Nodo Nacional de Bioinformática

Universidad Nacional Autónoma de México – Nodo Mexicano EMBNet

Taller 1. Introducción al biocómputo en Sistemas Linux y su aplicación en filoinformática

Semana 1. Descubriendo el poder del intérprete de comandos (shell)

Profesores:
Romualdo Zayas
Heladia Salgado
George Magklaras

CCG
Centro de Ciencias Genómicas

DIA 3. COMANDOS PARA MANIPULAR EL CONTENIDO DE UN ARCHIVO.

Temario DIA 3

- Comandos para manejar el contenido de un archivo
  - Contando líneas
  - Cortando líneas
  - Ordenando líneas
  - Manejo de líneas repetidos
  - Búsqueda de patrones
- Lenguajes
  - Awk
  - Perl command line
- Práctica 3. Manipulando la información de un archivo

Objetivos

Después de completar esta lección, el alumno será capaz de manipular el contenido de un archivo.

En esta lección usaremos los archivos que se encuentran bajo el directorio de *E_coli*.

- **E_coli_K12_genes.txt** Contiene la lista de genes de *Escherichia coli*, ordenados por posición en el genoma.

- **tf-gene-interactions.txt** Es la red de interacciones de factores de transcripción y sus genes regulados.

- **b????.txt** son las secuencias en formato fasta de ciertos genes de E. coli, el nombre del archivo es el identificador que le han asignado.

- **Gene_sequence.txt**. Es el archivo de secuencias de los genes del genoma de E. coli

- Crear un directorio *data* en tu home

- Copiar todos los directorios y archivos del path *heladia/data/* a tu directorio *data*

Contando datos

Contar líneas, palabras y caracteres usando *wc*

```
$ man wc

$ wc E_coli_K12_genes.txt

$ wc -l E_coli_K12_genes.txt

$ wc -w E_coli_K12_genes.txt

$ wc -m E_coli_K12_genes.txt
```

Cortando columnas

Extraer columnas de datos de un archivo usando *cut*

```
$ man cut

$ cut –f2 E_coli_K12_genes.txt            # campos

$ cut -f2,3,4,5,6 E_coli_K12_genes.txt
$ cut –f4,5,2,3,6 E_coli_K12_genes.txt    # ¿Qué pasa?

$ cut –f2-6 E_coli_K12_genes.txt

$ cut –c1-10 E_coli_K12_genes.txt         # caracteres
```

Ordenando líneas

Ordenar líneas de datos de un archivo usando *sort*

```
$ man sort

$ sort E_coli_K12_genes.txt

$ sort -k2 E_coli_K12_genes.txt       # por nombre del gene
$ sort –k4 E_coli_K12_genes.txt

$ sort –k4 –n E_coli_K12_genes.txt
$ sort –k4 –nr E_coli_K12_genes.txt

$ sort –k4 –n E_coli_K12_genes.txt –o E_coli_K12_genes_sort.txt
$ wc –l E_coli_K12_genes_sort.txt

$ sort –u E_coli_K12_genes.txt –o E_coli_K12_genes_uniq.txt
$ wc –l E_coli_K12_genes_uniq.txt
```

Repeticiones

Reporta u omite líneas repetidas usando *uniq*

```
$ man uniq

$ uniq -c E_coli_K12_genes_sort.txt

$ uniq -d E_coli_K12_genes_sort.txt  # solo las duplicadas

$ uniq -u E_coli_K12_genes_sort.txt  # solo las únicas
```

Búsqueda de patrones en archivos

Buscar líneas que contienen un patrón o cadena usando *grep*

```
$ man grep

$ grep araC E_coli_K12_genes.txt
$ grep ara E_coli_K12_genes.txt

$ grep crp E_coli_K12_genes.txt
$ grep –w crp E_coli_K12_genes.txt

$ grep ‘CTGTATGA’  pc_bn_site_prom__w_ids.txt

$ grep -E 'TACTGTA........CAGT' pc_bn_site_prom__w_ids.txt

$ grep -f lista_genes.txt E_coli_K12_genes.txt

$ grep -v '#' E_coli_K12_genes.txt

$ grep –v -f lista_genes.txt E_coli_K12_genes.txt  # líneas que no
contienen el patrón
```

Escenario 1.

El archivo **E_coli_K12_genes.txt** contiene información sobre los genes de E. coli.

1.- Queremos obtener el total de los genes.

Tips:

- Las líneas de los genes empiezan con ECK12
- Puede haber líneas repetidas
- Las líneas que inician con # son comentarios

2.- Queremos saber el total de genes que van en orientación 5’-3´(forward) y 3´-5´(reverse).

