GUIA DE LABORATÓRIO 2.5
GREP E EXPRESSÕES REGULARES (ßeta)

OBJECTIVOS

- Expansão de metacaracteres
- Pesquisas de texto e expressões regulares
- Introdução aos comandos de processamento de texto e à redirecção

INSTRUÇÕES

GREP e Expressões Regulares

1. Localize e analise os ficheiros `heroes.txt` e `texto.txt` que acompanham este guia de laboratório.

> *A maioria dos sistemas disponibiliza algumas facilidades para processar texto. Não só os ficheiros de configuração desses sistemas são ficheiros de texto, como também a maioria das aplicações e os próprios utilizadores necessitam de manipular ficheiros de texto. Um corrector ortográfico, um detector de spam, um motor de pesquisa, etc, são exemplos de aplicações que necessitam de filtrar texto para isolar "pedaços" de informação realmente importantes.*
> *Uma vez que os vários SOs, como o Linux, e as aplicações dependem tanto da intepretação e manipulação de ficheiros de texto, foi criada uma notação concisa para definir filtros (ou padrões) de texto. Essa notação é designada de **expressão regular** (regular expression) e abreviada para **regex**. Um dos vários utilitários do Unix que permitem utilizar expressões regulares é o comando **grep** que filtra linhas de texto que "obedeçam" a uma determinada regex.*

2. Introduza os seguinte comando:

```
$ grep -i man heroes.txt
```

Deverá obter os seguintes resultados:

```
Catwoman
Batman
Spider-Man
Wonder Woman
Ant-Man
Spider-Woman
```

> *A forma mais simples de utilizar o **grep** passa por localizar uma palavra - primeiro argumento do comando - num ficheiro - segundo argumento. Ou seja:*
>
> `$ grep manel pessoas.txt`
>
> *Neste caso, o **grep** exibe todas as linhas que possuam a palavra "manel". O **grep** suporta várias opções. No exemplo aqui ao lado, o **grep** localiza todas as linhas de texto do ficheiro `heroes.txt` com a palavra `man` em minúsculas ou maiúsculas. Indicamos que a capitalização da letra não é importante através do parâmetro `-i`.*

> *A síntaxe geral do **grep** é a seguinte:*
>
> `$ grep [OPÇÃO]... PADRÃO [FICHEIRO...]`
>
> *Se não for indicado nenhum ficheiro, o **grep** procura por PADRÃO na entrada-padrão (STANDARD OUTPUT).*

grep: OPÇÕES COMUNS	
-i	Ignora capitalização. Não distingue entre minúsculas e maiúsculas.
-v	Inverte os resultados da pesquisa, isto é, devolve as linhas que não verificam o padrão.
-c	Diz quantas linhas verificam o padrão, ao invés de exibir essas mesmas linhas.
-l	Indica o nome dos ficheiros que possuem o padrão.
-L	Indica o nome dos ficheiros que **não** possuem o padrão.
-n	Para além de exibir a linha, indica também qual o número da dita.
-o	Exibe apenas o padrão seleccionado (ou seja, não mostra o resto da linha seleccionada)
-h	Suprime o nome do ficheiro em pesquisas multi-ficheiro.
-E	Permite especificar o padrão de texto com uma **expressão regular extendida** (*Extended Regular Expressions* ou ERE). Por omissão, o **grep** utiliza BRE (*Basic Regular Expressions*). Neste laboratório vamos privilegiar ERE.

3. Introduza agora os seguintes comandos e registe o que observa:

```
$ grep bat heroes.txt
$ grep Bat heroes.txt
$ grep -i bat heroes.txt
$ grep -iv bat heroes.txt
$ grep -in bat heroes.txt
$ grep -ivn bat heroes.txt
$ grep -ic bat heroes.txt
$ grep -li woman heroes.txt
```

```
$ grep interface /etc/*.conf
$ grep -l interface /etc/*.conf
$ grep -n interface /etc/*.conf
$ grep -vn interface /etc/*.conf
```

*Uma **expressão regular** é uma notação utilizada para identificar padrões de texto. É semelhante à notação dos* wildcards *da shell, mas mais completa e abrangente. Várias ferramentas e linguagens de programação suportam expressões regulares, ainda que nem sempre a notação seja exactamente a mesma. Neste caso vamos utilizar a notação definida na norma POSIX, notação essa que é comum à da grande maioria de ferramentas que aceitam expressões regulares.*

Os exemplos anteriores já utilizam expressões regulares, muito simples é certo. A expressão regular `bat`, utilizada no exemplo `grep bat heroes.txt`, indica que serão seleccionadas as linhas de `heroes.txt` com os três caracteres `'b'`, `'a'` e `'t'`, por esta ordem e sem outros caracteres entre eles (ou seja, os três caracteres devem aparecer consecutivamente).

