

A linha de comando do Unix e GNU/Linux

A linha de comando do Unix e GNU/Linux

Michael Opdenacker

Free Electrons

<http://free-electrons.com>

Traduzido por

Klayson Sesana Bonatto



Criado com [OpenOffice.org](http://openoffice.org) 2.x

Direito de Cópia



Attribution – ShareAlike 2.0

Você é livre

- para copiar, distribuir, exibir e realizar o seu trabalho
- para criar trabalhos derivados
- para fazer uso comercial deste trabalho

Sob as seguintes condições



Atribuição. Você deve dar crédito ao autor original.



Compartilhar igualmente. Se você alterar, transformar ou basear-se neste trabalho você deverá distribuir o trabalho resultante apenas sob uma licença idêntica a essa.

- Para qualquer reuso ou distribuição você deve deixar claro aos outros os termos de licença deste trabalho.
- Quaisquer dessas condições podem ser renunciadas se você obtiver permissão do detentor do copyright.

Seu uso honesto (fair use) e outros direitos não são de forma alguma afetados pelos acima.

Texto da licença:

<http://creativecommons.org/licenses/by-sa/2.0/legalcode>

© Copyright 2006-2004

Michael Opdenacker

michael@free-electrons.com

Fontes dos documentos, atualizações e traduções:

http://free-electrons.com/training/intro_unix_linux

Correções, sugestões, contribuições e traduções são bem-vindas!



Melhor visualizado com...

Este documento é melhor visualizado com um leitor de arquivos PDF ou com o próprio OpenOffice.org!

- ▶ Utilize as facilidades dos hyperlinks internos ou externos. Não hesite em clicar neles!
- ▶ Encontre páginas rapidamente graças à pesquisa automática;
- ▶ Use miniaturas para navegar no documento de forma rápida;

Se você estiver lendo este documento na sua forma impressa ou no formato HTML, você pode obter uma cópia no formato PDF ou OpenOffice.org no endereço http://free-electrons.com/training/intro_unix_linux!



Memento de comandos mais utilizados



Este é um útil recurso que pode ser utilizado para acompanhar esta apresentação.

Exemplos para os comandos mais úteis são dados em uma única página.

Sugestões de utilização

Cole esta página na sua parede, use-a como wallpaper da área de trabalho do seu desktop, imprima-a nas suas roupas, corte-a e crie marcadores de página...

Faça o download em http://free-electrons.com/training/intro_unix_linux



Conteúdo do treinamento (1)

Shells, sistemas de arquivos e manipulação de arquivos

- ▶ Tudo é um arquivo
- ▶ Estrutura do sistema de arquivos do GNU/Linux
- ▶ Interpretadores de linha de comando
- ▶ Manipulando arquivos e diretórios
- ▶ Exibindo, pesquisando e ordenando arquivos
- ▶ Hard links e links simbólicos
- ▶ Direitos de acesso a arquivos



Conteúdo do treinamento (2)

E/S padrão, redirecionamentos, pipes

- ▶ Entrada e Saída (E/S) padrão, redirecionamentos para arquivos
- ▶ Pipes: redirecionamento da saída padrão para outros comandos
- ▶ Saída de erro padrão



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

Conteúdo do treinamento (3)

Controle de processos

- ▶ Controle total dos processos
- ▶ Executando em segundo plano, suspendendo , resumindo e abortando
- ▶ Lista dos processos ativos
- ▶ "Matando" processos
- ▶ Variáveis de ambiente
- ▶ Variáveis de ambiente PATH
- ▶ Shell aliases, arquivo .bashrc



Conteúdo do treinamento (4)

Miscelânea

- ▶ Editores de texto
- ▶ Utilitários de compressão e arquivamento
- ▶ Impressão de arquivos
- ▶ Comparando arquivos e diretórios
- ▶ Procurando arquivos
- ▶ Obtendo informações sobre usuários



Conteúdo do treinamento (5)

Informações básicas para a administração do sistema

- ▶ Propriedade de arquivos
- ▶ Configuração da rede
- ▶ Sistemas de arquivos: criando e montando

Indo um pouco mais além

- ▶ Obtendo ajuda, acessando as páginas do manual
- ▶ Pesquisando na Internet por recursos auxiliares



GNU/Linux e Software Livre

- ▶ Esta apresentação incluía uma introdução ao Software Livre e Código Aberto: sistemas operacionais, aplicações, principais projetos e regras para o sucesso.
- ▶ Ela agora está disponível como uma apresentação separada:
<http://free-electrons.com/articles/freesw>



Introdução ao Unix e ao GNU/Linux

Sistemas de arquivos Unix



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

Tudo é um arquivo

Quase tudo no Unix é um arquivo!

- ▶ **Arquivos comuns**

- ▶ **Diretórios**

Diretórios são apenas arquivos que listam um conjunto de outros arquivos.

- ▶ **Links simbólicos**

Arquivos que referenciam o nome de outro arquivo.

- ▶ **Dispositivos e periféricos**

Lê e grava em dispositivos como se fossem arquivos comuns.

- ▶ **Pipes**

Usados para concatenar programas

```
cat *.log | grep error
```

- ▶ **Sockets**

Comunicação interprocessos



Nomes de arquivos

Características dos nomes de arquivos presentes desde o início do Unix

- ▶ Diferencia maiúsculas e minúsculas (case sensitive).
- ▶ Não possui um limite óbvio de tamanho.
- ▶ Pode conter qualquer caractere (incluindo espaços, exceto “/”).
O tipo do arquivo é armazenado no arquivo (“números mágicos”).
Extensões dos nomes de arquivo não são obrigatórias e não são interpretadas. Apenas utilizadas para conveniência do usuário.

- ▶ Exemplos de nomes de arquivos:

README
index.htm

.bashrc
index.html

Windows Buglist
index.html.old

Caminhos (paths) de arquivos

Um caminho (*path*) é uma sequência de diretórios aninhados com um arquivo ou diretório no final, separados pelo caractere `/`.

▶ Caminho relativo:

`documents/fun/microsoft_jokes.html`

Relativo ao diretório atual.

▶ Caminho absoluto:

`/home/bill/bugs/crash9402031614568`

▶ `/` : diretório *root* (ou raiz).

É o início dos caminhos absolutos para todos os arquivos do sistema (até mesmo para arquivos existentes em mídias removíveis e compartilhamentos de rede).



Estrutura do sistema de arquivos GNU/Linux (1)

Não é imposta pelo sistema. Pode variar de um sistema para outro, mesmo entre duas instalações do GNU/Linux!

/	Diretório Root
/bin/	Comandos básicos essenciais do sistema
/boot/	Imagens do kernel, initrd e arquivos de configuração
/dev/	Arquivos que representam dispositivos Ex: <code>/dev/hda</code> : primeiro disco rígido IDE do sistema
/etc/	Aquivos de configuração do sistema
/home/	Diretórios dos usuários
/lib/	Bibliotecas compartilhadas básicas do sistema



Estrutura do sistema de arquivos GNU/Linux (2)

<code>/lost+found</code>	Arquivos corrompidos que o sistema tentou recuperar
<code>/mnt/</code>	Sistemas de arquivos montados (<code>/mnt/usbdisk/</code> , <code>/mnt/windows/</code> ...)
<code>/opt/</code>	Programas instalados pelo administrador do sistema. (<code>/usr/local/</code> às vezes usado com esse propósito)
<code>/proc/</code>	Acesso a informações do sistema (<code>/proc/cpuinfo</code> , <code>/proc/version</code> ...)
<code>/root/</code>	Diretório particular do usuário root
<code>/sbin/</code>	Comandos acessíveis apenas pelo administrador.
<code>/sys/</code>	Controles do sistema e dispositivos (frequência da CPU, módulos do kernel, etc.)



Estrutura do sistema de arquivos GNU/Linux (3)

<code>/tmp/</code>	Arquivos temporários
<code>/usr/</code>	Programas dos usuários (não essenciais ao sistema) (<code>/usr/bin/</code> , <code>/usr/lib/</code> , <code>/usr/sbin...</code>)
<code>/usr/local/</code>	Programas instalados pelo administrador do sistema. (usado algumas vezes no lugar de <code>/opt/</code>)
<code>/var/</code>	Dados usados pelo sistema ou programas servidores <code>/var/log/</code> (logs do sistema e programas) <code>/var/spool/mail</code> (e-mails recebidos) <code>/var/spool/lpd</code> (trabalhos de impressão)...



Introdução ao Unix e ao GNU/Linux

Shells e manipulação de arquivos



Free Electrons

Introdução ao Unix e ao GNU/Linux

© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

Interpretadores de linha de comando

- ▶ Shells: programas que executam comandos do usuário
- ▶ São chamados de “shells” (“conchas”) porque eles ocultam dos usuários os detalhes internos do sistema operacional sob a superfície da “concha”.
- ▶ Comandos são informados em um terminal de texto, esteja ele em uma janela de um ambiente gráfico ou em um console no modo caractere.
- ▶ Os resultados são exibidos também no terminal. Não há a necessidade de gráficos.
- ▶ Os Shells permitem a criação de scripts: oferecem todos os recursos necessários para a criação de programas complexos (variáveis, condicionais, iterações...).



Shells mais conhecidos

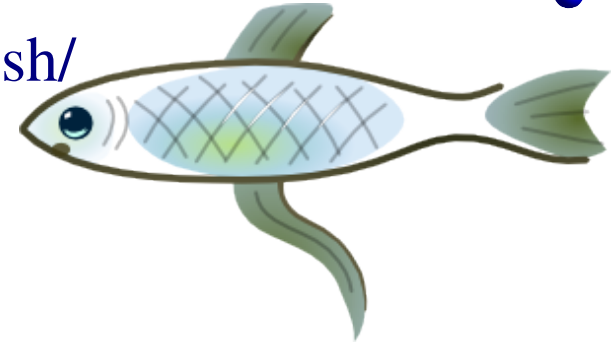
Os mais famosos e populares shells são:

- ▶ **sh**: The Bourne shell (obsoleto)
Tradicional, shell básico encontrado em sistemas Unix, criado por Steve Bourne.
- ▶ **csch**: The C shell (obsoleto)
Shell com uma sintaxe similar à encontrada na linguagem C.
- ▶ **tcsh**: The TC shell (ainda bastante popular)
Implementação compatível com C shell, mas com algumas funções aperfeiçoadas (auto-complete de comandos, histórico e outros).
- ▶ **bash**: The Bourne Again shell (o mais popular)
Um implementação sofisticada do sh com a adição de muitas funcionalidades.



fish: um novo shell muito bacana.

The Friendly Interactive Shell - <http://roo.no-ip.org/fish/>



- ▶ Características padrão: histórico, auto-complete de comandos e arquivos...
- ▶ Novas características: auto-complete de opções de comandos, auto-complete de comandos com descrição, destaque de sintaxe.
- ▶ Facilita a abertura de qualquer arquivo com o programa associado ao tipo do arquivo: comando interno `open`
- ▶ Sintaxe muito mais simples e consistente (mas não é compatível com o padrão POSIX), o que torna mais fácil a criação de shell scripts.

Iniciantes na utilização da linha de comando podem aprender muito mais rápido. Mesmos usuários experientes podem achar esse shell muito conveniente.



Comando ls

Lista os arquivos do diretório atual em ordem alfanumérica, exceto arquivos que iniciam com o caractere “.”.

▶ **ls -a** (all)

Lista todos os arquivos
(inclusive os arquivos .*)

▶ **ls -l** (long)

Listagem longa (tipo, data,
tamanho, proprietário,
permissões)

▶ **ls -t** (time)

Lista os arquivos mais
recentes primeiro

▶ **ls -S** (size)

Lista os maiores arquivos
primeiro

▶ **ls -r** (reverse)

Inverte a ordenação

▶ **ls -ltr** (opções podem ser
combinadas)

Listagem longa, com os
arquivos mais recentes no
final.

Padrões de substituição de nomes de arquivos

Isso é melhor explicado com exemplos!

▶ `ls *txt`

O shell primeiro substitui `*txt` por todos os nomes de arquivos e diretórios que terminam com `txt` (incluindo `.txt`), exceto aqueles que iniciam com “.”, e então executa o comando `ls`.

▶ `ls -d .*`

Lista todos os arquivos e diretório que inicial com “.”.
-d instrui o `ls` a não exibir o conteúdo dos `.*` diretórios.

▶ `cat ?.log`

Exibe todos os arquivos cujos nomes possuem 1 caractere e terminam com “`.log`”.



Diretórios Especiais (1)

`./`

- ▶ O diretório atual. Útil para comandos que levam um diretório como argumento. Também é útil para executar comandos localizados no diretório atual (veremos mais detalhes adiante).
- ▶ Dessa forma, `./readme.txt` e `readme.txt` são equivalentes.

`../`

- ▶ O diretório pai. Está sempre presente no diretório “.” (veja `ls -a`). Única referência ao diretório pai.
- ▶ Uso típico:
`cd ..`



Diretórios Especiais (2)

~/

- ▶ Na verdade não é um diretório especial. Os Shells apenas o substituem pelo diretório home do usuário atual.
- ▶ Não pode ser utilizado na maioria dos programas, já que ele não é um diretório real.

~sydney/

- ▶ Similarmente, é substituído pelos shells pelo diretório home do usuário `sydney`.



