas 201304L

8.33.1.110 ___cpp_guaranteed_copy_elision

#define ___cpp_guaranteed_copy_elision 201606L

8.33.1.111 ___cpp_hex_float

#define ___cpp_hex_float 201603L

8.33.1.112 ___cpp_if_constexpr

#define ___cpp_if_constexpr 201606L

8.33.1.113 ___cpp_impl_destroying_delete

#define ___cpp_impl_destroying_delete 201806L

8.33.1.114 ___cpp_inheriting_constructors

#define ___cpp_inheriting_constructors 201511L

8.33.1.115 ___cpp_init_captures

#define ___cpp_init_captures 201304L

8.33.1.116 ___cpp_initializer_lists

#define ___cpp_initializer_lists 200806L

8.33.1.117 ___cpp_inline_variables

#define ___cpp_inline_variables 201606L

8.33.1.118 ___cpp_lambdas

#define ___cpp_lambdas 200907L

8.33.1.119 ___cpp_named_character_escapes

#define ___cpp_named_character_escapes 202207L

8.33.1.120 ___cpp_namespace_attributes

#define ___cpp_namespace_attributes 201411L

8.33.1.121 ___cpp_nested_namespace_definitions

#define ___cpp_nested_namespace_definitions 201411L

8.33.1.122 ___cpp_noexcept_function_type

#define ___cpp_noexcept_function_type 201510L

8.33.1.123 ___cpp_nontype_template_args

#define ___cpp_nontype_template_args 201411L

8.33.1.124 ___cpp_nontype_template_parameter_auto

#define ___cpp_nontype_template_parameter_auto 201606L

8.33.1.125 ___cpp_nsdmi

#define ___cpp_nsdmi 200809L

8.33.1.126 ___cpp_pack_indexing

#define ___cpp_pack_indexing 202311L

8.33.1.127 ___cpp_placeholder_variables

#define ___cpp_placeholder_variables 202306L

8.33.1.128 ___cpp_range_based_for

#define ___cpp_range_based_for 201603L

8.33.1.129 ___cpp_raw_strings

#define ___cpp_raw_strings 200710L

8.33.1.130 ___cpp_ref_qualifiers

#define ___cpp_ref_qualifiers 200710L

8.33.1.131 ___cpp_return_type_deduction

#define ___cpp_return_type_deduction 201304L

8.33.1.132 ___cpp_rtti

#define ___cpp_rtti 199711L

8.33.1.133 ___cpp_rvalue_references

#define ___cpp_rvalue_references 200610L

8.33.1.134 ___cpp_static_assert

#define ___cpp_static_assert 201411L

8.33.1.135 ___cpp_static_call_operator

#define ___cpp_static_call_operator 202207L

8.33.1.136 ___cpp_structured_bindings

#define ___cpp_structured_bindings 202403L

8.33.1.137 ___cpp_template_auto

#define ___cpp_template_auto 201606L

8.33.1.138 ___cpp_template_template_args

#define ___cpp_template_template_args 201611L

8.33.1.139 ___cpp_threadsafe_static_init

#define ___cpp_threadsafe_static_init 200806L

8.33.1.140 ___cpp_unicode_characters

#define ___cpp_unicode_characters 200704L

8.33.1.141 ___cpp_unicode_literals

#define ___cpp_unicode_literals 200710L

8.33.1.142 ___cpp_user_defined_literals

#define ___cpp_user_defined_literals 200809L

8.33.1.143 ___cpp_variable_templates

#define ___cpp_variable_templates 201304L

8.33.1.144 ___cpp_variadic_templates

#define ___cpp_variadic_templates 200704L

8.33.1.145 ___cpp_variadic_using

#define ___cpp_variadic_using 201611L

8.33.1.146 ___DBL_DECIMAL_DIG___

#define ___DBL_DECIMAL_DIG___ 17

8.33.1.147 ___DBL_DENORM_MIN___

#define ___DBL_DENORM_MIN___ 4.9406564584124654e-324

8.33.1.148 ___DBL_DIG___

#define ___DBL_DIG___ 15

8.33.1.149 ___DBL_EPSILON___

#define ___DBL_EPSILON___ 2.2204460492503131e-16

8.33.1.150 ___DBL_HAS_DENORM___

#define ___DBL_HAS_DENORM___ 1

8.33.1.151 ___DBL_HAS_INFINITY___

#define ___DBL_HAS_INFINITY___ 1

8.33.1.152 ___DBL_HAS_QUIET_NAN___

#define ___DBL_HAS_QUIET_NAN___ 1

8.33.1.153 ___DBL_MANT_DIG___

#define ___DBL_MANT_DIG___ 53

8.33.1.154 ___DBL_MAX_10_EXP___

#define ___DBL_MAX_10_EXP___ 308

8.33.1.155 ___DBL_MAX___

#define ___DBL_MAX___ 1.7976931348623157e+308

8.33.1.156 ___DBL_MAX_EXP___

#define ___DBL_MAX_EXP___ 1024

8.33.1.157 ___DBL_MIN_10_EXP___

#define ___DBL_MIN_10_EXP___ (-307)

8.33.1.158 ___DBL_MIN___

#define ___DBL_MIN___ 2.2250738585072014e-308

8.33.1.159 ___DBL_MIN_EXP___

#define ___DBL_MIN_EXP___ (-1021)

8.33.1.160 ___DBL_NORM_MAX___

#define ___DBL_NORM_MAX___ 1.7976931348623157e+308

8.33.1.161 ___DECIMAL_DIG___

#define ___DECIMAL_DIG___ ___LDBL_DECIMAL_DIG___

8.33.1.162 ___DEPRECATED

#define ___DEPRECATED 1

8.33.1.163 ___DYNAMIC___

#define ___DYNAMIC___ 1

8.33.1.164 ___ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED___

#define ___ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED___ 150000

8.33.1.165 ___ENVIRONMENT_OS_VERSION_MIN_REQUIRED___

#define ___ENVIRONMENT_OS_VERSION_MIN_REQUIRED___ 150000

8.33.1.166 ___EXCEPTIONS

#define ___EXCEPTIONS 1

8.33.1.167 ___FINITE_MATH_ONLY___

#define ___FINITE_MATH_ONLY___ 0

8.33.1.168 ___FLT16_DECIMAL_DIG___

#define ___FLT16_DECIMAL_DIG___ 5

8.33.1.169 ___FLT16_DENORM_MIN___

#define ___FLT16_DENORM_MIN___ 5.9604644775390625e-8F16

8.33.1.170 ___FLT16_DIG___

#define ___FLT16_DIG___ 3

8.33.1.171 ___FLT16_EPSILON___

#define ___FLT16_EPSILON___ 9.765625e-4F16

8.33.1.172 ___FLT16_HAS_DENORM___

#define ___FLT16_HAS_DENORM___ 1

8.33.1.173 ___FLT16_HAS_INFINITY___

#define ___FLT16_HAS_INFINITY___ 1

8.33.1.174 ___FLT16_HAS_QUIET_NAN___

#define ___FLT16_HAS_QUIET_NAN___ 1

8.33.1.175 ___FLT16_MANT_DIG___

#define ___FLT16_MANT_DIG___ 11

8.33.1.176 ___FLT16_MAX_10_EXP___

#define ___FLT16_MAX_10_EXP___ 4

8.33.1.177 ___FLT16_MAX___

#define ___FLT16_MAX___ 6.5504e+4F16

8.33.1.178 ___FLT16_MAX_EXP___

#define ___FLT16_MAX_EXP___ 16

8.33.1.179 ___FLT16_MIN_10_EXP___

#define ___FLT16_MIN_10_EXP___ (-4)

8.33.1.180 ___FLT16_MIN___

#define ___FLT16_MIN___ 6.103515625e-5F16

8.33.1.181 ___FLT16_MIN_EXP___

#define ___FLT16_MIN_EXP___ (-13)

8.33.1.182 ___FLT16_NORM_MAX___

#define ___FLT16_NORM_MAX___ 6.5504e+4F16

8.33.1.183 ___FLT_DECIMAL_DIG___

#define ___FLT_DECIMAL_DIG___ 9

8.33.1.184 ___FLT_DENORM_MIN___

#define ___FLT_DENORM_MIN___ 1.40129846e-45F

8.33.1.185 ___FLT_DIG___

#define ___FLT_DIG___ 6

8.33.1.186 ___FLT_EPSILON___

#define ___FLT_EPSILON___ 1.19209290e-7F

8.33.1.187 ___FLT_HAS_DENORM___

#define ___FLT_HAS_DENORM___ 1

8.33.1.188 ___FLT_HAS_INFINITY___

#define ___FLT_HAS_INFINITY___ 1

8.33.1.189 ___FLT_HAS_QUIET_NAN___

#define ___FLT_HAS_QUIET_NAN___ 1

8.33.1.190 ___FLT_MANT_DIG___

#define ___FLT_MANT_DIG___ 24

8.33.1.191 ___FLT_MAX_10_EXP___

#define ___FLT_MAX_10_EXP___ 38

8.33.1.192 ___FLT_MAX___

#define ___FLT_MAX___ 3.40282347e+38F

8.33.1.193 ___FLT_MAX_EXP___

#define ___FLT_MAX_EXP___ 128

8.33.1.194 ___FLT_MIN_10_EXP___

#define ___FLT_MIN_10_EXP___ (-37)

8.33.1.195 ___FLT_MIN___

#define ___FLT_MIN___ 1.17549435e-38F

8.33.1.196 ___FLT_MIN_EXP___

#define ___FLT_MIN_EXP___ (-125)

8.33.1.197 ___FLT_NORM_MAX___

#define ___FLT_NORM_MAX___ 3.40282347e+38F

8.33.1.198 ___FLT_RADIX___

#define ___FLT_RADIX___ 2

8.33.1.199 ___FP_FAST_FMA

#define ___FP_FAST_FMA 1

8.33.1.200 ___FP_FAST_FMAF

#define ___FP_FAST_FMAF 1

8.33.1.201 ___FPCLASS_NEGINF

#define ___FPCLASS_NEGINF 0x0004

8.33.1.202 ___FPCLASS_NEGNORMAL

#define ___FPCLASS_NEGNORMAL 0x0008

8.33.1.203 ___FPCLASS_NEGSUBNORMAL

#define ___FPCLASS_NEGSUBNORMAL 0x0010

8.33.1.204 ___FPCLASS_NEGZERO

#define ___FPCLASS_NEGZERO 0x0020

8.33.1.205 ___FPCLASS_POSINF

#define ___FPCLASS_POSINF 0x0200

8.33.1.206 ___FPCLASS_POSNORMAL

#define ___FPCLASS_POSNORMAL 0x0100

8.33.1.207 ___FPCLASS_POSSUBNORMAL

#define ___FPCLASS_POSSUBNORMAL 0x0080

8.33.1.208 ___FPCLASS_POSZERO

#define ___FPCLASS_POSZERO 0x0040

8.33.1.209 ___FPCLASS_QNAN

#define ___FPCLASS_QNAN 0x0002

8.33.1.210 ___FPCLASS_SNAN

#define ___FPCLASS_SNAN 0x0001

8.33.1.211 ___GCC_ASM_FLAG_OUTPUTS___

#define ___GCC_ASM_FLAG_OUTPUTS___ 1

8.33.1.212 ___GCC_ATOMIC_BOOL_LOCK_FREE

#define ___GCC_ATOMIC_BOOL_LOCK_FREE 2

8.33.1.213 ___GCC_ATOMIC_CHAR16_T_LOCK_FREE

#define ___GCC_ATOMIC_CHAR16_T_LOCK_FREE 2

8.33.1.214 ___GCC_ATOMIC_CHAR32_T_LOCK_FREE

#define ___GCC_ATOMIC_CHAR32_T_LOCK_FREE 2

8.33.1.215 ___GCC_ATOMIC_CHAR_LOCK_FREE

#define ___GCC_ATOMIC_CHAR_LOCK_FREE 2

8.33.1.216 ___GCC_ATOMIC_INT_LOCK_FREE

#define ___GCC_ATOMIC_INT_LOCK_FREE 2

8.33.1.217 ___GCC_ATOMIC_LLONG_LOCK_FREE

#define ___GCC_ATOMIC_LLONG_LOCK_FREE 2

8.33.1.218 ___GCC_ATOMIC_LONG_LOCK_FREE

#define ___GCC_ATOMIC_LONG_LOCK_FREE 2

8.33.1.219 ___GCC_ATOMIC_POINTER_LOCK_FREE

#define ___GCC_ATOMIC_POINTER_LOCK_FREE 2

8.33.1.220 ___GCC_ATOMIC_SHORT_LOCK_FREE

#define ___GCC_ATOMIC_SHORT_LOCK_FREE 2

8.33.1.221 ___GCC_ATOMIC_TEST_AND_SET_TRUEVAL

#define ___GCC_ATOMIC_TEST_AND_SET_TRUEVAL 1

8.33.1.222 ___GCC_ATOMIC_WCHAR_T_LOCK_FREE

#define ___GCC_ATOMIC_WCHAR_T_LOCK_FREE 2

8.33.1.223 ___GCC_CONSTRUCTIVE_SIZE

#define ___GCC_CONSTRUCTIVE_SIZE 64

8.33.1.224 __GCC_DESTRUCTIVE_SIZE

#define __GCC_DESTRUCTIVE_SIZE 64

8.33.1.225 __GCC_HAVE_DWARF2_CFI_ASM

#define __GCC_HAVE_DWARF2_CFI_ASM 1

8.33.1.226 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_1

#define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_1 1

8.33.1.227 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_16

#define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_16 1

8.33.1.228 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_2

#define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_2 1

8.33.1.229 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_4

#define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_4 1

8.33.1.230 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_8

#define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_8 1

8.33.1.231 __GLIBCXX_BITSIZE_INT_N_0

#define __GLIBCXX_BITSIZE_INT_N_0 128

8.33.1.232 __GLIBCXX_TYPE_INT_N_0

#define __GLIBCXX_TYPE_INT_N_0 __int128

8.33.1.233 __GNUC__

#define __GNUC__ 4

8.33.1.234 __GNUC_GNU_INLINE__

#define __GNUC_GNU_INLINE__ 1

8.33.1.235 __GNUC_MINOR__

#define __GNUC_MINOR__ 2

8.33.1.236 __GNUC_PATCHLEVEL__

#define __GNUC_PATCHLEVEL__ 1

8.33.1.237 __GNUG__

#define __GNUG__ 4

8.33.1.238 __GXX_ABI_VERSION

#define __GXX_ABI_VERSION 1002

8.33.1.239 __GXX_EXPERIMENTAL_CXX0X__

#define __GXX_EXPERIMENTAL_CXX0X__ 1

8.33.1.240 ___GXX_RTTI

#define ___GXX_RTTI 1

8.33.1.241 ___GXX_WEAK___

#define ___GXX_WEAK___ 1

8.33.1.242 ___HAVE_FUNCTION_MULTI_VERSIONING

#define ___HAVE_FUNCTION_MULTI_VERSIONING 1

8.33.1.243 ___INT16_C_SUFFIX___

#define ___INT16_C_SUFFIX___

8.33.1.244 ___INT16_FMTd___

#define ___INT16_FMTd___ "hd"

