FMA 215

Aula 6: Processos e Shell Jobs

Prof. Jorge L. deLyra

25 de Abril de 2002

1 Alguns Conceitos Relevantes

Nesta aula vamos lidar com os processos que rodam no sistema, bem como a submissão de “jobs” através da shell, ou seja, a execução de processos no “background”, sem que o seu terminal fique preso a ele. Passaremos também pelo conceito de “pipeline” de processos, que nos permitirá combinar diversos programas para obter novas ações sobre os elementos do sistema, mais variadas, mais complexas, para fins mais específicos. Desta forma estaremos dando uma primeira olhada em como as coisas *acontecem* no sistema. É claro que já estivemos usando estes elementos nas aulas anteriores, pois o seu uso é inerente à operação do sistema, mas nesta aula vamos nos concentrar em entender como eles funcionam.

Processo: um processo é uma instância de um programa que está rodando no sistema. Um mesmo programa pode estar rodando várias vezes ao mesmo tempo, em vários processos diferentes. Por exemplo, um mesmo editor de texto pode estar sendo usado por vários usuários ao mesmo tempo. Cada processo é identificado por um número único entre todos que estão rodando, ao qual chamamos de PID (process identification). Além disso, ele está sempre associado a um usuário, ao qual pertence. Ou seja, há uma noção de ownership para processos, assim como para arquivos.

Esta é uma noção de ownership do tipo do ownership de user para arquivos. Existe também um conceito de grupos para processos, mas ele não é muito importante no que diz respeito ao que nos interessa como usuários do sistema. Existe um outro tipo de relação entre processos, de caráter hierárquico tipo pais-e-filhos, que é mais relevante. Cada processo é "filho" (child) de algum outro processo, que chamamos de "pai" ou de parent-process. Isto estabelece um tipo de estrutura hierárquica entre os processos que, em geral, está ligada ao fato de que um processo inicia outros, que são denominados seus filhos (children). A raiz da árvore deste sistema hierárquico é o processo `init`, que tem número PID 1. Este é o processo inicial que inicializa todo o sistema quando ele é colocado no ar.

Não se deve confundir este processo inicial com o kernel. O kernel é de fato um programa que está rodando na máquina, mas ele não é um processo e sim o programa subjacente que controla todos os processos. Assim como ele controla os dispositivos de armazenamento e de entrada e saída, o kernel também controla a utilização da memória RAM e do CPU, submetendo a eles outros programas para serem executados. Estes outros programas aparecem como processos dentro da estrutura de controle estabelecida pelo kernel. O programa `init` é o primeiro que é executado pelo kernel logo após o boot deste, tendo a finalidade de colocar para rodar todos os outros programas que são necessários para que tenhamos um sistema operacional multi-user e multi-tasking completo em funcionamento.

Jobs: a shell pode rodar vários programas ao mesmo tempo para você. Ela tem um sistema de controle dos processos que tenham sido submetidos através dela, os quais são chamados de "jobs". O sistema de controle em si é denominado "job control". Com este sistema, pode-se iniciar, suspender e reiniciar processos, terminar processos submetidos anteriormente, desconectar e reconectar processos ao terminal e listar os jobs, ou seja, os processos que estão sob o controle da shell.

Esta capacidade da shell é muito útil para lidar com a situação na qual se está rodando um determinado programa e aparece a necessidade de se suspender momentaneamente esta atividade para executar alguma outra, voltando-se posteriormente a ela. Basta suspender o processo, o que libera o terminal para outras tarefas, reconectando posteriormente o terminal ao processo que foi suspenso. Este sistema também é útil quando é necessário deixar um ou mais programas rodando permanentemente sem conexão com o terminal, como é o caso, por exemplo, para um relógio.