*Estes três caracteres são o que se designa por **literais** , no sentido que são utilizados literalmente para seleccionar linhas de texto. Além de literais, temos os **metacaracteres** que são caracteres com um outro significado que não o seu valor literal. Estes permitem introduzir padrões de pesquisa mais complexos. É o caso do * (asterisco) que indica zero ou mais caracteres. Ou do . (ponto) que indica um qualquer caractere.*

grep: METACARACTERES/OPERADORES MAIS COMUNS PARA EXPRESSÕES REGULARES (ERE)	
.	Representa qualquer caractere.
^	Representa o início da linha.
$	Representa o fim da linha.
A *(ou outra letra)*	Representa o A maiúsculo.

grep: METACARACTERES/OPERADORES MAIS COMUNS PARA EXPRESSÕES REGULARES (ERE)	
a *(ou outra letra)*	Representa o a minúsculo.
[…]	Indica um caractere dentro de uma classe de caracteres especificada entre [e] (parênteses rectos). Os metacaracteres perdem o seu significado entre [e]. O caractere ^ entre parênteses rectos significa negação e o - (traço) permite indicar uma gama de caracteres. Exemplos: [a-z] ⟶ Letras minúsculas [^a-z] ⟶ Qualquer caractere excepto minúsculas
A|B	A ou B. Exemplos: x|y ⟶ Caractere x ou y [aeiou]|m ⟶ Vogal ou letra m (abc|def)+ ⟶ Uma ou mais ocorrências do texto abc ou def. abc, def, abcdef, defabcabcdef, etc., seriam sequências aceites (ver em baixo o significado do operador/metacaractere + e dos parênteses).
*X**	Zero ou mais ocorrências de X, ou seja, zero ou mais ocorrências do caractere que precede o *.
X+	Uma ou mais ocorrências de X, ou seja, uma ou mais ocorrências do caractere que precede o +.
X?	Zero ou uma ocorrência de X, ou seja, zero ou uma ocorrência do caractere que precede o ?.
X{n}	Exactamente n ocorrências de X.
X{n,}	Selecciona n ou mais ocorrências de X. Ou seja, n indica o número mínimo de ocorrências.
X{n,m}	Selecciona de n a m ocorrências de X. Ou seja, n indica o número mínimo de ocorrências e m o máximo.
\d	Representa um dígito.
\D	Representa um **não**-dígito
\w	Representa um caractere alfa-numérico. Também se pode indicar [:alnum:].
\b	Selecciona a fronteira de uma palavra
\<caractere>	Indica que <caractere> deve ser avaliado literalmente. Exemplo: a+ vs. a\+ ⟶ a primeira expressão indica uma ou mais ocorrências de a, ao passo que a segunda indica um a seguido de um +.
(…)	Agrupa sub-expressões regulares.

*Como podemos constatar, a maioria dos metacaracteres utilizados em expressões regulares possuem também significado para a shell. Deste modo, devemos introduzir as expressões regulares entre aspas ou plicas de modo a que esses caracteres não sejam expandidos pela própria shell. Por exemplo, para a shell o significado dos metacaracteres * e o $ não é o mesmo que o **grep** lhes atribui. Para a shell, o * representa zero ou mais caracteres numa determinada parte de um nome de ficheiro. Para o **grep**, o * representa zero ou mais ocorrências de um operando à esquerda do *.*

4. Introduza agora os seguintes comandos e registe por escrito o que observa e qual o significado da expressão regular utilizada:

```
$ grep -E '^Bat' heroes.txt
$ grep -E '^bat' heroes.txt
$ grep -E '^(bat|Bat|cat|Cat)' heroes.txt
```

```
$ grep -i -E '^(bat|cat)' heroes.txt
$ grep -E '^[bBcC]at' heroes.txt
$ grep -i -E '^[bc]at' heroes.txt
$ grep -i -E '[bc]at' heroes.txt
$ grep -i -E 'man' heroes.txt
$ grep -i -E 'man$' heroes.txt
$ grep -E '^[A-Z]' heroes.txt
$ grep -E '^[^A-Z]' heroes.txt
$ grep -E '^[[:upper:]]' heroes.txt

$ grep -E 'anos' texto.txt
$ grep -E 'anos?' texto.txt
$ grep -E 'Go{2,}gle' texto.txt
$ grep -E '^Go{2,}gle' texto.txt
$ grep -E 'G(oo){2,}gle' texto.txt
$ grep -E '((na|os)\ ){2,}' texto.txt
$ grep -E '(\b(na|os)\ ){2,}' texto.txt
$ grep -E '(\b(na|os)\b){2,}' texto.txt
```

EXERCÍCIOS PRÁTICOS

1. Considere o ficheiro dados.txt com o seguinte conteúdo:

```
1  Alberto 10
2  Armando 20 Alberto 30
3  30 Bruno
4  20 Alberto
5  10 alberto 20 armando
6  bruno 30
7  100 ALBERTO 20 Bernardo
8  ### FIM
```

Qual a saída dos seguintes comandos?

