grep

3# Guía Completa de `grep`

El comando `grep` (Global Regular Expression Print) es una herramienta fundamental en sistemas Unix/Linux para buscar texto en archivos o en la salida de otros comandos.

Sintaxis Básica

```
grep [opciones] 'patrón' archivo(s)
```

Ejemplo:

```
grep 'error' archivo.log
```

Opciones Principales

Opción	Descripción	Ejemplo	
`-i`	Ignora mayúsculas/minúsculas	`grep -i 'error' archivo.log`	
`-v`	Invierte la búsqueda (excluye coincidencias)	`grep -v 'error' archivo.log`	
`-r` o `-R`	Busca de forma recursiva en directorios	`grep -r 'error' /var/log/`	
`-l`	Muestra sólo los nombres de los archivos coincidentes	`grep -l 'error' *.log`	
`-L`	Muestra archivos que **no** contienen el patrón	`grep -L 'error' *.log`	
`-n`	Muestra número de línea	`grep -n 'error' archivo.log`	
`-c`	Muestra solo el número de coincidencias	`grep -c 'error' archivo.log`	

Opción	Descripción	Ejemplo	
`-o`	Muestra solo el texto que coincide	`grep -o 'error' archivo.log`	
`-w`	Coincidencia exacta de palabra	`grep -w 'error' archivo.log`	
`-x`	Coincidencia exacta de línea	`grep -x 'error' archivo.log`	
`-q`	Modo silencioso (sin salida, devuelve solo código de salida)	`grep -q 'error' archivo.log && echo "Encontrado"`	
`-e`	Permite definir múltiples patrones	`grep -e 'error' -e 'fail' archivo.log`	
`-f`	Usa un archivo de patrones	`grep -f patrones.txt archivo.log`	
`-P`	Usa expresiones regulares Perl (PCRE)	`grep -P '\d{3}-\d{2}-\d{4}' archivo.txt`	
`-E`	Usa expresiones regulares extendidas (equivale a `egrep`)	`grep -E 'error	fail' archivo.log`
`-F`	Busca cadenas fijas (sin interpretar regex) (equivale a `fgrep`)	`grep -F 'cadena literal' archivo.txt`	

Combinaciones útiles

- Ver líneas alrededor de la coincidencia:

```
grep -C 3 'error' archivo.log   # 3 líneas antes y después
```

- Solo antes:

```
grep -B 2 'error' archivo.log
```

- Solo después:

```
grep -A 4 'error' archivo.log
```

Expresiones Regulares Básicas

Patrón	Significado	Ejemplo
`.`	Cualquier carácter	`grep 'c.t' archivo.txt`
`^`	Inicio de línea	`grep '^Error' archivo.log`
`$`	Fin de línea	`grep 'fin$' archivo.txt`
`*`	Cero o más ocurrencias	`grep 'lo*' archivo.txt`
`[]`	Rango de caracteres	`grep '[0-9]' archivo.txt`
`\`	Escapar un carácter especial	`grep '\.' archivo.txt`

Búsqueda Recursiva Ejemplo

```
grep -ri 'password' /etc/
```

Usando `grep` en Pipelines

```
dmesg | grep -i 'usb'
ps aux | grep 'apache'
ls -l | grep '^d'
```

Guardar Resultados

```
grep 'error' archivo.log > errores.txt
```

Códigos de Salida

- `0` → Se encontró al menos una coincidencia
- `1` → No se encontró ninguna coincidencia
- `2` → Error de uso o problema con archivos

Variantes de `grep`

Comando	Descripción
`egrep`	Equivalente a `grep -E`
`fgrep`	Equivalente a `grep -F`

Nota: En versiones modernas todos son enlaces simbólicos a `grep`.

Ejemplos Avanzados

1. Buscar varias palabras:

```
grep -E 'error|fail|critical' archivo.log
```

2. Buscar patrón exacto ignorando mayúsculas:

```
grep -iw 'Error' archivo.log
```

3. Buscar y contar:

```
grep -c '404' access.log
```

4. Mostrar nombre de archivo y número de línea:

```
grep -Hn 'fatal' *.log
```

Esta es una referencia rápida pero completa para dominar `grep` en cualquier entorno Unix/Linux.