MacPRO
o suplemento dos power users
Ano 5 - Nº 42
Parte integrante da revista Macmania
Não pode ser vendido separadamente

QuickTime 6 na cola do MPEG-4 (e vice-versa)

No momento em que escrevo estas linhas, a Apple diz que tem o QuickTime 6 pronto mas não vai lançá-lo ainda, por motivos que envolvem disputas de *royalties* em relação ao padrão MPEG-4, que já tem a segunda versão de sua especificação pronta e possui como base justamente a tecnologia do QuickTime. Pode ser que, quando esta revista estiver em suas mãos, a Apple tenha liberado a versão 6, mas eu não apostaria todas minhas fichas nisso. A verdade é que, conforme veremos a seguir, para entender o QuickTime 6 é necessário falar sobre MPEG-4 e vice-versa. Porém, ambos os assuntos estão no meio de uma confusa discussão burocrática de *royalties* da disputa pelo mercado de *streaming*, que envolve, além da Apple, a Microsoft e a RealNetworks. Para complicar ainda mais, reinam muitas dúvidas entre as pessoas comuns sobre o que é e o que não é o MPEG-4. Assim, vamos tentar esclarecer um pouco essa zona toda, botando alguma ordem na casa.

por Márcio Nigro

O **MPEG-4**, criado pelo Motion Picture Experts Group (MPEG), é tido como o sucessor para o MPEG-2, tecnologia atualmente usada em aplicações como filmes DVD e televisão digital via satélite. O MPEG-4 está em desenvolvimento desde 1993 e é um padrão amplo e complexo, podendo proporcionar melhor comunicação e maior compressão do que o MPEG-2. Ele é capaz de realizar *streaming* (transmissão ao vivo pela Internet) de diversos tipos de mídia e não apenas áudio e vídeo. Assim, o servidor pode enviar programas sob demanda para a TV de sua casa, tocar filmes DVD a partir de comunicação sem fio, transmitir rádio via satélite e outras diversas possibilidades.

Tudo parece lindo, mas tudo depende dos protocolos de comunicação que serão adotados pelas três grandes: Apple, Microsoft e RealNetworks. Se um padrão aberto for adotado para tais aplicações, o futuro delas será grandioso. Porém, se acontecer de um padrão proprietário ou extremamente dispendioso ser adotado, tais visões caem por terra.

Microsoft

A Microsoft, com sua eterna tendência monopolista, está louca para dominar o mercado de streaming, apostando suas fichas na tecnologia Windows Media. Bill Gates, no entanto, acreditava que a sua tecnologia **Advanced Streaming Format** (ASF) seria escolhida como base do MPEG-4 e, por isso, ela foi implementada no Windows Media Player e vendida como se fosse o MPEG-4 (o nome original era MS MPEG-4 v3). Mas os planos do velho Bill foram por água abaixo quando o padrão adotado foi o **QuickTime** ▶

da Apple, mais flexível e aberto. Por isso, não se engane: ASF não é MPEG-4, muito embora servidores e tocadores Windows Media possam tocar streams de MPEG-4, uma vez que incluem suporte a *codecs* (padrões de compressão e descompressão) de áudio MPEG-4. Contudo, a Microsoft não desiste fácil e encontrou um outro modo de ganhar terreno nesse mercado: seduziu os provedores de conteúdo, oferecendo encriptação e proteção contra cópia na tecnologia atual do Windows Media. É claro que isso chamou a atenção da indústria fono/cinematográfica, que quer, acima de tudo, controlar o modo como você vê filmes ou escuta música pela Internet.