Pipeline: o conceito de **pipeline** de comandos é um dos conceitos importantes do sistema. Uma pipeline é um processo pelo qual dois ou mais programas podem ser executados de forma coordenada, em uma determinada ordenação, de forma que o output de cada um é redirecionado como input do próximo. Assim, o conjunto dos programas que são executados desta forma passa a se comportar como um novo programa, com o input direcionado ao primeiro programa e o output vindo do último. Desta forma podemos efetivamente combinar a ação de vários executáveis binários de forma muito fácil e flexível, em vez de modificar e recompilar os respectivos programas, ou mesmo escrever novos programas para as tarefas que se queira realizar.

Em geral as shells permitem a montagem de pipelines de comandos, que podem ser usadas na linha de comando ou em programas de shell (shell scripts). A forma de se fazer isto é usar o caracter `|` para separar os diversos comandos, em ordem, na linha de comando. Por exemplo, para ver uma longa lista de arquivos, que não cabe no terminal, pode-se usar um paginador, um programa que mostra uma página de cada vez. Um exemplo disto é a combinação `ls -l /etc | more`, onde o comando `more` é o paginador. Com o uso deste programa pode-se passar de página em página do output do comando `ls -l` usando-se a barra de espaço do teclado.

2 Tarefas

Este conjunto de tarefas deve ser realizado no sistema de janelas X11, o que irá possibilitar o uso de âncoras visuais que facilitarão a compreensão dos elementos com os quais vamos lidar.

Para trabalhar com este tipo de interface gráfica, é necessário lembrar que existe nele o conceito de **foco**, como já vimos na aula anterior: a janela que tem o foco é aquela que recebe o que se digita no teclado. Para colocar o foco em uma determinada janela, a melhor forma é clicar o botão esquerdo do mouse na parte central da barra superior da janela, o que também a colocará acima de outras janelas que a estejam cobrindo parcialmente.

Às vezes pode ser necessário também movimentar uma janela para que ela fique mais visível. Para fazer isto, basta clicar e segurar o botão esquerdo do mouse sobre a parte central da barra superior da janela e movimentar o mouse, que carregará a janela consigo. A nova posição da janela será aquela na qual o botão do mouse for solto. Neste momento ela terá também o foco, ou seja, estará recebendo o input do teclado.

1. Faça um login no sistema, usando o sistema de janelas X11. Caso o computador esteja com o Linux rodando mas a bandeira de login do

sistema X11 não esteja visível, digite `[Ctrl]-[Alt]-F7`. Isto o levará de qualquer dos consoles virtuais para o sétimo, onde em geral estará rodando o sistema X11.

2. Quando você entrar, será aberto para você um terminal `xterm`, que aparecerá como uma janela em algum lugar da tela. Esta janela tem um papel especial em sua seção, se ela for eliminada toda a seção terminará e o X11 será reciclado, aparecendo novamente a bandeira de login. Para evitar que você termine sua seção acidentalmente, vamos evitar usar esta janela por enquanto.

   Abra duas outras janelas `xterm` para uso nestas tarefas. Para fazer isto, clique e segure o botão esquerdo do mouse sobre o fundo da tela. Aparecerá um menu, o primeiro ítem do qual abre um novo terminal. Leve o cursor do mouse para cima deste primeiro ítem e solte o botão da esquerda, o que causará o aparecimento de uma nova janela de terminal.

   Tendo aberto os dois novos terminais, iconize o terminal original, clicando o botão esquerdo do mouse no botão que contem um pequeno quadradinho, o penúltimo à direita na barra superior da janela. Organize os dois novos terminais na tela, mudando suas posições, até que se tenha boa visibilidade de ambas as janelas e que seja possível passar o foco facilmente de uma para a outra.

3. Execute, em uma das duas janelas, o comando `ps`. Você consegue reconhecer cada um dos processos que estão rodando e identificar a que janelas cada um está associado?

4. Execute, na outra janela, o comando `xclock`. Vai aparecer em sua tela um relógio. Observe que o terminal fica "preso", não responde mais ao seu input no teclado, não mostra mais o prompt quando você aperta a tecla `[Enter]`. Olhe os processos de novo, no outro terminal.

