Breve Introdução ao Linux

No que se segue faz-se uma revisão breve da informação necessária à utilização do sistema Linux e de algumas das ferramentas que o acompanham e fornecem-se, também, referências para outras fontes de informação. Para uma apresentação mais completa ver http://heather.cs.ucdavis.edu/~matloff/unix.html.

Os tópicos abordados nesta secção incluem: Interface, Sistema de Ficheiros, Comandos, Pseudónimo, Permissões, Variáveis de Ambiente, Redireccionamento, Encaminhamento, Substituição de Comandos, Programação, Compilação e Produção.

Interface

A interface com o computador hospedeiro faz-se através de um programa de sistema (imediatamente activo após a validação do utilizador) que actua de forma transparente para o utilizador, de forma a: i) ler comandos do teclado e informação de posição do rato; ii) executar as acções correspondentes à execução dos programas e iii) apresentar no ecrã os resultados pretendidos.

O mais popular daqueles programas de sistema em Unix é o *Bourne shell* (sigla **sh)**. Existem actualmente muitas outras variantes modernizadas e enriquecidas daquele *software* de sistema, sendo uma das mais conhecidas a **bash,** presente no ambiente Linux.

Comandos

O utilizador estabelece a interface com o sistema de exploração através da execução de comandos. Existem dois tipos de comandos em Unix: os internos (pré-construídos) e os externos (executáveis em directórios). Os comandos de interface mais habituais em Linux são os seguintes.

Sigla	Parâmetros	Exemplo	Significado
mkdir	<NomeDirectório>	`mkdir ~/descente/pontoA`	Cria directório
cd	<NomeDirectório>	`cd ../acima/aqui`	Altera directório actual
pwd		`pwd`	Identifica o directório actual
ls	[-a,-F…] directório>	`ls –a /`	Mostra todos os ficheiros na raiz
mv	f1, f2	`mv f1 ../f2`	Ficheiro *f1* é renomeado *f2* e movido para o directório ascendente
cp	f1, f2, .., fn /ali/directório	`mv f1 f2 destino`	Copia os ficheiros *f1,f2,..,fn* para o directório *destino*
chmod	modos permissões	`chmod o+rwx fich1`	As permissões de acesso a *fich1* incluem, para todos, a leitura, a execução e a escrita
less	<NomeFicheiro>	`less d1/d2/f1`	Visualiza o conteúdo de *f1*
exit		`exit`	Abandona a sessão corrente
man	<NomeDeComando>	`man ls`	Ajuda para o comando ls

Pseudónimo

Uma possibilidade interessante para a definição de comandos personalizados é a utilização do comando `alias` que permite criar um pseudónimo para um novo comando a partir de comandos anteriormente definidos.

```
alias ls "ls -F"
alias dir "ls –a"
alias more "less"
alias raiz "dir /"
```

Sistema de Ficheiros

Depois de feita a validação do utilizador dá-se o arranque automático de uma instância da *bash* que é imediatamente "posicionada" num directório inicial, atribuído pelo sistema a cada um dos utilizadores. Este é o directório usado, por omissão, pela *bash*, como referência para a leitura de ficheiros de entrada de dados e a escrita de ficheiros de resultados.

O utilizador pode, a todo o momento, alterar a definição da posição de leitura e de escrita dos ficheiros usados pela *bash* e pelos programas executados, através de comandos específicos. Os comandos usam uma notação própria para a identificação dos elementos presentes no sistema de ficheiros do sistema hospedeiro.

O símbolo "**~**" é usado como um substituto para a identificação da posição do directório inicial de arranque que em Unix é um nodo de um sistema de ficheiros hierárquico, descrito como uma árvore invertida em que o topo da árvore, designado por raiz, é representado pelo símbolo "*/*" isolado.

Qualquer ponto da árvore pode ser alcançado pela definição de um caminho construído tomando como referência um ponto inicial (a raiz (/), o directório corrente (**.**), o director ascendente (**..**), ou o directório inicial (**~**)), sufixado por uma lista de nomes (designativa de sub-directórios descendentes) separados pelo símbolo "*/*".

Permissões

As permissões para acesso aos ficheiros (*quem*?) são determinadas por uma combinação de propriedades definidas através dos caracteres [u,g,o]. Para o criador (u), para o grupo de que faz parte (g) e para todos os outros (o).

Para definir os modos de acesso (*o quê?)* usam-se os caracteres [r,w,x]. Para leitura (r), para escrita (w) e para execução (x); no caso de um directório, possuir a permissão (x) isso quer significar que pode ser atravessado, isto é, ser incluído da definição de um caminho.

São habitualmente usados os comandos `chmod` e `chgrp` para definir as permissões de acesso a ficheiros.

Um utilizador para poder executar um programa contido num ficheiro, tem de possuir permissões compatíveis com os atributos do mesmo, no sistema de ficheiros. Assim, para uma ou mais das classes de utilizadores a quem é garantido o acesso tem de estar, obrigatoriamente, definidos os atributos de **leitura** e de **execução**.