8.33.1.245 ___INT16_FMTi___

#define ___INT16_FMTi___ "hi"

8.33.1.246 ___INT16_MAX___

#define ___INT16_MAX___ 32767

8.33.1.247 ___INT16_TYPE___

#define ___INT16_TYPE___ short

8.33.1.248 ___INT32_C_SUFFIX___

#define ___INT32_C_SUFFIX___

8.33.1.249 ___INT32_FMTd___

#define ___INT32_FMTd___ "d"

8.33.1.250 ___INT32_FMTi___

#define ___INT32_FMTi___ "i"

8.33.1.251 ___INT32_MAX___

#define ___INT32_MAX___ 2147483647

8.33.1.252 ___INT32_TYPE___

#define ___INT32_TYPE___ int

8.33.1.253 ___INT64_C_SUFFIX___

#define ___INT64_C_SUFFIX___ LL

8.33.1.254 ___INT64_FMTd___

#define ___INT64_FMTd___ "lld"

8.33.1.255 ___INT64_FMTi___

#define ___INT64_FMTi___ "lli"

8.33.1.256 ___INT64_MAX___

#define ___INT64_MAX___ 92233720368547758O7LL

8.33.1.257 ___INT64_TYPE___

#define ___INT64_TYPE___ long long int

8.33.1.258 ___INT8_C_SUFFIX___

#define ___INT8_C_SUFFIX___

8.33.1.259 ___INT8_FMTd___

#define ___INT8_FMTd___ "hhd"

8.33.1.260 ___INT8_FMTi___

#define ___INT8_FMTi___ "hhi"

8.33.1.261 ___INT8_MAX___

#define ___INT8_MAX___ 127

8.33.1.262 ___INT8_TYPE___

#define ___INT8_TYPE___ signed char

8.33.1.263 ___INT_FAST16_FMTd___

#define ___INT_FAST16_FMTd___ "hd"

8.33.1.264 ___INT_FAST16_FMTi___

#define ___INT_FAST16_FMTi___ "hi"

8.33.1.265 ___INT_FAST16_MAX___

#define ___INT_FAST16_MAX___ 32767

8.33.1.266 ___INT_FAST16_TYPE___

#define ___INT_FAST16_TYPE___ short

8.33.1.267 ___INT_FAST16_WIDTH___

#define ___INT_FAST16_WIDTH___ 16

8.33.1.268 ___INT_FAST32_FMTd___

#define ___INT_FAST32_FMTd___ "d"

8.33.1.269 ___INT_FAST32_FMTi___

#define ___INT_FAST32_FMTi___ "i"

8.33.1.270 ___INT_FAST32_MAX___

#define ___INT_FAST32_MAX___ 2147483647

8.33.1.271 ___INT_FAST32_TYPE___

#define ___INT_FAST32_TYPE___ int

8.33.1.272 ___INT_FAST32_WIDTH___

#define ___INT_FAST32_WIDTH___ 32

8.33.1.273 ___INT_FAST64_FMTd___

#define ___INT_FAST64_FMTd___ "lld"

8.33.1.274 ___INT_FAST64_FMTi___

#define ___INT_FAST64_FMTi___ "lli"

8.33.1.275 ___INT_FAST64_MAX___

#define ___INT_FAST64_MAX___ 92233720368547758O7LL

8.33.1.276 ___INT_FAST64_TYPE___

#define ___INT_FAST64_TYPE___ long long int

8.33.1.277 ___INT_FAST64_WIDTH___

#define ___INT_FAST64_WIDTH___ 64

8.33.1.278 ___INT_FAST8_FMTd___

#define ___INT_FAST8_FMTd___ "hhd"

8.33.1.279 ___INT_FAST8_FMTi___

#define ___INT_FAST8_FMTi___ "hhi"

8.33.1.280 ___INT_FAST8_MAX___

#define ___INT_FAST8_MAX___ 127

8.33.1.281 ___INT_FAST8_TYPE___

#define ___INT_FAST8_TYPE___ signed char

8.33.1.282 ___INT_FAST8_WIDTH___

#define ___INT_FAST8_WIDTH___ 8

8.33.1.283 ___INT_LEAST16_FMTd___

#define ___INT_LEAST16_FMTd___ "hd"

8.33.1.284 ___INT_LEAST16_FMTi___

#define ___INT_LEAST16_FMTi___ "hi"

8.33.1.285 ___INT_LEAST16_MAX___

#define ___INT_LEAST16_MAX___ 32767

8.33.1.286 ___INT_LEAST16_TYPE___

#define ___INT_LEAST16_TYPE___ short

8.33.1.287 ___INT_LEAST16_WIDTH___

#define ___INT_LEAST16_WIDTH___ 16

8.33.1.288 ___INT_LEAST32_FMTd___

#define ___INT_LEAST32_FMTd___ "d"

8.33.1.289 ___INT_LEAST32_FMTi___

#define ___INT_LEAST32_FMTi___ "i"

8.33.1.290 ___INT_LEAST32_MAX___

#define ___INT_LEAST32_MAX___ 2147483647

8.33.1.291 ___INT_LEAST32_TYPE___

#define ___INT_LEAST32_TYPE___ int

8.33.1.292 ___INT_LEAST32_WIDTH___

#define ___INT_LEAST32_WIDTH___ 32

8.33.1.293 ___INT_LEAST64_FMTd___

#define ___INT_LEAST64_FMTd___ "lld"

8.33.1.294 ___INT_LEAST64_FMTi___

#define ___INT_LEAST64_FMTi___ "lli"

8.33.1.295 ___INT_LEAST64_MAX___

#define ___INT_LEAST64_MAX___ 9223372036854775807LL

8.33.1.296 ___INT_LEAST64_TYPE___

#define ___INT_LEAST64_TYPE___ long long int

8.33.1.297 ___INT_LEAST64_WIDTH___

#define ___INT_LEAST64_WIDTH___ 64

8.33.1.298 ___INT_LEAST8_FMTd___

#define ___INT_LEAST8_FMTd___ "hhd"

8.33.1.299 ___INT_LEAST8_FMTi___

#define ___INT_LEAST8_FMTi___ "hhi"

8.33.1.300 ___INT_LEAST8_MAX___

#define ___INT_LEAST8_MAX___ 127

8.33.1.301 ___INT_LEAST8_TYPE___

#define ___INT_LEAST8_TYPE___ signed char

8.33.1.302 ___INT_LEAST8_WIDTH___

#define ___INT_LEAST8_WIDTH___ 8

8.33.1.303 ___INT_MAX___

#define ___INT_MAX___ 2147483647

8.33.1.304 ___INT_WIDTH___

#define ___INT_WIDTH___ 32

8.33.1.305 ___INTMAX_C_SUFFIX___

#define ___INTMAX_C_SUFFIX___ L

8.33.1.306 ___INTMAX_FMTd___

#define ___INTMAX_FMTd___ "ld"

8.33.1.307 ___INTMAX_FMTi___

#define ___INTMAX_FMTi___ "li"

8.33.1.308 ___INTMAX_MAX___

#define ___INTMAX_MAX___ 9223372036854775807L

8.33.1.309 ___INTMAX_TYPE___

#define ___INTMAX_TYPE___ long int

8.33.1.310 ___INTMAX_WIDTH___

#define ___INTMAX_WIDTH___ 64

8.33.1.311 ___INTPTR_FMTd___

#define ___INTPTR_FMTd___ "ld"

8.33.1.312 ___INTPTR_FMTi___

#define ___INTPTR_FMTi___ "li"

8.33.1.313 ___INTPTR_MAX___

#define ___INTPTR_MAX___ 9223372036854775807L

8.33.1.314 ___INTPTR_TYPE___

#define ___INTPTR_TYPE___ long int

8.33.1.315 ___INTPTR_WIDTH___

#define ___INTPTR_WIDTH___ 64

8.33.1.316 ___LDBL_DECIMAL_DIG___

#define ___LDBL_DECIMAL_DIG___ 17

8.33.1.317 ___LDBL_DENORM_MIN___

#define ___LDBL_DENORM_MIN___ 4.9406564584124654e-324L

8.33.1.318 ___LDBL_DIG___

#define ___LDBL_DIG___ 15

8.33.1.319 ___LDBL_EPSILON___

#define ___LDBL_EPSILON___ 2.2204460492503131e-16L

8.33.1.320 __LDBL_HAS_DENORM__

#define __LDBL_HAS_DENORM__ 1

8.33.1.321 __LDBL_HAS_INFINITY__

#define __LDBL_HAS_INFINITY__ 1

8.33.1.322 __LDBL_HAS_QUIET_NAN__

#define __LDBL_HAS_QUIET_NAN__ 1

8.33.1.323 __LDBL_MANT_DIG__

#define __LDBL_MANT_DIG__ 53

8.33.1.324 __LDBL_MAX_10_EXP__

#define __LDBL_MAX_10_EXP__ 308

8.33.1.325 __LDBL_MAX__

#define __LDBL_MAX__ 1.7976931348623157e+308L

8.33.1.326 __LDBL_MAX_EXP__

#define __LDBL_MAX_EXP__ 1024

8.33.1.327 __LDBL_MIN_10_EXP__

#define __LDBL_MIN_10_EXP__ (-307)

8.33.1.328 __LDBL_MIN__

#define __LDBL_MIN__ 2.2250738585072014e-308L

8.33.1.329 __LDBL_MIN_EXP__

#define __LDBL_MIN_EXP__ (-1021)

8.33.1.330 __LDBL_NORM_MAX__

#define __LDBL_NORM_MAX__ 1.7976931348623157e+308L

8.33.1.331 __LITTLE_ENDIAN__

#define __LITTLE_ENDIAN__ 1

8.33.1.332 __LLONG_WIDTH__

#define __LLONG_WIDTH__ 64

8.33.1.333 __llvm__

#define __llvm__ 1

8.33.1.334 __LONG_LONG_MAX__

#define __LONG_LONG_MAX__ 9223372036854775807LL

8.33.1.335 __LONG_MAX__

#define __LONG_MAX__ 9223372036854775807L

8.33.1.336 __LONG_WIDTH__

#define __LONG_WIDTH__ 64

8.33.1.337 __LP64__

#define __LP64__ 1

8.33.1.338 __MACH__

#define __MACH__ 1

8.33.1.339 __MEMORY_SCOPE_DEVICE

#define __MEMORY_SCOPE_DEVICE 1

8.33.1.340 __MEMORY_SCOPE_SINGLE

#define __MEMORY_SCOPE_SINGLE 4

8.33.1.341 __MEMORY_SCOPE_SYSTEM

#define __MEMORY_SCOPE_SYSTEM 0

8.33.1.342 __MEMORY_SCOPE_WRKGRP

#define __MEMORY_SCOPE_WRKGRP 2

8.33.1.343 __MEMORY_SCOPE_WVFRNT

#define __MEMORY_SCOPE_WVFRNT 3

8.33.1.344 __NO_INLINE__

#define __NO_INLINE__ 1

8.33.1.345 __NO_MATH_ERRNO__

#define __NO_MATH_ERRNO__ 1

8.33.1.346 __nonnull

#define __nonnull _Nonnull

8.33.1.347 __null_unspecified

#define __null_unspecified _Null_unspecified

8.33.1.348 __nullable

#define __nullable _Nullable

8.33.1.349 __OBJC_BOOL_IS_BOOL

#define __OBJC_BOOL_IS_BOOL 1

8.33.1.350 __OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES

#define __OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES 3

8.33.1.351 __OPENCL_MEMORY_SCOPE_DEVICE

#define __OPENCL_MEMORY_SCOPE_DEVICE 2

8.33.1.352 ___OPENCL_MEMORY_SCOPE_SUB_GROUP

#define ___OPENCL_MEMORY_SCOPE_SUB_GROUP 4

8.33.1.353 ___OPENCL_MEMORY_SCOPE_WORK_GROUP

#define ___OPENCL_MEMORY_SCOPE_WORK_GROUP 1

8.33.1.354 ___OPENCL_MEMORY_SCOPE_WORK_ITEM

#define ___OPENCL_MEMORY_SCOPE_WORK_ITEM 0

8.33.1.355 ___ORDER_BIG_ENDIAN___

#define ___ORDER_BIG_ENDIAN___ 4321

8.33.1.356 ___ORDER_LITTLE_ENDIAN___

#define ___ORDER_LITTLE_ENDIAN___ 1234

8.33.1.357 ___ORDER_PDP_ENDIAN___

#define ___ORDER_PDP_ENDIAN___ 3412

8.33.1.358 ___PIC___

#define ___PIC___ 2

8.33.1.359 ___pic___

#define ___pic___ 2

8.33.1.360 ___POINTER_WIDTH___

#define ___POINTER_WIDTH___ 64

8.33.1.361 ___PRAGMA_REDEFINE_EXTNAME

#define ___PRAGMA_REDEFINE_EXTNAME 1

8.33.1.362 ___private_extern___

#define ___private_extern___ extern

8.33.1.363 ___PTRDIFF_FMTd___

#define ___PTRDIFF_FMTd___ "ld"