(a) `grep '10' dados.txt`

(b) `grep -v '10' dados.txt`

(c) `grep -i -n  '0 a' dados.txt`

***NOTA**: `'0'` (zero) seguido de um espaço seguido de `'a'`*

(d) `grep -E '^A' dados.txt`

(e) `grep -E '^[[:alpha:]]' dados.txt`

(f) `grep -vE '[[:digit:]]+' dados.txt`

(g) `grep -E '20 (B|Al)' dados.txt`

(h)
```
grep -E '[[:alpha:]]{7}' dados.txt |
grep -v '20'
```

2. Escreva expressões regulares apropriadas para utilizar com o **`grep`** para obter:

2.1 Linhas iniciadas por uma vogal.

2.2 Linhas terminadas numa vogal ou num dígito.

2.3 Linhas iniciadas por uma vogal ou terminadas num dígito. Linhas que verifiquem ambos os critérios ou nenhum deles devem ser rejeitadas.

2.4 Linhas com palavras com duas vogais apenas. Por palavra, entenda-se uma sequência de letras, isto é, de vogais e consoantes. Ignore acentos e a capitalização das letras (considere apenas minúsculas). As palavras podem ser delimitadas por qualquer caractere excepto por uma letra. Exemplos de linhas válidas: ae, kae, kale, kalep, aek, akel, kkallepp, kalep3, #kalep. Nos

exemplos anteriores, a palavra que sua expressão regular deve identificar encontra-se sublinhada.

Exemplos de linhas inválidas: `a`, `a e`, `aei`, `kal`, `kalepo`, `#kal`

2.5 Linhas com comentários de linha, que são comentários que começam com '//' e abrangem todos os caracteres até ao final da linha.

2.6 Linhas vazias, isto é, com zero caracteres além do caractere de fim de linha (ou seja, este é o único caractere presente na linha).

2.7 Linhas em branco, isto é, linhas vazias ou linhas contendo apenas caracteres "brancos" (*whitespace*) e o fim de linha. Como pode utilizar o **`grep`** e esta expressão regular para obter uma cópia de um ficheiro contendo apenas as linhas com conteúdo, isto é, as linhas "não-brancas"?

2.8 Linhas que contenham um padrão com uma letra minúscula seguida de dois ou quatro dígitos apenas.

2.9 O mesmo que o anterior mas o padrão não pode estar nem no ínicio nem no fim da linha.

2.10 Linhas com as palavras Linux ou UNIX. Depois repita o exercício mas agora pretende linhas com Linux e UNIX.

2.11 Linhas com números. Os números contêm um, dois ou três dígitos e devem estar delimitados por um caractere de espaçamento. Repita o exercício, mas agora os números devem estar delimitados por um caractere que não faça parte de uma palavra (ou seja, devem ser delimitados por uma "fronteira de palavra").

2.12 Como no exercício anterior, mas os números são seguidos de um ponto (e continuam a ser delimitados por um espaço).

2.13 Linhas com "pseudo-endereços" IP. Um pseudo-endereço IP é uma sequência de quatro números, todos com um, dois ou três dígitos, com os três primeiros números seguidos de um ponto. Todo o pseudo-endereço IP deve estar delimitado por espaçamento. Repita o exercício, mas agora deve aceitar linhas contendo apenas o pseudo-endereço IP. Repita novamente o exercício, mas agora o pseudo-endereço IP deve estar delimitado por "uma fronteira de palavra" (e, claro, também deve continuar a aceitar linhas contendo o pseudo-endereço IP).

2.14 Uma expressão regular para seleccionar um pseudo-endereço de email: uma sequência de caracteres alfanuméricos, começada por uma letra, seguida de uma arroba e de uma sequência de caracteres com duas partes, cada separada por um ponto. A primeira sequência pode conter no máximo um *underscore*.

Exemplos correctos: `xpto@mail.com` , `xpto_ypto@mail.com` , `x123@mail.com` , `x@m.c`

Exemplos incorrectos: `9xpto@mail.com` , `xpto.ypto.zpto@mail.com` , `x123@mail.com.pt` , `xpto_@mail.com` , `xpto_a_b@mail.com`