5. Coloque o foco no terminal onde você está rodando o relógio e digite `[Ctrl]-C`. Isto significa segurar a tecla `[Ctrl]` (control) enquanto você digita uma vez a tecla da letra `C` (ou `c`). Apesar de que em geral indicamos este tipo de coisa usando letras maiúsculas, não se deve apertar a tecla `[Shift]`. Caso isto seja necessário estará indicado explicitamente. Observe o que acontece com o relógio. Verifique a situação dos processos no outro terminal.

6. Execute mais uma vez, na mesma janela que usou antes, o comando `xclock`. Para desconectar o seu terminal do processo que foi iniciado, digite a combinação `[Ctrl]-Z`. Trata-se de um caracter de controle que faz com que um sinal seja enviado pela shell ao processo, provocando a sua suspensão temporária. Observe que você ganha de volta o acesso de input à janela. Verifique a situação dos processos no outro terminal.

7. Voltando ao terminal onde está rodando o `xclock`, execute o comando `jobs`. Ele irá mostrar todos os processos que estão sendo administrados pela shell que está rodando neste terminal. Examine o output do comando e veja se dá para entender o que aparece lá.

8. É possível rodar um programa de forma que ele já se inicie sem conexão com o terminal. Para fazer isto, na mesma janela que está usando para rodar o primeiro relógio, rode um segundo relógio usando o comando `xclock &`. Observe que desta vez o seu terminal ficou livre para outras tarefas.

9. Execute de novo o comando `jobs` e examine o resultado. Verifique a situação dos processos no outro terminal. Observe os dois relógios durante alguns minutos e verifique o que acontece com eles. Você sabe porquê os dois se comportam de forma diferente?

10. Para por para rodar de fato o job que está suspenso no background, como indicado no output do comando `jobs`, basta executar o comando `bg %<número-do-job>`. O número do job é o que aparece entre chaves no output do comando `jobs`. Execute de novo este comando, examine o resultado e verifique a situação dos processos no outro terminal. Espere mais alguns minutos e verifique como se comportam os relógios agora.

    Um processo estar no "background" significa que ele está rodando sob a administração da shell, sem, entretanto, que o sistema de input da sua sessão esteja ligado permanentemente a ele. Dizemos que um processo que esteja sendo rodado pela shell e que esteja ligado ao input do teclado está rodando no "foreground".

11. Execute o comando `jobs -l`, que irá listar os números de identificação (PID) dos processos associados a cada um dos jobs. No outro terminal, olhe os processos com o comando `ps`, identificando aquele que corresponde ao primeiro job. Em seguida, elimine este processo usando o

comando `kill <número-do-processo>`. O número do processo aparece tanto no output do comando `jobs -l` quanto no do comando `ps`. O que acontece com o relógio? Olhe de novo os processos e execute de novo o comando `jobs` no outro terminal.

12. Observe que, ao contrário do comando `ps`, o comando `jobs` dá resultados que são específicos da shell sob a qual está rodando. O comando `jobs` não é um programa como o comando `ps`, que está num arquivo que a shell lê e roda. Ele é o que chamamos de um "comando intrínseco" da shell, ou um "shell built-in", um comando interno do programa que chamamos de "shell". Em cada tipo de shell existe um número relativamente pequeno de comandos deste tipo. Execute o comando `jobs` em ambos os terminais e verifique a diferença nos outputs de cada um.

13. Para reconectar um job ao seu terminal, você executa o comando `fg %<número-do-job>`. Faça isto com o segundo relógio. Observe que o seu terminal ficou preso de novo. Você pode colocar ele no background de novo digitando `[Ctrl]-Z`. Para reconectar o job mais uma vez ao seu terminal, basta dar agora o comando `fg` sem argumento. Havendo apenas um job, todos os comandos de controle de jobs vão se aplicar a ele de forma automática. Elimine o segundo relógio usando `[Ctrl]-C` ou o comando `kill`.