Variáveis de Ambiente

A execução dos programas em Unix faz uso de definições de ambiente construídas a partir dos valores atribuídos a **variáveis** de sistema, criadas automaticamente no momento de arranque. Através de comandos específicos, o utilizador pode criar novas variáveis, ou visualizar e alterar os valores já atribuídos. Exemplos de variáveis pré-definidas e respectivos valores iniciais são:

```
LOGNAME="amp"
HOME="/home/amp",
PATH="/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/amp/bin"
SHELL="/bin/bash"
LANG="pt_PT.UTF-8"
```

Tomando como referência a *bash*, para visualizar o valor de uma determinada variável usa-se um comando apropriado e o nome da variável prefixado com o símbolo **$** como em:

```
echo $PATH
```

Uma nova variável pode ser criada localmente no ambiente actual, ou ser definida globalmente como se pode ver nas seguintes definições:

```
PATH=$PATH:/usr/novoBin
export MINHA=”/usr/outro/Caminho”
PATH=$PATH:$MINHA
```

Redireccionamento

Habitualmente os programas enviam resultados para o ecrã (*stdout*) para poderem serem examinados. Fazendo uso de uma facilidade conhecida como redireccionamento é possível preservar aqueles resultados sob a forma de um ficheiro. Assim, qualquer comando pode usar o símbolo "**>**" seguido do nome de um ficheiro para redireccionar toda os resultados para o ficheiro correspondente, como é o caso em:

```
dir > listaDedirectórios

pwd > directórioCorrente
```

Encaminhamento

Uma forma de redireccionamento mais poderosa é a que permite o encaminhamento dos resultados de saída de dados (*stdout*) produzidos por um determinado programa (comando) para a entrada de (*stdin*) de outro comando e assim sucessivamente, através do uso do símbolo "**|**".

Vejamos como poderíamos saber qual o número de ficheiros de um determinado directório de arranque usando o comando **wc** que permite contar o número de linhas e caracteres existentes num determinado ficheiro.

```
alias contaLinhas "wc –l"
dir  ~/esteDirectório | contaLinhas
```

Substituição

O resultado produzido por um determinado programa pode ser usado como entrada de outro programa usando um mecanismo de substituição disponível através da utilização de um par de plicas "` `" que delimitam o comando que se pretende substituir. Por exemplo para listar todos os ficheiros do directório inicial pode usar-se:

```
dir `pwd`
```

Programação

Linguagens como o C e o C++ oferecem aos programadores níveis muito profundo de programação que visam obter reduzidos tempos de execução para os executáveis programas. Contudo, em muitos domínios de aplicação o tempo de execução não é o principal factor a considerar.

A escrita de programas usando linguagens interpretadas de alto-nível é muitas vezes mais conveniente quando se pretende:

- Prototipificação rápida
- Evitar a declaração de tipos de variáveis
- Usar construtores de alto-nível
- Evitar a compilação

Uma das linguagens de programação interpretadas mais popular nos nossos dias é, provavelmente, o **Perl**. Para além da sua evidente capacidade expressiva e elegância, a semelhança com o C faz com que possa ser usada como uma linguagem de especificação rápida, cujos programas podem facilmente ser traduzidos para C.

Compilação

Para executar um programa escrito em linguagens como o C é necessário que o código fonte desenvolvido seja previamente compilado para produzir um ficheiro contendo o código máquina apropriado para correr em cada sistema de computação concreto.
Em Linux a compilação é, normalmente, levada a cabo através de um comando gcc emitido da consola que obedece ao formato geral que segue.

```
> gcc -Wall –O2 –o progExecutavel funcao-1.c funcao-2 objFuncao-3.o
```

A opção -***Wall*** acciona a geração automática de mensagens de diagnóstico que descrevem a existência de imprecisões ou potenciais fontes de erro exitentes em cada um dos módulos, a opção -***O2*** indica ao compilador que deve usar o nível 2 de optimização do código.

O programa final executável, resulta da ligação de vários módulos que podem estar em estágios diferentes de processamentos. Notar que em linguagem C, um e só um dos módulos pode conter na sua definição uma função com o nome main. A opção ***–o*** refere o nome do ficheiro resultante da execução do comando. Opcionalmente pode ser usada a opção -**S** para indicar que se pretende produzir código em linguagem de montagem (extensão .s)

Produção

A utilização de directivas de compilação é o modo preferido quando a complexidade de um programa constituídos por múltiplos ficheiros de código fonte é gerida através de ferramentas como é o caso do `make`.
Neste caso é preferível reunir aquelas directivas num ficheiro (habitualmente designado por **makefile**) com um conteúdo semelhante ao que segue.

```
CC = gcc
CFLAGS = -Wall -O2 -I .

PROGS =repDados\
      tamTipos\

all: $(PROGS)

repDados:  mostra_octetos.c repDados.c
      $(CC) $(CFLAGS) mostra_octetos.c repDados.c -o repDados

tamTipos: tamTipos.c
      $(CC) $(CFLAGS) tamTipos.c -o tamTipos

clean:
      rm -f $(PROGS) *.o *~core
```

O ficheiro `makefile` contém directivas capazes de, de uma forma fácil e coerente, garantir que todas as alterações aos fragmentos de código que compõem os programas são devidamente consideradas durante a geração dos executáveis:

Se considerarmos a existência, num determinado directório, de um ficheiro ***makefile*** com o conteúdo acima, a evocação do programa ***make***, evocado na linha seguinte, teria como resultado a criação (actualização) dos executáveis deduzíveis na directiva **all** que por seu lado, automaticamente, expande ***$(Progs)*** na lista de nomes especificadas através da variável ***Progs***:

```
>make
```