Implementação de uma rede de computadores para ensino de física

Aluno: Rogerio Villela Acquadro
Orientador: Pedro Miguel Raggio Santos

Nível avançado

Introdução:

Nos últimos anos, o uso de recursos computacionais no ramo da educação tem evoluído de maneira espantosa, tanto na melhoria da qualidade quanto na quantidade de ferramentas. Então, por que não explorar melhor estes recursos, tornando os estudos mais palpáveis e agradáveis aos estudantes?

Hoje em dia, a computação tem se tornado uma ferramenta imprescindível para o ensino e pesquisa de física, sendo utilizada em aquisições de dados, tratamentos estatísticos, simulações, entre tantas outras aplicações.

Mas, como conciliar física e computação em tempos de crise, baixas verbas destinadas a compra de recursos computacionais e altos preços de softwares? Isto é possível apenas com a adoção de softwares livres.

Este trabalho procura mostrar como é possível utilizar software livre em uma rede destinada ao ensino de física. Neste relatório, abordaremos os seguintes tópicos:

- *pequeno histórico*, dando um rápido panorama de como foi criado o projeto GNU e a Free Software Foundation.
- *o que é software livre*, explicando a definição de software livre e diferenciando-o do software proprietário.
- *por que vale a pena investir em software livre*, dando exemplos de por que é conveniente o investimento em software livre.
- *sobre o simulador de mecânica*, explicando o funcionamento e a construção do software.
- *sobre a rede de computadores*, explicando o funcionamento e a arquitetura da rede.

Apesar de ter sido desenvolvido durante este trabalho um software simulador de mecânica, a ênfase é, na verdade, a implementação de uma rede de computadores baseada em software livre. O software simulador é apenas um exemplo de aplicação computacional de baixo custo voltado para o ensino de física.

É importante ressaltar que durante estre trabalho não foi usado em momento algum qualquer software proprietário. Desde o desenvolvimento do software até a escrita deste relatório, foram usadas apenas ferramentas livres e de fácil aquisição na Internet.

Pequeno histórico: [9]

O verdadeiro significado do software livre só pode ser entendido se conhecermos bem o momento histórico de quando ele foi criado. Em uma carta no site do projeto GNU, Richard Stallman, fundador do projeto GNU e da Free Software Foundation, fala sobre o início do projeto e sobre todas as dificuldades encontradas naquela época.

Richard Stallman começou a trabalhar no laboratório de Inteligência Artificial do MIT (Massachusetts Institute of Technology) em 1971 e logo tornou-se parte da comunidade de compartilhamento de software. O compartilhamento de software é tão antigo quanto os computadores e funciona da mesma maneira que o compartilhamento de receitas de cozinha.

O laboratório de AI (Artificial Inteligence) usava um sistema operacional chamado ITS (Incompatible Timesharing System) que os hackers do laboratório desenvolveram em linguagem assembler para o Digital PDP-10, um dos maiores computadores da época.

Naquela época ainda não existia o termo “software livre”, mas o espírito era semelhante ao existente hoje em dia. Sempre que alguém de outra universidade ou companhia queria uma cópia do programa, este era totalmente disponibilizado. E sempre que se via alguém usando um software novo, desconhecido e interessante, era totalmente possível e permitido pedir para ver o código fonte, para copiá-lo, modificá-lo e canibalizá-lo para seus programas.

Mas a situação mudou drásticamente no começo dos anos 80 quando a Digital aposentou a série PDP-10. Isto também significou a aposentadoria do software ITS.

Na mesma época, a comunidade do laboratório de AI caiu drasticamente depois que a companhia Symbolics contratou quase todos os hackers. Em 1982, o laboratório de AI comprou um novo PDP-10 da Digital e seus administradores decidiram usar o novo sistema da Digital, proprietário.

Outros modernos computadores da época, como VAX ou 68020, tinham seus próprios sistemas operacionais, mas nenhum livre. Para poder usá-los, o proprietário era obrigado a assinar um termo de acordo com a companhia.

