
MANIPULACIÓN DE GRANDES ARCHIVOS DE TEXTO CON AWK

Verónica Jiménez Jacinto

Unidad de Secuenciación Masiva y Bioinformática

Laboratorio Nacional de Apoyo Tecnológico a las ciencias Genómicas

Instituto de Biotecnología

UNAM

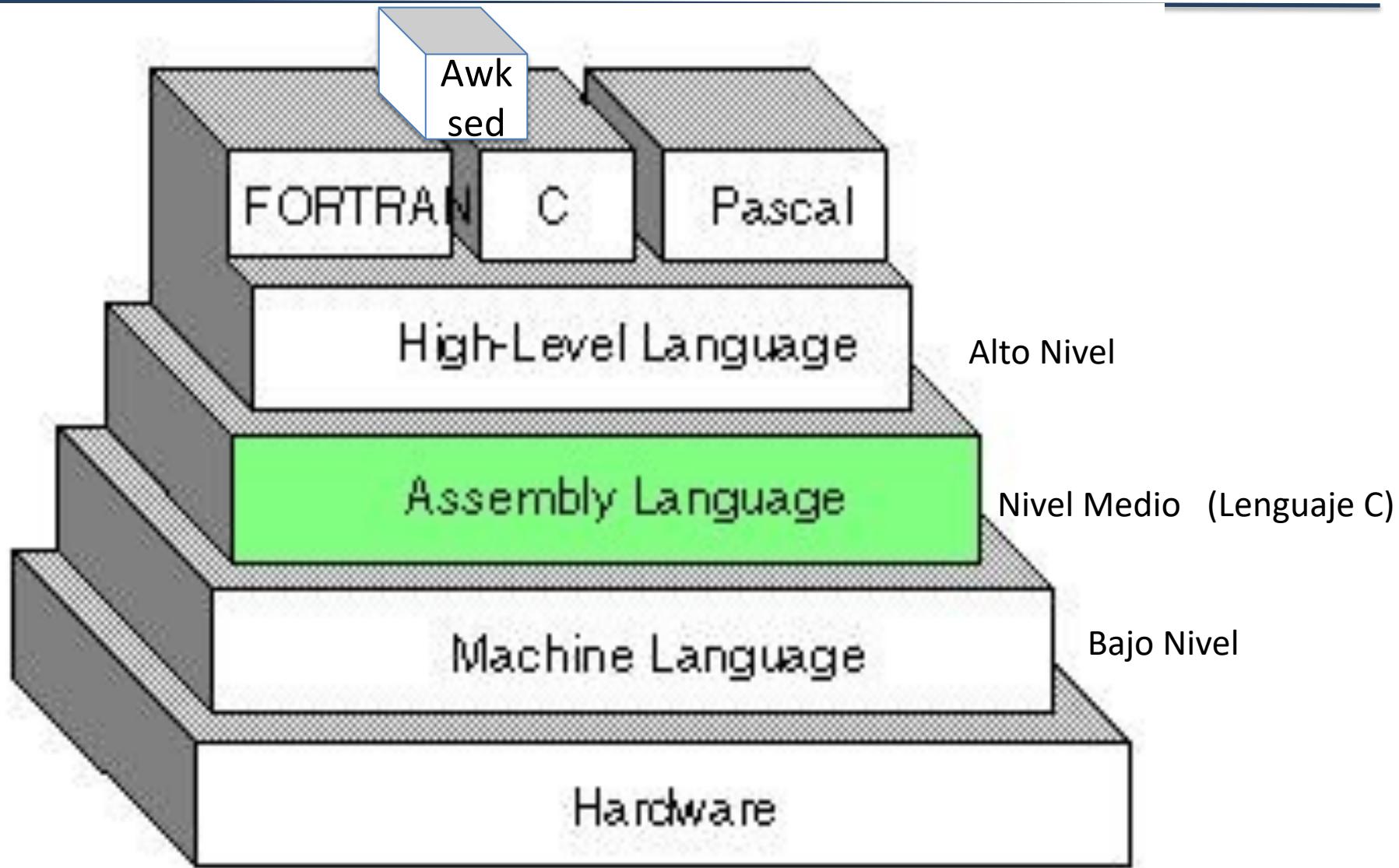
Temario

- Introducción
 - Origen del lenguaje awk
 - Ámbito de aplicación
 - Estructura básica
 - Variables y constantes
- Lectura y escritura de archivos
- Operadores y expresiones booleanas
- Arreglos asociativos

Objetivo

- Conocer un lenguaje que nos permita la manipulación eficiente de archivos de texto de gran magnitud

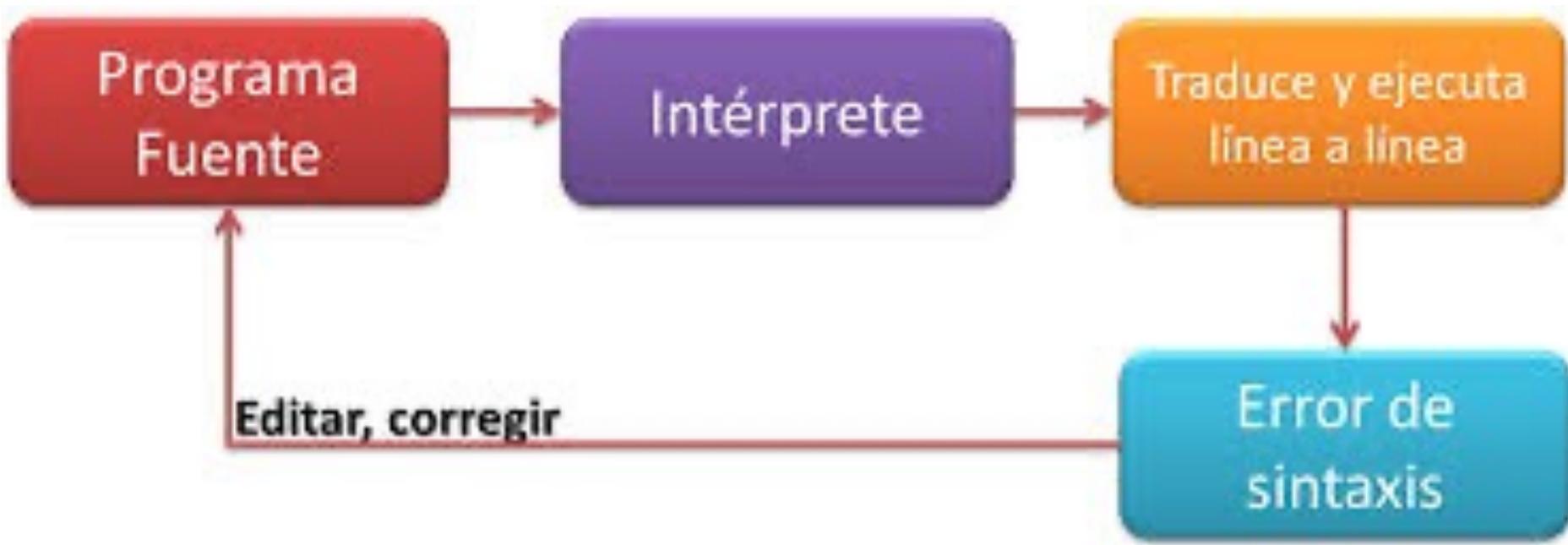
Niveles



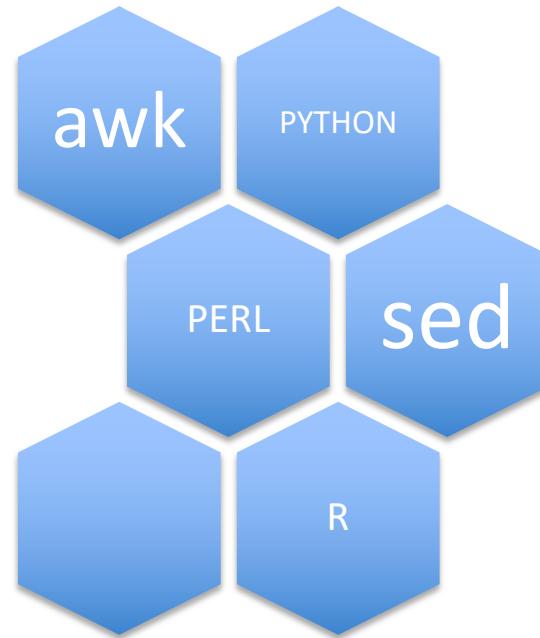
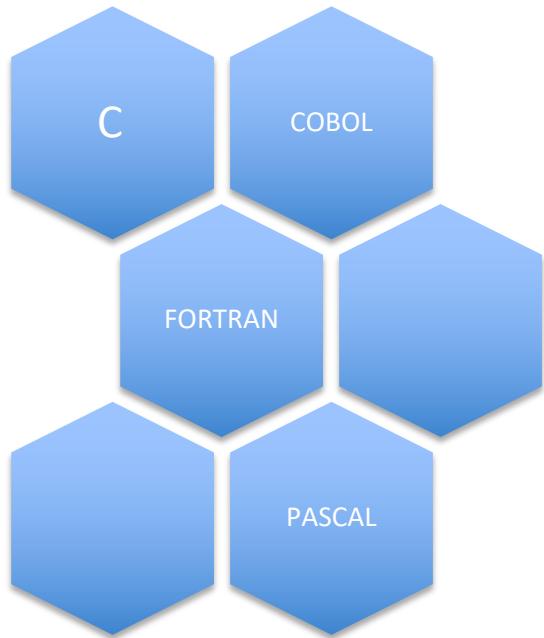
Compiladores



Interpretes...



