FMA 215

Aula 8: Editando com o Emacs

Prof. Jorge L. deLyra

16 de Maio de 2002

1 Alguns Conceitos Relevantes

Editor: o editor de texto ASCII é uma das mais importantes ferramentas do sistema. É com este tipo de programa que podemos colocar conteúdo útil em arquivos, para os mais variados fins. É assim que se faz quase tudo no sistema, desde escrever arquivos de configuração do próprio sistema, passando por escrever mensagens de correio eletrônico e o texto de documentos a serem formatados das mais diversas formas, até escrever programas a serem interpretados ou compilados por outros programas. Em resumo, o editor de texto é a principal ferramenta que usamos para inserir no sistema a *nossa* informação.

Existem no Linux duas classes gerais de editores de texto, os quais funcionam segundo dois estilos completamente diferentes, que chamaremos de estilo Emacs e de estilo Vi. Os editores do estilo Emacs funcionam de uma forma mais familiar, entrando diretamente em modo de inserção de texto e aceitando comandos através de caracteres de controle e teclas especiais. Já os do estilo Vi funcionam ao contrário: estão normalmente em modo de comando e é necessário entrar explicitamente em modo de inserção através de um comando para poder escrever algo no arquivo.

Nestas notas vamos estudar apenas editores do estilo Emacs. Na galeria de trabalhos dos alunos desta disciplina há vários trabalhos sobre o uso dos editores da classe Vi. Um deles é mencionado em um dos exercícios na seção de problemas e dicas desta apostila. Além disso, a forma de se sair destes editores, que está meio longe de ser intuitiva, está incluída no documento sobre as saídas de emergência.

Arquivos ASCII: o arquivo de simples texto ou arquivo ASCII (que é uma abreviação para "American Standard for Computer Information Interchange") é aquele que contem apenas caracteres que representam letras, números e outros símbolos legíveis, segundo um padrão que é de fato universalmente aceito. Arquivos deste tipo podem ser lidos e manipulados diretamente pelo usuário através do uso de editores ou paginadores de uso geral, pois a forma binária do seu conteúdo corresponde a um padrão aberto e bem conhecido de todos.

Ou seja, os arquivos ASCII são aqueles diretamente legíveis pelos usuários, através da utilização de programas de uso geral, sem a necessidade de aplicativos específicos ou proprietários. Uma das características importantes dos sistemas livres e abertos é que todos os seus arquivos de configuração sejam arquivos ASCII, facilmente legíveis ou copiáveis por qualquer um. Aliás, o espírito dos sistemas abertos é de forma geral, facilitar ao máximo a cópia e disseminação de informação, ao contrário dos sistemas fechados e proprietários, que visam dificultar ao máximo a cópia e distribuição de pelo menos parte da informação.

Arquivos de backup: o editor `emacs` mantém dois tipos de arquivos de backup dos arquivos que você editar com ele. Se você sair legalmente do editor, ou seja, sem interromper o processo e sem erros de sistema, o arquivo anterior, aquele que você começou editando, estará guardado no diretório corrente de trabalho com o nome `<arquivo>~` ou seja, com um `~` adicionado ao final do nome do arquivo. Apenas um backup deste tipo é mantido pelo editor, para cada arquivo que você editar, ou seja, se você editar de novo o mesmo arquivo este arquivo de backup será sobrescrito com o arquivo atual.

Se o processo do `emacs` for interrompido ou se houver um erro de sistema como, por exemplo, um crash, pode ser que parte de seu trabalho de edição não tenha sido perdido pois a intervalos regulares o editor faz um backup automático do arquivo que está sendo editando com o nome `#<arquivo>#`. Você pode usar este arquivo para recuperar o seu trabalho de edição até um certo ponto. Este arquivo de backup automático é apagado pelo editor se você sai dele legalmente, ele se transforma no arquivo `<arquivo>~`. Em geral apagamos os arquivos do tipo `<arquivo>~` depois de sair do editor sem erro, se estamos certos de que ele não será mais necessário.

2 Tarefas

O editor padrão que utilizaremos é o `emacs`. Ele pode ser utilizado tanto em terminais de caracteres quanto no sistema de janelas X11. Em nossos sistemas, o editor está configurado de forma mais completa quando roda no ambiente X11 e por isto vamos usá-lo dentro deste ambiente. Ele é um editor muito grande e poderoso, que por isto mesmo é, às vezes, um pouco lento, em especial nas tarefas mais complexas envolvendo programação de teclado para fins de edição automática.

