Instituto Superior de Engenharia do Porto
Departamento de Engenharia Informática

SISTEMAS OPERATIVOS I

Textos de Apoio às Aulas Práticas

Permissões

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

Fevereiro 2003

Lino Oliveira

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

PERMISSÕES

Índice

1 PERMISSÕES

Existe um mecanismo de protecção que é parte integrante do sistema operativo Unix e que permite especificar com exactidão quem é que autorizamos a trabalhar com os nosso ficheiros. Essas permissões, é este o nome genérico usado, permitem controlar quem tem acesso aos ficheiros e directórios que criamos.

1.1 UTILIZADORES E GRUPOS

Cada utilizador que tem acesso ao sistema Unix tem um *login id*, atribuído pelo administrador do sistema, que o identifica sem ambiguidade perante o sistema. Adicionalmente, cada utilizador pertence ao grupo de utilizadores relacionados que tem um determinado *group id*.

Um grupo é um conjunto de utilizadores que têm uma característica comum, por exemplo, pertencerem ao mesmo departamento. Um utilizador pode pertencer a mais do que um grupo.

Um dos motivos principais pelo qual os utilizadores estão relacionados através de um grupo é conceder-lhes a possibilidade de partilhar ficheiros. Deste modo, vários utilizadores poderão aceder e alterar aos mesmo ficheiros partilhando informação e tarefas de interesse comum.

Existe um ficheiro no directório /etc onde estão registados os diversos utilizadores autorizados a aceder ao sistema bem como os grupos a que eles pertencem. Esse ficheiro é o ***/etc/passwd*** e é basicamente uma base de dados contendo um registo (linha) por cada utilizador. Cada linha é composta por registos separados por ":" que fornece ao Unix informação importante de cada utilizador.

Cada registo de */etc/passwd* é composto da seguinte maneira:

- **Login id** – login que o utilizador usa para aceder ao sistema
- **Password encriptada** – nenhuma password é guardada no sistema sem ser encriptada
- **User id** – identificação numérica do utilizador; é normal que os computadores trabalhem melhor com números
- **Group id** – identificação numérica do grupo principal a que o utilizador pertence; informação acerca deste grupo, nomeadamente o seu nome, pode ser encontrada no ficheiro */etc/group*
- **Comentário** – normalmente este campo é usado para o nome completo do utilizador
- **Caminho completo para o *home directory* do utilizador** - este campo é usado pelo sistema durante o processo de *login* para determinar qual o local do sistema de ficheiros onde seremos "colocados"
- **Caminho completo do programa inicial** – isto identifica qual o programa a ser executado logo após o processo de *login*; normalmente é identificada aqui a *shell* usada pelo utilizador

Outro ficheiro importante é o ***/etc/group***. Enquanto que no ficheiro */etc/passwd* é identificado o grupo inicial a que cada utilizador pertence, o */etc/group* especifica quais os grupos disponíveis e a que cada utilizador pode pertencer.

Este ficheiro é também uma base de dados semelhante a /etc/passwd e é composto da seguinte maneira:

- **Group name** – da mesma maneira que cada utilizador tem um nome, cada grupo tem também um nome para mais fácil identificação
- **Password encriptada** – o Unix permite passwords para grupos da mesma maneira que para utilizadores, de tal maneira que se queremos pertencer ao um grupo teremos que conhecer a respectiva password
- **Group id** – identificação numérica do grupo e que é usada no ficheiro */etc/passwd*; para sabermos a que grupo o utilizador pertence, procuramos neste ficheiro a linha correspondente ao *group id* definido no registo do utilizador em */etc/passwd*
- **Lista de login id's** – esta é a lista de utilizadores autorizados a pertencer a um determinado grupo

1.2 QUEM É VOCÊ E A QUE GRUPO PERTENCE

Podemos saber qual é o nosso *login*, *user id*, *group name* e *group id* sem ter necessidade de pesquisar o ficheiro */etc/passwd*. O comando *id* apresenta esta informação.