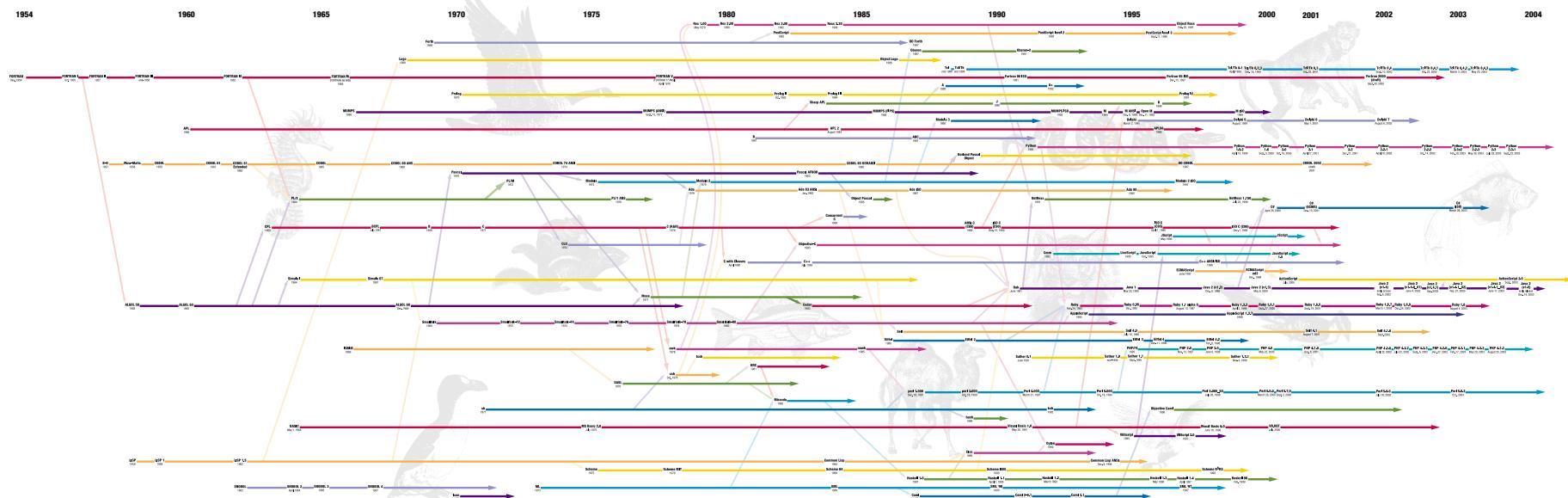
Compilados vs. interpretes



Lenguajes de Programación

History of Programming Languages

O'REILLY®

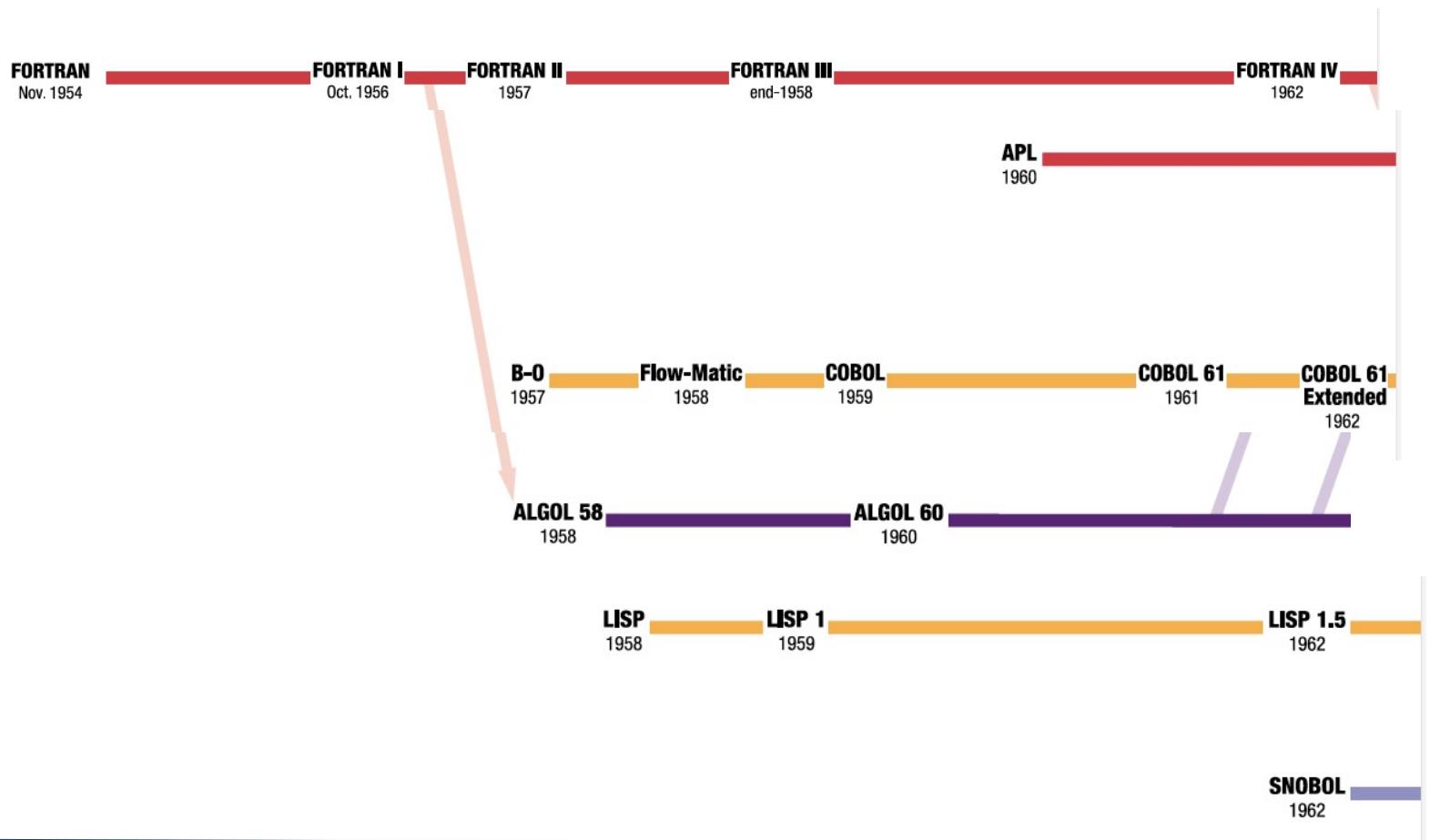


www.oreilly.com

For more than half of the 50 years computer programming have been using code. O'Reilly has created thousands of comprehensive, in-depth technical information. We've kept pace with rapidly changing technology as new languages have emerged and others have matured. Whether you're looking to learn something new or need answers to tough technical questions, you'll find what you need in O'Reilly books and on the O'Reilly Network.

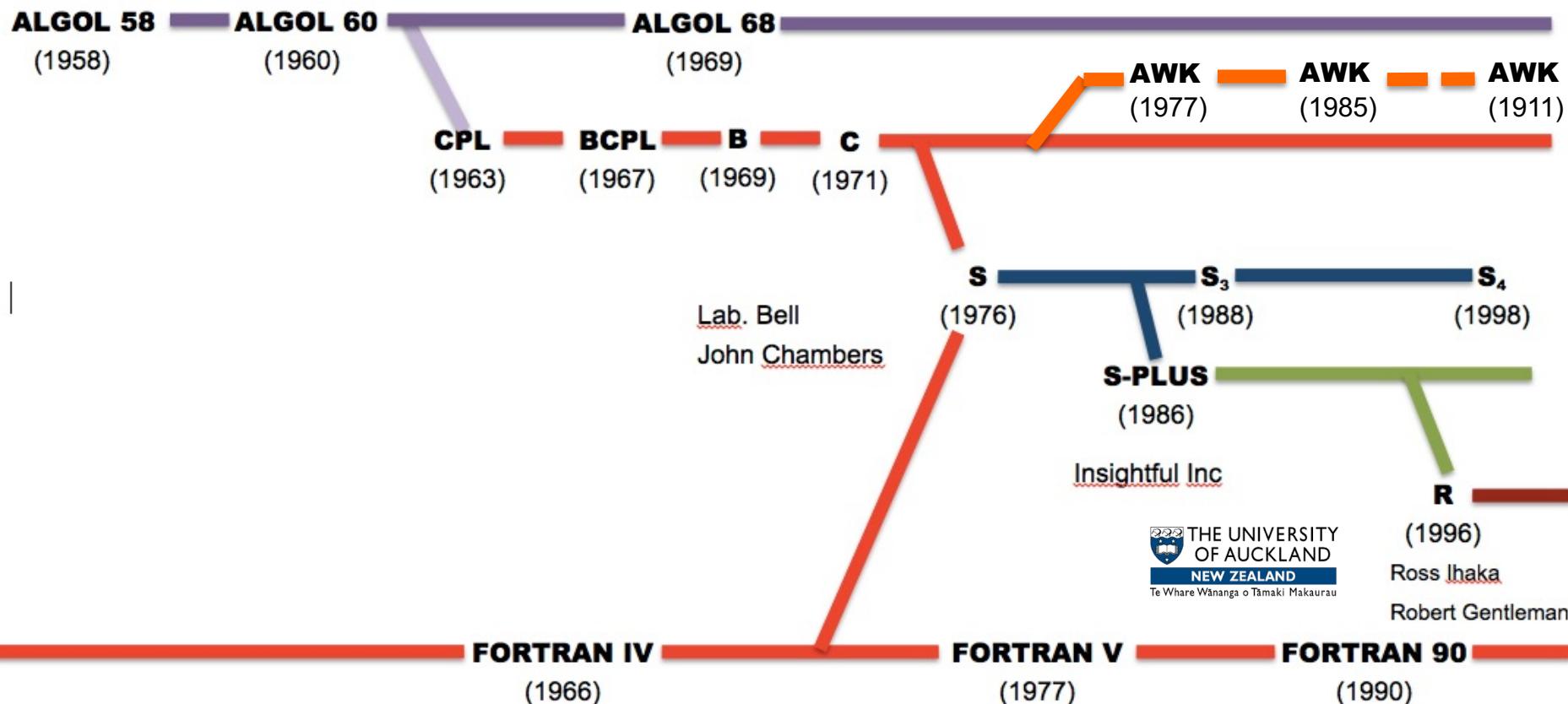


En un inicio...