A Microsoft até tentou, mas não conseguiu se apossar do setor

DivX

A tecnologia **DivX**;-) (o *emoticon* sorridente e piscante faz parte do nome mesmo... vá entender) também é uma tecnologia comumente confundida com o MPEG-4. Em essência, é apenas um *hack* do codec MS MPEG-4 v3 que permite que ele seja usado dentro de um arquivo AVI. O DivX também costuma ser relacionado à subcultura da pirataria de DVD (em que parece ser a ferramenta preferida) e ao movimento Open Source, muito embora o código não seja aberto, uma vez que se baseia em tecnologia da Microsoft. Enfim, também não tem nada a ver com MPEG-4 e com streaming. Em todo caso, já existe um plug-in "oficial" para fazer o QuickTime 5 entender arquivos DivX (em www.divx.com).

Real

A RealNetworks, por sua vez, continua vivendo de suas tecnologias proprietárias **RealAudio** e **RealVideo**, líderes em streaming, e promete implementar suporte a MPEG-4 até o final deste ano. Apesar de o RealPlayer trabalhar com padrões como AIFF, AVI e até algumas mídias QuickTime, o programa só está licenciado para tocar streams no formato fechado da empresa. Basicamente, a RealNetworks está em situação semelhante à da Microsoft, com a exceção de não poder forçar a entrada de seu produto em todos os PCs com Windows sobre a face do globo.

QuickTime

Em terceiro lugar na corrida do streaming, depois do RealPlayer e do Windows Media Player, está o nosso amigo **QuickTime**. Por outro lado, tem uma grande vantagem no mercado de mídia não-streaming, já que o QuickTime frequenta todos os Macs e muitos PCs há mais de dez anos, e quase todos os CD-ROMs multimídia usam a tecnologia. É claro que esse fato não garantiu o sucesso do QuickTime em mídia streaming, se bem que a Apple tem se esforçado bastante para mudar esse panorama. O QuickTime oferece suporte a streaming desde a versão 3; a Apple foi cuidadosa em usar métodos de "entrega" padrão como RTP e RTSP, o que forçou a RealNetworks a fazer o mesmo. Além disso, a Apple liberou tanto o servidor de streaming quanto o seu código fonte, oferecendo uma alternativa gratuita para quem quiser oferecer conteúdo na Internet (desde que você tenha um servidor Mac, é claro). O QuickTime usa alguns compressores proprietários para streaming, como o Sorenson Video; em compensação, esses codecs funcionam em diversas plataformas e você ainda pode optar por outros.

A Apple poderia lançar o QuickTime 6 por etapas, se quisesse

História mal contada

Como já dissemos, o MPEG-4 é baseado no padrão QuickTime, o que aparentemente garante uma vantagem à tecnologia da Apple. Também já foi comentado que a Apple diz ter o QuickTime 6 prontinho, mas se recusa a lançar. O motivo oficial é que a empresa estaria esperando que os termos de licença de vídeo MPEG-4 fossem melhorados.
A MPEG-LA, o maior grupo de detentores de patentes do MPEG-4, propõe que a licença do MPEG-4 inclua pagamento de *royalties* por parte de companhias, como a Apple, que fornecem codecs MPEG-4, e também de provedores de conteúdo que fazem streaming MPEG-4. A Apple até concorda em pagar *royalties* razoáveis por usar codecs MPEG-4 no QuickTime, mas acha que essa história de cobrar também de quem distribui comprometerá o sucesso do padrão.
Porém, toda essa história possui um cheiro esquisito. As últimas versões do QuickTime tiveram longos períodos de beta teste. Como essa versão já está prontinha e com suporte a MPEG-4? E mesmo que seja verdade, o QuickTime tem arquitetura modular, na qual os compressores e descompressores podem vir separados. Tanto que o QT 5 teve versão *preview* pública sem o Sorenson Video 3. Dessa forma, a versão 6 poderia sair sem os compressores MPEG-4.
Então, por que não sai?
O chute mais provável é: não está pronto ainda – ou, pelo menos, não estava (é difícil escrever um texto atual um mês antes de ele ser publicado).
De qualquer modo, a Apple está certa ao reclamar dos *royalties* exigidos aos provedores de conteúdo. A MPEG-LA propõe cobrar US$ 0,02 por hora como taxa de licença, mas o entendimento geral é de que qualquer tecnologia com direitos cobrados por hora está fadada a fracassar. Já existem alternativas por aí e, mesmo que não houvessem, a Microsoft ou a RealNetworks abocanhariam rapidamente tal oportunidade.
Toda essa história de o QuickTime 6 estar pronto parece ser mais uma estratégia da Apple para criar um "fuzuê" em cima de seu produto e pressionar a MPEG-LA a esquecer essa história de *royalties* por hora. O fato é que sem o MPEG-4 o QuickTime não tem novos recursos suficientes para justificar a mudança de versão. O mais provável é que a Apple esteja blefando para ver o que acontece.

