Aldemon Lage Bonifácio

ManC: Ajuda On-line para linguagem C

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação "*Lato Sensu*" Administração em Redes Linux, para a obtenção do título de especialista.

Orientador
Prof. DSc. Gustavo Guimarães Parma

Lavras
Minas Gerais - Brasil
2005

Aldemon Lage Bonifácio

ManC: Ajuda On-line para linguagem C

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação "*Lato Sensu*" Administração em Redes Linux, para a obtenção do título de especialista.

Aprovada em *17 de Abril de 2005*

Prof. Heitor Augustus Xavier Costa

Prof. Luciano Mendes dos Santos

Prof. DSc. Gustavo Guimarães Parma
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Meus sinceros agradecimentos ao professor Giacomin pela ajuda no início dos trabalhos, ao professor Parma pelo apoio e incentivo, aos colegas de curso que me apoiaram e principalmente à minha família.

Resumo

O objetivo desta monografia é apresentar um sistema de ajuda para desenvolvimento de programas em C. Para o melhor entendimento do programa, serão analisados conceitos teóricos, ferramentas utilizadas e o próprio sistema de ajuda.

Assim, o ManC (Manual de linguagem C), produto deste trabalho, é uma ferramenta cujo propósito é auxiliar no desenvolvimento de programas em linguagem C. Essa ferramenta pode ser usada como um guia rápido ou um manual de aprendizado e compreensão do uso de funções e estruturas de tal linguagem.

O sistema é composto de duas partes, quais sejam, a operacional e a apresentação. A primeira é responsável pelo armazenamento, pela pesquisa e pelo compartilhamento dos documentos. A segunda, como o próprio nome sugere, disponibiliza a interface com o usuário, na qual tem acesso às informações armazenadas no módulo operacional.

Pretende-se, inicialmente, elaborar documentos com informações básicas da linguagem C, objetivando-se dar auxílio aos programadores iniciantes nessa linguagem. A pretensão maior é organizar a documentação para que qualquer programador, iniciante ou não, possam sanar suas dúvidas e obter um aprendizado com a ferramenta.

Dedico este trabalho à minha esposa Sandra, aos meus filhos Eybraian e Aryanne, aos meus irmãos e especialmente aos meus pais.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O presente trabalho tem por objetivo a apresentação de uma ferramenta de ajuda, imprescindível ao programador de linguagem C, ManC.

Quando do desenvolvimento de certos programas, seja em Linguagem C ou não, eventuais dúvidas que surjam podem se apresentar como um empecilho muito grande, dificultando ou, até mesmo, impossibilitando a conclusão do trabalho.

Nesse sentido, o ManC tem papel fundamental, ao viabilizar ao programador que escreve em Linguagem C a superação de tais obstáculos, e, conseqüentemente, a finalização de seu programa de forma mais acertada e rápida.

Sendo assim, essa ferramenta pode ser usada como um guia rápido ou como um manual de aprendizado.

A importância e a necessidade de ditas ferramentas de ajuda podem ser confirmadas pela existência de vários outros programas, que igualmente têm por objetivo auxiliar o programador quando eventuais dúvidas surjam, facilitando-lhe o trabalho.

Assim sendo, vale destacar as seguintes ferramentas correlatas na Tabela 1.1.

Dentre as ferramentas de ajuda apresentadas, pode-se destacar a `perldoc` de (BURKE, 2004), que possibilita a pesquisa e a visualização da documentação das funções internas do `Perl` (THE PERL FOUNDATION, 2002-2005), além de recuperar a documentação embutida juntamente com o código fonte das bibliotecas.

O ManC (Manual de linguagem C), produto deste trabalho, pretende fornecer pesquisas e apresentar informações a respeito das bibliotecas e funções da linguagem C.

Visando facilitar a compreensão do funcionamento do ManC, a presente monografia foi dividida em 4 capítulos.

Ferramenta	Interface	Descrição
perldoc	texto	ferramenta que disponibiliza a documentação do perl e dos módulos instalados. `http://search.cpan.org/~sburke/`
man e info	texto	ferramentas que fornecem manuais e referências.
CPAN	html (web)	(CPAN: *Comprehensive Perl Archive Network*) consiste em um *site* de documentação sobre `Perl`. `http://search.cpan.org/`
devhelp	gráfica	é um navegador de referências a API (*Application Program Interface*) para GNOME 2. `http://www.imendio.com/projects/devhelp/`

Tabela 1.1: Ferramentas de ajuda correlatas

A metodologia e as ferramentas utilizadas no projeto serão apresentadas no Capítulo 2, ocasião em que serão destacadas, por exemplo, GCC, Anjuta, SQLite e GNU Libtool.

No Capítulo 3, por sua vez, serão abordadas a proposta da ferramenta para auxílio na busca de informações sobre a linguagem em questão, além das partes da ferramenta resultante.

Por fim, as conclusões sobre este trabalho serão destacadas no Capítulo 4.

Para o desenvolvimento de tal ferramenta de ajuda, foi de fundamental importância a leitura das seguintes obras: (GIACOMIN, 2002), (RESENDE, 2002), (SCHILDT, 1992), (KERNIGHAN; PIKE, 2000).

1.1 Programação em Linguagem C

Como se sabe, todo programador necessita de documentação sobre a linguagem com a qual está trabalhando.

Por essa razão, existem vários tipos de guias rápidos contendo a descrição e a forma de uso de comandos e as funções para auxílio na programação. Uma documentação em tempo real, disponibilizando um acesso rápido e simples, tornará o trabalho muito mais fácil.

A necessidade da documentação diminui à medida em que a experiência do programador aumenta. Assim, enquanto o programador iniciante necessita da documentação para conseguir escrever seus primeiros programas, um intermediário consegue guardar em sua memória a sintaxe, alguns comandos e algumas funções. O programador experiente, uma vez que conhece grande parte das regras, é capaz de escrever programas inteiros com poucas consultas às tais documentações.

Ainda sim, o leitor perceberá que mesmo o programador experiente terá domínio apenas sobre a sintaxe da linguagem, podendo não conhecer todos os recursos que ela oferece.

Para facilitar o entendimento, pode-se fazer uma analogia entre a documentação e o dicionário: a documentação de uma linguagem para um programador tem a mesma importância que um dicionário para um escritor. Ambas são fontes úteis e indispensáveis na realização do trabalho.

1.2 O Projeto

O projeto, em fase inicial, consistia na criação de uma ferramenta simples, para utilização no *shell* ou na *WEB* com páginas HTML. Com o surgimento de novas idéias, o projeto evoluiu, tornando-se o que será apresentado nos próximos parágrafos.

A primeira novidade foi a divisão da ferramenta em módulos, com a finalidade de permitir que várias pessoas trabalhassem no projeto ao mesmo tempo, de forma independente.