Existem vários clones muito úteis do `emacs` que, ao mesmo tempo em que não têm todas as poderosas propriedades dele, são bem menores e mais rápidos. Em particular, há o `jed`, que ainda tem muitas das capacidades mais úteis do `emacs`, sendo portanto muito útil para uso geral. Há uma versão deste editor para o sistema X11 que se chama `xjed`. Há também o `zile`, que é realmente muito pequeno e só tem as coisas mais básicas, sendo útil em tarefas de administração de sistemas. A maior parte dos comandos básicos de edição que vamos ver aqui vale para qualquer um destes editores, que são todos editores da classe Emacs.

Aprender a usar o editor é basicamente uma questão de treino repetitivo. Como o editor é um aplicativo que estabelece uma sessão interativa com o usuário, ele tem uma sintaxe interna própria. Aprender a usar o editor significa aprender esta sintaxe, que é o que faremos aqui. Nesta aula nos limitaremos às funções mais simples de edição com o `emacs`. Existe uma grande quantidade de comandos muito úteis para um ou outro fim, incluindo a possibilidade de edição automatizada de grandes quantidades de texto. As alternativas de edição mais sofisticadas e avançadas serão deixadas para uma outra oportunidade.

O número total de comandos do editor é muito grande, na realidade ele é essencialmente infinito, pois o editor é extensível tanto por programadores que escrevam macros complexos para ele quanto pelo próprio usuário, que pode combinar comandos de forma a definir novos comandos. Entretanto, apenas um pequeno número de comandos é essencial para que se possa começar a usar o editor, como veremos. Para fins de ilustração, vamos utilizar o editor em um ambiente de programação.

Já vimos antes qual o significado da combinação `[Ctrl]-C` (segura-se pressionada a tecla "control" enquanto se digita uma vez a tecla da letra C). Muitos dos comandos do `emacs` são executados através de combinações deste tipo, de forma que vamos adotar nesta aula a abreviação `^C` para este tipo de operação. Assim, `[Ctrl]-A` será o mesmo que `^A` e assim por diante. Note que isto é apenas uma abreviação para facilidade de leitura deste documento e que a tecla que produz o símbolo "`^`" não está envolvida nisto de todo.

1. Crie em sua conta um diretório chamado `decmat` e vá para dentro dele. Use um browser para fazer o download, a partir das páginas da disciplina, dos arquivos que estão nos links

   `http://latt.if.usp.br/fma215/materiais/aula-08/`
   `http://latt.if.usp.br/fma215/materiais/aula-08.tgz`

   No primeiro link você encontra os arquivos individuais, no segundo um único arquivo "tar" comprimido contendo todos os arquivos juntos. Para abrir este arquivo "tar" você terá de usar o comando `tar` com as opções `-xzvf`; para descobrir como, olhe as páginas de manual do comando `tar`. Alternativamente você pode obter cada um dos arquivos individualmente com o browser. Faça o que for mais confortável para você e coloque cópias de todos os arquivos dentro do diretório `decmat`. Trata-se de um conjunto de arquivos relativos a um programa escrito na linguagem Fortran.

2. Examine o resultado do procedimento com o comando `ls`. Em seguida execute, ainda dentro deste diretório, o comando `make` e veja o que acontece. Deverá ter aparecido automagicamente no diretório um arquivo executável chamado `decmat`. Execute este arquivo, digitando `./decmat`, digite o input solicitado, escolhendo a alternativa `1` e veja o que acontece. Em seguida execute `make clean` e verifique o efeito disto no diretório, usando o `ls`.

3. Para editar o arquivo contendo o programa principal, execute o comando `emacs Decompmat.f`. Aí está o programa em Fortran. Observe que várias regiões do texto estão coloridas com cores que caracterizam a sua função como parte da linguagem. Por exemplo, todo o texto que está em vermelho constitui comentários para o leitor, os quais são completamente ignorados pelo compilador da linguagem (o `g77`).

4. Talvez a coisa mais importante a fazer de imediato seja aprender como sair do editor! Faça isto, digitando a combinação `^X^C`. Se o editor lhe perguntar se você quer salvar o arquivo, isto significa que você o alterou

acidentalmente. Caso isto aconteça, diga que não quer, digitando `n`. Ele vai solicitar uma confirmação desta escolha, perguntando se você quer mesmo sair perdendo as mudanças que foram realizadas. Diga que sim digitando `yes` por extenso.

