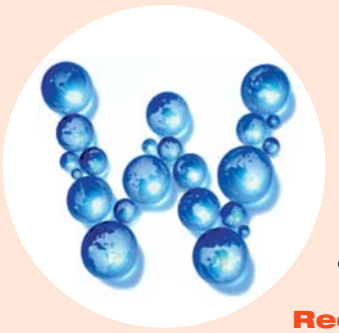
MacPRO
o suplemento dos power users
Ano 4 - Nº38
Parte integrante da revista Macmania
Não pode ser vendido separadamente

Quem tem medo do WebObjects?

Parte 2: Direct to Web



por Tiago Ribeiro e Fabio Ribeiro

Na primeira parte deste artigo, você viu o que é o WebObjects e para que ele serve. Agora, começaremos a mostrar um pouco mais sobre o que é, na prática, o desenvolvimento de uma aplicação Web com essa tecnologia, a partir do ponto zero – ou seja, a partir do momento em que instalamos o software em nossa máquina.

O que vamos fazer

Hoje começaremos a explorar duas tecnologias incorporadas ao WebObjects que permitem criar aplicações cliente/servidor sem a necessidade de incorporar uma só linha de código. São elas: **Direct to Web** e **Direct to Java Client**.

Para isso, vamos usar a título de exemplo o projeto de uma agenda de contatos de clientes que queremos disponibilizar através da Web ou através de um software que nossos usuários possam utilizar via rede. Essa agenda permitirá a inclusão e remoção de contatos e pesquisa por nome, email ou telefone, bem como a edição dos dados já inseridos.

Requisitos

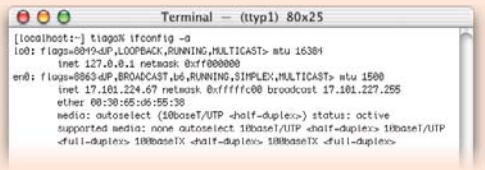
Para realizar os exercícios que apresentaremos neste artigo, você precisará ter, no mínimo, os seguintes softwares instalados:

- Mac OS X 10.x, Mac OS X Server 10.x ou Windows 2000.
- WebObjects 5 Developer.
- Navegador Web padrão.

Além disso, seu computador precisa estar configurado para ter acesso à rede (isto é, você deve ter pelo menos uma interface de rede ativa). Você pode verificar isso digitando o seguinte comando em uma janela do Terminal (/Applications/Utilities/Terminal.app):

```
ifconfig -a
```

No meu computador, o comando acima retorna as seguintes informações:



Certifique-se de que o item status esteja assinalado como active.

Iniciando

Conforme já tínhamos visto anteriormente, o desenvolvimento de uma aplicação com o WebObjects passa, inicialmente, pela criação de um modelo que represente as entidades envolvidas em nossa aplicação. Em uma aplicação de *e-commerce*, essas entidades podem ser o carrinho de compras, o cliente, o produto e assim por diante. Ao mesmo tempo, o modelo deve permitir a sua *persistência*, ou seja, permitir que as informações relevantes sejam armazenadas em um banco de dados para acesso futuro.

Ao contrário do que muitas pessoas acham, o WebObjects não é um Servidor de Bancos de Dados; pelo contrário, ele fornece uma interface para conexão com qualquer Servidor de Bancos de Dados compatível com a tecnologia JDBC (entre eles, os servidores Oracle, Sybase, Informix, PostgreSQL, OpenBase, FrontBase, Primebase etc.). Para o exemplo de aplicação que criaremos, usaremos um banco de dados cuja licença de desenvolvimento e respectivos softwares já vêm pré-instalados no pacote do WebObjects: **OpenBase SQL**.

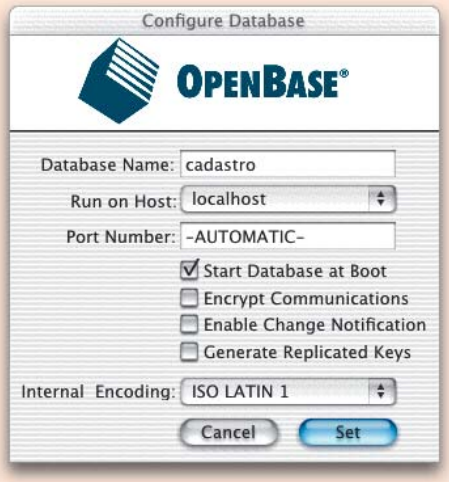
O primeiro passo, portanto, é criarmos no OpenBase um novo banco de dados, que será a base para a nossa aplicação.