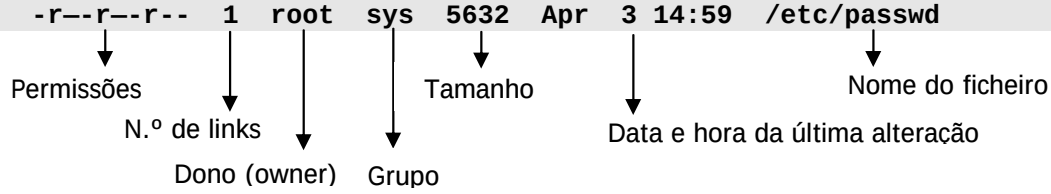
A sintaxe do comando *id* é a seguinte:

```
id
```

1.3 LISTAR NOMES DE FICHEIROS

Existem várias outras opções que podem ser usadas com o comando *ls*. A opção "-l" dá-nos uma lista longa de ficheiros e directórios. A seguir apresenta-se um exemplo do resultado possível do comand *ls –l* e a respectiva explicação:

```
-r—-r—-r--  1  root  sys  5632  Apr  3 14:59  /etc/passwd
```

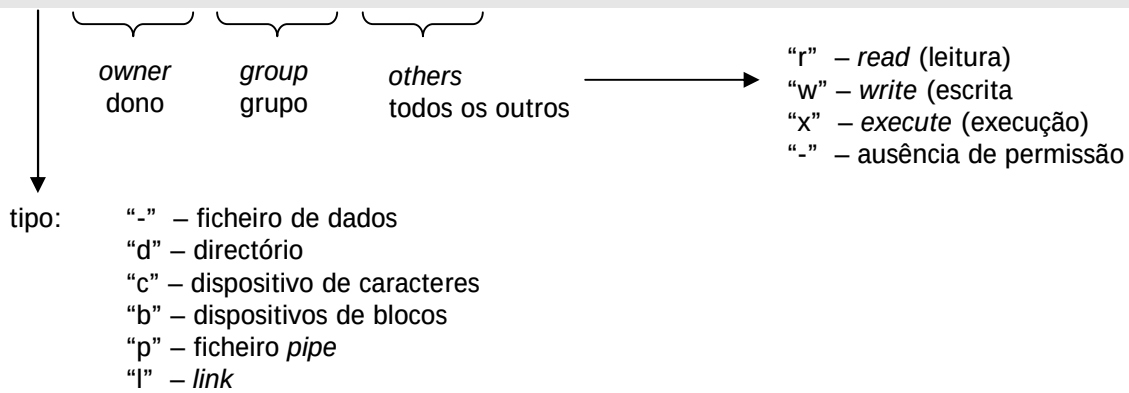


1.4 PERMISSÕES DE ACESSO

Sempre que um ficheiro ou directório é criado, o sistema operativo Unix atribui-lheum conjunto de permissões de acesso. Estas permissões podem ser vistas usando o comando *ls –l*.

As permissões são conjuntos de 10 caracteres divididos em 4 conjuntos que se encontram representados no exemplo anterior pelos caracteres:

```
-  r  w  x  r  w  -  r  -  -
```



Podemos constatar que é possível atribuir diferentes permissões a diferentes tipos de utilizadores:

- *Owner* – o dono do ficheiro é normalmente a pessoa que o criou
- *Group* – o grupo do ficheiro é normalmente o grupo a que o utilizador pertencia no momento em que criou o ficheiro, e refere-se aos utilizadores que pertencem ao mesmo grupo que o dono
- *Others* – os outros são simplesmente todas as pessoas que não são o dono nem pertencem ao mesmo grupo que ele

Cada uma das 3 posições dos caracteres na secção das permissões tem um significado especial que corresponde a 3 diferentes tipos de permissões que cada ficheiro ou directório possui e tem um determinado significado:

- *Read* – a existência da permissão de leitura indica que é possível ler o conteúdo do ficheiro
- *Write* – a existência da permissão de escrita indica que é possível escrever ou alterar o conteúdo do ficheiro
- *Execute* – a existência da permissão de execução indica que é possível, pelo menos, fazer uma tentativa para executar ("correr") o ficheiro como um comando Unix ou um utilitário

No exemplo apresentado acima (`-rwxrw-r--`), as permissões definidas são:

- *Owner* (`rwx`) – leitura (r), escrita (w) e execução (x)
- *Group* (`rw-`) – leitura (r) e escrita (w)
- *Others* (`r--`) – leitura (r)

Esta permissões são óbvias quando se referem a ficheiros: é necessário ter permissão de *read* ("r") para fazer o *cat* do conteúdo do ficheiro, é necessário ter permissão de *write* ("w") para editar e é necessário ter permissão de *execute* ("x") nos ficheiros criados pelas compilações ou contendo comandos Unix para que seja possível "corrê-los".