A primeira conseqüência ao assinar este termo era que o usuário não mais poderia ajudar o seu colega. Uma comunidade cooperativa como aquela estava proibida. A regra imposta pelos softwares proprietários era: “Se você compartilhar software com seu colega, você é um pirata. Se você deseja alguma alteração no software, nos procure”. Esta filosofia ia de encontro ao pensamento daquela comunidade.

Em sua carta, Stallman conta que com a maioria dos hackers trocando de emprego, ele deveria tomar uma difícil decisão. A escolha mais fácil era entrar para o mundo do software proprietário, assinando os acordos e prometendo não cooperar com seus colegas. Provavelmente teria ganho mais dinheiro e talvez se divertido escrevendo os programas. Mas isso significaria o fim de sua carreira e ele teria percebido que, mais tarde, olhando para trás, teria trabalhado para criar paredes e dividir os usuários.

A essa altura dos acontecimentos, Stallman já havia tido problemas com acordos de softwares proprietários, quando se recusaram a entregar ao laboratório de AI o código fonte do software que controlava sua impressora (a ausência de certas características neste programa tornava o uso da impressora extremamente frustrante).

A pergunta que ficava no ar era: existe alguma maneira de criar softwares que possam realmente ajudar as pessoas, tornando a comunidade de compartilhamento de software possível novamente?

A resposta era clara: o primeiro passo era construir um sistema operacional. Com um sistema operacional livre o compartilhamento voltaria, a comunidade ganharia novo adeptos.

O sistema operacional que seria criado deveria ser um sistema compatível com Unix, por ser facilmente portável para outras arquiteturas e para que os então usuários de Unix pudessem migrar com certa facilidade. O nome GNU foi escolhido seguindo uma tradição dos hackers, usando um acrônimo recursivo (GNU significa “GNU's Not Unix”).

Um sistema operacional não significa apenas o núcleo (kernel), mas sim todas as ferramentas que o cercam. Na década de 70, todo o sistema operacional incluia processadores de textos, assemblers, compiladores, interpretadores, debuggers, clientes de mail e muito mais. ITS tinha os seus, VMS tinha os seus, Unix tinha os seus. O sistema operacional GNU também deveria ter os seus.

Como desenvolver o projeto GNU era uma tarefa de proporções gigantescas foi decidido adaptar partes de software já previamente desenvolvidos, sempre que possível. Por exemplo, no início do projeto, foi decidido que o Tex seria o principal formatador de textos e alguns anos depois, que se adotaria o sistema X-Window ao invés de escrever um novo sistema.

Por causa desta decisão, o sistema GNU não significa o mesmo que uma coleção de todos os softwares GNU. O sistema GNU inclui também softwares que não são GNU, que são

desenvolvidos por outras pessoas e projetos, mas que poderiam ser utilizados pois são livres.

Em janeiro de 1984, Stallman abandonou seu trabalho no MIT dedicando-se inteiramente ao projeto GNU. Deixar o MIT era necessário pois desta maneira, os diretores do MIT não poderiam interferir no projeto. Caso contrário, o MIT poderia ter requisitado o projeto GNU para si, tornando-o proprietário.

O início do projeto foi marcado por dois softwares muito populares até hoje. O primeiro é o compilador C, o GCC (GNU C Compiler), talvez hoje um dos mais utilizados no mundo. O segundo era o editor de texto GNU Emacs, também muito utilizado pela comunidade hoje em dia.

Como o objetivo do projeto GNU é dar liberdade aos seus usuários e não apenas ser um software popular tornava-se necessário criar termos de distribuição que deixassem claras as liberdades do software. O método é chamado de “copyleft”.

A idéia principal do copyleft é dar permissão para executar, copiar, modificar e distribuir versões modificadas do software, mas não permitir adicionar restrições. As liberdades do software livre (veremos mais adiante) devem ser preservadas e inalteradas.

A implementação específica do copyleft no projeto GNU é o GNU General Public License, ou o GNU GPL. Existem outras licenças “copylefted” para outras circunstâncias, como publicações de manuais, onde a complexidade da GPL não é necessária.

