CRISTIANO FRANÇA DANIEL

DESENVOLVIMENTO DE UMA FERRAMENTA GRÁFICA PARA CONFIGURAÇÃO DE REDE VIA INTERFACES SERIAL, PARALELO E USB

Trabalho de Conclusão apresentado ao Departamento de Ciências da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação Lato Sensu em Administração de Redes Linux, para a obtenção do título de especialista em Administração de Redes Linux.

Orientador

Prof. Joaquim Quinteiro Uchoa

LAVRAS
MINAS GERAIS – BRASIL
2005

CRISTIANO FRANÇA DANIEL

DESENVOLVIMENTO DE UMA FERRAMENTA GRÁFICA PARA CONFIGURAÇÃO DE REDE VIA INTERFACES SERIAL, PARALELO E USB

Trabalho de Conclusão apresentado ao Departamento de Ciências da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação Lato Sensu em Administração de Redes Linux, para a obtenção do título de especialista em Administração de Redes Linux.

APROVADA em _____de_____de _____

Prof. _____

Prof. _____

Prof. _____
UFLA
(Joaquim Uchoa Quinteiro)

LAVRAS
MINAS GERAIS – BRASIL

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 - INTRODUÇÃO

O Linux é bastante flexível e poderoso quando o assunto é conectar duas ou mais máquinas em rede. Este trabalho vai explorar algumas configurações especiais de rede disponíveis do Linux, mais especificamente configurações de rede ponto a ponto utilizando as interfaces de *hardware* serial (RS232), paralela e USB.

Estas interfaces, especialmente a serial e a paralela, estão disponíveis na maioria dos computadores e pode ser bastante útil utilizá-las para conexões de rede quando um computador não possui uma placa de rede *ethernet*.

A finalidade deste trabalho é desenvolver uma ferramenta gráfica que possibilite de maneira rápida e fácil a configuração de uma interface de rede utilizando as portas de comunicação descritas anteriormente (serial, paralela e USB). Além disto o *software* será de código aberto (*open source)* sob os termos da licença GPL (GNU *Public License*).

Embora estas configurações sejam relativamente simples de serem feitas, o uso de uma ferramenta gráfica contribui para que usuários não acostumados com interfaces modo texto se sintam mais a vontade e encorajados a utilizar o sistema GNU Linux.

2 – REVISÃO DE LITERATURA

2.1 Comando *ifconfig*

O comando *ifconfig* do Linux é utilizado para configurar interfaces de rede. A sinopse deste comando, segundo a sua página de manual (*man page)*, é a seguinte:

- ifconfig [interface]

- ifconfig interface [aftype] opções | endereços

Onde a **interface** é o nome da interface de rede. Este nome é composto pelo tipo de interface seguido do número da interface. Por exemplo eth1, onde eth representa o tipo de interface de rede *ethernet* e 1 o número desta interface. Outros exemplos de tipos suportados são plip (rede via inteface paralela), sl e ppp (rede via interface serial), usb (rede via interface usb).

Quando usado sem nenhum argumento, *ifconfig* mostra o estado das interfaces correntemente definidas. Se somente o argumento **interface** é definido, *ifconfig* mostra apenas o estado da interface informada.

O argumento **aftype** define o tipo de família de endereçamento utilizado. Quando não especificado utiliza o tipo **inet** (TCP/IP) que é o padrão. Atualmente as famílias de endereçamento suportadas incluem além do **inet** os tipos **ax25** (AMPR Packet Radio), **ddp** (Appletalk Phase 2), **ipx** (Novell IPX) e **netrom** (AMPR Packet radio).

O parâmetro **endereço** é o endereço IP que será atribuído a interface configurada.

As **opções** mais comumente usadas do *ifconfig* são as seguintes:

•**up**: ativa a interface.

•**down**: desativa a interface.

•**netmask addr** : configura a máscara de rede IP para esta interface.

•**pointopoint [endereço]**: habilita o modo ponto-a-ponto da interface, significando que ela é um *link* direto entre duas máquinas.

A lista de opções completa deste comando pode ser vista na sua página de

manual (*man page)*.

2.2 Comando *route*

O comando *route* do Linux é utilizado para manipular a tabela de roteamentos IP do *kernel* do Linux. O *route* pode configurar rotas estáticas para *hosts* ou redes através de uma interface de rede já configurada.

A sinopse simplificada, levando-se em consideração o uso mais comum do comando *route* é a seguinte:

- route add [-net|-host] Alvo [netmask Nm] [gw Gw] [[dev] If]

- route del [-net|-host] Alvo [netmask Nm] [gw Gw] [[dev] If]

Quando *route* é executado sem nenhum parâmetro, ele mostra toda a tabela de roteamento.

As opções mais utilizadas para o comando *route* são:

•**add**: adiciona uma rota

•**del**: remove uma rota.

•**-net**: o **Alvo** é o endereço de uma rede.

•**-host**: o **Alvo** é o endereço de uma máquina.

•**Alvo**: máquina ou rede destino. Pode ser dado em formato de endereços IP ou o nome da máquina ou da rede.

• **netmask Nm:** máscara da rede do Alvo.

• **gw Gw**: define que qualquer pacote IP para a rede ou máquina destino (Alvo) serão roteadas através do *gateway* especificado (Gw).

• **dev If**: força a associação da rota com o dispositivo de interface de rede (If) especificado.

As interface de rede devem ser configuradas com o comando *ifconfig* antes de se adicionar rotas para estas interfaces com o comando *route.*

A lista de opções completa deste comando pode ser vista na sua página de manual (*man page)*.

2.3 Comando *slattach*

O comando *slattach* do Linux associa uma interface de rede serial a um dispositivo serial.

A sinopse de forma simplificada, levando em consideração o uso mais comum deste comando é:

- slattach [-p proto] [-s speed] [tty]

As opções mais utilizadas para o comando *slattach* são:

- **-p proto**: especifica o tipo de protocolo que será usado na rede serial. Quando o protocolo não é especificado o tipo utilizado será o **cslip** (**slip** comprimido). Outros possíveis protocolos válidos são o **slip** normal, o **kiss** e o **ppp**.

- **-s speed**: usado para informar a velocidade utilizada pelo dispositivo serial.

- **tty**: dispositivo serial que será colocado em modo de rede.

A lista de opções completa deste comando pode ser vista na sua página de manual (*man page)*.

2.4 Rede Via Interface Serial

Uma das maneiras de se implementar uma rede via interface serial no Linux é através do protocolo SLIP (*Serial Line Internet Protocol)* que permite criar uma interface de rede de tipo serial utilizando todos os recursos de uma rede normal. O protocolo SLIP está documentado na RFC 1055 (ROMKEY, 1998).

O SLIP faz parte do código-fonte do *kernel* do Linux, sendo assim ele pode ser compilado como parte integrante do *kernel* ou como módulo do mesmo. No caso de SLIP ser um módulo, ele deve ser carregado antes de ser utilizado. O SLIP utiliza uma interface identificada por sl*, onde * é o número da interface serial.

Para utilizar o SLIP em um dispositivo serial, deve-se associar este dispositivo com uma interface SLIP. Para isto é utilizado o comando *slattach*.

Segundo (SILVA, 2005), a rede via interface serial atinge em média 115,2 Kb/s, bem menor que uma *ethernet* comum que atinge até 100Mb/s. Além disto o cabo serial para conexões deste tipo não deve ultrapassar 15 metros sendo recomendado utilizar um cabo com até 13 metros sob risco de perda de sinal e até mesmo queima da controladora serial.

A maior vantagem deste tipo de conexão é a escalabilidade dado a existência de porta serial em praticamente todos os computadores e o baixíssimo custo de implementação, apenas um cabo *laplink* serial que custa na faixa de dez reais na época de escrita deste trabalho.

2.4.1 *Hardware* Necessário

O *hardware* necessário para usar a rede via interface SLIP são apenas uma porta serial livre em ambas as máquinas e um cabo *laplink* serial.

O cabo *laplink* é um cabo de 7 vias com dois conectores DB25 ou DB9 (depende do conector do computador) fêmea nas pontas. A ligação pino a pino dos terminais dos conectores é dado na Tabela 1 que foi construída segundo (SILVA, 2005).

Terminal 1				***Terminal 2***		
Sinal	**DB9**	**DB25**		**Sinal**	**DB9**	**DB25**
System Ground	5	7		System Ground	5	7
Transmit Data	3	2		Receive Data	2	3
Request to Send	7	4		Clear to Send	8	5
Data Set Ready	6	6		Data Terminal Ready	4	20
Receive Data	2	3		Transmit Data	3	2
Clear to Send	8	5		Request to Send	7	4
Data Terminal Ready	4	20		Data Set Ready	6	6

Tabela 1 - Estrutura do cabo *laplink* serial.

A imagem de um cabo *laplink* serial que possui os dois conectores (DB9 e DB25) ao mesmo tempo é mostrado na Figura 1.

Figura 1 – Cabo *laplink* serial.

2.4.2 Configuração da Interface SLIP

Para utilizar uma interface de rede serial SLIP o *kernel* deve possui suporte a SLIP (SLIP pode também ser um módulo do *kernel).* Caso não exista o suporte a SLIP, o *kernel* precisará ser recompilado com a adição do SLIP ou ainda compilar o SLIP como um módulo e carregá-lo.