Mas já não são tão óbvias quando se referem a directórios: *read* ("r") permite ver o conteúdo do directório, isto é, os ficheiros nele contidos, *write* ("w") permite copiar ficheiros para o directório ou apagá-los do directório, *execute* ("x") também designada permissão de pesquisa permite "ir para" o directório com o comando *cd* ou usar o directório em qualquer tipo de *pathname*.

1.5 ALTERAR PERMISSÕES DE ACESSO

Se as permissões atribuídas por defeito pelo sistema operativo não forem do nosso agrado, podemos alterá-las como o comando *chmod* (*change mode*). Este comando permite alterar as permissões de cada ficheiro ou directório que nos pertença.

A sintaxe do comando é a seguinte:

```
chmod expressão_permissões ficheiro ...
```

A "expressão_permissões" permite definir as novas permissões que queremos atribuir ao(s) ficheiro(s) indicados no comando.

Esta expressão pode ser especificada através um número octal de 3 dígitos cujo significado é indicado de seguida:

Permissões em octal		
0	nenhuma	`-`
1	execute	`x`
2	write	`w`
3	write e execute	`wx`
4	read	`r`
5	read e execute	`rx`
6	read e write	`rw`
7	read, write e execute	`rwx`

Exemplo de utilização do comando *chmod*:

```
$ chmod 645 fich_temp
$ ls –l fich_temp
-rw-r—-r-x   1  lino     profs    1123  Fev 17 19:10  fich_temp
```

```
$ chmod 440 fich_temp
$ ls –l fich_temp
-r--r—----   1  lino     profs    1123  Fev 17 19:10  fich_temp
```

Existe outro tipo de expressão que pode ser usado com o comando *chmod*. Esse tipo de expressão usa mnemónicas para especificar permissões.

Uma série de códigos mnemónicos podem ser usados em vez de números em octal e são apresentados nos quadros seguintes:

Classes de utilizadores	
u	*User* (dono)
g	*Group* (grupo)
o	*Others* (outros)
a	*All* (todos)

Operações de permissões	
+	adiciona
−	retira
=	estabelece

Valores das permissões	
r	*read*
w	*write*
x	*execute*

A utilização destes códigos torna mais fácil e intuitiva a atribuição das permissões, como se pode verificar pelo exemplo seguinte:

```
$ chmod u=rw ficheiro1
$ chmod go-rwx ficheiro2
```

Nestes exemplos são executadas as seguintes operações:

- `u=rw` – estabelece as permissões de leitura (*read* "r") e escrita (*write* "w") para o dono do ficheiro *ficheiro1*, independentemente das permissões que esse ficheiro tenha antes
- `go-rwx` – retira as perrmissões de leitura (*read* "r"), escrita (*write* "w") e execuação (*execute* "x") para os utilizadores do grupo (*group* "g") e para todos os outros (*others* "o") ao ficheiro *ficheiro2*.

Este método alternativo de estabelecimento de permissões têm ainda outra vantagem. Permite-nos conjugar diferentes expressões num único comando, separando-as com vírgulas, como a seguir se exemplifica:

```
$ chmod g+w, o-r ficheiro1
$ chmod u+x, g-w, o=r ficheiro2
```

1.6 TIPOS ESPECIAIS DE PERMISSÕES

Se percorrermos o sistema de ficheiros do Unix e analisarmos as permissões dos diferentes ficheiros, encontraremos, por vezes, permissões de execução (3ª posição de cada grupo de 3 letras) que não são "x".