Fase 1: OpenBaseManager

1 Abra o aplicativo OpenBaseManager em /Applications/OpenBase/OpenBaseManager.app.

2 No menu, selecione a opção Database ▸ New (⌘ Shift N).

3 Na janela que se abre *(Configure Database)*, vamos criar um novo banco de dados, que será utilizado para guardar as informações sobre nossa aplicação. O nome desse banco de dados será `cadastro`. Além disso, certifique-se de que a opção "Start Database at Boot" esteja selecionada; assim, o servidor irá inicializar o banco de dados toda vez que a máquina for ligada.



4 Na janela OpenBase Manager, você verá que, dentro da lista de bancos de dados existentes no banco de dados local (`localhost`) irá aparecer o banco de dados que acabamos de criar. Selecione-o e clique no botão Start para que o banco de dados seja inicializado. Uma vez criado e inicializado o banco de dados, precisamos agora informar que tipos de informação desejamos guardar dentro dele. É aí que entra em cena a primeira ferramenta de desenvolvimento do pacote do WebObjects: o EOModeler.

Fase 2: EOModeler

O EOModeler é a aplicação que nos permite transformar as informações pertinentes aos diferentes agentes que compõem o nosso aplicativo Web (por exemplo, o nome e endereço de email de nossos clientes) em dados persistentes – ou seja, informações que, uma vez criadas, possam ser salvas para posterior consulta e/ou edição. Para isso, esta aplicação se utiliza do framework EOF **(Enterprise Objects Framework)**, que permite, entre outras coisas, que nossa aplicação saiba onde e como salvar os dados que venham a ser fornecidos pelo usuário.

No nosso exemplo, armazenaremos um único tipo de informação (no caso, os dados dos clientes que queremos cadastrar no nosso sistema). Todo cliente possui informações vitais que queremos guardar como, por exemplo:

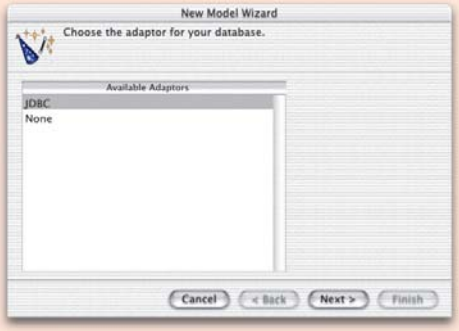
- Nome e sobrenome.
- Endereço de email.
- Telefone para contato.
- Cidade de origem.
- Etc.

Em programação orientada a objetos, as entidades que compõem e interagem com a aplicação (clientes, carrinhos de compras, pedidos, produtos etc.) recebem o nome de *objetos*. As informações relativas a esses objetos são conhecidas como *propriedades*. Assim, poderíamos dizer que `nome` e `sobrenome` são propriedades inerentes ao objeto `cliente`. É exatamente isso o que estamos prestes a fazer com o EOModeler.

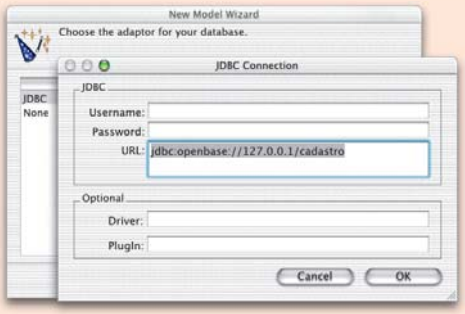
1 Abra o aplicativo EOModeler em `/Developer/Applications/EOModeler.app`.

2 No menu File, selecione a opção New Model (⌘N).

3 Aparecerá o assistente de criação do modelo; na primeira tela, ele irá solicitar que você especifique qual tipo de adaptador será utilizado para acesso ao banco de dados. Selecione a opção "JDBC" (Java DataBase Connectivity).