14. Num dos dois terminais que você abriu, rode o comando `ps` com as opções `a`, `u` e `x`, ou seja, use o comando da forma `ps aux`. O comando `ps` tem a peculiaridade de não exigir (na verdade, de não querer) os sinais de `-` antes das opções. Se você os colocar, ele entende o input e executa corretamente, mas dá um aviso de "deprecated", o que significa que um dia ele vai parar de aceitar esta forma de se colocar as opções. Em geral, com ou sem os sinais de `-`, quando um comando é submetido com várias opções, você pode juntar as opções, ou seja, em nosso caso `ps aux` é equivalente a `ps a u x`.

    Esta forma do comando vai colocar na tela uma lista de todos os processos que estão rodando no sistema. A opção `a` significa "all", ou seja, não apenas os seus processos, mas também os de outros usuários. A opção `u` faz com que o comando mostre o usuário ao qual pertence cada processo. A opção `x` inclui na lista os processos que não estejam conectados a um terminal. Caso você queira descobrir mais sobre este comando, olhe a sua página do manual, executando `man ps`.

Provavelmente você vai descobrir que a lista que aparece não cabe toda no seu terminal. Em geral há um número considerável de processos rodando em qualquer sistema, mesmo quando ele não está fazendo nada particularmente notável. A maior parte destes processos é do usuário root, o que indica que se trata de processos internos do sistema ou de serviços mantidos pelo sistema, que não pertencem a nenhum usuário em particular.

15. Para poder olhar toda a lista de processos de uma forma controlada, vamos agora usar o conceito de **pipeline** de comandos, que é um dos conceitos importantes do sistema. Desta forma, redirecionamos o output do comando `ps aux`, que é uma lista um tanto longa, para o input de um outro programa, um paginador como o programa `more`, que nos permita olhar uma página de cada vez. Faça isto, executando a pipeline `ps aux | more`, passando de cada página para a seguinte com a barra de espaço do teclado, ou indo de linha em linha com a tecla `[Enter]`.

16. O comando `more` também pode ser usado diretamente para paginar um arquivo qualquer. Use-o da forma `more /etc/passwd` para verificar este fato. Tente também `ypcat passwd | more`, caso o seu sistema esteja rodando o sub-sistema NIS (Network Information Service, vamos ver isto mais tarde). Alguns comandos e programas já paginam o seu output de forma automática como, por exemplo, o comando `man`. Execute o comando `man more` para verificar este fato.

    O comando `man` não tem um código de paginação interno. Em vez disso, ele está fazendo, internamente, uma pipeline e usando um programa de paginação. Note como, desta forma, economiza-se muita repetição de programação no sistema. Podemos ter dois ou três programas diferentes de paginação, mas não é necessário repetir todo o código de paginação em cada programa que necessitar deste tipo de ação sobre os dados de saída.

17. O comando `ps` nos permite olhar os processos do sistema num determinado momento. Há também uma utilidade, chamada `top`, que nos permite monitorar os processos de forma mais detalhada, pois ele fica rodando, atualizando o seu output a intervalos regulares, mostrando os processos em ordem decrescente de tempo de CPU utilizado por cada um. Ou seja, ele nos mostra os processos mais ativos do sistema por

ordem decrescente de atividade. Desta forma, ele nos dá uma boa idéia de como está a carga de processos no sistema. Execute este aplicativo em um dos dois terminais que você abriu, digitando `top`. Observe durante algum tempo o output que aparece na tela.

18. No outro terminal, vamos rodar um programa que vai consumir uma grande quantidade de tempo de CPU, para ver o que acontece no output do comando `top`. Execute no outro terminal o comando `cat /dev/zero > /dev/null`. Como o arquivo `/dev/zero` não é de fato um arquivo comum e sim uma conexão com o gerador de zeros binários do kernel, este comando vai usar muito tempo de CPU. O resultado de toda esta atividade é uma grande quantidade de zeros, que vão ser descartados, pois o output está re-direcionado para `/dev/null`.