Existem 3 outras permissões que podem ser aplicadas aos ficheiros. Apesar destas permissões serem relacionadas com permissões de execução, elas são, na verdade, bastante diferentes de todas as outras apresentadas até agora.

A seguir apresentam-se estes novos tipos de permissões:

set user-id mode ⟶ ---s--x--x

Quando executamos um programa com este modo activado, e enquanto durar a sua execução, tomamos a identidade do dono (*user*) do programa.

set group-id mode ⟶ ---x--s--x

Quando executamos um programa com este modo activado, e enquanto durar a sua execução, passamos a parte do grupo (*group*) do dono do programa.

sticky *mode* ⟶ ---x--x--t

Este último tipo de permissão é completamente diferente. Para o entender é preciso ter em atenção o que acontece quando se executa um comando Unix.

De cada vez que se escreve um comando no *prompt* da *shell*, esta interpreta-o e procura-o no sistema de ficheiros. Depois de o encontrar carrega-o para memória e executa-o. Quando o comando termina a sua execuação, o Unix liberta a memória utilizada pelo comando e devolve o controlo para a *shell*.

O que é que acontece se se repetir o mesmo comando? O processo atrás descrito repete-se.

Para que isto não aconteça, utiliza-se o *sticky mode*. Um comando que tenha este modo activado, permanece em memória mesmo depois de ter terminado a sua execução, permanecendo no *swap space*, uma espécie de memória virtual. Mantendo--o em memória, reduz-se o tempo necessário para o encontrar e executar. Permanecerá em memória até o sistema ser desligado.

Só o administrador do sistema pode activar o *sticky mode* num ficheiro.

Finalmente, se virmos um S ou um T maiúsculos, em vez de minúsculos, isso significa que a correspondente permissão de execução não está presente.

Para estabelecermos estes tipos de permissões usamos um 4º caracter ao número em octal de tal modo que o 1º caracter estabelece o tipo especial de permissão (*set user-id mode, set group-id mode, sticky mode*) e os 2º, 3º e 4º caracteres estabelecem as permissões normais.

O tipo especial de permissão é atribuído segundo o quadro apresentado a seguir:

Permissões em octal		
0	Nenhuma	---
1	Sticky	--t
2	Set group-id	-s-
3	Set group-id + sticky	-st
4	Set user-id	s--
5	Set user-id + sticky	s-t
6	Set user-id + set group-id	ss-
7	Todos os 3 modos	sst

Exemplo: `chmod 5644 fich1`

Exemplo: `chmod u=srw, g=r, o=rt fich2`

Mnemónicas:

u	+ - =	s =	Set user-id
g	+ - =	s	Set group-id
o	+ - =	s	Sticky

1.7 DEFINIÇÃO DAS PERMISSÕES POR DEFEITO

A cada ficheiro e directório que criamos é atribuído pelo sistema um conjunto de permissões definidas por defeito.

No entanto, podemos controlar essas permissões através dum parâmetro chamado valor de *umask*. Esse valor (valor da máscara de criação de ficheiros) de cada utilizador pode ser visualizador com o comando *umask* que tem a seguinte sintaxe:

```
umask [expressão_umask]
```

Quando executado sem parâmetros, permite-nos saber o valor de *umask* actual. A "expressão_umask" permite-nos alterar esse valor e, consequentemente, as permissões que são atribuídas por defeito na criação dos ficheiros.

O valor de *umask* é um valor em octal composto por 4 dígitos. Quando composto por menos dígitos, podemos considerar zeros à esquerda. Esses dígitos estão relacionados com as permissões do dono (2º dígito), grupo (3º dígito) e outros (4º dígito). O 1º dígito é sempre 0 e é indicativo de que o número apresentado é octal, tal como usado na linguagem C.

Eis o que poderemos obter executando o comando *umask*:

```
$ umask
0022
```

Os valores em octal do comando *umask* têm um significado ligeiramente diferente. Cada dígito de *umask* é subtraído do correspondente dígito usado pelo sistema na criação do ficheiro. Percebemos melhor se interpretarmos o valor com as permissões que queremos retirar aos ficheiros e directórios que criamos.

