UNIVERSIDAD DE COSTA RICA

FACULTAD DE CIENCIAS

Escuela de Física

Tarea 1: Bash

Estudiante

Ángel Fabricio Aguirre Bermúdez - C10152

Docente

Marlon E. Brenes Navarro

II Semestre 2024

05 de septiembre de 2024

Es importante mencionar que a la hora de copiar y pegar los comandos presentados en este documento, se debe de eliminar los cambios de línea (*Enter*); por lo demás, los comandos deberían estar listos para funcionar.

1. random.sh

1.1.

```
./random.sh | tail -n 11 | head -n 10 | grep '^1'
```

Tras invocar el ejecutable, se elimina el título y los decoradores con `head` y `tail`, donde `-n` se usa para referirse a líneas, y el número que sigue es la cantidad que conservo; con el `stdout` se alimenta `grep` para que busque el 1, y `^` se refiere al inicio de cada línea.

1.2.

```
./random.sh | tail -n 11 | head -n 10 | grep '^1' | awk 'length()<=4'
```

El resultado del punto anterior se redirecciona a `awk`, donde `length()` devuelve la cantidad de caracteres en cada línea, y con la desigualdad le pido que este valor sea igual o menor a 4.

1.3.

```
printf '%6s %6s %6s %6s %6s\n' $(for i in $(seq 500); do ./random.sh | tail -n 11 | head -n 10
    ; done) > matriz.txt
```

`printf` va acomodando los datos en forma de matriz: `%6s %6s %6s %6s %6s` indica que son 5 columnas, `6` indica que son 6 caracteres por celda, `s` indica que es una cadena de texto y `\n` indica cambio de línea. `%( . . . )` ejecuta el comando dentro de los paréntesis y luego reemplaza la expresión con el `stdout`. `seq(500)` genera una secuencia de números del 1 al 500. Entre el `do` y el `done` va el código anterior que me da 10 números (10 números ejecutado 500 veces es igual 5000 números). Finalmente todo se envia al archivo `matriz.txt` con `>`.

```
wc -c matriz.txt
```

Esto es para identificar el espacio que requiere el archivo, `-c` sirve para referirse a los bytes.

1.4.

```
sort -n -k 3 matriz.txt > matriz_ordenada.txt
```

`sort` ordena las líneas (`-n`: por número `-k`: por columna `3`: tercera columna). El resultado es redireccionado a `matriz_ordenada.txt`.

1.5.

```
sed 's/0/A/g; s/1/B/g; s/2/C/g; s/3/D/g; s/4/E/g; s/5/F/g; s/6/G/g; s/7/H/g; s/8/I/g; s/9/J/g'
    matriz_ordenada.txt
```

`sed` se alimanta de `s/patrón/reemplazo/g` donde `s` representa sustitución, y `g` indica que debe hacerse globalmente en la línea. Así se va reemplazando cada número de `matriz_ordenada.txt` por una letra.

2. usuarios.csv

2.1.

```
grep '^Brian,' usuarios.csv | grep '.com,' | grep ',Boeing'
```

Primero se busca "Brian", `^` se refiere al inicio de la línea, se agrega el separador (`,`) al final para evitar resultados indeseados. Luego se realiza el mismo procedimiento para ".com" y "Boeing", con el cuidado de considerar sus posiciones.

2.2.

```
grep -E '^James,|^Paul,' usuarios.csv | awk -F',' '$2 ~ /^J|^S/' | grep ',Ad Astra'
```

Primero se busca "James" o "Paul", con `-E` le pido a `grep` que utilice expresiones regulares extendidas (`|` actúa como operador OR). Después se utiliza `awk`, donde `-F','` establece que el delimitador es una coma, `$2` representa la segunda columna (apellidos), `~` se usa para comprobar si el valor de los apellidos coincide con la expresión que sigue, `/^J|^S/` se refiere a que comiencen con J o con S. Finalmente se usa otro grep para comprobar la coincidencia con "Ad Astra".

2.3.

```
grep '^M' usuarios.csv | wc -l
```

`wc -l` se utiliza para contar las líneas.

```
grep '^M' usuarios.csv > auxiliar.txt
```

Acá es lo mismo pero se redirecciona al archivo `auxiliar.txt`.

2.4.

```
sed 's/$/,5555-5555/' usuarios.csv | sed '1s/5555-5555/phone_number/'
```

Se vuelve a utilizar `sed`, pero acá `$` representa el final de cada fila, el único problema es que a la primera línea (la de los títulos) también se le agregó el número, esto se corrige con otro `sed`, donde `1s` indica que la sustitución solo debe aplicarse sobre la primera línea.

3. test.txt

```
paste <(grep -v '#' test1.txt | cut -d " " -f 1) <(grep -v '#' test1.txt | cut -d " " -f 2)
    <(grep -v '#' test2.txt | cut -d " " -f 2) <(grep -v '#' test3.txt | cut -d " " -f 2) <(grep
    -v '#' test4.txt | cut -d " " -f 2) > merged.txt
```

`paste` se usa para combinar las 5 columnnas en un solo archivo (`merged.txt`). Cada columna se logra de la siguiente manera: `<( . . . )` permite usar la salida de un comando como si fuera un archivo (importante porque `paste` espera archivos), `grep -v '#' test□.txt` busca las líneas que **no** contengan `#` de cada archivo (con `-v`), con esto se alimenta `cut`, que con `-d` se le señala el separador (1 espacio) y con `-f` □ se señala la columna deseada.