5. Vamos agora editar outro tipo de arquivo, executando o editor com outro argumento, `emacs Makefile`. Este é um arquivo de configuração do programa `make`, que você rodou na segunda tarefa. Estão codificadas neste arquivo as regras e dependências entre arquivos que permitem ao comando `make` compilar automagicamente o programa. Dê uma boa olhada e veja se você reconhece algo do que vê. Note, mais uma vez, que o display do `emacs` está colorizado organizadamente.

6. Vamos voltar a editar o programa principal, digitando novamente `emacs Decompmat.f`. Vamos agora passear pelo arquivo afora, usando os comandos de movimentação de cursor do `emacs.` A forma mais simples de se fazer isto é usando as teclas de cursor, com as setinhas. Elas estão organizadas da seguinte forma no teclado, num "keypad" logo à direita da parte principal deste:

   [↑]

   [←] [↓] [→]

   Elas fazem exatamente o que a intuição está gritando: movimentam o cursor um caracter na direção indicada. Experimente fazer isto e dê uma passeada até o final do arquivo e depois volte até o início. Observe que cada vez que o seu cursor é movimentado de forma a sair da área visível do texto, este pula para cima ou para baixo uma quantidade equivalente a metade da altura da janela que você estiver usando, tendendo, portanto, a retornar o cursor à região central da tela. Chamamos isto de "jump scrolling".

7. Se você segurar uma das teclas de cursor apertada, a tecla será repetida e o cursor se movimentará mais rapidamente pelo texto. Mas mesmo assim esta ainda é uma forma lenta de movimentar o cursor. Existem comandos que permitem uma movimentação mais rápida dele.

   Em particular, os comandos `^A` e `^E` levam o cursor para o início e para o final da linha, respectivamente. Observe que estes são os mesmos comandos que funcionavam na edição de linha de comando da shell. Isto está assim porque a shell que usamos (`tcsh`) foi configurada para que os seus comandos de edição de linha de comando funcionem no estilo da família Emacs de editores.

Além disso, as teclas `[PageUp]` e `[PageDown]` do seu teclado servem para paginar o texto do arquivo para cima e para baixo, movimentando de cada vez uma quantidade de linhas equivalente à altura da tela que você estiver usando. Note que o "Up" e "Down" a que estes comandos se referem representam a movimentação da sua janela de visão em relação ao conjunto completo do texto que existe.

Use estas teclas de movimentação rápida para passear pelo arquivo. Observe que, mais uma vez, manter-se uma tecla apertada causa a sua repetição contínua. Apesar de que isto não é muito útil no caso dos comandos `^A` e `^E`, certamente é útil no caso dos comandos `[PageUp]` e `[PageDown]`.

8. Vamos agora apagar alguns caracteres, para a frente e para trás do cursor. Não se preocupe com a possibilidade de destruir o conteúdo do arquivo, afinal você pode simplesmente não salvá-lo no fim. As mudanças que você fará estarão acontecendo numa cópia do arquivo que o editor colocou na memória do computador, não diretamente no disco. Chamamos a esta cópia em memória de "buffer". O arquivo que está no disco só será sobrescrito quando você salvar o buffer no qual está trabalhando, o que em geral é feito ao final da sessão de edição.

   Para apagar caracteres para trás use a tecla `[BackSpace]` e, para apagar caracteres que estão à direita do cursor, use o comando `^D` (do Inglês "delete", o nome do comando que está associado a esta tecla é `delete-char`). Vá em frente e apague alguns caracteres. Em ambos os casos, aperte e segure a tecla para apagar repetidamente.

9. Também é possível apagar texto em maiores quantidades de cada vez. O comando `^K` (de `kill-line`) apaga todo o texto à direita do cursor, até o final da linha. Se o cursor estiver no início da linha, `^K` apaga todo o conteúdo da linha, deixando a linha em branco. Se o cursor já estiver no final da linha, `^K` apaga o caracter de controle que representa uma quebra de linha no arquivo, com o resultado que a linha seguinte é concatenada com a linha onde está o cursor. Assim, com o cursor

no início da linha dois comandos ^K seguidos eliminam completamente a linha: um apaga o conteúdo e o outro elimina a linha em branco remanescente. Tente todas estas possibilidades em algumas das linhas do arquivo.

10. O editor mantém em memória as últimas coisas que foram apagadas com o comando ^K, de forma que é possível recuperá-las. Para fazer isto usamos o comando "`yank`", que é executado pela combinação de teclas ^Y. Execute este comando uma vez para verificar que a última coisa que você apagou com ^K é reinserida no arquivo, na posição atual do cursor.