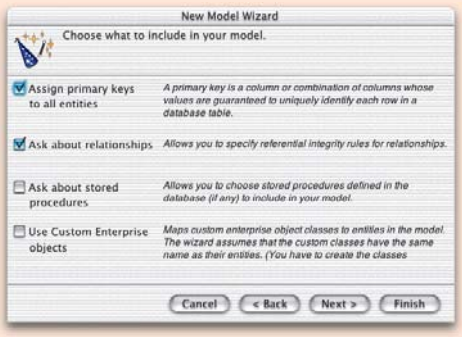


4 Na próxima tela, o Assistant irá solicitar que você especifique qual a base de dados que será utilizada em sua aplicação. No campo "URL", digite `jdbc:openbase://127.0.0.1/cadastro`.

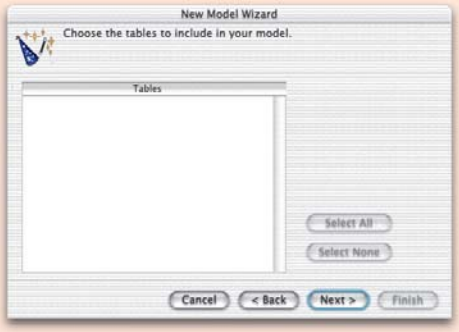


A URL é composta pelo protocolo a ser utilizado para comunicação (`jdbc`), pelo servidor de bancos de dados a ser utilizado (`openbase`), pelo endereço de IP da máquina na qual o servidor de bancos de dados está rodando (127.0.0.1 é o endereço *lookup*, que aponta para a máquina na qual estamos trabalhando), e pelo nome do banco de dados a ser usado (no caso, o banco "cadastro" criado na fase 1 deste exercício). Para este exemplo, não é necessário preencher os demais campos. Clique no botão "OK".

5 Na próxima tela, deselecione as duas últimas opções ("Ask About Stored Procedures" e "Use Custom Enterprise objects"). Clique no botão "Next".



6 A seguir, o Assistant solicitará que você determine quais tabelas já existentes no banco de dados serão utilizadas por sua aplicação. Como estamos criando o banco de dados a partir do zero, não existe ainda nenhuma tabela disponível. Por isso, clique no botão "Next".



7 Finalmente, clique no botão "Finish" para que o EOModeler abra a janela de edição de nosso modelo.

8 O que faremos a partir de agora é criar um objeto que represente o cliente que cadastraremos em nosso banco de dados. Além dele, também criaremos os campos (propriedades) que irão armazenar informações inerentes a esse cliente: nome, sobrenome, ID (código identificador), email e assim por diante.

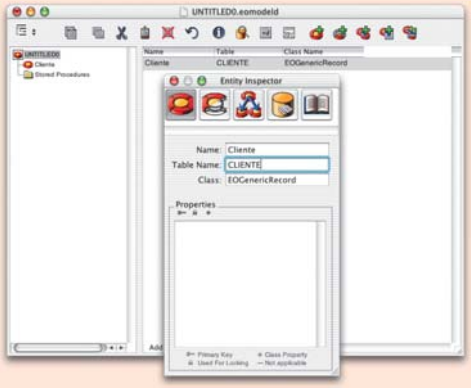
No menu Property, selecione a opção Add Entity (⌘⇧E). Isso irá criar uma nova entidade (objeto), cujo nome inicial é `Entity`.

