Sistemas operativos
Taller 1

Juan David Lara Camacho

1. Analizamos qué hace cada comando:
    a. $ ls | wc -l
    Este comando es un pipe que consta de dos comandos: **ls** y **wc**.
    - El comando **ls** lista los archivos y directorios en el directorio actual.
    - El carácter **|** (pipe) toma la salida del comando **ls** y la pasa como entrada al siguiente comando en la tubería.
    - El comando **wc** (word count) cuenta el número de palabras, líneas y caracteres de su entrada.

    Por lo tanto, cuando ejecutamos el comando completo **ls | wc -l**, estamos contando el número de líneas que se muestran en la salida de **ls**. Es decir, el resultado de este comando es el número de archivos y directorios en el directorio actual.

    b. $ last | grep reboot

    El comando last muestra un registro de los últimos inicios y cierres de sesión en el sistema. Mientras tanto, grep es una herramienta que se utiliza para buscar patrones de texto en una entrada.
    Entonces, el comando completo last | grep reboot utiliza una tubería (pipe) para tomar la salida del comando last y pasarla al comando grep. En este caso, el patrón de texto que se busca con grep es "reboot".
    Por lo tanto, al ejecutar este comando, se muestran las líneas del registro de last que contienen la palabra "reboot", lo que nos permite ver cuándo se reinició el sistema por última vez.

    c. $ history

    El comando history muestra una lista de los comandos que se han ejecutado previamente en la sesión actual del terminal. Por defecto, history muestra los últimos 500 comandos ejecutados, junto con su número de identificación en el historial. Para ejecutar uno de los comandos anteriores, se puede utilizar !n, donde "n" es el número de identificación del comando en el historial.

```
root@ubuntu:~# ls | wc -l
0
root@ubuntu:~# last | grep reboot
reboot   system boot  4.4.0-87-generic Fri Mar 31 15:36   still running
reboot   system boot  4.4.0-87-generic Fri Mar 31 15:09 - 15:12  (00:03)
reboot   system boot  4.4.0-87-generic Fri Mar 31 15:01 - 15:12  (00:10)
reboot   system boot  4.4.0-87-generic Sat Aug 11 13:38 - 15:12 (1693+01:34)
reboot   system boot  4.4.0-87-generic Thu Jul 19 14:38 - 15:12 (1716+00:34)
reboot   system boot  4.4.0-87-generic Thu Jul 19 14:34 - 15:12 (1716+00:38)
reboot   system boot  4.4.0-87-generic Thu Jul 19 13:12 - 15:12 (1716+02:00)
reboot   system boot  4.4.0-87-generic Tue Jun 12 13:16 - 15:12 (1753+01:55)
reboot   system boot  4.4.0-87-generic Thu Jan 25 09:45 - 15:12 (1891+04:27)
reboot   system boot  4.4.0-87-generic Tue Jan 23 14:11 - 15:12 (1893+00:01)
reboot   system boot  4.4.0-87-generic Sun Jan 21 17:53 - 15:12 (1894+20:19)
reboot   system boot  4.4.0-87-generic Tue Jan 16 12:53 - 15:12 (1900+01:18)
reboot   system boot  4.4.0-87-generic Tue Jan  9 16:36 - 15:12 (1906+21:35)
reboot   system boot  4.4.0-87-generic Tue Jan  9 13:40 - 15:12 (1907+00:32)
reboot   system boot  4.4.0-87-generic Mon Jan  8 16:03 - 15:12 (1907+22:09)
reboot   system boot  4.4.0-87-generic Fri Jan  5 14:54 - 15:12 (1910+23:18)
reboot   system boot  4.4.0-87-generic Fri Jan  5 14:29 - 14:32  (00:02)
reboot   system boot  4.4.0-87-generic Fri Jan  5 14:25 - 14:29  (00:04)
reboot   system boot  4.4.0-87-generic Fri Jan  5 12:54 - 14:29  (01:35)
root@ubuntu:~# history
    1  mount -rw -o remount /
    2  passwd root
    3  reboot
    4  ls | wc -l
    5  last | grep reboot
    6  history
```