O que vem por aí

A Apple diz que são nove os principais novos recursos do QuickTime 6. Veja quais:
•**MPEG-4** – Os componentes MPEG-4 da versão 6 não são licenciados da DivX Net-

Variações do MPEG

MPEG-1

Utilizado para compressão de vídeo para CD-ROM e VideoCD. Não oferece grande qualidade, mas quebra um galho em resolução de 320x240 pixels.

MPEG-2

Oferecendo vídeo de alta qualidade e boa compressão, o MPEG-2 liderou a revolução do vídeo digital, possibilitando a disseminação do DVD e sistemas digitais via satélite e cabo, além da HDTV, também conhecida como "TV de alta definição". O MPEG-2 não se tornou uma tecnologia relevante de playback nos computadores, devido a questões de licenciamento e não por motivos tecnológicos.

MPEG-3

Não existe tal coisa. O MPEG-3 era para ser uma versão de maior definição do MPEG-2, que acabou mostrando-se bem mais flexível e expansível do que o inicialmente planejado.

MPEG-7

Ainda em desenvolvimento, o MPEG-7 é centrado em metadados, indexação e organização. O padrão poderá oferecer, por exemplo, um modo universal de catalogar e buscar vídeo.

MPEG-21

Projeto de longo prazo e ainda em estágio embrionário, pretende criar um padrão para gerenciamento de direitos autorais, sistemas de pagamento, monitoração de qualidade de serviço e verificação, entre outros recursos.

works ou de outra fonte; são desenvolvidos pela própria Apple. A vantagem disso é que não haverá taxas extras para desenvolver streams MPEG-4 profissionais, como acontece com o formato Sorenson, que oferece melhor qualidade e mais opções de codificação com o pacote Sorenson Video 3 (US$ 500). Assim, é provável que o QuickTime Pro 6 ofereça um bom codec MPEG-4 que dispense outros programas.

•**Advanced Audio Coding (ACC)** – O MP3, também conhecido como MPEG-1 Layer 3, é o mais conhecido padrão de compressão de áudio, mas já existe há uma década, e há outros formatos melhores. O AAC é parte da especificação MPEG-2 e é um dos melhores compressores de áudio, podendo lidar com vários canais (de um a 48), 15 canais de baixa frequência *(low-frequency enhancement channels)*, 15 streams de dados embutidos e até suporte a múltiplos idiomas. Mas a grande vantagem do AAC é oferecer cinco canais de áudio que, codificados a 384 kbps, são indistinguíveis do som original, mesmo para os ouvidos mais treinados. Para se ter uma idéia, para obter o mesmo resultado no formato MP3 a codificação tem que ser de 640 a 896 kbps. O ACC faz parte da especificação MPEG-4 como um dos vários compressores de áudio disponíveis. Apesar de a Apple se limitar a dizer que o QuickTime 6 terá suporte a ACC, é possível que ele seja o primeiro grande tocador de mídia com esse codec.

