Capítulo 10

Manual introdutório de referência

Na internet você pode encontrar um mundo de informações oferecidas, com frequüência, "gratuitamente". Por exemplo

- **Software** `http://www.gnu.org/graphics/agnuhead.html` Este é o site da Fundação para Software Livre (Free Software Foundation - FSF). Lá você pode encontrar uma grande variedade de programas ou `links` para outros sites onde programas podem ser encontrados.
- Linux e software rodando em Linux `http://debian.org` Este é um dos principais pontos de referência para LinuX. Lá você vai encontrar centenas de outros sites.
- **Dicionário on line** `http://www.yourdictionary.com` de repente, se você precisar de saber a tradução de uma palavra...

são alguns dos sites onde você pode encontrar informações on line. Experimente!

Neste capítulo vamos mostrar como você pode obter informações na internet ou na máquina LinuX em que você estiver trabalhando. Mas se acostume com uma idéia relativamente nova: *não precisamos ter tudo guardado na memória*, as informações tem que ficar guardadas em bancos de dados, e nós vamos até elas quando precisarmos. Aqui você vai saber onde existem informações para você usar quando necessitar. Não tente decorar nada, não tente copiar tudo que encontrar. Use e aprenda onde se encontram as *coisas*.

Se você tiver chegado aqui sem ter lido nenhum programa aí gostariamos de conhecê-lo pessoalmente, você deve ser um tipo muito especial, mas por favor corrija este defeito...
Claro, você já deve ter lido muitos programas e visto nas primeiras linhas, logo depois dos comentários iniciais, o símbolo #. A mais comum de todas é # include ...
Este símbolo indica ao compilador que alí se encontra uma "`instrução de compilação`, quer dizer, um comando que o `gcc` deve executar antes de começar a ler o programa.
Vamos discutí-las aqui, mas não se esqueça da observação tantas vezes já repetida: estamos apenas levantando uma pontinha da coberta... há ainda muita coisa que pode ser dita e um local onde você pode encontrar **tudo** é no manual da `Libc` de que já falamos muitas vezes e que você pode consultar em LinuX com a sequência:

- Digite `info libc`. Ai você esta no manual da Libc, edição 1.0 mas não se assuste com este número tão baixo, ela já surgiu muito boa... Digitando apenas `info` você abre o sistema de informações do `LinuX`.
- Procure o que você deseja usando Crl-S e reponsdendo com a palavra chave desejada. Funciona digitando apenas a /.
- Ou comece por ler informalmente o seu conteúdo para ver o que tem lá dentro. Você vai então ver como este livro é diminuto... Você toda libc apenas dando enter, e as páginas irão passando.
- Para sair aperte "q", algumas vezes será preciso colocar o cursor, com o ratinho, dentro do texto, primeiro.

10.1 O Sistema operacional e a shell

Para que um computador "rode" são necessárias dois processos básicos invisíveis para o usuário:

- O assembler Uma pequena linguagem de máquina que é chamada de *assembler* da máquina. É o assembler que caracteriza a diferença entre as máquinas.

- *O sistema operacional* que é um programa de nível um pouco mais alto e bem mais sofisticado que o `assembler` e que vai se comunicar com o usuários, com os programas e com os periféricos. LinuX é um sistema operacional.

Em geral, a grande maioria dos usuários, mesmo usuários avançados, não fazem uso direto do *sistema operacional*, alguns até podem ignorar que ele existe. Em `LinuX`, por exemplo, existe um terceiro programa que é quase um sistema operacional, o `X-windows`, que toma conta da máquina e faz a comunicação

`usuários-programas-LinuX-periféricos`

tudo isto num ambiente gráfico em que o `ratinho` tem uma importância quase tão grande quanto a do teclado.

Se você estiver acostumado e quiser se manter preso a esta gaiola gráfica, salte desta seção para a próxima e simplesmente ignore

X-windows, LinuX, assembler

mas isto não seria típico para um futuro programador na linguagem $\mathcal{C}$ ...