O interesse no Emacs estava aumentando e mais pessoas estavam se juntando ao projeto GNU. Era hora de começar a procurar por investimentos. Então, em 1985, a FSF, uma entidade sem fins lucrativos e voltada para o software livre, foi criada, tomando conta do projeto GNU, desenvolvendo novos softwares, como o interpretador de comandos *bash* (Bourne Again Shell), bibliotecas C, entre outros. Atualmente, a FSF aceita doações de outras instituições, mas a maior parte da receita da entidade é baseada na venda de CD-ROMs, manuais impressos, distribuições, entre outros materiais.

O que é sofware livre?

Existe uma confusão quando se pensa em software livre. É muito comum as pessoas confundirem software livre com software grátis. Mas, diferentemente do que todo mundo pensa, software livre é uma questão de liberdade, e não de preço. Para entender o conceito, você deve pensar no termo “liberdade de expressão”, não em “cerveja grátis”.

Esta confusão é muito freqüente e normal de acontecer por dois motivos: o primeiro é que a maioria dos softwares livres podem ser conseguidos gratuitamente, seja pela Internet ou por outros meios. O segundo é que a palavra em inglês (free software) pode remeter ao significado de “de graça”, e não “livre” como realmente é.

A palavra “livre” se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade:

1. A liberdade de executar o programa, para qualquer propósito
2. A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades. O acesso ao código fonte é um pré-requisito para esta liberdade.
3. A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo.
4. A liberdade de aperfeiçoar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie.

Em outras palavras, ser livre significa que você não tem que pedir ou pagar pela permissão de uso, modificação ou distribuição do programa. A liberdade de utilizar um software significa a liberdade para qualquer tipo de pessoa utilizar o software em qualquer tipo de sistema computacional, para qualquer tipo de trabalho ou atividade, sem que seja necessário comunicar ao desenvolvedor ou a qualquer outra entidade em especial.

A condição *sine qua non* para que o software possa ser considerado livre é a disponibilização do código fonte (mas não a única condição, pois o software pode ter seu código aberto e não ser livre). O acesso ao código fonte respeita as liberdades acima citadas e garante que o software poderá ser estudado, modificado e aperfeiçoado.

Apesar de parecer estranho, certos tipos de regras sobre a maneira de distribuir software livre são aceitáveis, desde que não entrem em conflito com as quatro liberdades principais. Você pode ter pago para receber cópias de software livre, ou você pode ter obtido cópias sem nenhum custo. Mas independente de como você obteve a sua cópia, você sempre tem a liberdade de copiar e modificar o software, ou mesmo de vender novas cópias.

É importante não confundir software livre com "shareware" ou "freeware". O termo "freeware" não possui uma definição clara, mas se aplica normalmente a pacotes que permitem sua redistribuição mas não sua modificação (o seu código fonte não é disponibilizado). Estes pacotes não são software livre e, por favor, não utilize o termo "freeware" ao se referir software livre.

"Shareware" é um software que vem com a permissão de distribuição entre as pessoas, mas diz que qualquer um que use a cópia é obrigado a pagar uma taxa. "Shareware" não é software livre por dois motivos:

1. Para a maior parte dos "sharewares", o código fonte não é disponibilizado. Logo, não é possível fazer modificações no software.
2. "Shareware" não permite fazer uma cópia e instalá-lo sem o pagamento de uma taxa, nem mesmo para fins particulares ou não lucrativos.

Por que vale a pena investir em sofware livre?

Para podermos explicar por que é interessante investir e apostar em software livre, vamos traçar um paralelo entre os dois tipos de software: o software livre e o proprietário.

	Software livre	***Software proprietário***
Abrangência de usuários	grande abrangência devido aos baixos custos de aquisição	pequena abrangência devido aos elevados custos de aquisição
Interação usuário – desenvolvedor	alto nível de interação; é fácil trocar e-mails com os desenvolvedores do software	baixo nível de interação; os usuários tem dificuldades de encontrar meios de se comunicar com os desenvolvedores
Adaptação	fácilmente adaptável devivo a disponibilização do código fonte	a menos que se entre em contato com o desenvolvedor, não é possível adaptar o software
Liberdade de cópias	o usuário é livre para fazer quantas cópias quiser	apenas uma cópia; se for necessário mais cópias, deve-se fazer novo pagamento da licença
Velocidade de desenvolvimento	alta velocidade de desenvolvimento pois mais pessoas estão aptas a interagir e aperfeiçoar o software	baixa velocidade de desenvolvimento, pois o código fonte está em poder de um pequeno grupo de programadores.