Os novos codecs de áudio e vídeo são muito mais eficientes

•**Code Excited Linear Prediction (CELP)** – Enquanto o AAC é direcionado à alta qualidade de áudio no MPEG-4, o CELP é a escolha para compressão a baixas taxas de *bitrate*, trabalhando com *sample rate* de 8 ou 16 kHz – chamadas de CELP de banda estreita e de banda larga, respectivamente. Como o PureVoice, esse formato funciona melhor com voz e com *bitrates* entre 6 e 24 kbps. Não é um recurso de grande utilidade, mas como foi incluído na especificação MPEG-4, faz parte do QuickTime 6.

•**ISMA 1.0** – A solução de streaming de MPEG-4 pela qual a Apple optou foi a especificação ISMA 1.0, criada pela Internet Streaming Media Alliance. O MPEG-4 é um padrão amplo, com diversas escolhas de áudio e vídeo. O ISMA identifica qual parte do stream MPEG-4 deve ser implementado no cliente, para garantir que mídias de streaming compatíveis com ISMA toquem corretamente, utilizando outros padrões se necessário, como RTP e RTSP. Porém, não é um formato aberto, mas sim "dedicado para o desenvolvimento de produtos e tecnologias que aderem aos padrões de mercado", como diz a própria Apple, tentando se enganar.

A interoperabilidade ISMA significará alguma coisa se surgirem bons programas oferecendo suporte à especificação. Além disso, requer clientes e servidores específicos, como é o caso do QuickTime, Real ou Windows Media. E – adivinhe – nem a Microsoft nem a RealNetworks aderiram à ISMA, de modo que seus clientes não são compatíveis. Sem pelo menos um deles, fica difícil a tecnologia plantar raízes na comunidade cibernética.

•**MPEG-1, MPEG-2, DVCPRO (PAL)** – Não são exatamente características novas. O QuickTime 5 Pro já inclui decodificação MPEG-1 e codificação e decodificação MPEG-2. O mesmo vale para o suporte a DVPRO que, apesar de a Apple dizer que é um grande novo recurso da versão 6, o QuickTime 5 também já o oferece, pelo menos no Mac OS X 10.1.2.

•**Flash 5** – O QuickTime 5 já suportava Flash 4 e versões anteriores. O suporte a Flash 5 já era mais do que esperado, mas mudará pouco a vida dos desenvolvedores.

De resto, pouco

Além das novidades mencionadas, o QuickTime 6 terá uma interface atualizada com um novo "Favoritos" e acesso mais fácil ao conteúdo QT, mas é mais ou menos isso o que se poderá esperar da nova versão.

A Apple não mencionou nada sobre o Sorenson Video 4 ou o QDX da Qdesign, que oferecem melhor qualidade de compressão que seus antecessores. A grande novidade é mesmo o suporte a MPEG-4, o que é natural, já que a Apple é fundadora da ISMA e a especificação é uma parte do MPEG-4.

Porém, como ninguém viu nenhum beta do QuickTime 6 – e conhecendo a Apple como conhecemos –, é possível que haja algo além do que já foi anunciado.

O que o MPEG-4 tem

A especificação MPEG-4 é vasta e complexa, possuindo diversas faces, de modo que é impossível resumir tal tecnologia em poucas páginas. Por isso, vamos apenas sobrevoar a área, a título de reconhecimento, priorizando os aspectos voltados a vídeo e áudio para a Internet. Enquanto os padrões MPEG-1 e MPEG-2 focavam métodos de compressão e descompressão de áudio e vídeo, o MPEG-4 é baseado em "objetos audiovisuais", o que permite a criação de conteúdo com elementos interativos para Internet, PDAs, celulares e outros dispositivos. Por isso, é bem possível que nos próximos meses comecemos a ver uma grande variedade de produtos que ofereçam serviços "movidos" a MPEG-4.

O MPEG-4 se ajusta automaticamente às condições sob as quais está rodando

Como já foi dito, o formato MPEG-4 (também chamado de "MP4") é baseado no QuickTime, para a alegria de nós, macmaníacos. O padrão também toma emprestadas as *hint tracks* do QuickTime, separando a informação sobre o que está sendo transmitido via streaming do conteúdo propriamente dito. O MPEG-4 expande esse suporte para múltiplas *hint tracks*, de modo que um arquivo pode definir como será o streaming numa grande variedade de ambientes.