3.- Cómo obtenemos aquellos genes a los que no les anotaron la orientación (forward, reverse).

4.- Con lo que hasta ahora hemos visto, ¿Podríamos obtener el tamaño de los genes?

```
# 1.1

$ grep ECK12 E_coli_K12_genes.txt |sort –u | wc

# 1.2
$ grep ECK12 E_coli_K12_genes.txt | grep forward |wc

$ grep ECK12 E_coli_K12_genes.txt | grep reverse |wc

# 1.3
$ grep ECK12 E_coli_K12_genes.txt | grep -v -E "forward|reverse”

# 1.4
$
```

Escenario 2

El archivo tf-gene-interactions.txt contiene la lista de interacciones entre las proteínas y los genes a los que regulan, donde

- \+ indica que lo activa,
- el – que lo reprime,
- +- que lo activa y reprime en distintas condiciones y
- ? que se desconoce como la proteína afecta al gene

1.- Obtener el total de proteínas únicas que son las que regulan a los genes.

2.- Obtener el total de genes únicos regulados por las proteínas.

3.- Cuantas interacciones +,-,+- y ? Existen ?

4.- Obtener el total de genes regulados por cada proteína.

5.- Obtener el total de proteínas que regulan al gene.

6.- Obtener la distribución de genes regulados por 1,2,3,etc proteínas.

```
# 2.1

$ grep -v "#" tf-gene-interactions.txt  | cut -f1 | sort -u |wc

# 2.2
$ grep -v "#" tf-gene-interactions.txt  | cut -f2 | sort -u |wc

# 2.3
$ grep -v "#" tf-gene-interactions.txt  |cut -f1-3 |sort -u |cut -f3 |
sort | uniq -c | sort -nr

# 2.4
$ grep -v "#" tf-gene-interactions.txt  |cut -f1,2 |sort -u |cut -f1 |
uniq -c | sort -nr |more

# 2.5
$ grep -v "#" tf-gene-interactions.txt  |cut -f1,2 |sort | uniq | cut -f2
| sort |uniq -c | sort -nr |more

# 2.6
$grep -v "#" tf-gene-interactions.txt  |cut -f1,2 |sort | uniq | cut -f2 |
sort |uniq -c | sort -nr |cut -c1-4 |uniq -c |sort -nr
```

Escenario 3

En RegulonDB se han curado/revisado las referencias(papers) asociados al proceso de regulación de la bacteria E. coli disponibles en la base de datos pubmed.

Usando el identificador de las referencias en pubmed, se extrajo información importante tal como el año de la publicación, los autores, el abstract de la referencia, etc… en formato medline. Esta información esta en el archivo PMIDs_RegulonDB_8.5_medline.txt

1.- Queremos obtener la distribución de papers revisados asociados al proceso, por año (se puede usar el campo DP)

2.- Queremos obtener la distribución de papers por journal. Esto nos ayudará a determinar cuales son las revistas de donde más hay información del proceso.

```
# 3.1

$ grep "DP  -" PMIDs_RegulonDB_8.5_medline.txt |cut -c7-10 | sort | uniq –c


# 3.2
$ grep "JT  -" PMIDs_RegulonDB_8.5_medline.txt | sort | uniq -c |sort -nr |
more
```

Sustituciones -- sed

Sed permite sustituir una cadena por otra.

Sintáxis. Hay dos formas de usar sed

- Editando las instrucciones sobre línea de comando

  sed [-e] 'instrucciones' archivo

- Editando las instrucciones en un archivo

  sed –f archivoInstrucciones archivo

sed [-e] ‘instrucciones’ archivo

```
#
$ sed –e 's/forward/D/'  E_coli_K12_genes.txt

$ sed 's/forward/D/'  E_coli_K12_genes.txt



#
$ sed 's/forward/D/; s/reverse/R/;' E_coli_K12_genes.txt

$ sed -e 's/forward/D/' -e ' s/reverse/R/;' E_coli_K12_genes.txt  |more
```

sed –f archivoInstrucciones archivo

```
# sustituye los patrones y despliega todos los registros
$ sed -f sedScriptFile.txt E_coli_K12_genes.txt

$ sed -f sedScriptFile.txt E_coli_K12_genes.txt > output_file.txt


# Elimina las líneas que no fueron alteradas (p print)
$ sed -n -e 's/forward/D/p; s/reverse/R/p;' E_coli_K12_genes.txt  |more


# elimina las lineas que contienen la palabra
$  sed '/reverse/d' E_coli_K12_genes.txt
```

AWK

Dentro de las herramientas del sistema UNIX **awk** es útil para modificar archivos, buscar y transformar bases de datos, generar informes simples y otras muchas cosas.