Sobre o simulador de mecânica:

O software simulador foi desenvolvido totalmente a partir do zero, mas baseado em softwares já desenvolvidos. Desde suas primeiras versões até as mais recentes, foram usadas apenas ferramentas livres para seu desenvolvimento.

O objetivo do software é oferecer ao aluno de física uma ferramenta de simulação de exercícios que podem ser normalmente encontrados nos livros didáticos, auxiliando o entendimento da disciplina. O software pode ser útil ao educador também, por oferecer uma maneira simples, rápida e prática de demonstração dos exemplos estudados em sala de aula.

Para atender a estes objetivos, o software se propõe a resolver um certo número de problemas normalmente encontrados nas disciplinas F 128 (Física Geral I), F 129 (Física Experimental I), F 228 (Física Geral II), F 229 (Física Experimental II), F 313 (Mecânica Geral) e F 315 (Mecânica Geral II).

Em seu atual estágio de desenvolvimento, o software conta com os quatro tipos de problemas propostos no início do curso: queda de corpos (com e sem atrito viscoso), osciladores com molas (com e sem atrito viscoso), lançamento de projéteis (com e sem atrito viscoso) e pêndulos. A idéia de modularizar o software vem da idéia de tornar mais fácil uma atualização, aumentando a gama de problemas oferecidos. Para isso, cada módulo é independente dos demais; existe um corpo principal do software onde são devidamente encaixados os módulos responsáveis por cada uma das simulações. O programa principal chama os módulos separadamente, dependendo da ação do usuário. Para incrementar o programa, basta escrever o novo módulo e encaixá-lo no programa principal.

A linguagem de programação adotada para o desenvolvimento do software foi a linguagem C, a mais utilizada pelos programadores. Isso foi feito pensando que, no futuro, outros usuários possam realizar alterações ou implementações do programa com certa facilidade.

Por estar sendo escrito em linguagem C, o software torna-se facilmente portável para diferentes sistemas operacionais. Inicialmente, ele está sendo desenvolvido para Linux, mas poderá ser compilado para FreeBSD, Solaris, VMS, e qualquer outro sistema operacional que suporte linguagem C padrão ANSI.

O software é todo executado em modo texto, mas para que o usuário possa ver os gráficos gerados, é necessário que ele possua acesso a alguma interface gráfica (seja ele via SVGALib ou via X-Window). Se o usuário não tiver acesso a este tipo de recurso, o software apenas apresentará os resultados numéricos. Mas, mesmo sem acesso a interfaces gráficas, existe a possibilidade de salvar os gráficos em um arquivo de formato PNG ou PostScript. A possibilidade de salvar gráficos em arquivos é extremamente útil quando deseja-se, por exemplo, ilustrar um relatório ou uma publicação. O formato PNG é suportado por uma gama grande de editores de texto, livre e não livres.

Para apresentar o resultado gráfico do problema, o software gera um script que é lido pelo GNUPlot [11] (responsável pelos gráficos gerados). Neste script estão condidas as funções dos gráficos, os nomes dos eixos X e Y, seus limites, entre outras informações. O software GNUPlot foi escolhido pois é de fácil programação e pode ser encontrado em qualquer distribuição Linux (além de ser livre, é claro).

A apresentação do software para o usuário é feita da maneira mais simples possível. As opções são acessadas a partir de menus, onde cada opção chama um módulo respectivo do software. Ao acessar um módulo, o usuário deverá entrar com dados específicos para o problema. Ao entrar com os dados, ele escolhe o tipo de simulação e o software se encarrega de apresentar soluções numéricas e gráficas.

Desde a versão 0.4 do simulador (lançada no dia 18 de setembro) o programa está disponível na Internet [8] para que qualquer interessado possa baixar o código fonte do programa. Graças a isso, o programa recebeu grande colaboração de Daniel Ome, professor de segundo grau que, mesmo morando na Argentina, pode colaborar muito com a resolução e a implementação de técnicas de programação. Isso só foi possível por que o software foi licenciado sob os termos da GPL, deixando-o livre.

