# Descripció dels símbols emprats més comunment a expresions regulars

Les "expressions regulars" (abreujades sovint com "regexp") fan referència a una determinada cadena que descriu un patró per fer coincidir tot un conjunt de cadenes diferents però que s'ajusten a una determinada regla morfològica. En molts editors de text les expressions regulars s'utilitzen sovint per recuperar i/o substituir contingut de text que s'ajusta a un patró; també són molt usades en formularis web per validar les entrades escrites per l'usuari, i en moltes altres eines, com ara les comandes *grep* o *sed*, entre d'altres (a més d'estar disponibles en la gran majoria de llenguatges de programació). A continuació es mostra una llista i descripció dels símbols que podem incloure en una expressió regular per tal de construir el patró de recerca adient en cada recerca que desitgem fer:

> NOTA: Un "motor" d'expressions regulars és el software encarregat de processar expressions regulars, intentant fer coincidir el patró amb la cadena donada. Normalment, el motor forma part d'una aplicació més gran en forma de llibreria, així que no s'accedeix directament al motor sinó que, més aviat, l'aplicació l'invoca quan és necessari. Actualment existeixen diversos "motors" que, desgraciadament, no són totalment compatibles entre si (és a dir, la sintaxi de les expressions regulars reconegudes per un motor concret -i el seu comportament general- pot diferir en detalls amb la d'altres motors. Així, tenim el motor genèric PCRE, el motor genèric POSIX i varis motors particulars per sengles llenguatges de programació (PHP, .NET, Java, JavaScript, Python, etc).

El símbol "." : Similar al comodí "?" que coneixem del shell Bash: equival a un (i només un) caràcter qualsevol (excepte els caràcters que representen un salt de línia). Per exemple, l'expressió regular "c.sa" pot equivaler a "casa", "cosa", "c1sa" etc (però no a "causa" o "csa"). Si s'escriuen diferents "." seguits s'estarà indicant llavors l'obligatorietat d'un nombre determinats de caràcters (per exemple, "c..." indica qualsevol paraula que comenci amb "c" i que tingui quatre caràcters).

Els símbols"[ ]" : Similar al punt, però obliga a què els caràcters possibles només puguin ser els especificats dins dels claudàtors, i no pas qualssevol com passava amb el ".". Per exemple, "c[iu]sa" només pot ser o bé "cisa" o bé "cusa". També es pot posar un rang de caràcters (per exemple "[a-z]" indica totes les lletres minúscules, "[A-Z]" indica totes les majúscules, "[0-9]" indica tots els dígits, etc).

> NOTA: Existeix la possibilitat d'indicar que la llista de caràcters inclosos dins dels claudàtors siguin els que NO hagin d'aparèixer a l'expressió regular. Això s'aconsegueix indicant com a primer símbol dins dels claudàtors el símbol "^". Per exemple, ,"c[^iu]sa" pot ser "cosa", "casa", "cbsa", etc, però mai "cisa" ó "cusa" (com tampoc paraules que tinguin menys ni més que quatre lletres. També es pot posar un rang de caràcters no desitjat: per exemple "[^a-d]" indica qualsevol caràcter excepte "a","b","c" o "d".
>
> NOTA: Si es vol incloure el símbol "-" com un símbol més dins del claudàtors i fer llavors que perdi el seu significat especial d'indicador de rang de caràcters, s'ha d'"escapar". "Escapar un caràcter" (en aquest cas el guió) vol dir fer-li perdre el seu significat especial i convertir-lo en el caràcter tal qual; això es pot fer per qualsevol altre símbol que tingui algun significat concret en les expressions regulars, com el punt, els claudàtors i els propers que veurem. Per escapar un caràcter simplement cal precedir-lo del símbol "\". Així, doncs, en aquest exemple, per incloure el símbol "-" com un símbol més caldria escriure "\-".

El símbol "*" : Indica que el símbol anterior (o símbols, si aquests estan encerclats per parèntesi) pot aparèixer cap vegada (és a dir, pot no aparèixer!) o més vegades, repetit consecutivament, fins el infinit. Per exemple, "[a-zA-Z]*" pot equivaler a la cadena buida, a una lletra (minúscula o majúscula), a dues lletres qualssevol (minúscules, majúscules o una de cada), a tres lletres qualssevol, a quatre, etc.