11. Pode-se usar o comando ^Y para copiar linhas ou partes de linhas de um arquivo de um lado para o outro, quantas vezes se quiser. Aperte ^Y várias vezes seguidas para ver o que acontece. Observe que a combinação de comandos ^K^K^Y apaga e recupera uma linha inteira. Trata-se na realidade de uma operação de "copy" que coloca uma cópia da linha em memória, em um buffer especial. Daí em diante esta linha pode ser copiada em outros lugares com um simples ^Y, o que é portanto uma operação de "paste". Experimente fazer isto com algumas das linhas do arquivo.

12. É possível copiar, cortar e colar quantidades maiores de texto, movendo-as ou repetindo-as dentro do arquivo. Para fazer isto é necessário primeiro fazer uma marca lógica em algum lugar do arquivo, ou seja, definir uma posição de referência. O comando que faz isto é o "`set-mark-command`", executado por `^[Space]`, que marca a posição onde está o cursor. Em seguida podemos movimentar o cursor para outra posição, digamos várias linhas para baixo, cortando então todo o texto que está entre a posição da marca e a posição do cursor com o comando ^W. Tente fazer isto com alguma parte do arquivo.

13. Mais uma vez, o texto cortado vai para o buffer especial e pode ser recuperado pelo comando "`yank`", executado por ^Y. Faça isto para ver como funciona. Assim como antes, o uso repetido de ^Y copiará o texto repetidas vezes no arquivo. Desta forma pode-se mover ou copiar qualquer quantidade de texto, desde um único caracter até o texto integral do arquivo. A cópia de blocos de texto com posteriores pequenas modificações é uma das principais ferramentas para poupar tempo e

trabalho na edição e criação de arquivos, em especial em trabalhos de programação e de configuração de sistemas.

14. Um comando extremamente útil é aquele que permite que você desfaça mudanças que tenha feito no texto, na ordem inversa daquela em que as mudanças foram feitas. Toda mudança que você faz no texto é registrada pelo editor, em blocos de alguns caracteres ou comandos de inserção. O comando que desfaz as mudanças é o "`undo`", executado através de (`^/`). Digite duas ou três linhas quaisquer de texto no arquivo e, em seguida, use o comando (`^/`) algumas vezes, até ter eliminado completamente as mudanças que introduziu.

15. Uma outra propriedade muito útil do editor é a sua capacidade de procurar coisas no texto. O sistema de procura funciona de uma forma muito interessante, chamada de "procura incremental". Comece indo para o início do arquivo. Para entrar em modo de procura incremental para a frente, execute o comando `^S` (`isearch-forward`). Digitando agora uma letra de cada vez, entre a palavra `determinant`. Observe como, a cada letra que você digita, o cursor vai até a primeira instância da combinação de letras que foi formada até aquele momento.

16. Para sair do modo de procura você pode usar o comando `^G`, que é a forma de sair de qualquer comando ou modo e voltar ao modo padrão de inserção. A procura também pode ser feita para trás, de forma semelhante, com o uso do comando `^R` (`isearch-backward`). Execute este comando e digite a palavra `decompmat`, uma letra de cada vez. Para sair do modo de procura use mais uma vez o comando `^G`. Na realidade, você pode sair do modo de procura incremental digitando qualquer comando do editor, como `^A` ou `^E`, por exemplo.

17. O comando `^G` (`keyboard-quit`) é a forma universal de sair de qualquer coisa dentro do editor. Se necessário, use-o mais de uma vez em seguida. Ele é a saída de emergência de qualquer tipo de enrosco no qual você acabe se encontrando dentro do editor. Um outro comando muito útil para você se recuperar de confusões é o comando `^L`, que limpa a tela do terminal e recoloca nele uma nova cópia do conteúdo do buffer no qual você está trabalhando.

    Às vezes pode acontecer da tela do seu terminal ficar confusa ou de algum outro programa enviar algum output indesejado para ele. Nestes casos use o comando `^L` (`recenter`) para limpar a tela. Além disso, este comando é útil para centralizar na tela o trecho do texto onde você está trabalhando, pois ele irá fazer com que a posição atual do cursor fique bem no meio dela, da direção vertical. Experimente usar este comando algumas vezes.