Os comandos cd e pwd

▶ `cd <dir>`

Alterna o diretório atual para `<dir>`

▶ `pwd`

Exibe o diretório atual ("diretório de trabalho")



O comando cp

- ▶ `cp <arquivo_origem> <arquivo_destino>`
Copia o arquivo origem para o arquivo destino.
- ▶ `cp arquivo1 arquivo2 arquivo3 ... dir`
Copia os arquivos para o diretório destino (último argumento).
- ▶ `cp -i` (interativo)
Solicita confirmação ao usuário caso o arquivo destino já exista.
- ▶ `cp -r <diretório_origem> <diretório_destino>`
(recursivo)
Copia todo o diretório.



Cópia inteligente de diretórios com rsync

rsync (remote sync) foi projetado para sincronizar diretórios em duas máquinas interligadas por uma conexão de baixa velocidade

- ▶ Apenas copia arquivos que sofreram alterações. Arquivos com o mesmo tamanho são comparados por meio dos seus checksums.
- ▶ Apenas transfere os blocos do arquivo que sofreram alteração!
- ▶ Pode compactar os blocos transferidos.
- ▶ Preserva links simbólicos e as permissões de arquivos: também é muito útil para cópias realizadas na mesma máquina.
- ▶ Pode ser usado via ssh (shell remoto seguro). Muito útil para atualizar o conteúdo de um website, por exemplo.



Exemplos de utilização do rsync (1)

▶ `rsync -a /home/arvin/sd6_agents/ /home/sydney/misc/`

-a: modo archive. Equivalente a `-rlptgoD...` forma fácil de dizer ao programa que você quer recursão e quer preservar quase todos os atributos dos arquivos.

▶ `rsync -Pav --delete /home/steve/ideas/ /home/bill/my_ideas/`

-P: `--partial` (mantém arquivos parcialmente transferidos)
e `--progress` (mostra o progresso durante a transferência)

`--delete`: deleta arquivos no destino caso eles não existam na origem.



Exemplos de utilização do rsync (2)

- ▶ Copiando para uma máquina remota

```
rsync -Pav /home/bill/legal/arguments/ \
bill@www.sco.com:/home/legal/arguments/
```

Será solicitada a senha para o usuário `bill`

- ▶ Copiando de uma máquina remota via ssh

```
rsync -Pav -e ssh \ homer@tank.duff.com/prod/
beer/ \
fridge/homer/beer/
```

Será solicitada a senha ssh para o usuário `homer`



Comandos mv e rm

- ▶ `mv <nome_antigo> <novo_nome>` (move)
Renomeia o arquivo ou diretório passado como parâmetro.
- ▶ `mv -i` (interativo)
Solicita confirmação ao usuário caso o arquivo destino já exista.
- ▶ `rm arquivo1 arquivo2 arquivo3 ...` (remove)
Remove os arquivos passados como parâmetro.
- ▶ `rm -i` (interativo)
Solicita confirmação do usuário antes de excluir o arquivo.
- ▶ `rm -r dir1 dir2 dir3` (recursivo)
Remove recursivamente os diretórios passados como parâmetro.



Criando e removendo diretórios

- ▶ `mkdir dir1 dir2 dir3 ...` (cria diretórios)
Cria diretórios a partir dos nomes passados como parâmetros.
- ▶ `rmdir dir1 dir2 dir3 ...` (remove diretórios)
Remove os diretórios passados como parâmetros.
- ▶ Seguro: apenas funciona quando os diretórios estão vazios.
Alternativa: `rm -r`



Exibindo o conteúdo de arquivos

Existem várias formas de exibir o conteúdo de arquivos:

`cat arquivo1 arquivo2 arquivo3 ...` (concatena)

Concatena e exibe o conteúdo dos arquivos passados como parâmetros.

▶ `more arquivo1 arquivo2 arquivo3 ...`

A cada página, solicita que o usuário pressione uma tecla para continuar. Também permite a localização de palavras.
(comando /)

▶ `less arquivo1 arquivo2 arquivo3 ...`

Faça mais do que o `more` com o `less`.

Não lê todo o arquivo antes de iniciar.

Permite o movimento de retrocesso no arquivo (comando ?)



Os comandos head e tail

▶ `head [-<n>] <arquivo>`

Exibe as primeiras <n> linhas (ou 10, por padrão) do arquivo..

Não é necessário abrir todo o arquivo para isso!

▶ `tail [-<n>] <arquivo>`

Exibe as últimas <n> linhas (ou 10, por padrão) do arquivo..

Não é necessário carregar todo o arquivo na RAM! Muito útil para arquivos grandes.

▶ `tail -f <arquivo> (contínuo)`

Exibe as últimas 10 linhas do arquivo e continua a exibir novas linhas quando elas são acrescentadas ao arquivo..

Muito útil para acompanhar as mudanças em um arquivo de log, por exemplo.

▶ Exemplos

```
head windows_bugs.txt
```

```
tail -f outlook_vulnerabilities.txt
```

O comando grep

▶ `grep <padrão> <arquivos>`

Pesquisa os arquivos passados como parâmetros e exibe as linhas que possuem o padrão.

▶ Exemplo: `grep error *.log`

Exibe todas as linhas que contém a string `error` nos arquivos `*.log`.

▶ `grep -i error *.log`

Mesma situação, porém não diferencia maiúsculas de minúsculas.

▶ `grep -ri error .`

Mesma situação, porém faz a pesquisa recursivamente em todos os arquivos no diretório `.` e nos seus subdiretórios.

▶ `grep -v info *.log`

Exibe todas as linhas dos arquivos `*.log` exceto aquelas que contêm a string `info`.



O comando sort

▶ `sort <arquivo>`

Ordena as linhas do arquivo passado como parâmetro e exibe-as na tela.

▶ `sort -r <arquivo>`

Faz a mesma coisa, porém na ordem reversa.

▶ `sort -ru <arquivo>`

u: unique. Faz a mesma coisa, porém exibe linhas idênticas apenas uma vez, evitando duplicidades.

▶ Veremos mais possibilidades em seguida!



Links simbólicos

Um link simbólico é um arquivo especial que é apenas uma referência para o nome de outro arquivo ou diretório.

Útil para reduzir a utilização de disco e a complexidade quando 2 arquivos têm o mesmo conteúdo.

▶ Exemplo:

```
anakin_skywalker_biography -> darth_vador_biography
```

▶ Como identificar links simbólicos:

▶ `ls -l` exibe “->” e o nome do arquivo “linkado”.

▶ GNU `ls` exibe links com uma cor diferente (azul ciano).



Criando links simbólicos

▶ Para criar um link simbólico (mesma ordem do comando cp):

▶ `ln -s nome_do_arquivo nome_do_link`

▶ Para criar um link para um arquivo em outro diretório, com o mesmo nome:

```
ln -s ../README.txt
```

▶ Para remover um link:

```
rm nome_do_link
```

Obviamente, isso não removerá o arquivo “linkado”!



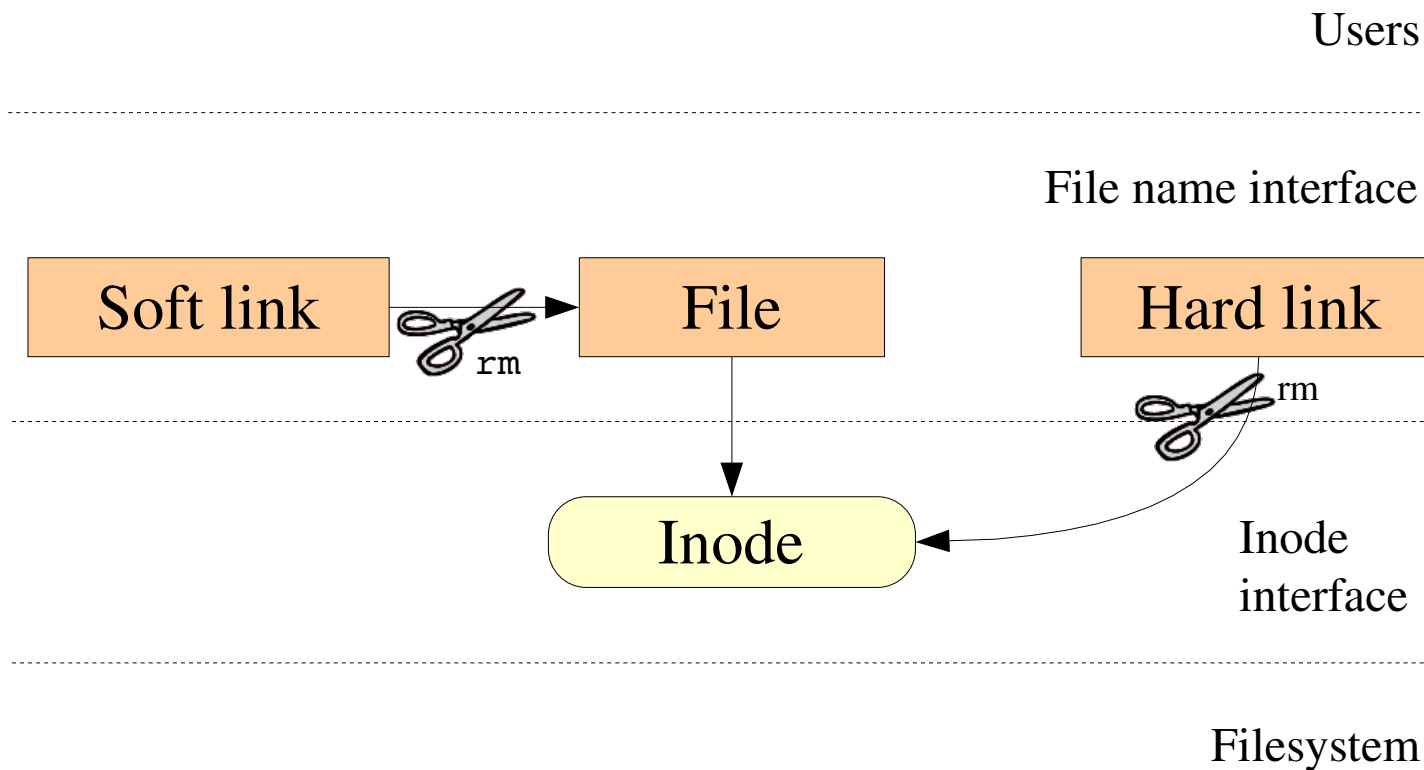
Hard links

- ▶ O comportamento padrão do `ln` é criar *hard links*.
- ▶ Um *hard link* para um arquivo é um arquivo comum com exatamente o mesmo conteúdo físico.
- ▶ Apesar de também economizarem espaço em disco, não há como distinguir um hard link do arquivo original.
- ▶ Se você remover o arquivo original, não há impacto no conteúdo do hard link.
- ▶ O conteúdo apenas é removido quando não há mais arquivos (ou hard links) relacionados a ele.



Nomes de arquivos e inodes

Tornando mais fácil a compreensão de hard links e links simbólicos.



Direitos de acesso a arquivos

Use `ls -l` para verificar os direitos de acesso a um arquivo.

3 tipos de direitos de acesso:

- ▶ Acesso de Leitura - Read (r)
- ▶ Acesso de Gravação – Write (w)
- ▶ Acesso de Execução – Execute (x)

3 tipos de níveis de acesso:

- ▶ Usuário - User (u): para o proprietário do arquivo.
- ▶ Grupo - Group (g): cada arquivo tem um atributo de “grupo”, que corresponde a uma determinada lista de usuários.
- ▶ Outros - Others (o): para todos os outros usuários.

Restrições relacionadas a direitos de acesso

- ▶ x sem r é permitido, mas inútil.
Você tem que ser capaz de ler um arquivo para poder executá-lo.
- ▶ As permissões r e x são necessárias para diretórios: x para entrar, r para listar o conteúdo.
- ▶ Você não pode renomear, remover ou copiar arquivos em um diretório se você não tem a permissão w neste diretório.
- ▶ Se você tiver a permissão w em um diretório, você PODE remover um arquivo mesmo se você não tiver permissão de gravação neste arquivo (lembre-se de que um diretório é apenas um arquivo que descreve uma lista de arquivos). Isso até mesmo permite que você modifique (removendo + criando) um arquivo sem a permissão w .



Exemplos de direitos de acesso

▶ `-rw-r--r--`

Pode ser lido e gravado pelo proprietário do arquivo e apenas lido pelos demais usuários.

▶ `-rw-r-----`

Pode ser lido e gravado pelo proprietário do arquivo e apenas lido pelos usuários que pertencem ao grupo do arquivo.

▶ `drwx-----`

Diretório acessível apenas ao proprietário.

▶ `-----r-x`

Arquivo executável pelos outros usuários do sistema, menos pelos seus amigos ou por você mesmo. Ótima proteção para uma armadilha...



chmod: alterando permissões

▶ `chmod <permissões> <arquivos>`

2 formatos para permissões:

▶ Formato Octal (abc):

$a, b, c = r*4 + w*2 + x$ (r, w, x: booleans)

Exemplo: `chmod 644 <arquivo>`

(rw para u, r para g e o)

▶ Ou o formato simbólico. Facilitando a compreensão com exemplos:

`chmod go+r`: adiciona direito de leitura para o grupo e outros.

`chmod u-w`: remove direito de gravação do usuário (proprietário).

`chmod a-x`: (a: all) remove direito de execução de todos.