No exemplo acima, no valor de *umask* 0022, o primeiro 0 indica que o valor é em octal, o segundo 0 indica que não queremos retirar qualquer permissão ao dono e os dois dígitos 2 indicam que queremos retirar permissões de escrita ao grupo e aos outros utilizadores. Podemos verificar o significado destes valores na tabela de valor em octal definida anteriormente, na explicação do comando *chmod*. A diferença reside no facto de que, com o comando *chmod,* estes valores correspondem aqueles que queremos atribuir, enquanto que com o comando *umask* estes valores dizem respeito às permissões que queremos retirar.

Sem considerar qualquer valor de *umask*, qualquer comando em Unix que cria um ficheiro, tenta criá-lo com permissões 666 (`rw-rw-rw-`); todos os directórios e ficheiros executáveis tentarão ser criados com permissões 777 (`rwxrwxrwx`).

Quando definimos um valor de *umask* diferente de 0000, a estas permissões (666 e 777) são retiradas as permissões definidas no valor de *umask*. Com o valor de *umask* de 0022 apresentado anteriormente, os ficheiros serão criados com permissões 644 (`rw-r--r--`) e os directórios e ficheiros executáveis com permissões 755 (`rwxr-xr-x`).

Para alterar o valor de *umask* , basta executar o comando *umask* usando como parâmetro o novo valor que pretendemos, composto por 3 dígitos. Por exemplo:

```
$ umask 027
```

1.8 ALTERAÇÃO DO DONO DE UM FICHEIRO

Outro comando que nos permite alterar um dos atributos de um ficheiro é o *chown* (*change owner*). Permite-nos conceder a posse de um ficheiro a um outro utilizador.

A sintaxe do comando é a seguinte:

```
chmod login_id ficheiro ...
```

Só o dono de um ficheiro pode alterar a pertença de um ficheiro e o seu "login_id" tem de ser uma entrada válida no ficheiro */etc/passwd*. O comando *chown* não altera as permissões, apenas altera o utilizador a quem passa a pertencer o ficheiro.

A partir do momento em que alteramos o dono de um ficheiro que nos pertencia, deixamos de ter as permissões do dono do ficheiro. O novo dono é o único que pode, a partir de agora, executar o *chmod* e o *chown*.

A seguir, exemplifica-se a utilização do comando *chown*:

```
$ ls –l permissoes.pdf
-rwxr--r--  1  lino      profs   146297   Feb 28  11:39  permissoes.pdf
$ chown oliveira permissoes.pdf
-rwxr--r--  1  oliveira  profs   146297   Feb 28  11:39  permissoes.pdf
```

1.9 ALTERAÇÃO DO GRUPO DE UM FICHEIRO

Outro comando que nos permite alterar um dos atributos de um ficheiro é o *chgrp* (*change group*). Permite-nos dar as permissões de grupo num ficheiro a outro grupo. A sintaxe do comando é a seguinte:

```
chgrp nome_grupo ficheiro ...
```

Só o dono do ficheiro pode alterar o seu grupo e o "nome_grupo" tem de ser uma entrada válida no ficheiro */etc/group*. O comando *chgrp* não altera as permissões, apenas altera o grupo que passa a ter as permissões de grupo do ficheiro.

```
$ ls –l permissoes.pdf
-rwxr--r--  1  oliveira  profs   146297   Feb 28  11:39  permissoes.pdf
$ chgrp alunos permissoes.pdf
-rwxr--r--  1  oliveira  alunos  146297   Feb 28  11:39  permissoes.pdf
```

1.10 ALTERAÇÃO TEMPORÁRIA DA IDENTIDADE DE UM UTILIZADOR

Se precisar de assumir as permissões e capacidades de um outro utilizador, poderá fazê-lo através do comando *su*. O comando *su* permite-nos alternar entre diferentes identidades (*user id's*). A sintaxe do comando é a seguinte:

```
su [-] [login_id]
```

Sem qualquer parâmetro, o comando *su* permite-nos assumir a identidade de *root* ou administrador do sistema. Podemos assumir a identidade que qualquer outro utilizador executando o comando *su* com o *login id* desse utilizador como parâmetro. Esse *login id* é pesquisado no ficheiro */etc/passwd* e, caso seja um nome válido, seremos questionados pela password desse utilizador.