9 Vamos editar as informações sobre o objeto que estamos criando; mantendo selecionada a entidade que acabamos de criar, use o atalho de teclado ⌘1. Uma nova janela (Entity Inspector) irá se abrir. Nela, digite no campo "Name" um novo nome para nosso objeto (vamos usar `Cliente` com "C" maiúsculo); no campo "Table Name", escolha o nome

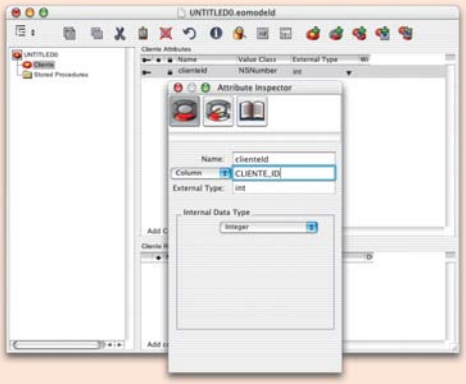
Propriedades para a entidade "Cliente"

Name	Column	External Type	Internal Data Type	Primary Key	Propriedade
clienteId	CLIENTE_ID	int	*integer*	•	
nome	NOME	char	*String* (50 caract.)		•
sobrenome	SOBRENOME	char	*String* (50 caract.)		•
email	EMAIL	char	*String* (50 caract.)		•
telefone	TELEFONE	char	*String* (50 caract.)		•

que vai ser utilizado para representar esse nosso objeto dentro do banco de dados (vamos usar `CLIENTE` – note o uso da caixa alta). Uma vez feito isso, feche o Inspector.



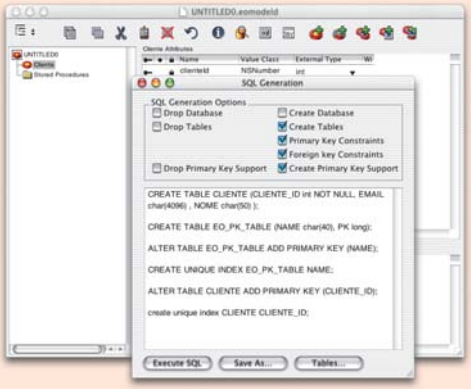
10 Ainda mantendo selecionada a entidade `Cliente` na janela principal do EOModeler, selecione o menu Property ▸ Add Attribute (⌘Shift A). Selecione o item que acabamos de criar na lista de atributos (Properties) e clique na coluna com o símbolo de chave (primeira coluna) para que esse item se torne uma *chave primária*, ou seja, um identificador que diferencie cada cliente que estamos cadastrando – como se fosse o seu RG. Desselecione o losango na segunda coluna – ele indica se o atributo será ou não considerado uma propriedade da entidade "cliente" (ou seja, será incluído como opção nas buscas), e não queremos isso.



11 Mantendo ainda esse atributo selecionado, abra o Inspector (usando ⌘1) e altere as informações "Name" para `clienteId`, "Column" para `CLIENTE_ID`, "External Type" para `int` e "Internal Data Type" para "integer".
A seguir, feche o Inspector.

12 Repita os passos 10 e 11 para as demais propriedades do objeto `Cliente`, completando as informações de acordo com os dados da tabela acima, até que todas as propriedades tenham sido criadas.

13 Selecione o menu Property ▸ Generate SQL para que o EOModeler crie, no Servidor de Bancos de Dados, as tabelas que irão armazenar as informações de seus clientes. Na janela que aparece, acione apenas os *checkboxes* "Create Tables", "Primary Key Constraints", "Foreign Key Constraints" e "Create Primary Key Support". Clique no botão "Execute SQL" e, caso não ocorra nenhum erro, feche a janela e salve o arquivo no seu disco (⌘S). ▸