Mais sobre o chmod (1)

```
chmod -R a+rX linux/
```

Torna o diretório `linux` e tudo dentro dele disponível para todos!

- ▶ R: aplica as mudanças recursivamente.
- ▶ X: similar ao x, mas apenas para diretórios e arquivos que já são executáveis.

Muito útil para realizar acesso recursivo a diretórios, sem adicionar direito de execução a todos os arquivos.



Mais sobre o chmod (2)

```
chmod a+t /tmp
```

- ▶ **t**: (sticky). Permissão especial para diretórios, permitindo que apenas o proprietário de um diretório ou arquivo possa deletá-los.
- ▶ Útil para diretórios com acesso de gravação para todos os usuários, como o `/tmp`.
- ▶ Exibido pelo comando `ls -l` com um caractere **t**.



Introdução ao Unix e ao GNU/Linux

E/S padrão, redirecionamentos, pipes



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

Saída padrão

Mais informações sobre a saída de comandos.

- ▶ Todos os comandos que exibem texto no seu terminal o fazem por meio da gravação de informações na sua saída padrão.
- ▶ A saída padrão pode ser gravada em (redirecionada para) um arquivo usando o símbolo >
- ▶ A saída padrão pode ser adicionada ao final de um arquivo existente usando o símbolo >>



Exemplos de redirecionamento da saída padrão

- ▶ `ls ~saddam/* > ~gwb/weapons_mass_destruction.txt`
- ▶ `cat obiwan_kenobi.txt > starwars_biographies.txt`
`cat han_solo.txt >> starwars_biographies.txt`
- ▶ `echo "README: No such file or directory" > README`
Forma prática de criar um arquivo sem um editor de textos.
Neste caso também é uma infame piada do mundo Unix.



Entrada padrão

Mais informações sobre a entrada de comandos.

- ▶ Diversos comandos, quando não recebem argumentos, obtêm sua entrada a partir da *entrada padrão*.

- ▶ `sort`

`windows`

`linux`

`[Ctrl][D]`

`linux`

`windows`

O comando `sort` obtém sua entrada a partir da entrada padrão: neste caso, aquilo que você digita no terminal (finalizado com `[Ctrl][D]`).

- ▶ `sort < participants.txt`

A entrada padrão do comando `sort` é obtida a partir do arquivo `participants.txt`.

Pipes

- ▶ Os pipes são muito úteis para redirecionar a saída padrão de um comando para a entrada padrão de outro comando.
- ▶ Exemplos
 - ▶ `cat *.log | grep -i error | sort`
 - ▶ `grep -ri error . | grep -v "ignored" | sort -u > serious_errors.log`
 - ▶ `cat /home/*/homework.txt | grep mark | more`
- ▶ Essa é uma das mais poderosas características dos shells do Unix!



O comando tee

`tee [-a] arquivo`

▶ O comando `tee` pode ser usado para enviar a saída padrão para a tela e para um arquivo simultaneamente.

▶ `make | tee build.log`

Executa o comando `make` e armazena a sua saída no arquivo `build.log`

▶ `make install | tee -a build.log`

Executa o comando `make` e adiciona a sua saída no arquivo `build.log`



Erro padrão

- ▶ Mensagens de erro são geralmente exibidas (se o programa for bem escrito) na saída de *erro padrão* e não na saída padrão.
- ▶ A saída de erro padrão pode ser redirecionada usando `2>` ou `2>>`
- ▶ Exemplo:

```
cat f1 f2 nofile > newfile 2> errfile
```
- ▶ Nota: 1 é o descritor da saída padrão, então `1>` é equivalente a `>`
- ▶ É possível redirecionar tanto a saída padrão quanto a saída de erro padrão para o mesmo arquivo usando `&>`

```
cat f1 f2 nofile &> wholefile
```



O comando yes

Útil para preencher a entrada padrão sempre com uma mesma string.

► `yes <string> | <comando>`

Preenche a entrada padrão do `<comando>` com `<string>` (y por padrão).

► Exemplos

```
yes | rm -r dir/
```

```
bank> yes no | credit_applicant  
("bank>" representa o prompt)
```

► `yes "" | make oldconfig`

(equivalente a pressionar **Enter** para aceitar todas as configurações padrão)



Dispositivos especiais (1)

Eles se parecem com arquivos reais, mas

▶ /dev/null

O detonador de dados! Descarta todos os dados gravados nesse arquivo. Útil para eliminar saída não desejada, tipicamente informação de log:

```
mplayer black_adder_4th.avi &> /dev/null
```

▶ /dev/zero

A leitura desse arquivo sempre retorna caracteres \0

Útil para criar um arquivo preenchido com zeros:

```
dd if=/dev/zero of=disk.img bs=1k count=2048
```



Dispositivos especiais (2)

▶ `/dev/random`

Retorna bytes aleatórios quando lido. Usado principalmente por programas de criptografia. Usa interrupções obtidas a partir de alguns drivers de dispositivos como fonte da sua verdadeira aleatoriedade (“entropia”). Leituras podem ser bloqueadas até que a entropia suficiente seja obtida.

▶ `/dev/urandom`

Utilizado por programas onde números pseudo aleatórios são suficientes. Sempre gera bytes randômicos, mesmo se não houver entropia suficiente disponível (nesses casos é possível, apesar da dificuldade, prever futuras seqüências de bytes a partir de seqüências anteriormente geradas).

Veja `man random` para detalhes.



Introdução ao Unix e ao GNU/Linux

Gerenciamento de processos



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

Controle total de processos

- ▶ Desde o início, o Unix suporta multitarefa preemptiva.
- ▶ Possibilidade de executar várias atividades em paralelo e abortá-las mesmo se elas corromperem seu próprio estado e dados.
- ▶ Possibilidade de escolher quais programas executar.
- ▶ Possibilidade de escolher qual entrada será dada aos seus programas e para onde irá sua saída.



Processos

“Tudo no Unix é um arquivo.”

“Tudo no Unix que não é um arquivo é um processo.”

Processos

- ▶ Instâncias de programas em execução.
- ▶ Várias instâncias de um mesmo programa podem ser executadas ao mesmo tempo.
- ▶ Dados associados a processos:
Arquivos abertos, memória alocada, pilha de execução, identificador do processo, processo pai, prioridade, estado, etc.



Executando programas em segundo plano

Mesma forma de utilização em todos os shells.

► Utilidade

- Para comandos cuja saída pode ser examinada mais tarde, especialmente para aqueles que demoram muito tempo.
- Para iniciar aplicações gráficas a partir da linha de comando e continuar usando o terminal.
- Iniciando um programa: adicione & no final da linha:

```
find_prince_charming --cute --clever --rich &
```



Controlando programas em segundo plano

▶ jobs

Retorna a lista de programas em segundo plano do mesmo shell em que foram executados.

```
[1]-  Running ~/bin/find_meaning_of_life --without-god &  
[2]+  Running make mistakes &
```

▶ fg

fg %<n>

Coloca em primeiro plano o último n comando que havia sido colocado em segundo plano.

▶ Movendo o comando corrente para segundo plano, suspendendo-o:

[Ctrl] Z

bg

▶ kill %<n>

Aborta o n último comando colocado em segundo plano.

Exemplos

```
> jobs
[1]-  Running ~/bin/find_meaning_of_life --without-god &
[2]+  Running make mistakes &

> fg
make mistakes

> [Ctrl] Z
[2]+  Stopped make mistakes

> bg
[2]+ make mistakes &

> kill %1
[1]+  Terminated ~/bin/find_meaning_of_life --without-god
```



Listando todos os processos

... que foram iniciados a partir do shell, de scripts ou outros processos.

▶ `ps -ux`

Lista todos os processos pertencentes ao usuário atual.

▶ `ps -aux` (Obs: `ps -edf` em sistemas System V)

Lista todos os processos do sistema.

```
▶ ps -aux | grep bart | grep bash
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
bart      3039  0.0  0.2   5916  1380 pts/2    S      14:35   0:00 /bin/bash
bart      3134  0.0  0.2   5388  1380 pts/3    S      14:36   0:00 /bin/bash
bart      3190  0.0  0.2   6368  1360 pts/4    S      14:37   0:00 /bin/bash
bart      3416  0.0  0.0      0     0 pts/2    RW     15:07   0:00 [bash]
```

▶ PID: Identificador do processo.

VSZ: Tamanho virtual do processo (código + dados + pilha)

RSS: Tamanho residente do processo: número de KB atualmente na RAM

TTY: Terminal

STAT: Status: R (Runnable), S (Sleep), W (paging), Z (Zombie)...

Atividade dos processos

- ▶ **top** — Exibe os processos mais importantes, ordenados por percentual de utilização da CPU.

```
top - 15:44:33 up 1:11, 5 users, load average: 0.98, 0.61, 0.59
Tasks: 81 total, 5 running, 76 sleeping, 0 stopped, 0 zombie
Cpu(s): 92.7% us, 5.3% sy, 0.0% ni, 0.0% id, 1.7% wa, 0.3% hi, 0.0% si
Mem: 515344k total, 512384k used, 2960k free, 20464k buffers
Swap: 1044184k total, 0k used, 1044184k free, 277660k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3809	jdoe	25	0	6256	3932	1312	R	93.8	0.8	0:21.49	bunzip2
2769	root	16	0	157m	80m	90m	R	2.7	16.0	5:21.01	X
3006	jdoe	15	0	30928	15m	27m	S	0.3	3.0	0:22.40	kdeinit
3008	jdoe	16	0	5624	892	4468	S	0.3	0.2	0:06.59	autorun
3810	jdoe	16	0	2892	916	1620	R	0.3	0.2	0:00.06	top

- ▶ Você pode alterar a ordem teclando...
M: Utilização da Memória, P: %CPU, T: Tempo.
- ▶ Você pode “matar” (kill) um processo digitando **k** e o número de identificação do processo (process id).

Matando processos (1)

► `kill <pids>`

Envia um sinal de término para os processos passados como parâmetro. Permite que os processos salvem os seus dados e terminem por si mesmos. Deve ser usado primeiro. Exemplo:

```
kill 3039 3134 3190 3416
```

► `kill -9 <pids>`

Envia um sinal de término imediato. O próprio sistema termina os processos. Útil quando um processo está realmente travado (não responde ao `kill -1`).

► `kill -9 -1`

Mata todos os processos do usuário atual. `-1`: significa todos os processos.



Matando processos (2)

▶ `killall [-<signal>] <command>`

Mata todos os processos executando <command>. Exemplo:
`killall bash`

▶ `xkill`

Permite matar uma aplicação gráfica clicando nela! Muito rápido! Conveniente quando você não sabe o nome do comando da aplicação. (Adendo do tradutor: estando no ambiente gráfico, tente também a combinação Ctrl+Alt+Esc)



Recuperando um ambiente gráfico travado

- ▶ Se sua sessão gráfica estiver travada e você não conseguir mais digitar no seu terminal, não reinicie o computador!
- ▶ É bem provável que o seu sistema ainda esteja funcionando. Tente acessar uma console baseada em texto pressionando as teclas `[Ctrl][Alt][F1]`.
(ou `[F2]`, `[F3]` para mais consoles baseadas em texto)
- ▶ Estando no console você pode tentar matar a aplicação travada.
- ▶ Feito isso, você pode voltar à sessão gráfica pressionando `[Ctrl][Alt][F5]` ou `[Ctrl][Alt][F7]` (dependendo da sua distribuição)
- ▶ Se você não conseguir identificar o programa travado você pode matar todos os seus processos: `kill -9 -1`
Você será levado de volta à tela de login.



Comandos em sequência

- ▶ Você pode digitar o próximo comando no seu terminal mesmo quando o comando atual não tiver terminado.
- ▶ Você pode separar comandos com o símbolo “;” :
`echo "I love thee"; sleep 10; echo " not"`
- ▶ Condicionais: use `||` (or) ou `&&` (and):
`cat God || echo "Sorry, God doesn't exist"`
Executa o comando `echo` apenas se o primeiro comando falhar.

`ls ~sd6 && cat ~sd6/* > ~sydney/recipes.txt`

Apenas executa o comando `cat` se o comando `ls` for executado com sucesso (significa permissão de leitura).



Aspas (1)

Aspas duplas (") podem ser utilizadas para evitar que o shell interprete espaços como separadores de argumentos, bem como evitar a expansão de padrões de nomes de arquivos.

- > echo "Olá Mundo"
Olá Mundo
- > echo "Você está logado com o usuário \$USER"
Você está logado com o usuário as bgates
- > echo *.log
find_prince_charming.log cosmetic_buys.log
- > echo "*.log"
*.log



Aspas (2)

Aspas simples oferecem uma funcionalidade similar, mas o que está entre aspas simples nunca é substituído.

