Asignatura	Datos del alumno	Fecha
Introducción a la Programación Científica	Apellidos: Santos Tedim Sousa Pedrosa	25/01/2026
	Nombre: Ana Sofia	

Fundamentos de Bash: Manejo de archivos, directorios y scripts

1. **¿Cuál es el nombre del usuario actual? ¿Y el nombre del sistema?** Indique dónde se puede conocer esta información.

Para saber el nombre del usuario actual podemos usar los comandos:

whoami o *echo $USER*.

Para saber cuál es el nombre del sistema podemos usar tanto los comandos:

uname -n o *hostname*

Esto comandos muestran el nombre asignado al equipo dentro del sistema operativo. Esta información se obtiene colocando los comandos referidos arriba directamente en la línea de comando de Linux.

2. **¿Qué versión de Bash estás utilizando?**

La versión de Bash utilizada en el sistema se puede conocer ejecutando el comando:

bash --version

Este comando muestra por pantalla la versión del intérprete de comandos Bash instalada y en uso en el sistema. Para realizar esta actividad se usó la versión 5.2.21(1)-release (x86_64-pc-linux-gnu).

3. **¿Cuál es el directorio de trabajo actual?** Crea en el directorio de inicio tres subdirectorios: «curso2024», «curso2025» y «UNIR».

El directorio de trabajo actual se puede conocer mediante el comando:

pwd

Este comando muestra la ruta absoluta del directorio en el que se está trabajando. En este caso, devuelve */home/anaptedim*, que corresponde al directorio de inicio del usuario.

Para crear los subdirectorios en el directorio de inicio se utiliza el comando:

mkdir curso2024 curso2025 UNIR

Este comando permite crear varios directorios simultáneamente.

Para confirmar que los directorios se han creado correctamente en el directorio de inicio, se puede usar el comando ***ls***, que muestra el listado de archivos y subdirectorios del directorio de trabajo actual.

4. **Sin acceder al directorio «UNIR», crea dentro los siguientes 10 ficheros: Fichero1.txt, Fichero2.txt … Fichero10.txt.** Indica dos formas diferentes de crearlos. Ayuda: Explorar el uso de los caracteres `{}`.

Para responder a esta pregunta se puede usar el comando:

touch UNIR/Fichero{1..10}.txt.

Este comando permite crear diez ficheros con extensión .txt dentro del directorio UNIR sin necesidad de acceder a él, ya que *touch* crea archivos vacíos indicando directamente la ruta al directorio destino. El uso de la expansión de llaves {1..10} permite generar automáticamente múltiples nombres de fichero.

Una segunda opción es usar el comando:

for i in {1..10}; do echo "" > UNIR/Fichero$i.txt; done

En el cual se usa un bucle for junto con el comando echo y el operador de redirección >. Mediante este bucle se generan de forma iterativa los ficheros Fichero1.txt a Fichero10.txt dentro del directorio UNIR, sin necesidad de acceder a dicho directorio.

Para verificar que los ficheros se han creado correctamente en ambos caso, se puede acceder al directorio UNIR mediante el comando ***cd UNIR*** y ver su contenido con el comando ***ls***.

5. **Elimina los directorios «UNIR» y «curso2025» utilizando el comando `rmdir`.** ¿Puedes eliminar ambos directorios? Si no puedes eliminar alguno, explica cómo lo harías y por qué.

Para responder a la pregunta se usó inicialmente el comando:

mkdir curso2025 UNIR

Sin embargo, el comando ***rmdir*** solo permite eliminar directorios que estén vacíos. El directorio curso2025 se puede eliminar correctamente porque no contiene ningún archivo en su interior. No obstante, el directorio UNIR no se puede eliminar usando *rmdir* porque contiene los ficheros previamente creados.

Para eliminar el directorio UNIR sería necesario borrar primero los archivos que contiene o utilizar el comando:

rm -r UNIR

Este comando permite eliminar de forma recursiva el directorio junto con todo su contenido.

Los ficheros contenidos dentro del directorio UNIR se pueden eliminar utilizando el comando *rm*, indicando la ruta a los archivos sin necesidad de acceder al directorio. Por ejemplo, se puede ejecutar:

rm UNIR/Fichero.txt*

Este comando permite borrar todos los ficheros con extensión .txt dentro del directorio UNIR. Una vez eliminado el contenido, el directorio quedaría vacío y podría eliminarse con el comando *rmdir* UNIR.