También se puede utilizar para realizar el tipo de funciones que proporcionan muchas de las otras herramientas del sistema UNIX –buscar patrones, como egrep, o modificar archivos, como tr o sed -.

Pero puesto que también es un lenguaje de programación, resulta más potente y flexible que cualquiera de ellos.

Awk está especialmente diseñado para trabajar con archivos estructurados y patrones de texto.

Muchos programas útiles awk solamente son de una línea de longitud, pero incluso un programa awk de una línea puede ser el equivalente de una herramienta regular del sistema UNIX.

Por ejemplo, con un programa awk de una línea puede contarse el número de líneas de un archivo (como wc), imprimir el primer campo de cada línea (como cut), imprimir todas las líneas que contienen la palabra (como grep), etc.

Sin embargo, awk es un lenguaje de programación con estructuras de control, funciones, y variables. Así, si se aprenden órdenes awk adicionales, pueden escribirse programas más complejos.

Un programa o script awk consiste en una serie de reglas y definiciones de funciones.

Una **regla** (rule) contiene un **patrón** y una **acción**, y cualquiera de las dos puede ser omitida.

El propósito de la acción es decirle a awk qué tiene que hacer una vez que se encuentra el patrón en un registro de entrada. Por lo que el programa completo parecería algo como esto:

[patrón] [{ acción}]
[patrón] [{ acción}]

Una **acción** consiste en una o más sentencias awk, encerradas entre llaves `{‘ y `}’.

Cada **sentencia** especifica una cosa que se tiene que hacer. Las sentencias son separadas por caracteres de nueva línea o puntos y comas (si hay más de una sentencia en una misma línea)

Las **llaves** alrededor de una acción deben ser usadas incluso si la acción contiene una sola sentencia, o si no contiene ninguna sentencia. Sin embargo, si omites la acción por completo, omite también las llaves. (La omisión de la acción es equivalente a `{ print $0 }'

Partes de un archivo

	campos					
	$1	$2	$3	.....	....	
	ECK120001251	thrL	b0001	190	255	forward
	ECK120000987	thrA	b0002	337	2799	forward
$0	ECK120000988	thrB	b0003	2801	3733	forward
	ECK120000989	thrC	b0004	3734	5020	forward
	ECK120002701	yaaX	b0005	5234	5530	forward
	ECK120000009	yaaA	b0006	5683	6459	reverse
	ECK120001508	yaaJ	b0007	6529	7959	reverse
	ECK120001509	talB	b0008	8238	9191	forward
	ECK120001467	mog	b0009	9306	9893	forward
	ECK120001468	satP	b0010	9928	10494	reverse

línea

Ejemplo

El siguiente comando ejecuta un programa awk simple que busca la cadena de caracteres ‘araC’ en el fichero de entrada `tf-gene-interactions.txt` y si la encuentra la imprime.

```
$ awk '/araC/ {print $0} ' tf-gene-interactions.txt
```

- Cuando se encuentran líneas que contengan ‘araC’, éstas son impresas, ya que ‘`print $0`’ hace que se imprima la línea actual.
- Las barras indican que ‘`araC`’ es un patrón para búsqueda. Este tipo de patrón se llama expresión regular.
- Existen comillas simples alrededor del programa awk para que el shell no interprete el programa, como caracteres especiales de la shell.

En una regla awk, el patrón o la acción puede ser omitida, pero no ambos. Si el patrón se omite, entonces la acción se realiza para cada línea de entrada. Si se omite la acción, la acción por defecto es imprimir todas las líneas que concuerden con el patrón.

```
# sin la acción

$ awk '/araC/ ' tf-gene-interactions.txt


# sin el patrón

$ awk '{print $0} ' tf-gene-interactions.txt


# sin indicar la instrucción print

$ awk '/araC/ {} ' tf-gene-interactions.txt
```

Ejemplo con 2 reglas. Awk lee los ficheros de entrada, línea a línea. Para cada línea, awk comprueba todos los patrones de todas las reglas. Si concuerdan varios patrones entonces se ejecutan las distintas acciones de cada patrón, en el orden en que aparecen en el programa awk. Si no concuerda ningún patrón, entonces no se ejecuta ninguna acción.

```
# buscando dos valores en patrones por separado
$ awk '/araC/ {print $0} \
>      /AraC/ {print $0}' tf-gene-interactions.txt

$ awk '/araC/ {print $0}; /AraC/ {print $0}' tf-gene-
interactions.txt
```

Awk y egrep

```
# buscando dos valores en un mismo patrón
$ awk '/araC|AraC/ {print $0} ' tf-gene-interactions.txt

$ awk '/araC|AraC/' tf-gene-interactions.txt



$ grep -E "AraC|araC" tf-gene-interactions.txt
```