Durante esta fase de desenvolvimento inicial (até a versão 1.0), estará disponível na Internet apenas o código fonte do software. A partir da versão 1.0, estará disponível, além do código fonte, os arquivos executáveis compilados inicialmente apenas para Linux.

Ao término do desenvolvimento, foi escrito um Guia do Usuário, para que os usuários pudessem ter algum documento de orientação do uso do programa. Este guia de referência explica como o problema é simulado, quais as equações envolvidas nos cálculos e quais os resultados retornados ao usuário pelo software.

Sobre a rede de computadores:

A rede de computadores tem como objetivos:

- revitalizar velhos hardwares que estavam destinados ao esquecimento e desuso, vítimas da atual política dos softwares proprietários. O uso de softwares livres adequados torna o uso deste tipo de equipamento uma alternativa viável à crescente demanda dos usuários.
- procurar aproximar o meio universitário do software livre, com a implementação paulatina de softwares livres em seus laboratórios.
- apresentar o GNU/Linux como uma boa alternativa aos softwares proprietários.

O projeto piloto da rede conta, atualmente com três microcomputadores IBM 80486 DX 33 MHz, com 16 Mbytes de memória RAM, nomeados *projeto1*, *projeto2* e *projeto3*. Apenas o *projeto2* possui uma unidade de CD-ROM. Durante o trabalho, foi instalado o sistema operacional Debian GNU/Linux 2.2. Nestes micros é possível executar softwares “leves”, como um navegador de internet, um leitor de e-mails, um editor de texto, entre outros.

Existem hoje em dia inúmeros sistemas operacionais livres, entre eles, FreeBSD, Darwin, Linux, Hurd (em desenvolvimento), entre outros. A escolha pelo Linux foi em base ao enorme sucesso que ele vem apresentando, pelo fato de sua licença estar mais próxima do espírito do software livre (o Linux é distribuido sob os termos da GPL, enquanto FreeBSD, NetBSD e OpenBSD são distribuidos sob os termos da licença BSD) e pela maturidade de seu projeto.

Hoje em dia, grande parte das empresas estão trocando seus servidores (sejam estes baseados em Windows ou em outras arquiteturas) e portando eles para Linux. Além de representar uma queda brutal no custo de aquisição de softwares, pesquisas tem mostrado que os servidores baseados em Linux estão menos vulneráveis a ataques do que servidores rodando Windows, por exemplo.

A maturidade do projeto pode ser um ponto de maior controvérsia. Existe uma tendência natural das pessoas pensarem que sistemas *BSD são mais robustos e confiáveis do que sistemas Linux. Isto até pode ser verdade (embora não concordemos), mas pensamos que a relação custo/benefício favorece muito mais sistemas Linux do que sistemas *BSD.

O Linux surgiu há pouco mais de dez anos e vem tendo avanços surpreendentes. Hoje, o Linux oferece suporte à maioria dos periféricos encontrados. É possível rodar Linux em uma variedade enorme de arquiteturas diferentes, desde PCs i386, passando por Alpha, Sun, Itanium, Apple, sem contar nos processadores ARM, econtrados em palms. Todos estes fatores reunidos nos levaram a escolher o Linux nosso sistema operacional.

Dentro do mundo Linux, exitem inúmeras distribuições. O que difere uma distribuição da outra são os software que acompanham o produto final, personalizações e filosofia de desenvolvimento. A distribuição que se encaixa melhor na filosofia do projeto é o Debian GNU/Linux (veja o *Social Contract* e o *Debian Free Software Guidelines* [10] para maiores informações a respeito da filosofia do Debian).

Como o espaço em disco de cada computador é extremamente limitado (estamos trabalhando com discos de 340 Mbytes), resolvemos espalhar os softwares para que pudéssemos aproveitar melhor os recursos disponíveis. No *projeto2*, foram instalados os softwares AbiWord (editor de textos) e Gnumeric (planilha eletrônica); no *projeto3*, apenas o AbiWord foi instalado.