8.33.1.364 ___PTRDIFF_FMTi___

#define ___PTRDIFF_FMTi___ "li"

8.33.1.365 ___PTRDIFF_MAX___

#define ___PTRDIFF_MAX___ 9223372036854775807L

8.33.1.366 ___PTRDIFF_TYPE___

#define ___PTRDIFF_TYPE___ long int

8.33.1.367 ___PTRDIFF_WIDTH___

#define ___PTRDIFF_WIDTH___ 64

8.33.1.368 __REGISTER_PREFIX__

#define __REGISTER_PREFIX__

8.33.1.369 __SCHAR_MAX__

#define __SCHAR_MAX__ 127

8.33.1.370 __SHRT_MAX__

#define __SHRT_MAX__ 32767

8.33.1.371 __SHRT_WIDTH__

#define __SHRT_WIDTH__ 16

8.33.1.372 __SIG_ATOMIC_MAX__

#define __SIG_ATOMIC_MAX__ 2147483647

8.33.1.373 __SIG_ATOMIC_WIDTH__

#define __SIG_ATOMIC_WIDTH__ 32

8.33.1.374 __SIZE_FMTo__

#define __SIZE_FMTo__ "lo"

8.33.1.375 __SIZE_FMTu__

#define __SIZE_FMTu__ "lu"

8.33.1.376 __SIZE_FMTX__

#define __SIZE_FMTX__ "lX"

8.33.1.377 __SIZE_FMTx__

#define __SIZE_FMTx__ "lx"

8.33.1.378 __SIZE_MAX__

#define __SIZE_MAX__ 18446744073709551615UL

8.33.1.379 __SIZE_TYPE__

#define __SIZE_TYPE__ long unsigned int

8.33.1.380 __SIZE_WIDTH__

#define __SIZE_WIDTH__ 64

8.33.1.381 __SIZEOF_DOUBLE__

#define __SIZEOF_DOUBLE__ 8

8.33.1.382 __SIZEOF_FLOAT__

#define __SIZEOF_FLOAT__ 4

8.33.1.383 __SIZEOF_INT128__

#define __SIZEOF_INT128__ 16

8.33.1.384 __SIZEOF_INT__

#define __SIZEOF_INT__ 4

8.33.1.385 __SIZEOF_LONG__

#define __SIZEOF_LONG__ 8

8.33.1.386 __SIZEOF_LONG_DOUBLE__

#define __SIZEOF_LONG_DOUBLE__ 8

8.33.1.387 __SIZEOF_LONG_LONG__

#define __SIZEOF_LONG_LONG__ 8

8.33.1.388 __SIZEOF_POINTER__

#define __SIZEOF_POINTER__ 8

8.33.1.389 __SIZEOF_PTRDIFF_T__

#define __SIZEOF_PTRDIFF_T__ 8

8.33.1.390 __SIZEOF_SHORT__

#define __SIZEOF_SHORT__ 2

8.33.1.391 __SIZEOF_SIZE_T__

#define __SIZEOF_SIZE_T__ 8

8.33.1.392 __SIZEOF_WCHAR_T__

#define __SIZEOF_WCHAR_T__ 4

8.33.1.393 __SIZEOF_WINT_T__

#define __SIZEOF_WINT_T__ 4

8.33.1.394 __SSP__

#define __SSP__ 1

8.33.1.395 __STDC__

#define __STDC__ 1

8.33.1.396 __STDC_EMBED_EMPTY__

#define __STDC_EMBED_EMPTY__ 2

8.33.1.397 __STDC_EMBED_FOUND__

#define __STDC_EMBED_FOUND__ 1

8.33.1.398 __STDC_EMBED_NOT_FOUND__

#define __STDC_EMBED_NOT_FOUND__ 0

8.33.1.399 __STDC_HOSTED__

#define __STDC_HOSTED__ 1

8.33.1.400 __STDC_NO_THREADS__

#define __STDC_NO_THREADS__ 1

8.33.1.401 __STDC_UTF_16__

#define __STDC_UTF_16__ 1

8.33.1.402 __STDC_UTF_32__

#define __STDC_UTF_32__ 1

8.33.1.403 __STDCPP_DEFAULT_NEW_ALIGNMENT__

#define __STDCPP_DEFAULT_NEW_ALIGNMENT__ 16UL

8.33.1.404 __STDCPP_THREADS__

#define __STDCPP_THREADS__ 1

8.33.1.405 __strong

#define __strong

8.33.1.406 __UINT16_C_SUFFIX__

#define __UINT16_C_SUFFIX__

8.33.1.407 __UINT16_FMTo__

#define __UINT16_FMTo__ "ho"

8.33.1.408 __UINT16_FMTu__

#define __UINT16_FMTu__ "hu"

8.33.1.409 __UINT16_FMTX__

#define __UINT16_FMTX__ "hX"

8.33.1.410 __UINT16_FMTx__

#define __UINT16_FMTx__ "hx"

8.33.1.411 __UINT16_MAX__

#define __UINT16_MAX__ 65535

8.33.1.412 __UINT16_TYPE__

#define __UINT16_TYPE__ unsigned short

8.33.1.413 __UINT32_C_SUFFIX__

#define __UINT32_C_SUFFIX__ U

8.33.1.414 __UINT32_FMTo__

#define __UINT32_FMTo__ "o"

8.33.1.415 __UINT32_FMTu__

#define __UINT32_FMTu__ "u"

8.33.1.416 ___UINT32_FMTX___

#define ___UINT32_FMTX___ "X"

8.33.1.417 ___UINT32_FMTx___

#define ___UINT32_FMTx___ "x"

8.33.1.418 ___UINT32_MAX___

#define ___UINT32_MAX___ 4294967295U

8.33.1.419 ___UINT32_TYPE___

#define ___UINT32_TYPE___ unsigned int

8.33.1.420 ___UINT64_C_SUFFIX___

#define ___UINT64_C_SUFFIX___ ULL

8.33.1.421 ___UINT64_FMTo___

#define ___UINT64_FMTo___ "llo"

8.33.1.422 ___UINT64_FMTu___

#define ___UINT64_FMTu___ "llu"

8.33.1.423 ___UINT64_FMTX___

#define ___UINT64_FMTX___ "llX"

8.33.1.424 ___UINT64_FMTx___

#define ___UINT64_FMTx___ "llx"

8.33.1.425 ___UINT64_MAX___

#define ___UINT64_MAX___ 18446744073709551615ULL

8.33.1.426 ___UINT64_TYPE___

#define ___UINT64_TYPE___ long long unsigned int

8.33.1.427 ___UINT8_C_SUFFIX___

#define ___UINT8_C_SUFFIX___

8.33.1.428 ___UINT8_FMTo___

#define ___UINT8_FMTo___ "hho"

8.33.1.429 ___UINT8_FMTu___

#define ___UINT8_FMTu___ "hhu"

8.33.1.430 ___UINT8_FMTX___

#define ___UINT8_FMTX___ "hhX"

8.33.1.431 ___UINT8_FMTx___

#define ___UINT8_FMTx___ "hhx"

8.33.1.432 ___UINT8_MAX___

#define ___UINT8_MAX___ 255

8.33.1.433 ___UINT8_TYPE___

#define ___UINT8_TYPE___ unsigned char

8.33.1.434 ___UINT_FAST16_FMTo___

#define ___UINT_FAST16_FMTo___ "ho"

8.33.1.435 ___UINT_FAST16_FMTu___

#define ___UINT_FAST16_FMTu___ "hu"

8.33.1.436 ___UINT_FAST16_FMTX___

#define ___UINT_FAST16_FMTX___ "hX"

8.33.1.437 ___UINT_FAST16_FMTx___

#define ___UINT_FAST16_FMTx___ "hx"

8.33.1.438 ___UINT_FAST16_MAX___

#define ___UINT_FAST16_MAX___ 65535

8.33.1.439 ___UINT_FAST16_TYPE___

#define ___UINT_FAST16_TYPE___ unsigned short

8.33.1.440 ___UINT_FAST32_FMTo___

#define ___UINT_FAST32_FMTo___ "o"

8.33.1.441 ___UINT_FAST32_FMTu___

#define ___UINT_FAST32_FMTu___ "u"

8.33.1.442 ___UINT_FAST32_FMTX___

#define ___UINT_FAST32_FMTX___ "X"

8.33.1.443 ___UINT_FAST32_FMTx___

#define ___UINT_FAST32_FMTx___ "x"

8.33.1.444 ___UINT_FAST32_MAX___

#define ___UINT_FAST32_MAX___ 4294967295U

8.33.1.445 ___UINT_FAST32_TYPE___

#define ___UINT_FAST32_TYPE___ unsigned int

8.33.1.446 ___UINT_FAST64_FMTo___

#define ___UINT_FAST64_FMTo___ "llo"

8.33.1.447 ___UINT_FAST64_FMTu___

#define ___UINT_FAST64_FMTu___ "llu"

8.33.1.448 ___UINT_FAST64_FMTX___

#define ___UINT_FAST64_FMTX___ "llX"

8.33.1.449 ___UINT_FAST64_FMTx___

#define ___UINT_FAST64_FMTx___ "llx"

8.33.1.450 ___UINT_FAST64_MAX___

#define ___UINT_FAST64_MAX___ 18446744073709551615ULL

8.33.1.451 ___UINT_FAST64_TYPE___

#define ___UINT_FAST64_TYPE___ long long unsigned int

8.33.1.452 ___UINT_FAST8_FMTo___

#define ___UINT_FAST8_FMTo___ "hho"

8.33.1.453 ___UINT_FAST8_FMTu___

#define ___UINT_FAST8_FMTu___ "hhu"

8.33.1.454 ___UINT_FAST8_FMTX___

#define ___UINT_FAST8_FMTX___ "hhX"

8.33.1.455 ___UINT_FAST8_FMTx___

#define ___UINT_FAST8_FMTx___ "hhx"

8.33.1.456 ___UINT_FAST8_MAX___

#define ___UINT_FAST8_MAX___ 255

8.33.1.457 ___UINT_FAST8_TYPE___

#define ___UINT_FAST8_TYPE___ unsigned char

8.33.1.458 ___UINT_LEAST16_FMTo___

#define ___UINT_LEAST16_FMTo___ "ho"

8.33.1.459 ___UINT_LEAST16_FMTu___

#define ___UINT_LEAST16_FMTu___ "hu"

8.33.1.460 ___UINT_LEAST16_FMTX___

#define ___UINT_LEAST16_FMTX___ "hX"

8.33.1.461 ___UINT_LEAST16_FMTx___

#define ___UINT_LEAST16_FMTx___ "hx"

8.33.1.462 ___UINT_LEAST16_MAX___

#define ___UINT_LEAST16_MAX___ 65535

8.33.1.463 ___UINT_LEAST16_TYPE___

#define ___UINT_LEAST16_TYPE___ unsigned short

8.33.1.464 ___UINT_LEAST32_FMTo___

#define ___UINT_LEAST32_FMTo___ ”o”

8.33.1.465 ___UINT_LEAST32_FMTu___

#define ___UINT_LEAST32_FMTu___ ”u”

8.33.1.466 ___UINT_LEAST32_FMTX___

#define ___UINT_LEAST32_FMTX___ ”X”

8.33.1.467 ___UINT_LEAST32_FMTx___

#define ___UINT_LEAST32_FMTx___ ”x”

8.33.1.468 ___UINT_LEAST32_MAX___

#define ___UINT_LEAST32_MAX___ 4294967295U

8.33.1.469 ___UINT_LEAST32_TYPE___

#define ___UINT_LEAST32_TYPE___ unsigned int

8.33.1.470 ___UINT_LEAST64_FMTo___

#define ___UINT_LEAST64_FMTo___ ”llo”

8.33.1.471 ___UINT_LEAST64_FMTu___

#define ___UINT_LEAST64_FMTu___ ”llu”

8.33.1.472 ___UINT_LEAST64_FMTX___

#define ___UINT_LEAST64_FMTX___ ”llX”

8.33.1.473 ___UINT_LEAST64_FMTx___

#define ___UINT_LEAST64_FMTx___ ”llx”

8.33.1.474 ___UINT_LEAST64_MAX___

#define ___UINT_LEAST64_MAX___ 18446744073709551615ULL

8.33.1.475 ___UINT_LEAST64_TYPE___

#define ___UINT_LEAST64_TYPE___ long long unsigned int

8.33.1.476 ___UINT_LEAST8_FMTo___

#define ___UINT_LEAST8_FMTo___ ”hho”

8.33.1.477 ___UINT_LEAST8_FMTu___

#define ___UINT_LEAST8_FMTu___ ”hhu”

8.33.1.478 ___UINT_LEAST8_FMTX___

#define ___UINT_LEAST8_FMTX___ ”hhX”

8.33.1.479 ___UINT_LEAST8_FMTx___

#define ___UINT_LEAST8_FMTx___ ”hhx”

8.33.1.480 ___UINT_LEAST8_MAX___

#define ___UINT_LEAST8_MAX___ 255

8.33.1.481 ___UINT_LEAST8_TYPE___

#define ___UINT_LEAST8_TYPE___ unsigned char

8.33.1.482 ___UINTMAX_C_SUFFIX___

#define ___UINTMAX_C_SUFFIX___ UL

8.33.1.483 ___UINTMAX_FMTo___

#define ___UINTMAX_FMTo___ "lo"