```
> echo 'Você está logado com o usuário $USER'  
Você está logado com o usuário $USER
```

Crases (`) podem ser utilizadas para chamar um comando “dentro” de outro

```
> cd /lib/modules/`uname -r`; pwd  
/lib/modules/2.6.9-1.6_FC2
```

Crases podem ser utilizadas dentro de aspas duplas

```
> echo "Você está usando Linux `uname -r`"  
Você está usando Linux 2.6.9-1.6_FC2
```

Medindo o tempo gasto

```
▶ time find_expensive_housing --near  
<...command output...>  
real      0m2.304s (tempo gasto)  
user      0m0.449s (tempo de CPU executando o código do  
programa)  
sys       0m0.106s (tempo de CPU executando system calls)
```

$real = user + sys + espera$

$espera = \text{Tempo de espera por I/O} + \text{tempo ocioso (processador executando outros processos)}$



Variáveis de Ambiente

- ▶ Shells permitem que o usuário defina *variáveis*.
Elas podem ser reutilizadas em comandos do shell.
Convenção: nomes em letras *minúsculas*.
- ▶ Você também pode definir *variáveis de ambiente*:
variáveis que também são visíveis por scripts ou executáveis chamados a partir do shell.
Convenção: nomes em letras *maiúsculas*.
- ▶ `env`
Lista todas as variáveis de ambiente e os seus valores.



Exemplos de variáveis do Shell

Variáveis do Shell (bash)

- ▶ `projdir=/home/marshall/coolstuff`
`ls -la $projdir; cd $projdir`

Variáveis de ambiente (bash)

- ▶ `cd $HOME`

- ▶ `export DEBUG=1`
`./find_extraterrestrial_life`
(exibe informações de debug se a variável DEBUG for setada)



Principais variáveis de ambiente padrão

Usados por várias aplicações!

- ▶ **LD_LIBRARY_PATH**
Caminho de pesquisa das bibliotecas compartilhadas
- ▶ **DISPLAY**
Identificação da tela que exibirá aplicações X (gráficas)
- ▶ **EDITOR**
Editor padrão (vi, emacs...)
- ▶ **HOME**
Diretório home do usuário corrente
- ▶ **HOSTNAME**
Nome da máquina local
- ▶ **MANPATH**
Caminho de pesquisa das páginas de manual
- ▶ **PATH**
Caminho de pesquisa dos comandos
- ▶ **PRINTER**
Nome da impressora default
- ▶ **SHELL**
Nome do shell corrente
- ▶ **TERM**
Nome/modo do terminal corrente
- ▶ **USER**
Nome do usuário corrente



Variáveis de ambiente PATH

▶ PATH

Especifica a ordem de pesquisa de comandos do shell.

```
/
home/adox/bin:/usr/local/bin:/usr/kerberos/bin:/usr
/bin:/bin:/usr/X11R6/bin:/bin:/usr/bin
```

▶ LD_LIBRARY_PATH

Especifica a ordem de pesquisa de bibliotecas compartilhadas (código binário de bibliotecas compartilhadas por aplicações, como a biblioteca C) para o comando ld (Linker GNU)

```
/usr/local/lib:/usr/lib:/lib:/usr/X11R6/lib
```

▶ MANPATH

Especifica a ordem de pesquisa para as páginas do manual

```
/usr/local/man:/usr/share/man
```



Cuidados com o uso da variável PATH

É altamente recomendado não ter o diretório “.” na sua variável de ambiente PATH, principalmente como um dos primeiros diretórios:

- ▶ Um cracker pode, por exemplo, colocar um arquivo `ls` malicioso no seu diretório. Ele será executado quando você digitar `ls` neste diretório e poderá realizar alguma ação indesejada aos seus dados.
- ▶ Se você tem um arquivo executável chamado `test` em um diretório, ele irá se sobrepor ao programa `test` default e alguns scripts não funcionarão mais da forma correta.
- ▶ Cada vez que você executar um `cd` em um diretório o shell gastará tempo atualizando a sua lista de comandos disponíveis.

Execute os comandos do diretório corrente da seguinte forma:: `./test`

Alias

Os Shells permitem que você defina apelidos (*aliases*): atalhos para comandos que você usa frequentemente.

Exemplos

▶ `alias ls='ls -la'`

Útil para sempre executar os comandos com os seus argumentos preferidos.

▶ `alias rm='rm -i'`

Útil para forçar o comando `rm` a pedir uma confirmação a cada arquivo deletado.

▶ `alias frd='find_rambaldi_device --asap --risky'`

Útil para substituir comandos freqüentes e muito longos.

▶ `alias cia='. /home/sydney/env/cia.sh'`

Útil para configurar o ambiente de forma rápida.

(`.` é um comando do shell que executa o conteúdo de um shell script)



O comando which

Antes de executar um comando, o `which` diz onde ele pode ser encontrado.

- ▶ `bash> which ls`
`alias ls='ls --color=tty'`
`/bin/ls`
- ▶ `tcsh> which ls`
`ls:            aliased to ls --color=tty`
- ▶ `bash> which alias`
`/usr/bin/which: no alias in`
`(/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin)`
- ▶ `tcsh> which alias`
`alias: shell built-in command.`



Arquivo ~/.bashrc

- ▶ ~/.bashrc

Shell script lido cada vez que um shell `bash` é iniciado.

- ▶ Você pode usar esse arquivo para definir:

- ▶ Suas variáveis de ambiente default (`PATH`, `EDITOR`...)

- ▶ Seus aliases

- ▶ Seu prompt (veja o manual do `bash` para detalhes)

- ▶ Uma mensagem de boas-vindas.

Introdução ao Unix e ao GNU/Linux

Utilitários Diversos



Edição de Comandos

- ▶ Você pode usar as setas da esquerda e da direita para mover o cursor no comando atual.
- ▶ Você pode usar `[Ctrl][a]` para ir ao início da linha de comando e `[Ctrl][e]` para ir ao final da linha de comando.
- ▶ Você pode usar as setas “para cima” e “para baixo” para selecionar os comandos recentemente digitados.



Históricos de Comandos (1)

- ▶ `history`

Exibe os últimos comandos executados e os seus respectivos números. Você pode copiar e colar as strings dos comandos.

- ▶ Você pode reexecutar o último comando:

!!

- ▶ Você pode reexecutar um comando pelo seu número:

!1003

- ▶ Você pode reexecutar o último comando cujas primeiras letras “casam” com uma determinada string:

!cat



Históricos de Comandos (2)

- ▶ Você pode fazer substituições no último comando:
`^more^less`
- ▶ Você pode executar outro comando com os mesmos argumentos:
`more !*`



Editores de Texto

Editores de texto gráficos:

Atendem a maioria das necessidades

- ▶ nedit
- ▶ Emacs, Xemacs

Editores de “texto-puro:

Algumas vezes necessários para administradores do sistema e excelentes para usuários avançados:

- ▶ vi
- ▶ nano



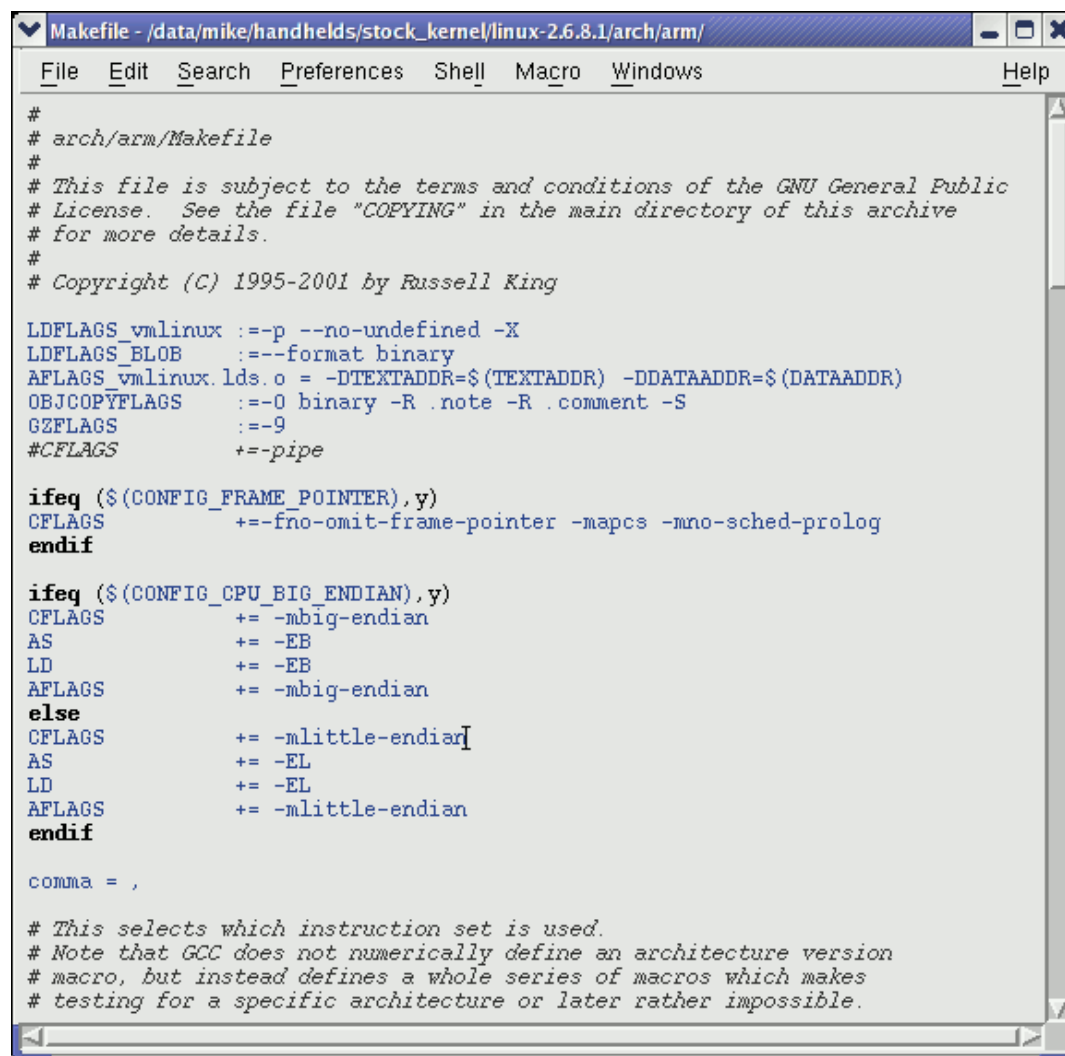
O editor de texto nedit

<http://www.nedit.org/>

- O melhor editor de texto para aqueles que não são experts no `vi` ou `emacs`.
- ▶ Principais funcionalidades:
 - É muito fácil selecionar e mover textos.
 - Destaque de sintaxe para a maioria de linguagens e formatos. Pode ser configurado para destacar determinadas palavras (de erro ou aviso) dos seus arquivos de log.
 - Fácil de customizar por meio de menus.
- ▶ Não é instalado por default na maioria das distribuições.



nedit screenshot



```
#
# arch/arm/Makefile
#
# This file is subject to the terms and conditions of the GNU General Public
# License. See the file "COPYING" in the main directory of this archive
# for more details.
#
# Copyright (C) 1995-2001 by Russell King

LDFLAGS_vmlinux :=-p --no-undefined -X
LDFLAGS_BLOB :=--format binary
AFLAGS_vmlinux.lds.o = -DTEXTADDR=$(TEXTADDR) -DDATAADDR=$(DATAADDR)
OBJCOPYFLAGS :=-O binary -R .note -R .comment -S
GZFLAGS :=-9
#CFLAGS +=-pipe

ifeq ($(CONFIG_FRAME_POINTER),y)
CFLAGS +=-fno-omit-frame-pointer -mapcs -mno-sched-prolog
endif

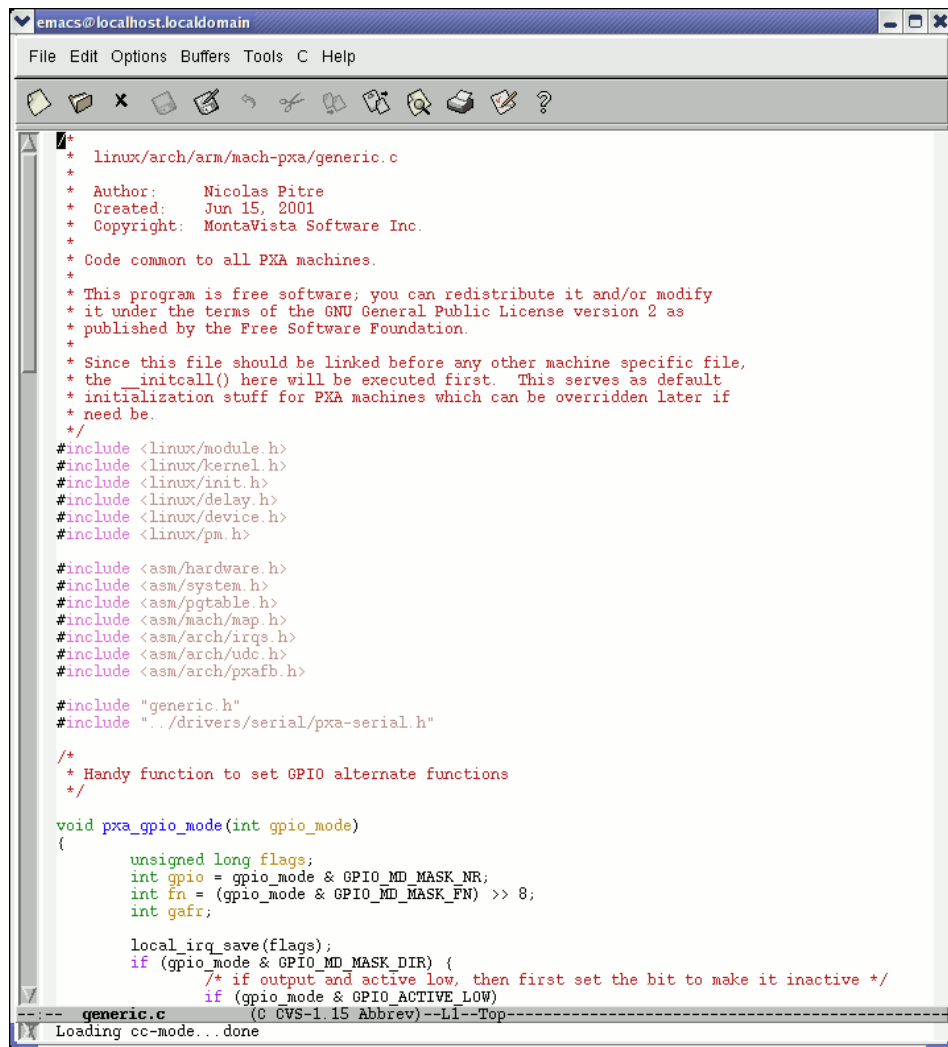
ifeq ($(CONFIG_CPU_BIG_ENDIAN),y)
CFLAGS +=-mbig-endian
AS +=-EB
LD +=-EB
AFLAGS +=-mbig-endian
else
CFLAGS +=-mlittle-endian
AS +=-EL
LD +=-EL
AFLAGS +=-mlittle-endian
endif

comma = ,

# This selects which instruction set is used.
# Note that GCC does not numerically define an architecture version
# macro, but instead defines a whole series of macros which makes
# testing for a specific architecture or later rather impossible.
```



Emacs / Xemacs