O MPEG-4 também resolve alguns problemas ainda pendentes no QuickTime, podendo se ajustar dinamicamente à velocidade de conexão do usuário. Além disso, o padrão possui especificações de servidor bem organizadas, garantindo a difusão do streaming por cabo ou sem fio *(wireless)*.

Outra diferença do MPEG-4 para o QuickTime 5 é o suporte a streaming não apenas de áudio e vídeo, mas também de conteúdo sintético, como o mecanismo *text-to-speech*, texturas 3D e outros. Também inclui suporte a proteção e gerenciamento de direitos autorais (para infelicidade de muitos) e comunicações seguras para aplicações *pay-per-view*.

A implementação do MPEG-4 em tocadores de mídia dependerá de cada desenvolvedor, que pode interpretar o padrão como quiser e implementar diferentes subconjuntos de recursos, uma vez que a especificação é muito ampla. Assim, é bem possível que logo comecemos a ver conteúdos MPEG-4 que rodam no QuickTime Player mas não no Windows Media, e vice-versa.

Vídeo

O MPEG-4 traz muitos avanços em relação aos codecs atuais, principalmente no que se refere a dados com baixa taxa de transferência (20 kbps a 1000 kbps), garantindo qualidade muito superior ao MPEG-1. E, diferentemente da maioria dos codecs para a Web, ele tem suporte completo a vídeo entrelaçado, resoluções de até 4096x4096 pixels e transferência de dados de 5 kbps a 10 Mbps – ou seja, prepara terreno para aplicações que vão do telefone celular até a HDTV.

O codec de vídeo MPEG-4 suporta nativamente canais alfa (máscaras de recorte), de modo que o conteúdo pode ser separado internamente em elementos de frente e de fundo em cada cena. É claro que essa segmentação, embora melhore bastante a qualidade em determinadas faixas de *bitrate*, é mais complexa durante a codificação e o *playback*. Para entender a vantagem da segmentação, imagine um apresentador de TV em frente a um cenário. Com um codec convencional, a informação visual do cenário teria que ser retransmitida sempre que o apresen-

tador saísse do lugar. Com a imagem de fundo segmentada isso não acontece, pois o codec sabe que o cenário está lá o tempo todo. Para completar, o MPEG-4 é capaz de comprimir imagens paradas cerca de 25% a mais do que o padrão JPEG.

Áudio

O MPEG-4 oferece diversos recursos de áudio, com diferentes codecs para diferentes aplicações. Por exemplo, o Harmonic Vector Excitation Encoding (HVXC) é usado para *bitrates* entre 2 e 4 kbps, o CELP para de 4 a 24 kbps, o AAC para MPEG-2 e o TwinVQ para uso geral de banda larga.

Todas as mídias

Para o padrão MPEG-4, o vídeo não é composto apenas de amostras de áudio e quadros de imagem. Tudo é construído a partir de *objetos de mídia*, que podem ser imagens paradas, texto, voz sintetizada, modelos 3D etc. E o mais interessante é que se pode mapear qualquer uma das mídias suportadas para qualquer objeto numa cena. Assim, por exemplo, o áudio pode ser mapeado em um objeto com atributos para posicionar esse som em qualquer ponto num espaço tridimensional.
Para completar, as mídias são interativas, de modo que o MPEG-4 acaba unindo o melhor do Shockwave, Flash, VRML e vídeo tradicional em apenas um formato, servidor e cliente. As possibilidades são ilimitadas. Um dos protocolos do MPEG-4 permite até controlar a animação facial de um modelo 3D em tempo real. Combinando isso com a voz sintetizada via conversão *text-to-speech*, é possível obter perfeita sincronização labial. Porém, é preciso ter em mente que o modelo 3D não está especificado: apenas o protocolo para controlá-lo.

