Instituto Superior de Engenharia do Porto
Departamento de Engenharia Informática

SISTEMAS OPERATIVOS I

Textos de Apoio às Aulas Práticas

Introdução ao Unix

baseado no livro
"UNIX For Application Developers"
William A. Parrete

Abril de 2002

Lino Oliveira

Sugestões e participações de erros para: lino@dei.isep.ipp.pt

INTRODUÇÃO AO UNIX

Índice

1 INTRODUÇÃO AO UNIX

1.1 O QUE É UM SISTEMA OPERATIVO ?

- Conjunto de programas quer permitem a gestão de recursos disponíveis num computador
- Disponibiliza um interface para fácil utilização do hardware

1.2 UNIX

- Trabalha em praticamente qualquer hardware
- Trabalha da mesma maneira em qualquer computador
- Origem e grande popularidade no meio universitário

1.3 COMANDO VERSUS PROGRAMA

- Outros sistemas operativos têm comandos para permitem indicar ao sistema operativo quais as acções a executar. Estes comandos são interpretados e executados com parte do sistema operativo.
- No Unix não existem comandos. Existem utilitários que são pequenas aplicações destinadas a executar determinadas operações.
- Para não confundir aqueles que estão habituados a lidar com outros sistemas operativos, normalmente usa-se, indistintamente, os termos comando, programa ou utilitário.

1.4 ALGUMAS CARACTERÍSTICAS DO UNIX

- MULTI-UTILIZADOR: apesar de existir apenas um computador, vários utilizadores podem estar ligados à máquina e o UNIX faz parecer que cada um deles tem o seu próprio sistemas operativo

- INTERACTIVIDADE: espera a escrita de um comando, executa-o, apresenta o resultado e espera um novo comando

- MULTI-TAREFA: cada utilizador tem a possibilidade de executar diversos utilitários (ou tarefas, como são normalmente designados) em simultâneo

- SEGURANÇA E PRIVACIDADE: cada ficheiro num sistema UNIX possui um conjunto de permissões associadas que impedem acessos indevidos. Cada utilizador tem também uma área de disco reservada que não interfere com mais ninguém

- INDEPENDÊNCIA DOS DISPOSITIVOS ENTRADA/SAÍDA: uma característica importante do UNIX reside no facto de que cada dispositivo periférico ligado aio computador é apenas mais um ficheiro para o sistema operativo UNIX. Deste modo, todas as operações de escrita e leitura são executadas da mesma maneira, independentemente do dispositivo em causa
- COMUNICAÇÃO ENTRE PROCESSOS: as aplicações podem ser desenvolvidas de maneira a ser possível a comunicação entre elas sem necessidade de intervenção do utilizador

- REDE: o UNIX possui todo o software necessário para instalar o computador numa rede. A partir desse momento, o UNIX permite trabalhar com outros utilizadores de outros computadores, partilhando ficheiros e comunicando com eles

- COMANDOS/UTILITÁRIOS: comandos são apenas programas utilitários. Esta flexibilidade permite-nos desenvolver os nosso próprios comandos e integrá-los no próprio sistemas operativo

- SHELL: é uma características do UNIX que outros sistemas operativos têm vindo a "copiar". De facto, a *shell* é apenas mais um utilitário que nos permite lançar os comandos a executar. Como utilitário que é, se não gostarmos da que nos é disponibilizada, podemos desenvolver a nossa própria *shell*. Disponibiliza também uma interface completamente programável.

1.5 COMPONENTES DO UNIX

- KERNEL: parte do sistema operativo que pode ser verdadeiramente chamado de sistema operativo
- FILE SYSTEM (Sistema de Ficheiros): guarda e recolhe ficheiros para os utilizadores e os periféricos
- SHELL: lê, interpreta e executa os comandos que os utilizadores escrevem
- UTILITÁRIOS: pequenas aplicações que acompanham o UNIX e que permitem uma mais fácil utilização do sistema operativo

1.6 OBTENÇÃO DE AJUDA

Para além de material impresso (livros, guias de referência, revistas) ou acessível na Internet, poderemos encontrar ajuda sobre os comandos no próprio sistema Unix.

Todas as versões do Unix disponibilizam um comando "*man*" (abreviatura de manual) que nos permite obter informação detalhada sobre cada comando instalado no sistema. A ajuda obtém-se executando o comando da seguinte forma:

```
man comando
```

Este comando gere um output semelhante ao existente no manuais originais do sistema operativo.