Apesar da rede ser de baixa performance (composta por micros antigos e ultrapassados) é possível mostrar todo o seu potencial, com a demonstração de alguns recursos:

1. Instalação remota de micros:
   O problema encontrado ao montar a rede foi que apenas um dos micros possuia unidade de CD-ROM e desejava-se instalar o sistema em outros micros pertencentes à mesma rede. Como resolver? O Debian GNU/Linux inclui em seus CD-ROMs imagens de disquetes para este tipo de instalação (quando o micro não possui unidade de CD-ROM). O micro deve ser inicializado por estes disquetes e deve-se apontar o caminho dos arquivos de instalação, a unidade de CD-ROM do outro computador. Isto só é possível se o outro computador possuir suporte a servidor de NFS (Network File System) e estiver compartilhando a unidade de CD-ROM.
2. Concentração de recursos (utilizando NIS e NFS):
   É possível eleger um dos micros como o servidor de nomes (NIS – Name Information Service) e o servidor de arquivos. Isto significa que, neste micro, estariam concentrados todos os nomes dos usuários autorizados a logar na rede e estariam concentrados todos os arquivos pessoais desses usuários (sistema similar ao PDC do Windows NT).
3. Execução remota de softwares:
   Uma vez que nem todos os softwares estão instalados em todos os computadores, pode-se executar um aplicativo em um micro e enviar as informações gráficas para um outro terminal. Isto só é possível graças ao uso do sistema X-Window e ao seu protocolo.
4. Criação de um X-Terminal:
   Um micro sem nenhum aplicativo instalado e de baixa performance é capaz de executar aplicativos pesados, utilizando a técnica acima proposta com alguns avanços. É capaz de fazer com que o computador local apenas receba as informações processadas em um computador remoto. Isto é vendido como solução de informática pela Conectiva S.A., sob o nome de Application Server.

Obs.: Estas duas últimas caraterísticas são completamente impensáveis em ambientes Windows.

Se for adotado o uso de micros mais robustos, a performance do sistema terá uma sensível melhora em relação aos níveis atuais. A adoção de um servidor de alta performance pode significar uma maior disponibilidade de softwares e recursos, podendo extrair ainda mais potencial de uma rede como esta.

Ao implementar uma solução X-Terminal, deseja-se ter uma idéia do potencial do servidor e da quantidade de máquinas que poderão ser atendidas. Para isso, podemos construir uma tabela como a abaixo:

nº de usuários	*CPU*	*Hard Disk*	*Mem. mínima*	*Mem. Recomendada*
até 10	500 MHz	IDE 10 Gb	256 Mb	384 Mb
até 20	700 MHz	IDE 20 Gb	512 Mb	768 Mb
até 30	1,4 GHz	IDE 30 Gb	768 Mb	1 Gb
até 40	1,8 GHz ou 2 x 1,5 GHz	IDE 30 Gb ou SCSI 30 Gb	1 Gb	1,5 Gb
até 50	2,2 GHz ou 2 x 1,8 GHz	IDE 40 Gb ou 2 x SCSI 20 Gb	1,5 Gb	2 Gb

Os dados acima foram estimados levando em conta que cada usuário utilize em média 32 Mbytes de memória e que irá ocupar um espaço máximo em disco de 500 Mbytes. A estimativa para o uso dos processadores é um tanto arbitrária e deve-se levar em conta a simultaneidade dos usuários (os 40 usuários estarão efetivamente usando o computador, todos ao mesmo tempo?).

O futuro do projeto:

O software simulador, por ter sido desenvolvido em C e ser modularizado como já foi explicado, pode ser facilmente expandido por qualquer interessado que queira escrever seu módulo de simulação. A idéia e ir aumentando aos poucos a oferta de módulos do software. Esta expansão pode ser feita por professores, alunos ou qualquer um que esteja interessado em ter o seu módulo simulador agregado ao software.

Mesmo que a pessoa interessada não possua domínio suficiente em programação de computadores, ela poderia fornecer suporte na parte física do problema, apresentando as equações envolvidas e os resultados que espera obter da simulação.

Quando a rede, por se tratar de uma proposta de implementação de Linux em computadores antigos, planejamos instalar o Linux em uma pequena quantidade de computadores disponíveis no instituto para o uso de um grupo pequeno de usuários, para que se possa medir quais as reais necessidades em uma rede deste tipo, e onde o projeto mais necessita de estudos.