Antes de configurar uma interface SLIP deve-se associar um dispositivo serial a interface SLIP. Isto é feito através do comando *slattach* como no exemplo a seguir:

slattach -p slip -s 115200 /dev/ttyS0

O parâmetro **/dev/ttyS0** está associando o primeiro dispositivo da porta serial (RS232) a uma interface SLIP. O parâmetro **-p slip** informa que o protocolo usado na associação é o SLIP e o parâmetro **-s 115200** informa a velocidade do dispositivo em bits por segundos.

Após associar uma interface SLIP a um dispositivo serial, a configuração

de rede via SLIP é feita através do comando *ifconfig* onde a interface que será levantada é o tipo **sl*,** sendo que * é o número da interface serial. Um exemplo de configuração é dado a seguir:

ifconfig sl0 10.0.0.1 pointopoint 10.0.0.2 up

No comando anterior **sl0** é a interface de rede serial que está sendo configurada. O endereço 10.0.0.1 é o endereço IP atribuído a interface. O parâmetro **pointopoint** indica que a conexão é uma conexão ponto a ponto com a interface de IP igual a 10.0.0.2.

Basta fazer esta simples configuração nas duas máquinas, lembrando de atribuir corretamente os endereços IPs, e a conexão de rede via interface serial está pronta para ser usada.

Para finalizar uma interface **sl**, utiliza-se também o comando *ifconfig* passando a interface que será finalizada mais a opção **down**. Como o exemplo a seguir:

ifconfig sl0 down

Esta sessão foi produzida tomando-se como base (SILVA, 2005) e (DAWSON, 1999).

2.5 Rede Via Interface Paralela

A rede via interface paralela no Linux é feita através do protocolo PLIP (*Parallel Line Internet Protocol)* que permite criar uma interface de rede para a porta paralela utilizando todos os recursos de uma rede normal. O protocolo PLIP pode ser visto em detalhes em (CONTROZZI, 1998) e (LAMIRAL, 2002).

O PLIP faz parte do código-fonte do *kernel* do Linux, sendo assim ele pode ser compilado como parte integrante do *kernel* ou como módulo do mesmo. No caso de PLIP ser um módulo, ele deve ser carregado antes de ser utilizado. O PLIP utiliza uma interface identificada por plip*, onde * é o número da porta paralela que servirá de interface de rede.

Segundo (SILVA, 2005), a rede via interface paralela pode atingir até 1

Mb/s, bem menor que uma *ethernet* comum que atinge até 100Mb/s. Além disto o cabo paralelo para conexões deste tipo não pode ultrapassar 4,5 metros sendo recomendado utilizar um cabo com até 2,5 metros sob risco de perda de sinal e até mesmo de queima da controladora paralela.

A maior vantagem deste tipo de conexão é a escalabilidade dado a existência de porta paralela em praticamente todos os computadores e o baixíssimo custo de implementação, apenas um cabo *laplink* paralelo que custa na faixa de dez reais na época de escrita deste trabalho.

2.5.1 *Hardware* Necessário

O *hardware* necessário para usar a rede via interface PLIP são uma porta paralela livre em ambas as máquinas e o cabo *laplink* paralelo.

O cabo *laplink* é um cabo de 15 vias com dois conectores DB25 macho nas pontas. A ligação pino a pino dos terminais dos conectores é dado na Tabela 2 que foi construída segundo (SILVA, 2005).

Terminal 1	*Terminal 2*
1 (STROBE)	1 (STROBE)
2 (D0)	15 (ERROR)
3 (D1)	13 (SLCT)
4 (D2)	12 (PAPOUT)
5 (D3)	10 (ACK)
6 (D4)	11 (BUSY)
10 (ACK)	5 (D3)
11 (BUSY)	6 (D4)
12 (PAPOUT)	4 (D2)
13 (SLCT)	3 (D1)
14 (FEED)	14 (FEED)
15 (ERROR)	2 (D0)
16 (INIT)	16 (INIT)
17 (SLCTIN)	17 (SLCTIN)
25 (GROUND)	25 (GROUND)

Tabela 2 - Estrutura do cabo *laplink* paralelo.

A imagem de um cabo *laplink* paralelo é mostrado na Figura 2.

Figura 2 – Cabo *laplink* paralelo.

2.5.2 Configuração da Interface PLIP

Para utilizar uma interface de rede paralela o *kernel* deve possui suporte a PLIP (PLIP pode também ser um módulo do *kernel)*. Caso não exista o suporte a PLIP, o *kernel* precisará ser recompilado com a adição do PLIP ou ainda compilar o PLIP como um módulo e carregá-lo.

A configuração de rede via PLIP é feita através do comando *ifconfig* onde a interface que será levantada é o tipo **plip*,** sendo que * é o número da interface paralela. Um exemplo de configuração é dado a seguir:

ifconfig plip1 10.0.0.1 pointopoint 10.0.0.2 up

No comando anterior **plip1** é a interface de rede paralela que está sendo configurada e corresponde a porta paralela 1. O endereço 10.0.0.1 é o endereço IP atribuído a interface. O parâmetro **pointopoint** indica que a conexão é uma conexão ponto a ponto com a interface de IP igual a 10.0.0.2.

Basta fazer esta simples configuração nas duas máquinas, lembrando de atribuir corretamente os endereços IPs, e a conexão de rede via interface paralela está pronta para ser usada.

Para finalizar uma interface **plip**, utiliza-se também o comando *ifconfig* passando a interface que será finalizada mais a opção **down**. Como o exemplo a seguir:

ifconfig plip1 down

Esta sessão foi produzida tomando-se como base (SILVA, 2005), (DAWSON, 1999) e (CONTROZZI, 1998).

2.6 Rede Via Interface USB

USB é um barramento tipo mestre-escravo, onde o computador faz o papel do mestre e os diversos tipos de dispositivos USBs são os escravos. Devido a este comportamento dois computadores não podem ser ligados diretamente com um cabo. Embora exista no mercado um cabo USB com os dois conectores do tipo A que possibilita mecanicamente a conexão de dois computadores, o mesmo não deve ser usado para esta finalidade sob risco de queima da controladora USB, pois o cabo irá colocar as controladoras em curto, segundo (BROWNELL, 2004).

Para se fazer uma conexão ponto a ponto de dois computadores é necessário utilizar um dispositivo USB conhecido como cabo *host-to-host* USB ou ainda cabo *laplink* USB. Um dispositivo deste tipo custa na faixa de cem reais na época de escrita deste trabalho.

No Linux é utilizado o *driver* **usbnet** para controlar os dispositivos USB de rede. Detalhes deste *driver* pode ser visto em (BROWNELL, 2004). Ainda segundo (BROWNELL, 2004), alguns dispositivos de rede USB versão 2.0 podem atingir a velocidade de até 480 Mb/s o que supera a velocidade de um dispositivo *ethernet* comum que é de 100 Mb/s.

2.6.1 *Hardware* Necessário

O *hardware* necessário para usar a rede via interface USB são uma porta USB livre em ambas as máquinas e um cabo *host-to-host* USB.

O cabo *host-to-host* USB é um dispositivo USB que trabalha como escravo no barramento para ambos os computadores. Ele possui dois conectores tipo 'A' para conectar nos computadores mestres. A principal diferença deste dispositivo USB para outros dispositivos, é que ele permite a conexão de dois mestres ao mesmo tempo, ao invés de apenas um.

Conforme o *menuconfig* do *kernel* do Linux versão 2.6.11, os *chipsets* dos cabos *host-to-host* USB suportados no Linux são:

- ALi M5632

- AnchorChips 2720
- eTeck
- GeneSys GL620USB-A
- NetChip 1080
- Prolific PL-2301/2302
- KT Technology KC2190

A imagem de um cabo *host-to-host* USB é mostrado na Figura 3.

Figura 3 – Cabo *host-to-host* USB.

2.6.2 Configuração da Interface de Rede USB

Para utilizar uma interface de rede USB o *kernel* deve possui o *driver* **usbnet** dentro dele ou como um módulo. Caso não exista o driver **usbnet**, o *kernel* precisará ser recompilado com a adição deste *driver* ou ainda compilar o *driver* como um módulo e carregá-lo.

Após conectar o cabo *host-to-host* em ambos os computadores deve-se usar o comando *ifconfig* para configurar a interface de rede USB. A interface

USB de rede é do tipo **usb***, onde * é o número do dispositivo USB conectado. Um exemplo de configuração é dado a seguir:

ifconfig usb1 10.0.0.1 pointopoint 10.0.0.2 up

No comando anterior **usb1** é a interface de rede USB que está sendo configurada. O endereço 10.0.0.1 é o endereço IP atribuído a interface. O parâmetro **pointopoint** indica que a conexão é uma conexão ponto a ponto com a interface de IP igual 10.0.0.2.

Basta fazer esta simples configuração nas duas máquinas, lembrando de atribuir corretamente os endereços IPs, e a conexão de rede via interface USB está pronta para ser usada.