Versátil até demais

Não sei se você percebeu, mas a especificação MPEG-4 é tão rica que dificilmente haverá um conjunto de ferramentas de autoria que suporte todas as possibilidades. O mais provável é que seja necessário combinar diversos produtos para poder cobrir pelo menos uma parte das possibilidades. No entanto, resta saber de que modo o MPEG-4 será aplicado. Para a Internet, tudo depende de como a Apple, a Microsoft, a RealNetworks e – o mais importante – os usuários irão se posicionar em relação ao assunto. E tenha certeza de que muitas pedras ainda rolarão. **M**

MÁRCIO NIGRO
Concorda com o Guilherme Arantes: "no final será o que não sei, mas será".

Metendo a mão no Unix

Parte 8: introdução ao shell

por Alberto V. Mendonça

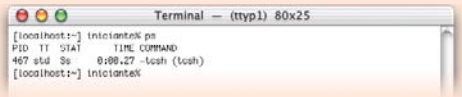
Atenção!

Tome muito cuidado ao utilizar os comandos `defaults` e `niutil`. Lembre que fazer alterações em arquivos do sistema, sem prévio conhecimento do que está sendo feito, pode vir a causar sérios problemas no funcionamento do seu Macintosh.

Ao se deparar com o assunto Unix, você deve ter lido muitas vezes a palavra *shell*. Na Macmania 85 demos uma rápida explicação sobre o shell no Mac OS X.
O shell é definido como o *interpretador de comandos* do Unix – o responsável por converter o vocabulário utilizado pelos seres humanos em instruções que o computador entenda.
Por ficar entre o usuário e o núcleo do sistema operacional *(kernel)*, formando uma concha *(shell* em inglês), ele serve como interface para o usuário dizer, em forma de linha de comando, o que o computador deve fazer. Ou seja, tem a mesma função que os comandos de mouse e teclado na interface Aqua, só que é muito mais poderosa e flexível.
Por que mais poderosa? Porque, além de ser um interpretador de comandos, o shell também é uma interface para linguagem de programação, o que permite criar sequências de comandos que serão executados na ordem em que foram escritos – ou seja, *scripts*.
Todos os sistemas Unix possuem o shell C `csh` (desenvolvido pelo mesmo autor do editor de texto vi, que conhecemos nas duas lições anteriores) e seu antecessor, o shell Bourne (`sh`). Em alguns casos ainda encontramos uma versão mais recente do shell Bourne, denominado Korn (`ksh`). No Mac OS X, além do shell C e do Bourne, temos como padrão o shell baseado no Terminal C (`tcsh`) e ainda encontramos o shell `zsh`, muito parecido com o ksh.
Na Macmania 90 mostramos como alterar o shell padrão, assim como instalar e utilizar no Mac OS X o `bash` (Bourne Again Shell), muito popular no Linux.
Curiosidade: o `ksh` é padrão no sistema operacional UNIX da IBM, chamado AIX, utilizado por muito tempo nos computadores IBM baseados no processador PowerPC – digamos assim, primos distantes dos nossos Power Macs...

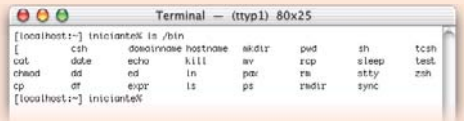
Passo 1

Vamos aprender a identificar o shell que estamos utilizando. Uma forma simples é verificar seu prompt. Se o seu prompt contém um %, provavelmente você está utilizando tcsh, mas se contém um $, você pode estar usando Bourne, Korn ou um de seus derivados.
Outra forma simples e confiável de checar o shell que você esta usando é perguntar ao sistema operacional o programa que você está usando no momento, com o comando `ps`:



Passo 2

Agora que sabemos reconhecer o shell que estamos utilizando, pode ser que nos interesse utilizar outro shell.
Para alterarmos o shell que estamos utilizando, precisamos primeiro saber os demais shells que estao disponíveis no sistema. Para isso, vamos dar uma olhada no diretório `/bin`. Podemos reconhecer os shells pelo "`sh`" em algum lugar de seus nomes. Assim sendo, verificamos que o Mac OS X tem os shells `csh`, `sh`, `tcsh` e `zsh`, como citado anteriormente.