De los 70's a los 2000...

HISTORIA DE R



AWK

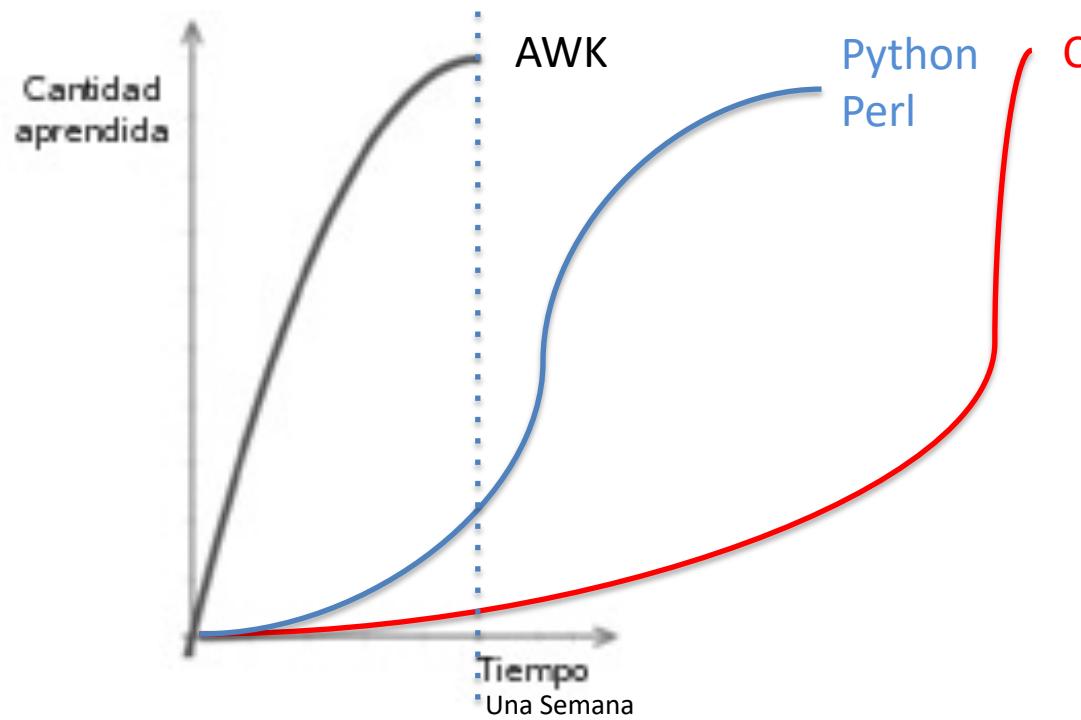
Autores

- Alfred V. Aho
- Peter J. Weinberger
- Brian W. Kernighan

AT&T Bell Laboratories

¿Ventajas de usar AWK?

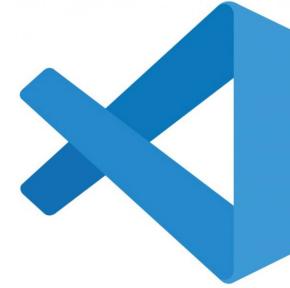
- Curva de aprendizaje logarítmica



- AWK y SED son lenguajes de propósito específico

Utilería básica en cualquier Unix/Linux

- Forma parte de cualquier instalación de Unix o Linux.
- También se puede usar desde cualquier terminal en macOS
- En Windows se puede utilizar en una terminal desde Visual Code o PowerShell, e instalar vía <https://gnuwin32.sourceforge.net/packages.html>



Pre-cuestionario

- <https://forms.gle/UqLbaWNo67Ag3ZiDA>

Para empezar...

Abra una terminal y consulte el manual de awk escribiendo:

```
$ man awk
```

```
# 1. genere una carpeta llamada curso_<alumno>_awk:
```

```
$ mkdir -p curso_vjj_awk/data
```

```
# 2. Localice la carpeta data y haga una liga o copia en su  
# ambiente de trabajo:
```

```
$ cd curso_<alumno>_awk/data
```

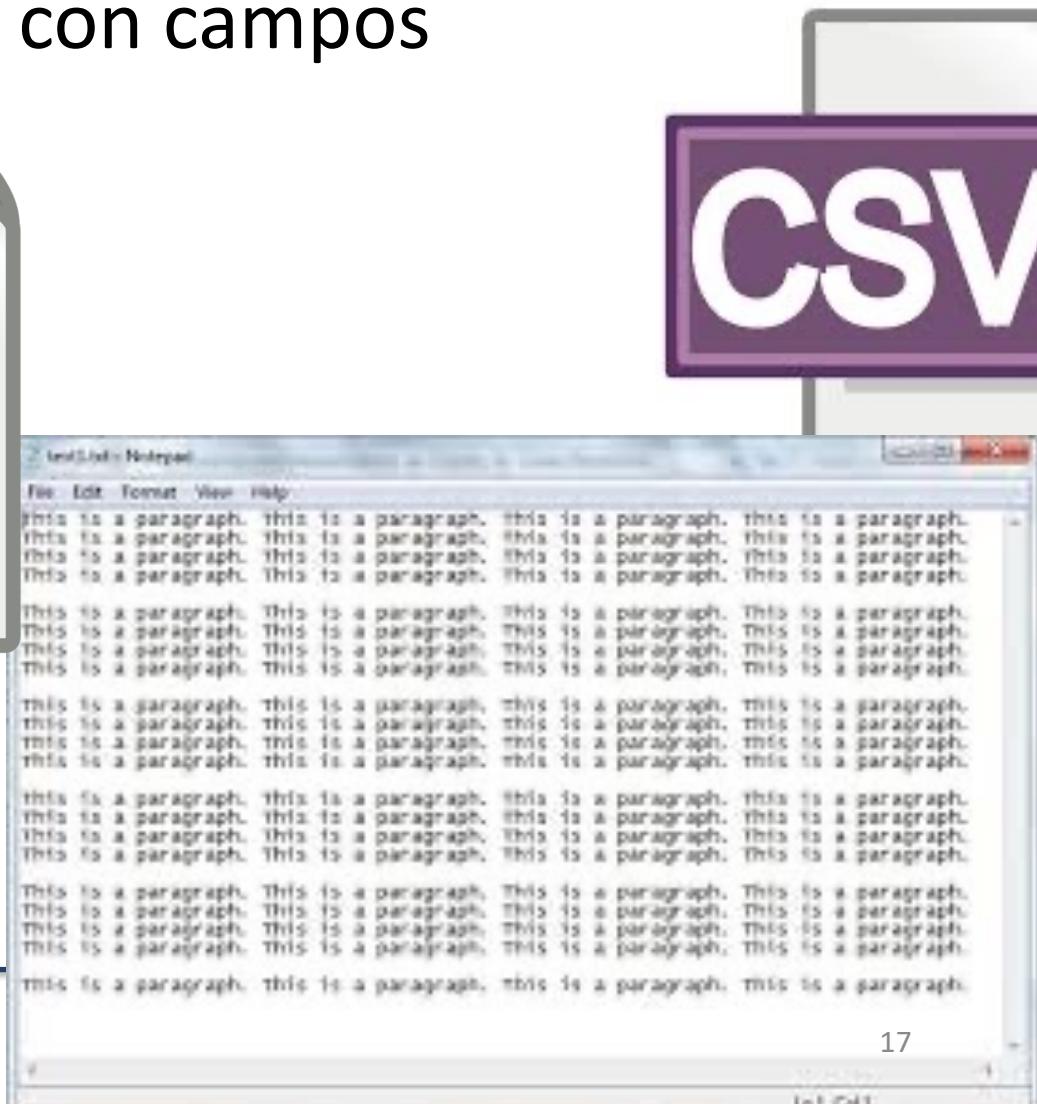
```
#Descarga los archivos de datos...
```

Si tienes windows y no tienes awk...

```
$ ssh invitado@132.248.32.114
$ mkdir -p curso_usuario_awk/data
$ cd curso_usuario_awk/data
$ ln -s /tmp/awk/data .
```

AWK

- Fue creado con la idea de procesar archivos de texto plano, con campos



Ejemplo de PFAM y GOs

PF13919.1^ASXH^Asx homology domain^P1-206^E
GO:0000790^cellular_component^nuclear chromatin`
binding^GO:0042974^molecular_function^retinoic
ID=id14;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=misc_R
ID=gene2;Dbxref=GeneID:102466751,HGNC:HGNC:50039,miRBase:MI0022705;Name=MIR6859-1;descrip
- . ID=rna2;Parent=gene2;Dbxref=GeneID:102466751,Genbank:NR_106918.1,HGNC:HGN
ID=id15;Parent=rna2;Dbxref=GeneID:102466751,Genbank:NR_106918.1,HGNC:HGNC:50039,miRBase:M

Estructura Basica

```
awk '/expresion/{instrucciones}' archivo
```

↑
Invocación
al interprete

↑
Expresion que
Determina a que
Registros se aplicarán
Las instrucciones

↑
Instuccion(es) que
Se aplican a cada uno
De los renglones
obtendios

↑
Archivo sobre
el que se trabaja

Referencias a columnas (ejem. con gff)

\$3

f-version 3
f-spec-version 1.21
ocessor NCBI annotwriter
name-build GRCh38.p4
name-build-accession NCBI