Referências:

Bibliografia:

1. J. B. Marion & S. T. Thornton, *Classical Dynamics of particles and systems,* 4a. Edição, Sauders College Publishing, Philadelphia, 1995.
2. K. R. Symon, *Mecânica*, 5a. Edição, Editora Campus, Rio de Janeiro, 1982.
3. R. Johnsonbaugh & M. Kalin, *C for Scientists and Engineers*, Prentice Hall, New Jersey, 1997
4. H. M. Nussenzveig, *Física Básica*, 1a. Edição, Editora Edgar Blücher LTDA., São Paulo, 1981.
5. M. Abramawitz & I. A. Stegun, *Handbook of mathematical functions*, 1a. Edição, Dover Publications, New York, 1970
6. A. Medeiros, *Possibilidades e Limitações das Simulações Computacionais no Ensino da Física*, Revista Brasileira de Ensino de Física, vol. 24, 2002, p. 77
7. T. Williams, *GNUPLOT: an interactive plotting program*

Internet:

8. http://sourceforge.net/projects/mecanica
9. http://www.gnu.org/
10. http://www.debian.org/
11. http://www.gnuplot.info
12. http://www.freshmeat.net

Agradecimentos especiais:

- a Pedro Raggio, pelo apoio desde o início do projeto, cedendo uma sala inteira, micros e colocando o estagiário a disposição
- a Juan Carlos Acquadro, pela ajuda na resolução e construção de alguns módulos.
- a Daniel Ome, pelo apoio no desenvolvimento, solução de bugs e modularização do software.
- a Carlos Gaspar, pelo apoio e suporte na instalação e configuração da rede no Instituto de Física.
- ao diretor do LEI, Flávio Caldas pelo apoio e colaboração no projeto.
- a Marcus Aguiar, Márcio Pudenzzi e Marcelo Guzzo, pelo apoio, interesse e palpites no projeto.
- a todos aqueles que apoiaram o projeto, desde seu início, com sugestões para o andamento e desenvolvimento.

Glossário

- **linguagem de máquina:** é a linguagem que os computadores realmente entendem. Não é composta por frases que fazem sentido a uma pessoa e sim por caracteres estranhos. Por cada sistema (Apple, PC, Alpha, etc.) interpretarem e executarem os comandos de formas diferentes, cada um possui sua própria linguagem de máquina.
- **hackers:** o uso da palavra "hacker" para representar um "invasor" é uma confusão gerada pela grande mídia. Os hacker se recusam a usar este tipo de termo. Utilizam com o significado que realmente ele é: alguém que ama programação e se diverte desenvolvendo novos algoritmos.
- **código fonte:** são as instruções que são digitadas pelo programador e podem ser entendidas e modificadas por qualquer um. Pode ser escrito em diferentes linguagens de programação.
- **sistema operacional**: é o software principal do computador. É ele o responsável por acessos a discos, memória e todos os outros recursos do computador. É sob ele que os demais softwares são executados.
- **Unix:** tipo de sistema operacional (proprietário) criado pelo laboratório da AT&T no início da década de 70, juntamente com a linguagem de programação C. Foi escrito em C para que

pudesse ser facilmente adaptável em diferentes sistemas. Hoje em dia representa uma enorme fatia do mercado de computadores de grandes portes.

- **compilador:** é o software responsável em traduzir as instruções digitadas no código fonte pelo programador em linguagem de máquina. Cada sistema possui seu compilador uma vez que cada sistema possui uma linguagem de máquina diferente.
- **kernel:** é o núcleo do sistema operacional, responsável pela inicalização do computador, o gerenciamento de seus recursos e pela inicalização dos demais serviços.
- **NIS:** Name Information Service, ou Servidor de Nomes. Um serviço inventado pela Sun mas usado por muitas outras empresas. É o serviço responsável por armazenar as informações dos usuários autorizados a logar na rede.
- **NFS:** Network File System, ou Sistema de Arquivos da Rede. É também um serviço criado pela Sun e usado por outras empresas, responsável pelo compartilhamento de arquivos entre computadores Unix dentro de uma mesma rede.