2. En el siguiente código nos ubicamos en el directorio ‘home’ en donde con el comando ‘mkdir’ creamos el directorio ‘jlara’ que corresponde a mi usuario en el dominio unal. Luego, nos movemos al directorio que acabamos de crear y de la misma manera creamos los directorios ejemplo’ y ‘prueba’. Ahora, con el comando cp y el flag -p copiamos el archivo que se encuentra en /etc/ llamado os-release y lo copiamos con el nombre de ‘msg’ en /home/jlara/prueba de manera que se conservan los atributos del archivo original, incluyendo la hora de la última modificación.

```
root@ubuntu:~# ls
root@ubuntu:~# cd /
root@ubuntu:/# ls
bin   dev  home        lib    lost+found  mnt  proc  run   snap  sys  usr  vmlinuz
boot  etc  initrd.img  lib64  media       opt  root  sbin  srv   tmp  var
root@ubuntu:/# cd home/
root@ubuntu:/home# mkdir jlara
root@ubuntu:/home# cd jlara/
root@ubuntu:/home/jlara# mkdir ejemplo
root@ubuntu:/home/jlara# mkdir prueba
root@ubuntu:/home/jlara# cp -p /etc/os-release /home/jlara/prueba/msg
root@ubuntu:/home/jlara# cd prueba/
root@ubuntu:/home/jlara/prueba# ls
msg
```

3. En el siguiente código tomamos la salida del comando ‘ls’ y la escribimos en un archivo llamado texto1.txt. Podemos observar que este archivo está vacío, esto dado que en el directorio en el que estamos ‘home/jlara/ejemplo´ no hay archivos. Pero luego de crearlo podemos verificar la creación del archivo texto1.txt.

```
root@ubuntu:/home/jlara/prueba# cd ../ejemplo
root@ubuntu:/home/jlara/ejemplo# ls > texto1.txt
root@ubuntu:/home/jlara/ejemplo# ls
texto1.txt
root@ubuntu:/home/jlara/ejemplo# less texto1.txt
texto1.txt
texto1.txt (END)
```

4. Para agregar ahora la salida del comando ps -fea al final del archivo text2.txt, utilizamos el operador >> en lugar de > para agregar la salida al final del archivo en lugar de sobrescribirlo. Finalmente, para agregar la salida del comando **top** al final del archivo **texto2.txt**, se puede utilizar el siguiente comando:

```
root@ubuntu:/home/jlara/ejemplo# ls
texto1.txt
root@ubuntu:/home/jlara/ejemplo# ps > texto2.txt
root@ubuntu:/home/jlara/ejemplo# ps -fea >> texto2.txt
root@ubuntu:/home/jlara/ejemplo# top -b -n 1 >> texto2.txt
root@ubuntu:/home/jlara/ejemplo# ls
texto1.txt  texto2.txt
```

5. Este comando busca todas las líneas en texto2.txt que contienen el patrón '[k', que es una cadena de texto que se usa para identificar los procesos de kernel en la salida del comando ps. El flag -c indica que se cuente el número de líneas que contienen ese patrón.

```
root@ubuntu:/home/jlara/ejemplo# grep "\[k" texto2.txt
root         2     0  0 15:36 ?        00:00:00 [kthreadd]
root         3     2  0 15:36 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 15:36 ?        00:00:00 [kworker/0:0H]
root        11     2  0 15:36 ?        00:00:00 [kdevtmpfs]
root        14     2  0 15:36 ?        00:00:00 [khungtaskd]
root        16     2  0 15:36 ?        00:00:00 [ksmd]
root        17     2  0 15:36 ?        00:00:00 [khugepaged]
root        19     2  0 15:36 ?        00:00:00 [kintegrityd]
root        21     2  0 15:36 ?        00:00:00 [kblockd]
root        28     2  0 15:36 ?        00:00:00 [kswapd0]
root        47     2  0 15:36 ?        00:00:00 [kthrotld]
root       130     2  0 15:36 ?        00:00:00 [kpsmoused]
root       131     2  0 15:36 ?        00:00:00 [kworker/0:1H]
root       361     2  0 15:36 ?        00:00:00 [kauditd]
root      1185     2  0 16:16 ?        00:00:00 [kworker/0:2]
root      1268     2  0 16:17 ?        00:00:04 [kworker/0:5]
root      1343     2  0 17:19 ?        00:00:00 [kworker/u2:2]
root      1344     2  0 17:25 ?        00:00:00 [kworker/u2:1]
root      1358     2  0 17:30 ?        00:00:00 [kworker/u2:0]
root@ubuntu:/home/jlara/ejemplo# grep -c "\[k" texto2.txt
19
```