```
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help

/*
 * linux/arch/arm/mach-pxa/generic.c
 *
 * Author:      Nicolas Pitre
 * Created:     Jun 15, 2001
 * Copyright:   MontaVista Software Inc.
 *
 * Code common to all PXA machines.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 *
 * Since this file should be linked before any other machine specific file,
 * the __initcall() here will be executed first. This serves as default
 * initialization stuff for PXA machines which can be overridden later if
 * need be.
 */
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/device.h>
#include <linux/pm.h>

#include <asm/hardware.h>
#include <asm/system.h>
#include <asm/pgtable.h>
#include <asm/mach/map.h>
#include <asm/arch/irqs.h>
#include <asm/arch/udc.h>
#include <asm/arch/pxafb.h>

#include "generic.h"
#include "../drivers/serial/pxa-serial.h"

/*
 * Handy function to set GPIO alternate functions
 */

void pxa_gpio_mode(int gpio_mode)
{
    unsigned long flags;
    int gpio = gpio_mode & GPIO_MD_MASK_NR;
    int fn = (gpio_mode & GPIO_MD_MASK_FN) >> 8;
    int gafr;

    local_irq_save(flags);
    if (gpio_mode & GPIO_MD_MASK_DIR) {
        /* if output and active low, then first set the bit to make it inactive */
        if (gpio_mode & GPIO_ACTIVE_LOW)
            generic.c
            (C CVS-1.15 Abbrev)--LI--Top
Loading cc-mode... done
```

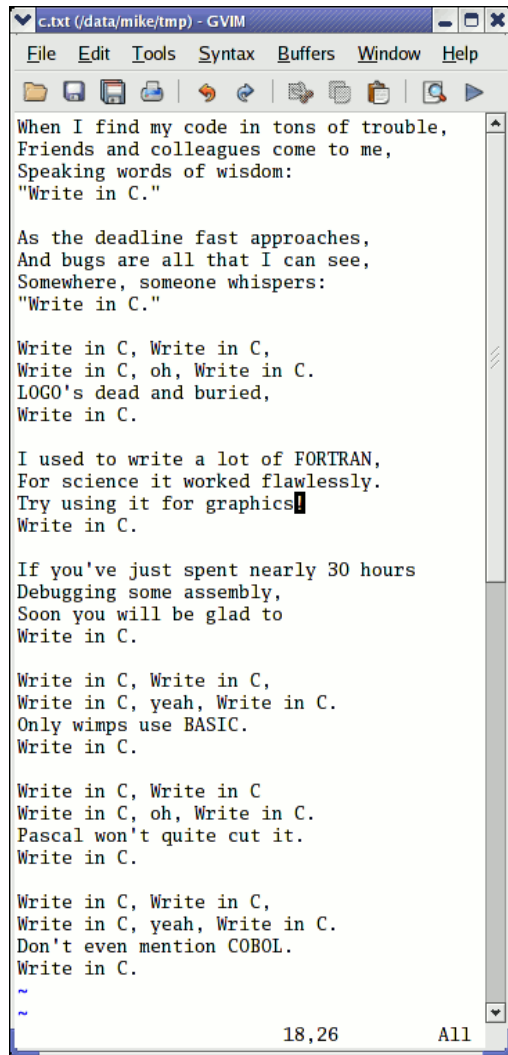
- Emacs e Xemacs são muito parecidos.
- Funcionalidades extremamente poderosas de edição de texto.
- Excelente para usuários avançados.
- Menos ergonômico do que nedit.
- Atalhos não padronizados.
- Muito mais do que um editor de texto (jogos, e-mail, shell, browser)
- Alguns comandos poderosos têm de ser aprendidos.

Editor modo texto disponível em todos os sistemas Unix.
Criado antes do surgimento do mouse.

- Difícil de aprender por iniciantes acostumados com editores de texto gráficos.
- Muito produtivo para usuários avançados.
- Muitas vezes não pode ser substituído na edição de arquivos durante a administração de sistemas ou em sistemas embarcados, quando você só tem um console texto.



vim - vi improved



```
c.txt (/data/mike/tmp) - GVIM
File Edit Tools Syntax Buffers Window Help

When I find my code in tons of trouble,
Friends and colleagues come to me,
Speaking words of wisdom:
"Write in C."

As the deadline fast approaches,
And bugs are all that I can see,
Somewhere, someone whispers:
"Write in C."

Write in C, Write in C,
Write in C, oh, Write in C.
LOGO's dead and buried,
Write in C.

I used to write a lot of FORTRAN,
For science it worked flawlessly.
Try using it for graphics
Write in C.

If you've just spent nearly 30 hours
Debugging some assembly,
Soon you will be glad to
Write in C.

Write in C, Write in C,
Write in C, yeah, Write in C.
Only wimps use BASIC.
Write in C.

Write in C, Write in C
Write in C, oh, Write in C.
Pascal won't quite cut it.
Write in C.

Write in C, Write in C,
Write in C, yeah, Write in C.
Don't even mention COBOL.
Write in C.

~
~

18,26 All
```

- ▶ Implementação do vi disponível na maioria dos sistemas GNU/Linux.
- ▶ Implementa várias funcionalidades disponíveis em editores modernos: destaque de sintaxe, histórico de comandos, help, undo (desfazer) ilimitado e muito, muito mais.
- ▶ Exemplo de uma funcionalidade interessante: pode abrir arquivos compactados diretamente.
- ▶ Vem com interface gráfica GTK (gvim)
- ▶ Infelizmente não é software livre (por causa de uma pequena restrição na liberdade de fazer mudanças).



Comandos básicos do vi



Apesar do **vi** ser extremamente poderoso, seus 30 principais comandos são facilmente aprendidos e suficientes para 99% das necessidades de todos.

Você também pode ler o tutorial rápido executando **vimtutor**.

Obtenha o nosso guia de referência rápida do vi em:
http://free-electrons.com/training/intro_unix_linux

GNU nano

<http://www.nano-editor.org/>

- ▶ Outro pequeno editor de texto-puro;
- ▶ Um clone avançado do editor pico (editor não livre do Pine);
- ▶ Amigável e fácil de aprender por iniciantes graças aos atalhos de teclado presentes na tela;
- ▶ Disponível em pacotes binários para várias plataformas;
- ▶ Uma alternativa ao vi em sistemas embarcados. Entretanto, não está disponível como um built-in do busybox*.

* <https://launchpad.net/products/busybox>

Screenshot do GNU nano

```
GNU nano 1.2.3          File: fortune.txt

The herd instinct among economists makes sheep look like independent thinkers.

Klingon phaser attack from front!!!!
100% Damage to life support!!!

Spock: The odds of surviving another attack are 13562190123 to 1, Captain.

Quantum Mechanics is God's version of "Trust me."

I'm a soldier, not a diplomat.  I can only tell the truth.
      -- Kirk, "Errand of Mercy", stardate 3198.9

Did you hear that there's a group of South American Indians that worship
the number zero?

Is nothing sacred?

They are called computers simply because computation is the only significant
job that has so far been given to them.

As far as the laws of mathematics refer to reality, they are not
certain, and as far as they are certain, they do not refer to reality.
      -- Albert Einstein

Tact, n.:
      The unsaid part of what you're thinking.

Support bacteria -- it's the only culture some people have!

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Txt  ^T To Spell
```



Miscelânea

Compactação e Arquivamento



Medindo o uso do disco (1)

Cuidado: é diferente do tamanho do arquivo!

- ▶ `du -h <arquivo>` (espaço ocupado no disco)
 - h: retorna o espaço ocupado no disco pelo arquivo passado como parâmetro, em um formato compreensível por humanos: K (kilobytes), M (megabytes) ou G (gigabytes). Sem o –h, du retorna o número de blocos de disco ocupados pelo arquivo (o que é difícil de compreender). Observe que a opção –h apenas existe na versão GNU do du.
- ▶ `du -sh <dir>`
 - s: retorna a soma da utilização do disco de todos os arquivos de um determinado diretório.



Medindo o uso do disco (2)

► `df -h <dir>`

Informa a utilização do disco e o espaço livre do sistema de arquivos que contém o diretório passado como parâmetro. Similarmente, a opção `-h` apenas existe na versão GNU do `df`.

► Exemplo:

```
> df -h .
```

filesystem	Size	Used	Avail	Use%	Mounted on
/dev/hda5	9.2G	7.1G	1.8G	81%	/

► `df -h`

Retorna informações sobre a utilização do espaço em disco de todos os sistemas de arquivos disponíveis no sistema. Quando surgem erros, é útil para identificar falta de espaço em disco.



Compactação

Muito útil para dividir arquivos grandes e para economizar espaço.

- ▶ `[un]compress <arquivo>`

Utilitário tradicional de compressão do Unix. Cria arquivos .Z. Apenas mantido para compatibilidade. Performance mediana.

- ▶ `g[un]zip <arquivo>`

Utilitário GNU de compressão zip. Cria arquivos .gz. Performance muito boa (similar ao Zip).

- ▶ `b[un]zip2 <arquivo>`

Utilitário de compressão mais recente e mais efetivo. Cria arquivos .bz2. Geralmente é de 20 a 25% melhor do que o gzip.

Use esse! Disponível agora em todos os sistemas Unix.



Arquivamento (1)

Útil para realizar backups ou distribuir um conjunto de arquivos dentro de um único arquivo (pacote).

▶ `tar`: originalmente “tape archive” (“Arquivador em fita”).

▶ Criando um arquivo:

```
tar cvf <pacote> <arquivos ou diretórios>
```

`c`: create

`v`: verbose. Útil para acompanhar o progresso do arquivamento.

`f`: pacote. Pacote a ser criado (caso contrário, a fita será usada)

▶ Exemplo:

```
tar cvf /backup/home.tar /home  
bzip2 /backup/home.tar
```



Arquivamento (2)

- ▶ Visualizar o conteúdo de um pacote ou verificar sua integridade:

```
tar tvf <pacote>
```

t: test

- ▶ Extrair todos os arquivos de um pacote:

```
tar xvf <pacote>
```

- ▶ Extrair apenas poucos arquivos de um pacote:

```
tar xvf <pacote> <arquivos ou  
diretórios>
```

Arquivos ou diretórios são passados com o caminho (path) relativo ao diretório root do pacote.



Opções extras do GNU tar

`tar = gtar = GNU tar on GNU / Linux`

Pode compactar e descompactar pacotes. Útil para evitar a criação de enormes arquivos intermediários.

Muito mais simples do que usar o `tar` e, em seguida, o `bzip2`!

▶ Opção `j` : [des]compacta com `bzip2`

▶ Opção `z` : [des]compacta com `gzip`

▶ Exemplos (qual deles é mais fácil de lembrar?)

▶ `gtar jcvf bills_bugs.tar.bz2 bills_bugs`

▶ `tar cvf - bills_bugs | bzip2 > bills_bugs.tar.bz2`



O comando wget

Ao invés de realizar o download de arquivos a partir do seu browser, apenas copie e cole a sua URL e faça o download com **wget**!

Principais funcionalidades do **wget** :

- ▶ Suporte a http e ftp;
- ▶ Pode reiniciar downloads interrompidos;
- ▶ Pode realizar o download de sites inteiros ou ao menos verificar links quebrados;
- ▶ Muito útil em scripts ou quando a interface gráfica não está disponível (administração do sistema, sistemas embarcados);
- ▶ Suporte a proxy (Variáveis de ambiente `http_proxy` e `ftp_proxy`).



wget - Exemplos

- ▶ `wget -c \`
`http://microsoft.com/customers/dogs/winxp4dogs.zip`
Continua um download interrompido.
- ▶ `wget -m http://lwn.net/`
Espelha um site.
- ▶ `wget -r -np http://www.xml.com/ldd/chapter/book/`
Faz o download recursivo de um livro on-line para acesso off-line.
-np: "no-parent". Apenas segue os links no diretório corrente.



Verificando a integridade de arquivos

Solução de baixo custo para verificar a integridade de arquivos.

▶ `md5sum FC3-i386-disk*.iso > MD5SUM`

Computa um checksum de 128 bits MD5 (Message Digest Algorithm 5) dos arquivos passados como parâmetro. A saída é geralmente redirecionada para um arquivo.

▶ Saída exemplo:

```
db8c7254beeb4f6b891d1ed3f689b412 FC3-i386-disc1.iso  
2c11674cf429fe570445afd9d5ff564e FC3-i386-disc2.iso  
f88f6ab5947ca41f3cf31db04487279b FC3-i386-disc3.iso
```

▶ `md5sum -c MD5SUM`

Verifica a integridade dos arquivos em MD5SUM pela comparação dos checksums MD5 armazenados no arquivo com os checksums reais.

Introdução ao Unix e ao GNU/Linux

Diversos
Impressão



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

103

Impressão Unix

- ▶ Impressão multiusuário, com suporte a múltiplos trabalhos de impressão, múltiplos clientes e múltiplas impressoras.
No Unix/Linux os comandos de impressão na verdade não imprimem. Eles enviam trabalhos de impressão para as filas, que podem estar localizadas localmente, em servidores de impressão da rede ou em impressoras de rede.
- ▶ Sistema independente da impressora:
Servidores de impressão apenas aceitam trabalhos de impressão no formato PostScript ou texto.
Os drivers da impressora no servidor se encarregam da conversão para o formato utilizado pela impressora.
- ▶ Sistema robusto:
Reinicie o sistema e ele continuará a imprimir trabalhos de impressão pendentes.



Comandos de Impressão

- ▶ Variável de ambiente útil: `PRINTER`

Configura a impressora default do sistema. Exemplo:

```
export PRINTER=lp
```

- ▶ `lpr [-P<queue>] <arquivos>`

Envia os arquivos passados como parâmetro para a fila de impressão especificada. Os arquivos devem estar no formato texto ou PostScript. Caso contrário, você apenas imprimirá “lixo”.

- ▶ `a2ps [-P<queue>] <arquivos>`

“Any to PostScript” converte vários formatos para o formato PostScript e envia a saída para a fila especificada. Funcionalidades úteis: várias páginas por folha, numeração de página, resumo informativo, etc.



Controle de trabalhos de impressão

► `lpq [-P<queue>]`

Lista todos os trabalhos de impressão da fila passada como parâmetro ou da fila padrão.

```
lp is not ready
```

Rank	Owner	Job	arquivo(s)	Total Size
1st	asloane	84	nsa_windows_backdoors.ps	60416 bytes
2nd	amoore	85	gw_bush_iraq_mistakes.ps	65024000 bytes

► `cancel <job#> [<queue>]`

Remove da fila padrão o trabalho de impressão cujo número foi passado como parâmetro.



Usando arquivos PostScript e PDF

Visualizando um arquivo PostScript

- ▶ Existem visualizadores PostScript, mas sua qualidade deixa a desejar.
- ▶ É melhor converter o arquivo PostScript para PDF com `ps2pdf`:
`ps2pdf decss_algorithm.ps`
`xpdf decss_algorithm.pdf &`

Imprimindo um arquivo PDF

- ▶ Você não precisa abrir um leitor de arquivos PDF!
- ▶ Melhor converter para PostScript com `pdf2ps`:
`pdf2ps rambaldi_artifacts_for_dummies.pdf`
`lpr rambaldi_artifacts_for_dummies.ps`



Introdução ao Unix e ao GNU/Linux

Diversos

Comparando arquivos e diretórios



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

108

Comparando arquivos e diretórios

▶ `diff arquivo1 arquivo2`

Reporta a diferença entre 2 arquivos, ou nada caso os arquivos sejam idênticos.

▶ `diff -r dir1/ dir2/`

Reporta todas as diferenças entre arquivos com o mesmo nome em 2 diretórios.

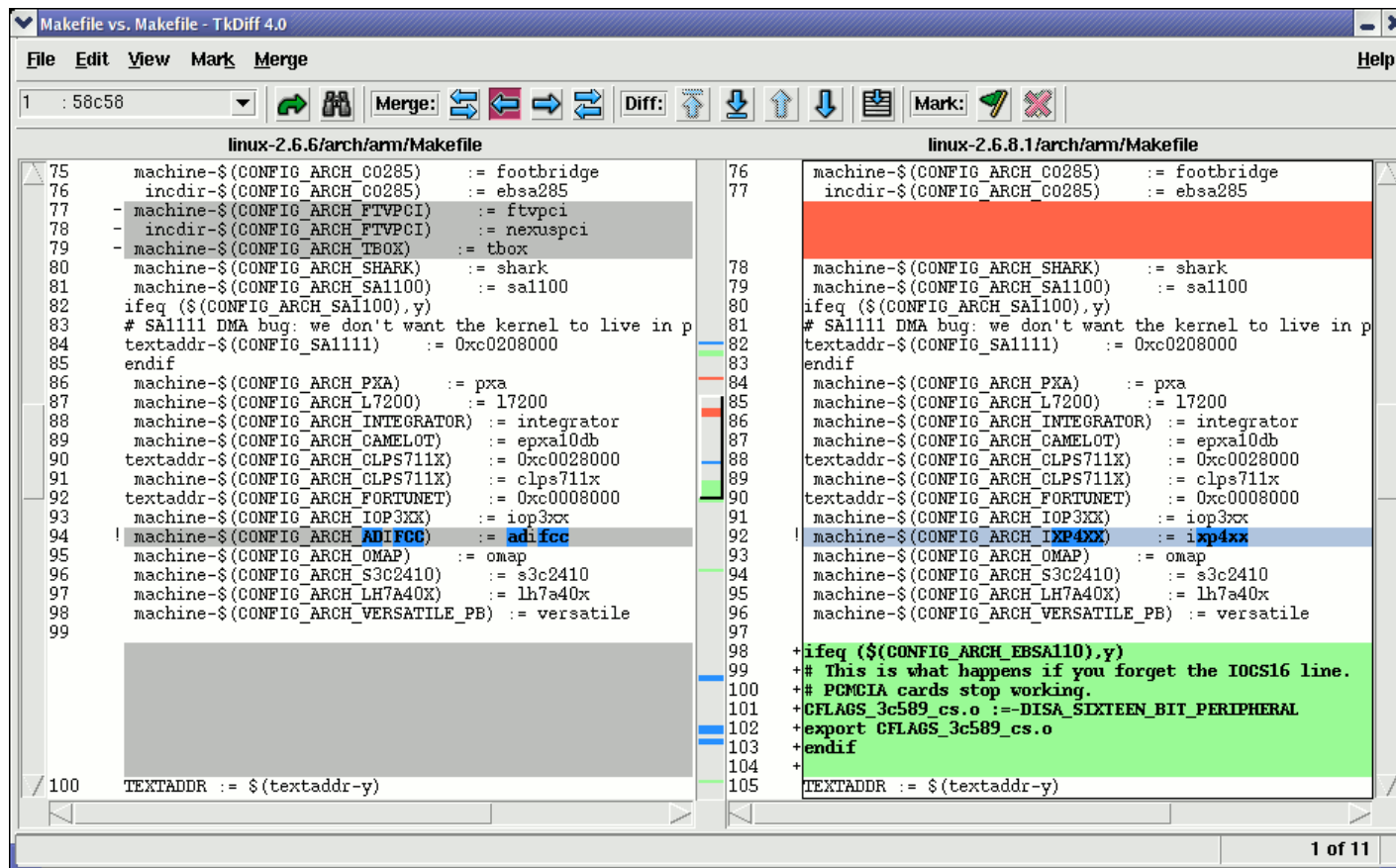
▶ Para investigar as diferenças em detalhes é melhor usar ferramentas gráficas!



tkdiff

<http://tkdiff.sourceforge.net/>

Ferramenta útil para comparar arquivos e consolidar as diferenças.



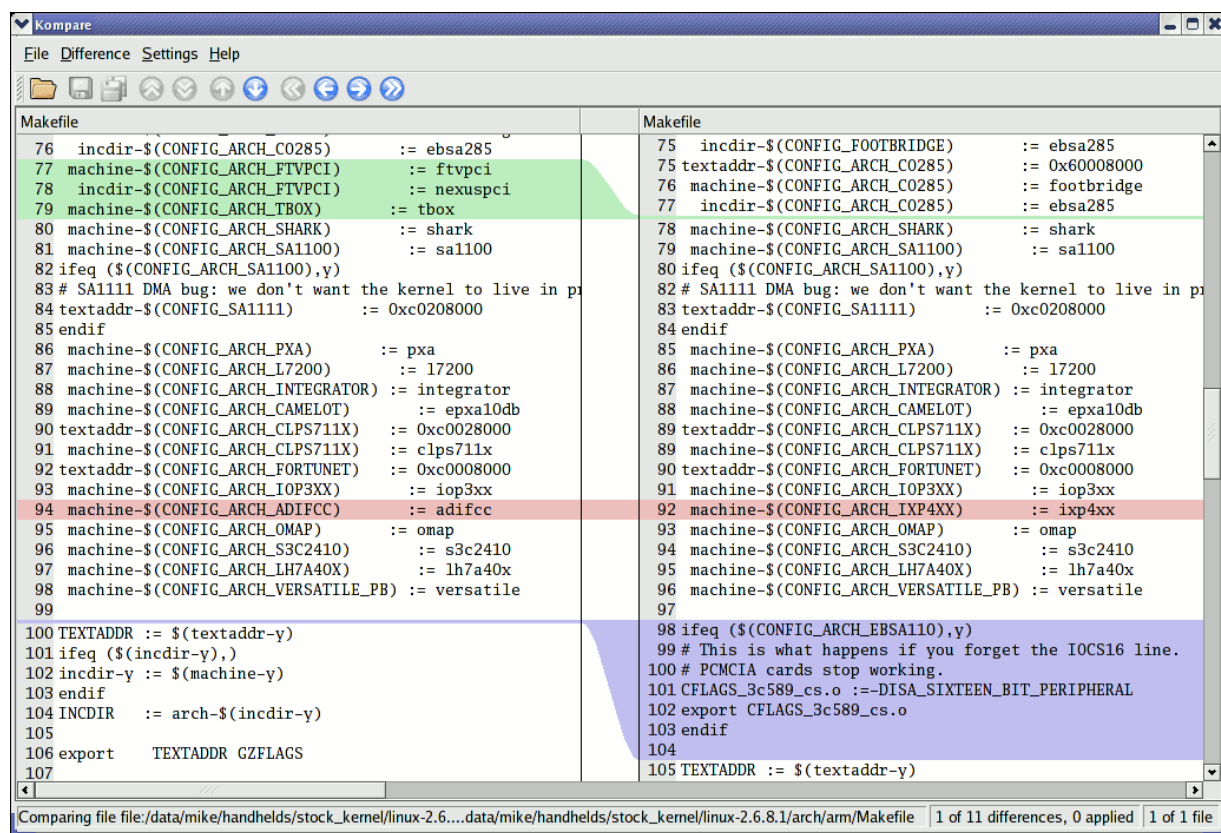
```
1 : 58c58
File Edit View Mark Merge Help
Merge: Diff: Mark:

linux-2.6.6/arch/arm/Makefile
75 machine-$(CONFIG_ARCH_C0285) := footbridge
76 incdir-$(CONFIG_ARCH_C0285) := ebsa285
77 - machine-$(CONFIG_ARCH_FTVPCI) := ftvpci
78 - incdir-$(CONFIG_ARCH_FTVPCI) := nexuspai
79 - machine-$(CONFIG_ARCH_TBOX) := tbox
80 machine-$(CONFIG_ARCH_SHARK) := shark
81 machine-$(CONFIG_ARCH_SA1100) := sa1100
82 ifeq ($(CONFIG_ARCH_SA1100),y)
83 # SA1111 DMA bug: we don't want the kernel to live in p
84 textaddr-$(CONFIG_SA1111) := 0xc0208000
85 endif
86 machine-$(CONFIG_ARCH_PXA) := pxa
87 machine-$(CONFIG_ARCH_L7200) := l7200
88 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
89 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
90 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
91 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
92 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
93 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
94 ! machine-$(CONFIG_ARCH_ADIFCC) := adifcc
95 machine-$(CONFIG_ARCH_OMAP) := omap
96 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
97 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
98 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
99
100 TEXTADDR := $(textaddr-y)

linux-2.6.8.1/arch/arm/Makefile
76 machine-$(CONFIG_ARCH_C0285) := footbridge
77 incdir-$(CONFIG_ARCH_C0285) := ebsa285
78 machine-$(CONFIG_ARCH_SHARK) := shark
79 machine-$(CONFIG_ARCH_SA1100) := sa1100
80 ifeq ($(CONFIG_ARCH_SA1100),y)
81 # SA1111 DMA bug: we don't want the kernel to live in p
82 textaddr-$(CONFIG_SA1111) := 0xc0208000
83 endif
84 machine-$(CONFIG_ARCH_PXA) := pxa
85 machine-$(CONFIG_ARCH_L7200) := l7200
86 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
87 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
88 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
89 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
90 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
91 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
92 ! machine-$(CONFIG_ARCH_IXP4XX) := ixp4xx
93 machine-$(CONFIG_ARCH_OMAP) := omap
94 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
95 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
96 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
97
98 +ifeq ($(CONFIG_ARCH_EBSA110),y)
99 +# This is what happens if you forget the IOCS16 line.
100 +# PCMCIA cards stop working.
101 +CFLAGS_3c589_cs.o := -DISA_SIXTEEN_BIT_PERIPHERAL
102 +export CFLAGS_3c589_cs.o
103 +endif
104
105 TEXTADDR := $(textaddr-y)
```

kompare

Outra ferramenta interessante para comparar arquivos e consolidar as diferenças. Parte do pacote kdesdk (Fedora Core).