Assim, o projeto foi dividido nos seguintes módulos:

Operacional esse módulo é o repositório dos dados da ferramenta. É responsável pelo armazenamento e pesquisa das informações e pela sua disponibilização para os módulos de apresentação;

Apresentação os módulos de apresentação são responsáveis pela interação com o usuário, disponibilizando uma interface para que possa, assim, realizar suas pesquisas e visualizar a documentação. Os módulos de apresentação podem ser:

- **textual** utilizado no *shell*, na interface textual do sistema operacional;
- **gráfica** usado em sistemas gráficos, com utilização de janelas, botões, imagens, mouse, etc;
- **HTML** a apresentação em páginas HTML disponibiliza as informações para serem acessadas utilizando-se um navegador de acesso à internet;
- ***plugin*** consiste na criação de uma interface em algum sistema ou ferramenta existente, como a criação de uma ferramenta de ajuda dentro da ferramenta Anjuta (comentada na Seção 2.4);
- **outras apresentações** da maneira que o projeto foi organizado, podem-se criar outras formas de apresentação, sem maiores problemas, como por

exemplo, a criação de uma biblioteca em `Perl` que utilize o repositório.

Na Figura 1.1 tem-se uma visualização organizacional do projeto.

Esta monografia tratará apenas dos módulos operacional e apresentação textual, ficando os módulos de apresentação gráfico, WEB e plugin como proposta de continuidade.

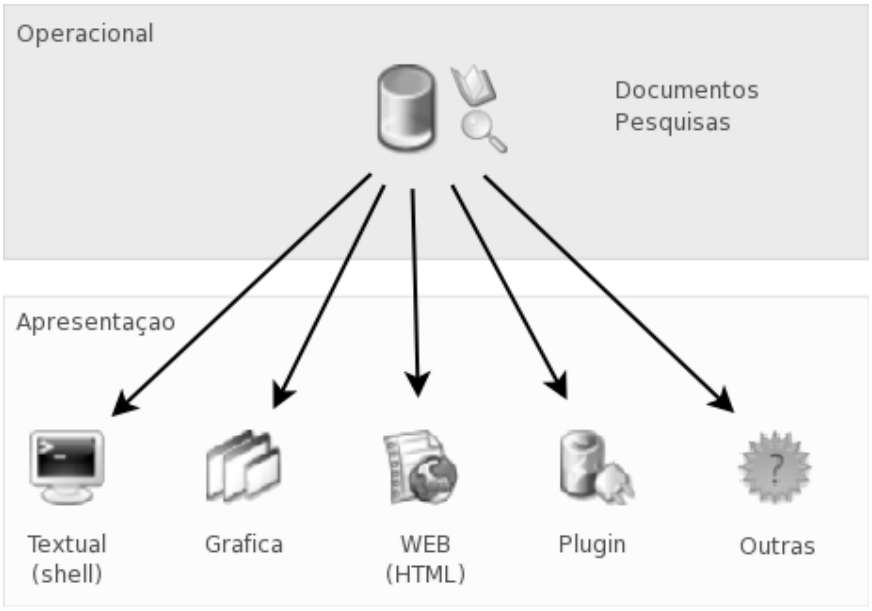

Figura 1.1: Projeto ManC

Capítulo 2

Ferramentas Utilizadas

As ferramentas utilizadas no projeto foram escolhidas tendo em vista alguns critérios, a saber: confiabilidade, qualidade, código aberto e gratuito, além de serem multiplataforma.

O ambiente de desenvolvimento, as ferramentas utilizadas, a linguagem usada no desenvolvimento e o sistema operacional base são comentados nas próximas seções.

2.1 Sistema Operacional

O sistema operacional escolhido para base de desenvolvimento foi o Linux.

Como descrito em (LINUX ONLINE, INC., 1994-2004), a funcionalidade, a adaptabilidade e a robustez do Linux têm feito dele a principal alternativa frente a sistemas operacionais proprietários, como UNIX e Microsoft Windows (`http://www.microsoft.com/windows`). Contribui para a sua crescente utilização o fato de ser livremente distribuído.

IBM, Hewlett-Packard e outras gigantes da computação mundial têm adotado o Linux e vêm dando suporte ao seu desenvolvimento. Mais de uma década depois de sua primeira liberação, o Linux vem sendo adotado em todo o mundo como uma plataforma de servidor primária.

Seu uso como um sistema operacional doméstico e comercial está também em ascensão. O sistema operacional pode ser igualmente incorporado diretamente nos *microchips*, em processos chamados '*embedding*' (embutido), sendo constantemente usado em ferramentas e dispositivos.

Inicialmente, durante boa parte da década de 1990, estudos tecnológicos, pela maior parte inconscientes do potencial do Linux, apresentavam-no como um pro-

jeto de hobby de computador, inadequado, portanto, para as necessidades da computação pública geral.

Com os esforços de desenvolvedores de sistemas gerenciadores de ambientes, tais como KDE e GNOME, de conjunto de ferramentas para escritório, do projeto OpenOffice.org, e de navegadores WEB, do projeto Mozilla, entre vários outros, há agora uma grande escala de aplicações que funcionam em Linux e podem ser usadas mesmo por aqueles que agora iniciam seus estudos na área da computação.

Constatam-se as capacidades do Linux obtendo uma versão *live CD* (distribuição Linux que funciona diretamente do CD-ROM, sem necessidade de instalação) tais como Kurumin Linux de (MORIMOTO, 1999-2004) ou Knoppix de (KNOPPER.NET, 2005). Ambos vêm com todas as ferramentas necessárias para realizar tarefas cotidianas no computador e não necessitam de instalação no disco rígido. Eles funcionam em computadores capazes de inicializar o sistema a partir do CD-ROM.

Ao escolher continuar usando Linux, encontram-se disponíveis uma variedade de versões ou 'distribuições' Linux que são fáceis de instalar, configurar e usar.

Se o leitor estiver interessado em aprender sobre Linux, necessitará de ajuda em alguns aspectos de uso do sistema operacional. Caso seja entusiástico e queira ajudar a estimular a adoção, pode começar entrando em um grupo de usuários Linux na sua área.

2.2 Linguagem C/C++

A parte repositório e a interface shell simples foram escritas utilizando-se `C/C++`.

Como descrito em (THE C++ RESOURCES NETWORK, 2000), C++ é uma linguagem de programação que literalmente significa 'C incrementado', sendo ele derivado da linguagem C.

Como é sabido, atualmente, os computadores podem executar muitas tarefas diferentes, desde operações matemáticas simples às representações gráficas sofisticadas. Todavia, tais tarefas não são feitas pelo próprio computador, eis que executadas depois de uma série de instruções pré-definidas, chamadas de programa de computador.

Isso ocorre porque o computador não tem criatividade suficiente para fazer as tarefas por conta própria, sendo-lhe possível seguir somente as instruções dos programas. Desta maneira, os programadores são encarregados de gerar programas de modo que os computadores possam executar novas tarefas.

Ao escolher uma linguagem de programação para fazer um projeto, algumas considerações hão que ser observadas. Inicialmente, o conhecimento 'em nível da linguagem de programação'. O nível determina quão próximo à máquina a

linguagem de programação está. Instruções de linguagens de ‘baixo nível’ são escritas pensando-se diretamente na execução interna da máquina, enquanto em linguagens de ‘alto nível’ escreve-se um código mais abstrato ou mais conceitual.

Geralmente, o código de alto nível é mais portátil, o que significa que se pode trabalhar em máquinas diferentes com pouquíssimas modificações, visto que uma linguagem de baixo nível é limitada pelas peculiaridades da máquina na qual se escreveu. Apesar disso, o código de baixo nível é geralmente mais rápido, porque é escrito empregando-se vantagens de uma máquina concreta.