Assembly:GCF_000001405.30
1 248956422
ecies http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=9606

00001.11 RefSeq region 1 248956422 +

00001.11	BestRefSeq	gene	11874	14409	.	+	.	ID=gene0;Dbxref=GeneID:100287102, HGNC:HGNC:37102;Name=DDX11L1;description=DEAD/H (A
00001.11	BestRefSeq	transcript	11874	14409	.	+	.	ID=id1;Parent=rna0;Dbxref=GeneID:100287102,Genbank:NR_046018.2,HGNC:HGNC:37102;gbkey=
00001.11	BestRefSeq	exon	11874	12227	.	+	.	ID=id2;Parent=rna0;Dbxref=GeneID:100287102,Genbank:NR_046018.2,HGNC:HGNC:37102;gbkey=
00001.11	BestRefSeq	exon	12613	12721	.	+	.	ID=id3;Parent=rna0;Dbxref=GeneID:100287102,Genbank:NR_046018.2,HGNC:HGNC:37102;gbkey=
00001.11	BestRefSeq	exon	13221	14409	.	+	.	ID=gene1;Dbxref=GeneID:653635,HGNC:HGNC:38034;Name=WASH7P;description=WAS protein f
00001.11	BestRefSeq	gene	14362	29370	.	-	.	ID=rna1;Parent=gene1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	transcript	14362	29370	.	-	.	ID=id4;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	29321	29370	.	-	.	ID=id5;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	24738	24891	.	-	.	ID=id6;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	18268	18366	.	-	.	ID=id7;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	17915	18061	.	-	.	ID=id8;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	17606	17742	.	-	.	ID=id9;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	17233	17368	.	-	.	ID=id10;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	16858	17055	.	-	.	ID=id11;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	16607	16765	.	-	.	ID=id12;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	15796	15947	.	-	.	ID=id13;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	14970	15038	.	-	.	ID=id14;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGNC:HGNC:38034;gbkey=m
00001.11	BestRefSeq	exon	14362	14829	.	-	.	ID=gene2;Dbxref=GeneID:102466751,HGNC:HGNC:50039,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	gene	17369	17436	.	-	.	- . ID=rna2;Parent=gene2;Dbxref=GeneID:102466751,Genbank:NR_106918.1,HGNC:HGNC:50039,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	primary_transcript	17369	17436	.	-	.	ID=id15;Parent=rna2;Dbxref=GeneID:102466751,Genbank:NR_106918.1,HGNC:HGNC:50039,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	exon	17369	17436	.	-	.	ID=rna3;Parent=gen2;Dbxref=GeneID:102466751,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	ncRNA	17369	17391	.	-	.	ID=id16;Parent=rna3;Dbxref=GeneID:102466751,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	exon	17369	17391	.	-	.	ID=rna4;Parent=gen2;Dbxref=GeneID:102466751,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	ncRNA	17409	17431	.	-	.	ID=id17;Parent=rna4;Dbxref=GeneID:102466751,miRBase:MI0022705;Name=MIR6859-1;d
00001.11	BestRefSeq	exon	17409	17431	.	-	.	ID=gene3;Dbxref=GeneID:100302278,HGNC:HGNC:35294,miRBase:MI006363;Name=MIR1302-2;d
00001.11	BestRefSeq	gene	30366	30503	.	+	.	+ . ID=rna5;Parent=gen3;Dbxref=GeneID:100302278,Genbank:NR_036051.1,HGNC:HGNC:35294,miRBase:MI006363;Name=MIR1302-2;d
00001.11	BestRefSeq	primary_transcript	30366	30503	.	+	.	ID=id18;Parent=rna5;Dbxref=GeneID:100302278,Genbank:NR_036051.1,HGNC:HGNC:35294,miRBase:MI006363;Name=MIR1302-2;d
00001.11	BestRefSeq	exon	30366	30503	.	+	.	ID=rna6;Parent=gen3;Dbxref=GeneID:100302278,miRBase:MI006363;Name=MIR1302-2;d
00001.11	BestRefSeq	ncRNA	30438	30458	.	+	.	ID=id19;Parent=rna6;Dbxref=GeneID:100302278,miRBase:MI006363;Name=MIR1302-2;d
00001.11	BestRefSeq	exon	30438	30458	.	+	.	ID=gene4;Dbxref=GeneID:645520,HGNC:HGNC:32334;Name=FAM138A;description=family with
00001.11	BestRefSeq	gene	34611	36081	.	-	.	ID=rna7;Parent=gen4;Dbxref=GeneID:645520,Genbank:NR_026818.1,HGNC:HGNC:32334;Name=
00001.11	BestRefSeq	ncRNA	34611	36081	.	-	.	ID=id20;Parent=rna7;Dbxref=GeneID:645520,Genbank:NR_026818.1,HGNC:HGNC:32334;gbkey=
00001.11	BestRefSeq	exon	35721	36081	.	-	.	ID=id21;Parent=rna7;Dbxref=GeneID:645520,Genbank:NR_026818.1,HGNC:HGNC:32334;gbkey=
00001.11	BestRefSeq	exon	35277	35481	.	-	.	ID=id22;Parent=rna7;Dbxref=GeneID:645520,Genbank:NR_026818.1,HGNC:HGNC:32334;gbkey=
00001.11	BestRefSeq	exon	34611	35174	.	-	.	ID=gen5;Dbxref=GeneID:105379212;Name=LOC105379212;gbkey=Gene;gene=LOC105379212;gene_biotyp
00001.11	Gnomon gene	51943	53959	.	+	.	ID=rna8;Parent=gen5;Dbxref=GeneID:105379212,Genbank:XR_948874.1;Name=XR_948874.1;gbkey=ncRN	
00001.11	Gnomon ncRNA	51943	53959	.	+	.	ID=id23;Parent=rna8;Dbxref=GeneID:105379212,Genbank:XR_948874.1;gbkey=ncRNA;gene=LOC1053792	
00001.11	Gnomon exon	51943	51994	.	+	.	ID=id24;Parent=rna8;Dbxref=GeneID:105379212,Genbank:XR_948874.1;gbkey=ncRNA;gene=LOC1053792	
00001.11	Gnomon exon	53282	53959	.	+	.	ID=gen6;Dbxref=GeneID:79504,HGNC:HGNC:14822;Name=OR44P;description=olfactory rece	
00001.11	Curated Genomic gene	52453	53396	.	+	.	ID=gen7;Dbxref=GeneID:403263,HGNC:HGNC:31276;Name=OR4G11P;description=olfactory re	
00001.11	Curated Genomic gene	63016	63885	.	+	.	ID=gen8;Dbxref=GeneID:79501,HGNC:HGNC:14825;PRD:14974;Name=OR4F5;description=olfac	
00001.11	BestRefSeq	gene	69091	70008	.	+	.	ID=rna9;Parent=gen8;Dbxref=GeneID:79501,Genbank:NM_001005484.1,HGNC:HGNC:14825,HPRI
00001.11	BestRefSeq	mRNA	69091	70008	.	+	.	

Tomemos solo los registros de genes

- Todo el registro:

```
$ awk '/gene/' data/TAIR10_genomic.gff
```

- Solo la columna 9:

```
$ awk '/gene/{print $9}' data/TAIR10_genomic.gff
```

- Solo la primera sección de la columna 9:

```
$ awk '/gene/{split($9,item,";");print item[0]}' data/TAIR10_genomic.gff
```

```
ID=gene0;Dbxref=GeneID:100287102,HGNC:HGNC:37102;Name=DDX11L1;d  
. ID=rna0;Parent=gene0;Dbxref=GeneID:100287102,Genbank:NR  
ID=id1;Parent=rna0;Dbxref=GeneID:100287102,Genbank:NR_046018.2,  
ID=id2;Parent=rna0;Dbxref=GeneID:100287102,Genbank:NR_046018.2,  
ID=id3;Parent=rna0;Dbxref=GeneID:100287102,Genbank:NR_046018.2,  
ID= gene1;Dbxref=GeneID:653635,HGNC:HGNC:38034;Name=WASH7P;descr  
. ID=rna1;Parent= gene1;Dbxref=GeneID:653635,Genbank:NR_02  
ID=id4;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id5;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id6;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id7;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id8;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id9;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id10;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id11;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id12;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id13;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID=id14;Parent=rna1;Dbxref=GeneID:653635,Genbank:NR_024540.1,HGN  
ID= gene2;Dbxref=GeneID:102466751,HGNC:HGNC:50039,mirBase:MI0022  
-. ID=rna2;Parent= gene2;Dbxref=GeneID:102466751,Ge  
ID=id15;Parent=rna2;Dbxref=GeneID:102466751,Genbank:NR_106918.1  
ID=rna3;Parent= gene2;Dbxref=GeneID:102466751,mirBase:MIMAT00276  
ID=id16;Parent=rna3;Dbxref=GeneID:102466751,mirBase:MIMAT002761  
ID=rna4;Parent= gene2;Dbxref=GeneID:102466751,mirBase:MIMAT00276  
ID=id17;Parent=rna4;Dbxref=GeneID:102466751,mirBase:MIMAT002761  
ID= gene3;Dbxref=GeneID:100302278,HGNC:HGNC:35294,mirBase:MI0006  
+. ID=rna5;Parent= gene3;Dbxref=GeneID:100302278,Ge  
ID=id18;Parent=rna5;Dbxref=GeneID:100302278,Genbank:NR_036051.1  
ID=rna6;Parent= gene3;Dbxref=GeneID:100302278,mirBase:MIMAT00058  
ID=id19;Parent=rna6;Dbxref=GeneID:100302278,mirBase:MIMAT000589  
ID= gene4;Dbxref=GeneID:645520,HGNC:HGNC:32334;Name=FAM138A;desc  
ID=rna7;Parent= gene4;Dbxref=GeneID:645520,Genbank:NR_026818.1,H  
ID=id20;Parent=rna7;Dbxref=GeneID:645520,Genbank:NR_026818.1,HG  
ID=id21;Parent=rna7;Dbxref=GeneID:645520,Genbank:NR_026818.1,HG  
ID=id22;Parent=rna7;Dbxref=GeneID:645520,Genbank:NR_026818.1,HG  
5;Dbxref=GeneID:105379212;Name=LOC105379212;gbkey=Gene;gene=LOC1  
;Parent= gene5;Dbxref=GeneID:105379212,Genbank:XR_948874.1;Name=X  
;Parent= rna8;Dbxref=GeneID:105379212,Genbank:XR_948874.1;gbkey=r  
;Parent= rna8;Dbxref=GeneID:105379212,Genbank:XR_948874.1;gbkey=r  
ID= gene6;Dbxref=GeneID:79504,HGNC:HGNC:14822;Name=OR4G4P;desc  
ID= gene7;Dbxref=GeneID:403263,HGNC:HGNC:31276;Name=OR4G11P;desc  
ID= gene8;Dbxref=GeneID:79501,HGNC:HGNC:14825,HPRD:14974;Name=OR  
ID= rna9;Parent= gene8;Dbxref=GeneID:79501,Genbank:NM_001005484.1
```