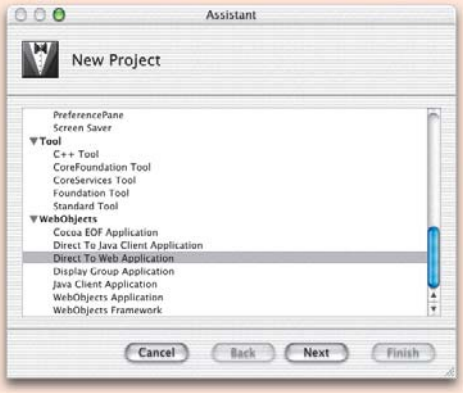


Fase 3: Project Builder

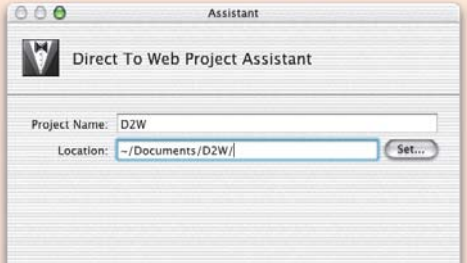
Uma vez criados os componentes necessários para a existência de nossa aplicação, vamos criar a própria aplicação. Para isso, abra a ferramenta **Project Builder**, em "`/Developer/Applications/Project Builder.app`". O Project Builder é o software responsável pelo gerenciamento do projeto de nossa aplicação Web, incluindo o acesso aos arquivos de interface (via WebObjects Builder), código e modelos (via EOModeler). Além disso, permite a codificação através de um editor integrado, a compilação e a realização de testes de execução (depuração).

1 Selecione no menu File ▸ NewProject (⌘⇧N).

2 Selecione, na janela de assistente que se abre, que tipo específico de aplicação iremos criar. Selecione a opção "WebObjects/Direct to Web Application" e clique no botão "Next".

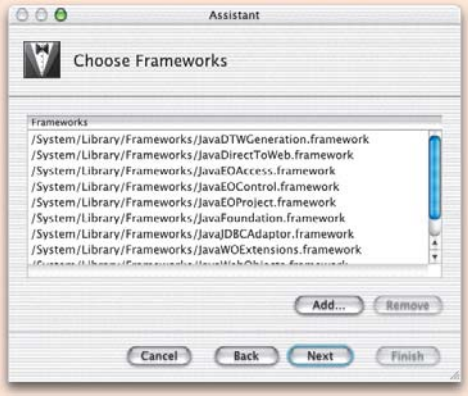


3 A seguir, vamos definir o nome de nosso projeto e o local para salvá-lo. No campo "Project Name", digite um nome qualquer para nosso projeto (por exemplo, `D2W`). No campo "Location", defina o diretório para onde serão copiados os arquivos de seu projeto, incluindo o nome do projeto.

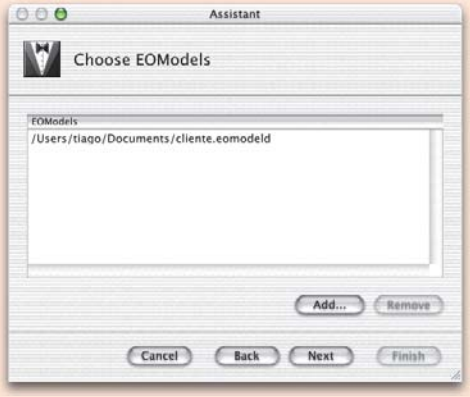


Por exemplo, `~/Documents/D2W` irá salvar os seus arquivos de projeto na pasta `D2W` dentro da pasta `Documents` de seu diretório Home (`~`). Clique no botão "Next".

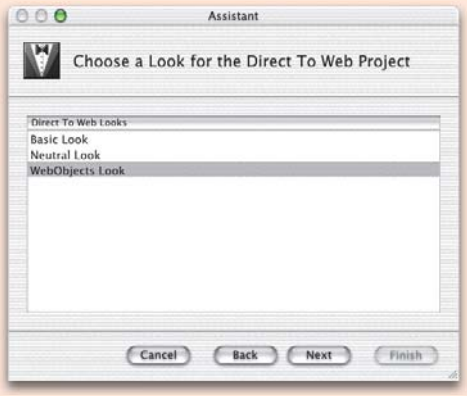
4 A janela seguinte do assistente permite a inclusão de *frameworks* adicionais (além dos já fornecidos pelo WebObjects) dentro da sua aplicação. Simplesmente clique em "Next".



5 A seguir, você irá incluir dentro do projeto o arquivo do EOModeler que criamos na fase 1. Clique no botão "Add" e localize o arquivo que criamos (com a extensão `.eomodel`). Uma vez incluído, clique em "Next".