Ao escolher o tipo de programação, deve-se analisar, também, o programa que se deseja produzir. Por exemplo, ao programar um administrador de dispositivo de *hardware* para um sistema operacional, usa-se a programação de baixo nível. Entretanto, ao programar aplicações grandes, geralmente é utilizado um nível mais elevado, ou uma combinação das partes críticas escritas em baixo nível e outras em alto nível.

É sabido que algumas linguagens são claramente de baixo nível, como `Assembler`, cujas instruções variam de acordo com o processador na qual o código é feito, e outras de alto nível, como o JAVA, que é totalmente independente da plataforma. A linguagem C++ está no meio termo, uma vez que ela pode interagir diretamente com a máquina quase sem nenhuma limitação. Além disso, dá suporte a bibliotecas específicas combinadas, trabalhando como uma das mais poderosas linguagens de alto nível.

Dentre as várias características da linguagem C++, pode-se destacar:

Programação orientada a objetos A possibilidade de orientar a programação aos objetos permite que o programador projete aplicações de um ponto de vista mais como uma comunicação entre os objetos que uma seqüência estruturada do código. Além do mais, permite a reutilização do código de uma maneira mais lógica e mais produtiva;

Portabilidade Pode-se praticamente compilar o mesmo código C++ em quase qualquer tipo de computador e de sistema operacional sem realizar mudanças radicais. C++ é uma linguagem de programação das mais usadas, além de ser portada para diversas plataformas;

Reduzida O código escrito em C++ é muito curto em comparação com outras linguagens, pois o uso de caracteres especiais é preferido antes das palavras chaves, conservando o esforço;

Programação modular O corpo de uma aplicação em C++ pode ser composto de diversos arquivos de código fonte que são compilados separadamente e ligados posteriormente. Assim, economiza-se tempo, uma vez que não é

necessário recompilar a aplicação completa ao fazer uma única mudança, mas somente o arquivo que a contém. Além disso, esta característica permite ligar o código C++ com código produzido em outras linguagens como `Assembler` ou `C`;

Compatibilidade com C Qualquer código escrito em C pode facilmente ser incluído em um programa C++ sem mudanças drásticas;

Velocidade O código resultante de uma compilação C++ é muito eficiente, devido certamente a sua dualidade como linguagem de alto nível e baixo nível e ao tamanho reduzido da própria linguagem, além do seu excelente compilador.

2.3 GCC - GNU Compiler Collection

A ferramenta utilizada para compilação do fonte é o GCC (FREE SOFTWARE FOUNDATION, 2005a, *GNU Compiler Collection*). GCC é uma coleção de compiladores da GNU (FREE SOFTWARE FOUNDATION, 2005b), a qual contém, atualmente, ambientes de desenvolvimento para `C`, `C++`, `Objetive-C`, `Fortran`, `Java` e `Ada`, bem como bibliotecas para essas linguagens.

Na Tabela 2.1 estão alguns exemplos de compiladores populares.

2.4 Anjuta

O ambiente de desenvolvimento utilizado nesse trabalho foi o Anjuta.

Como descrito pelo autor (KUMAR; PIPER, 2001-2002), o Anjuta é um ambiente de desenvolvimento integrado e versátil para C e C++ em Linux. Foi escrito para GTK-GNOME (THE GTK+ TEAM, 2005), tendo diversas características e diversas facilidades para programação avançada. Isso inclui gerência de projeto, criação de aplicação passo a passo, *debugger* integrado e um poderoso editor de código com navegação e destaque de sintaxe.

Pode-se dizer que ele é uma interface gráfica que reúne o poder e flexibilidade das linhas de comando com a facilidade de uso da interface gráfica GNOME da (THE GNOME FOUNDATION, 2003).

Entre as características dessa ferramenta, podem-se destacar a auto-complementação de códigos, endentação automática, execução interativa, assistentes para a criação de aplicações GTK, GNOME ou modo texto, gerenciamento de favoritos, janelas soltas ou anexadas ao ambiente e suporte para outras linguagens como Pascal e Java.

Um exemplo do ambiente Anjuta pode ser visto na Figura 2.1.

Compilador	Plataforma	Descrição
Gratuitos		
Borland C++ 5 Free Compiler	Windows	Essa é uma liberação gratuita do compilador Borland C++ Builder de 32-bit. Mas é limitado apenas às ferramentas de linha de comando.
GNU GCC	Windows, DOS, UNIX, Linux, entre outros	GCC é um compilador multi-linguagem originalmente desenvolvido para sistemas compatíveis com UNIX que podiam compilar código C++.
Comerciais		
Microsoft Visual C++ 6.0	Windows	Compilador C++ de grande aceitação pública. A maioria das aplicações comerciais para Windows são desenvolvidos utilizando esse compilador.
Borland C++ Builder 5	Windows	Esse é conhecido como o melhor compilador ANSI-C++ atualmente disponível no mercado.
Metrowerks Codewarrior Pro	Windows, Mac, Palm, Java, Linux	Ele está disponível para várias plataformas e processadores usando o mesmo projeto em formato e apresentação em uma interface similar.
Sybase PowerBuilder 7.0	Windows	Ambiente de desenvolvimento empresarial com suporte a utilitários basedos na WEB. Completo suporte para desenvolver aplicações cliente/servidor rápidas e escaláveis para web.
IBM Visual Age C++ Pro 4.0	OS/2, AIX, Windows NT	O compilador C++ da IBM é um completo ambiente de desenvolvimento para programação de aplicações multi-plataforma e orientado a objetos com suporte ao padrão ANSI-C++.

Tabela 2.1: Compiladores populares

2.5 SQLite

O repositório de dados armazena as informações em um banco de dados gerenciado pelo SQLite.

Como descrito em (HIPP, 2005), SQLite é uma pequena biblioteca C que implementa um gerenciador de base de dados SQL. Ele se diferencia da maioria dos outros gerenciadores de base de dados por ter como objetivo a simplicidade. Assim, é ele:

- simples para administrar;
- simples para operar;
- simples para uso em programas;
- simples para manutenção e adaptações.