Referencia a renglones (ejem trinotate)

#gene_id	transcript_id	sprot_Top_BLASTX_hit	prot_coords	Pfam	gene_ontology_blast
c1_g1	c1_g1_i1	sp P07314 GGT1_RAT^P07314^Q:211-2,H:148-215^40%ID^E:8e-08^RecName: Full=Gamma-glutamyltranspeptidase 1;^Eukaryota; Metazoa; Chordata; Craniata	.	.	GO:0005615^cellular_component^extracellular space`GO:0005887^cellular_component^integral component of plasma membrane`GO:0005886^cellular_compon
e activity`GO:0007568^biological_process^aging`GO:0034599^biological_process^cellular response to oxidative stress`GO:0006536^biological_process^glutamate metabolic p	31179^biological_process^peptide modification`GO:0032355^biological_process^response to estradiol`GO:0032496^biological_process^response to lipopolysaccharide`GO:0034
c9_g1	c9_g1_i1
c11_g1	c11_g1_i1	sp 09HDC9 APMAP_HUMAN^Q9HDC9^Q:252-16,H:246-324^46.84%ID^E:1e-15^RecName: Full=Adipocyte plasma membrane-associated protein;^Eukaryota; Metazo	mo .	GO:0009986^cellular_component^cell surface`GO:0016021^cellular_component^integral component of membrane`GO:0016020^cellular_component^membrane`GO:00	
cess^biosynthetic process
c11_g2	c11_g2_i1	sp Q803F5 APMAP_DANRE^Q803F5^Q:268-5,H:158-245^43.18%ID^E:8e-17^RecName: Full=Adipocyte plasma membrane-associated protein;^Eukaryota; Metazo	oa	GO:0016021^cellular_component^integral component of membrane`GO:0016844^molecular_function^strictosidine synthase activity`GO:0009058^biological_p	
c13_g1	c13_g1_i1
c16_g1	c16_g1_i1
c17_g1	c17_g1_i1
c24_g1	c24_g1_i1
c32_g1	c32_g1_i1
c33_g1	c33_g1_i1
c34_g1	c34_g1_i1	sp Q8IXJ9 ASXL1_HUMAN^Q8IXJ9^Q:567-1,H:164-406^55.33%ID^E:2e-51^RecName: Full=Putative Polycomb group protein ASXL1;^Eukaryota; Metazoa; Chord	[-]	PF13919.1^ASXH^Asx homology domain^81-206^E:1.9e-47	GO:0000790^cellular_component^nuclear chromatin`GO:0035517^cellular_component^PR-DUB complex`GO:000
or activated receptor binding`GO:0005515^molecular_function^protein binding`GO:0042974^molecular_function^retinoic acid receptor binding`GO:0003713^molecular_function	development`GO:0035522^biological_process^monoubiquitinated histone H2A deubiquitination`GO:0045599^biological_process^negative regulation of fat cell differentiat				
al_process^negative regulation of retinoic acid receptor signaling pathway`GO:0000122^biological_process^negative regulation of transcription from RNA polymerase II p	itive regulation of transcription from RNA polymerase II promoter`GO:0032526^biological_process^response to retinoic acid`GO:0006351^biological_process^transcription,
c36_g1	c36_g1_i1
c37_g1	c37_g1_i1
c38_g1	c38_g1_i1	sp Q5R946 PIGP_PONAB^Q5R946^Q:643-242,H:1-134^63.43%ID^E:1e-54^RecName: Full=Phosphatidylinositol N-acetylglucosaminyltransferase subunit P;^E	i; Hominidae; Pongo	254-643[-] PF08510.7^PIG-P^PIG-P^10-124^E:1e-39	GO:0016021^cellular_component^integral component of membrane`GO:0017176^molecular_f
c39_g1	c39_g1_i1
c40_g1	c40_g1_i1
c40_g2	c40_g2_i1
c41_g1	c41_g1_i1	sp Q9NR61 DLL4_HUMAN^Q9NR61^Q:228-1,H:195-270^63.16%ID^E:9e-28^RecName: Full=Delta-like protein 4;^Eukaryota; Metazoa; Chordata; Craniata; Ver	16021^cellular_component^integral component of membrane`GO:0005886^cellular_component^plasma membrane`GO:0005509^molecular_function^calcium ion binding`GO:0005112^mol		
ogical_process^blood vessel lumenization`GO:0001974^biological_process^blood vessel remodeling`GO:0003209^biological_process^cardiac atrium morphogenesis`GO:0003208	`GO:0035924^biological_process^cellular response to vascular endothelial growth factor stimulus`GO:0035912^biological_process^dorsal aorta morphogenesis`GO:0090051^bi				
tion of cell proliferation`GO:0010596^biological_process^negative regulation of endothelial cell migration`GO:0000122^biological_process^negative regulation of trans	naling involved in heart development`GO:0007219^biological_process^Notch signaling pathway`GO:0001569^biological_process^patterning of blood vessels`GO:0003344^biolog				
074^biological_process^regulation of neural retina development`GO:0050767^biological_process^regulation of neurogenesis`GO:0007165^biological_process^signal transduct	yocardium morphogenesis
c43_g1	c43_g1_i1	sp Q9N0C7 EPDR1_MACFA^Q9N0C7^Q:517-2,H:37-208^65.7%ID^E:2e-81^RecName: Full=Mammalian ependymin-related protein 1;^Eukaryota; Metazoa; Chord	at	ithecinae; Macaca 2-520[-] PF00811.13^Ependymin^Ependymin^46-173^E:5.5e-36	GO:0005576^cellular_component^extracellular region`GO:0005509^molecular_function^ca
c45_g1	c45_g1_i1	sp Q86L99 GACHH_DICDI^Q86L99^Q:18-260,H:349-431^34.94%ID^E:2e-07^RecName: Full=Rho GTPase-activating protein gacHH;^Eukaryota; Amoebozoa; Myce	activator activity`GO:0007165^biological_process^signal transduction	.	.
c46_g1	c46_g1_i1	sp Q9QXZ0 MACF1_MOUSE^Q9QXZ0^Q:252-1,H:93-176^94.05%ID^E:9e-46^RecName: Full=Microtubule-actin cross-linking factor 1;^Eukaryota; Metazoa; Cho	rinae; Mus; Mus	. GO:0015629^cellular_component^actin cytoskeleton`GO:0005737^cellular_component^cytoplasm`GO:0005794^cellular_component^Golgi appara	
eton`GO:0032587^cellular_component^ruffle membrane`GO:0003779^molecular_function^actin binding`GO:0016887^molecular_function^ATPase activity`GO:0005509^molecular_func	logical_process^ATP catabolic process`GO:0007050^biological_process^cell cycle arrest`GO:0006928^biological_process^cellular component movement`GO:0007163^biological_				
07^biological_process^mesoderm formation`GO:0030177^biological_process^positive regulation of Wnt signaling pathway`GO:0006620^biological_process^posttranslational pr	ession of focal adhesion assembly`GO:0032886^biological_process^regulation of microtubule-based process`GO:0016055^biological_process^Wnt signaling pathway`GO:0042060^bi				
c46_g1	c46_g1_i2	sp Q91ZU6 DYST_MOUSE^Q91ZU6^Q:402-1,H:1-133^82.35%ID^E:7e-69^RecName: Full=Dystonin;^Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Eute	llul	PF00307.26^CH^Calponin homology (CH) domain^39-134^E:1.2e-17	GO:0015629^cellular_component^actin cytoskeleton`GO:0030424^cellular_component^axon`GO:0009925
llular_component^cell leading edge`GO:0005737^cellular_component^cytoplasm`GO:0016023^cellular_component^cytoplasmic membrane-bound vesicle`GO:0005829^cel	0031252^cellular_component^hemidesmosome`GO:0016021^cellular_component^integral component of membrane`GO:0005882^cellular_component^intermediate filament`GO:0045111^cellular_c				
bule plus-end`GO:0060053^cellular_component^neurofilament cytoskeleton`GO:0005635^cellular_component^nuclear envelope`GO:0005634^cellular_component^nucleus`GO:0030018).