8.33.1.484 ___UINTMAX_FMTu___

#define ___UINTMAX_FMTu___ "lu"

8.33.1.485 ___UINTMAX_FMTX___

#define ___UINTMAX_FMTX___ "lX"

8.33.1.486 ___UINTMAX_FMTx___

#define ___UINTMAX_FMTx___ "lx"

8.33.1.487 ___UINTMAX_MAX___

#define ___UINTMAX_MAX___ 18446744073709551615UL

8.33.1.488 ___UINTMAX_TYPE___

#define ___UINTMAX_TYPE___ long unsigned int

8.33.1.489 ___UINTMAX_WIDTH___

#define ___UINTMAX_WIDTH___ 64

8.33.1.490 ___UINTPTR_FMTo___

#define ___UINTPTR_FMTo___ "lo"

8.33.1.491 ___UINTPTR_FMTu___

#define ___UINTPTR_FMTu___ "lu"

8.33.1.492 ___UINTPTR_FMTX___

#define ___UINTPTR_FMTX___ "lX"

8.33.1.493 ___UINTPTR_FMTx___

#define ___UINTPTR_FMTx___ "lx"

8.33.1.494 ___UINTPTR_MAX___

#define ___UINTPTR_MAX___ 18446744073709551615UL

8.33.1.495 ___UINTPTR_TYPE___

#define ___UINTPTR_TYPE___ long unsigned int

8.33.1.496 ___UINTPTR_WIDTH___

#define ___UINTPTR_WIDTH___ 64

8.33.1.497 ___unsafe_unretained

#define ___unsafe_unretained

8.33.1.498 ___USER_LABEL_PREFIX___

#define ___USER_LABEL_PREFIX___ _

8.33.1.499 ___VERSION___

#define ___VERSION___ "Apple LLVM 17.0.0 (clang-1700.0.13.3)"

8.33.1.500 ___WCHAR_MAX___

#define ___WCHAR_MAX___ 2147483647

8.33.1.501 ___WCHAR_TYPE___

#define ___WCHAR_TYPE___ int

8.33.1.502 ___WCHAR_WIDTH___

#define ___WCHAR_WIDTH___ 32

8.33.1.503 ___weak

#define ___weak ___attribute___((objc_gc(weak)))

8.33.1.504 ___WINT_MAX___

#define ___WINT_MAX___ 2147483647

8.33.1.505 ___WINT_TYPE___

#define ___WINT_TYPE___ int

8.33.1.506 ___WINT_WIDTH___

#define ___WINT_WIDTH___ 32

8.33.1.507 _LP64

#define _LP64 1

8.33.1.508 QT_CHARTS_LIB

#define QT_CHARTS_LIB 1

8.33.1.509 QT_CORE_LIB

#define QT_CORE_LIB 1

8.33.1.510 QT_GUI_LIB

#define QT_GUI_LIB 1

8.33.1.511 QT_NETWORK_LIB

#define QT_NETWORK_LIB 1

8.33.1.512 QT_NO_DEBUG

#define QT_NO_DEBUG 1

8.33.1.513 QT_OPENGL_LIB

#define QT_OPENGL_LIB 1

8.33.1.514 QT_OPENGLWIDGETS_LIB

#define QT_OPENGLWIDGETS_LIB 1

8.33.1.515 QT_WIDGETS_LIB

#define QT_WIDGETS_LIB 1

8.33.1.516 SIZEOF_DPTR

#define SIZEOF_DPTR (sizeof(void*))