Para finalizar uma interface **usb**, utiliza-se também o comando *ifconfig* passando a interface que será finalizada mais a opção **down**. Como o exemplo a seguir:

ifconfig usb1 down

Esta sessão foi produzida tomando-se como base (BROWNELL, 2004) e (HARDS, 2000).

3 – MATERIAL E MÉTODOS

Dado o objetivo deste trabalho de construir um *software* gráfico para a plataforma Linux, e este *software* ser de código aberto baseado na licença GPL (GNU *Public License*) foram utilizadas as seguintes linguagens e ferramentas de desenvolvimento de *software*:

- Linguagem de programação: C++. Esta linguagem de programação foi escolhida por se tratar de uma linguagem bastante utilizada no universo Linux/Unix. Além do que o uso do C++ veio de encontro com a linguagem utilizada pela biblioteca gráfica adotada. O compilador usado para a linguagem C++ foi o GCC versão 3.3.3.

- Biblioteca gráfica: Qt. O Qt foi escolhido como a biblioteca gráfica a ser usada pelo fato de ser uma biblioteca já consolidada na plataforma Linux, poder ser utilizada sob os termos da licença GPL e utilizar a linguagem de programação C++ que era a preferida para este projeto. Foi usada a versão 3.3.3 do Qt.

- Ambiente de desenvolvimento (IDE): KDevelop. O KDevelop foi escolhido por possuir integração com outras ferramentas de desenvolvimento de *software* como por exemplo o GDB, Valgrind, Doxygen, Ident, e outras mais. Outro motivo forte para a escolha do KDevelop foi que ele possui suporte para projetos Qt . Foi usada a versão 3.1.2 do KDevelop.

- Documentação: Doxygen. Adicionalmente foi utilizado a ferramenta de geração de documentação de código-fonte Doxygen, com o objetivo de conseguir um resultado mais profissional do projeto de *software*. Foi usada a versão 1.3.5 do Doxygen.

Um excelente trabalho que faz um estudo de ferramentas de desenvolvimento para Linux é (ATAIDES, 2004).

3.1 Sobre o Qt

O Qt não é apenas uma biblioteca gráfica e sim um *framework* completo para desenvolvimento de aplicações C++ nas plataformas Linux/X11, Windows e

Mac Os X.

O Qt possui uma biblioteca (Qt Class Library) com mais de 400 classes C++ de vários tipos como classe de programação gráfica (GUI), programação de redes, banco de dados, XML, etc. O pacote inclui ainda uma ferramenta de *designer* de aplicações gráficas (Qt Designer), uma ferramenta de internacionalização (Qt Linguist) e ainda um assistente para acessar a vasta documentação do Qt (Qt Assintant).

A licença de uso do Qt é uma licença dupla onde o desenvolvedor pode optar por desenvolver uma aplicação de código aberto e distribuir o seu *software* sob a licença GPL, ou se preferir, ele pode comprar uma licença comercial junto a Trolltech (empresa responsável pelo Qt) e distribuir o seu *software* sob a licença que ele escolher.

O *site* do Qt na internet é o http://www.trolltech.com/products/qt/. O manual de referência utilizado neste trabalho foi (TROLLTECH, 2004) onde pode ser encontrado todos os detalhes de funcionamento do Qt.

3.1.1 Qt Class Library

A biblioteca de classes do Qt tem algumas características e facilidades que ajudam bastante a programação. Entre as características são destaques: a facilidade de tratamento de eventos gráficos e de usuários, o controle de processo de *threads,* o mecanismo robusto de comunicação entre objetos *(signal and slots),* o tratamento de arquivos, a programação de redes, o acesso a banco de dados, o processamento de XML e a integração com OpenGL.

Uma das facilidades do Qt é o seu gerenciamento de memória. Todo objeto Qt ao ser destruído remove da memória também os seus filhos, evitando assim que fiquem na memória objetos que não estejam sendo mais utilizados.

Outra facilidade são as classes de gerenciamento de *layout* de objetos gráficos que possibilitam que objetos sejam redimencionados, dinamicamente e em tempo de execução, de várias formas pré-programadas.

O mecanismo de comunicação entre os objetos Qt é uma das principais facilidades da biblioteca. Este mecanismo funciona da seguinte forma: um *signal* é emitido quando um evento particular ocorre. Um *slot* é uma função que é chamada em resposta a um *signal* particular. Objetos Qt tem vários *signals* e *slots* predefinidos, mas pode-se criar classes herdeiras de classes Qt e definir os próprios *signals* e *slots*. Um *slot* deve ser conectado aos *signals* que necessita responder. Um *signal* pode ser conectado a vários *slots* assim como um *slot* pode receber vários *signals*. Quando *signals* e *slots* possuem parâmetros, estes devem ser dos mesmos tipos.

A seguir um exemplo de uma classe herdeira da classe QObject do Qt que implementa *signals* e *slots*.

```
#include <QObject>

class Counter : public QObject
{
   Q_OBJECT

   public:
   Counter() { m_value = 0; }

   int value() const { return m_value; }

   public slots:
   void setValue(int value);

   signals:
   void valueChanged(int newValue);

   private:
   int m_value;
};

void Counter::setValue(int value)
{
   if (value != m_value) {
       m_value = value;
       emit valueChanged(value);
   }
}
```

As classes que necessitem implementar seus próprios *signals* e *slots* devem mencionar **Q_OBJECT** no topo da declaração. As declarações de *slots* devem ser precedidas da palavra **slots** e os *signals* na palavra **signals**, como pode ser observado em destaque no exemplo anterior.

Slots são como métodos comuns, possuem declaração e corpo. Já os *signals* só possuem a declaração.

Um *signal* é emitido através da diretiva **emit** destacado no exemplo anterior.

O método **connect** de QObject é utilizado para conectar *signals* e *slots*. A seguir um exemplo do uso do **connect** onde foi conectado o *signal* **valueChanged** de **obj1** ao *slot* **setValue** de **obj2**.

```
Counter obj1, obj2;
QObject::connect(&obj1, SIGNAL(valueChanged(int)),
                 &obj2, SLOT(setValue(int)));
```

Para desconectar um *signal* de um *slot* utiliza-se o método **disconnect** de QObject, da mesma forma que é feito com **connect.**

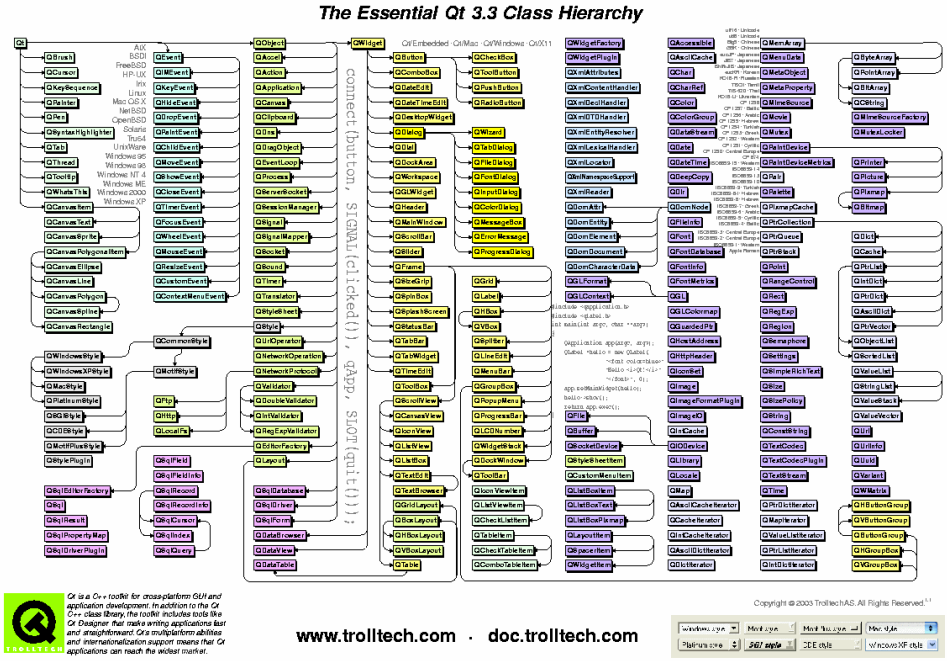

Figura 4 – Diagrama de hierarquia das principais classe do Qt.

3.1.2 Qt Designer

O Qt Designer é uma ferramenta gráfica para construção de telas de aplicativos Qt. Com esta ferramenta é possível construir telas utilizando todos os objetos e controles gráficos do Qt.

A construção das telas é feita de maneira bastante simples, do tipo, clicar e arrastar um objeto até a posição desejada e depois configurar os atributos deste objeto. Após a construção da interface gráfica é possível gerar o seu código-fonte e inseri-lo em qualquer projeto Qt.

A Figura 5 ilustra a tela principal do Qt Designer com um projeto aberto.