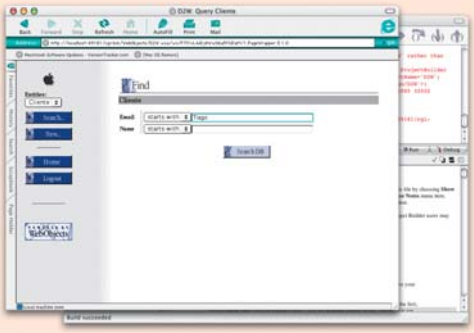
6 Em seguida, escolha qual *look and feel* (aparência de interface) será utilizada na aplicação. Você pode escolher um dos estilos existentes e criar o seu depois. Vamos selecionar a opção "WebObjects Look".



7 Ative o *checkbox* "Build and launch project now" para que o projeto seja compilado e executado imediatamente. Clique em "Finish".

Construindo e executando

Uma vez criado nosso projeto Direct to Web, basta compilar e executar a aplicação; se você seguiu todos os passos até aqui, o Project Builder se encarregará de fazer isso automaticamente. Após alguns segundos compilando e iniciando o servidor de aplicações (você verá uma série de informações na tela do Project Builder), ele irá abrir o navegador Web padrão e conectá-lo à aplicação que você acabou de criar. Clique no botão "Login" (não é necessário entrar com nome de usuário ou senha) e experimente criar novos clientes, executar buscas, editar dados etc. Você acabou de criar uma interface Web para seu banco de dados, sem a necessidade de escrever nenhum código – isso foi feito automaticamente por você!



WebObjects não é apenas para a Web

A flexibilidade do WebObjects permite não apenas a criação de aplicações HTML (baseadas em um navegador Web), mas também aplicações Java, que podem ser executadas a partir de qualquer computador (Mac, PC, workstation) compatível com a plataforma Java 2, Standard Edition (J2SE). Vamos ver como criar esse tipo de aplicação na próxima parte deste artigo. Não perca! M

TIAGO RIBEIRO tiago.r@apple.com.br
FABIO RIBEIRO fabio.g@apple.com.br
Trabalham na ADC da Apple Brasil.

Metendo a mão no Unix

Você já ouviu dizer que o Mac OS X é um sistema *multiusuário*, certo? Mas o que isso pode significar?
Nos sistemas multiusuário, além de um ambiente gráfico diferenciado para cada um, você pode ter vários usuários trabalhando na máquina ao mesmo tempo, cada um executando uma tarefa.
A grande maioria dos sistemas Unix trabalha sem interface gráfica, utilizando apenas a linha de comando. Mas no Mac OS X, por termos sempre a interface Aqua funcionando, a maneira mais comum de vários usuários trabalharem simultaneamente na mesma máquina é utilizar o acesso remoto pela linha de comando (que pode ser habilitado no painel System Preferences, seção Sharing, aba Application, opção "Allow Remote Login").
Nesta lição não explicaremos como trabalhar com o acesso remoto através da linha de comando, pois liberar esse tipo de acesso pode trazer problemas a administradores e usuários novatos, principalmente ataques de hackers e vândalos da Internet *(crackers)*.
Conheceremos inicialmente os comandos básicos para saber quem possui acesso, quem está conectado e o que cada usuário está fazendo na sua máquina. Enfim, terminaremos com comandos que ajudam a saber o espaço ocupado por arquivos, diretórios e o espaço livre no seu HD.

Passo 1

Um administrador de um sistema multiusuário precisa estar muito atento a esse ponto. Quem possui acesso ao sistema, e o que essas pessoas andam fazendo?
Inicialmente veremos um conjunto de comandos que não são exatamente do Unix, mas específicos do Mac OS X. Eles utilizam o aplicativo NetInfo Manager (situado em "/Applications/Utilities/NetInfo Manager") através da linha de comando.
Para ter uma lista dos usuários cadastrados no sistema, damos o comando `niutil -list . /users`:

Você pode verificar que aparecem alguns usuários estranhos logo de início; são os usuários originais do Mac OS X, adicionados automaticamente na instalação do sistema.
Os unicos usuários que nós adicionamos são `iniciante` e `meuvizinho`.
Para ter uma relação dos grupos existentes no sistema, utilizamos o mesmo comando, mas especificando `groups` no lugar de `users`:

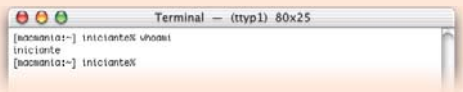
```
niutil -list . /groups
```

Uma outra variação, não menos importante, serve para conhecer as impressoras configuradas no nosso sistema:

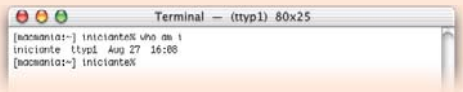
```
niutil -list . /printers
```

Passo 2

Agora temos um leve conhecimento sobre quais pessoas e grupos estão trabalhando em nosso sistema. Não ainda o suficiente para termos controle sobre o que está acontecendo.
Voltando aos comandos Unix: vamos tratar de descobrir quem é você? Isso mesmo; nada mais importante do que saber quem é você, para depois conhecer melhor os demais usuários do seu sistema.
A forma mais simples de descobrir isso é com o comando `whoami`:

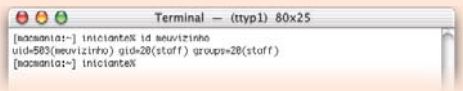


Os comandos do Unix costumam ser muito sensíveis e podem mudar de acordo com o espaçamento digitado; até mesmo letras maiúsculas e minúsculas são consideradas diferentes. Um bom exemplo disso, que podemos dar aqui, é o comando `who am i`, que é muito parecido com o anterior, mas traz uma diferença no resultado:



Em qualquer uma das formas acima temos a resposta de que somos o usuário `iniciante`, que é o resultado que buscávamos. Porém, podemos ter um resultado ainda mais completo, utilizando o comando `id`. Observe:

Também utilizamos o comando `id` para saber os mesmos dados sobre os demais usuários cadastrados no nosso sistema. Faça um teste com outro usuário que você encontrou com o primeiro comando desta lição, `niutil`:

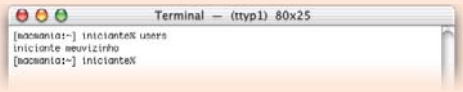


Através desse comando temos não apenas o nome do usuário como também o seu ID (`uid`). Assim como o nome, o ID é único (não há outro igual no mesmo sistema). O comando também apresenta os grupos aos quais o usuário está ligado e o respectivo ID de cada um (`gid`).

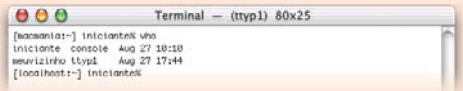
Importante! Na Lição 4 aprendemos que um usuário tem seus privilégios definidos individualmente e de acordo com os grupos aos quais ele pertence. Ou seja, o usuario `iniciante` tem os privilégios de acesso definidos pelos seus privilégios pessoais e aqueles definidos para os usuarios dos grupos `staff`, `wheel` e `admin`. Dessa forma, os privilégios de acesso a arquivos ou diretórios dados aos grupos dos quais o usuário faz parte também são válidos para esse usuário.
É muito importante aprender não só a utilizar os comandos, mas compreender os resultados apresentados e inter-relacioná-los para chegar a conclusões ainda mais significativas.

Passo 3

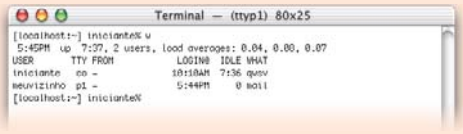
Vamos supor que você tenha habilitado o acesso remoto ao sistema por linha de comando. E agora, como sabemos quem está conectado e o que está sendo feito? Esse é um passo importante para ter controle de um sistema multiusuário. Para isso temos comandos simples, como o `users`:



No caso acima, temos uma simples relação dos usuários que estão trabalhando no seu sistema. Para ter dados mais detalhados, utilizamos um comando que, além do nome do usuário, informa a que horas ele começou a trabalhar no computador e em qual linha de comando (terminal) ele se encontra.
O comando é `who`:



Podemos ver que estamos trabalhando diretamente a partir do console e desde as 10:10 horas, enquanto o usuario 'meuvizinho' está conectado na linha de comando ttyp1, desde às 17:44 horas. Mas o que eles estão fazendo? Poderiam estar executando algo que possa prejudicar seu sistema? Para isso existe um comando ainda mais poderoso, `w`:



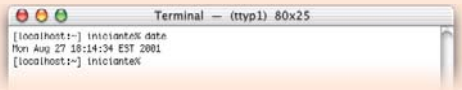
Esse comando é mais complexo, pois oferece diversas informações adicionais. A primeira linha informa a hora atual, seguida pelo tem-

por Alberto V. Mendonça

po desde a última vez que o computador foi iniciado e o número de usuários trabalhando no sistema. Em seguida temos os dados de cada usuário, onde temos, além daquelas obtidas como o comando 'who ', informações sobre a partir de onde ele está se conectando, quanto tempo está inativo e que programas está executando. No caso do usuário iniciante ele está conectado pelo console, diretamente na máquina onde roda o sistema, desde as 10:10 da manhã, sem executar tarefas há 7:36 minutos e executando o qwsv (servidor do jogo Quake World).

Passo 4

Dois comandos rápidos e simples, que também podemos utilizar em diversas situações, são os de verificação de data e hora. Isso pode ser importante, pois a máquina à qual você está conectado pode estar em um fuso horário diferente do seu, ou mesmo com a data e hora errados. Para saber a data e a hora atual no sistema, utilize o comando `date`:

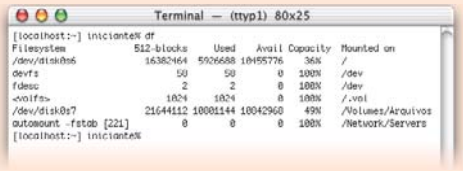


Passo 5

E como saber quanto cada diretório e arquivo ocupa no seu HD? Ou quanto espaço livre existe no HD?
Descobrir o espaço total disponível em um HD não é uma tarefa fácil para iniciantes no Unix, mas tentaremos ajudá-los.
Para isso iremos utilizar em primeiro lugar o comando `du`, mas prestando bastante atenção em alguns detalhes.
O comando `du` apresenta uma lista do tamanho (em kilobytes) de *todos* os diretórios no sistema, a partir da sua localização atual. É uma lista imensa com mais informação do que você necessita.

Para resolver esse problema, precisamos saber o que adicionar ao comando `du` para filtramos o resultado e obtermos só aquilo que procuramos.
Desde a Lição 1 estamos estendendo nossos comandos com o - (sinal de menos) seguido de algumas letras; essa extensão de comando a partir de agora denominaremos *flag*. Assim vamos aprimorando nosso vocabulário Unix, certo? Se você estiver interessado apenas no tamanho total ocupado por aquele diretório, pode usar o *flag* `-s`:

Existe um outro comando, o `df`, que apresenta vários dados sobre a utilização do HD:



Acima temos uma relação que mostra o nome do dispositivo de disco, o tamanho total do dispositivo, quanto está sendo utilizado, quanto está disponível, a porcentagem usada e o ponto no sistema de arquivo no qual o dispositivo está montado.
O que isso significa? Muita coisa... e quase nada, ao mesmo tempo! Você acaba obtendo muitas informações, mas não é fácil fazer a soma para encontrar rapidamente o espaço total disponível. E, mesmo após muita conta, você talvez não tenha informações precisas.
Apresentamos esses comandos com um outro intuito: ensinar como o Unix trabalha com os arquivos. O sistema de arquivos do Unix é como um gigantesco bloco de notas, repleto de folhas de informações.
Basicamente, um bloco equivale a uma folha de papel com informações no bloco de notas virtual representado pelo HD. Um disco é composto (tipicamente) de muitas dezenas ou centenas de blocos de informações, cada qual com 512 bytes. Cada um é como uma guia de índice, indicando o local onde o arquivo se inicia no bloco de notas e a forma como as muitas folhas são usadas. As guias são chamadas de *i-nodes* e a lista de guias (o índice do bloco de notas) é a *i-list*.
E assim finalizamos mais uma lição com novos comandos e novos termos do mundo Unix para os usuários do Mac OS X. Continue utilizando o comando `man` para ler o manual de cada comando no próprio sistema e boa sorte! **M**

ALBERTO V. MENDONÇA