    Não se recomenda rodar algo assim durante a operação normal do sistema, pois trata-se de um exemplo paradigmático de atividade 100 % ineficiente, que vai consumir grande quantidades de recursos do sistema com absolutamente nenhum resultado. Observe o output do comando `top` que está rodando no outro terminal durante algum tempo. Depois disto, interrompa o processo que você está rodando neste terminal, digitando `[Ctrl]-C`. Volte a observar mais um pouco o output na outra janela.

19. Termine o comando `top` digitando a letra `q` no terminal onde ele está rodando. O comando `top` também pode ser rodado num terminal comum, fora do sistema de janelas X11. Para fazer isto, é necessário que você faça um novo login no sistema, em um dos consoles de caracteres. Ao fazer isto, não saia da sua seção X11, você vai passar a ter duas seções independentes no sistema, uma no X11 e outra num console de caracteres.

    Para fazer isto, digite `[Ctrl]-[Alt]-F1`, o que o levará para o primeiro console virtual, que é o terminal `tty1` do sistema. Faça um login neste terminal. Este console é agora um novo terminal de acesso ao sistema, onde está rodando mais uma shell pertencente a você. Rode o comando `top` neste terminal. Verifique que ele funciona exatamente como você já viu antes ao rodá-lo dentro do sistema de janelas X11.

20. Deixe o `top` rodando no primeiro console e volte ao sistema X11, digitando `[Alt]-F7` (ou `[Ctrl]-[Alt]-F7`, tanto faz, quando você está

executando isto fora do sistema X11). Execute o comando `ps` dentro de uma das duas janelas de terminal que ainda estão lá. Você consegue identificar os processos que se referem à sua outra seção? Mate o processo do comando `top` com o comando `kill` e volte ao primeiro console para verificar o que aconteceu. Termine sua seção no console executando o comando `exit`. Observe a bandeira de login do terminal de caracteres voltar a aparecer e depois volte ao X11.

21. Existe outra forma de se executar um processo, que às vezes é útil. Trata-se do uso do comando `exec`, que é outro "built-in" da shell. Para usar este comando, basta executar `exec <programa>`. Quando executamos programas usando este comando, a shell comete um tipo de suicídio: em vez de rodar o programa como um novo processo, ela roda o programa em lugar dela mesma e morre. Para testar isto, execute o comando `exec top` em um dos dois terminais que você tem no sistema X11. Em seguida, digite neste mesmo terminal a letra `q` para terminar o top e veja o que acontece.

22. Isto sugere que qualquer programa que rode num terminal de caracteres pode ser rodado dentro do sistema X11 através do uso de um terminal e de uma operação do tipo que é realizada pelo comando `exec`. De fato, o comando `xterm` tem uma opção que permite exatamente este tipo de uso.

    Normalmente, quando se executa o `xterm`, uma shell é iniciada dentro dele, para que o terminal possa ser usado para a interação com o sistema. Verifique isto executando o comando `xterm &`. Você agora voltou a ter duas janelas de terminal disponíveis. Execute agora o comando `xterm -e top &`, em qualquer das duas janelas de terminal. A opção `-e <programa>` troca a shell pelo programa discriminado. Verifique que aparece mais uma janela de terminal rodando o top dentro. Digite `q` nesta nova janela e veja o que acontece.

23. Você acaba de fabricar um aplicativo X11 a partir de um aplicativo para terminais de caracteres, que poderíamos até chamar de `xtop`! Com isto, podemos associar a um botão do sistema de janelas qualquer comando, mesmo que ele rode apenas em terminais de caracteres.