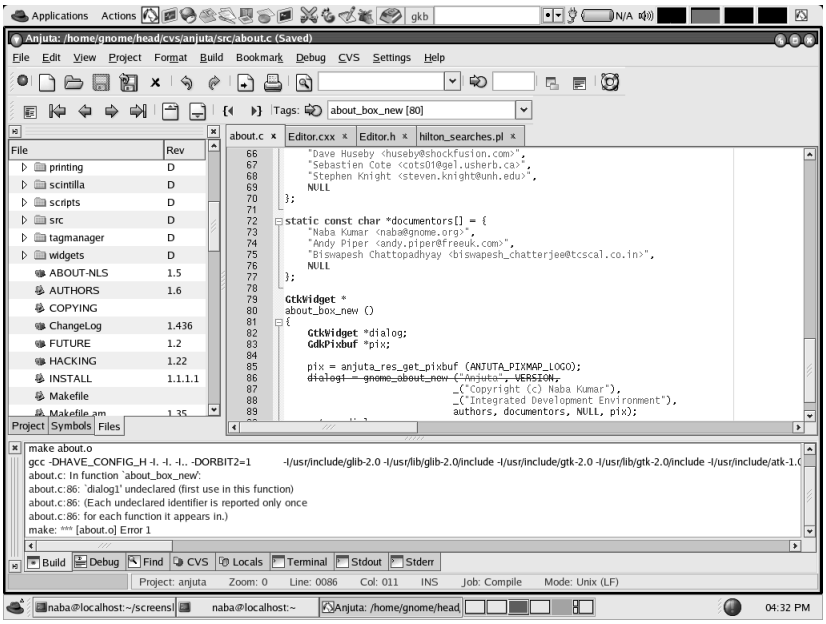

Figura 2.1: Exemplo do ambiemte Anjuta

Muitas pessoas apreciam o SQLite porque é pequeno e rápido. Embora não sabido pela maioria das pessoas, aquelas qualidades são apenas acidentes felizes. Os usuários também encontram no SQLite um sistema muito confiável. A confiabilidade é uma conseqüência da simplicidade. Afinal, com menos complicação, há menos chances de errar. Assim, SQLite é pequeno, rápido e confiável, mas o seu principal objetivo é a simplicidade. O SQLite esforça-se, principalmente, para ser simples. A simplicidade em um gerenciador de base de dados pode ser uma força ou uma fraqueza, dependendo do que se está tentando fazer.

A fim de conseguir a simplicidade, SQLite teve que sacrificar outras características que são úteis para alguns usuários, como a alto concorrência, controle de acesso refinado, um conjunto rico de funções internas, procedimentos armazenados, características de língua esotérica do SQL, XML e/ou extensões Java e escalabilidade de tera-byte ou penta-byte.

Se esses tipos de características forem necessários e a complexidade adicionada não importar, então o SQLite provavelmente não será a base de dados ideal. SQLite não pretende ser um gerenciador de base de dados empresarial. Ele não foi projetado para competir com Oracle (ORACLE CORPORATION, 2005) ou PostgreSQL (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 1996-2005).

A regra básica para identificar quando é apropriado usar SQLite é a seguinte: usar o SQLite nas situações em que a simplicidade da administração, da execução e da manutenção são mais importantes do que as características complexas incontáveis que os gerenciadores de base de dados empresariais fornecem.

As características do SQLite foram essenciais para a sua escolha. Ao invés de criar um sistema para armazenamento e pesquisa dos dados, chegou-se a conclusão que essa ferramenta supriria todas as expectativas, além de poupar retrabalho.

A tabela Tabela 2.2, apresenta situações em que o SQLite trabalha bem.

Tipo de Aplicação	**Informações**
Websites	SQLite trabalhará notavelmente como um banco de dados para sites com tráfego pequeno a médio.
Aplicações e dispositivos embutidos	Por conta de uma base de dados SQLite requer quase nenhuma administração, torna-se uma boa escolha para dispositivos ou serviços que devem trabalhar desacompanhados e sem sustentação humana.
Formato de arquivo de Aplicação	SQLite foi usado com grande sucesso como o formato de arquivo no disco para aplicações *desktop* tais como ferramentas financeiras de análise, pacotes CAD e assim por diante.
Realocação de arquivos em discos de acesso direto	Muitos programas usam o `fopen()`, o `fread()` e o `fwrite()` para criar e controlar arquivo de dados em formatos caseiros. SQLite trabalha bem como um realocador para estes arquivos de dados de acesso direto.
Base de dados internas ou temporárias	Para programas que têm muitos dados que devem ser separados e organizados em diversas maneiras, é quase sempre mais fácil e mais rápido carregar os dados em uma base de dados SQLite em memória.
ferramenta de linha de comando para análise de conjunto de dados	Usuários SQL experientes podem empregar o programa SQLite de comando de linha para analisar séries de dados variadas
Substituto para uma base de dados empresarial durante apresentações e testes	Caso o desenvolvedor esteja escrevendo uma aplicação cliente para um gerenciador de base de dados empresarial, faz sentido usar uma base de dados genérica de fundo que permita conectar a muitos tipos diferentes de gerenciadores de base de dados SQL
Base de dados pedagógica	Por causa da simples configuração e uso, o SQLite torna-se um ótimo gerenciador de base de dados para utilização em aprendizado de SQL
Extensões de linguagem SQL experimental	O projeto simples, modular do SQLite torna-o uma boa plataforma para novos protótipos, características ou idéias de linguagem de base de dados experimental

Tabela 2.2: Situações onde o SQLite trabalha bem

Das características de destaque do SQLite, algumas são comentadas a seguir:

Pequeno Código fonte pequeno.

Rápido Eficiente no resultado das sentenças.

Confiável Confiável por consequência da simplicidade.

Simples Fácil de usar.

2.6 GNU Libtool

O GNU Libtool (FREE SOFTWARE FOUNDATION, 2005c) é utilizado na criação de uma interface de comunicação entre o repositório e as interfaces para o usuário.

GNU Libtool é um código genérico de suporte à biblioteca. Ele encapsula a complexibilidade de uso de bibliotecas compartilhadas atrás de uma interface consistente e portável.

No passado, se um desenvolvedor de pacote de código fonte quisesse usufruir das vantagens dos recursos de bibliotecas compartilhadas, ele necessitaria escrever um código com suporte adequado para cada plataforma em que seu pacote funcionasse.

Também teria que projetar uma interface de configuração de modo que o instalador do pacote pudesse escolher em qual ordem as bibliotecas seriam geradas.

O trabalho do desenvolvedor é simplificado pelo encapsulamento da dependência específica da plataforma e a interface do usuário em um simples código.

GNU Libtool é projetado de modo que a funcionalidade completa de cada tipo de plataforma esteja disponível através de uma interface genérica, a fim de ocultar do programador as dificuldades encontradas em cada uma das plataformas.

A interface consistente de GNU Libtool trás de volta a tranqüilidade, razão pela qual desenvolvedores não necessitam ler a documentação obscura a fim ter seu código fonte favorito transformado em bibliotecas compartilhadas. Basta executar o código de configuração e o GNU Libtool fará o trabalho difícil.

Desde 1995, vários desenvolvedores GNU reconheceram a importância de ter seus códigos fonte suportando biblioteca compartilhada. A motivação preliminar para tal mudança é o incentivo à modularidade e à reutilização do código em programas da GNU.

Tal demanda significa que o caminho das bibliotecas é construir códigos GNU genéricos. O problema é composto pela ausência de um procedimento padrão para criar bibliotecas compartilhadas em plataformas diferentes.

As seguintes especificações foram usadas para desenvolver e avaliar o GNU Libtool:

1. O sistema deve ser tão elegante como possível;

2. O sistema deve ser inteiramente integrado com os utilitários GNU *Autoconf* e *Automake*, de modo que seja fácil para mantenedores GNU usarem. Entretanto, o sistema não deve requerer estas ferramentas, de modo que possa ser usado por ferramentas não GNU;

3. A portabilidade para outras arquiteturas e ferramentas não GNU é desejável.

A inclusão do GNU Libtool no projeto aumenta a expectativa ao facilitar a disponibilização do ManC para diversas plataformas.

2.7 GetPot

Na parte de apresentação textual, é preciso reconhecer os parâmetros que o usuário informar na linha de comando. A ferramenta utilizada para tal processamento é o GetPot. De acordo com (SCHAEFER, 2002), GetPot permite interpretar parâmetros de linha de comando de maneira muito eficiente.