Algumas versões de Unix mais recentes permitem outras formas de obtenção de ajuda através do comando *help* que nos dá uma informação mais abreviada e que pode ser usado de duas maneiras:

```
help comando      ou      comando --help
```

1.7 LOGIN

Todos os utilizadores se identificam perante o UNIX com o USERNAME e PASSWORD, atribuídos pelo administrador do sistema, antes de iniciarem cada sessão de trabalho.
Depois da correcta validação, por parte do sistema operativo, dos dados introduzidos, o utilizador é encaminhado para o ambiente de trabalho – a *shell* - que nos permite trabalhar com os outros componentes dos UNIX – o *kernel*, o *file system*, e os utilitários.

A *shell* é responsável pela interpretação e execução dos comandos.

1.8 SINTAXE GERAL DOS COMANDOS UNIX

```
$ comando [ opção ... ] [ expressão ] [ ficheiro ... ]
```

```
$           - prompt, indicativo de que estamos na shell
[ ]         - indicam que esta parte do comando é opcional
...         - indicam que a parte em causa se pode repetir
```

`opção` - parâmetros que condicionam a execução do comando
`expressão` - dados necessários para a execução do comando
`ficheiro` - se o comando opera com ficheiro(s) , este aparecem sempre no fim do comando

1.9 EXECUÇÃO DE COMANDOS MÚLTIPLOS

Normalmente os comandos são executados sequencialmente:

- prompt da shell
- introdução do comando
- execução do comando
- apresentação do resultado no ecrã
- controlo retornado para a shell
- e novamente prompt da shell

Se quisermos executar uma série de comandos por uma determinada sequência, podemos mandar executá-los de uma vez, escrevendo-os todos, separados por ";" (ponto-e-vírgula):

```
$ comando1 ; comando2 ; ...
```

A shell executa-os todos, um de cada vez, como se eles tivessem sido introduzidos individualmente.

1.10 LINHAS DE COMANDO LONGAS

Alguns comandos UNIX poderão necessitar de mais caracteres do que aqueles que podem ser apresentados no ecrã. A shell possibilita-nos lidar com este problema de duas maneiras:

- Quando chegarmos ao limite do ecrã, podemos continuar a escrever. Se o ecrã estiver correctamente configurado, os caracteres surgirão automaticamente no início da linha seguinte. Se não estiver correctamente configurado, o cursor ficará no limite do ecrã, e os caracteres serão continuamente apresentados na última posição da linha, à medida que os formos escrevendo. De qualquer maneira, a shell interpretará correctamente os caracteres introduzidos, sejam eles apresentados correctamente ou não

- Outra maneira é finalizar a linha com um "\" (backslash) mesmo antes de pressionarmos a tecla ENTER. O backslash dá indicações à shell de que o comando continua na linha seguinte. Exemplo:

```
bash>  ls \<enter>
> -l <enter>
```

A alteração do prompt para ">" é uma indicação da shell de que está à espera da conclusão do comando.

1.11 ALTERAÇÃO DO PROMPT

Prompt é o conjunto de caracteres que a shell nos apresenta para nos indicar que está à espera da introdução de um comando.

Por defeito, no sistema original Unix o símbolo do prompt é o cifrão ($) para a Bourne e Korn shell e o sinal de percentagem (%) para a C shell. No ISEP, é usual o prompt da shell estar definido, por defeito, com o nome da máquina onde estamos a trabalhar, por exemplo:

```
asterix>
```

Se não gostarmos do símbolo de shell que nos é oferecido, poderemos definir um para cada uma das shells que nos são disponibilizadas pelo Unix. A maneira com é feita a definição depende da shell com que estivermos a trabalhar.

- Para Bourne shell, Bash shell e Kourne shell

```
$ PS1=”novo_prompt “
```

- Para C shell

```
% set prompt = “novo_prompt “
```

Note o caracter em branco no fim da string. Permite a separação do prompt do comando que estamos a escrever.