18. Como o número de combinações possíveis do tipo `^A`, `^B` e `^C` é limitado, existem certos comandos como, por exemplo, o comando `^X`, que são interpretados como prefixos de outros comandos e não como comandos completos em si. Assim, `^X` é sempre o início de algum outro comando. Já vimos um caso como este, era o comando para se sair do editor, `^X^C`. Execute-o mais uma vez, evitando salvar o arquivo ao sair do editor.

19. Execute o comando `emacs` sem argumento. Será aberta pelo editor uma janela sem conteúdo, correspondente a um buffer de rascunho chamado "scratch". Como trata-se de um rascunho, qualquer conteúdo que você coloque nele será descartado pelo editor quando você sair, a não ser que você execute explicitamente um comando para salvar o conteúdo antes de sair. Vamos então examinar os comandos de leitura e escrita de arquivos a partir do editor.

20. Execute o comando `^X^F` (`find-file`) para ler e editar um arquivo de dentro do editor. Aparecerá um prompt na última linha de baixo na janela, abaixo da barra de status que contém algumas informações sobre o status do editor. Digite neste prompt o nome do arquivo `checkmats.f`, que é uma das rotinas usadas no programa `decmat`, apertando em seguida a tecla `[Enter]`. Observe como é criado um novo buffer e o arquivo aparece na tela. Execute mais uma vez o comando `^X^F` e digite no prompt apenas a letra `D` seguida de um `[Tab]`. Você reconhece o comportamento? Aperte `[Enter]` para ler mais uma vez o programa principal `Decompmat.f`. Lembre-se de que se você quiser sair do comando `^X^F` sem de fato executá-lo, basta usar o comando geral de cancelamento `^G`.

21. Observe que o programa `Decompmat.f` aparece mais uma vez na tela. Entretanto, o conteúdo do buffer referente ao arquivo `checkmats.f` que estava sendo editado anteriormente não foi perdido, o buffer continua

em memória. Você pode verificar isto usando o botão esquerdo do mouse para abrir o menu `Buffers` na barra de menus que fica no topo da janela do editor. Segurando o botão do mouse e levando o cursor do mouse até um dos ítens que aparecem no menu fará com que o buffer correspondente apareça na tela. Observe que alguns buffers correspondem aos arquivos que você está editando, enquanto outros são buffers internos do editor. Dê uma olhada em cada um deles.

22. Volte para o buffer `Decompmat.f`. O comando para salvar as mudanças feitas num buffer é `^X^S`, o que vai sobrescrever o arquivo correspondente no disco. Tente executar este comando. Se você não fez de fato nenhuma mudança no buffer, vai receber uma mensagem (na linha de comando abaixo da barra de status de baixo) dizendo que não há o que salvar. Faça uma mudança qualquer no buffer, por exemplo acrescente algumas linhas em branco, repetindo em seguida o comando `^X^S`. Desta vez o arquivo será salvo com as mudanças feitas. Saia do editor e entre de novo com o comando `emacs Decompmat.f` para verificar isto.

23. Caso você tenha feito mudanças mas não queira sobrescrever o arquivo original, pode escrever o buffer para um novo arquivo, escolhendo um novo nome para ele. O comando para se escrever um novo arquivo é `^X^W`. Aparecerá um prompt na linha de comando, que você poderá completar com um nome de arquivo de sua livre escolha, apertando em seguida a tecla `[Enter]`. Observe que, assim como no caso do comando `^X^F` que permite a leitura de um novo arquivo, a tecla `[Tab]` pode ser usada nesta linha de comando para completar de forma automática o nome dos arquivos. Naturalmente, isto só é útil neste caso se você quiser escrever em cima de algum outro arquivo já existente. Lembre-se de que se você quiser sair do comando `^X^W` sem de fato executá-lo, basta usar o comando geral de cancelamento `^G`.

24. Já vimos que é possível estar editando com o `emacs` dois ou mais arquivos ao mesmo tempo, que estarão em buffers diferentes. Também é possível mostrar mais de um arquivo simultaneamente na janela. Para isto, dividimos a janela em duas, que passarão a funcionar como se fossem dois editores diferentes. O comando para fazer isto é `^X2`. Note que não há tecla de controle no caso do segundo caracter desta sequências. Deve-se a possibilidade de fazer isto ao fato de que está sendo usado um caracter de controle no prefixo, o caracter `^X`. Execute

este comando, verificando que aparecem de fato duas janelas, uma em cima da outra, ambas mostrando o mesmo buffer.