Além de interpretar parâmetros de linha de comando, o GetPot é capaz de ler arquivos de configurações. Esse recurso é utilizado tanto pela interface shell simples quanto pelo repositório para uso de arquivos de configurações.

Capítulo 3

A Ferramenta ManC

O ManC é uma ferramenta que permitirá ao programador pesquisar e conhecer as bibliotecas, as funções, os comandos e a forma de usá-los de maneira simples e rápida.

Assim, com opções de apresentação da documentação em ambientes diversos, o programador terá a liberdade e a comodidade de escolher o ambiente que mais lhe agradar.

Entre as características do projeto, destacam-se:

- pesquisa por bibliotecas;
- pesquisa por funções;
- facilidade e simplicidade de uso;
- apresentação em diversos ambientes;
- modularidades de programas.

De acordo com as características apresentadas, o ManC tem grandes chances de se tornar uma ferramenta muito útil no auxílio ao desenvolvimento de programas em linguagem C.

A opção pela criação de programas modulares baseia-se na possibilidade de produção de várias interfaces, aproveitando-se o mesmo programa de armazenamento e pesquisa. Desta maneira, tem-se um reaproveitamento de trabalho.

Na Seção 3.1, será apresentada a parte operacional do projeto, a qual é responsável pelo armazenamento da documentação. Na Seção 3.2, será mostrada uma das partes da apresentação, vale dizer, a textual, a qual interage com o usuário no *shell*.

3.1 Operacional - Repositório de dados

Essa é a parte operacional do projeto, responsável pelo armazenamento, pesquisa e disponibilização das informações.

Tem como objetivo ser único, para facilitar e centralizar a manipulação dos dados. Assim, quando ocorrer uma atualização das informações, estas estarão disponíveis em todos os módulos de apresentação.

O repositório foi escrito utilizando a linguagem C/C++, comentada na Seção 2.2, e compilado na forma de uma biblioteca dinâmica. O uso de tal biblioteca oferece uma liberdade de modificação sem que seja necessário alterar todos os programas que a utilizam. A biblioteca recebeu o nome `ManCRepository`.

As informações estão armazenadas em um banco de dados SQLite, abordado na Seção 2.5, e organizado como apresentado na Figura 3.1. O banco de dados é armazenado em um arquivo e acessado diretamente pelo `ManCRepository`.

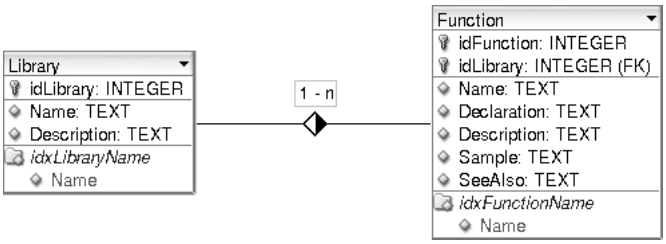

Figura 3.1: Estrutura da base de dados do ManC

Pode-se visualizar uma descrição de cada campo das tabelas da base de dados na Tabela 3.1.

As funções exportadas pela biblioteca são:

- **int Initialize()**

  Inicializa variáveis e informações necessárias para utilização da biblioteca ManCRepository.

  Retorna:

  0 (zero) para sucesso e diferente de zero se ocorrer algum erro.

Campo	Descrição
Tabela *Library* (biblioteca)	
idLibrary	Identificação interna da biblioteca.
Name	Nome da biblioteca.
Description	Uma breve descrição da biblioteca.
Tabela *Function* (função)	
idFunction	Identificação interna da função.
idLibrary	Identificação da biblioteca à qual a função pertence.
Name	Nome da função.
Declaration	Forma de utilização da função.
Description	Uma breve descrição da função.
Exemplo	Exemplo de uso da função.
SeeAlso	Sugestões para leitura da documentação de outras funções.

Tabela 3.1: Campos das tabelas da base de dados

- **int Terminate()**

  Libera memória e finaliza informações utilizadas pela biblioteca `ManCRepository`.

  Retorna:

  0 (zero) para sucesso e diferente de zero se ocorrer algum erro.

- **string getDatabaseName()**

  Fornece o nome da base de dados utilizada pela MancRepository.

  Retorna:

  nome da base de dados.

- **bool dbConnected()**

  Fornece a informação indicando se a ManCRepository está conectada na base de dados.

  Retorna:

  true indicando que está usando a biblioteca e *false* caso contrário.

- **repLibrary* getLibrary(string Library)**

  Busca as informações da biblioteca solicitada.

  Parâmetros:

  Library - nome da biblioteca.

Retorna:

repLibrary* - uma referência para um objeto `repLibrary`. Se nenhuma biblioteca for encontrada, retorna `NULL`.

- **repLibrary* getLibraryByID(int idLibrary)**

  Busca as informações da biblioteca solicitada.

  Parâmetros:

  idLibrary - identificação interna da biblioteca.

  Retorna:

  repLibrary* - uma referência para um objeto `repLibrary`. Se nenhuma biblioteca for encontrada, retorna *NULL*.

- **repFunction* getFunction(string Function, string Library)**

  Busca as informações da função solicitada.

  Parâmetros:

  Function - nome da função.

  Library - nome da biblioteca.

  Retorna:

  repFunction* - uma referência para um objeto `repFunction`. Se nenhuma função for encontrada, retorna *NULL*.

- **repFunction* getFunctionByID(int idFunction)**

  Busca as informações da função solicitada.

  Parâmetros:

  idFunction - nome da função.

  Retorna:

  repFunction* - uma referência para um objeto `repFunction`. Se nenhuma função for encontrada, retorna *NULL*.

- **vector<repLibrary*> getLibraries(string Criteria = “”)**

  Busca as bibliotecas armazenadas na ManCRepository.

  Parâmetros:

Criteria - inicial do nome para seleção das bibliotecas. Se nenhuma informação for passada, todas as bibliotecas serão retornadas.

Retorna:

vector<repLibrary*> - um vetor com nenhum ou vários objetos `repLibrary`.

- **vector<repFunction*> getFunctions(string Criteria = "", string Library = "", int idLibrary = 0)**

  Busca as funções armazenadas na ManCRepository.

  Parâmetros:

  Criteria - inicial do nome para seleção das funções. Se nenhuma informação for passada, todas as funções serão retornadas.

  Library - inicial da biblioteca para seleção das funções. Se nenhuma informação for passada, apenas os parâmetros *Criteria* e/ou *idLibray* serão analisados.

  idLibrary - identificação interna da biblioteca para seleção das funções. Se nenhuma identificação for passada, apenas os parâmetros *Criteria* e/ou *Libray* serão analisados.

  Retorna:

  vector<repFunction*> - um vetor com nenhum ou vários objetos `repFunction`.

Outras funções serão criadas e exportadas no futuro, para melhorar a interação com a biblioteca ManCRepository.

A única dependência para instalação da ManCRepository é a biblioteca SQLite (HIPP, 2005) versão 3.0.8 ou superior. O SQLite, com seus pacotes de cabeçalho, podem ser instalados a partir de pacotes de várias distribuições Linux ou a partir do código fonte disponível no site do projeto.