Vamos deixar de lado o `assembler`, vamos conversar um pouquinho sobre `LinuX`. Vamos também ignorar o `X-windows` que ele é tão perfeito qual um burocrata de categoria: trabalha sem que ninguém se dê contas de sua existência.

Quando você atingir a perfeição você poderá se interessar pelo `X` para eventualmente incluir algumas de suas propriedades em seus programas.

`LinuX` é um `unixoide` e $\mathcal{C}$ foi feito por programadores que "nasceram" na perspectiva do `Unix`[1], consequentemente é natural que quem deseje programar em $\mathcal{C}$ precise saber como funciona o sistema operacional. A recíproca é fundamental, quem quiser entender o sistema operacional tem que saber $\mathcal{C}$. Porisso é muito natural que em toda instalação `LinuX` se encontre o `gcc`.

Mas ninguém usa `LinuX` diretamente. Usamos uma linguagem que se comunica com o sistema operacional chamada `bash` que existe em diversos sabores `sh, bash, tcsh` e outros. Sempre tem alguém que acha que uma delas é superior às outras, mas elas fazem as mesmas coisas.

Esta *linguagem* se chama `shell` e você pode ficar sabendo tudo (que é mais importante) sobre `shell` com

`info shell.`

Leia a respeito do `info` em uma das seções adiante neste capíutlo (procure no índice remissivo). Você pode ler todo o manual sobre a `shell` apenas acionando a barra de espaços, experimente.

A palavra inglesa *shell* também se refere a *área de trabalho*, talvez a razão desta confusão venha do fato de que somente podemos usar os comandos da shell quando abrirmos uma área de trabalho. E, que é uma *área de trabalho*?

- Se você estiver no `X` possivelmente, no pé da tela tem um ícone que lembra uma "tv". Clique nele e irá se abrir uma área de trabalho. Agora digite

  `info shell`

  e leia o manual da `shell.` Obviamente, o ícone com a pequena "tv" não precisa estar no pé da tela, onde ele estiver, clique nele...

- Se você não estiver no `X`, no modo gráfico, então você já se encontra numa `shell` ... digite

  `info shell`

  e leia o manual da `shell.`

- Se não houver nenhum ícone lembrando uma "tv" no `X`, clique no botão que inicia os menus e você vai encontrar a palavra `shell`, escolha uma das opções que lhe oferecerem e, quando aparecer uma pequena janela, ative com o ratinho, digite

  `info shell`

  e leia o manual da `shell.`

- Se nada isto der certo, entre em contacto comigo...mas me dê alguma pista sobre o que estiver acontecendo, por exemplo, se o computador está ligado, sem tem energia elétrica, enfim qualquer coisa que ajude a saber como começar. Uma alternativa: se houver alguém a volta,

  `pergunte,`

  lhe asseguramos, foi assim que começamos.

[1] ⌋ foi feito para fazer o Unix

10.2 As instruções de compilação.

1. *#define* Veja como exemplo `traducao.h`. Objetivo definir `macros`, constantes. Por exemplo você pode definir um *modesto* π assim
   `# define pi 3.1415`

2. *#ifdef#endif* Muito poderosa, em geral você vai necessitar dela em programas mais avançados. Serve para alterar outras diretivas, como `include` ou outras definições. Veja o exemplo
   `lookup.c`
   logo no início do programa. Se você usar `ifdef` tem que usar também `endif` para marcar o fim do bloco.

3. `# else` Cria uma alternativa dentro de uma definição, ver `lookup.c`.

4. *#ifndef* Tradução: *se não estiver definido*. Semelhante a `# ifdef` para o caso de não estar ainda definido.

5. *#include* Inclusão de bibliotecas, veja `traducao.h`

6. *#undef* Cancela uma definição. . Se você quiser corrigir um π "muito pequeno" use esta diretiva e depois defina novamente π com "tamanho melhor"...