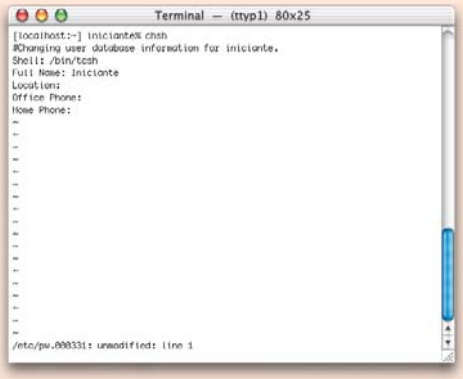


Passo 3

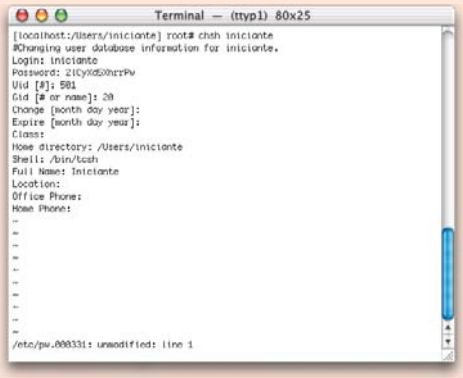
Para alterarmos o shell que utilizamos, existe o comando de UNIX `chsh` *(change shell)*. No entanto, no Mac OS X esse comando, no lugar de permitir a alteração imediata do shell, permite a visualização das configurações do usuário atraves do vi. O comando é utilizado da seguinte maneira: `chsh <nome_do_usuário>`, não sendo obrigatório o nome do usuário para visualizar as informações do usuário atual. Digite `chsh` e você terá o arquivo de texto que guarda as suas configurações de usuário, que ▶

Metendo a mão no Unix continuação

entre outras coisas inclui seu shell padrão.



Estando conectado como usuário *root*, as informações são ainda mais detalhadas:



Utilizando os comandos do vi, que você já conhece, podemos alterar o shell do usuário iniciante de `tcsh` para `sh`. Mas uma outra caracteristica do Mac OS X é que aparentemente ele não nos permite efetuar alterações nesses arquivos manualmente, utilizando os comandos do vi. Ao executarmos as alterações, quando finalizada a operação de modificação do shell, o sistema exibe uma mensagem como se a alteração nao tivesse sido realizada.

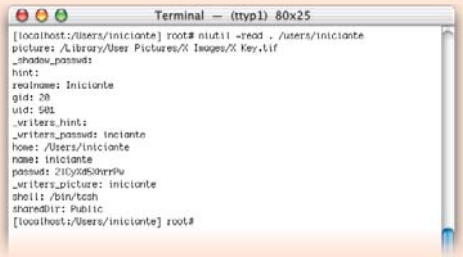
```
chsh: rebuilding the database...
chsh: done
chsh: /etc/master.passwd: unchanged
```

No entanto, ao acessarmos o arquivo `/etc/passwd` ou através do usuário *root*, a alteração constará no arquivo `/etc/master.passwd`, onde são armazenadas informações sobre cada usuário.

Mesmo assim, entramos em outro ponto característico do Mac OS X. As alterações realizadas em alguns arquivos de configuração da base UNIX do sistema não se sobrepõem às configurações específicas do Mac OS X.
Para que possamos realmente alterar o shell, precisamos alterá-lo nos dados do **NetInfo**.
E para isso temos que estar como usuário *root* e utilizar o comando `niutil`.
Basicamente, o comando possui a seguinte forma:

```
niutil <subcomando> <domínio> <diretório>
<palavra_chave> <valor>
```

Mas para maior compreensão do comando utilize `man niutil`.
Vamos, através de alguns subcomandos do `niutil`, visualizar as características do usuário `iniciante`.