Para instalar a ManCRepository é simples, basta executar a sequência de comandos apresentadas na Figura 3.2.

```
%./configure
%make
%install
```

Figura 3.2: Instalação da `ManCRepository`

É necessário executar o último comando como usuário *root* do sistema operacional.

Os arquivos instalados pela ManCRepository estão listados na Tabela 3.2.

Arquivo	**Descrição**
`/usr/local/lib/`	
libmancrepository.la	Biblioteca estática.
libmancrepository.so.0.0.0	Biblioteca dinâmica.
libmancrepository.so.0	Ligação para a biblioteca dinâmica libmancrepository.so.0.0.0.
libmancrepository.so	Ligação para a biblioteca dinâmica libmancrepository.so.0.0.0.
`/usr/local/include/ManCRepository/`	
mancRepository.h	Arquivo de cabeçalho das funções exportadas pela ManCRepository.
converter.h	Arquivo de cabeçalho, dependência do arquivo mancRepository.h.
fsqlite3.h	Arquivo de cabeçalho, dependência do arquivo mancRepository.h.
repository.h	Arquivo de cabeçalho, dependência do arquivo mancRepository.h.
`/usr/local/share/ManCRepository/`	
index.manc	Arquivo de dados da ManCRepository.

Tabela 3.2: Arquivos instalados pela ManCRepository

Para que outras aplicações possam usar a ManCRepository, é necessário incluir o caminho da biblioteca (`/usr/local/lib/`) no arquivo `/etc/ld.so.conf`. Esse arquivo contém os caminhos para as aplicações procurarem as bibliotecas. Uma maneira de saber se o caminho está no arquivo é executando os comandos apresentados na Figura 3.3. Se o caminho não existir, pode-se inclui-lo utilizando-se os comandos da Figura 3.4.

```
grep '^/usr/local/lib$' /etc/ld.so.conf > /dev/null \
   && echo "caminho existe no arquivo"              \
   || ( echo -e "\n**"                              \
      && echo "** caminho NÃO existe no arquivo **" \
      && echo -e "**\n")
```

Figura 3.3: Confere se `/usr/local/lib` existe em `/etc/ld.so.conf`

```
grep -q '^/ausr/local/lib$' /etc/ld.so.conf      \
   || echo "/ausr/local/lib" >> /etc/ld.so.conf
```

Figura 3.4: Adiciona `/usr/local/lib` em `/etc/ld.so.conf` se não existir

A ManCRepository disponibiliza a classe `mancRepositoryH`, no arquivo de cabeçalho `manRepository.h`, para facilitar a utilização das funções exportadas. A interface da classe pode ser vista na Figura 3.5 e tem praticamente as mesmas funções exportadas pela biblioteca.

```
// classe-mancRepository.cc

class mancRepositoryH
{
  public:
    mancRepositoryH();
    ~mancRepositoryH();
    bool dbConnected();
    string getDatabaseName();
    repLibrary* getLibrary( string Library );
    repLibrary* getLibrary( int idLibrary );
    repFunction* getFunction( string Function,
        string Library = "" );
    repFunction* getFunction( int idFunction );
    vector<repLibrary*> getLibraries(
        string Criteria = "" );
    vector<repFunction*> getFunctions(
        string Criteria = "", string Library = "" );
    vector<repFunction*> getFunctions(
        string Criteria = "", int idLibrary = 0 );
};
```

Figura 3.5: Classe `mancRepositoryH`

Além da classe `mancRepositoryH`, outras duas classes são definidas no arquivo de cabeçalho `mancrepository.h`: `repLibrary` e `repFunction`. A primeira armazena as informações das bibliotecas, e a segunda, as informações das funções. A definição das classes pode ser vista na Figuras 3.6 e 3.7.

```
// classe-repLibrary.cc

class repLibrary {
  public:
    repLibrary(int idLibrary = 0, string Name = "",
        string Description = "");
    // -
    int idLibrary;
    string Name;
    string Description;
};
```

Figura 3.6: Classe `repLibrary`

```
// classe-repFunction.cc

class repFunction {
  public:
    repFunction(int idLibrary = 0, int idFunction = 0,
     string Library = "", string Name = "",
     string Declaration = "", string Description = "",
     string Sample = "", string SeeAlso = "");
    // -
    int idLibrary;
    int idFunction;
    string Library;
    string Name;
    string Declaration;
    string Description;
    string Sample;
    string SeeAlso;
};
```

Figura 3.7: Classe `repFunction`

Inicialmente, a base de dados da `ManCRepository` conterá a documentação apenas das bibliotecas `stdio.h` e `string.h`. A documentação foi traduzida da página de referência de C/C++ de (KOHL, 2005).

Exemplo de uso da classe mancRepositoryH

O ManCRepository leva em seus arquivos de instalação um arquivo de cabeçalho mancRepository.h, que contém a classe mancRepositoryH, para utilização em aplicações que queiram trabalhar com essa biblioteca.

A classe `mancRepositoryH` utiliza a GNU Libtool, comentada na Seção 2.6, para uso da biblioteca ManCRepository. A ferramenta GNU Libtool foi utilizada para facilitar o reconhecimento das funções de biblioteca. Além de exportar as funções, ela facilita a codificação em outros sistemas operacionais.

O objetivo dessa classe é facilitar a utilização da ManCRepository para aqueles que queiram trabalhar com a biblioteca. Com o uso da referida classe, torna-se transparente a inclusão da biblioteca em outras aplicações, uma vez que, o trabalho de ligação com as funções exportadas por essa são feitas pela própria classe.

Na Figura 3.8 tem-se um exemplo de utilização da classe. Para compilar e criar um arquivo executável desse referido código, utilize os comandos apresentados na Figura 3.9.

3.2 Apresentação - Textual (*Shell*)

A parte de apresentação textual (*shell*), nomeada `ManCShell`, interage com o usuário disponibilizando as informações em formato texto no *shell* do sistema operacional. Ela foi escrita utilizando a linguagem C/C++, comentada na Seção 2.2.

As dependências para instalação da ManCShell são a biblioteca GNU Libtool (ltdl.h), comentada na Seção 2.6 e os arquivos de cabeçalhos da ManCRepository, apresentada na Seção 3.1.

Para instalar a ManCShell é simples, basta executar a seqüência de comandos mostrados na Figura 3.10, da mesma maneira como foi feito com a ManCRepository.

É preciso executar o último comando como usuário *root* do sistema operacional.

O usuário tem várias escolhas de pesquisas utilizando opções e parâmetros de linha de comando, estando disponibilizada, igualmente, a visualização da documentação das funções e bibliotecas.