10.3 Instruções na linha de comando

Ao compilar um programa qualquer você deve ter digitado:
`gcc -Wall -oprog programa.c`
em que `programa.c` é o nome do arquivo que contém o programa que você deseja rodar, o *código fonte* `prog` é nome que você deseja dar ao programa executável, `-Wall` diz ao `gcc` que ele deve ser falador e lhe explicar em detalhe todos os erros que encontrar. Naturalmente `gcc` é o nome do compilador. Aqui `gcc` obedece a uma regra do `Unix`, e `LinuX` é `Unix`, que estabelece que os programas podem ser *modificados* por parâmetros fornecidos na linha de comando.

Você pode compilar um programa simplesmente digitando
`gcc programa.c`
e então o `gcc` irá criar, silenciosamente, um arquivo executável chamado `a.out` e se você digitar
`a.out`
o programa irá rodar, naturalmente se tudo tiver dado certo. Se alguma coisa der errado, `gcc` irá avisá-lo, mesmo que você não tenha solicitado a ajuda, entretanto erros menores que gerariam apenas uma *advertência* não irão aparecer.

- A instrução `-Wall` na linha de comandos é reponsável pela lista de erros e advertências. Sem ela `gcc` vai trabalhar mudo a não ser que haja erros alguns dos quais serão informados ao usuário.

- A intrução `-o` indica ao `gcc` o nome do arquivo executável. Se você a omitir ele vai criar o arquivo `a.out`.
- Se houver um arquivo terminado com ".c" `gcc` vai considerar este como a origem do *código fonte*, o programa que deve ser compilado.
- Se você incluir a opção `-c`, o `gcc` irá apenas fazer a análise sintática do código fonte e criar um arquivo `programa.o` em que `programa` é prefixo do arquivo que contém o código fonte. Mas se você também incluir

  `-oprog`

  ele vai colocar o arquivo criado em `prog` que, entretanto, não será executável. Esta opção `-c` é útil para grandes programas afim de verificar a sintaxe porque é um pouco mais rápida.

Se você digitar

```
gcc --help | more
```

o compilador vai lhe mostrar uma listagem das instruções de compilação que podem ser fornecidas na linha de comando com uma *breve* descrição. Ver na `Libc` descrições mais detalhadas ou com `info gcc`[2] uma ajuda mais técnica sobre o compilador. Evite de se perder, entretanto, vá devagar porque há muita informação.

Em LinuX existe um formato compilado das funções que se encontram nas bibliotecas que é mais eficiente e mais rápido de ser usado, mas para acessá-los é preciso usar a instrução de compilação $-lm$ na linha de comando. O programa `grafun.c`, por exemplo, se compilado com

```
gcc -Wall -oprog grafun.c
```

vai lhe apresentar erros inexperados, como dizer-lhe que a função "sin" é desconhecida. A saída é compilá-lo com

```
gcc -lm -Wall -oprog grafun.c
```

que vai informar ao `gcc` que deve usar a biblioteca compilada. Abaixo você tem uma listagem parcial do comando `gcc --help`. Se você quiser ler o seu conteúdo com calma, execute o seguinte:

- `gcc --help > teste` e o resultado desta listagem vai ficar guardada no arquivo "teste". Preste atenção, se o arquivo existir, ele vai ser perdido ficando agora os dados desta listagem nele. Porisso sugerimos "teste", ou "lixo", etc... nomes que você deve usar para arquivos temporários. Evite de usar `temp` ou `tmp` porque estes arquivos são usados pelo sistema.
- Leia o arquivo com um editor de textos.
- O símbolo > é um comando do LinuX chamado de "redirecionador da saída de dados". Sem ele os dados iriam para a "saída de dados padrão" que normalmente é o vídeo.

Segue abaixo um trecho do resultado de `gcc --help` com a tradução para português.