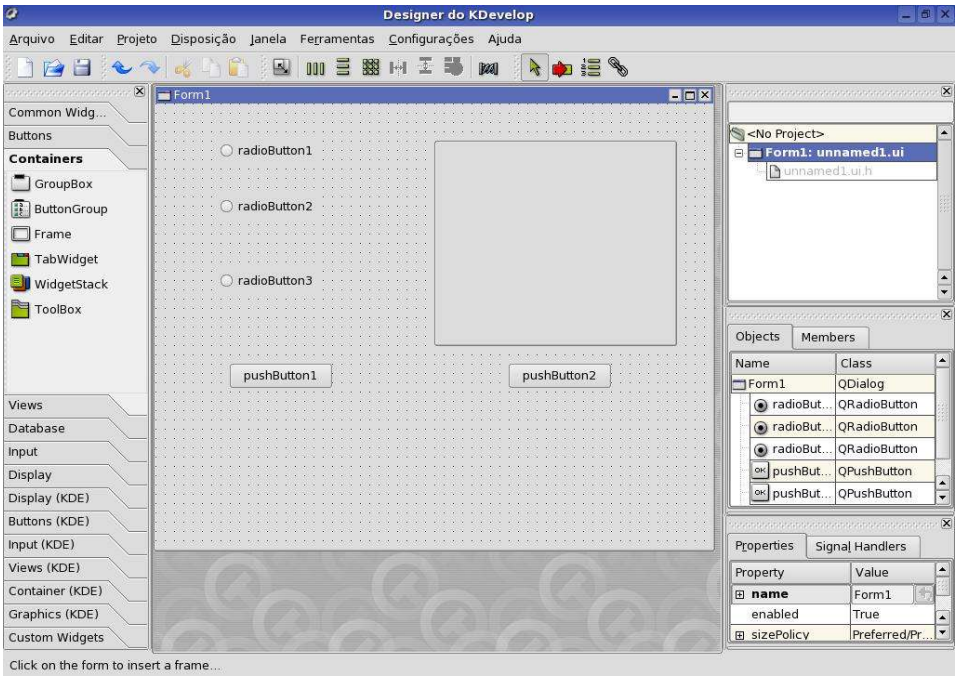

Figura 5 – Tela principal do Qt Designer.

3.1.3 Qt Linguist

O Qt Linguist é uma ferramenta para auxiliar na internacionalização de aplicativos Qt. Com ele é possível gerenciar facilmente os diversos idiomas que o aplicativo terá que possuir.

Esta ferramenta não foi utilizada neste trabalho e fica aqui citada apenas como ilustração e possíveis trabalhos futuros.

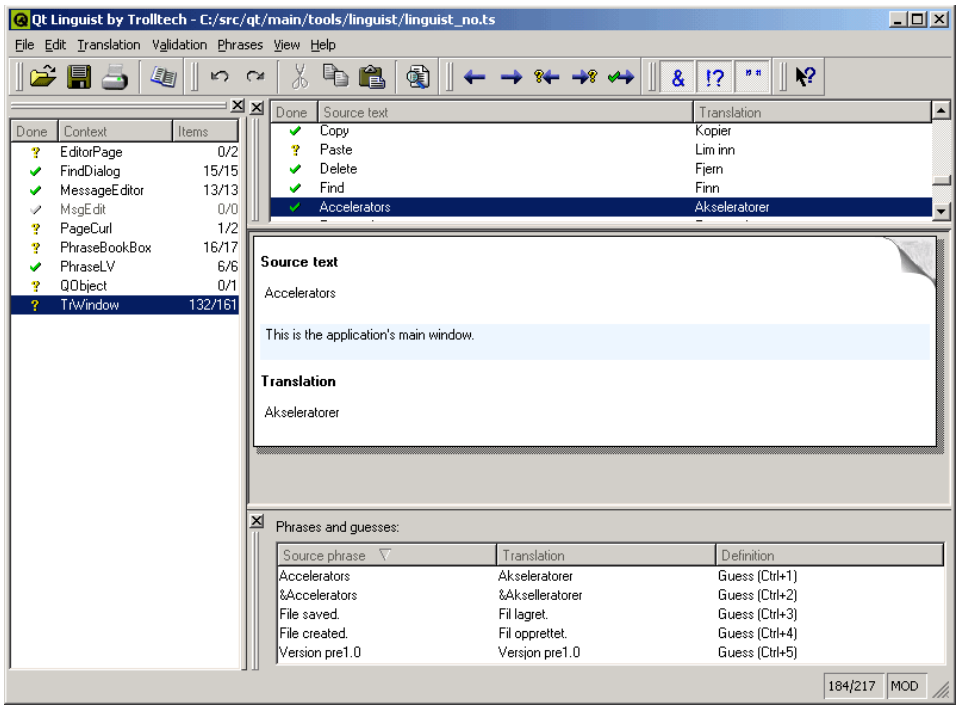

Figura 6 – Tela principal do Qt Linguist.

3.1.4 Qt Assintant

O Qt Assistant é uma ferramenta de ajuda que permite a visualização e busca em toda a documentação do Qt. A ferramenta é muito similar a um *web browser* o que torna o seu uso bastante simples.

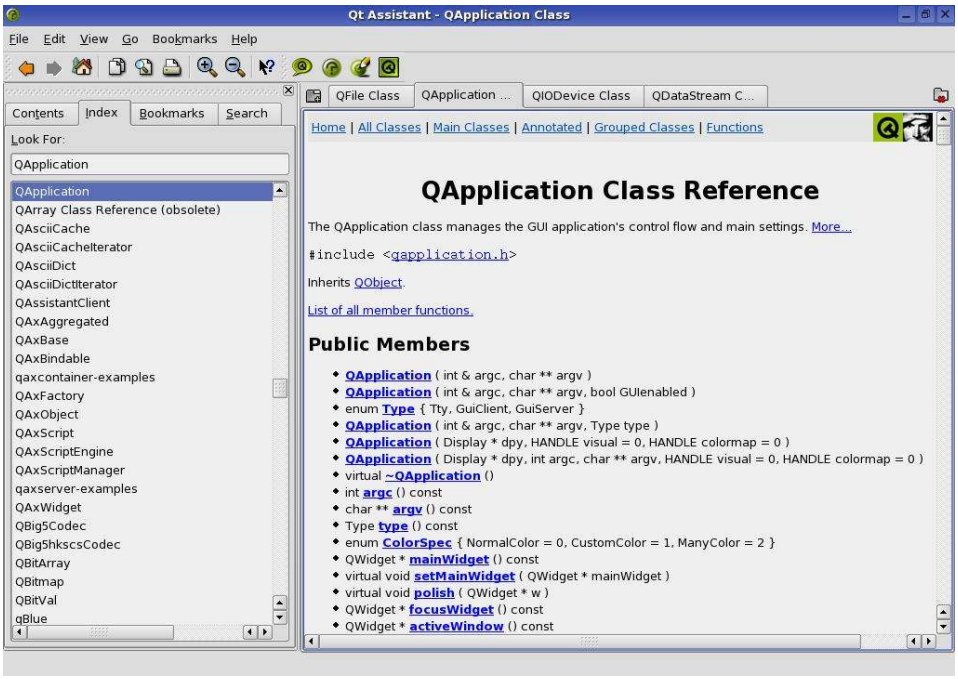

Figura 7 – Tela principal do Qt Assitant.

3.2 Sobre o Doxygen

O Doxygen é uma ferramenta para geração de documentação de códigos-fonte construídos nas linguagens C, C++, Java, Python entre outras.

O Doxygen pode gerar documentação nos formatos HTML, Latex, RTF, PostScript, PDF e páginas de manual Unix (*man pages)*.Ele extrai as informações diretamente do código-fonte, o que facilita a manutenção e a consistência do código-fonte com a documentação.

O Doxygen trabalha varrendo os arquivos de código a procura das estruturas de programação e de palavras chaves colocadas dentro de comentários. Fazendo-se um bom uso estas palavras chaves é possível gerar uma documentação bem completa.

O Doxygen é distribuído sob a licença GPL, e a documentação gerada por ele não está sujeita a esta licença. O *site* na internet do projeto é o

http://www.doxygen.org . O manual completo do Doxygen pode ser consultado em (HEESCH, 2005).

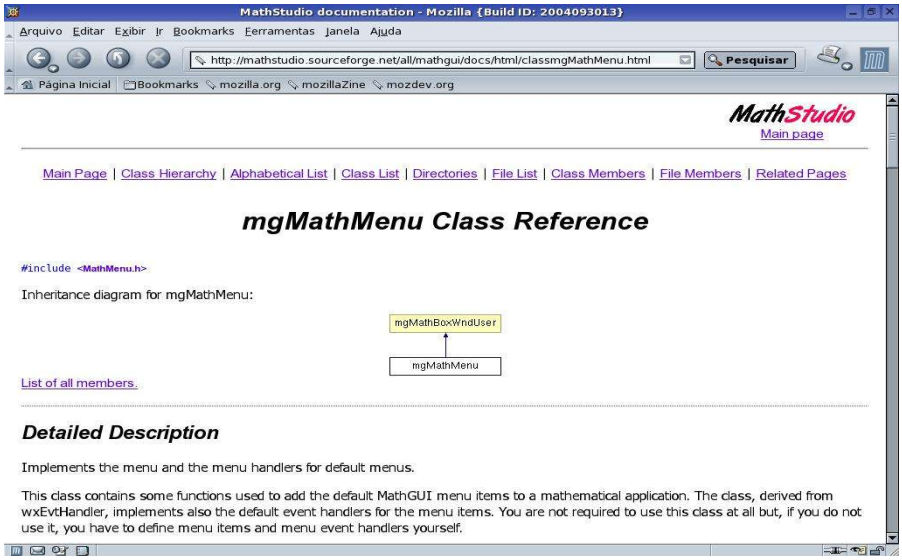

Figura 8 – Documentação *on line* do MathStudio gerado com o Doxygen.