24. Saia da sua seção X11. Para isto, de-iconize a janela de terminal que apareceu originalmente quando você entrou. Dependendo de como es-

tiver configurado o sistema de gerenciamento de janelas você estiver usando, o ícone que representa aquela janela pode estar dentro de uma caixa, um gerenciador de ícones, ou em qualquer lugar da tela. Nestas notas estaremos assumindo que você está usando o gerenciador `fvwm`, configurado de uma certa forma padronizada.

Caso o gerenciador de ícones esteja sendo usado, basta clicar uma vez o botão esquerdo do mouse em cima do botão que representa a janela. Caso contrário, basta dar um double-click, ou seja, clicar rapidamente duas vezes com o botão esquerdo do mouse, em cima do ícone que representa a janela, o qual pode estar em qualquer lugar da tela.

Uma vez aberta a janela, execute o comando `exit` nela para terminar a sua seção. Sempre que abandonar um terminal, verifique que você fechou todas as seções que tiver iniciado nele, tanto no X11 quando nos consoles de caracteres, caso contrário sua conta estará aberta e sem proteção, à mercê de qualquer um que tenha acesso ao terminal.

3 Problemas e Dicas

1. Ou outro programa apropriado para a manufatura de um aplicativo X11 a partir de um aplicativo para terminal de caracteres é o cliente de mail `pine`. Este é um programa que você provavelmente já está tentando usar, que permite a leitura e o envio de mensagens de email. Tomando como guia o exemplo do texto, crie uma linha de comando para rodar um `pine` dentro de um terminal `xterm`.

2. Uma das coisas que deixa os usuários iniciantes confusos em relação ao uso do sistema de mail em geral e do `pine` em particular é que, em geral, as mensagens de mail são recebidas apenas em uma determinada máquina de um conjunto de máquinas que atuam em conjunto numa rede, máquina esta que é denominada de servidor de mail. Em nosso caso, o servidor de mail é o `socrates.if.usp.br`. O usuário faz um login em alguma outra máquina, executa o `pine` e descobre que não há nenhuma mensagem recebida. Na realidade, há, mas elas estão no servidor de mail.

   Há formas mais estruturadas de resolver este problema, mas por enquanto vamos tentar inventar uma linha de comando que rode o pine sempre na máquina correta, mesmo se ele for executada em alguma outra máquina. Para isto, tente dar uma olhada no comando `rsh`, que serve para executar um comando numa máquina remota. Combine os elementos de tal forma que este comando seja executado dentro de um `xterm`, com a consequência de que ele subsequentemente execute o `pine` no `socrates`. Esta é uma combinação muito útil, que poderíamos no futuro associar a um botão do sistema de janelas, criando assim uma forma conveniente de ler o mail.

3. Você consegue imaginar onde os programas `ps` e `top` vão buscar todas as informações que eles nos mostram? Dê uma boa passeada pelo filesystem virtual que está montado em `/proc` e dê uma boa olhada por lá para ver se dá para compreender a natureza do conteúdo dos arquivos que existem lá. Este filesystem virtual é, na realidade, um dispositivo através do qual podemos ver (e, às vezes, mudar, desde que se tenha autorização de root) as tabelas internas de dados do kernel.

4. Se você sabe um pouco de programação na linguagem C, escreva um pequeno programa chamado `upcase.c` para ler caracteres do canal padrão de entrada `stdin` (use para isto a função `getchar()`) e escrever caracteres no canal padrão de saída `stdout` (usando a função `putchar`), de tal forma que todos os caracteres que correspondem às letras sejam transformados em caracteres maiúsculos, independentemente de entrarem como minúsculos ou maiúsculos. Não se esqueça de incluir a cedilha e todas as letras acentuadas.

   Use este programa numa pipeline junto com os comandos `cat` e `more` para transformar em maiúsculas todo o conteúdo do arquivo fonte desta apostila ou de algum outro arquivo de texto. Escreva outro programa chamado `downcase.c` para fazer o inverso. Se você não sabe nada de C mas conhece alguma outra linguagem de programação, use-a, seja ela compilada ou interpretada.