1.12 ALTERAÇÃO DA PASSWORD

Um novo utilizador deve mudar a sua password quando fizer login a primeira vez. Existe um comando que permite fazer essa operação. É *passwd*

```
$ passwd
Changing password for lino
Old password:
New password
Re-enter password:
$
```

Sendo um dos pilares da segurança num sistema Unix, existem algumas regras que devem ser usadas na definição das *passwords*:

- Deve ter 6 no mínimo 6 caracteres. Existe um número máximo de caracteres que são considerados
- Deve ser uma combinação de letras e números
- Não pode ser o *username*, o seu inverso ou o *username* deslocado de um ou mais caracteres
- Uma nova password deverá ser sempre diferente da anterior

1.13 ABANDONAR A SHELL

Quando tivermos concluído o nosso trabalho no sistema Unix, não devemos apenas abandonar o terminal onde estivemos. Isto pode constituir um problema grave uma vez que qualquer pessoa que o passe a utilizar, o fará com a nossa identidade no sistema e terá acesso aos nossos ficheiros e dados. Deveremos por isso fazer o *logout*.

O *logout* informa a shell que pretendemos abandonar o sistema. Como a shell é mais um dos programas Unix que lê os comandos como se o estivesse a fazer de um ficheiro, deveremos indicar o fim desse “ficheiro”. Isso faz-se com a introdução do “Control-D” que é o caracter de fim de ficheiro do sistema Unix.

Existe um comando que executa a mesma função que o Control-D. É o comando *exit*.
Em C shell existe o comando *logout*.

1.14 QUESTÕES

1. Qual a informação necessária para um utilizador poder trabalhar numa maquina UNIX ?
2. Como se consegue saber quando é que a *shell* está pronta a aceitar comandos?
3. Diga qual a sintaxe geral de um comando UNIX.
4. Diga duas maneiras de obter ajuda sobre os comandos UNIX.
5. Para que serve um *backslash* (\) no fim de uma linha de comandos ?
6. Para que é que serve o comando **who** ?
7. Qual é o comando para se mudar a *password* ? Qual a sequência de passos a que esse comando força o utilizador ?
8. Para que se usa a sequência de teclas Ctrl-D, e qual o comando que tem o mesmo efeito na *shell* ?
9. Explique o objectivo das seguintes opções do comando **who**: "**-H**", "**-q**" e "**am i**".
10. Use o comando **who** para ver quem são os utilizadores activos no sistema e saber há quanto tempo eles estão inactivos.
11. Como é que se muda o *prompt* numa maquina UNIX?
12. Mude o seu *prompt* para "==>".
13. Qual é a informação fornecida pelo comando **date**?

2 COMUNICAÇÃO ENTRE UTILIZADORES

2.1 COMUNICAÇÃO INTERACTIVA

O comando *write* permite o envio de mensagens para um utilizador que esteja a trabalhar no sistema, isto é, que esteja *logged in*. A sintaxe é a seguinte

```
write nome_login [número do terminal]
```

Aparece no terminal do utilizador destinatário uma mensagem indicando a proveniência, seguida do texto enviado pelo utilizador remetente.

O *write* envia a mensagem linha a linha, logo que carregamos no <Enter> para passarmos para a linha seguinte. O mensagem termina com Control-D.

Se o utilizador destinatário estiver *logged in* em mais do que um terminal, podemos direccionar a mensagem para um terminal específico:

```
$ write lino tty10
```

Se não indicarmos um terminal, a mensagem é enviada para o primeiro onde foi efectuado o login.

Este comando pode ser um pouco inconveniente uma vez que a mensagem mistura-se com aquilo que o utilizador destinatário estiver a fazer no momento da recepção. Para evitar recepções indesejadas, podemos usar o comando *mesg* para definir se desejamos ou não receber mensagens. A sintaxe é:

```
mesg [y | n]
```

O comando usado sem parâmetro informa do estado da recepção.

Existe um outro comando que permite comunicação entre dois utilizadores – *talk*.
As duas diferenças fundamentais entre este comando e o *write*

- O utilizador vê os caracteres à medida que são escritos e não apenas linha a linha
- Para terminarmos parcialmente, pressionamos no <Enter>. Neste momento surge no nosso terminal o caracter "<" indicando que estamos à espera e no terminal do outro utilizador, o sinal ">" indicando que pode escrever.