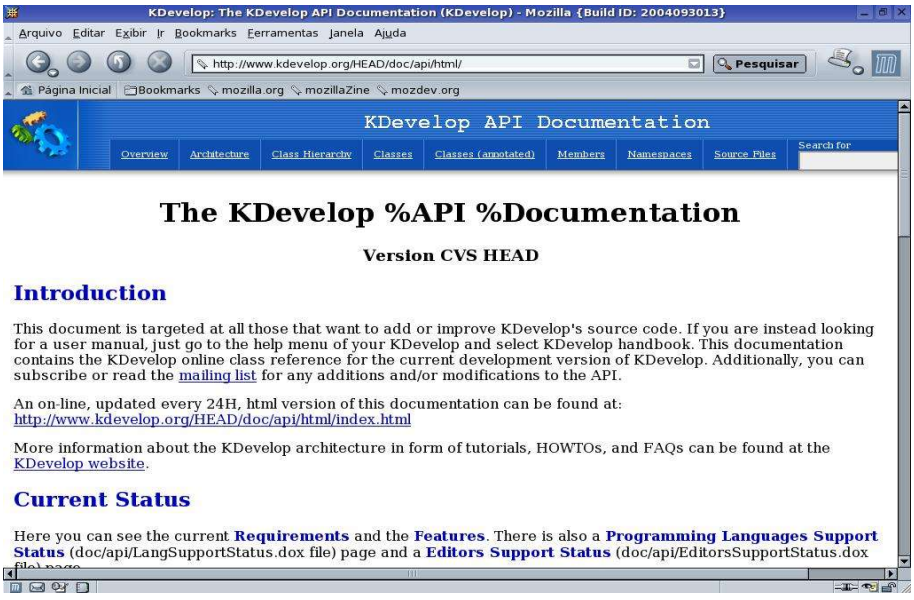

Figura 9 – Documentação *on line* do KDevelop gerado com o Doxygen.

3.3 Sobre o KDevelop

O KDevelop é um IDE (*Integrated Development Enviroment* - Ambiente de Desenvolvimento Integrado) fácil de usar para o KDE. O projeto KDevelop foi iniciado em 1988 e deste então está disponível publicamente sob a licença GPL e suporta várias linguagens de programação, entre elas estão o C/C++, Ada, Fortran, Java, Pascal, Perl, PHP, Python entre outras.

Entre as principais características deste IDE estão:

- Gerenciamento de várias ferramentas de desenvolvimento necessárias para C/C++ como compilador, *linker*, depurador, etc.
- Um gerenciador de arquivos de códigos-fonte e cabeçalhos que integra códigos ao projeto.
- Gerador de classes que cria e integra classes ao projeto.
- Ferramentas de ajuda para criar documentação.
- Integração com controladores de versão como por exemplo o CVS.
- Integração com depuradores.
- *Syntax highlighting* no código-fonte.
- Auto-complementação de código-fonte.
- *Templates* para criação de vários projetos (módulos KControl, Kicker Applets, KIOSlaves, Konqueror *plugins*, projetos Qt, etc).

O *site* do projeto KDevelop na internet é o http://www.kdevelop.org/. O manual completo do KDevelop pode ser consultado em (GEHRMANN, 2004).

As Figuras 10, 11 e 12 ilustram do uso do KDevelop durante o desenvolvimento deste trabalho.

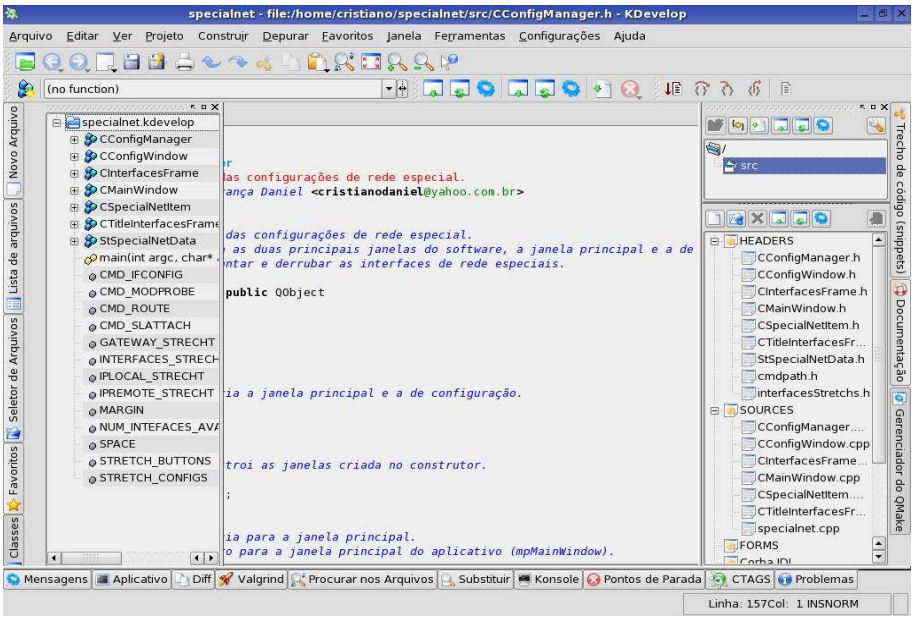

Figura 10 – Gerenciador de classes e de arquivos do KDevelop.

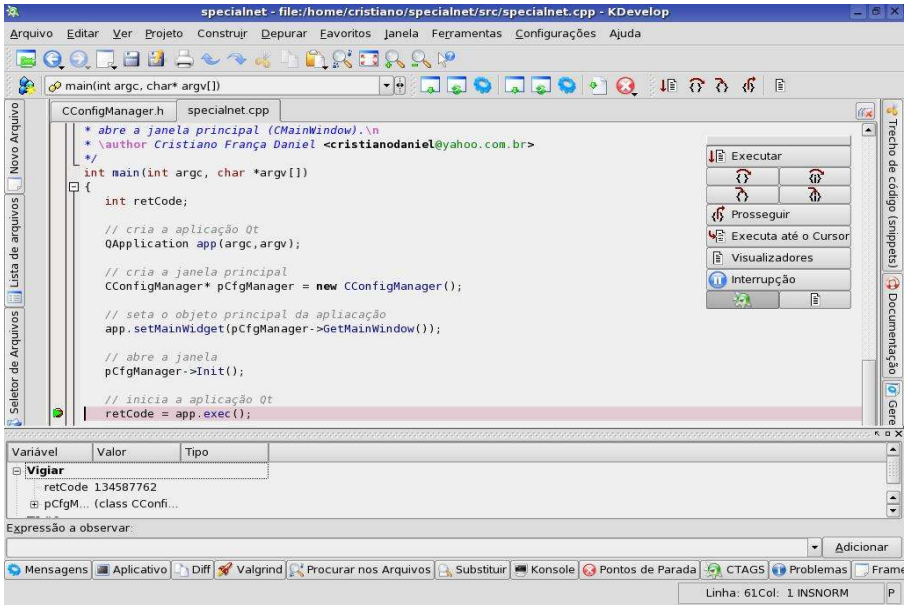

Figura 11 – Depuração com o KDevelop.

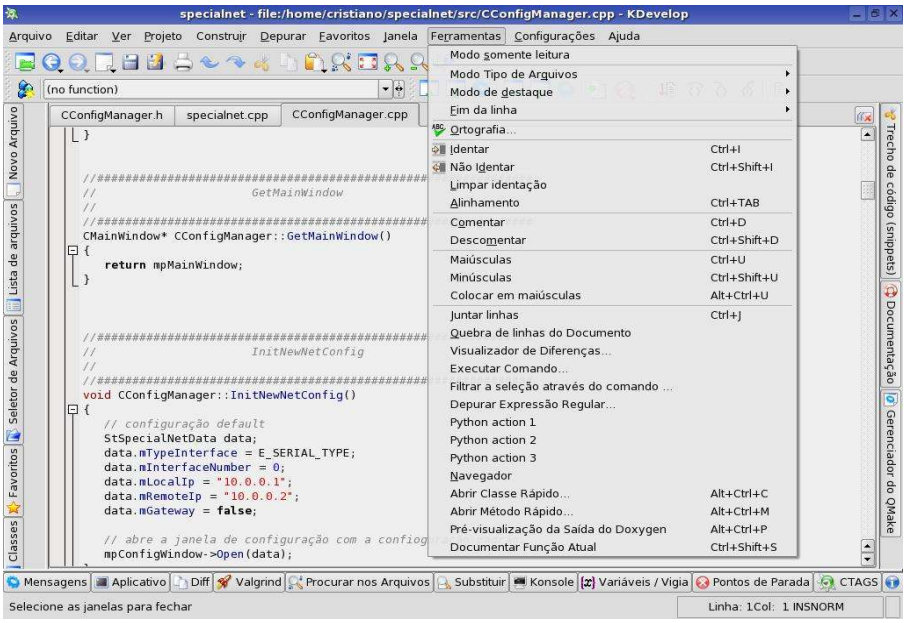

Figura 12 – Opções do menu Ferramentas do KDevelop.

