



Uso detallado AWK

Realizado por Lukas Häring García

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash
- III. Introducción
 - A. Estructura
 - B. Variables
 - C. Operadores
 - D. Bucles
 - E. Funciones

Tabla de contenidos

- I. Instalación**
- II. Diferencias con Bash
- III. Introducción
 - A. Estructura
 - B. Variables
 - C. Operadores
 - D. Bucles
 - E. Funciones

Instalación

La edición de escritorio de Ubuntu tiene mawk instalado por defecto. Se ajusta al estándar AWK, con algunas extensiones.

Algunas funciones no están implementadas en mawk mientras que en AWK sí, por lo que podemos instalar en Ubuntu la versión más general.

```
> sudo apt-get install gawk
```

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash**
- III. Introducción
 - A. Estructura
 - B. Variables
 - C. Operadores
 - D. Bucles
 - E. Funciones

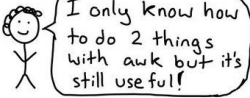
Diferencias con Bash

1. AWK implementa un lenguaje más cercano al C, por lo que es más accesible.
2. Bash es utilizado para ejecutar comandos de la shell (por ejemplo AWK), mientras que AWK tiene como propósito escribir reportes o filtros.
3. AWK tiene arrays asociativos y funciones de manipulación de cadenas.

awk

SULIA EVANS
@b0rk

awk is a tiny programming language for manipulating columns of data



basic awk program structure

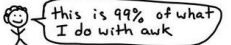
```
BEGIN { ... }  
CONDITION { action }  
CONDITION { action }  
END { ... }
```

do action on lines matching CONDITION

extract a column of text with awk

```
awk -F, '{ print $5 }'
```

column separator, single quotes, print the 5th column



SO MANY unix commands print columns of text (ps! ls!)

so being able to get the column you want with awk is GREAT

A few more awk programs →

sum the numbers in the 3rd column

```
{ s += $3 }  
END { print s }
```

at the end, print the sum!

print every line over 80 characters

```
length($0) > 80
```

condition

(there's an implicit {print} as the action)

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash
- III. Introducción**
 - A. Estructura**
 - B. Variables
 - C. Operadores
 - D. Bucles
 - E. Funciones

Estructura

```
awk 'BEGIN { <código> }  
{  
    <código>  
}  
END { <código> }'
```

ó

```
awk <archivo>
```

Como comando

```
#!/bin/awk -f  
BEGIN { <código> }  
{  
    <código>  
}  
END { <código> }
```

Como ejecutable

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash
- III. Introducción**
 - A. Estructura
 - B. Variables**
 - C. Operadores
 - D. Bucles
 - E. Funciones

Variables

Creamos una nueva variable en el código de la siguiente manera:

1. `awk [-v [<var 1>=<val 1> [-v <var 2>=<val 2> [...]] (código | archivo) [arg1 [arg2 [...]]]`
2. `<código> := <var archivo 1> = <val archivo 1>`

Utilizaremos las variables creadas de la siguiente forma:

`print <var 1>` o `print <var archivo 1>`

Accedemos a los argumentos (argk) a través del vector “ARGV[i]”

Para acceder a las variables del sistema, podemos utilizar la variable hashmap “ENVIRON”.

Podemos inicializar **arrays** `<vector>[<indice>] = <valor>;`

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash
- III. Introducción
 - A. Estructura
 - B. Variables
 - C. Operaciones**
 - D. Bucles
 - E. Funciones

Operaciones

Como se ha comentado anteriormente, es muy parecido a C, por lo que tenemos las operaciones entre variables +, -, *, /, %, etc.

Además tenemos implementada de forma nativa las funciones: sqrt, sin, cos, etc.

Los operadores condicionales, >=, <=, >, <, ==, != están también implementados.

Se permite realizar expresiones regulares: Match: ~/<RE>/ ; No match !~/<RE>/

Operadores de asignación +=, -=, /=, %=, etc.

Y mucho más.

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash
- III. Introducción
 - A. Estructura
 - B. Variables
 - C. Operaciones
 - D. Bucles**
 - E. Funciones

Bucles

Tenemos los bucles típicos de C y algunos más modernos.

```
while ( <condicional > ) <código>
```

```
for( <expresion> ; <condicional> ; <expresion> ) <código>
```

```
for( <variable> in <array> ) <código>
```

Además tenemos los if then else:

```
if(<condicional>) <código> else <código>
```

Tabla de contenidos

- I. Instalación
- II. Diferencias con Bash
- III. Introducción
 - A. Estructura
 - B. Variables
 - C. Operaciones
 - D. Bucles
 - E. **Funciones**

Funciones

Para cadena de caracteres:

1. Función **substring** requiere 3 argumentos: Texto, posición inicial y posición final.
2. **tolower** y **toupper**, modifica la cadena en completamente mayúsculas o minúsculas.
3. La función **match** funciona de forma equivalente a `~`.
4. Podemos separar una cadena en un array con **split**(<cadena>, <vector>, <separador>).
5. Para contar el número de caracteres de una cadena, utilizamos la función **length**.

Podemos ejecutar comandos del sistema mediante la función “**system**(<comando>)”.

Funciones para números:

1. Función de redondeo o casteo, **int** .
2. Función de generación de números aleatorio: **rand**

Referencias

1. <https://www.cse.iitb.ac.in/~br/courses/cs699-autumn2013/refs/awk-tutorial-bruce-barnett.html>
2. <https://askubuntu.com/questions/710379/do-i-need-to-install-awk-or-is-it-inbuilt-in-ubuntu>
3. <https://stackoverflow.com/questions/24822005/difference-between-bash-scripting-and-awk>
4. <https://blog.desdelinux.net/awk-shell-scripting-comando/>