6. **Accede al directorio «curso2024» y descarga el fichero `plasmids.txt` a través del siguiente enlace. ¿Cómo puedes conocer más información acerca de este comando?** Indica dos formas de acceder a la ayuda de los comandos. https://ftp.ncbi.nlm.nih.gov/genomes/GENOME_REPORTS/plasmids.txt

Para acceder al directorio curso2024 se utiliza el comando:

cd curso2024.

La descarga del fichero plasmids.txt desde el enlace proporcionado se realiza mediante el comando:

wget https://ftp.ncbi.nlm.nih.gov/genomes/GENOME_REPORTS/plasmids.txt

Este comando permite descargar archivos desde Internet usando la línea de comandos.

Para conocer más información acerca de este comando se puede acceder a su ayuda de distintas formas, como por ejemplo usando el manual del sistema con el comando:

man wget

o con la opción de ayuda rápida del propio comando:

wget --help.

7. **Lista el contenido del directorio actual. ¿Qué permisos (usuario/grupo/todos) tiene el fichero `plasmids.txt`? ¿Cuánto tamaño ocupa en Mb?** Busca en la ayuda del comando `ls` el parámetro utilizado para mostrar este tamaño en un formato fácilmente leíble por humanos.

Para ver el contenido del directorio actual se utiliza el comando:

ls

El uso de este comando indica que dentro del directorio curso2024 hay un fichero llamado plasmids.txt.

Para consultar los permisos del fichero plasmids.txt se usó el comando:

ls -l

Este comando devuelve el listado de ficheros o directorios en formato largo permitiendo ver los permisos diferenciados para usuario, grupo y todos, así como el propietario y el grupo asociados al archivo. En este caso el resultado obtenido es:

-rw-rw-r-- 1 anaptedim anaptedim 7566137 oct 10 2024 plasmids.txt

La información obtenida indica que tanto el usuario como el grupo tiene permisos para leer y escribir en el fichero, pero todos solo tienen permisos para leer el fichero. El propietario del archivo en anaptedim y el grupo al que pertenece también es anaptedim.

El tamaño del fichero plasmids.txt se puede ver utilizando el comando:

ls -lh

En este comando el parámetro -h permite ver el tamaño en unidades como KB, MB o GB según corresponda en un formato fácilmente legible para humanos. Este comando devuelve:

total 7,3M

-rw-rw-r-- 1 anaptedim anaptedim 7,3M oct 10 2024 plasmids.txt

Indicando que el fichero plasmidos.txt tiene un tamaño de 7,3MB.

8. **Comprime y descomprime este fichero en los formatos *.tar* y *.gz***. Mantén el archivo original sin modificar.

Para comprimir el fichero plasmids.txt en formato .tar se utiliza el comando tar, que permite agrupar archivos en un único fichero sin modificar el archivo original. La opción -c crea el archivo, -v muestra información del proceso y -f indica el nombre del archivo resultante. El comando utilizado fue:

tar -cvf plasmids.tar plasmids.txt

Para descomprimir el fichero plasmids.tar sin modificar el archivo original, se creó previamente un nuevo directorio mediante el comando *mkdir extraer_tar* y se utilizó el comando:

tar -xvf plasmids.tar -C extraer_tar

Este comando permite extraer el contenido del archivo plasmids.tar en el directorio extraer_tar. En este caso, el parámetro -x se usa para extraer el contenido, -v muestra por pantalla los archivos que se van extrayendo durante el proceso, -f especifica el nombre del archivo sobre el que se realiza la operación y -C indica el directorio donde se desea descomprimir el archivo .tar.

Para comprimir el fichero en formato .gz se utilizó el algoritmo gzip con el comando:

gzip -c plasmids.txt > plasmids.txt.gz

En este comando, el parámetro -c indica que el resultado de la compresión se envíe a la salida estándar en lugar de reemplazar el archivo original. Sin esta opción, gzip eliminaría el fichero plasmids.txt y dejaría únicamente el archivo comprimido. El operador > redirige la salida estándar al archivo plasmids.txt.gz, que es el fichero comprimido resultante.

Para descomprimir el archivo plasmids.txt.gz sin sobrescribir el archivo original se utilizó el comando:

gunzip -c plasmids.txt.gz > plasmids_desc.txt

En este caso, la opción -c permite enviar el contenido descomprimido a la salida estándar, creando un nuevo archivo llamado plasmids_desc.txt y manteniendo intactos tanto el fichero original plasmids.txt como el archivo comprimido plasmids.txt.gz.

Para verificar que todas las operaciones se realizaron correctamente se utilizó el comando *ls*.

9. **Crea un nuevo archivo llamado `plasmids_final.txt` que contenga las últimas 8 filas del archivo *plasmids.txt***. Para ello, explora el uso de `>` y `>>`. ¿Para qué se utiliza cada uno?

