AWK se parece más a un lenguaje de programación que a un simple comando. Su especialidad es procesar información en modo texto.

Se puede ejecutar con instrucciones simples o crear script completos que pueden leer información, usar variables, operaciones matemáticas, estructuras condicionales, etc.

Podrá leer ficheros de texto o la entrada estándar, igual que otros comandos como sed.

Divide el texto en "registros" (cada linea) y "campos" (cada palabra), de forma predeterminada y cuenta con variables predefinidas que nos ayudan a gestionar la información a procesar.

Cada "campo" (normalmente una palabra) de una linea se asocia a la variable `$N`, siendo N el número de "columna".

Cuando llamamos a awk le indicamos entre comillas simples lo que debe hacer, con el formato: `condición { acción }`

- **Condición**: puede ser que en la línea que se tiene que aplicar la acción se deba cumplir una expresión regular, en cuyo caso iría entre barras: `/regex/ { acción }`
  O que se cumpla una expresión, por ejemplo: `$2 > 1 { acción }`
- **Acción**: Lo que debe hacer en cada linea que se cumpla la condición.
  Por defecto, mostrar la línea: `print` o `printf`

Ejemplos:

```
awk '/root/' /etc/passwd
date | awk '{print $5}
```

Variables predefinidas

- **$N**: Contenido del campo N
- **$0**: Toda la linea que acaba de leer
- **FS**: Especifica un separador de campos.
- **RS**: Especifica el separador de registros.
- **NR**: Número de la línea que está siendo procesada.
- **NF**: Total de campos procesados.

Se pueden indicar varias lineas de condición{acción} y también diferenciar una acción para el inicio y otra para el final, con las condiciones BEGIN y END

```
BEGIN {acción}
patrón {acción}
   ...
patrón {acción}
END {acción}
```

Principales opciones

- **-F** modifica el carácter de separación (FS)
- **-f** indica el fichero donde estarán las instrucciones
- **-v** permite pasarle una variable con un valor

Ejemplos

```
awk -F: '{print $1}' /etc/passwd
awk -f instrucciones datos.csv
awk -v EDAD=14 '$3<EDAD' datos.csv
```

Funciones predefinidas

- **Matemáticas**
  - `atan2(y, x), cos(expr),sin(expr), exp(expr), int(expr), log(expr), sqrt(expr), rand(), srand(expr)`
- **De Cadenas**
  - `gsub(b,c,s)` **sustituir** `b` por `c` en la cadena `s` (si no se pone será $0)
  - `index(c,s)` devuelve la **posición** de `c` en `s`
  - `length(c)` **longitud** de la cadena `c`
  - `match(c,r)` **posición** en `c` donde ocurre la expresión regular `r`

https://www.gnu.org/software/gawk/manual/html_node/Built_002din.html

Funciones predefinidas

- **De Cadenas**
  - `split(c,a,r)` parte la cadena `c` en el array `a` en función del separador determinado por la expresión regular `r` (si no se pone será FS)
  - `printf(f,e)` imprime el/los valor/es `e`, con el formato `f`

    `%d` Número entero `%nd` Número entero formateado a n caracteres
    `%f` Número real. `%n.mf` real con n enteros y m decimales
    `%s` Cadena de carácteres

  - `substr(s,i,n)` devuelve `n` caracteres de `s` comenzando en la posición `i`
  - `tolower(s) / toupper(s)` cambia a minúscula /mayúscula

https://www.gnu.org/software/gawk/manual/html_node/Built_002din.html

Funciones predefinidas

- **Otras**
  - `system(comando)` Devuelve la salida tras ejecutar el `comando`
  - `strftime(f,t)` Devuelve fecha `t` con el formato `f`

```
print strftime("Hoy es = %Y-%m-%d", systime())
```

https://www.gnu.org/software/gawk/manual/html_node/Built_002din.html

Sentencias de control

IF

```
if (condición) {
  instrucciones
} else {
  instrucciones
}
```

FOR

```
for (i=1;i<=50;i=i+1){
  instrucciones
}
```

VALORES DEL ÍNDICE

```
for (i in lista) {
  instrucciones
}
```

WHILE

```
while (condición){
  instrucciones
}
```

```
do {
  instrucciones
}while (condición)
```