> NOTA: Cal tenir en compte que l'asterisc sempre tracta de trobar la major quantitat possible de caràcters (és el que se'n diu "greedy mode", molt ben explicat aquí: https://javascript.info/regexp-greedy-and-lazy) . Això vol dir que, per exemple, una expressió com "a.*b" trobarà dins de la frase "aholabadeubfin" la cadena "aholabadeub" en comptes de "aholab" (que seria el resultat si el motor d'expressions regulars treballés en "lazy mode"). Per treballar en "lazy mode" cal afegir darrera de l'asterisc el símbol "?" (així, "*?"), però no tots els motors ho suporten (per exemple, la comanda *grep* només pot treballar en "lazy mode" si s'hi afegeix el paràmetre *-P*).

El símbol "+" : Indica que el símbol anterior (o símbols, si aquests estan encerclats per parèntesi) pot aparèixer una o més vegades, repetit consecutivament, fins el infinit. Per exemple, "[a-zA-Z]+" pot equivaler a una lletra (minúscula o majúscula), a dues lletres qualssevol (minúscules, majúscules o una de cada), a tres lletres qualssevol, a quatre, etc. Fixeu-vos que la diferència amb l'asterisc és que amb "+" s'obliga a què el/s símbol/s afectat/s en qüestió apareixi com a mínim una vegada. Aquest símbol també funciona per defecte en "greedy mode" però pot activar-se el "lazy mode" si ve seguit d'un símbol "?".

El símbol "?" : Indica que el símbol anterior (o símbols, si aquests estan encerclats per parèntesi) pot aparèixer zero o una vegada. És a dir, que és opcional. Per exemple, "[a-zA-Z]?" pot equivaler a la cadena buida o a una lletra (minúscula o majúscula), ja està.

Els símbols "{}" : Condensa els símbols *,+ i ? vistos anteriorment (i més possibilitats) en una sola sintaxi. Concretament (on "x" i "y" representen un número sencer positiu):

**{x}** : Indica que el caràcter anterior ha d'aparèixer exactament un número x de vegades consecutivament Així, per exemple, en comptes d'indicar "c..." com vam veure al principi, podríem haver escrit "c.{3}"
**{x,}** : Indica que el caràcter anterior ha d'aparèixer com a mínim un número x de vegades consecutivament. Es pot veure que l'expressió "{0,}" seria equivalent a "*" i "{1,}" a "+"
**{x,y}** : Indica que el caràcter anterior ha d'aparèixer com a mínim un número x de vegades consecutivament i com a màxim un número y de vegades. Es pot veure que l'expressió "{0,1}" seria equivalent a "?"

El símbol "$" : Indica que tots els símbols escrits abans d'ell han d'aparèixer obligatòriament al final de la cadena a inspeccionar (o bé, si aquesta està composada de vàries línies, al final de cada línia). Per exemple, si s'escriu l'expressió regular "ion$" el motor trobarà totes les ocurrències on "ion" finalitzi la línia però no on aparegui en qualsevol altra ubicació.

El símbol "^" : Indica que tots els símbols escrits després d'ell han d'aparèixer obligatòriament al principi de la cadena a inspeccionar (o bé, si aquesta està composada de vàries línies, al principi de cada línia). Per exemple, si s'escriu l'expressió regular "^ion" el motor trobarà totes les ocurrències on "ion" comenci la línia però no on aparegui en qualsevol altre ubicació.

NOTA: Es poden combinar els símbols "^" i "$" per tal d'indicar expressions regulars similars a com per exemple "^ion$". En aquest cas estaríem buscant línies exclusivament formades per la cadena "ion" (ja que aquesta expressió regular diu que a cada línia no ho pot haver res per davant ni per darrera de la cadena "ion")

El símbol ">" : Indica que tots els símbols escrits abans d'ell han d'aparèixer obligatòriament al final de paraula (és a dir, davant d'un espai en blanc). Per exemple, si s'escriu l'expressió regular "ion>" el motor trobarà ocurrències com "orion" però no "ionitzar" o "Baiona". La comanda *grep* necessita escapar aquest caràcter per a què funcioni