Cambiando el orden de las columnas

```
#
$ awk '{print $4,$5,$2}' E_coli_K12_genes.txt

# grep ECK12 E_coli_K12_genes.txt | awk '{print $4,$5,$2}'
|more
```

Variables Implicitas que te cambian el comportamiento de awk

FS. Es el separador de los **campos** de **entrada**. El valor es un único carácter o una expresión regular multi-carácter que busca las separaciones entre campos en un registro de entrada. El valor por defecto es “ “, una cadena consistente en un único espacio en blanco. Como excepción especial, este valor significa realmente que cualquier secuencia de espacios y tabuladores forman un único separador. Hace también que los espacios y tabuladores al principio de línea sean ignorados.

Puedes fijar el valor de FS en la línea de comando usando la opción ‘-F’:

```
awk -F, ‘programa ’ ficheros_de_entrada
```

Operaciones con awk

Operadores aritméticos:

+	suma
-	resta
*	multiplicación
/	división
^ **	exponenciación

Obtener el tamaño de los genes del genoma.

- El archivo `E_coli_K12_genes.txt` sus columnas están separadas por tabuladores.
- Queremos imprimir el tamaño como última columna

```
#
$ awk -F"\t" '/ECK12/ {print $0,($5-$4)} ' E_coli_K12_genes.txt  |more

# separador de campos de salida es un tabulador
$ awk -F"\t"  'BEGIN {OFS = "\t"} /ECK12/ {print $0,($5-$4)} '
E_coli_K12_genes.txt  |more
```

- Cuál es el tamaño mas grande y mas pequeño del gene?

Usando funciones de awk

```
# Total de genes cuya secuencia de DNA sea mayor a 3000
$ grep -v  '#' Gene_sequence.txt| cut -f7 | awk 'length($0)>3000' |wc

#
$ grep -v  '#' Gene_sequence.txt| cut -f2,7 | awk 'length($2)>3000' |
more
```

BEGIN y END

Son patrones especiales. Ellos no son usados para encajar con registros de entrada. En su lugar, ellos son usados para suministrar al script awk **qué hacer antes de empezar a procesar y después de haber procesado los registros de la entrada**. Una regla BEGIN se ejecuta una vez, antes de leer el primer registro de entrada. Y la regla END se ejecuta una vez después de que se hayan leído todos los registros de entrada. Por ejemplo:

```
$ awk 'BEGIN { print "Análisis de araC" } \
/araC/ { ++gene} \
END { print ”araC aparece " gene " veces.” }' tf-
gene-interactions.txt
```

```
$ awk 'BEGIN {print "analisis de araC" }; /araC/
{ ++gene; print $0;}; END {print "araC aparece "
gene " veces" } ' tf-gene-interactions.txt
```

Imprimir las líneas que existen entre dos patrones

```
#
$ awk '/STAT-/,/DP  -/' PMIDs_RegulonDB_8.5_medline.txt |more
```

Perl

- Perl significa Practical Extraction and Report Language, algo así como lenguaje práctico de extracción y de informes.

- Perl es un lenguaje interpretado, aunque en realidad, el intérprete de Perl, como todos los intérpretes modernos, compila los programas antes de ejecutarlos. Por eso se habla de scripts, y no de programas, concepto referido principalmente a programas compilados al lenguaje máquina nativo del ordenador y sistema operativo en el que se ejecuta.

Aplicaciones del lenguaje Perl

- Prácticamente, sirve para todo. Todas las tareas de administración de UNIX se pueden simplificar con un programa en Perl. Se usa también para tratamiento y generación de ficheros de texto. También hay proyectos completos y complejos escritos en Perl.

Help

Ejecutar un programa en Perl

```
perl [-opciones] [programfile] [arguments]
```

La ayuda

```
$ perl -help
```

Opciones

- -e one line of program
- -n assume "while (<>) { ... }" loop around program
- -a autosplit mode with -n or -p (splits $_ into @F)

```
perl –e ‘instrucciones’ archivo
```

- Usa Perl para facilitar el trabajo rutinario: Perl one-liners
- Para muchos trabajos puntuales como edición y parseo de archivos de texto es muy practico saber usar Perl desde la línea de comandos.

```
$ perl -e ‘print "Hello\n"'
$ perl -e 'print " 10*2 \n"; '
$ perl -e 'print 10*2, "este es el resultado \n"; ’

$ perl -n -e 'print; ' tf-gene-interactions.txt
$ perl -ne 'print $_; ' tf-gene-interactions.txt
```