Para cada patrón, una acción

- /patron/{accion}
- /patron/{accion}
- ...
- /patron/{accion}

Ejecución de un awk

- Hay varias maneras de correr un programa awk. Si el programa es corto, lo más fácil es incluirlo en el comando que corre awk:

```
$ awk '{instrucciones}' archivo1 archivo2
```

- Cuando el programa es largo, es más conveniente ponerlo en un archivo y ejecutarlo con un comando como este:

```
$ awk -f programa-file.awk input-file1 input-file2
```

Como ejecutar awk

- Por default imprime la línea que cumple con el patrón:

```
$ awk '/bar/' data/Expresiones1.txt
```

- Cuando no se especifica un patrón, la acción se realiza sobre todas las líneas del archivo:

```
$ awk '{print "Hola Mundo"}' data/Expresiones1.txt
Hola Mundo
```

Variables

- Las variables sirven para almacenar valores en un punto del programa para su uso posterior en otra parte de su programa.
- Las variables pueden ser manipuladas por completo dentro del texto del programa
- También se les pueden asignar valores desde la línea de comando awk.

A	matches
8	1

Variables

- El nombre de las variables debe ser una secuencia de letras, números o guiones bajos.
- NO debe empezar con un número
- Las letras deben ser cualesquiera de las 23 del abecedario inglés.
- Es sensible a mayúsculas y minúsculas: entonces A es diferente de *a*
- No se aceptan espacios en blanco...el nombre de la variable termina con un espacio.

- Edad Nombres variables validos 
- Peso** 
- 5datos 
- Valor absoluto 
- Valor_Nomial 
- Estancia#temporal 
- Nombre 
- día 
- X18 

Variables vs constantes

- Las constantes de carácter se delimitan con las comillas:

```
awk '{print "Hola Mundo"}' Expresiones1.txt
```

- Si eliminamos las comillas considera que Hola y Mundo son variables vacías...

```
awk '{print Hola Mundo}' Expresiones1.txt
```

Como ejecutar awk

- Pero hay que prestar atención a que las comillas son los delimitadores:

```
awk '{print "Don't worry"}' data/Expresiones1.txt
```

>



- ¿Como podemos solucionarlo?:

```
awk '{print "Don`t worry"}' data/Expresiones1.txt
```

```
awk '{print "Don\'t worry"}' data/Expresiones1.txt
```

LECTURA Y ESCRITURA DE ARCHIVOS EN AWK

Temas:

- Lectura de Archivos
 - Los registros
 - Los campos
 - Cambiando el valor de un campo
 - Los separadores
- Salida
 - print
 - Separador en la salida
 - printf
 - Redireccionado la salida

Objetivo

- Que el participante identifique entradas y salidas en AWK
- Que el participante pueda manipular flujos de entrada y salida

Entradas

- En un programa awk típico, se leen todas las entradas desde la entrada estándar (el teclado, pero a menudo se trata de una tubería desde otro comando) o desde archivos cuyos nombres se especifica en la línea de comando awk.

```
awk '/condición/{instrucciones}' file.txt
```

- Si especifica varios archivos de entrada, awk los lee en orden, procesando todos los datos de un archivo antes de pasar al siguiente.
- El nombre del archivo de entrada se puede encontrar en la variable predefinida FILENAME

Los registros...

- awk divide el archivo de entrada en registros y campos.

DS1306

Figure 2. RTC REGISTERS AND ADDRESS MAP

HEX ADDRESS		Bit 7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	RANGE			
READ	WRITE	00H	80H	0	10 SEC				SEC	00-59			
		01H	81H	0	10 MIN				MIN	00-59			
02H	82H	0	12	A	10-HR	HOURS				01-12 + P/A			
			24	10	0	0	DAY			01-07			
DATE					DATE	1-31							
MONTH					MONTH	01-12							
					YEAR	00-99							
ARM 0					SEC ALARM 0	00-59							
ARM 0					MIN ALARM 0	00-59							
10-HR					HOUR ALARM 0	01-12 + P/A							
						00-23							
0	0				DAY ALARM 0	01-07							
ARM 1					SEC ALARM 1	00-59							
ARM 1					MIN ALARM 1	00-59							
10-HR					HOUR ALARM 1	01-12 + P/A							
						00-23							
0	0				DAY ALARM 1	01-07							
—					—	—							
CONTROL REGISTER					—	—							
STATUS REGISTER					—	—							
ACCELEROMETER CHARGER REGISTER					—	—							
RESERVED					—	—							
96-BYTES USER RAM					35	—							

Figure 3. Income Data Across Cities

Fields: City, Tab, City, Income, PPP, Adjusted Net Yearly Income (\$), Ranking (Highest to Lowest PPP Adjusted Wage)

Registers: \$0, \$1, \$2, \$3

Separador

Variables predefinidas FNR, NR

- **FNR** almacena el número de renglones que se han leído y se reinicia a cero cuando leemos un nuevo archivo.
- **NR** Lleva la cuenta de cuantos registros han sido procesados (a lo largo de varios archivos inclusive)
- **NF** indica el número de campos por registro, y \$NR muestra el contenido del último campo
- **FILENAME** Nombre del archivo que esta procesando.

Ejemplo

¿Que hace el siguiente segmento?

```
$ awk '{print FILENAME, $1, FNR, NR,NF}' data/mail-list \
data/inventory-shipped
```

Ejercicios

El archivo

homo_sapiens.OnlyTranscript.gff.gz

- ¿tiene formato de GFF (es decir, 9 columnas separadas por un tabulador)?
- ¿qué contiene la última columna?
- ¿Qué gen esta descrito en el registro 74815?

Constantes de Expresiones Regulares

- Es importante tener en cuenta que hacer una asignación a un campo existente cambia el valor de \$ 0, pero no cambia el valor de NF, incluso cuando se asigna la cadena vacía a un campo.
- El campo sigue ahí; sólo que tiene un valor vacío, delimitado por el signo ":" entre "a" y c'. Este ejemplo muestra lo que sucede si se crea un nuevo campo:

```
$ echo a b c d | awk '{ OFS = ":"; $2 = ""; $6 = "new"  
>   print $0; print NF }'  
a::c::d::new  
6
```

Imprimiendo Salidas

- Una de las acciones más comunes en awk es imprimir.
- *print* produce una salida con formato simple, a la salida estándar.
- Los elementos son impresos separados por espacios individuales (o lo que este definido como Separador de Salida (OFS), seguido de una nueva línea
- Itemx puede ser una constante numérica ó de cadena, una variable, un campo \$1, \$2...

```
print item1, item2, item3 ← Esto imprime OFS entre cada campo  
print item1 item2 item3 ← Esto concatena item1 con item2 con item3  
print ← Esto imprime todo el registro $0
```

separadores

- ¿Cuál es la diferencia?:

```
$ awk {print "forma1:",$1, $2, $3;  
print "forma2:" $1 $2 $3  
print  
} data/grades.txt
```

awk –f imprime.awk data/grades.txt

```
$ echo "uno dos tres" |awk '{ print "line $1\nline $2\nline $3" } '
```

```
$ echo "uno dos tres" |awk '{ print "line \"$1\"\nline \"$2\"\nline \"$3\" }'
```

Expresiones Booleanas

- Una expresión booleana o lógica es una expresión que puede ser evaluado a verdadero o falso .
- En awk, lo verdadero evalúa a 1 y lo falso evalúa a 0
 - a) $8 < 7$
 - b) $5 == 5$
 - c) "Nuevo Amanecer" ~ /Nuevo/

Operadores de Comparación

$x < y$	Verdad si x es menor que y
$x \leq y$	Verdad si x es menor o igual que y
$x > y$	Verdad si x es mayor que y
$x \geq y$	Verdad si x es mayor o igual que y
$x == y$	Verdad si x es igual que y
$x != y$	Verdad si x no es igual que y
$x \sim y$	Verdad si la cadena x “hace match” con la expresión regular indicada en y
$x !\sim y$	Verdad si la cadena x “NO hace match” con la expresión regular indicada en y

Operador	Descripción
$x ^ y$	X elevado al exponente y
$x ** y$	
$-x$	Negación
$+x$	Adición Unaria
$x * y$	Multiplicación
x / y	División
$x \% y$	Modulo o resto
$x + y$	Adición
$x - y$	substracción



¿A cuánto evalúa la expresión?

a) $5 + 4 * 2 - 10 / 2$

¿4? u ¿8?

A = 0 ; B = 1 ; C= 2

b) $B < C \quad || \quad B < A \quad \&\& \quad A \neq C$

c) $B < C \quad || \quad (B < A \quad \&\& \quad A \neq C)$

Prioridad operadores

- (. . .) Paréntesis, decremento.
- $\wedge \text{ } **$ Exponente. De derecha a izquierda.
- + - ! Operador Unario positivo, negativo, negación lógica
- * / % Multiplicación, división, residuo.
- + - Adición, sustracción.