El símbol "<" : Indica que tots els símbols escrits després d'ell han d'aparèixer obligatòriament al principi de paraula. (és a dir, darrera d'un espai en blanc). Per exemple, si s'escriu l'expressió regular "<ion" el motor trobarà ocurrències com "ionitzar" però no "orion" o "Baiona". La comanda *grep* necessita escapar aquest caràcter per a què funcioni

El símbol "\" : S'utilitza per "escapar" el següent símbol que aparegui a l'expressió regular, de forma que aquest deixi de tenir el seu significat especial, si és que en té o que n'agafi d'algun significat especial si no en tenia.. Per exemple, en el primer cas, l'expressió "\." serveix per buscar el caràcter "punt" literal. Similarment, l'expressió "\.$" serveix per buscar les línies que finalitzin per un punt. En el segon cas, ens podem trobar amb les següents possibilitats (¡només si fem servir el motor PCRE, però!; si usem *grep* això es faria indicant el paràmetre *-P* en comptes de *-E*):

\d : Representa un dígit (del 0 al 9)
\w : Representa qualsevol caràcter alfanumèric
\s : Representa un espai en blanc
\D : Representa qualsevol caràcter que no sigui un dígit
\W : Representa qualsevol caràcter no alfanumèric
\S : Representa qualsevol caràcter que no sigui un espai en blanc
\A : Representa l'inici de la cadena (no és un caràcter sinó una posició)
\Z : Representa el final de la cadena (no és un caràcter sinó una posició)
\b : Marca l'inici i el final d'una paraula

NOTA: Per defecte aquests símbols funcionen en "greedy mode" però es pot activar-se el "lazy mode" si s'escriuen seguit d'un símbol "?".

El símbol "|" : Indica vàries opcions. Per exemple, l'expressió regular "nord|sud" cercarà qualsevol ocurrència de "nord" o "sud" dins del text inspeccionat.

Els símbols "( )" **:** Serveix per agrupar expressions (les quals poden incloure al seu torn altres símbols especials). S'usa molt juntament amb els símbols *,+,?,{nº...} per a què aquests no afectin només al caràcter d'abans sinó a tot el conjunt dins dels parèntesis. Per exemple, l'expressió "(ab){3}" equival a "ababab". També s'usa molt amb el símbol "|" per definir els límits de les opcions a escollir; per exemple, l'expressió "(ca|co)sa" pot equivaler tant a "casa" com a "cosa"; un altre exemple: l'expressió "Nov(\.|embre|ember)?" permet trobar tant "Nov" com "Nov.", "Novembre" i "November".

> NOTA: Un bon tutorial que resumeix tot el que s'ha explicat als paràgrafs anteriors (i més) és https://qntm.org/re_en . Un altre bon resum és https://www.redhat.com/sysadmin/regex-grep-data-flow . També és interessant la lectura dels articles https://dev.to/ziishaned/learn-regex-the-easy-way-c4g i https://dev.to/awwsmm/20-small-steps-to-become-a-regex-master-mpc

A més dels símbols anteriors, també es poden escriure expressions especials que permeten simplificar una mica la construcció d'expressions regulars. Per exemple:

**[[:alnum:]]** : Equival a [0-9A-Za-z]
**[[:alpha:]]** : Equival a [A-Za-z]
**[[:lower:]]** : Equival a [a-z]
**[[:upper:]]** : Equival a [A-Z]
**[[:digit:]]** : Equival a [0-9]
**[[:xdigit:]]** : Equival a [0-9A-Fa-f]
**[[:punct:]]** : Equival a 1 dels caràcters `! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~`
**[[:graph:]]** : Equival a [[:alnum:]] + [[:punct:]]
**[[:print:]]** : Equival a [[:graph:]] + l'espai en blanc
**[[:blank:]]** : Equival a l'espai en blanc i al tabulador
**[[:space:]]** : Equival a l'espai en blanc, al tabulador, al salt de línia, retorn de carro i tabulador vertical

Un exemple: per trobar codis postals, podríem utilitzar l'expressió "[[:digit:]]{5}"

---

Per saber més sobre expressions regulars, recomano la consulta de http://www.regular-expressions.info , web que conté moltíssima informació sobre aquest tema. Altres recursos online que són interessants són:

* https://regex101.com : "Taula de treball" web que permet indicar interactivament un text i una expressió regular i veure en temps real segons els canvis que es van realitzant en un o en l'altre, quina porció del text concorda amb l'expressió regular indicada (segons diferents "motors" a escollir). A més a més informa de per què apareixen les coincidències i per què no. Perfecte per aprendre de forma autodidacta.