3.3.1 KDevelop e Qt

Os projetos Qt utilizam um arquivo com formato próprio de extensão .**pro** onde estão as configurações do projeto assim como os caminhos completos dos arquivos que o compõem. Para tratar os arquivos de projetos o Qt tem o aplicativo **QMake** que lê os arquivos de projetos e gera o Makefile que posteriormente será utilizado pelo **Make** para costruir o aplicativo.

O KDevelop suporta projetos Qt. Este suporte se dá pela automatização do processo de construção e configuração dos arquivos de projeto (.pro) e da compilação correta do projeto, onde o KDevelop executa automaticamente o QMake antes de executar o Make para construir a aplicação. Isto faz com que o processo de construção de projetos Qt fique trasparente para o desenvolvedor.

O KDevelop possui também integração total com o Qt Designer e com a documentação do Qt, possibilitando assim que o desenvolvedor construa toda a sua aplicação utilizando apenas o IDE.

A Figura 13 ilustra a tela de criação de novos projetos do KDevelop com a opção de projetos Qt em destaque. Esta janela pode ser acessada através do menu **Projeto/Novo Projeto**.

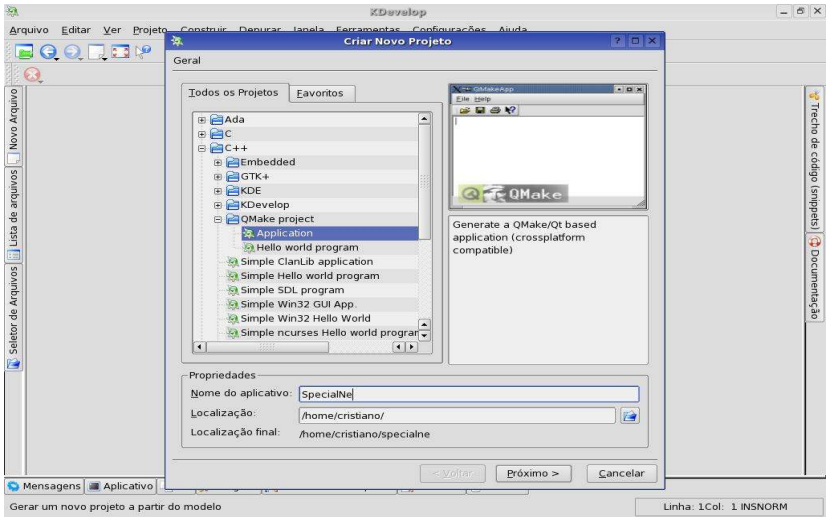

Figura 13 – Criação de projetos no KDevelop.

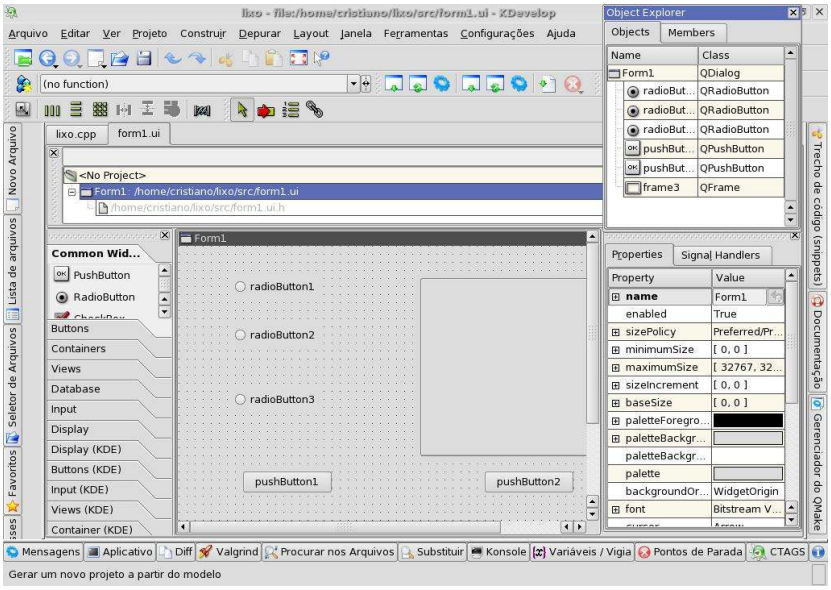

Figura 14 – Qt Designer integrado no KDevelop.

3.3.2 KDevelop e Doxygen

Através das opções do projeto, que pode ser acessado pelo menu **Projeto/Opções do Projeto,** é possível configurar as opções do Doxygen para gerar a documentação do projeto aberto no IDE.

A Figura 15 ilustra a tela de configuração do projeto deste trabalho, com destaque para as configurações do Doxygen.

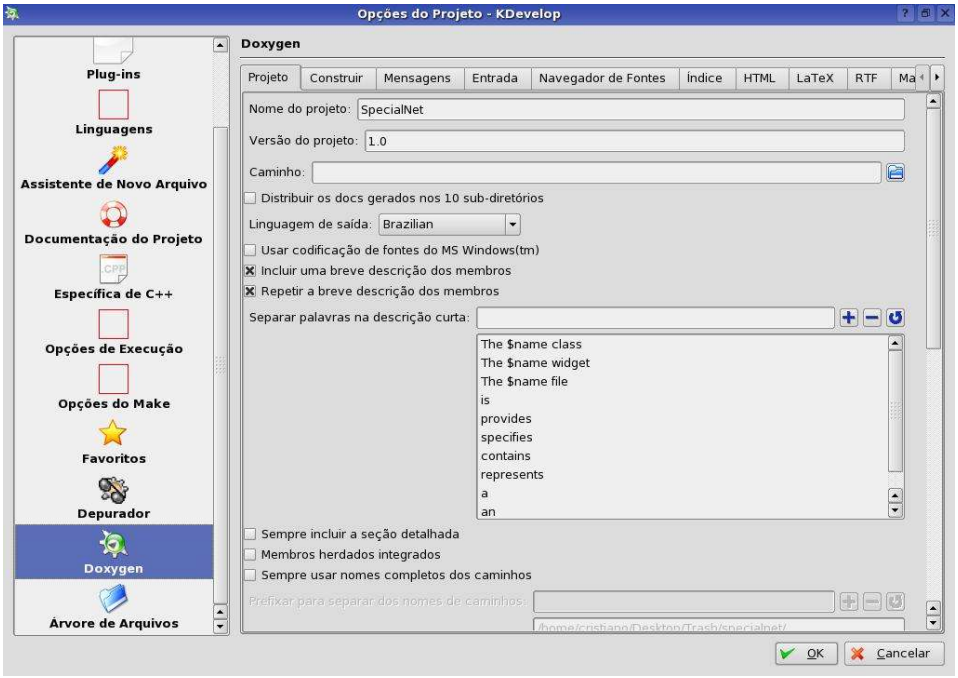

Figura 15 – Configurando o Doxygen através do KDevelop.

Para gerar a documentação do projeto através do KDevelop basta executar o comando **Gerar Documentação da API** através do menu **Construir.** O Doxygen será executado e a documentação será gerada em subdiretórios dentro do diretório do projeto. Estes subdiretórios terão os nomes dos formatos do documento gerado (html, latex, pdf, etc).

4 – RESULTADOS OBTIDOS

Após o estudo das ferramentas, comandos e bibliotecas descritos até aqui, foi desenvolvido o *software* gráfico que recebeu o nome de **SpecialNet**, cuja finalidade é oferecer uma interface gráfica amigável para iniciar e interromper interfaces de redes via portas seriais (RS232), portas paralelas e portas USBs.

4.1 Instalação do SpecialNet

O arquivo **specialnet.tar.gz** que pode ser obtido no *site* http://paginas.terra.com.br/informatica/specialnet/, contém todo o código-fonte do projeto e um *script* para automatizar a compilação da aplicação assim como a instalação.

Para gerar a aplicação serão necessário as ferramentas padrão de desenvolvimento C++, como o Make, GCC, GLibc, etc. Além disto será necessário o pacote de desenvolvimento do Qt3.

Após descompactar o arquivo **specialnet.tar.gz** basta executar com privilégios de *root* o *script* **install** que se encontra dentro do diretório specialnet. Este *script* irá executar o aplicativo Qmake do Qt que irá gerar o Makefile. Em seguida é executado o Make para compilar e ligar os arquivos-fonte. Logo após, o arquivo binário da aplicação (**specialnet**) é copiado para o diretório /sbin e seus atributos de acesso alterados. Após isto, basta executar o comando /sbin/specialnet com privilégios de *root* para abrir a aplicação **SpecialNet**.

4.2 Funcionamento do *software*

O *software* utiliza os comandos *ifconfig, route, slattach* e *modprobe* para configurar as interfaces de redes. Como tais comandos necessitam de privilégios de *root* deve-se executar o **specialnet** também com privilégios de *root.* Caso contrário o programa não irá conseguir configurar as interfaces de rede.