String concatenation No hay un carácter especial “ “

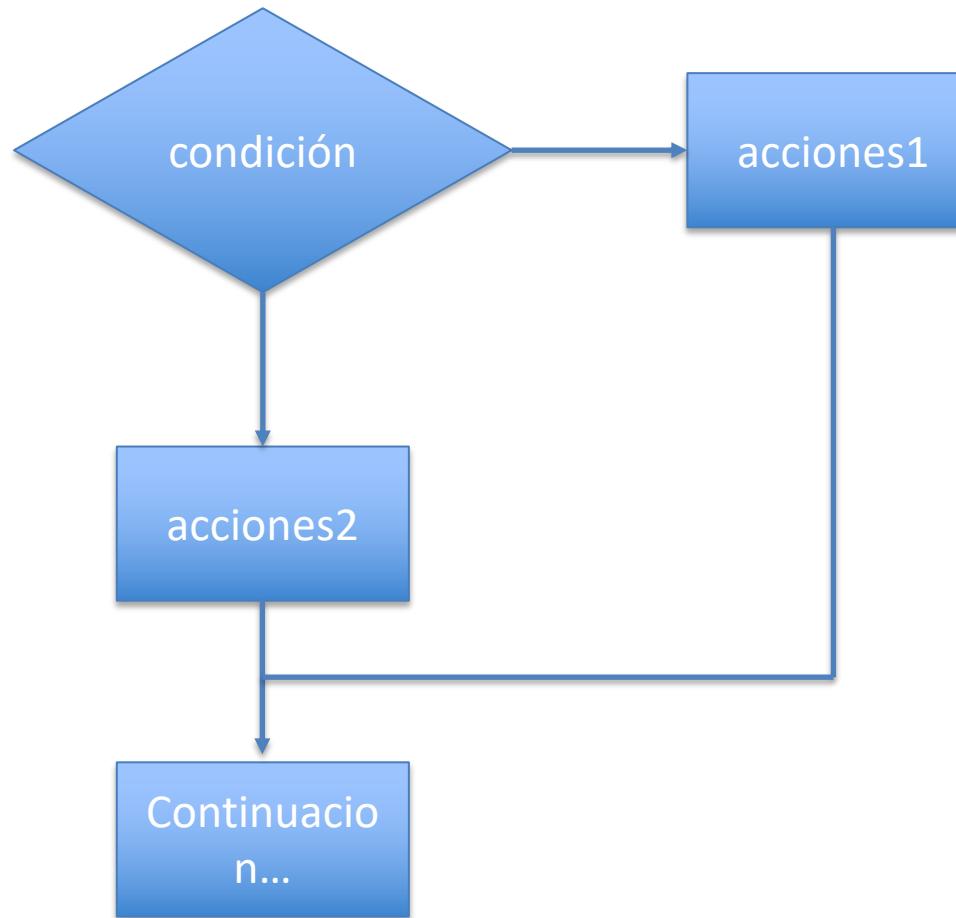
< <= == != > >= >> | |& Relacionales y Lógicos

Ejercicio

- Vuelve a recorrer el archivo grades, pero esta vez imprime el nombre y el promedio de cada alumno. Recuerde que el separador entre los campos es un tabulador. (\t)

condicionales

- **If** (condición){ acciones1} **else** {acciones2}



Condicionales

- Imagina que ademas de la calificación final o promedio, debes especificar si un alumno está aprobado ($\text{prom} \geq 70$) o reprobado ($\text{prom} < 70$) al calcular su promedio.

```
if( prom >= 70 ){
    print "Aprobado"
else{
    print "Reprobado"
}
```

El inicio y el final de un AWK

- Podemos definir variables antes del procesamiento de todos los renglones o imprimir estadísticas finales, usando las secciones BEGIN y END

```
awk 'BEGIN{...} /patron/{accion} END{...}'
```

ejemplo

- Nuevamente usando el archivo de grades, agrega a tu script de awk el cálculo del promedio general del grupo

```
BEGIN{prom=0; promGen=0}
{ prom=( $2 + $3 + $4)/3;
  print $1, "promedio :", prom;
  promGen=promGen+prom
}
END {print "Promedio General", (promGen/NR)}
```

Re direccionando la salida

- print usualmente mandan la impresión a la salida estándar : el monitor
- Pero podemos “redirigir” su salida a otros lugares. A eso llamamos “re direccionar la salida”

Redireccionando salida

formato	Descripción	Ejemplo
print items > output-file	Redirecciona la salida al archivo output-file	awk '{ print \$2 > "phone-list" print \$1 > "name-list" }' mail-list
print items >> output-file	Redirecciona la salida al archivo pre-existente output-file	awk '{ print \$2 >> "phone-list" print \$1 >> "name-list" }' mail-list
print items command	Envía la salida a otro comando	awk '{ print \$1 > "names.unsorted" command = "sort -r > names.sorted" print \$1 command }' mail-list

Ejercicios

1. Modifica el programa para calcular los promedios, para que esta vez genere un archivo con los alumnos aprobados y otro con los alumnos reprobados.

Constantes de exp reg

- Cuando una constante de expresión regular aparece sola, o a mano izquierda del operador, se interpreta como si estuvieran antecedida por : \$0 ~
- Una bizarra consecuencia de esto, es que la siguiente expresión booleana es valida, pero no hará los que su autor probablemente intenta:

```
$ awk '/con/ ~ $1 { print }' data/Expresiones1.txt
```

Operadores Lógicos

Operador And:

- Expr1 && Expr2: Solo es verdadero cuando ambos son verdaderos.

```
$0 ~ /edu/ && $0 ~ /li/ { print }
```

Operador Or:

- Exp1 || Exp2: Solo es falso cuando ambos son falsos
- !Exp1 : Es verdadero cuando Exp2 es falso y falso cuando Exp1 es verdadero

Constantes de Expresiones Regulares

- Una constante de expresión regular es una descripción de una expresión regular encerrada entre barras inclinadas, a mano derecha de lo que desea consultar:

/^beginning and end\$/

\$0 es la variable que refiere a una línea completa

```
awk '$0 ~ /barfly/ || $0 ~ /camelot/ { print }' data/Expresiones1.txt
```

```
awk '/barfly/ || /camelot/ {print}' data/Expresiones1.txt
```

```
awk '/barfly/ && /camelot/ {print}' data/Expresiones1.txt
```

```
awk '/con/ && /palabra/ {print}' data/Expresiones1.txt
```

Ejercicios

1. A partir del archivo data/Expresiones1.txt, genere un programa awk para imprimir exclusivamente las líneas que inician con la palabra “que” e imprima:

/ / “ Esta línea inicia con que”, \$0

Si la línea terminan con la letra “a”, imprime:

“ Esta línea termina con la letra a:”, \$0

Built-in Variables

variables predefinidas en awk

VARIABLE	DESCRIPCION	EJEMPLO/default
FS	File Separator = carácter de separación de los campos. Se puede establecer el valor de FS en la línea de comandos mediante la opción -F: awk -F, "programa" input-files	\t " "
OFS	Output File Separator= carácter que separa los campos de salida	" " (espacio en blanco)
ORS	Output Record Separator= Separador de salida de los registros	\n (nueva línea)
RS	Record Separator= carácter de separación de los registros de entrada	\n (salto de línea) \t (tabulador)
FILENAME	Nombre del archivo de entrada En la sección de BEGIN siempre es " "	Ejemplo.txt - " "
FNR	The current Record number	1

Built-in Variables

VARIABLE	DESCRIPCION	EJEMPLO/default
NF	Number Field: El número de campos en el registro de entrada actual. NF se establece cada vez que se lee un nuevo registro, cuando se crea un nuevo campo, o cuando \$ 0 cambia	16
NR	Number Row: awk incrementa NR cada vez que se lee un nuevo registro.	100
ARGV	Los argumentos de línea de comandos disponibles para los programas awk se almacenan en un arreglo llamado ARGV.	<pre>awk 'BEGIN { for (i = 0; i < ARGC; i++) print ARGV[i] }' inventory-shipped mail-list</pre>
ARGC	es el número de argumentos de línea de comandos presentes.	3

Práctica 1.

Trabajemos con el archivo Expresiones1.txt como archivo de entrada:

1. Genera un awk que imprime el ultimo campo por línea, en el archivo Expresiones1.txt, y luego repítelo pero considerando que el separador de los campos es un tabulador (-F\t)
2. Imprime el primer campo del archivo Expresiones, antecedido por el número de renglón.
3. Imprime el nombre del mes, con la 5ta columna dividida entre la 4ta columna. (Evita los renglones vacíos)
4. Imprime, los renglones pares del archivo mail-list

Práctica 1. Respuesta

El contenido de último campo por línea:

```
$ awk '{print $NF}' data/Expresiones1.txt
```

ejemplo

archivo

línea

palabra

barfly

palabra

camelot

Imprimir el primer campo, antecedido por el número de renglón:

```
$ awk '{print NR,$1}' data/Expresiones1.txt
```

1 Este

2 de

3 que

4 con

5 barfly

6 o

7 camelot

Práctica 1. Respuesta

1. imprime el nombre del mes, seguido del resultado de Dividir
la columna 5 entre la columna 4,
el archivo Inventory-shipped. Evite los renglones vacíos

```
awk 'NF>0{divide= $5/$4; print $1,divide}' data/Inventory-shipped
```

Jan 7.66667

Feb 9.41667

...

Feb 8.15

Mar 7.07143

Apr 6.94595

#2. Imprime solo los registros pares del archivo mail-list:

```
awk 'NR % 2 == 0' data/mail-list
```

Anthony 555-3412 anthony.asserturo@hotmail.com A

Bill 555-1675 bill.drowning@hotmail.com A

Camilla 555-2912 camilla.infusarum@skynet.be R

Julie 555-6699 julie.perscrutabor@skeeve.com F

Samuel 555-3430 samuel.lanceolis@shu.edu A

Práctica 2

1. El archivo bad.fasta, no inicia cada nombre de secuencia con el simbolo ">". Lo que sabemos de cierto es que todos los renglones impares son nombres de secuencias y los pares, sus secuencias correspondientes, agrega el signo ">" y guarda la salida en un nuevo archivo good.fasta
2. Un archivo fastq, tiene 4 renglones por secuencia:
 - a. El nombre de la secuencia
 - b. La secuencia propiamente
 - c. Otra vez el nombre de la secuencia
 - d. Calidad de la secuencia.A partir de este archivo, genera un archivo en formato fasta

Práctica2 . (respuestas)

2. El archivo bad.fasta, no inicia cada nombre de secuencia con el simbolo “>” . Lo que sabemos de cierto es que todos los renglones impares son nombres de secuencias y los pares, sus secuencias correspondientes, agrega el signo “>” y guarda la salida en un nuevo archivo good.fasta

```
awk 'NR % 2 == 1{print ">" $0} NR % 2 == 0 {print }' data/bad.fasta > good.fasta
```

3. Un archivo fastq, tiene 4 renglones por secuencia:

- a. El nombre de la secuencia
- b. La secuencia propiamente
- c. Otra vez el nombre de la secuencia
- d. Calidad de la secuencia.