Se a password introduzida for a correcta, assumiremos nesse momento a identidade do utilizador correspondente ao *login id* indicado., e iniciaremos um novo nível de *shell*. A qualquer momento poderemos reassumir a anterior identidade, pressionando <control-d>. A seguir exemplifica-se a utilização do comando *su:*

```
$ id
uid=4002(lino) gid=500(profs)
$ su oliveira
Password:
$ id oliveira
```

```
uid=4004(oliveira) gid=500(profs)
$ <control-d>
$ id
uid=4002(lino) gid=500(profs)
```

Se colocarmos um hífen (“-“) entre o comando e o login_id, estamos a indicar ao comando *su* que queremos assumir a identidade correspondente ao login_id exactamente como se tivéssemos feito o *login* normal como esse utilizador. Isto é feito executando todos os ficheiros de arranque que são executados quando o utilizador indicado faz o *login*.

O comando *su* inicia sempre um novo nível de *shell*.

2 QUESTÕES

1. Diga o tipo de informação que se encontra nos campos do ficheiro /etc/passwd.
2. Diga o tipo de informação que se encontra nos campos do ficheiro /etc/group.
3. Qual é o comando do UNIX que permite saber o userid e o groupid ?
4. Diga quais são os três tipos de utilizadores a que as permissões são aplicadas.
5. Diga quais são os três tipos de permissão que estão disponíveis.
6. Quais são as permissões necessárias para se poder editar um ficheiro? Para se poder usar esse ficheiro como um comando ?
7. Quais são as permissões necessárias para se poder listar o conteúdo de um directório? Para se poder remover um ficheiro desse directório ?
8. Descreva a sintaxe e os objectivos do comando chmod.
9. Explique o significado das seguintes expressões de permissão:
   "640", "751", "go+r" e "u+x,o-x".
10. Descreva o uso do comando umask
11. Quais são os comandos para mudar o dono e o grupo de um ficheiro ?
12. Qual é o comando para mudar para um novo userid ?
13. Copie os ficheiros /etc/passwd e /etc/group para o seu directório home.
14. Crie um novo directório chamado WORK no seu directório home.
15. Mude as permissões da sua cópia do ficheiro passwd, para que toda a gente tenha apenas permissão de escrita no ficheiro. Verifique as permissões e tente mudar o ficheiro com o JOE.
16. Mude as permissões da sua cópia do ficheiro passwd, para que você e os membros do seu grupo possam ler o ficheiro, mas os outros utilizadores não lhe possam fazer nada. Verifique as permissões e tente mudar o ficheiro com o JOE.
17. Mude as permissões da sua cópia do ficheiro group, para que ninguém possa fazer nada com o ficheiro. Verifique as permissões e tente mudar o ficheiro com o JOE.
18. Mova as cópias dos ficheiros passwd e group para o directório WORK.

19. Mude as permissões do directório WORK para que o dono tenha só permissão de leitura e os restantes tipos de utilizadores não tenham nenhuma permissão. Verifique as permissões. Tente agora remover o ficheiro group. Tente mudar para o directório WORK.

20. Mude as permissões do directório WORK para que o dono só tenha permissão de escrita no directório. Verifique as permissões. Tente listar o conteúdo do directório WORK. Tente mudar para o directório WORK.

21. Mude as permissões do directório WORK para que o dono só tenha permissão de execução do directório. Verifique as permissões. Tente mudar para o directório WORK, e depois listar o seu conteúdo.

22. Remova o directório WORK.

23. Mude a máscara de criação de ficheiros de modo a que os ficheiros criados tenham permissão de escrita e leitura para si, permissão de leitura para o seu grupo e nenhuma permissão para outros utilizadores. Verifique que quando um directório é criado você fica com permissão de leitura, escrita e execução, o seu grupo tem permissão de leitura e execução, e os outros apenas tem permissão de execução. Crie um novo ficheiro e um novo directório para testar a máscara.