```
File Difference Settings Help
Makefile
76 incdir-$(CONFIG_ARCH_C0285) := ebsa285
77 machine-$(CONFIG_ARCH_FTVPCI) := ftvpci
78 incdir-$(CONFIG_ARCH_FTVPCI) := nexuspci
79 machine-$(CONFIG_ARCH_TBOX) := tbox
80 machine-$(CONFIG_ARCH_SHARK) := shark
81 machine-$(CONFIG_ARCH_SA1100) := sa1100
82 ifeq ($(CONFIG_ARCH_SA1100),y)
83 # SA1111 DMA bug: we don't want the kernel to live in p
84 textaddr-$(CONFIG_SA1111) := 0xc0208000
85 endif
86 machine-$(CONFIG_ARCH_PXA) := pxa
87 machine-$(CONFIG_ARCH_L7200) := l7200
88 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
89 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
90 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
91 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
92 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
93 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
94 machine-$(CONFIG_ARCH_ADIFCC) := adifcc
95 machine-$(CONFIG_ARCH_OMAP) := omap
96 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
97 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
98 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
99
100 TEXTADDR := $(textaddr-y)
101 ifeq ($(incdir-y),)
102 incdir-y := $(machine-y)
103 endif
104 INCDIR := arch-$(incdir-y)
105
106 export TEXTADDR GZFLAGS
107
Makefile
75 incdir-$(CONFIG_FOOTBRIDGE) := ebsa285
75 textaddr-$(CONFIG_ARCH_C0285) := 0x60008000
76 machine-$(CONFIG_ARCH_C0285) := footbridge
77 incdir-$(CONFIG_ARCH_C0285) := ebsa285
78 machine-$(CONFIG_ARCH_SHARK) := shark
79 machine-$(CONFIG_ARCH_SA1100) := sa1100
80 ifeq ($(CONFIG_ARCH_SA1100),y)
82 # SA1111 DMA bug: we don't want the kernel to live in p
83 textaddr-$(CONFIG_SA1111) := 0xc0208000
84 endif
85 machine-$(CONFIG_ARCH_PXA) := pxa
86 machine-$(CONFIG_ARCH_L7200) := l7200
87 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
88 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
89 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
89 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
90 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
91 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
92 machine-$(CONFIG_ARCH_IXP4XX) := ixp4xx
93 machine-$(CONFIG_ARCH_OMAP) := omap
94 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
95 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
96 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
97
98 ifeq ($(CONFIG_ARCH_EBSA110),y)
99 # This is what happens if you forget the IOCS16 line.
100 # PCMCIA cards stop working.
101 CFLAGS_3c589_cs.o := -DISA_SIXTEEN_BIT_PERIPHERAL
102 export CFLAGS_3c589_cs.o
103 endif
104
105 TEXTADDR := $(textaddr-y)
```

Comparing file file:/data/mike/handhelds/stock_kernel/linux-2.6....data/mike/handhelds/stock_kernel/linux-2.6.8.1/arch/arm/Makefile 1 of 11 differences, 0 applied 1 of 1 file

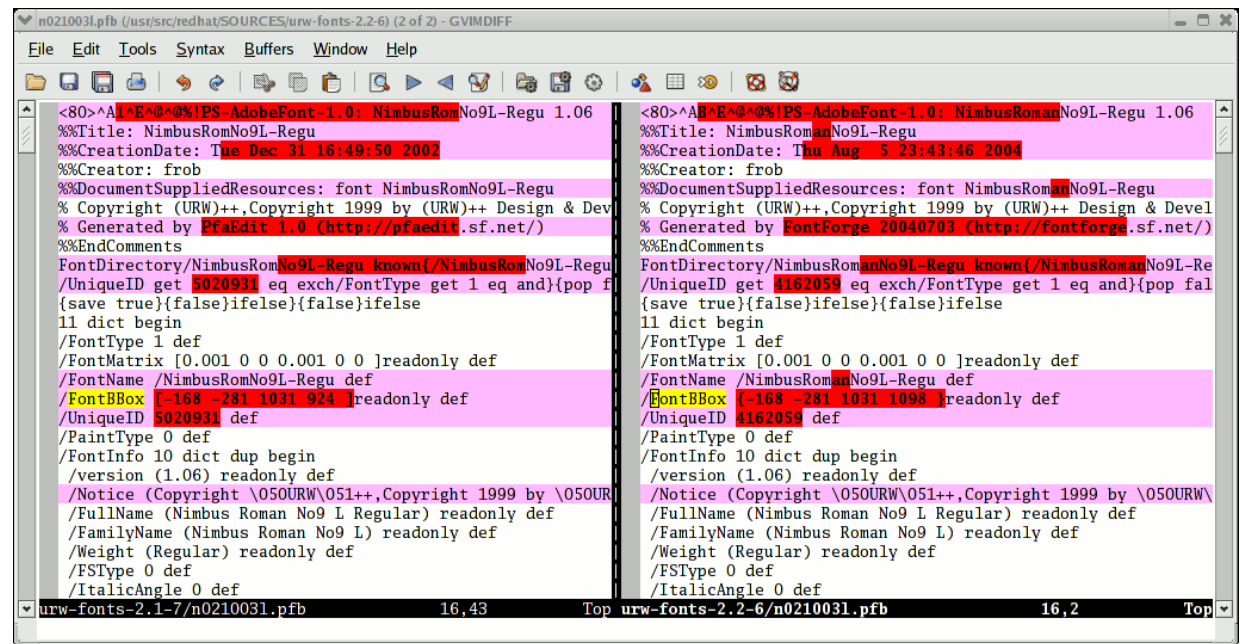
gvimdiff

Outra ferramenta interessante para visualizar as diferenças de arquivos.

Disponível na maioria das distribuições, junto com o programa `gvim`.

Aparentemente não usa o `diff`.

Trabalha inclusive com arquivos binários!



Introdução ao Unix e ao GNU/Linux

Diversos Procurando arquivos



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

113

O comando find

Melhor explicado por meio de alguns exemplos!

▶ `find . -name "*.pdf"`

Lista todos os arquivos `*.pdf` no diretório corrente (`.`) e em seus subdiretórios. Você precisa incluir as aspas para prevenir a expansão do caractere `*` pelo shell.

▶ `find docs -name "*.pdf" -exec xpdf {} ';'`

Encontra todos os arquivos `*.pdf` no diretório `docs` e exibe um após o outro com o programa `xpdf`.

▶ Existem muitas outras possibilidades! Entretanto, os dois exemplos acima cobrem a maioria das necessidades.

O comando locate

Realiza pesquisas usando expressões regulares muito mais rápido que o `find`

- ▶ `locate keys`

Lista todos os arquivos no seu sistema com a string `keys` no nome.

- ▶ `locate "*.pdf"`

Lista todos os arquivos `*.pdf` que existem no sistema.

- ▶ `locate "/home/fridge/*beer*"`

Lista todos os arquivos `*beer*` no diretório passado como parâmetro (usa caminho absoluto)

- ▶ `locate` é muito mais rápido pois ele indexa todos os arquivos em uma base de dados dedicada, que é atualizada regularmente.

- ▶ `find` é mais recomendado para pesquisar por arquivos criados recentemente.



Introdução ao Unix e ao GNU/Linux

Diversos Vários outros comandos



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

116

Obtendo informações sobre os usuários

- ▶ **who**

Lista todos os usuários logados no sistema

- ▶ **whoami**

Informa com qual usuário eu estou logado

- ▶ **groups**

Informa os grupos nos quais estou cadastrado.

- ▶ **groups <user>**

Informa os grupos nos quais o usuário <user> está cadastrado.

- ▶ **finger <user>**

Exibe mais detalhes (nome real, etc) sobre o usuário <user>

Vem desabilitado em alguns sistemas (por razões de segurança)



Trocando de usuário

Você não precisa sair da sua conta de usuário para efetuar o logon com outra conta de usuário!

▶ `su hyde`

(Uso raro) Troca para a conta `hyde`, mas mantém as variáveis de ambiente do usuário original.

▶ `su - jekyll`

(Uso mais freqüente) Troca para a conta `jekyll`, com exatamente as mesmas configurações deste novo usuário.

▶ `su -`

Quando não é passado nenhum argumento, significa que a conta do usuário `root` será usada.



Comandos diversos (1)

▶ `sleep 60`

Aguarda 60 segundos (não consome recursos do sistema)

▶ `wc report.txt` (word count)

```
438  2115 18302 report.txt
```

Conta o número de linhas, palavras e caracteres de um arquivo ou da entrada padrão.



Comandos diversos (2)

- ▶ **bc** ("basic calculator?")

bc é uma útil calculadora, cheia de recursos. Inclui até mesmo uma linguagem de programação! Use a opção **-l** para obter suporte à sua biblioteca matemática.

- ▶ **date**

Retorna a data corrente. Útil em scripts para registrar quando os comandos começaram ou terminaram.



Introdução ao Unix e ao GNU/Linux

Administração básica de sistemas



Free Electrons

Introdução ao Unix e ao GNU/Linux

© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

121

Propriedades de arquivos

- ▶ `chown -R sco /home/linux/src` (-R: recursivo)
Torna o usuário `sco` o novo proprietário de todos os arquivos localizados em `/home/linux/src`
- ▶ `chgrp -R empire /home/askywalker`
Torna `empire` o novo grupo de todos os arquivos localizados em `/home/askywalker`.
- ▶ `chown -R borg:aliens usss_entreprise/`
`chown` pode ser utilizado para alterar o proprietário e o grupo ao mesmo tempo.



Desligando o sistema

- ▶ `shutdown -h +5` (`-h`: halt)
Desliga o sistema em 5 minutos. Os usuários receberão um aviso em seus consoles.
- ▶ `shutdown -r now` (`-r`: reboot)
- ▶ `init 0`
Outra forma de desligar o sistema (usado pelo comando `shutdown`)
- ▶ `init 6`
Outra forma de rebotar (usado pelo comando `shutdown`)
- ▶ `[Ctrl][Alt][Del]`
Também funciona no GNU/Linux (ao menos em PCs!)



Configuração de rede (1)

- ▶ `ifconfig -a`
Exibe detalhes sobre as interfaces de rede disponíveis no seu sistema.
- ▶ `ifconfig eth0`
Exibe detalhes sobre as interfaces de rede `eth0`.
- ▶ `ifconfig eth0 192.168.0.100`
Associa o endereço IP `192.168.0.100` à interface `eth0` (1 endereço IP por interface).
- ▶ `ifconfig eth0 down`
Desativa a interface `eth0`.
(libera seu endereço IP)



Configuração de rede (2)

- ▶ `route add default gw 192.168.0.1`

Configura a rota default para pacotes não destinados à rede local. O gateway (192.168.0.1) é responsável por enviá-los para o próximo gateway, e assim sucessivamente, até atingir o destino final.

- ▶ `route`

Lista as rotas existentes.

- ▶ `route del default`
`route del <IP>`

Deleta a rota passada como parâmetro.

Útil para definir uma nova rota.



Teste de funcionamento da rede

- ▶ `ping freshmeat.net`
`ping 192.168.1.1`

Tenta enviar pacotes para o computador passado como parâmetro e obtém como retorno uma confirmação (ou não) de cada pacote enviado.

- ▶

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_seq=0 ttl=150 time=2.51 ms  
64 bytes from 192.168.1.1: icmp_seq=1 ttl=150 time=3.16 ms  
64 bytes from 192.168.1.1: icmp_seq=2 ttl=150 time=2.71 ms  
64 bytes from 192.168.1.1: icmp_seq=3 ttl=150 time=2.67 ms
```

- ▶ Quando você consegue “pingar” seu gateway, isso significa que sua interface de rede está funcionando corretamente.
- ▶ Quando você consegue “pingar” um endereço externo (fora da sua rede local), isso significa que suas configurações de rede estão corretas.

Resumo de configuração de rede

Apenas para casos com uma única interface, sem servidor DHCP...

- ▶ Conecte-se à rede (cabo, wireless, etc...)
- ▶ Identifique qual é a sua interface de rede:
`ifconfig -a`
- ▶ Associe um endereço IP à sua interface (assumindo aqui `eth0`)
`ifconfig eth0 192.168.0.100` (exemplo)
- ▶ Adicione uma rota até o seu gateway (assumindo `192.168.0.1`) para pacotes destinados ao exterior da rede.
`route add default gw 192.168.0.1`

Resolução de nomes

- ▶ Seus programas precisam saber qual endereço IP corresponde a um determinado nome de host (por exemplo, `kernel.org`)
- ▶ Domain Name Servers (DNS) se encarregam disso.
- ▶ Você apenas tem de especificar o endereço IP de 1 ou mais servidores DNS no seu arquivo `/etc/resolv.conf`:
`nameserver 217.19.192.132`
`nameserver 212.27.32.177`
- ▶ As alterações produzem efeito imediatamente!