A partir de este archivo, genera un archivo en formato fasta

```
awk 'NR % 4 == 1{sub("@",">"); print } NR % 4 == 2 {print }'  
data/Illumina.Single.fastq > salida.fasta
```

Concepto de Arreglo

- Un arreglo es una tabla de valores llamados elementos.
- Los elementos de un arreglo se distinguen por sus índices.
- Los índices pueden ser números o cadenas.

Index	Value
1	"foo"
2	""
3	30

Arreglos asociativos

- En awk los arreglos son asociativos, esto es, un arreglo es una colección de pares: índice – valor:

Index Value

3 30

1 "foo"

0 8

2 ""

- Puede haber indices perdidos

- Los indices no necesitan ser valores positivos...ni siquiera numeros...

Index Value

"dog" "chien"

"cat" "chat"

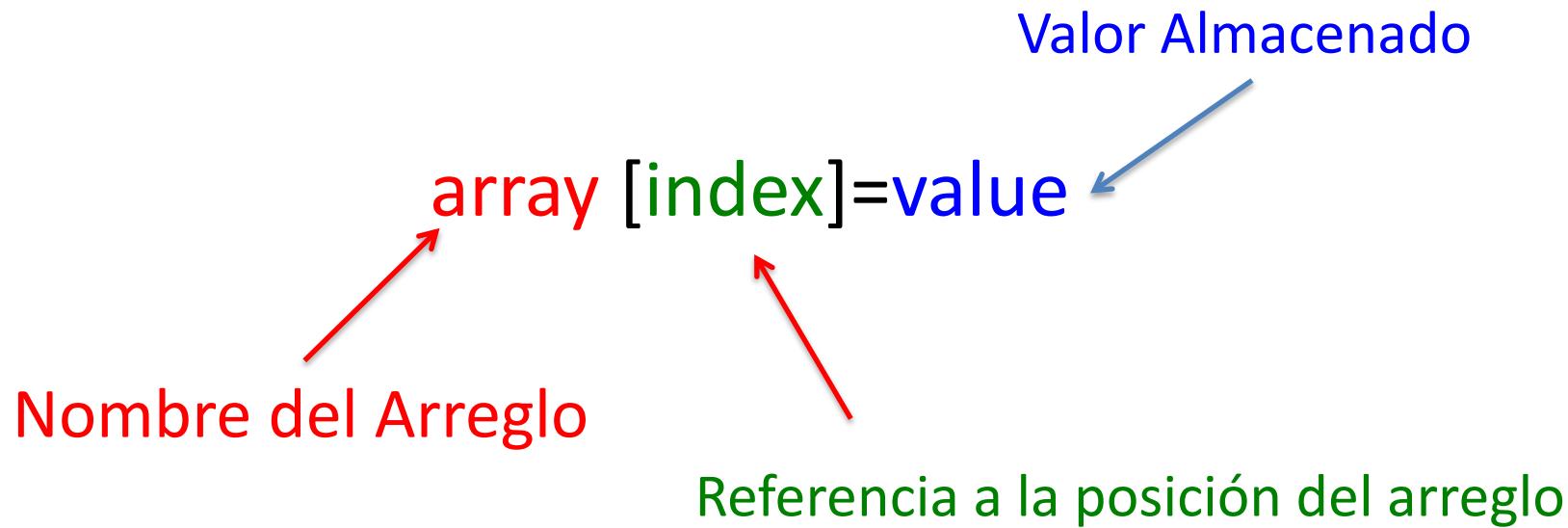
"one" "un"

1 "un"

Refiriéndose a un elemento

La principal forma de utilizar una arreglo es referirse a uno de sus elementos.

Una referencia al elemento de un arreglo tiene esta sintaxis:



Asignando valores a un arreglo

- La sintaxis básica es la misma que la asignación en las variables:

array [index]=value

frecuencias[1] =123

Frecuencias[5] =75

Frecuencias[7] =84

...

Frecuencias[10] =92

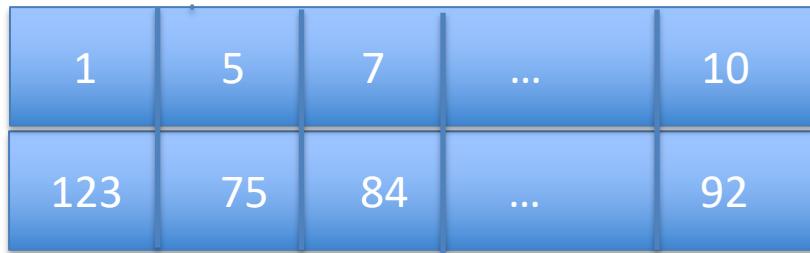
1	5	7	...	10
123	75	84	...	92

- Una referencia a un elemento que no existe crea automáticamente el elemento en el arreglo, con la cadena vacía como su valor. (Esto es desafortunado, ya que podría saturar la memoria o llenarla de basura)
- Los programadores novatos de awk suelen cometer el error de comprobar si existe un elemento comprobando si el valor está vacío:

```
# Checar si "foo" existe in a:      Incorrect!  
if (a["foo"] != "") ...
```

- Para determinar si un elemento existe en un arreglo use la siguiente expresion:
idx in array

Esto da uno cuando existe y cero en otro caso.



```
if (2 in frecuencias)
print "El elemento 2 existe con el valor de " frecuencias[2]
```

Uniendo dos archivos

- Imagine que quiero unir los archivos A y B. Ambos contienen valores de expresión en dos condiciones wt.

```
$ awk 'BEGIN{OFS="\t"} \
NR == FNR{gene[$1] = $2;next;}
NR > FNR {print $1, gene[$1], $2}' \
data/A.txt data/B.txt
```

Práctica 3

1. Cambiar un tabular por ";" en el archivo grades
2. El archivo TAIR10_genomic.gff, tienen más de 484 mil registros. Si solo queremos tomar todos registros que correspondan a la descripción de "gene", y cuyos genes esten ubicados en el genoma en el intervalo de 50,000 a 10000pb. Las posiciones realtivas del la región genómica están en las columnas 4 y 5 Guarda el resultado en un nuevo archivo llamado MyTAIR.gff
3. Para llevar a cabo el análisis de expresión diferencial de un conjunto de genes es muy común tener que unir los archivos de conteos. Unir 2 archivos a partir de la columna definida por el usuario en común. (Conteos1.txt y Conteos2.txt) y pon la salida en un nuevo archivo llamado TotalConteo.txt, separados por comas (,)
4. Los archivos donde se reporta el resultado de la expresion diferencial de un conjunto de genes, contiene todos los valores de expresión calculados. Es muy común querer filtrar un archivo con múltiples columnas que cumpla con varias condiciones. En este caso elige los registros que cumplan $LFC > 2.00$, y $FDR < .010$, e imprime, las dos primeras columnas y la última Usa el archivo DE_result.txt
5. Se obtuvo un archivo con secuencias de proteínas de la base de datos de uniprot_swissprot. Que es una de las bases de datos suiza más consultada a nivel mundial. Y queremos obtener una lista de los identificadores de secuencias que se encuentran en los encabezados (que inician con >) entre los pipes (|) y guardarla en nuevo archivo.

Práctica 3. Respuestas

1. Cambiar un tabular por “;” en el archivo grades

```
$ awk 'BEGIN{OFS=";"}{print $1,$2,$3,$4}' data/grades
```

Práctica 3. Respuestas

2. El archivo TAIR10_genomic.gff, tienen mas de 484 mil registros. Si solo queremos tomar todos registros que correspondan a la descripción de “gene”, y cuyos genes esten ubicados en el genoma en el intervalo de 50,000 a 10000pb. Las posiciones realltivas del la región genómica están en las columnas 4 y 5 Guarde el resultado en un nuevo archivo llamado MyTAIR.gff

```
$ awk '$3 == "gene" && $4>=50000 && $5<=100000{print}' \
data/TAIR10_genomic.gff > myTAIR.gff
```

Práctica 3. Respuestas

3. Para llevar a cabo el análisis de expresión diferencial de un conjunto de genes es muy común tener que unir los archivos de conteos. Unir 2 archivos a partir de la columna definida por nombre del gen. (Conteos1.txt y Conteos2.txt) y pon la salida en un nuevo archivo llamado TotalConteo.txt, separados por comas (,)

```
awk 'BEGIN{ OFS=""}
      NR == FNR{a[$1]=$2;next}
      {print $1,$2,a[$1]} ' \
data/Conteos1.txt data/Conteos2.txt > data/TotalConteos.txt
```

4. Los archivos donde se reporta el resultado de la expresión diferencial de un conjunto de genes, contiene todos los valores de expresión calculados. Es muy común querer filtrar un archivo con múltiples columnas que cumpla con varias condiciones. En este caso elige los registros que cumplan $LFC > 2.00$, y $FDR < .010$, e imprime, las dos primeras columnas y la última Usa el archivo DE_result.txt

```
$ awk '$2 >= 2 && $5<0.01{print $1,$2,$NF}' \
data/DE_results.txt
```

-
5. Se obtuvo un archivo con secuencias de proteínas de la base de datos de uniprot_swissprot. Que es una de las bases de datos suiza más consultada a nivel mundial. Y queremos obtener una lista de los identificadores de secuencias que se encuentran en los encabezados (que inician con >) entre los pipes (|) y guardarla en nuevo archivo.

```
$ awk -F"|" '/>/ {print $2}' \
data/uniprot_sprot.fasta
```

Referencias

GAWK: Effective AWK Programming

A User's Guide for GNU Awk

Edition 4.1

April, 2015

Arnold D. Robbins

Por su atención, ¡ gracias!!