25. Você pode mudar o seu cursor de uma janela para a outra usando o mouse (basta clicar na janela) ou o comando `^XO` (trata-se da letra O, não do zero). Faça isto e vá para a janela de baixo. Uma vez lá, use o comando `^X^F` para ler o arquivo `Makefile` para dentro desta janela. Pronto, agora você está editando os dois arquivos ao mesmo tempo. Isto é quase como ter dois editores rodando ao mesmo tempo no sistema X11, com a diferença de que o que você matar com `^K` ou `^W` em uma janela poderá ser recuperado na outra, pois o buffer especial que guarda o texto que foi cortado é o mesmo para as duas janelas. Experimente fazer isto, trata-se de uma forma muito útil de se transferir material selecionado de um arquivo para outro. É claro que isto não é muito útil no nosso caso aqui, pois os dois arquivos são de natureza muito diferente um do outro. Portanto, desfaça as mudanças que fizer neles, depois de ter usado os comandos `^K` e `^Y`.

26. Uma coisa importante a saber, como sempre, é como sair desta situação de duas janelas. Há duas formas de fazer isto. Primeiro, você pode matar todas as janelas menos aquela na qual está o cursor neste momento, usando o comando `^X1` (`delete-other-windows`). Experimente fazer isto. Note que isto só elimina a janela, não o buffer que estava nela. Verifique este fato dando uma olhada no menu de buffers na barra superior de menus. Em seguida execute de novo o comando `^X2` (`split-window-vertically`) e, estando o cursor na janela de cima, coloque nela o buffer correspondente ao outro arquivo que estivemos editando, usando o mouse no menu de buffers. A outra forma de sair desta situação de duas janelas é eliminar uma janela, aquela na qual você está no momento, com o comando `^X0` (`delete-window`). Experimente fazer isto.

27. Em algumas circunstâncias o editor vai automaticamente mostrar uma segunda janela. Por exemplo, para se pegar um buffer é possível usar, além do mouse na barra de menus, o comando `^X^B`. Este comando vai mostrar uma lista dos buffers existentes em uma segunda janela que vai aparecer automaticamente. Clicando o botão do meio do mouse num dos nomes da lista, ou indo para esta nova janela com `^XO`, movimentando o cursor para cima do nome e apertando `[Enter]`, o buffer

correspondente será colocado na tela. Com a execução do comando esta segunda janela será fechada automaticamente. Caso você desista de executar o comando, pode simplesmente fechar a janela com ^XO. Experimente pegar um novo buffer das duas maneiras descritas acima.

28. A pequena janela de uma linha que aparece na linha de comando, abaixo da linha de status inferior, quando usamos comandos como o ^X^F ou o ^X^W, também é uma janela e podemos entrar e sair dela usando o comando ^XO, como fazemos com duas janelas de texto. Use o comando ^X^F para ler algum outro arquivo e, enquanto a janela de comando estiver aberta, use o comando ^XO para sair e entrar nela. Para terminar abra quatro janelas, repetindo o comando ^X2 dentro de cada uma de duas janelas que já existam devido a um ^X2 anterior. Use o comando ^XO para mudar de janela em janela. Depois elimine todas as janelas e saia do editor.

29. Verifique se há no seu diretório corrente de trabalho arquivos do tipo #<arquivo># ou <arquivo>~, ou seja, arquivos de backup indicando que você vez de fato algumas mudanças nos arquivos que puxou das páginas da disciplina. Se você tiver de fato realizado mudanças e tentar rodar de novo o comando make, é bem possível que aconteça algum erro. A forma mais fácil de consertar isto é puxar os arquivos de novo, de forma que simplesmente apague todos estes arquivos de backup com comandos como rm *~ e rm #*#.

3 Problemas e Dicas

1. Traduza os comentários que aparecem nos diversos arquivos do programa decmat para o Português. Não se preocupe com acentos ou cedilhas, por enquanto.

2. O emacs tem um tutorial interno. Nem todo mundo se dá bem com ele mas não custa tentar. Para usá-lo, basta entrar no editor com o comando emacs, sem argumentos, digitando a seguir o comando ^H, que entra no sistema interno de help do editor, seguido de um t, que chama o tutorial. Em suma, use ^Ht e divirta-se à vontade com o tutorial.

3. Traduza o programa que apresentamos para alguma outra linguagem de programação que você conheça. Além do compilador Fortran (g77)

existem no sistema compiladores para as linguagens C (`gcc`), C++ (`g++`) e Pascal (`gpc`), bem como interpretadores de Basic (`yabasic` ou `bwbasic`), entre outros.