8.33.1.517 TARGET_IPHONE_SIMULATOR

#define TARGET_IPHONE_SIMULATOR 0

8.33.1.518 TARGET_OS_ARROW

#define TARGET_OS_ARROW 1

8.33.1.519 TARGET_OS_BRIDGE

#define TARGET_OS_BRIDGE 0

8.33.1.520 TARGET_OS_DRIVERKIT

#define TARGET_OS_DRIVERKIT 0

8.33.1.521 TARGET_OS_EMBEDDED

#define TARGET_OS_EMBEDDED 0

8.33.1.522 TARGET_OS_IOS

#define TARGET_OS_IOS 0

8.33.1.523 TARGET_OS_IOSMAC

#define TARGET_OS_IOSMAC 0

8.33.1.524 TARGET_OS_IPHONE

#define TARGET_OS_IPHONE 0

8.33.1.525 TARGET_OS_LINUX

#define TARGET_OS_LINUX 0

8.33.1.526 TARGET_OS_MAC

#define TARGET_OS_MAC 1

8.33.1.527 TARGET_OS_MACCATALYST

#define TARGET_OS_MACCATALYST 0

8.33.1.528 TARGET_OS_NANO

#define TARGET_OS_NANO 0

8.33.1.529 TARGET_OS_OSX

#define TARGET_OS_OSX 1

8.33.1.530 TARGET_OS_SIMULATOR

#define TARGET_OS_SIMULATOR 0

8.33.1.531 TARGET_OS_TV

#define TARGET_OS_TV 0

8.33.1.532 TARGET_OS_UIKITFORMAC

#define TARGET_OS_UIKITFORMAC 0

8.33.1.533 TARGET_OS_UNIX

#define TARGET_OS_UNIX 0

8.33.1.534 TARGET_OS_VISION

#define TARGET_OS_VISION 0

8.33.1.535 TARGET_OS_WATCH

#define TARGET_OS_WATCH 0

8.33.1.536 TARGET_OS_WIN32

#define TARGET_OS_WIN32 0

8.33.1.537 TARGET_OS_WINDOWS

#define TARGET_OS_WINDOWS 0

8.33.1.538 TARGET_OS_XR

#define TARGET_OS_XR 0

8.34 moc_predefs.h

浏览该文件的文档.

```
00001 #define QT_CHARTS_LIB 1
00002 #define QT_CORE_LIB 1
00003 #define QT_GUI_LIB 1
00004 #define QT_NETWORK_LIB 1
00005 #define QT_NO_DEBUG 1
00006 #define QT_OPENGLWIDGETS_LIB 1
00007 #define QT_OPENGL_LIB 1
00008 #define QT_WIDGETS_LIB 1
00009 #define SIZEOF_DPTR (sizeof(void*))
00010 #define TARGET_IPHONE_SIMULATOR 0
00011 #define TARGET_OS_ARROW 1
00012 #define TARGET_OS_BRIDGE 0
00013 #define TARGET_OS_DRIVERKIT 0
00014 #define TARGET_OS_EMBEDDED 0
00015 #define TARGET_OS_IOS 0
00016 #define TARGET_OS_IOSMAC 0
00017 #define TARGET_OS_IPHONE 0
00018 #define TARGET_OS_LINUX 0
00019 #define TARGET_OS_MAC 1
00020 #define TARGET_OS_MACCATALYST 0
00021 #define TARGET_OS_NANO 0
```

```
00022 #define TARGET_OS_OSX 1
00023 #define TARGET_OS_SIMULATOR 0
00024 #define TARGET_OS_TV 0
00025 #define TARGET_OS_UIKITFORMAC 0
00026 #define TARGET_OS_UNIX 0
00027 #define TARGET_OS_VISION 0
00028 #define TARGET_OS_WATCH 0
00029 #define TARGET_OS_WIN32 0
00030 #define TARGET_OS_WINDOWS 0
00031 #define TARGET_OS_XR 0
00032 #define __LP64 1
00033 #define ___AARCH64EL___ 1
00034 #define ___AARCH64_CMODEL_SMALL___ 1
00035 #define ___AARCH64_SIMD___ 1
00036 #define ___APPLE_CC___ 6000
00037 #define ___APPLE___ 1
00038 #define ___ARM64_ARCH_8___ 1
00039 #define ___ARM_64BIT_STATE 1
00040 #define ___ARM_ACLE 200
00041 #define ___ARM_ALIGN_MAX_STACK_PWR 4
00042 #define ___ARM_ARCH 8
00043 #define ___ARM_ARCH_8_3___ 1
00044 #define ___ARM_ARCH_8_4___ 1
00045 #define ___ARM_ARCH_8_5___ 1
00046 #define ___ARM_ARCH_ISA_A64 1
00047 #define ___ARM_ARCH_PROFILE 'A'
00048 #define ___ARM_FEATURE_AES 1
00049 #define ___ARM_FEATURE_ATOMICS 1
00050 #define ___ARM_FEATURE_BTI 1
00051 #define ___ARM_FEATURE_CLZ 1
00052 #define ___ARM_FEATURE_COMPLEX 1
00053 #define ___ARM_FEATURE_CRC32 1
00054 #define ___ARM_FEATURE_CRYPTO 1
00055 #define ___ARM_FEATURE_DIRECTED_ROUNDING 1
00056 #define ___ARM_FEATURE_DIV 1
00057 #define ___ARM_FEATURE_DOTPROD 1
00058 #define ___ARM_FEATURE_FMA 1
00059 #define ___ARM_FEATURE_FP16_FML 1
00060 #define ___ARM_FEATURE_FP16_SCALAR_ARITHMETIC 1
00061 #define ___ARM_FEATURE_FP16_VECTOR_ARITHMETIC 1
00062 #define ___ARM_FEATURE_FRINT 1
00063 #define ___ARM_FEATURE_IDIV 1
00064 #define ___ARM_FEATURE_JCVT 1
00065 #define ___ARM_FEATURE_LDREX 0xF
00066 #define ___ARM_FEATURE_NUMERIC_MAXMIN 1
00067 #define ___ARM_FEATURE_PAUTH 1
00068 #define ___ARM_FEATURE_QRDMX 1
00069 #define ___ARM_FEATURE_RCPC 1
00070 #define ___ARM_FEATURE_SHA2 1
00071 #define ___ARM_FEATURE_SHA3 1
00072 #define ___ARM_FEATURE_SHA512 1
00073 #define ___ARM_FEATURE_UNALIGNED 1
00074 #define ___ARM_FP 0xE
00075 #define ___ARM_FP16_ARGS 1
00076 #define ___ARM_FP16_FORMAT_IEEE 1
00077 #define ___ARM_NEON 1
00078 #define ___ARM_NEON_FP 0xE
00079 #define ___ARM_NEON___ 1
00080 #define ___ARM_PCS_AAPCS64 1
00081 #define ___ARM_SIZEOF_MINIMAL_ENUM 4
00082 #define ___ARM_SIZEOF_WCHAR_T 4
00083 #define ___ARM_STATE_ZA 1
00084 #define ___ARM_STATE_ZT0 1
00085 #define ___ATOMIC_ACQUIRE 2
00086 #define ___ATOMIC_ACQ_REL 4
00087 #define ___ATOMIC_CONSUME 1
00088 #define ___ATOMIC_RELAXED 0
00089 #define ___ATOMIC_RELEASE 3
00090 #define ___ATOMIC_SEQ_CST 5
00091 #define ___BIGGEST_ALIGNMENT___ 8
00092 #define ___BITINT_MAXWIDTH___ 128
00093 #define ___BLOCKS___ 1
00094 #define ___BOOL_WIDTH___ 8
00095 #define ___BYTE_ORDER___ ___ORDER_LITTLE_ENDIAN___
00096 #define ___CHAR16_TYPE___ unsigned short
00097 #define ___CHAR32_TYPE___ unsigned int
00098 #define ___CHAR_BIT___ 8
00099 #define ___CLANG_ATOMIC_BOOL_LOCK_FREE 2
00100 #define ___CLANG_ATOMIC_CHAR16_T_LOCK_FREE 2
00101 #define ___CLANG_ATOMIC_CHAR32_T_LOCK_FREE 2
00102 #define ___CLANG_ATOMIC_CHAR_LOCK_FREE 2
00103 #define ___CLANG_ATOMIC_INT_LOCK_FREE 2
00104 #define ___CLANG_ATOMIC_LLONG_LOCK_FREE 2
00105 #define ___CLANG_ATOMIC_LONG_LOCK_FREE 2
00106 #define ___CLANG_ATOMIC_POINTER_LOCK_FREE 2
00107 #define ___CLANG_ATOMIC_SHORT_LOCK_FREE 2
00108 #define ___CLANG_ATOMIC_WCHAR_T_LOCK_FREE 2
```

```
00109 #define __CONSTANT_CFSTRINGS__ 1
00110 #define __DBL_DECIMAL_DIG__ 17
00111 #define __DBL_DENORM_MIN__ 4.9406564584124654e-324
00112 #define __DBL_DIG__ 15
00113 #define __DBL_EPSILON__ 2.2204460492503131e-16
00114 #define __DBL_HAS_DENORM__ 1
00115 #define __DBL_HAS_INFINITY__ 1
00116 #define __DBL_HAS_QUIET_NAN__ 1
00117 #define __DBL_MANT_DIG__ 53
00118 #define __DBL_MAX_10_EXP__ 308
00119 #define __DBL_MAX_EXP__ 1024
00120 #define __DBL_MAX__ 1.7976931348623157e+308
00121 #define __DBL_MIN_10_EXP__ (-307)
00122 #define __DBL_MIN_EXP__ (-1021)
00123 #define __DBL_MIN__ 2.2250738585072014e-308
00124 #define __DBL_NORM_MAX__ 1.7976931348623157e+308
00125 #define __DECIMAL_DIG__ __LDBL_DECIMAL_DIG__
00126 #define __DEPRECATED 1
00127 #define __DYNAMIC__ 1
00128 #define __ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__ 150000
00129 #define __ENVIRONMENT_OS_VERSION_MIN_REQUIRED__ 150000
00130 #define __EXCEPTIONS 1
00131 #define __FINITE_MATH_ONLY__ 0
00132 #define __FLT16_DECIMAL_DIG__ 5
00133 #define __FLT16_DENORM_MIN__ 5.960464477539062 5e-8F16
00134 #define __FLT16_DIG__ 3
00135 #define __FLT16_EPSILON__ 9.765625e-4F16
00136 #define __FLT16_HAS_DENORM__ 1
00137 #define __FLT16_HAS_INFINITY__ 1
00138 #define __FLT16_HAS_QUIET_NAN__ 1
00139 #define __FLT16_MANT_DIG__ 11
00140 #define __FLT16_MAX_10_EXP__ 4
00141 #define __FLT16_MAX_EXP__ 16
00142 #define __FLT16_MAX__ 6.5504e+4F16
00143 #define __FLT16_MIN_10_EXP__ (-4)
00144 #define __FLT16_MIN_EXP__ (-13)
00145 #define __FLT16_MIN__ 6.103515625e-5F16
00146 #define __FLT16_NORM_MAX__ 6.5504e+4F16
00147 #define __FLT_DECIMAL_DIG__ 9
00148 #define __FLT_DENORM_MIN__ 1.40129846e-45F
00149 #define __FLT_DIG__ 6
00150 #define __FLT_EPSILON__ 1.19209290e-7F
00151 #define __FLT_HAS_DENORM__ 1
00152 #define __FLT_HAS_INFINITY__ 1
00153 #define __FLT_HAS_QUIET_NAN__ 1
00154 #define __FLT_MANT_DIG__ 24
00155 #define __FLT_MAX_10_EXP__ 38
00156 #define __FLT_MAX_EXP__ 128
00157 #define __FLT_MAX__ 3.40282347e+38F
00158 #define __FLT_MIN_10_EXP__ (-37)
00159 #define __FLT_MIN_EXP__ (-125)
00160 #define __FLT_MIN__ 1.17549435e-38F
00161 #define __FLT_NORM_MAX__ 3.40282347e+38F
00162 #define __FLT_RADIX__ 2
00163 #define __FPCLASS_NEGINF 0x0004
00164 #define __FPCLASS_NEGNORMAL 0x0008
00165 #define __FPCLASS_NEGSUBNORMAL 0x0010
00166 #define __FPCLASS_NEGZERO 0x0020
00167 #define __FPCLASS_POSINF 0x0200
00168 #define __FPCLASS_POSNORMAL 0x0100
00169 #define __FPCLASS_POSSUBNORMAL 0x0080
00170 #define __FPCLASS_POSZERO 0x0040
00171 #define __FPCLASS_QNAN 0x0002
00172 #define __FPCLASS_SNAN 0x0001
00173 #define __FP_FAST_FMA 1
00174 #define __FP_FAST_FMAF 1
00175 #define __GCC_ASM_FLAG_OUTPUTS__ 1
00176 #define __GCC_ATOMIC_BOOL_LOCK_FREE 2
00177 #define __GCC_ATOMIC_CHAR16_T_LOCK_FREE 2
00178 #define __GCC_ATOMIC_CHAR32_T_LOCK_FREE 2
00179 #define __GCC_ATOMIC_CHAR_LOCK_FREE 2
00180 #define __GCC_ATOMIC_INT_LOCK_FREE 2
00181 #define __GCC_ATOMIC_LLONG_LOCK_FREE 2
00182 #define __GCC_ATOMIC_LONG_LOCK_FREE 2
00183 #define __GCC_ATOMIC_POINTER_LOCK_FREE 2
00184 #define __GCC_ATOMIC_SHORT_LOCK_FREE 2
00185 #define __GCC_ATOMIC_TEST_AND_SET_TRUEVAL 1
00186 #define __GCC_ATOMIC_WCHAR_T_LOCK_FREE 2
00187 #define __GCC_CONSTRUCTIVE_SIZE 64
00188 #define __GCC_DESTRUCTIVE_SIZE 64
00189 #define __GCC_HAVE_DWARF2_CFI_ASM 1
00190 #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_1 1
00191 #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_16 1
00192 #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_2 1
00193 #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_4 1
00194 #define __GCC_HAVE_SYNC_COMPARE_AND_SWAP_8 1
00195 #define __GLIBCXX_BITSIZE_INT_N_0 128
```

```
00196 #define __GLIBCXX_TYPE_INT_N_0 __int128
00197 #define __GNUC_GNU_INLINE__ 1
00198 #define __GNUC_MINOR__ 2
00199 #define __GNUC_PATCHLEVEL__ 1
00200 #define __GNUC__ 4
00201 #define __GNUG__ 4
00202 #define __GXX_ABI_VERSION 1002
00203 #define __GXX_EXPERIMENTAL_CXX0X__ 1
00204 #define __GXX_RTTI 1
00205 #define __GXX_WEAK__ 1
00206 #define __HAVE_FUNCTION_MULTI_VERSIONING 1
00207 #define __INT16_C_SUFFIX__
00208 #define __INT16_FMTd__ "hd"
00209 #define __INT16_FMTi__ "hi"
00210 #define __INT16_MAX__ 32767
00211 #define __INT16_TYPE__ short
00212 #define __INT32_C_SUFFIX__
00213 #define __INT32_FMTd__ "d"
00214 #define __INT32_FMTi__ "i"
00215 #define __INT32_MAX__ 2147483647
00216 #define __INT32_TYPE__ int
00217 #define __INT64_C_SUFFIX__ LL
00218 #define __INT64_FMTd__ "lld"
00219 #define __INT64_FMTi__ "lli"
00220 #define __INT64_MAX__ 9223372036854775807LL
00221 #define __INT64_TYPE__ long long int
00222 #define __INT8_C_SUFFIX__
00223 #define __INT8_FMTd__ "hhd"
00224 #define __INT8_FMTi__ "hhi"
00225 #define __INT8_MAX__ 127
00226 #define __INT8_TYPE__ signed char
00227 #define __INTMAX_C_SUFFIX__ L
00228 #define __INTMAX_FMTd__ "ld"
00229 #define __INTMAX_FMTi__ "li"
00230 #define __INTMAX_MAX__ 9223372036854775807L
00231 #define __INTMAX_TYPE__ long int
00232 #define __INTMAX_WIDTH__ 64
00233 #define __INTPTR_FMTd__ "ld"
00234 #define __INTPTR_FMTi__ "li"
00235 #define __INTPTR_MAX__ 9223372036854775807L
00236 #define __INTPTR_TYPE__ long int
00237 #define __INTPTR_WIDTH__ 64
00238 #define __INT_FAST16_FMTd__ "hd"
00239 #define __INT_FAST16_FMTi__ "hi"
00240 #define __INT_FAST16_MAX__ 32767
00241 #define __INT_FAST16_TYPE__ short
00242 #define __INT_FAST16_WIDTH__ 16
00243 #define __INT_FAST32_FMTd__ "d"
00244 #define __INT_FAST32_FMTi__ "i"
00245 #define __INT_FAST32_MAX__ 2147483647
00246 #define __INT_FAST32_TYPE__ int
00247 #define __INT_FAST32_WIDTH__ 32
00248 #define __INT_FAST64_FMTd__ "lld"
00249 #define __INT_FAST64_FMTi__ "lli"
00250 #define __INT_FAST64_MAX__ 9223372036854775807LL
00251 #define __INT_FAST64_TYPE__ long long int
00252 #define __INT_FAST64_WIDTH__ 64
00253 #define __INT_FAST8_FMTd__ "hhd"
00254 #define __INT_FAST8_FMTi__ "hhi"
00255 #define __INT_FAST8_MAX__ 127
00256 #define __INT_FAST8_TYPE__ signed char
00257 #define __INT_FAST8_WIDTH__ 8
00258 #define __INT_LEAST16_FMTd__ "hd"
00259 #define __INT_LEAST16_FMTi__ "hi"
00260 #define __INT_LEAST16_MAX__ 32767
00261 #define __INT_LEAST16_TYPE__ short
00262 #define __INT_LEAST16_WIDTH__ 16
00263 #define __INT_LEAST32_FMTd__ "d"
00264 #define __INT_LEAST32_FMTi__ "i"
00265 #define __INT_LEAST32_MAX__ 2147483647
00266 #define __INT_LEAST32_TYPE__ int
00267 #define __INT_LEAST32_WIDTH__ 32
00268 #define __INT_LEAST64_FMTd__ "lld"
00269 #define __INT_LEAST64_FMTi__ "lli"
00270 #define __INT_LEAST64_MAX__ 9223372036854775807LL
00271 #define __INT_LEAST64_TYPE__ long long int
00272 #define __INT_LEAST64_WIDTH__ 64
00273 #define __INT_LEAST8_FMTd__ "hhd"
00274 #define __INT_LEAST8_FMTi__ "hhi"
00275 #define __INT_LEAST8_MAX__ 127
00276 #define __INT_LEAST8_TYPE__ signed char
00277 #define __INT_LEAST8_WIDTH__ 8
00278 #define __INT_MAX__ 2147483647
00279 #define __INT_WIDTH__ 32
00280 #define __LDBL_DECIMAL_DIG__ 17
00281 #define __LDBL_DENORM_MIN__ 4.9406564584124654e-324L
00282 #define __LDBL_DIG__ 15
```

```
00283 #define __LDBL_EPSILON__ 2.2204460492503131e-16L
00284 #define __LDBL_HAS_DENORM__ 1
00285 #define __LDBL_HAS_INFINITY__ 1
00286 #define __LDBL_HAS_QUIET_NAN__ 1
00287 #define __LDBL_MANT_DIG__ 53
00288 #define __LDBL_MAX_10_EXP__ 308
00289 #define __LDBL_MAX_EXP__ 1024
00290 #define __LDBL_MAX__ 1.7976931348623157e+308L
00291 #define __LDBL_MIN_10_EXP__ (-307)
00292 #define __LDBL_MIN_EXP__ (-1021)
00293 #define __LDBL_MIN__ 2.2250738585072014e-308L
00294 #define __LDBL_NORM_MAX__ 1.7976931348623157e+308L
00295 #define __LITTLE_ENDIAN__ 1
00296 #define __LLONG_WIDTH__ 64
00297 #define __LONG_LONG_MAX__ 9223372036854775807LL
00298 #define __LONG_MAX__ 9223372036854775807L
00299 #define __LONG_WIDTH__ 64
00300 #define __LP64__ 1
00301 #define __MACH__ 1
00302 #define __MEMORY_SCOPE_DEVICE 1
00303 #define __MEMORY_SCOPE_SINGLE 4
00304 #define __MEMORY_SCOPE_SYSTEM 0
00305 #define __MEMORY_SCOPE_WRKGRP 2
00306 #define __MEMORY_SCOPE_WVFRNT 3
00307 #define __NO_INLINE__ 1
00308 #define __NO_MATH_ERRNO__ 1
00309 #define __OBJC_BOOL_IS_BOOL 1
00310 #define __OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES 3
00311 #define __OPENCL_MEMORY_SCOPE_DEVICE 2
00312 #define __OPENCL_MEMORY_SCOPE_SUB_GROUP 4
00313 #define __OPENCL_MEMORY_SCOPE_WORK_GROUP 1
00314 #define __OPENCL_MEMORY_SCOPE_WORK_ITEM 0
00315 #define __ORDER_BIG_ENDIAN__ 4321
00316 #define __ORDER_LITTLE_ENDIAN__ 1234
00317 #define __ORDER_PDP_ENDIAN__ 3412
00318 #define __PIC__ 2
00319 #define __POINTER_WIDTH__ 64
00320 #define __PRAGMA_REDEFINE_EXTNAME 1
00321 #define __PTRDIFF_FMTd__ "ld"
00322 #define __PTRDIFF_FMTi__ "li"
00323 #define __PTRDIFF_MAX__ 9223372036854775807L
00324 #define __PTRDIFF_TYPE__ long int
00325 #define __PTRDIFF_WIDTH__ 64
00326 #define __REGISTER_PREFIX__
00327 #define __SCHAR_MAX__ 127
00328 #define __SHRT_MAX__ 32767
00329 #define __SHRT_WIDTH__ 16
00330 #define __SIG_ATOMIC_MAX__ 2147483647
00331 #define __SIG_ATOMIC_WIDTH__ 32
00332 #define __SIZEOF_DOUBLE__ 8
00333 #define __SIZEOF_FLOAT__ 4
00334 #define __SIZEOF_INT128__ 16
00335 #define __SIZEOF_INT__ 4
00336 #define __SIZEOF_LONG_DOUBLE__ 8
00337 #define __SIZEOF_LONG_LONG__ 8
00338 #define __SIZEOF_LONG__ 8
00339 #define __SIZEOF_POINTER__ 8
00340 #define __SIZEOF_PTRDIFF_T__ 8
00341 #define __SIZEOF_SHORT__ 2
00342 #define __SIZEOF_SIZE_T__ 8
00343 #define __SIZEOF_WCHAR_T__ 4
00344 #define __SIZEOF_WINT_T__ 4
00345 #define __SIZE_FMTX__ "lX"
00346 #define __SIZE_FMTo__ "lo"
00347 #define __SIZE_FMTu__ "lu"
00348 #define __SIZE_FMTx__ "lx"
00349 #define __SIZE_MAX__ 18446744073709551615UL
00350 #define __SIZE_TYPE__ long unsigned int
00351 #define __SIZE_WIDTH__ 64
00352 #define __SSP__ 1
00353 #define __STDCPP_DEFAULT_NEW_ALIGNMENT__ 16UL
00354 #define __STDCPP_THREADS__ 1
00355 #define __STDC_EMBED_EMPTY__ 2
00356 #define __STDC_EMBED_FOUND__ 1
00357 #define __STDC_EMBED_NOT_FOUND__ 0
00358 #define __STDC_HOSTED__ 1
00359 #define __STDC_NO_THREADS__ 1
00360 #define __STDC_UTF_16__ 1
00361 #define __STDC_UTF_32__ 1
00362 #define __STDC__ 1
00363 #define __UINT16_C_SUFFIX__
00364 #define __UINT16_FMTX__ "hX"
00365 #define __UINT16_FMTo__ "ho"
00366 #define __UINT16_FMTu__ "hu"
00367 #define __UINT16_FMTx__ "hx"
00368 #define __UINT16_MAX__ 65535
00369 #define __UINT16_TYPE__ unsigned short
```

```
00370 #define __UINT32_C_SUFFIX__ U
00371 #define __UINT32_FMTX__ "X"
00372 #define __UINT32_FMTo__ "o"
00373 #define __UINT32_FMTu__ "u"
00374 #define __UINT32_FMTx__ "x"
00375 #define __UINT32_MAX__ 4294967295U
00376 #define __UINT32_TYPE__ unsigned int
00377 #define __UINT64_C_SUFFIX__ ULL
00378 #define __UINT64_FMTX__ "llX"
00379 #define __UINT64_FMTo__ "llo"
00380 #define __UINT64_FMTu__ "llu"
00381 #define __UINT64_FMTx__ "llx"
00382 #define __UINT64_MAX__ 18446744073709551615ULL
00383 #define __UINT64_TYPE__ long long unsigned int
00384 #define __UINT8_C_SUFFIX__
00385 #define __UINT8_FMTX__ "hhX"
00386 #define __UINT8_FMTo__ "hho"
00387 #define __UINT8_FMTu__ "hhu"
00388 #define __UINT8_FMTx__ "hhx"
00389 #define __UINT8_MAX__ 255
00390 #define __UINT8_TYPE__ unsigned char
00391 #define __UINTMAX_C_SUFFIX__ UL
00392 #define __UINTMAX_FMTX__ "lX"
00393 #define __UINTMAX_FMTo__ "lo"
00394 #define __UINTMAX_FMTu__ "lu"
00395 #define __UINTMAX_FMTx__ "lx"
00396 #define __UINTMAX_MAX__ 18446744073709551615UL
00397 #define __UINTMAX_TYPE__ long unsigned int
00398 #define __UINTMAX_WIDTH__ 64
00399 #define __UINTPTR_FMTX__ "lX"
00400 #define __UINTPTR_FMTo__ "lo"
00401 #define __UINTPTR_FMTu__ "lu"
00402 #define __UINTPTR_FMTx__ "lx"
00403 #define __UINTPTR_MAX__ 18446744073709551615UL
00404 #define __UINTPTR_TYPE__ long unsigned int
00405 #define __UINTPTR_WIDTH__ 64
00406 #define __UINT_FAST16_FMTX__ "hX"
00407 #define __UINT_FAST16_FMTo__ "ho"
00408 #define __UINT_FAST16_FMTu__ "hu"
00409 #define __UINT_FAST16_FMTx__ "hx"
00410 #define __UINT_FAST16_MAX__ 65535
00411 #define __UINT_FAST16_TYPE__ unsigned short
00412 #define __UINT_FAST32_FMTX__ "X"
00413 #define __UINT_FAST32_FMTo__ "o"
00414 #define __UINT_FAST32_FMTu__ "u"
00415 #define __UINT_FAST32_FMTx__ "x"
00416 #define __UINT_FAST32_MAX__ 4294967295U
00417 #define __UINT_FAST32_TYPE__ unsigned int
00418 #define __UINT_FAST64_FMTX__ "llX"
00419 #define __UINT_FAST64_FMTo__ "llo"
00420 #define __UINT_FAST64_FMTu__ "llu"
00421 #define __UINT_FAST64_FMTx__ "llx"
00422 #define __UINT_FAST64_MAX__ 18446744073709551615ULL
00423 #define __UINT_FAST64_TYPE__ long long unsigned int
00424 #define __UINT_FAST8_FMTX__ "hhX"
00425 #define __UINT_FAST8_FMTo__ "hho"
00426 #define __UINT_FAST8_FMTu__ "hhu"
00427 #define __UINT_FAST8_FMTx__ "hhx"
00428 #define __UINT_FAST8_MAX__ 255
00429 #define __UINT_FAST8_TYPE__ unsigned char
00430 #define __UINT_LEAST16_FMTX__ "hX"
00431 #define __UINT_LEAST16_FMTo__ "ho"
00432 #define __UINT_LEAST16_FMTu__ "hu"
00433 #define __UINT_LEAST16_FMTx__ "hx"
00434 #define __UINT_LEAST16_MAX__ 65535
00435 #define __UINT_LEAST16_TYPE__ unsigned short
00436 #define __UINT_LEAST32_FMTX__ "X"
00437 #define __UINT_LEAST32_FMTo__ "o"
00438 #define __UINT_LEAST32_FMTu__ "u"
00439 #define __UINT_LEAST32_FMTx__ "x"
00440 #define __UINT_LEAST32_MAX__ 4294967295U
00441 #define __UINT_LEAST32_TYPE__ unsigned int
00442 #define __UINT_LEAST64_FMTX__ "llX"
00443 #define __UINT_LEAST64_FMTo__ "llo"
00444 #define __UINT_LEAST64_FMTu__ "llu"
00445 #define __UINT_LEAST64_FMTx__ "llx"
00446 #define __UINT_LEAST64_MAX__ 18446744073709551615ULL
00447 #define __UINT_LEAST64_TYPE__ long long unsigned int
00448 #define __UINT_LEAST8_FMTX__ "hhX"
00449 #define __UINT_LEAST8_FMTo__ "hho"
00450 #define __UINT_LEAST8_FMTu__ "hhu"
00451 #define __UINT_LEAST8_FMTx__ "hhx"
00452 #define __UINT_LEAST8_MAX__ 255
00453 #define __UINT_LEAST8_TYPE__ unsigned char
00454 #define __USER_LABEL_PREFIX__ _
00455 #define __VERSION__ "Apple LLVM 17.0.0 (clang-1700.0.13.3)"
00456 #define __WCHAR_MAX__ 2147483647
```

```
00457 #define __WCHAR_TYPE__ int
00458 #define __WCHAR_WIDTH__ 32
00459 #define __WINT_MAX__ 2147483647
00460 #define __WINT_TYPE__ int
00461 #define __WINT_WIDTH__ 32
00462 #define __aarch64__ 1
00463 #define __apple_build_version__ 17000013
00464 #define __arm64 1
00465 #define __arm64__ 1
00466 #define __block __attribute__((__blocks__(byref)))
00467 #define __clang__ 1
00468 #define __clang_literal_encoding__ "UTF-8"
00469 #define __clang_major__ 17
00470 #define __clang_minor__ 0
00471 #define __clang_patchlevel__ 0
00472 #define __clang_version__ "17.0.0 (clang-1700.0.13.3)"
00473 #define __clang_wide_literal_encoding__ "UTF-32"
00474 #define __cplusplus 201703L
00475 #define __cpp_aggregate_bases 201603L
00476 #define __cpp_aggregate_nsdmi 201304L
00477 #define __cpp_alias_templates 200704L
00478 #define __cpp_aligned_new 201606L
00479 #define __cpp_attributes 200809L
00480 #define __cpp_binary_literals 201304L
00481 #define __cpp_capture_star_this 201603L
00482 #define __cpp_constexpr 201603L
00483 #define __cpp_constexpr_in_decltype 201711L
00484 #define __cpp_decltype 200707L
00485 #define __cpp_decltype_auto 201304L
00486 #define __cpp_deduction_guides 201703L
00487 #define __cpp_delegating_constructors 200604L
00488 #define __cpp_deleted_function 202403L
00489 #define __cpp_digit_separators 201309L
00490 #define __cpp_enumerator_attributes 201411L
00491 #define __cpp_exceptions 199711L
00492 #define __cpp_fold_expressions 201603L
00493 #define __cpp_generic_lambdas 201304L
00494 #define __cpp_guaranteed_copy_elision 201606L
00495 #define __cpp_hex_float 201603L
00496 #define __cpp_if_constexpr 201606L
00497 #define __cpp_impl_destroying_delete 201806L
00498 #define __cpp_inheriting_constructors 201511L
00499 #define __cpp_init_captures 201304L
00500 #define __cpp_initializer_lists 200806L
00501 #define __cpp_inline_variables 201606L
00502 #define __cpp_lambdas 200907L
00503 #define __cpp_named_character_escapes 202207L
00504 #define __cpp_namespace_attributes 201411L
00505 #define __cpp_nested_namespace_definitions 201411L
00506 #define __cpp_noexcept_function_type 201510L
00507 #define __cpp_nontype_template_args 201411L
00508 #define __cpp_nontype_template_parameter_auto 201606L
00509 #define __cpp_nsdmi 200809L
00510 #define __cpp_pack_indexing 202311L
00511 #define __cpp_placeholder_variables 202306L
00512 #define __cpp_range_based_for 201603L
00513 #define __cpp_raw_strings 200710L
00514 #define __cpp_ref_qualifiers 200710L
00515 #define __cpp_return_type_deduction 201304L
00516 #define __cpp_rtti 199711L
00517 #define __cpp_rvalue_references 200610L
00518 #define __cpp_static_assert 201411L
00519 #define __cpp_static_call_operator 202207L
00520 #define __cpp_structured_bindings 202403L
00521 #define __cpp_template_auto 201606L
00522 #define __cpp_template_template_args 201611L
00523 #define __cpp_threadsafe_static_init 200806L
00524 #define __cpp_unicode_characters 200704L
00525 #define __cpp_unicode_literals 200710L
00526 #define __cpp_user_defined_literals 200809L
00527 #define __cpp_variable_templates 201304L
00528 #define __cpp_variadic_templates 200704L
00529 #define __cpp_variadic_using 201611L
00530 #define __llvm__ 1
00531 #define __nonnull _Nonnull
00532 #define __null_unspecified _Null_unspecified
00533 #define __nullable _Nullable
00534 #define __pic__ 2
00535 #define __private_extern__ extern
00536 #define __strong
00537 #define __unsafe_unretained
00538 #define __weak __attribute__((objc_gc(weak)))
```