Ou, ainda mais especificamente:

```
[localhost:/Users/iniciante] root# niutil -readprop .
/users/iniciante shell
/bin/tcsh
```

Para alterarmos o shell do usuário, precisamos utilizar outros subcomandos. Digite:

```
[localhost:/Users/iniciante] root# niutil -appendprop .
/users/alberto shell /bin/sh
[localhost:/Users/iniciante] root# niutil -destroyval .
/users/alberto shell /bin/tcsh
```

O primeiro comando é para adicionar o shell `sh` para o usuário em questão e o segundo para remover o shell `tcsh`.
Retorne à configuração original e prossiga.

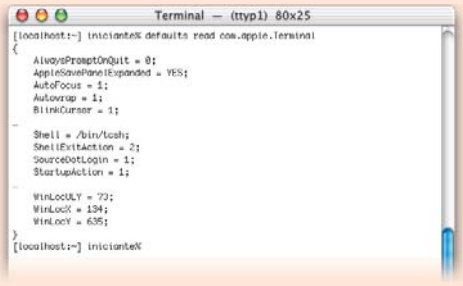
Passo 4

Nesse caso em específico, para alterarmos o shell imediatamente, alteraremos a configuração do Terminal, através do comando `defaults`. Esse comando permite que você leia, altere ou apague configurações do Mac OS X através do shell. Basicamente, o comando possui a seguinte forma:

```
defaults <subcomando> <arquivo_configuração>
<palavra_chave>
```

Como não apresentaremos todos eles nesta lição, utilize `man defaults` para obter maiores informações sobre cada subcomando e as suas utilizações.

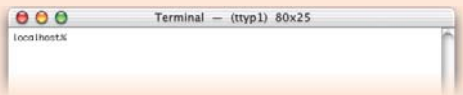
As configurações do shell estão contidas dentro das configurações do aplicativo Terminal, e por isso trabalharemos com o arquivo de configuração desse aplicativo. Vamos utilizar o comando `defaults`, inicialmente para olhar o que temos no arquivo de configuração do Terminal. Digite o subcomando `read <arquivo_configuração>`:



Nota – Arquivos de configuração como esse do Terminal estão localizados nos diretórios `~/Library/Preferences` e `/Library/Preferences`, seguindo o padrão de nomenclatura `com.apple.<Nome>.plist`. Nem todos os arquivos de configuração são editáveis pelo comando `defaults`.
Note que temos a linha iniciada pela palavra `Shell` seguida do shell que estamos utilizando no momento (`/bin/tcsh`). Para alterar essa linha vamos utilizar outro subcomando do defaults. Digite o subcomando `write <arquivo_configuração> <palavra_chave> '<valor>'`:

```
[localhost:~] iniciante% defaults write
com.apple.Terminal Shell /bin/sh
```

Assim, escrevemos na linha `Shell` que nosso novo shell é `sh`, como você pode conferir agora utilizando novamente o subcomando `read`.
Após verificada a alteração no arquivo de configuração, é possível passar a utilizar o novo shell abrindo uma nova janela de Terminal (⌘N). Note que na nova janela temos um *prompt* diferente daquele que estamos acostumados a visualizar em nossas lições:



É recomendável que você retorne agora ao shell `tcsh` (padrão), para que possamos prosseguir com as próximas lições sobre shell, onde nos familiarizaremos ainda mais com nossa interface "não-gráfica".
Nota – Na Macmania 85 você tem outros exemplos de como pode ser utilizado o comando `defaults`. Tente utilizar o que aprendeu nesta lição para entender melhor o que mostramos naquela. M

ALBERTO V. MENDONÇA