4 Uma Lista de Comandos Relevantes

`bg %<número-do-job>` : coloca o job de número indicado para rodar no background; é um comando built-in da shell.

`cat <arquivo>` : mostra o conteúdo do arquivo.

`exec <programa>` : executa o programa, que passa a substituir a shell, a qual termina; é um comando built-in da shell.

`exit` : termina a sessão, ou seja, a shell; é um comando built-in da shell.

`fg %<número-do-job>` : conecta o job de número indicado com o terminal; é um comando built-in da shell.

`jobs [-l]` : mostra o conjunto de jobs que estão sob a administração da shell; com a opção `-l`, mostra também os números dos processos (PID) associados aos jobs; os brackets quadrados que se colocamos na descrição deste comando são uma forma comum de indicar que o que está dentro deles é opcional e não necessário para a execução do programa em questão; é um comando built-in da shell.

`kill [-9] <número-do-processo>` : elimina o processo cujo número (PID) é o indicado; sem a opção `-9` isto é feito enviando um sinal de terminação, chamado TERM, para o processo; alguns programas podem não entender este sinal e se recusarem a morrer; com a opção `-9` é enviado um sinal KILL, que com certeza faz o processo morrer; é um comando built-in da shell.

`man <nome-de-comando>` : mostra a página de manual do comando; para sair, basta digitar a letra `q` (de "quit").

`more <arquivo>` : mostra o conteúdo do arquivo, uma página de cada vez; a página tem o tamanho da tela do terminal e passa-se de cada página para a próxima digitando-se espaços, ou seja, usando a barra de espaço do teclado.

`ps [a][u][x]` : mostra processos que estão rodando no sistema; sem opções, mostra apenas os processos do usuário que estão conectados a um terminal; a opção `a` inclui os processos de outros usuários; a opção `u` faz com que o comando mostre o usuário ao qual pertence cada processo; a opção `x` inclui os processos que não estejam conectados a um terminal.

`top` : monitora os processos do sistema, mostrando os mais ativos em ordem decrescente de atividade; trata-se de um aplicativo interativo, para sair dele digita-se a letra `q`.

`xclock` : é um aplicativo X11 que abre uma janela contendo um relógio.

`xterm [-e <programa>]` : é um aplicativo X11 que abre uma janela de terminal; em geral uma shell é executada dentro do terminal; com a opção `-e <programa>` o programa discriminado é executado em lugar da shell.

`ypcat <mapa-do-NIS>` : mostra o conteúdo de uma mapa de dados do sistema NIS (Network Information System).

5 Arquivos de Configuração

A shell que usamos como default, a `tcsh`, é controlado por dois arquivos de configuração, chamados `.cshrc` e `.login`, na raiz da sua conta. Entretanto, relativamente poucas configurações dentro destes arquivos dizem respeito diretamente aos assuntos que examinamos nesta apostila. Olhe dentro deles ou dos arquivos que são chamados de dentro deles com o comando `source` e descubra se há alguma definição de `alias` usando pipelines. Você poderia imaginar alguma utilidade para um alias deste tipo, que combinasse vários comandos para executar operações mais complexas? Tente implementá-lo em sua conta.

Olhe também os arquivos de configuração do sistema de janelas `fvwm` dentro do diretório `.fvwm/` na raiz da sua conta ou, caso ele não exista, na raiz de alguma outra conta que esteja aberta para leitura. Veja se você pode identificar em algum lugar a encodificação de comandos, programas ou pipelines cuja execução esteja associada a botões ou menus. Caso ele ainda não exista, crie um diretório chamado `.fvwm/` na raiz da sua conta e copie para dentro deles as configurações de algum colega ou de qualquer outro usuário do sistema para poder brincar um pouco com elas. Lembre-se, é lícito ler e copiar qualquer coisa que esteja aberta para leitura, este sistema funciona na base de livre compartilhamento de informação.