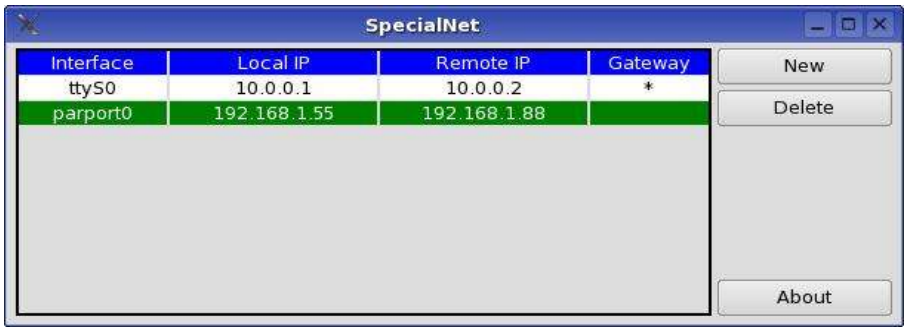

Figura 16 – Tela principal da aplicação SpecialNet.

A Figura 16 mostra a tela principal da aplicação SpecialNet com duas interfaces de rede especiais configuradas. A primeira interface é a serial **ttyS0**, configurada com o IP 10.0.0.1 ligada ponto a ponto com outra interface serial de IP 10.0.0.2 que é também o *gateway* padrão. A segunda interface é a paralela **parport0** configurada com IP 192.168.1.55 ligada ponto a ponto com outra interface paralela de IP 192.168.1.88.

O botão **About** mostra uma tela com informações da versão do SpecialNet e a licença de uso dele. Esta tela é mostrada na Figura 17.

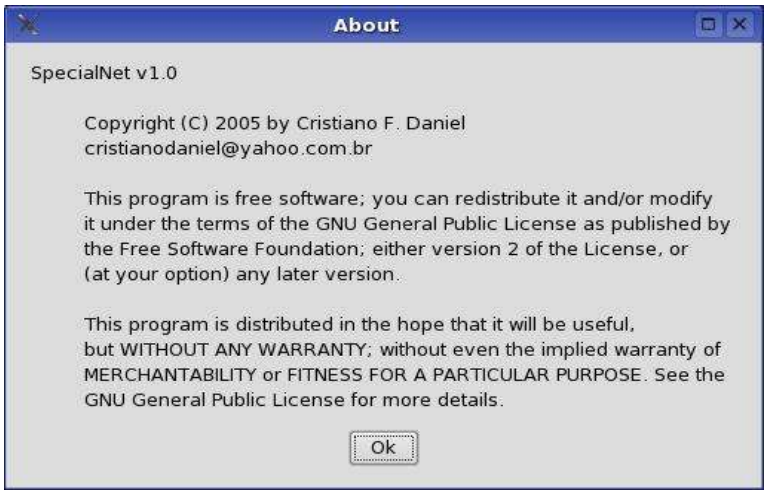

Figura 17 – Tela About da aplicação SpecialNet.

O botão **Delete** remove e desativa a interface que estiver selecionada na tela principal. Segundo a Figura 16 esta interface seria a **parport0**. Antes de remover a interface uma tela de confirmação é exibida (Figura 18).

Figura 18 – Tela de confirmação de remoção de uma interface da aplicação SpecialNet.

Internamente a aplicação executa o seguinte comando para remover uma interface de rede:

ifconfig INTERFACE_LOCAL down

Caso o *gateway* da interface removida esteja ativo, o comando a seguir também é executado:

route del default gw IP_REMOTO

Onde INTERFACE_LOCAL é o tipo e número da interface de rede que será desativada e removida e IP_REMOTO é o endereço IP da interface remota que será desconectada.

O botão **New** abre a tela de configuração de interface de rede especial mostrada na Figura 19.

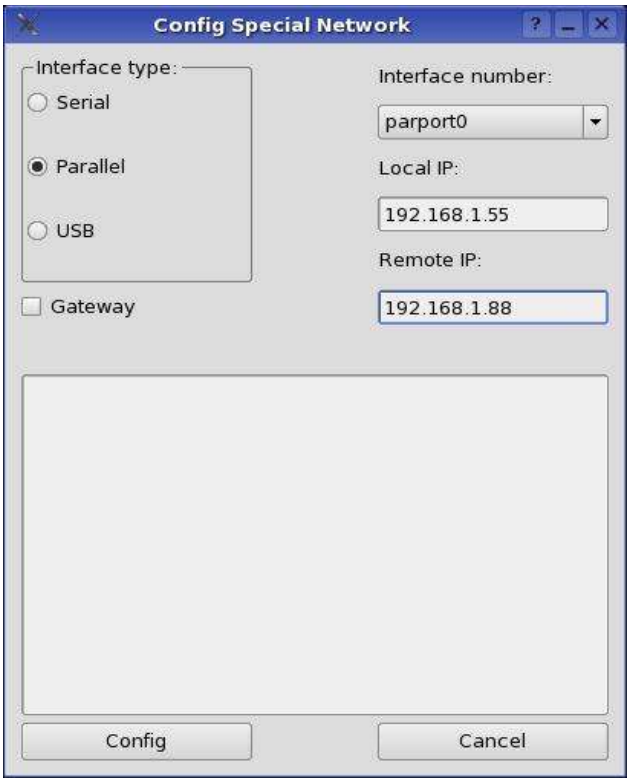

Figura 19 – Tela de configuração de interface de rede especial da aplicação SpecialNet.

Nesta tela de configuração é possível selecionar o tipo de interface que se deseja configurar (serial, paralela ou USB), o número da interface e os endereços IPs da interface local e da interface remota. Todos estes campos são obrigatório, se não forem preenchidos ou preenchidos com erro a configuração da interface irá falhar. Alternativamente pode-se configurar o *gateway* padrão para a interface remota marcando-se a opção **Gateway**.

Após entrar com os dados de configuração basta pressionar o botão **Config** para ativar e configurar a interface. Em caso de desistência da configuração basta pressionar o botão **Cancel** e voltar para a tela principal.

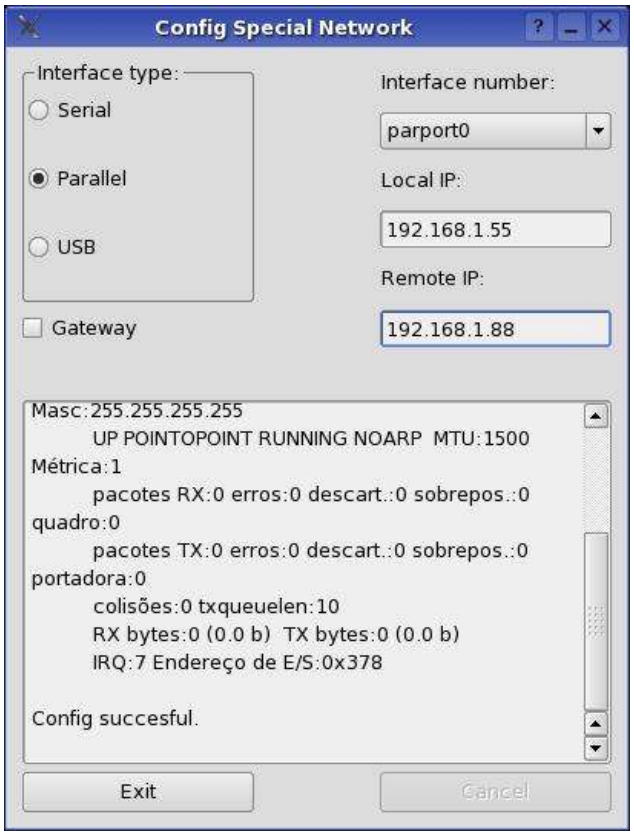

Figura 20 – Resultado da tentativa de configuração de uma interface de rede especial da aplicação SpecialNet.

O resultado da tentativa de ativar e configurar a interface é visualizado no grande quadro em branco na tela de configuração (Figura 20). Caso a tentativa de configuração tenha tido sucesso, a interface recém configurada estará visível na tela principal.

Internamente a aplicação executa os seguinte comandos para ativar e configurar uma interface do tipo serial:

modprobe slip

slattach -p slip -s 115200 /dev/ttyS*

ifconfig sl* IP_LOCAL pointopoint IP_REMOTO up

Para ativar e configurar uma interface de rede paralela os seguintes comandos são executados:

modprobe plip

ifconfig plip* IP_LOCAL pointopoint IP_REMOTO up

Para ativar e configurar uma interface de rede USB os seguintes comandos são executados:

modprobe usbnet

ifconfig usb* IP_LOCAL pointopoint IP_REMOTO up

Para qualquer interface, se o opção de *gateway* estiver marcada, o comando a seguir também é executado:

route add default gw IP_REMOTE

Nos comandos anterior * representa o número da interface que será configurada, IP_LOCAL é o endereço IP da interface e IP_REMOTO é o endereço IP da interface remota que será conectada.

4.3 Documentação do *software*

Foi utilizado a ferramenta Doxygen para gerar toda a documentação do *software*. O KDevelop disponibiliza uma interface fácil e amigável de configuração do Doxygen. Tais configurações permitem habilitar a criação de diversos tipos de diagramas, selecionar os tipos de informações que devem ser geradas e os formatos de arquivos de saída que se deseja obter.