Criando sistemas de arquivos (filesystems)

Exemplos

▶ `mkfs.ext2 /dev/sda1`

Formata seu pendrive USB (`/dev/sda1`: 1ª partição) no formato EXT2

▶ `mkfs.ext2 -F disk.img`

Formata um arquivo de imagem de disco no formato EXT2

▶ `mkfs.vfat -v -F 32 /dev/sda1` (-v: verbose)

Formata seu pendrive USB no formato FAT32

▶ `mkfs.vfat -v -F 32 disk.img`

Formata um arquivo de imagem de disco no formato FAT32

Imagens de disco vazias podem ser criadas como no exemplo abaixo:

```
dd if=/dev/zero of=disk.img bs=1024 count=65536
```



Montando dispositivos (1)

- ▶ Para tornar os sistemas de arquivos de qualquer dispositivo de armazenamento (internos ou externos) visíveis no sistema, você tem de *montá-los*.
- ▶ Na primeira vez, crie um ponto de montagem no seu sistema:
`mkdir /mnt/usbdisk (example)`
- ▶ Agora, monte-o:
`mount -t vfat /dev/sda1 /mnt/usbdisk`
`/dev/sda1: physical device`
`-t:` especifica o tipo (formato) do sistema de arquivos
(`ext2`, `ext3`, `vfat`, `reiserfs`, `iso9660`...)



Montando dispositivos (2)

- ▶ Diversas opções de montagem estão disponíveis, como as que permitem a escolha das permissões ou a definição do proprietário e grupo dos arquivos. Veja a página do manual do `mount` para detalhes.
- ▶ As opções de montagem para cada dispositivo podem ser armazenadas no arquivo `/etc/fstab`
- ▶ Você pode montar uma imagem de sistema de arquivos armazenada em um arquivo (*dispositivos loopback*).
 - ▶ Útil para acessar o conteúdo de uma imagem ISO de um CDROM sem a necessidade de gravá-la em mídia.
 - ▶ Útil para criar uma “partição” Linux em um disco rígido que possui apenas partições Windows.

```
cp /dev/sda1 usbkey.img  
mount -o loop -t vfat usbkey.img /mnt/usbdisk
```



Listando os sistemas de arquivos montados

- ▶ Apenas execute o comando **mount** sem argumentos:

```
/dev/hda6 on / type ext3 (rw,noatime)
none on /proc type proc (rw,noatime)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hda4 on /data type ext3 (rw,noatime)
none on /dev/shm type tmpfs (rw)
/dev/hda1 on /win type vfat (rw,uid=501,gid=501)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

- ▶ Ou exiba o conteúdo do arquivo **/etc/mtab**
(Mesmo resultado. Atualizado pelo **mount** e **umount** cada vez que são executados.)



Desmontando dispositivos

▶ `umount /mnt/usbdisk`

Grava todas as escritas pendentes e desmonta o dispositivo passado como parâmetro, o qual poderá ser removido de forma segura.

▶ Para ser capaz de desmontar um dispositivo você deve antes fechar todos os arquivos abertos do dispositivo.

▶ Feche as aplicações que estão utilizando dados da partição montada.

▶ Certifique-se de que nenhum dos seus shells tem como diretório de trabalho algum diretório da partição montada.

▶ Você pode executar o comando `lsof` (**l**ist **o**pen **f**iles) para descobrir quais processos ainda tem arquivos abertos na partição montada.

Introdução ao Unix e ao GNU/Linux

GNU/Linux: pacotes utilizados por distribuições



Free Electrons

Introdução ao Unix e ao GNU/Linux

© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

134

Como encontrar pacotes

- ▶ Pacotes Debian: <http://www.debian.org/distrib/packages>
Pesquisa por pacote ou nome do arquivo.
- ▶ rpmfind: <http://rpmfind.net/>
Diversos pacotes RPM para Red Hat, Mandriva, Suse...



Identificando pacotes

A qual pacote um arquivo pertence?

- ▶ Útil para obter mais informações, obter o código fonte, encontrar novas versões, reportar bugs...
- ▶ Distribuições que utilizam pacotes **RPM**:
(Red Hat, Fedora, Mandriva, Suse...)

```
> rpm -qf /bin/ls  
coreutils-5.2.1-7
```

- ▶ Debian:

```
> dpkg -S /bin/ls  
fileutils: /bin/ls
```

Informações sobre pacotes

- ▶ Acesso à descrição do pacote, número de versão, fontes, etc.
- ▶ Distribuições baseadas em **RPM** :

```
rpm -qi <package-name>
```

- ▶ Debian:

```
dpkg -s <package-name>
```



Introdução ao Unix e ao GNU/Linux

Indo além



Free Electrons

Introdução ao Unix e ao GNU/Linux
© Copyright 2006-2004, Michael Opdenacker
Creative Commons Attribution-ShareAlike 2.0 license
<http://free-electrons.com>



15 de Set de 2009

138

Help de comandos

Alguns comandos do Unix e a maioria dos comandos do GNU/Linux oferece ao menos um argumento de help:

▶ `-h`

(`-` é na maioria das vezes usado para introduzir opções de 1 caractere)

▶ `--help`

(`--` é sempre usado para introduzir a opção correspondente “longa”, o que torna os scripts mais fáceis de compreender)

Você também freqüentemente obtém um curto resumo das opções quando informa um argumento inválido.



Páginas do Manual

`man <keyword>`

Exibe uma ou várias páginas de manual para `<keyword>`

▶ `man man`

A maioria das páginas de manual disponíveis são sobre comandos Unix, mas algumas são sobre funções C, cabeçalhos (headers) ou estruturas de dados, ou mesmo sobre arquivos de configuração do sistema!

▶ `man stdio.h`

▶ `man fstab` (for `/etc/fstab`)

Os arquivos das páginas de manual estão localizados no diretório especificado pela variável de ambiente `MANPATH`.



Páginas Info

- ▶ No GNU/Linux, páginas de manual estão sendo substituídas por páginas info. Algumas páginas do manual, quando acessadas, referem-se às páginas info.

`info <command>`

- ▶ Recursos do comando `info`:
 - ▶ Documentação estruturada em seções (“nós”) e subseções (“subnós”)
 - ▶ Possibilidade de navegar nessa estrutura: `top`, `next`, `prev`, `up`
 - ▶ Páginas info geradas a partir dos mesmos fontes `texinfo` como páginas de documentação HTML.



Pesquisando na Internet por recursos (1)

Investigando a solução de problemas

- ▶ A maioria dos fóruns e listas de e-mail é pública, e são indexadas com bastante frequência pelo Google.
- ▶ Se você está investigando uma mensagem de erro, escreva-a no formulário de pesquisa da mesma forma como é exibida, entre aspas duplas (“mensagem de erro”). Há muitas chances de que outras pessoas já tenham passado pelo mesmo problema e o solucionado.
- ▶ Não se esqueça de usar o Google Groups:
<http://groups.google.com/> Esse site indexa mais de 20 anos de mensagens de grupos de notícias.

Pesquisando na Internet por recursos (2)

Procurando documentação

- ▶ Pesquise por `<tool>` ou `<tool> page` para encontrar a ferramenta (`<tool>`) ou a home page do seu respectivo projeto e então localize os últimos recursos de documentação.
- ▶ Pesquise por `<tool> documentation` ou `<tool> manual` no seu mecanismo de busca favorito.

Procurando informação técnica geral

- ▶ WikiPedia: <http://wikipedia.org>
Diversas definições úteis em ciência da computação. Uma enciclopédia real! Aberta a contribuições de qualquer pessoa.



Material de treinamento

O material de treinamento também está disponível em:

http://free-electrons.com/training/intro_unix_linux

Ele é um útil complemento para consolidar o que você aprendeu aqui. Ele não diz como fazer os exercícios, entretanto, reforça o aprendizado das noções e ferramentas aqui introduzidas.

Se você não estiver conseguindo finalizar algum dos exercícios é sinal que você não prestou atenção a alguma das leituras e deve retornar aos slides para encontrar o material relacionado ao assunto em que você está enfrentando dificuldades.



Como ajudar

Se você apóia este trabalho, você pode ajudar...

- ▶ Enviando correções, sugestões, contribuições e traduções.
- ▶ Solicitando à sua organização que compre sessões de treinamento realizadas pelo autor deste documento (veja <http://free-electrons.com/training>)
- ▶ Comentando sobre ele com seus amigos, colegas e comunidade de Software Livre local.
- ▶ Adicionando links aos nossos materiais on-line no seu website para aumentar a nossa visibilidade em resultados de mecanismos de busca.



Agradecimentos

- ▶ Ao projeto OpenOffice.org, pelas ferramentas de apresentação e processamento de texto que satisfazem a todas as minhas necessidades.
- ▶ À comunidade Handhelds.org, por dar-me tanta ajuda e tantas oportunidades de ajudar.
- ▶ Aos membros de toda a comunidade de Software Livre e Código Aberto, por compartilhar o melhor que possuem: seu trabalho, seu conhecimento, sua amizade.
- ▶ Às pessoas que me enviaram comentários e correções: Laurent Thomas, Jeff Ghislain, Leif Thande, Frédéric Desmoulins, Przemysław Ciesielski





Related documents



Free Electrons

Embedded Freedom

HOME DEVELOPMENT SERVICES TRAINING DOCS COMMUNITY COMPANY BLOG

Recent blog posts

ELC Europe in Grenoble

Free Electrons at ELC

Linux kernel 2.6.29 - New features for embedded users

The Buildroot project begins a new life

FOSDEM 2009 videos

USB-Ethernet device for Linux

Program for Embedded Linux Conference 2009 announced

Public session changes

Real hardware in our training sessions

Call for presentations for the LSM embedded track

Docs

Most of the below documents are presentations used in our [training sessions](#), or in technical conferences.

License

 All our documents are available under the terms of the [Creative Commons Attribution-ShareAlike 3.0 license](#). This essentially means that you are free to download, distribute and even modify them, provided you mention us as the original authors and that you share these documents under the same conditions.

Linux kernel

- [Embedded Linux kernel and driver development](#)
- [New features in Linux 2.6](#) (since 2.6.10)
- [Kernel initialization](#)
- [Porting Linux to new hardware](#)
- [Power management in Linux](#)
- [Linux PCI drivers](#)
- [Block device drivers](#)
- [Linux USB drivers](#)
- [DMA](#)

Architecture specific documents

- [ARM Linux specifics](#)
- [Linux on TI OMAP processors](#)

Embedded Linux system development

- [Embedded Linux system development](#)
- [Real time in embedded Linux systems](#)
- [Block filesystems](#)
- [Flash filesystems](#)
- [Free software development tools](#)
- [The U-boot bootloader](#)
- [The GRUB bootloader](#)
- [The blob bootloader](#)
- [Hotplugging with udev](#)
- [Introduction to uClinux](#)
- [Java in embedded Linux](#)
- [Embedded Linux optimizations](#)
- [Audio in embedded Linux systems](#)
- [Multimedia in embedded Linux systems](#)
- [Embedded Linux From Scratch... in 40 minutes!](#)
- [Building embedded Linux systems with Buildroot](#)
- [Developing embedded distributions with OpenEmbedded](#)
- [The Scratchbox development environment](#)

Miscellaneous

- [Introduction to the Unix command line](#)
- [SSH](#)
- [Linux virtualization solutions](#) (with an embedded perspective)
- [Advantages of Free Software and Open Source in embedded systems](#)
- [Introduction to GNU/Linux and Free Software](#)

All our technical presentations
on <http://free-electrons.com/docs>

- ▶ Linux kernel
- ▶ Device drivers
- ▶ Architecture specifics
- ▶ Embedded Linux system development



How to help

You can help us to improve and maintain this document...

- ▶ By sending corrections, suggestions, contributions and translations
- ▶ By asking your organization to order development, consulting and training services performed by the authors of these documents (see <http://free-electrons.com/>).
- ▶ By sharing this document with your friends, colleagues and with the local Free Software community.
- ▶ By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Linux kernel

- Linux device drivers
- Board support code
- Mainstreaming kernel code
- Kernel debugging

Embedded Linux Training

All materials released with a free license!

- Unix and GNU/Linux basics
- Linux kernel and drivers development
- Real-time Linux, uClinux
- Development and profiling tools
- Lightweight tools for embedded systems
- Root filesystem creation
- Audio and multimedia
- System optimization

Free Electrons

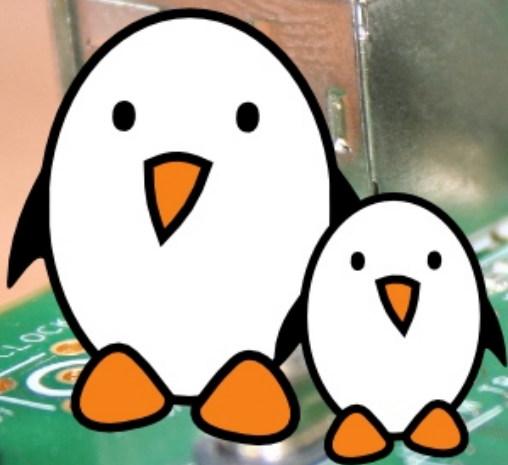
Our services

Custom Development

- System integration
- Embedded Linux demos and prototypes
- System optimization
- Application and interface development

Consulting and technical support

- Help in decision making
- System architecture
- System design and performance review
- Development tool and application support
- Investigating issues and fixing tool bugs



Free Electrons
Embedded Linux Experts

<http://free-electrons.com>