Se no uso dessa ferramenta o usuário não passar parâmetro algum, um texto explicando o modo de uso será apresentado. Caso o usuário queira, é possível visualizar mais informações com a opção `-a` ou `-ajuda`. Na Figura 3.11 pode-se ver a documentação de uso do ManCShell.

```
// exemplo.cc

#include <iostream>
#include <vector>
#include <string>
#include "mancRepository.h"

int main() {

  // -
  mancRepositoryH mancRepository;

  // Recupera a documentação das funções cadastradas.
  vector<repFunction*> Functions =
              mancRepository.getFunctions( "", "" );

  // -
  std::cout << "FUNÇÂO␣(BIBLIOTECA):" << std::endl;
  std::cout << "----" << std::endl;
  for ( int i = 0; i < Functions.size();  i++ ) {
    std::cout << "␣" << Functions.at(i)->Name <<
                 "␣-␣" << Functions.at(i)->Library <<
           std::endl;
  }
  std::cout << std::endl;

  // -
  return(0);

}
```

Figura 3.8: Exemplo de uso da classe `mancRepositoryH`

```
g++ -I/usr/local/include/ManCRepository -c -o \
   exemplo.o exemplo.cc
libtool --tag=CXX --mode=link g++ -lltdl -o   \
   exemplo exemplo.o
```

Figura 3.9: Compilação e geração do arquivo executável

```
%./configure
%make
%install
```

Figura 3.10: Instalação da `ManCShell`

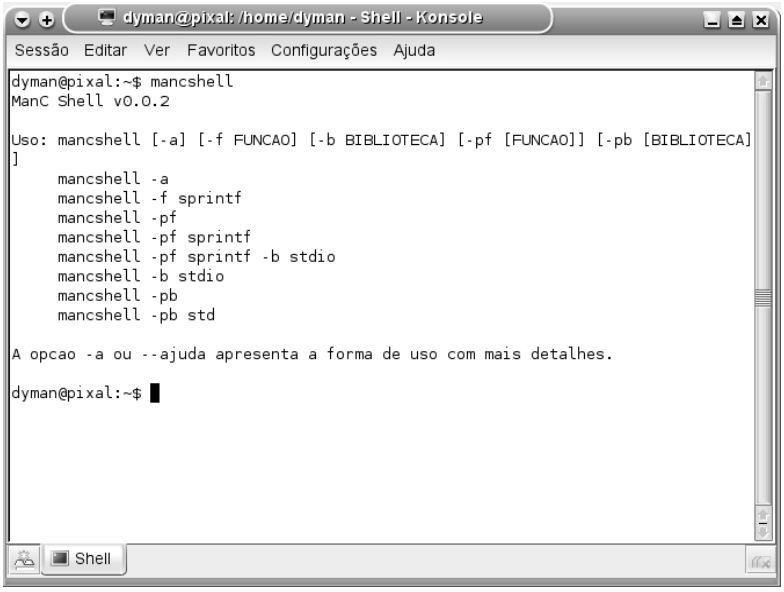

Figura 3.11: Documentação de uso do ManCShell

As chaves (`[` e `]`) são indicadoras de opção e/ou parâmetro opcional. Dessa forma, o ManCShell pode ser acionado com nenhuma opção, apenas a opção `-pb` ou, ainda, com a opção `-pb` e o parâmetro `string`.

Os exemplos de comandos a seguir utilizam, ou não, as opções e parâmetros de várias formas:

- `mancshell`

  Como comentado anteriormente, o ManCShell acionado sem opções apresenta sua forma de uso;

- `mancshell -a`

  A opção `-a` (**a**juda) faz com que ManCShell detalhe as opções de uso;

- `mancshell -f strcat`

  Por sua vez, a opção `-f` (**f**unção) com o parâmetro `strcat` apresenta a documentação da função, como pode ser visto na Figura 3.12;

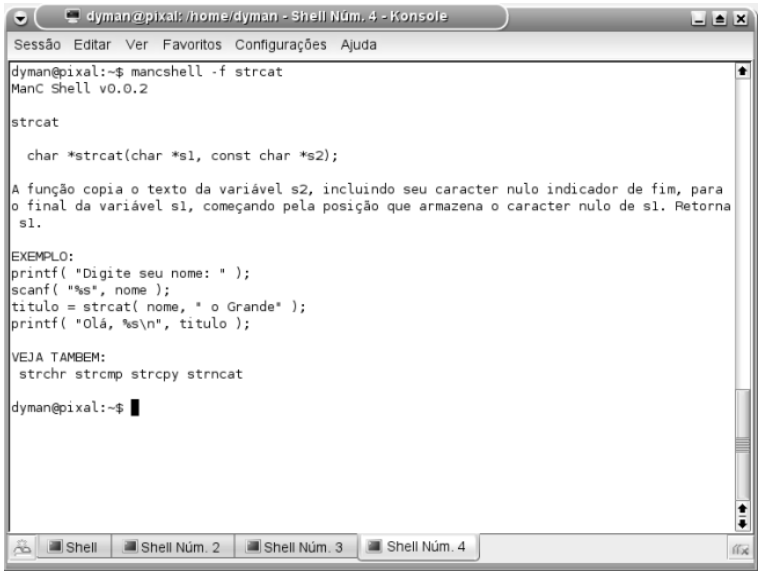

Figura 3.12: Documentação da função `strcat`

- `mancshell -pf`

  Como não teve parâmetro adicional para a opção `-pf` (**p**esquisar **f**unção), todas as funções cadastradas no repositório serão listadas;

- `mancshell -pf str`

  Com o parâmetro `str` para a opção `-pf`, serão mostradas todas as funções com nome iniciado por `str`;

- `mancshell -b string`

  A opção `-b` (**b**iblioteca) com o parâmetro `string` apresenta a documentação da biblioteca e lista as funções dessa biblioteca. Como apresentado na Figura 3.13;

- `mancshell -pb`

  Ao utilizar a opção `-pb` (**p**esquisar **b**iblioteca) sem parâmetro, a lista de todas as bibliotecas armazenadas no repositório será apresentada;

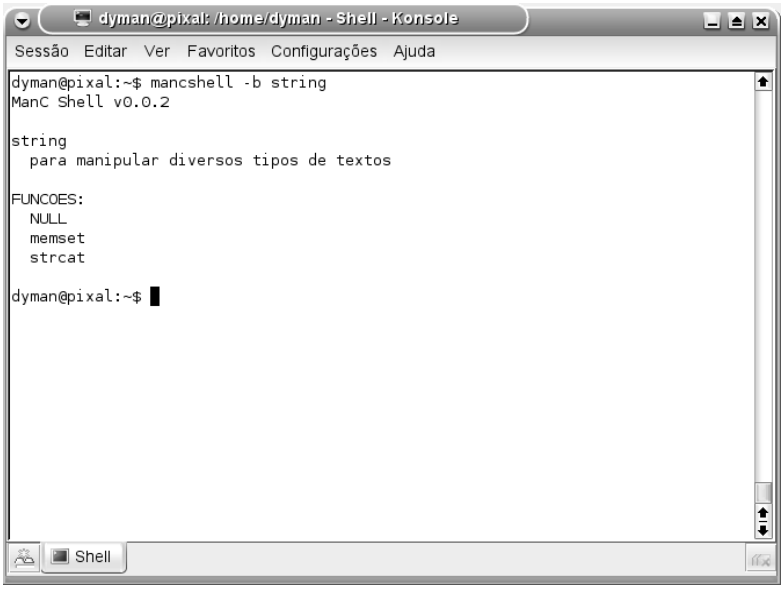

Figura 3.13: Documentação da biblioteca `string`

- `mancshell -pb st`

  Com o parâmetro `st` para a opção `-pb`, todas as bibliotecas com nome iniciado por `st` serão listadas.

A `ManCShell` interage com o usuário recebendo as solicitações e recupera as informações comunicando-se com a `ManCRepository`, comentada na Seção 3.1, que armazena as documentações das bibliotecas e funções.