Em todo o código-fonte foram colocados vários comandos do Doxygen que permitem a geração de uma documentação mais completa. Comandos do Doxygen são comentários iniciados por '!' antes dos itens que se deseja documentar. Estes itens podem ser classes, métodos, atributos, funções, enums, estruturas, etc. Existem ainda várias *tags* que podem ser colocadas dentro dos comandos para melhorar ainda mais a documentação. As *tags* são algumas palavras reservadas precedidas de '\'. A seguir alguns exemplos de comandos do

Doxygen:

```
/*!
* \class CMainWindow
* Classe da janela principal da aplicação.
*/
class CMainWindow
{
...
};
```

O comando anterior está sendo utilizado para descrever a classe CMainWindow.

Outro exemplo de comando:

```
//! Botão Novo da janela principal
QButton mButtonNew;
```

O comando anterior está sendo utilizado para descrever um atributo de uma classe.

Outros exemplos de uso dos comandos do Doxygen podem ser vistos no código-fonte do projeto disponível em http://paginas.terra.com.br/informatica/specialnet/.

Detalhes de utilização do KDevelop e do Doxygen podem ser encontrados em (GEHRMANN, 2004) e (HEESCH, 2005) respectivamente.

A Figura 21 e Figura 22 ilustram páginas em HTML da documentação do projeto SpecialNet geradas pelo Doxygen. Todo o restante da documentação do *software* que é composta de diagramas de hierarquia de classes, diagramas de colaboração de classes, descrições das classes, dos métodos e dos atributos, encontram-se disponível no *site* http://paginas.terra.com.br/informatica/specialnet/.

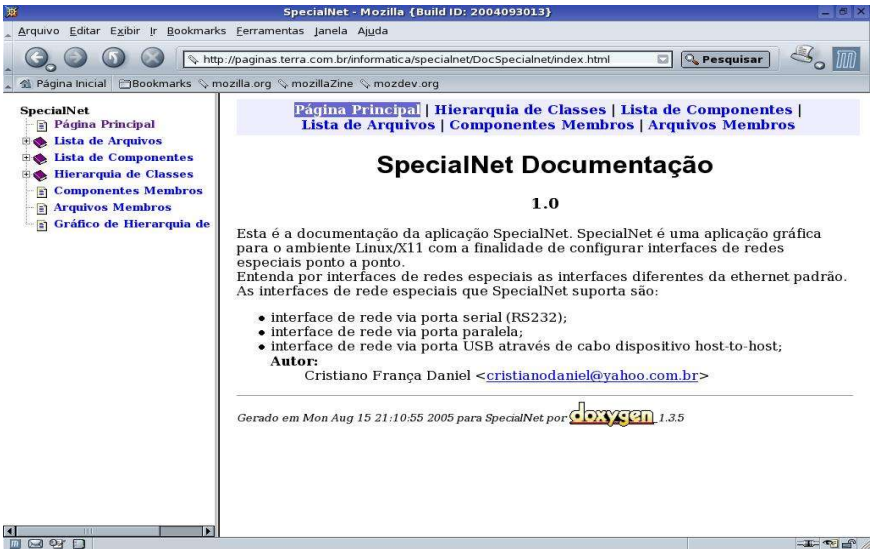

Figura 21 – Página principal da documentação do SpecialNet em HTML

Figura 22 – Página de referência da classe CConfigManager da documentação do SpecialNet em HTML

4.4 Estrutura do *software*

Toda a estrutura do *software* está descrita na documentação oficial do SpecialNet disponível no *site* http://paginas.terra.com.br/informatica/specialnet/.

Todo o código-fonte do projeto também pode ser obtido pelo *site* citado acima.

5 – CONCLUSÃO

O Linux é bastante poderoso quando se trata de métodos para se conectar duas ou mais máquinas em rede. Neste trabalho foi possível aprender a configurar três tipos de redes diferentes da rede padrão *ethernet.* Destas três foi possível ver em funcionamento duas delas, a rede via interface serial e a via interface paralela. A rede via interface USB não foi testada devido a dificuldade de se obter um cabo *host-to-host* USB.

Sobre as ferramentas utilizadas no projeto, o Qt mostrou ser uma excelente opção para o desenvolvimento de aplicações gráficas para Linux. Possui uma biblioteca de classes bastante variada e ferramentas auxiliares muito boas. Além do mais possui uma documentação vasta e completa, o que é imprescindível para o desenvolvedor de *software*.

O KDevelop mostrou ser uma opção boa de IDE agrupando todas as principais ferramentas de desenvolvimento de *software*.

O Doxygen surpreendeu pela qualidade da documentação que é capaz de criar, inclusive com criação de diagramas. Outro ponto positivo do Doxygen é a grande quantidade de formatos de documentos que ele pode gerar.

Todas as ferramentas se mostraram fáceis de serem utilizadas, o que comprova que o desenvolvimento para o ambiente Linux é viável e que a falta de ferramentas adequadas ou dificuldade de uso destas está deixando de ser verdade.

Com a contribuição da comunidade *open source* e de empresas que apoiam o movimento, tais ferramentas devem melhorar cada vez mais, contribuindo para que mais usuários, desenvolvedores e administradores de sistemas adotem o GNU Linux como a sua plataforma preferida.

5.1 Contribuições

Hoje em dia, praticamente todos computadores possuem uma interface de rede *ethernet.* Além do que o baixo custo de tais interfaces (cerca de trinta reais uma placa *ethernet)* viabiliza a instalação de mais de uma interface deste tipo por

máquina.

Porém, pode acontecer em uma emergência necessitar-se de uma interface de rede que não está disponível. Nestas horas então, pode-se usar uma interface de rede alternativa, como as interfaces via porta serial, paralela ou USB.

O fato é que quando alguém precisa configurar uma interface deste tipo, nem sempre sabe o procedimento ou tem paciência para pesquisar. É aí que entra os aplicativos que automatizam as configurações e principalmente os programas gráficos que disponibilizam uma interface mais amigável para o usuário, como é o caso do **SpecialNet**.

A contribuição deste trabalho é disponibilizar mais um *software* livre gráfico e amigável para os usuários do Linux. Sendo que tal *software,* pela sua característica de uso raro e específico, possibilite aos usuários poupar tempo pesquisando como ativar tais interfaces de redes e/ou decorando os comandos necessários para tal procedimento.

5.2 Trabalhos Futuros

O **SpecialNet** pode melhorar em alguns aspectos, por exemplo, sugerindo automaticamente endereços IPs para o usuário. Pode também melhorar a saída de erros com descrições mais claras dos erros.

O suporte a outras línguas pode ser implementado facilmente utilizando o suporte a internacionalização do Qt (Qt Linguist).

Outra melhoria interessante seria salvar as configurações ativas em disco e reativá-las automaticamente na inicialização do sistema.

Por fim, a criação do *site* do projeto, onde seria possível copiar o programa, acompanhar liberações de novas versões, novidades, detalhes do projeto, documentação e outras coisas relativas ao **SpecialNet**.

REFERÊNCIAS BIBLIOGRÁFICAS

SILVA, G. M. da. Guia Foca GNU/Linux. Disponível em <http://focalinux.cipsga.org.br/guia/avancado/index.htm>. Acesso em 15 ago. 2005.

DAWSON, T.; et al. Linux Networking-HOWTO. Disponível em <http://www.tldp.org/HOWTO/NET3-4-HOWTO.html>. Acesso em 15 ago. 2005.

CONTROZZI, A. LINUX PLIP MINI-HOWTO. Disponível em <http://www.ibiblio.org/pub/Linux/docs/HOWTO/PLIP>. Acesso em 15 ago. 2005.

LAMIRAL, G. PLIP Install HOWTO. Disponível em <http://www.ibiblio.org/pub/Linux/docs/HOWTO/PLIP-Install-HOWTO>. Acesso em 15 ago. 2005.

HARDS, B. The Linux USB sub-system. Disponível em <http://www.linux-usb.org/USB-guide/book1.html>. Acesso em 15 ago. 2005.

BROWNELL, D. The GNU/Linux "usbnet" Driver. Disponível em <http://www.linux-usb.org/usbnet/>. Acesso em 15 ago. 2005.

HEESCH, D. Doxygen Manual. Disponível em <http://www.stack.nl/~dimitri/doxygen/manual.html>. Acesso em 15 ago. 2005.

TROLLTECH. Qt Reference Documentation (Open Source Edition). Disponível em <http://doc.trolltech.com/3.3/index.html>. Acesso em 15 ago. 2005.

GEHRMANN, B; et al. KDevelop User Manual. Disponível em <http://docs.kde.org/development/en/kdevelop/kdevelop/>. Acesso em 15 ago. 2005.

ATAIDES, A. Desenvolvimento de Sistemas em Linux: Uma Análise das Ferramentas de Desenvolvimento para Linux. Disponível em <http://http://www.ginux.ufla.br/documentacao/monografias.html/>. Acesso em 15 ago. 2005.

ROMKEY, J. A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP. Internet Engineering Task Force (IETF), Junho 1988. (Request for Comments: 1055). URL:http://www.ietf.org/rfc/rfc1055.txt.