[[1]](#footnote-2)

Patrones GLOB (Febrero 2014)

Raúl Andrés Ortiz Fuentes, 112508.

*Índice de términos*—

# INTRODUCCIÓN

E

n el siguiente articulo se da una descripción detallada sobre los patrones *glob*, así como su sintaxis y una serie de ejemplos que complementan los dos puntos anteriores.

# Definición y Descripción

"Glob" es el nombre común de un conjunto de características del Bash (línea de comandos) que igualan o amplían determinados tipos de patrones y se utiliza especialmente para referirse a la coincidencia de patrones basados en caracteres comodín o generación de nombres de caminos globbing.[1]

El módulo glob encuentra todas las rutas que concuerdan con un patrón específico, según las normas de la shell de Unix, ya que esta tratará de igualar los patrones GLOB ingresados contra los componentes de las rutas o pathname, para obtener una lista de los archivos o elementes existentes dentro de dicha ruta. Por tal motivo pueden igualar o ampliar algunos caracteres, incluso los: espacios, salto de línea y caracteres no imprimibles. Sin embargo el único carácter que no se puede igualar es el slashes “/”, ya que este es el separador de los componentes (palabra) o partes de la ruta. [2]

Una cadena de caracteres es un patrón con comodines si contiene uno al menos de los caracteres `?', `\*' o `['. La generación de nombres de caminos es la operación que expande un patrón con comodines formando una lista de nombres de caminos de ficheros que concuerdan con el patrón. El Shell interpreta el estándar de wild card characters, lista de caracteres en paréntesis cuadrados y ciertos caracteres especiales (como ^ o ! en el inicio para la negación).

## Sintaxis de los patrones GLOB

No existe un estándar para la generación de patrones, sin embargo existen una lista de caracteres que permiten la expansión de los patrones:

1. **Carácter “?”**

Si una expresión se encuentra contenido el carácter “?”, éste se corresponde con uno y sólo un carácter. [3]

**Ejemplos:**

*# Ignora los ficheros bg-01.jpg bg-lt.jpg*

*#*

*# No ignora los ficheros bg-1.jpg y bg-001.jpg*

*#*

*bg-??.jpg*

Este patrón ignorará los ficheros cuyos nombres empiezan por “bg-”, van seguidos exactamente de dos caracteres, y luego van seguidos de “.jpg”. Por eso el fichero bg-1.jpg no se ignora, porque entre “bg-” y “.jpg” no hay dos caracteres, sólo uno. Este ejemplo ilustra que el símbolo “?” no indica un carácter opcional.

1. **Carácter “\*”**

Si una expresión se encuentra contenido el carácter ” \* “, éste se corresponde con cualquier número de caracteres. [3]

**Ejemplos:**

*# Ignora todos los ficheros acabados en .nib*

*#*

*# Ignora tanto MainWindow.nib y builds/MainWindow.nib*

*\*.nib*

*#Ignora todos los ficheros que empiecen por "bg-" y acaben en ".jpg"*

*bg-\*.jpg*

En el primer ejemplo, estamos ignorando cualquier fichero que acabe en .nib, independientemente de la carpeta en la que se encuentre. En el segundo ejemplo, estamos ignorando cualquier fichero que empiece por “bg-” y acabe por “.jpg”. Una pregunta: ¿se ignorará el fichero “bg-.jpg” usando este patrón?. La respuesta la tenéis un poco más abajo.

1. **Carácter “[]”**

Si una expresión se encuentra contenido los caracteres “[ ]”, éstos se corresponden con uno y sólo un carácter de los que están incluidos en los corchetes. Además, de los proceso. [3]

**Ejemplos:**

*# Ignora los ficheros \*.o y \*.a*

*\*.[ao]*

*# Ignora los ficheros .nib y .xib*

*\*.[xn]ib*

Estos patrones no ignoran los ficheros liberia.so o los ficheros mainwindow.ib. Es necesario que alguno de los caracteres que están entre corchetes aparezcan en el fichero.