[2] na prósima secção discutimos o sistema `info`

```
Usage: gcc [options] file...
Options:
--help Mostra esta ajuda
(Use '-v --help', em geral para obter ajuda sobre processos)
-save-temps Preserva os arquivos temporarios
-pipe      Usa pipes em vez de arquivos temporarios
-B <directory> Acrescenta <directory> aos caminhos de busca do
compilador
-b <machine>   Executa gcc para <machine>, se instalado
-V <version>   Roda a versao <version>, do gcc, se instalado
-v Mostra os programas chamados pelo compilador
-E Apenas Preprocessamento, nao compila nem
   assembla nem link-edita.
-lm  Use a biblioteca compilada de matematica
-S Apenas compila, nao cria o assembler ...
-c Compila e cria o assembler, mas nao link-edita.
-o <file>   Coloca o executavel em <file>
-x <language>  Escolhe a linguaguem:
Linguagens aceitas inclue: c, c++, assembler, none
'none' significa que gcc deve escolher
a linguagem a partir da extensao.
```

Observe que as opções não apresentadas podem ser importantes, entretanto não agora no início. Inclusive algumas dessas que apresentamos não lhe serão significativas agora. Se você tiver analisado o resultado do `gcc --help` vai observar que acrescentamos uma que lá não estava explicitada, $-lm$.

10.4 Operadores aritméticos e lógicos

Veja no índice remissivo alfabético em que capítulo se encontram os operadores aritméticos e lógicos.

10.5 A libc

> Você acessa a `libc` com
>
> `info libc`
>
> É o manual mais completo sobre a linguagem $\mathcal{C}$ que você pode encontrar, e se encontra ao alcance de seus dedos, dentro da máquina `LinuX` que você estiver usando. Aqui vamos apresentar alguns flashes deste manual.

Para falar a verdade vamos mostrar-lhe como, usando `info`, você pode passear num dos "`sites`" mais completos da internet , a máquina `LinuX` em que você estiver trabalhando. Como no resto do livro, coloque os dedos no teclado do computador e nos acompanhe na viagem.

Se você digitar apenas `info` lhe vai ser apresentado um índice geral com todas as informações sobre programas ou pacotes instalados na máquina. Vale a pena começar por aí se você ainda não fez isto.

Se você estiver usando `xemacs` basta clicar com o ratinho no canto superior direito, `help` e escolher `info (online docs)`. Neste caso você pode viajar apenas usando o ratinho. Se você for um professor talvez seja interessante ler as primeiras linhas do `info` para ver que você pode criar informações sobre o seu curso dentro do sistema `Linux`...

Os manuais dentro do `info` estão marcados com um asterisco, dê um `enter` ao lado de um nome com asterisco e você entra no manual daquele pacote. Procure `libc` usando:

- Ctrl S
- usando a barra / se você estiver entrado no `info` fora do `xemacs`

Dentro do `xemacs` você pode usar os botões a semelhança do que você faria na `internet` para ir e voltar (ou sair) `exit`.

Se você tiver entrado no `info` no modo texto, use `q` para sair. Para retornar para um nível superior, dentro da árvore de informações, use `u (up)` ou `p (previous)`. As teclas `pgU` e `pgDn` funcionam para subir ou descer o texto dentro da página em uso.

E, naturalmente, `info info` lhe dá todas as informações sobre o sistema de informações. E tudo isto é uma ótima oportunidade para aprender inglês...

Procure agora `libc` usando um dos métodos descritos acima.

No momento em que estamos terminando esta versão do livro, se encontra instalada neste máquina a versão 0.10 da `libc`, um número extremamente modesto para o seu poder de informações. Os primeiros tópicos são:

- * Introduction:: Purpose of the GNU C Library. Aqui você vai encontrar uma descrição do que lhe oferece a `libc` e como usá-la.
- * Error Reporting:: How library functions report errors.

  O título diz tudo, mas deixe-nos acrescentar, aqui você vai descobrir para que serve o último comando da função `principal()`, o `return numero`.
- * Memory:: Allocating virtual memory and controlling paging.