8.35 NetScanner_autogen/mocs_compilation.cpp 文件参考

#include "JRIAJ772TK/moc_deviceanalyzer.cpp"
#include "JRIAJ772TK/moc_mainwindow.cpp"
#include "JRIAJ772TK/moc_networkscanner.cpp"
#include "JRIAJ772TK/moc_networktopology.cpp"
#include "JRIAJ772TK/moc_scanhistory.cpp"
#include "EWIEGA46WW/moc_deviceanalyzer.cpp"
#include "EWIEGA46WW/moc_mainwindow.cpp"
mocs_compilation.cpp 的引用 (Include) 关系图:

8.36 networkscanner.cpp 文件参考

网络扫描器类的实现
#include "networkscanner.h"
#include <QDebug>
#include <QTime>
#include <QTimer>
#include <QMutexLocker>
#include <QProcess>
#include <QFile>
#include <QTextStream>
#include <QCoreApplication>
#include <QDir>
#include <QRegularExpression>
#include <QNetworkInterface>
#include <QThread>
#include <QEventLoop>
#include <QElapsedTimer>
#include <QMessageBox>
#include <QRandomGenerator>
#include <random>
#include <algorithm>
#include <QSemaphore>
#include <QScopedPointer>
networkscanner.cpp 的引用 (Include) 关系图:

8.36.1 详细描述

网络扫描器类的实现
提供网络设备发现和端口扫描功能的实现

作者

Network Scanner Team

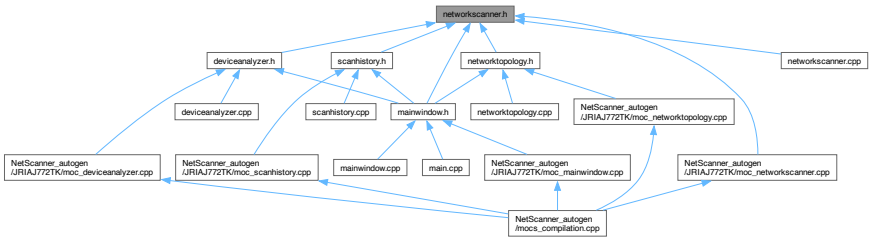
版本

2.1.0

8.37 networkscanner.h 文件参考

网络扫描器类定义
#include <QObject>
#include <QHostAddress>
#include <QList>
#include <QMap>
#include <QMutex>
#include <QTcpSocket>
#include <QDateTime>
#include <QNetworkInterface>
#include <QFuture>
#include <QtConcurrent/QtConcurrent>
#include <QHostInfo>
#include <QThreadPool>
#include <QRunnable>
#include <QElapsedTimer>
#include <QSemaphore>
networkscanner.h 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- struct HostInfo
  存储主机信息的结构体
- class ScanStrategy
  扫描策略类
- class ScanTask
  扫描任务类
- class NetworkScanner
  网络扫描器类

枚举

- enum ScanMode { QUICK , STANDARD , THOROUGH }
  扫描模式枚举 (旧的，建议使用 ScanStrategy::ScanMode)

8.37.1 详细描述

网络扫描器类定义
提供网络设备发现和端口扫描功能

作者

Network Scanner Team

版本

2.1.0

8.37.2 枚举类型说明

8.37.2.1 ScanMode

enum ScanMode
扫描模式枚举 (旧的，建议使用 ScanStrategy::ScanMode)