Si se necesita utilizar los caracteres ? \* [ ] en un patrón, estos deberán escaparse con \

**Ejemplo:**

*# Ignorando ficheros que contienen [ y ] en el nombre*

*# Este patrón ignorará los ficheros*

*# cache/cache-[hG7].cache y cache/cache-[009].cache*

*cache/cache-\[???\].cache*

## Complementación

Se puede anular el funcionamiento de los caracteres `?', `\*' y `[' haciéndolos preceder de una barra inclinada invertida `\', o, en caso de formar parte de una orden del shell, encerrándolos entre comillas. Entre corchetes, estos caracteres se representan a sí mismos. Así, `[[?\*\]' concuerda con uno de los cuatro caracteres `[', `?', `\*' y `\'. Adicional, se puede realizar la combinación de diferentes caracteres para obtener patrones más complejo. [4]

## Ejemplos

TABLA I

Ejemplos Patrones GLOB

|  |  |  |
| --- | --- | --- |
|  | **PATRON** | **DESCRIPCIÓN** |
| 1 | ls +(ab|def)\*+(.jpg|.gif) | Lista todos los archivos JPEG y GIF que inician con las letras "ab" o "def" |
| 2 | rm [A-Z]\*.jpg | Remueve los archivos JPEG que inicial con letra mayúscula. |
| 3 | ls ab+(2|3).jpg | Lista los archivos que cumple con la expresión regular dada. |
| 4 | ls \*!(.jpg|.gif) | Lista todos los archivos que no son JPEGs o GIFs. |
| 5 | ls !(+(ab|def)\*+(.jpg|.gif)) | Lista todos los archivos que no son de tipo JPEG o GIF y que inicia con "ab" o "def". |
| 6 | ls -alt [a-z]\*.{jpg,gif} | Lista todos los archivos jpg/gif y que inicia con la letra minúscula seguido de cualquier cantidad de caracteres. |
| 7 | ls -alt t{[0-9]\*,[a-z]\*}.{jpg,gif} | Lista los archivos jpg/gif que inician con la letra t seguida de un digito o una letra minúscula y un número indefinido de caracteres |
| 8 | cp !(04\*).mp3 /mnt | Copia todos los archivos MP3 uno a uno al equipo. |
| 9 | [SK][0-9][0-9]???\* | Por ejemplo, para encontrar algo empieza con 'S' o 'K' seguida de dos números, seguido de al menos 3 caracteres |
| 10 | ls {a,b}{c,d}\* | Mostrar todos los archivos cuyo primer carácter es "a" o "b" y la segunda es "c" o "d"] |
| 11 | ls \*.!(Rnw|tex) | Por ejemplo, las siguientes pautas bien coincide con todos los archivos en el directorio actual no tener "Rnw" o "tex" extensión de la línea de comandos |
| 12 | Ls +(fi|fr)\* | Cómo buscar una o más apariciones de archivos basado en primeros caracteres del nombre de archivo |
| 13 | imagen/{autos,espacio[0-9]}/\*.jpg | Todos los archivos cuyo nombre termina en .jpg en los "directorios" se encontrara imagen/autos, imagen/espacio0, ... , Imagen/espacio9, si es que cristina "directorios" se encontrara que hay. |
| 14 | /usr/share/doc/\*/LEAME: | Todos los archivos denominados LEAME en todos los subdirectorios inmediatos del directorio /usr/share/doc. |
| 15 | \*[!a-z] | Todos los archivos en el directorio corriente cuyo nombre no termine con una letra minúscula. |
| 16 | echo /././././tmp/./././\* | Estos patrones coinciden con todos los nombres ocultos en el directorio /tmp (padre): |
| 17 | echo ../../tmp/idallen/\* | Estos patrones coinciden con todos los nombres ocultos en el /tmp/idallen directorio: |
| 18 | ls /usr/\*/date | Lista los archivos en la carpeta date |
| 19 | /\*/\*/\* | Trae todas los archivos contenidos dentro de carpetas y subcarpetas |
| 20 | /tmp/./\* | Estos patrones coinciden con todos los nombres ocultos en el directorio /tmp (padre): |

Referencias

1. mailbizz02, glob, 20130816,Recuperado de: http://mywiki.wooledge.org/glob.
2. Ian D. Allen, 2005, GLOB patterns (wildcard pathname matching), Recuperado de: http://teaching.idallen.com/dat2330/06w/notes/glob\_patterns.txt
3. Alfonso, 20130114, Ignorando ficheros en git – Parte IV: Más patrones, Recuperado de: http://aprendegit.com/ignorando-ficheros-en-git-parte-iv-mas-patrones/.
4. Benjamin Poulain, 1998, Glob, Recuperado de: http://www.linuxcertif.com/man/7/glob/es/

**Primer autor** Raul Andrés Ortiz Fuentes, 112508, Aplicaciones Web Avanzadas

1. [↑](#footnote-ref-2)