  É um tópico avançado, os itens do manual absolutamente não se encontram arrumados em ordem crescente de dificuldade.
- * Character Handling:: Character testing and conversion functions.

  Traduzindo a descrição: “testando caracteres e funções de conversão”.
- * String and Array Utilities:: Utilities for copying and comparing strings and arrays.

  Traduzindo a descrição: “utilitários para copiar e comparar vetores de caracteres (strings) e vetores”.
- * Character Set Handling:: Support for extended character sets.

- * Locales:: The country and language can affect the behavior of library functions.

  Traduzindo a descrição: "País e lingua podem afetar o comportamento das funções nas bibliotecas." Explicando melhor, com um exemplo, as funções que manipulam *data* podem se acondicionar para a forma como escrevemos as datas no Brasil.

- * Message Translation:: How to make the program speak the user's language.

  Traduzindo a descrição: "Como fazer os programas falar a linguagem do usuário". Aqui você já pode ver um defeito do nosso livro, não usamos esta possibilidade, as mensagens de erro podem aparecer em diversas linguas naturais.

  Se você entrar neste tópico verá que o primeiro parágrafo diz uma das verdades menos compreendidas pelos programadores:

  "A comunicação dos programas com os seres humanos deve ser planejado para facilitar a tarefa destes, (dos humanos). Uma das possibilidades consiste em usar as mensagens na lingua que o usuário preferir." Exatamente o contrário do que muita gente imagina quando escreve programas, "que os humanos é que têm que se adaptar à informática"... Entretanto ainda não está fácil de traduzir as mensagens de erro, é poriso que eu deixei esta questão de lado, pelo menos no momento.

O pequeno excerto que apresentei acima deve-lhe mostrar que vale a pena estudar a `libc`.

Poderiamos continuar analisando *todo* o manual, mas isto transforamaria este livro em volume de mil páginas, e inutilmente porque você tem a `libc` na ponta dos seus dedos, em inglês, é verdade. Mas se você não se preocupar em dominar o inglês, pelo menos para leitura (fácil), tem muito coisa interessante que vai ficar ficar longe do seu alcance.

10.6 Manual do compilador `gcc`

Faremos referência aqui às informações *on line* que podem ser obtidas numa máquina `LinuX` sobre o compilador `gcc`. Vamos discutir `info gcc`. Leia o parágrafo anterior, neste capítulo, sobre o sistema `info`.

Vamos supor que você digitou numa `shell` do `Linux`

```
info gcc
```

vamos lhe mostrar uma forma diferente de usar `info`

Você pode ler todo o manual do `gcc` como se fosse um livro, (pouco útil), é melhor procurar questões que resolvam problemas pendentes.

Mas se você quiser ler como se fosse um livro eletrônico (que de fato ele é) basta ir dando toques na barra de espaço e as páginas irão correndo antes os

seus olhos na tela do micro. Vale a pena fazer para uma primeira leitura (se aplica melhor à `libc`).

Os primeiros toques na barra de espaço vão lhe permitir uma leitura do índice. Em seguida ao índice você vai *cair* no primeiro capítulo se continuar dando toques na barra de espaço e assim sucessivamente.

Observe que as teclas `PgUp, PgDn` funcionam como esperado, para subir e descer páginas. Se você tiver uma boa instalação `LinuX` provavelmente não precisara de alguns itens como `instalação` e semelhantes. Mas certamente será interessante fazer várias leituras do capítulo `*Invoking GCC::`. Faça várias leituras, uma primeira, rápida, e depois outras em que você vai se demorar em algum tópico que lhe chame sua atenção.

Quando seus programas ganharam mais densidade, quando tiver um projeto, vai precisar usar do utilitário `make` que também existe no `BC`. Ele serve para interligar diversos programas e bibliotecas em único código otimizando o resultado.

Finalmente você encontra quase ao final do "livro" a famosa `GPL`.