6. El comando **sort** ordena las líneas que recibe y el flag **-u** le indica que solo muestre las líneas únicas, eliminando así duplicados. Después de ejecutar el siguiente código, el archivo texto3.txt contendrá una lista ordenada y única de los procesos de kernel que se encontraron en texto2.txt.

```
root@ubuntu:/home/jlara/ejemplo# grep -c "\[k" texto2.txt | sort -u > texto3.txt
```

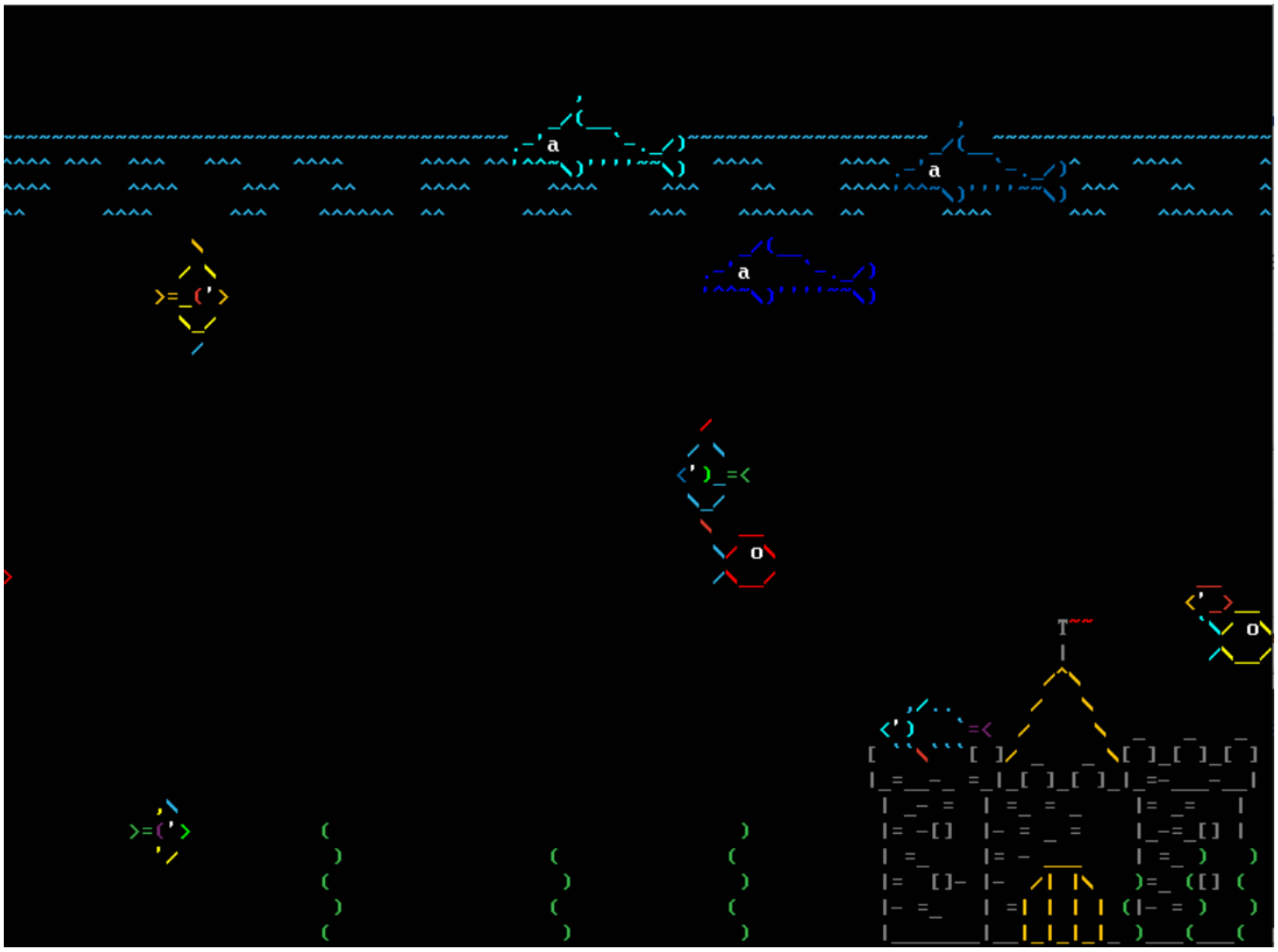
7. Corremos los comandos mencionados:

```
root@ubuntu:/home/jlara/prueba# pwd > info.txt
root@ubuntu:/home/jlara/prueba# date >> info.txt
root@ubuntu:/home/jlara/prueba# cal 2023 >> info.txt
root@ubuntu:/home/jlara/prueba# cal 3 2023 >> info.txt
root@ubuntu:/home/jlara/prueba# finger osc >> info.txt
root@ubuntu:/home/jlara/prueba# hostname >> info.txt
root@ubuntu:/home/jlara/prueba# who >> info.txt
root@ubuntu:/home/jlara/prueba# whoami >> info.txt
root@ubuntu:/home/jlara/prueba# echo "Esto es un taller" >> info.txt
root@ubuntu:/home/jlara/prueba#
```

