Herramientas Grep y AWK

Grep: Globally search for Reg. Expr. and Print
(UNIX - editor “ed” - g/re/ p)

AWK: Aho, Weinberger y Kernighan

Expresiones regulares

- Expresiones simples

x	→	carácter x, si es carácter normal
`.`	→	cualquier carácter
`^`	→	principio del texto, si va al comienzo
`$`	→	fin del texto, si va al final
`[aeiou]`	→	un carácter del conjunto
`[a-z]`	→	un carácter del rango
`[^aeiou0-9]`	→	complementa el conjunto
`\x`	→	carácter x, incluso especial

Expresiones regulares

- Expresiones compuestas

  `x+` → una o más repeticiones de x

  `x*` → cero o más repeticiones de x

  `x?` → cero o una aparición de x

  una `|` otra → una u otra expresión

  `(`x`)` → expresión x

  xy → expresión x seguida de y

- Ejemplo: valor real al comienzo del texto

```
^[+\-0-9][0-9]*\.[0-9]*
```

Herramienta Grep

- Se invoca de la forma:

  `> grep` patrón ficheros...

- Efecto: Lee los ficheros y envía a la salida cada línea que se ajuste al patrón
- Ejemplo:

  `> grep ^/[/\*] *.h *.cpp`

  (obtiene todas las líneas de código que empiecen con un comentario **`//`** o **`/*`**)

Lenguaje AWK

- Esquema general del programa

  Colección de cláusulas → patrón { acción }

  Patrón omitido → se cumple siempre

  Acción omitida → { print } (imprimir línea)

  Patrón /expr.regular/ → debe ajustarse a la línea

  Patrón expr. aritmét. → cumple si no nulo (0 o “”)

- Ejecución de un programa AWK
  - Leer los ficheros de entrada línea por línea y aplicar a cada línea la colección de cláusulas, por su orden.

Lenguaje AWK

- Patrones especiales

  `BEGIN` → se cumple antes de leer la entrada

  `END` → se cumple al final de todo el proceso

  pOn, pOff → lo cumple un rango de líneas
  la acción se ejecuta siempre que esté activa
  se activa cuando se verifica pOn
  se desactiva en la siguiente línea a la que verifica pOff

- Código de las acciones
  - Las acciones se escriben como en lenguaje C
  - No hay que declarar las variables (se crean al usar con valores nulos)
  - Valores numéricos o de texto, indistintamente.

Lenguaje AWK

- Invocar la herramienta AWK

```
awk  "programa"  ficheros...
awk  -f programa  ficheros...
awk  ... -v var=valor ...
```

  - Programa en la misma orden o en fichero de texto

- Ejemplo:

```
awk "/^[0-9]/ {print $1}" *.txt
```

  - Imprime el primer campo de cada línea de un fichero de texto que empiece por un número

Lenguaje AWK

- Campos en las líneas de entrada:
  - $n → *n*-simo campo
  - $0 → toda la línea

  - $0 = “ejemplo de línea de texto”

    $1 = “ejemplo” $2 = “de” $3 = “línea”

    $4 = “de” $5 = “texto”

  - La referencia a un campo puede ser calculada:

    k = 3 $k = “línea”

Lenguaje AWK

- Algunas variables predefinidas:
  - `NF` → número de campos
  - `NR` → número de la línea (global)
  - `FNR` → número de la línea (local al fichero)
  - `FILENAME` → nombre del fichero actual

  Las variables anteriores toman valor automáticamente con cada línea
  - `FS` → sep. de campos de entrada (" ")
  - `RS` → sep. de líneas de entrada ("`\n`")

Lenguaje AWK

- Algunas sentencias de uso frecuente:
  - var = expresión
  - `if` (condición) acción [`else` acción]
  - `while` (condición) acción
  - `for` (k=ini; k<=fin; k++) acción
  - { sentencia; sentencia ... }
  - `print` [expresión, expresión ...]
  - `printf` (formato, expresión, expresión ...)

Lenguaje AWK

- Tablas (vectores asociativos)
  - Colección de pares: (clave, información)
  - Notación de *array*: tabla`[`clave`]` = información
  - Claves de cualquier tipo (números o texto)
  - Recorrido:

    `for (`clave `in` tabla`)` acción
  - Creación automática, al referirse a un elemento
  - Destrucción explícita:

    `delete` tabla`[`clave`]` `delete` tabla

Ejemplo AWK: Concordancias

- Frecuencia de aparición de cada palabra

```
{
    for (k=1; k<=NF; k++) {
        cuenta[$k]++
    }
}
END {
    for (pal in cuenta) {
        print pal, cuenta[pal]
    }
}
```