枚举值

QUICK	快速扫描
STANDARD	标准扫描
THOROUGH	彻底扫描

8.38 networkscanner.h

浏览该文件的文档.

```
00001 //
00002 // networkscanner.h
00003 //
00004
00012
00013 #ifndef NETWORKSCANNER_H
00014 #define NETWORKSCANNER_H
00015
00016 #include <QObject>
00017 #include <QHostAddress>
00018 #include <QList>
00019 #include <QMap>
00020 #include <QMutex>
00021 #include <QTcpSocket>
00022 #include <QDateTime>
00023 #include <QNetworkInterface>
00024 #include <QFuture>
00025 #include <QtConcurrent/QtConcurrent>
00026 #include <QHostInfo>
00027 #include <QThreadPool>
00028 #include <QRunnable>
00029 #include <QElapsedTimer>
00030 #include <QSemaphore>
00031
00037 struct HostInfo {
00038     QString ipAddress;
00039     QString hostName;
00040     QString macAddress;
00041     QString macVendor;
00042     bool isReachable;
00043     QDateTime scanTime;
00044     QMap<int, bool> openPorts;
00045 };
00046
00051 enum ScanMode {
00052     QUICK,
00053     STANDARD,
00054     THOROUGH
00055 };
00056
00062 class ScanStrategy {
00063 public:
00068     enum ScanMode {
00069         QUICK_SCAN,
00070         STANDARD_SCAN,
00071         DEEP_SCAN
00072     };
00073
00078     ScanStrategy(ScanMode mode = STANDARD_SCAN);
```

```
00079
00084     QList<int> getPortsToScan() const;
00085
00091     int getScanTimeout(const QString &ip) const;
00092
00097     int getMaxParallelTasks() const;
00098
00104     void updateHostResponseTime(const QString &ip, int responseTime);
00105
00110     ScanMode getMode() const { return m_mode; }
00111
00116     void setMode(ScanMode mode) { m_mode = mode; }
00117
00118 private:
00119     ScanMode m_mode;
00120     QMap<QString, int> m_hostResponseTimes;
00121 };
00122
00128 class ScanTask : public QRunnable {
00129 public:
00137     ScanTask(QObject* parent, const QHostAddress &address,
00138              const QList<int> &ports, int timeout);
00139
00144     void run() override;
00145
00146     QObject* m_parent;
00147     QHostAddress m_address;
00148     QList<int> m_ports;
00149     int m_timeout;
00150 };
00151
00157 class NetworkScanner : public QObject
00158 {
00159     Q_OBJECT
00160
00161 public:
00166     explicit NetworkScanner(QObject *parent = nullptr);
00167
00172     ~NetworkScanner();
00173
00178     void setCustomPortsToScan(const QList<int> &ports);
00179
00184     void setScanTimeout(int msecs);
00185
00192     void setCustomIPRange(const QString &startIP, const QString &endIP);
00193
00198     void setScanStrategy(ScanStrategy::ScanMode mode);
00199
00204     QList<HostInfo> getScannedHosts() const;
00205
00210     void saveResultsToFile(const QString &filename) const;
00211
00217     void startScan();
00218
00223     void stopScan();
00224
00229     bool isScanning() const;
00230
00236     QList<QHostAddress> quickPingScan(const QList<QHostAddress> &addresses);
00237
00244     bool isHostReachable(const QHostAddress &address, int timeout);
00245
00253     bool isReachableOnPorts(const QHostAddress &address, const QList<int> &ports, int timeout);
00254
00259     void scanHost(const QHostAddress &address);
00260
00266     QString lookupHostName(const QHostAddress &address);
00267
00273     QString lookupMacAddress(const QHostAddress &address);
00274
00280     QString lookupMacVendor(const QString &macAddress);
00281
00287     QString generatePseudoMACFromIP(const QString &ip);
00288
00293     void setScanMode(ScanMode mode) { m_scanMode = mode; }
00294
00299     void setDebugMode(bool debug) { m_debugMode = debug; }
00300
00305     void setRandomizeScan(bool randomize) { m_randomizeScan = randomize; }
00306
00313     bool checkHostReachable(const QHostAddress &address, int timeout);
00314
00315 signals:
00320     void hostFound(const HostInfo &host);
00321
00325     void scanStarted();
00326
```

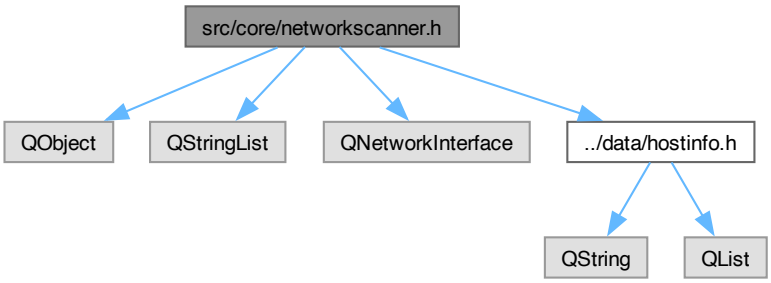
```
00330     void scanFinished();
00331
00336     void scanProgress(int progress);
00337
00342     void scanError(const QString &errorMessage);
00343
00344 public slots:
00349     void onScanTaskFinished(const HostInfo &hostInfo);
00350
00354     void updateScanProgress();
00355
00360     void onHostNameLookedUp(const QHostInfo &hostInfo);
00361
00362 private:
00367     QList<QHostAddress> getLocalNetworkAddresses();
00368
00374     QString normalizeMacAddress(const QString &macAddress);
00375
00382     bool pingTargetWithTimeout(const QString &ip, int timeout);
00383
00389     QString getMacAddressFromSystemCalls(const QString &ip);
00390
00395     QList<QPair<QHostAddress, QString» performARPScan();
00396
00401     void scanHostPorts(HostInfo &hostInfo);
00402
00407     QList<QHostAddress> getAddressesToScan();
00408
00412     void processScanResults();
00413
00415     void terminateAllPingProcesses();
00417     void forceKillAllPingProcesses();
00418
00419     bool m_isScanning;
00420
00421     int m_scannedHosts;
00422     int m_totalHosts;
00423     int m_scanTimeout;
00424
00425     bool m_useCustomRange;
00426     QHostAddress m_startIPRange;
00427     QHostAddress m_endIPRange;
00428
00429     QList<int> m_portsToScan;
00430
00431     QList<HostInfo> m_scannedHostsList;
00432     QMutex m_mutex;
00433
00434     QList<QFuture<void» m_scanFutures;
00435     QThreadPool m_threadPool;
00436
00437     QMap<QString, QString> m_macAddressCache;
00438
00439     ScanStrategy m_scanStrategy;
00440
00441     QList<QHostAddress> m_activeHosts;
00442
00443     ScanMode m_scanMode;
00444     bool m_debugMode;
00445     bool m_randomizeScan;
00446
00447     QElapsedTimer m_scanStartTime;
00448     QDateTime m_lastProgressUpdate;
00449
00461     static bool executeProcess(const QString &program, const QStringList &args, QString &stdOutOutput, QString
        &stdErrOutput, int startTimeout = 2000, int finishTimeout = 5000);
00462
00463     QSemaphore* m_pingSemaphore;
00464     const int m_maxConcurrentPings = 8;
00465
00466     // 新增的负责 ping 操作的方法
00474     bool pingHost(const QString& ipAddress, int timeoutMs = 1500);
00475 };
00476
00477 #endif // NETWORKSCANNER_H
```

8.39 src/core/networkscanner.h 文件参考

定义网络扫描器核心类，负责执行网络发现和主机信息收集。

#include <QObject>
#include <QStringList>
#include <QNetworkInterface>

#include "../data/hostinfo.h"
networkscanner.h 的引用 (Include) 关系图:



类

- class NetworkScanner
  网络扫描器类

8.39.1 详细描述

定义网络扫描器核心类，负责执行网络发现和主机信息收集。

版权所有

Copyright (c) 2024

8.40 networkscanner.h

浏览该文件的文档.

```
00001
00008
00009 #ifndef NETWORKSCANNER_H
00010 #define NETWORKSCANNER_H
00011
00012 #include <QObject> // NetworkScanner 可能需要 QObject 以支持信号和槽
00013 #include <QStringList>
00014 #include <QNetworkInterface> // 用于获取本地网络接口信息
00015 #include "../data/hostinfo.h" // 包含 HostInfo 结构体
00016
00027 class NetworkScanner : public QObject
00028 {
00029     Q_OBJECT
00030
00031 public:
00036     explicit NetworkScanner(QObject *parent = nullptr);
00037
00041     ~NetworkScanner();
00042
00048     void setScanTargets(const QStringList &targets);
00049
00056     // void setScanConfiguration(const ScanConfig &config); // ScanConfig 需要定义
00057
00064     void startScan();
00065
00069     void stopScan();
00070
00075     bool isScanning() const;
00076
00081     QList<QNetworkInterface> getLocalInterfaces();
00082
00083     // 更多具体的扫描方法可以根据需要添加，例如：
00084     // void performArpScan(const QString &subnet); // 执行 ARP 扫描
00085     // void performPingSweep(const QString &subnet); // 执行 Ping 扫描
00086     // void scanPorts(const QString &ipAddress, const QList<int> &ports); // 扫描指定主机的端口
```

```
00087
00088 signals:
00093     void hostFound(const HostInfo &hostInfo);
00094
00100     void scanProgress(int percentage, const QString &message);
00101
00106     void scanFinished(const QList<HostInfo> &results);
00107
00112     void scanError(const QString &errorMessage);
00113
00114 private slots:
00115     // 私有槽函数，用于处理内部异步操作的结果等
00116
00117 private:
00118     QStringList m_targets;
00119     bool m_isScanning;
00120     QList<HostInfo> m_foundHosts;
00121
00122     // 内部实现细节，例如线程、进程管理、原始套接字等
00123 };
00124
00125 #endif // NETWORKSCANNER_H
```

8.41 networktopology.cpp 文件参考

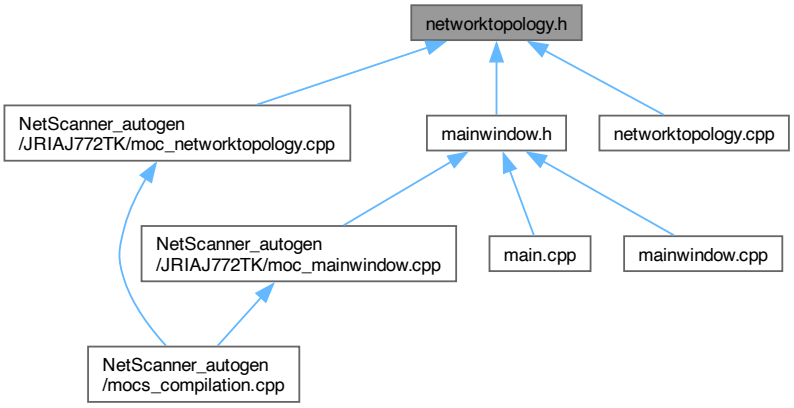
#include ”networktopology.h”
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QStyleOptionGraphicsItem>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QLabel>
#include <QPushButton>
#include <QComboBox>
#include <QApplication>
#include <QPalette>
#include <QDebug>
#include <QtMath>
#include <QGraphicsItemAnimation>
#include <QTimeLine>
#include <QToolTip>
#include <QScrollBar>
#include <QProcess>
#include <QRegularExpression>
networktopology.cpp 的引用 (Include) 关系图:

8.42 networktopology.h 文件参考

#include <QWidget>
#include <QGraphicsView>
#include <QGraphicsScene>
#include <QGraphicsItem>
#include <QMap>
#include <QPair>
#include <QProcess>
#include ”networkscanner.h”

networktopology.h 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- class TopologyAnalyzer

  网络拓扑分析器类。
- class DeviceNode

  代表网络拓扑图中的一个设备节点，继承自 QGraphicsItem。
- class ConnectionLine

  代表网络拓扑图中两个设备节点之间的连接线，继承自 QGraphicsItem。
- class NetworkTopologyView

  网络拓扑图的视图类，继承自 QGraphicsView。
- class NetworkTopology

  网络拓扑组件的主控件，继承自 QWidget。

枚举

- enum DeviceType { DEVICE_UNKNOWN , DEVICE_ROUTER , DEVICE_SERVER , DEVICE_PC , DEVICE_MOBILE , DEVICE_PRINTER , DEVICE_IOT }

  定义网络中可能出现的设备类型。
- enum ConnectionType { CONNECTION_UNKNOWN , CONNECTION_DIRECT , CONNECTION_WIRELESS , CONNECTION_VPN , CONNECTION_ROUTED }

  定义设备之间的连接类型。

8.42.1 枚举类型说明

8.42.1.1 ConnectionType

enum ConnectionType
定义设备之间的连接类型。

枚举值

CONNECTION_UNKNOWN	未知连接类型
CONNECTION_DIRECT	直接物理连接 (例如，以太网)
CONNECTION_WIRELESS	无线连接 (例如，Wi-Fi)
CONNECTION_VPN	VPN 连接
CONNECTION_ROUTED	通过一个或多个路由器间接连接

8.42.1.2 DeviceType

enum DeviceType
定义网络中可能出现的设备类型。

枚举值

DEVICE_UNKNOWN	未知设备类型
DEVICE_ROUTER	路由器
DEVICE_SERVER	服务器
DEVICE_PC	个人电脑
DEVICE_MOBILE	移动设备 (手机、平板等)
DEVICE_PRINTER	网络打印机
DEVICE_IOT	物联网设备 (智能家居等)

8.43 networktopology.h

浏览该文件的文档.

```
00001 #ifndef NETWORKTOPOLOGY_H
00002 #define NETWORKTOPOLOGY_H
00003
00004 #include <QWidget>
00005 #include <QGraphicsView>
00006 #include <QGraphicsScene>
00007 #include <QGraphicsItem>
00008 #include <QMap>
00009 #include <QPair>
00010 #include <QProcess>
00011 #include "networkscanner.h"
00012
00016 enum DeviceType {
00017     DEVICE_UNKNOWN,
00018     DEVICE_ROUTER,
00019     DEVICE_SERVER,
00020     DEVICE_PC,
00021     DEVICE_MOBILE,
00022     DEVICE_PRINTER,
00023     DEVICE_IOT
00024 };
00025
00029 enum ConnectionType {
00030     CONNECTION_UNKNOWN,
00031     CONNECTION_DIRECT,
00032     CONNECTION_WIRELESS,
00033     CONNECTION_VPN,
00034     CONNECTION_ROUTED
00035 };
```