8. Corremos los códigos y se instalan correctamente

9. El código hace lo siguiente:
    a. sudo apt-get install libcurses-perl: Este comando instala la biblioteca "libcurses-perl" que proporciona una interfaz para programación de aplicaciones para la manipulación de pantallas de terminal en modo texto.
    b. wget http://search.cpan.org/CPAN/authors/id/K/KB/KBAUCOM/Term-Animation-2.4.tar.gz: Este comando descarga el archivo comprimido "Term-Animation-2.4.tar.gz" desde el sitio web de CPAN.
    c. tar Term-Animation-2.4.tar.gz: Este comando descomprime el archivo "Term-Animation-2.4.tar.gz" utilizando el comando tar. Se debe leer el manual del comando para encontrar la opción adecuada para descomprimir, que en este caso es -zxf.
    d. sudo make install: Este comando instala el paquete "Term-Animation-2.4" utilizando el archivo Makefile que se generó previamente. El comando make se utiliza para compilar el código fuente y generar los archivos binarios que se instalan.
    e. /usr/local/bin/asciiquarium: Este comando ejecuta el programa "asciiquarium" que se instaló previamente en /usr/local/bin utilizando los comandos sudo cp y sudo chmod. El

programa muestra una animación en la terminal que simula un acuario con peces ASCII nadando en la pantalla.



10.

a. Se crearon tres directorios: "home/ jlara ", "home/ jlara /prueba" y "home/ jlara /ejemplo". El propietario de estos directorios es "root" y los permisos son "drwxr-xr-x", lo que significa que el dueño puede leer, escribir y ejecutar, mientras que los otros usuarios solo pueden leer y ejecutar.
b. Se crearon cinco archivos: "home/ jlara /ejemplo/texto1.txt", "home/ jlara /ejemplo/texto2.txt", "home/ jlara /ejemplo/texto3.txt", "home/ jlara /prueba/msg" y "home/ jlara /prueba/punto7.txt". El propietario de estos archivos es "root" y los permisos son "-rw-r--r--", lo que significa que el dueño puede leer y escribir, mientras que los otros usuarios solo pueden leer.
c. Se creó un directorio "home/ jlara /prueba/asciiquarium_1.1". El propietario de este directorio es "osc" y los permisos son "drwxrwxr-x", lo que significa que el dueño y los

otros usuarios de su grupo pueden leer, escribir y ejecutar, mientras que los demás usuarios solo pueden leer y ejecutar.

d. Se creó un directorio "home/Term-Animation/prueba/Term-Animation-2.4". El propietario de este directorio es "osc" y los permisos son "drwxr-xr-x", lo que significa que el dueño puede leer, escribir y ejecutar, mientras que los otros usuarios solo pueden leer y ejecutar.
e. Se creó un archivo "usr/local/bin/asciiquarium". El propietario de este archivo es "root" y los permisos son "-rwxr-xr-x", lo que significa que el dueño puede leer, escribir y ejecutar, mientras que los otros usuarios solo pueden leer y ejecutar.