2.2 COMUNICAÇÃO NÃO-INTERACTIVA

Existem alguns comandos para comunicarmos com pessoas que não estão a trabalhar no sistema. O programa *mail* permite o envio de mensagens para um utilizador dos sistema que não esteja *logged in* num dado momento. A sintaxe é:

```
mail nome_login
```

Vejamos um exemplo:

```
$ mail lino
Subject: Isto é um teste
Aqui começa a mensagem que pretendo enviar.
Para finalizar termino colocando um ponto final na primeira posição
De uma linha
.
Cc:
```

O mesmo programa quando usado sem destinatário, serve para fazermos a gestão das mensagens recebidas. As mensagens são apresentadas uma de cada vez segundo a ordem LIFO (da mais recente para a mais antiga).

O prompt do programa *mail* difere do da shell. Pode ser "?" ou "&" e espera comandos para serem executados sobre as mensagens da nossa caixa de correio.

2.3 QUESTÕES

1. Qual o comando que permite uma comunicação interactiva entre dois utilizadores ?
2. Qual a sintaxe deste comando ?
3. Como se comunica com alguém que está a escrever no nosso terminal?
4. Quando é que o comando **`write`** envia para o terminal do destinatário aquilo que estamos a escrever no nosso terminal ?
5. Como é que se termina o comando **`write`** ?
6. Explique as diferenças entre o comando **`write`** e o comando **`talk`** ?
7. Qual a finalidade do comando **`mesg`** ?
8. Como é que o comando **`mesg`** é utilizado ?
9. Qual é o comando utilizado para enviar mensagens de um modo não interactivo, para outros utilizadores.
10. Qual é a sintaxe deste comando ?
11. Como se pode ler o **`mail`** ?
12. Enquanto se está a ler o **`mail`**, qual o significado dos seguintes comandos: "**`+`**", "**`d`**", "**`s`**" **e** "**`q`**".
13. Use o comando **`who`** para ver quem são os utilizadores activos no sistema e escolha um para comunicar.

3 UTILITÁRIOS DIVERSOS

Cada sistema Unix possui um número enorme (mais de 300) de comandos ou programas utilitários. Para além dos comandos essenciais presentes em todos os sistemas Unix, diversos outros são adicionados por cada fabricante/distribuidor por forma a torná-lo mais interessante e útil, sobretudo graças à facilidade devida ao facto od Unix ser um sistema aberto..

É por isso possível que um determinado comando não existe num determinado sistema, ou que possua diferentes opções. Devemos ter o cuidado de consultarmos o manual sempre que lidarmos com sistemas diferentes.

3.1 LISTAR FICHEIROS

```
ls [-aCqrstux] [nome ...]
```

3.2 APRESENTAR CALENDÁRIO

```
cal [[mês] ano]
```

3.3 APRESENTAR O DIA E A HORA

```
date [string_formato]
```

A string_formato inicia-se com o sinal "+" e cada componente é precedido de "%"

Exemplos: %m – mês
%h – abreviatura do mês
%H – hora
%M – minutos

3.4 CALCULADORA BÁSICA

```
bc [-l] [-c] [ficheiro ...]
```

Quando usado com ficheiros, estes fornecem as operações a calcular.

Operações básicas:

+ Adição	% Módulo (resto de divisão inteira)
- Subtracção	^ Exponenciação
* Multiplicação	
/ Divisão	

scale - define número de casas decimais
ibase - base numérica de entrada
obase - base numérica de saída

bc pode tornar-se uma calculadora programável com uma sintaxe semelhante à da linguagem C.
É uma interface de uma calculadora potente denominada *dc*.

3.5 QUESTÕES

1. Explique porque é que um comando UNIX, pode não estar disponível no sistema que utiliza.
2. Qual é o comando que mostra no terminal o calendário de um ano inteiro ? Como se usa este comando ?
3. Descreva para que serve a opção "**+**" no comando **`date`**.
4. Descreva 4 itens utilizados na formatação de saída do comando **`date`**.
5. Qual é o comando utilizado para fazer cálculos aritméticos ? Quais são as operações básicas que este comando permite efectuar ?
6. Como se pode alterar o número de dígitos à direita do ponto decimal ?
7. Como se pode alterar a base numérica de entrada desse programa ? E a base numérica de saída ?
8. Liste o nome dos seus ficheiros em varias colunas, ordenados pela data da ultima modificação.
9. Crie o calendário do mês de setembro de 1752.
10. Mostre no écran o calendário do mês corrente.
11. Calcule a média dos seguintes números: 123, 410, 211, 99, 314, 793 e 5.
12. Calcule as diferenças entre os seguintes números em octal: 6372 e 5435.