* https://regexr.com : Similar a l'anterior. Inclou una referència online dels símbols més utilitzats en regexp.

* https://www.regexpal.com : Similar a les anteriors

* https://regexone.com : Web que inclou vàries lliçons explicatives de cada símbol que pot formar part d'una expressió regular, les quals inclouen també exercicis interactius per confirmar l'asssoliment de cada lliçó

* http://www.regexplanet.com : Permet veure com interpreten la expressió regular introduïda diferents llenguatges de programació. A més, al seu apartat "Cookbook" vénen molts exemples d'expressions regulars habituals (codis postals, telèfons, adreces IP, etc) i com s'implementarien en diferents llenguatges.

> NOTA: Una web similar però només enfocada al llenguatge Python és https://pythex.org

* https://ihateregex.io/expr : Web que disposa de diferents patrons comuns (nom d'usuari, adreces IP, etc) i, a partir d'un cercado indicant el tipus de patró ("username", "IP", data, etc) mostra diferents alternatives d'expressions regulars que es podrien fer servir per concordar amb el tipus indicat, juntament amb una explicació

* https://regexcrossword.com : Videojoc interactiu que permet aprendre expressions regulars de forma interactiva i divertida intentant endevinar quina expressió seria l'adient donat un text proposat (o viceversa). Un altre videojoc és https://alf.nu/RegexGolf Un altre és https://www.therobinlord.com/projects/slash-escape

ADDENDUM: Comanda *grep*

| | |
|---|---|
| *grep "paraula" unarxiu* | Mostra les línies de l'arxiu especificat on es troba la paraula buscada |
| *{ -E \| - P }* | Obligatori si el que es vol buscar no és una paraula exacta sinó una expressió regular ("regexp"). Serveix per triar el "motor": o POSIX o PCRE, respect. |
| -*i* | Trobarà la "regexp" buscada ja sigui escrita amb majúscules o minúscules. Per defecte *grep* només troba "regexps" que siguin escrites tal qual |
| -c | Conta el número de línies que contenen la "regexp" especificada |
| -n | Mostra a més de les línies on es troba la "regexep" buscada, el nº de cada una |
| -v | Inverteix la recerca: mostra les línies que NO contenen la "regexp" buscada |
| -o | En comptes de mostrar la línia sencera on es troba la paraula, només mostra la paraula concreta trobada concordant |
| -r | Si en comptes d'indicar un arxiu es pot indicar una carpeta, aquest paràmetre realitza una recerca de la "regexp" a tot el seu contingut de forma recursiva. Es pot combinar amb *-n, -v*... |
| -l | En comptes de mostrar la línia on es troba la "regexp" només mostra els noms del fitxers on està. Sols té sentit si es combina amb *-r* o fent ús de comodins |
| -A nº | Mostra les nº línies següents a cada línia on s'hagi trobat la "regexp" buscada |
| -B nº | Mostra les nº línies anteriors a cada línia on s'hagi trobat la "regexp" buscada |
| -q | No mostra res. Per saber si ha trobat alguna cosa caldrà veure el valor de la variable especial *$?* (que valdrà 0 si sí ha trobat alguna cosa i 1 si no s'ha trobat res). Això és útil per scripts. |
| -a | Permet buscar el patró textual indicat dins d'un fitxer binari (és a dir, que no sigui de text). |
| -m nº | Mostra només les primeres nº línies amb la "regexp" indicada, la resta, si n'hi ha, no les mostra |

NOTA: Per a què *grep* accepti els símbols *\t* o *\n* s'ha d'escriure el paràmetre *-P* en comptes de *-E*. Una altra opció seria usar "ansi c quotes" així (¡aquí les cometes simples són importants!): *grep -E $'.{2}\t.{3}' fitxer.txt* (https://www.gnu.org/software/bash/manual/html_node/ANSI_002dC-Quoting.html) ; bàsicament la idea és que escrivint $'...' l'interior de les cometes s'interpreta pel shell directament. Si no es vol aquesta opció tampoc, sempre es pot pulsar el tabulador explícitament amb CTRL+V+TAB