4. Crie um diretório chamado `apostilas` e coloque dentro dele cópias dos arquivos contendo as fontes das apostilas desta disciplina, que podem ser obtidos nas páginas da disciplina,

   `http://latt.if.usp.br/fma215/apostilas/`

   em arquivos `tar`, através dos links com nome `LaTeX` que existem para cada uma das apostilas. Você vai ter de descobrir como abrir um arquivo `tar` comprimido, usando o comando de mesmo nome. Para este exercício basta que você puxe uma das apostilas, digamos a primeira.

   Uma vez que consiga fazer isto, vá para dentro do diretório e execute o comando `make clean` e verifique o que acontece. Os arquivos com terminação `.tex` são os arquivos contendo o texto das aulas desta disciplina. Eles estão escritos na linguagem LaTeX, que é um poderoso processador de texto, que veremos mais tarde. Edite uma das aulas, por exemplo o arquivo `aula-01.tex` e verifique que o texto fica colorizado na tela do editor.

   Edite o arquivo `Makefile`, ache um local onde está o comando `Latex` e troque-o por `latex`. Salve o arquivo, saia do editor e em seguida execute o comando `make aula-01.dvi` e veja a aula ser formatada automagicamente pelo processador de texto. Você poderá ver a aula no seu terminal X11 com o comando `xdvi aula-01.dvi`, por exemplo. Execute `make aula-01.ps` para criar um arquivo que poderia ser enviado para uma impressora. Você também pode ver este arquivo na tela do seu terminal X11 com `gv aula-01.ps`, por exemplo.

5. A outra grande classe de editores do sistema é a classe do editor Vi. Trata-se do editor tradicional dos antigos sistemas Unix, ainda muito usado, em especial para atividades de gerenciamento de sistema. Como acontece com tantas coisas, há usuários que são fanáticos de uma ou de outra das duas classes principais de editores. O nome do Vi vem de *Visual*, mas para os que não suportam a sua forma um tanto peculiar de funcionamento, talvez *Virtually Impossible* seja mais descritivo. Da mesma forma, o nome do Emacs vem de *Editor Macros* mas, para o

outro time ele é uma abreviação de *Eats Memory And Crashes the System*.

Apesar de que não vamos fazer muito uso dos editores da classe Vi nestas apostilas, é uma boa idéia ter pelo menos uma idéia rudimentar de como usar este tipo de editor. Na galeria de trabalhos do curso há uma apostila bem humorada sobre o editor `vi`, que foi escrita pelo Guilherme O'Connor de Lungarzo, um dos alunos da primeira turma deste curso. Ela pode ser vista no endereço

`http://latt.if.usp.br/fma215/galeria/guioc/`.

Dê uma olhada nesta apostila e veja se você consegue pelo menos entrar e sair do editor. Se possível, aprenda a fazer algumas modificações simples em um arquivo, pois isto poderá ser útil algum dia.

4 Uma Lista de Comandos Relevantes

`bwbasic` : é um interpretador da linguagem Basic.

`emacs <arquivo>` : edita o arquivo, ou seja, faz uma cópia dele em memória e coloca uma cópia dele na tela, onde você pode examinar e modificar o conteúdo; é um poderoso editor de texto, muito grande e um pouco lento, mas com capacidades imensas e bem fácil de aprender e usar.

As seguintes teclas de comandos (entre muitas, muitas outras) estão disponíveis:

- [↑] `(previous-line)` : movimenta o cursor um caracter para cima.
- [←] `(backward-char)` : movimenta o cursor um caracter para a esquerda.
- [↓] `(next-line)` : movimenta o cursor um caracter para baixo.
- [→] `(forward-char)` : movimenta o cursor um caracter para a direita.
- `^A (beginning-of-line)` : movimenta o cursor para o início da linha.
- `^E (end-of-line)` : movimenta o cursor para o fim da linha.
- `[PageUp] (scroll-down)` : movimenta a janela de visão do texto para cima por uma altura de janela.
- `[PageDown] (scroll-up)` : movimenta a janela de visão do texto para baixo por uma altura de janela.

`[BackSpace]` `(delete-backward-char)` : apaga o caracter que está imediatamente à esquerda do cursor.

`^D` `(delete-char)` : apaga o caracter onde está o cursor.

`^K` `(kill-line)` : apaga todo o texto do cursor até o final da linha; usado já no final da linha, apaga o caracter de controle que representa a quebra de linha.

`^Y` `(yank)` : recoloca no texto a última coisa que foi apagada com `^K` ou `^W`.