Capítulo 4

Conclusões e Propostas de Continuidade

Nas seções a seguir serão destacadas as conclusões deste trabalho e apresentadas algumas propostas de continuidade.

4.1 Conclusões

Para o melhor desenvolvimento do trabalho, é imprescindível ao programador ter em mãos uma ferramenta de ajuda que possa, eventualmente, auxiliar-lhe a superar as dúvidas que porventura surjam.

Visando justamente facilitar o desenvolvimento de programas, a ferramenta de Ajuda Online para C, o ManC, foi criada. Dita ferramenta foi desenvolvida para sua utilização em linguagem C. Todavia, ela pode ser modificada para uso em qualquer outra linguagem de programação, desde que alterados os documentos da base de dados.

Além disso, muito embora essa ferramenta não possua ainda suporte à várias bases de dados com documentos de outras linguagens de programação, aquelas podem ser criadas, se necessário, sem grandes dificuldades. Para isso, basta algumas modificações na ferramenta atual.

Em um futuro próximo, pretende-se fazer com que a ManC rode em outros sistemas operacionais, como, por exemplo, o Windows.

O projeto está registrado e será hospedado no servidor do `CodigoLivre` e no `SourceForge`. Os endereços de acesso são `http://manc.codigolivre.org.br/` e `http://manc.sourceforge.net/`.

O desenvolvimento do projeto apresentou-se como um desafio, ainda que em um primeiro momento, vez que foi o primeiro trabalho desenvolvido em `C/C++`

e, também, Linux, por este programador. Posteriormente, todavia, a medida que os empecilhos foram superados, esse desafio tornou-se cada vez mais prazeroso e gratificante de ser realizado.

4.2 Propostas de continuidade

Outros módulos de interface podem ser desenvolvidos para aumentar a funcionalidade deste trabalho. Da mesma forma, a adição de funcionalidade no módulo de repositório de dados também contribui para o melhoramento do projeto.

Inicialmente, é necessário aumentar a documentação contida no banco de dados dessa ferramenta, pois, até o momento, têm-se apenas as funções das bibliotecas `stdio` e `string`.

Outra sugestão é a criação de vários bancos de dados, um para cada idioma que tenha documentação disponível. Recuperando, assim, as configurações da máquina e selecionando o banco de dados de acordo com idioma.

A sugestão mais audaciosa seria a criação de documentações de outras linguagem de programação, uma vez que as informações contidas nas documentações da linguagem C têm a mesma estrutura que outras linguagens, tais como, `Pascal`, `Basic`, `Bash`, entre outras.

Dentre as sugestões para módulos de apresentação tem-se:

Gráfica A interface gráfica teria uma estrutura semelhante às ferramentas de ajuda como (IMENDIO, 2004) e Ajuda do Windows;

WEB Essa interface faria uso de um sistema CGI (W3C, 1999, *Common Gateway Interface*) para possibilitar buscas e apresentação da documentação;

Plugin O objetivo seria integrar essa ferramenta de ajuda a uma ferramenta de desenvolvimento. As opções são desde a associação de uma tecla de atalho para fazer pesquisa na interface gráfica, até a criação de uma interface própria para a ferramenta de desenvolvimento;

Perl Desenvolvimento de uma biblioteca Perl que faça uso do repositório de dados para facilitar a criação de ferramentas utilizando essa linguagem de programação.

Com a modularização do projeto, pode-se criar várias outras interfaces sem maiores problemas, dependerá exclusivamente da criatividade de cada um.

Referências Bibliográficas

BURKE, S. M. *Pod-Perldoc*. [S.l.], 2004. Acesso em março de 2005. Disponível em: <http://search.cpan.org/~sburke/>.

FREE SOFTWARE FOUNDATION. *GCC*. fevereiro 2005. Acesso em março de 2005. Disponível em: <http://gcc.gnu.org/>.

FREE SOFTWARE FOUNDATION. *GNU*. fevereiro 2005. Acesso em março de 2005. Disponível em: <http://www.gnu.org/>.

FREE SOFTWARE FOUNDATION. *GNU Libtool - The GNU Portable Library Tool*. fevereiro 2005. Acesso em março de 2005. Disponível em: <http://www.gnu.org/software/libtool/>.

GIACOMIN, J. C. *Introdução à Linguagem C*. [S.l.]: UFLA/FAEPE, 2002. 105 p.

HIPP, D. R. *SQLite*. [S.l.], 2005. Acesso em fevereiro de 2005. Disponível em: <http://www.sqlite.org/>.

IMENDIO. *Devhelp*. 2004. Acesso em março de 2005. Disponível em: <http://www.imendio.com/projects/devhelp/>.

KERNIGHAN, B. W.; PIKE, R. *A Prática da Programação*. [S.l.]: Editora Campus, 2000. 304 p. Tradução de Kátia A. Roque. ISBN 85-352-0546-2.

KNOPPER.NET. *Knoppix*. 2005. Acesso em março de 2005. Disponível em: <http://www.knoppix.org/>.

KOHL, N. *C/C++ Reference*. 2005. Acesso em março de 2005. Disponível em: <http://www.cppreference.com/>.

KUMAR, N.; PIPER, A. *Anjuta IDE Manual*. [S.l.], 2001–2002. Versão 0.0.4, acesso em novembro de 2004. Disponível em: <http://www.anjuta.org-/documents/C/anjuta-manual/>.

LINUX ONLINE, INC. *What is Linux*. 1994–2004. Acesso em novembro de 2004. Disponível em: <http://www.linux.org/info/>.

MORIMOTO, C. *Kurumin Linux*. 1999–2004. Acesso em março de 2005. Disponível em: <http://www.guiadohardware.net/kurumin/>.

ORACLE CORPORATION. *Oracle*. 2005. Acesso em março de 2005. Disponível em: <http://www.oracle.com/>.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL: The world's most advanced open source database*. 1996–2005. Acesso em março de 2005. Disponível em: <http://www.postgresql.org/>.

RESENDE, A. M. P. de. *Introdução à Linguagem C++*. [S.l.]: UFLA/FAEPE, 2002. 105 p.

SCHAEFER, F. R. *GetPot Version 1.0 - Powerful Input File and Command Line Parser*. [S.l.], 2002. Acesso em novembro de 2004. Disponível em: <http://getpot.sourceforge.net/documentation-index.html/>.

SCHILDT, H. *Turbo C++ - Guia do Usuário*. [S.l.]: Makron Books, 1992. 592 p. Tradução de Pedro Manfredi Jr. ISBN 0-07-460647-6.

THE C++ RESOURCES NETWORK. *Description of C++*. 2000. Acesso em novembro de 2004. Disponível em: <http://www.cplusplus.com/info-/description% -.html/>.

THE GNOME FOUNDATION. *GNOME: The Software Desktop Project*. 2003. Acesso em março de 2005. Disponível em: <http://www.gnome.org/>.

THE GTK+ TEAM. *GTK+ - The GIMP Toolkit*. 2005. Acesso em março de 2005. Disponível em: <http://www.gtk.org/>.

THE PERL FOUNDATION. *Perl*. 2002–2005. Acesso em março de 2005. Disponível em: <http://www.perl.org/>.

W3C. *CGI: Common Gateway Interface*. 1999. Acesso em março de 2005. Disponível em: <http://www.w3.org/CGI/>.