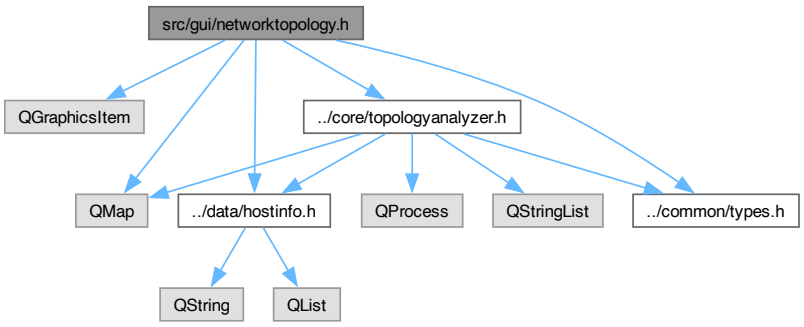
```
00036
00043 class TopologyAnalyzer {
00044 public:
00048     TopologyAnalyzer();
00049
00056     QMap<QString, QStringList> inferDeviceConnections(const QList<HostInfo> &hosts);
00057
00064     QMap<int, QStringList> analyzeTTLLayers(const QList<HostInfo> &hosts);
00065
00071     QMap<QString, QStringList> analyzeSubnets(const QList<HostInfo> &hosts);
00072
00079     QList<QStringList> clusterDevicesByResponseTime(const QList<HostInfo> &hosts);
00080
00086     int getTTLValue(const QString &ipAddress);
00087
00094     QStringList performTraceRoute(const QString &targetIP);
00095
00102     QString calculateSubnet(const QString &ip, int prefixLength = 24);
00103
00111     bool inSameSubnet(const QString &ip1, const QString &ip2, int prefixLength = 24);
00112
00113 private:
00114     // 已移除 private 中的 calculateSubnet 声明
00115 };
00116
00122 class DeviceNode : public QGraphicsItem
00123 {
00124 public:
00130     DeviceNode(const HostInfo &host, DeviceType type = DEVICE_UNKNOWN);
00131
00136     QRectF boundingRect() const override;
00145     void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;
00146
00151     void setDeviceType(DeviceType type);
00156     DeviceType deviceType() const { return m_type; }
00157
00162     QString ipAddress() const { return m_host.ipAddress; }
00167     HostInfo hostInfo() const { return m_host; }
00172     void setPosition(const QPointF &pos);
00173
00174     // 新增：设置网络层级
00179     void setNetworkLayer(int layer) { m_networkLayer = layer; }
00184     int networkLayer() const { return m_networkLayer; }
00185
00186     // 新增：设置子网组
00191     void setSubnetGroup(const QString &subnet) { m_subnetGroup = subnet; }
00196     QString subnetGroup() const { return m_subnetGroup; }
00197
00198 protected:
00199     void mousePressEvent(QGraphicsSceneMouseEvent *event) override;
00200     void mouseMoveEvent(QGraphicsSceneMouseEvent *event) override;
00201     void mouseReleaseEvent(QGraphicsSceneMouseEvent *event) override;
00202     void hoverEnterEvent(QGraphicsSceneHoverEvent *event) override;
00203     void hoverLeaveEvent(QGraphicsSceneHoverEvent *event) override;
00204
00205 private:
00206     HostInfo m_host;
00207     DeviceType m_type;
00208     bool m_highlight;
00209     QPointF m_dragStartPosition;
00210     int m_networkLayer;
00211     QString m_subnetGroup;
00212 };
00213
00219 class ConnectionLine : public QGraphicsItem
00220 {
00221 public:
00228     ConnectionLine(DeviceNode *source, DeviceNode *target,
00229                    ConnectionType type = CONNECTION_DIRECT);
00230
00231     QRectF boundingRect() const override;
00232     void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;
00233
00237     void updatePosition();
00238
00243     void setConnectionType(ConnectionType type);
00248     ConnectionType connectionType() const { return m_connectionType; }
00249
00250 private:
00251     DeviceNode *m_source;
00252     DeviceNode *m_target;
00253     ConnectionType m_connectionType;
00254 };
00255
00262 class NetworkTopologyView : public QGraphicsView
00263 {
00264     Q_OBJECT
```

```
00265
00266 public:
00271     NetworkTopologyView(QWidget *parent = nullptr);
00272
00278     void setHosts(const QList<HostInfo> &hosts);
00282     void clear();
00286     void autoLayout();
00287
00292     void hierarchicalLayout(const QMap<int, QStringList> &layers);
00297     void groupedLayout(const QMap<QString, QStringList> &groups);
00298
00299 signals:
00304     void nodeSelected(const HostInfo &host);
00305
00306 private:
00307     QGraphicsScene *m_scene;
00308     QMap<QString, DeviceNode*> m_nodes;
00309     QList<ConnectionLine*> m_connections;
00310
00311     TopologyAnalyzer m_analyzer;
00312
00318     DeviceType determineDeviceType(const HostInfo &host);
00325     void createConnection(DeviceNode *source, DeviceNode *target,
00326                           ConnectionType type = CONNECTION_DIRECT);
00327 };
00328
00334 class NetworkTopology : public QWidget
00335 {
00336     Q_OBJECT
00337
00338 public:
00343     NetworkTopology(QWidget *parent = nullptr);
00344
00349     void updateTopology(const QList<HostInfo> &hosts);
00353     void clear();
00354
00359     void scale(qreal factor);
00363     void resetView();
00364
00369     void setLayoutMode(int mode);
00370
00371 signals:
00376     void deviceSelected(const HostInfo &host);
00377
00378 private:
00379     NetworkTopologyView *m_topologyView;
00380     QWidget *m_controlPanel;
00381
00385     enum LayoutMode {
00386         LAYOUT_AUTO,
00387         LAYOUT_HIERARCHICAL,
00388         LAYOUT_GROUPED
00389     };
00390
00391     LayoutMode m_layoutMode;
00392     QList<HostInfo> m_currentHosts;
00393 };
00394
00395 #endif // NETWORKTOPOLOGY_H
```

8.44 src/gui/networktopology.h 文件参考

#include <QGraphicsItem>
#include <QMap>
#include "../data/hostinfo.h"
#include "../common/types.h"
#include "../core/topologyanalyzer.h"

networktopology.h 的引用 (Include) 关系图:



8.45 networktopology.h

浏览该文件的文档.

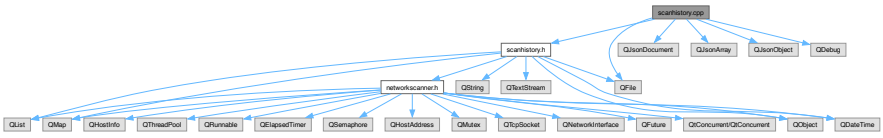
```
00001 #include <QGraphicsItem>
00002 #include <QMap>
00003 // #include <QPair> // QPair 通常不需要单独包含，它是 QMap 的一部分或通过其他 Qt 模块间接包含
00004 // #include <QProcess> // 已移至 TopologyAnalyzer
00005 #include "../data/hostinfo.h"     // 直接包含 HostInfo
00006 #include "../common/types.h"      // 包含 DeviceType, ConnectionType
00007 #include "../core/topologyanalyzer.h" // 包含 TopologyAnalyzer
00008
```

8.46 README.md 文件参考

8.47 scanhistory.cpp 文件参考

```
#include "scanhistory.h"
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>
#include <QFile>
#include <QDebug>
```

scanhistory.cpp 的引用 (Include) 关系图:



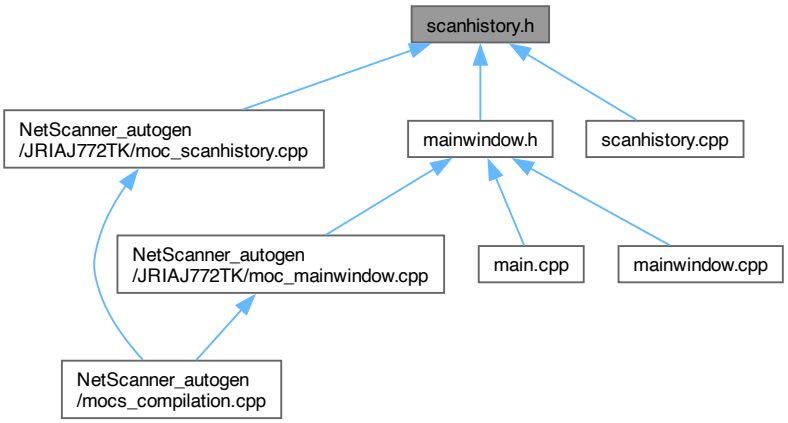
8.48 scanhistory.h 文件参考

扫描历史记录和会话管理类的定义

```
#include <QObject>
#include <QDateTime>
#include <QList>
#include <QMap>
#include <QString>
#include <QFile>
#include <QTextStream>
#include "networkscanner.h"
```

scanhistory.h 的引用 (Include) 关系图:

此图展示该文件被哪些文件直接或间接地引用了:



类

- struct ScanSession

  存储单次扫描会话的信息。
- class ScanHistory

  扫描历史管理器类。

8.48.1 详细描述

扫描历史记录和会话管理类的定义
提供存储、加载、比较和管理多次网络扫描会话的功能。

作者

Network Scanner Team

版本

2.0.1

8.49 scanhistory.h

浏览该文件的文档.

```
00008
00009 #ifndef SCANHISTORY_H
00010 #define SCANHISTORY_H
00011
00012 #include <QObject>
00013 #include <QDateTime>
00014 #include <QList>
00015 #include <QMap>
00016 #include <QString>
00017 #include <QFile>
00018 #include <QTextStream>
00019 #include "networkscanner.h"
00020
00026 struct ScanSession {
00027     QDateTime dateTime;
```

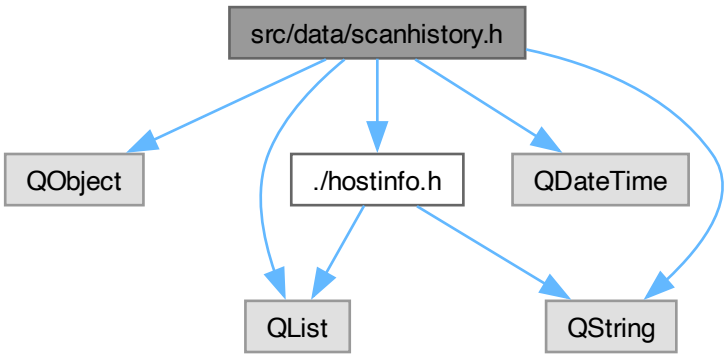
```
00028     QString description;
00029     QList<HostInfo> hosts;
00030     int totalHostsScanned;
00031
00036     ScanSession() : dateTime(QDateTime::currentDateTime()), totalHostsScanned(0) {}
00037
00044     ScanSession(const QString &desc, const QList<HostInfo> &hostList, int totalScanned = 0)
00045         : dateTime(QDateTime::currentDateTime()), description(desc),
00046           hosts(hostList), totalHostsScanned(totalScanned) {}
00047
00052     int totalHosts() const { return hosts.size(); }
00053     int reachableHosts() const;
00054     int unreachableHosts() const { return totalHosts() - reachableHosts(); }
00055     QMap<int, int> portDistribution() const;
00056 };
00057
00064 class ScanHistory : public QObject
00065 {
00066     Q_OBJECT
00067
00068 public:
00073     explicit ScanHistory(QObject *parent = nullptr);
00074
00082     void addSession(const QList<HostInfo> &hosts, const QString &description = "");
00083
00088     QList<ScanSession> getSessions() const;
00089
00095     ScanSession getSession(int index) const;
00096
00101     int sessionCount() const;
00102
00109     bool removeSession(int index);
00110
00115     void clearHistory();
00116
00122     bool saveToFile(const QString &filename) const;
00123
00130     bool loadFromFile(const QString &filename);
00131
00141     QPair<QList<HostInfo>, QList<HostInfo» compareScans(int index1, int index2) const;
00142
00143 signals:
00147     void historyChanged();
00148
00149 private:
00150     QList<ScanSession> m_sessions;
00151 };
00152
00153 #endif // SCANHISTORY_H
```

8.50 src/data/scanhistory.h 文件参考

定义扫描历史记录的管理类和相关数据结构。

```
#include <QObject>
#include <QList>
#include <QString>
#include <QDateTime>
#include "./hostinfo.h"
```

scanhistory.h 的引用 (Include) 关系图:



类

- struct ScanSession

  存储单次扫描会话的信息。
- class ScanHistoryManager

  管理扫描历史记录的类。

8.50.1 详细描述

定义扫描历史记录的管理类和相关数据结构。

版权所有

Copyright (c) 2024

8.51 scanhistory.h

浏览该文件的文档.

```
00001
00008
00009 #ifndef SCANHISTORY_H
00010 #define SCANHISTORY_H
00011
00012 #include <QObject> // For potential signals/slots if history operations are async or notify UI
00013 #include <QList>
00014 #include <QString>
00015 #include <QDateTime>
00016 #include "./hostinfo.h" // HostInfo is in the same 'data' directory
00017
00022 struct ScanSession {
00023     QString sessionId;
00024     QDateTime startTime;
00025     QDateTime endTime;
00026     QString scanTarget;
00027     int deviceCount;
00028     QList<HostInfo> foundHosts;
00029     QString notes;
00030     // 可以添加其他元数据，如扫描配置参数等
00031 };
00032
00040 class ScanHistoryManager : public QObject
00041 {
00042     Q_OBJECT
00043
00044 public:
00050     explicit ScanHistoryManager(const QString &storagePath = QString(), QObject *parent = nullptr);
00051
00057     ~ScanHistoryManager();
00058
00064     bool addScanSession(const ScanSession &session);
00065
```