`^[Space]` `(set-mark-command)` : define uma marca lógica na posição do cursor.

`^W` `(kill-region)` : apaga todo o texto entre a marca lógica e a atual posição do cursor.

`^/` `(undo)` : desfaz o último pequeno conjunto de modificações que foi feito.

`^S` `(isearch-forward)` : entra em modo de procura incremental para a frente.

`^R` `(isearch-backward)` : entra em modo de procura incremental para trás.

`^G` `(keyboard-quit)` : cancela qualquer comando do editor; às vezes é necessário usar duas vezes seguidas.

`^L` `(recenter)` : limpa e refresca os conteúdos da tela, colocando a posição do cursor no meio dela, verticalmente.

`^X` : é um prefixo para uma coleção de comandos.

`^X^C` `(save-buffers-kill-emacs)` : sai do editor, salvando as modificações dos buffers mediante confirmação para cada um que tenha sido modificado.

`^X^F` `(find-file)` : lê um novo arquivo a ser editado.

`^X^S` `(save-buffer)` : salva o buffer onde você está no momento, caso ele tenha sido modificado, sem sair do editor.

`^X^W` `(write-file)` : escreve o conteúdo do buffer onde você está no momento em um novo arquivo, solicitando o nome deste arquivo.

`^X2` `(split-window-vertically)` : cria uma nova janela, dividindo a janela atual.

`^XO` `(other-window)` : muda o cursor de uma janela para a próxima.

`^X1` `(delete-other-windows)` : elimina todas as janelas exceto aquela onde está o cursor.

`^XO (delete-window)` : elimina a janela onde está o cursor.

`^X^B (list-buffers)` : coloca um novo buffer na janela onde está o cursor.

`g++` : é um compilador para a linguagem C++.

`g77` : é um compilador para a linguagem Fortran 77.

`gcc` : é um compilador para a linguagem C.

`gpc` : é um compilador para a linguagem Pascal.

`gv <arquivo>.ps` : é um aplicativo X11 para a visualização de arquivos no formato PS (PostScript) ou EPS (Encapsulated PostScript); é independente de qual tenha sido a origem do arquivo, interpreta qualquer arquivo neste tipo de formato.

`jed` : é um clone bem completo do `emacs`, muito útil para tarefas gerais de edição.

`latex <arquivo>.tex` : é um compilador que lê e interpreta os comandos de formatação da linguagem LaTeXescrevendo arquivos em formato DVI contendo documentos formatados; trata-se de um poderoso programa de processamento de texto, ou programa de "typesetting", que produz output formatado de alta qualidade.

`make` : é um programa que permite a execução automática de programas que atuam em arquivos, segundo configurações colocadas em um arquivo chamado `Makefile` no diretório de trabalho.

`xdvi <arquivo>.dvi` : é um aplicativo X11 para a visualização de arquivos no formato DVI produzido pelo LaTeX.

`yabasic` : é um interpretador da linguagem Basic.

`zile` : é um clone do `emacs`, muito simples e pequeno.

5 Arquivos de Configuração

O editor `emacs` usa um arquivo de configuração chamado `.emacs` localizado na raiz da conta do usuário. Este arquivo usa uma linguagem do tipo Lisp em sua sintaxe, chamada Elisp, ou seja o Lisp do Emacs. Trata-se de uma linguagem de programação do tipo que se usa normalmente em problemas envolvendo inteligência artificial. Este tipo de linguagem é muito diferente de todas as outras com as quais você talvez já tenha tido contato e é muito pouco intuitiva para quem está habituado com coisas mais corriqueiras.

Muito pouca gente programa em linguagens deste tipo e na prática a única forma de configurar o editor é copiar trechos de código de alguém que já tenha feita a coisa que você quer. Há listas e grupos na Internet dedicadas apenas a fazer isto. Se você quiser tentar aprender algo sobre o Lisp do Emacs, pode tentar ler a documentação introdutória usando

`info emacs-lisp-intro`

Você estará usando um viewer que funciona mais ou menos como o Emacs para ler a documentação, que está no formato `info` e portanto estruturada com algumas propriedades de hipertexto. Se você quiser apenas ver alguns exemplos pode olhar no arquivo `/usr/local/cetc/config.emacs.elisp`, que tem tudo que usamos de forma padronizada por aqui. Se você procurar pelo sistema afora, em contas de colegas e nos diretórios de documentação como `/usr/doc` e `/usr/share/doc`, é provável que ache outras coisas interessantes.