Para crear al archivo plasmids_final.txt como pedido en la pregunta se usó el comando:

tail -n 8 plasmids.txt > plasmids_final.txt

El comando tail -n 8 se utiliza para extraer las últimas 8 líneas del archivo plasmids.txt. El operador de redirección > envía la salida del comando a un nuevo archivo llamado plasmids_final.txt. Si el archivo no existe, se crea; si existe, su contenido se sobrescribe.

El operador de redirección >> se utiliza para añadir contenido al final de un archivo existente, sin sobrescribir la información que ya contiene. A diferencia de >, que reemplaza el contenido previo.

10. **Muestra las 4 primeras filas del archivo `plasmids_final.txt` por pantalla**.

Para mostrar por pantalla las 4 primeras filas del archivo plasmids_final.txt se usó el comando:

head -n 4 plasmids_final.txt

El comando *head* permite mostrar las primeras líneas de un archivo. Se usó la opción *-n 4* para indicar que se muestren apenas las cuatro primeras filas del archivo plasmids_final.txt por pantalla.

11. **¿Cómo se podrían hacer los pasos 9 y 10 en un solo paso?** Ayuda: haz uso de los operadores lógicos.

Para hacer los pasos 9 y 10 en un solo se usó el comando:

tail -n 8 plasmids.txt > plasmids_final.txt && head -n 4 plasmids_final.txt

Se utiliza el operador lógico &&, que permite encadenar dos comandos de tal forma que el segundo comando solo se ejecute si el primero comando se ha ejecutado correctamente.

12. **En el archivo `plasmids.txt`, ¿cuántos genomas hay de *Klebsiella michiganensis*? ¿Y de *Pantoea agglomerans*?** Indica cuántas líneas y cuántos caracteres tiene dicho archivo.

Para saber el número de genomas de *Klebsiella michiganensis* y de *Pantoea agglomerans* que contiene el archivo plasmids.txt se usaron los comandos:

grep -c "Klebsiella michiganensis" plasmids.txt

grep -c "Pantoea agglomerans" plasmids.txt

El comando *grep* permite buscar texto en un archivo y se le añadimos el parámetro *-c* permite contar el número de líneas que contienen una cadena específica dentro de un archivo. Estos comando devuelven los números 225 y 148 respectivamente que corresponden al número de genomas de *Klebsiella michiganensis* y de *Pantoea agglomerans* presentes en el archivo plasmids.txt.

Para conocer el número total de líneas del archivo plasmids.txt se usó el comando:

wc -l plasmids.txt

La ejecución de este comando indica que el archivo plasmids.txt tiene 59388 líneas.

Para conocer el número total de caracteres del archivo plasmids.txt se usó el comando:

wc -m plasmids.txt

La ejecución de este comando indica que el archivo plasmids.txt contiene 7566137 caracteres.

13. **Crear un *Shell script* de Bash llamado `dormir.sh` que ejecute el comando `sleep` durante 500 segundos**.

Para crear el Shell script llamado *dormir.sh* en el directorio de trabajo actual se usó el editor de texto nano desde la línea de comandos, creando el archivo mediante el comando:

nano dormir.sh.

Dentro del archivo se incluyó el comando sleep 500, que hace que el proceso permanezca en espera durante 500 segundos. Además, se añadió la línea #!/bin/bash al inicio del archivo para indicar que el script debe ejecutarse utilizando el intérprete Bash.

Tras guardar el archivo (Ctrl-O) y cerrar el editor de texto (Ctrl-X), se comprobó su correcta creación listando el contenido del directorio con el comando *ls*.

14. **Cambia los permisos del *script* `dormir.sh`.** Combina la forma de dar/quitar permisos a través del código octal (Ejemplo: *777*) y modo archivo (Ejemplo: *u+x*).

Primero para comprobar los permisos del archivo dormir.sh se ejecutó el comando *ls -l* en el directorio actual de trabajo, que permite ver el listado de archivos y directorios en formato longo donde se muestra los permisos de usuario, grupo y todos. Este comando devuelve la respuesta:

-rw-rw-r-- 1 anaptedim anaptedim 82 ene 25 12:25 dormir.sh

Observándose que el usuario y el grupo tiene permisos de escritura y lectura y todos solo permiso de lectura.

El comando *chmod* permite modificar los permisos de acceso a archivos y directorios. Se usaron dos formas de asignación de permisos: el modo octal, que utiliza valores numéricos para definir los permisos, y el modo simbólico, que permite añadir o eliminar permisos mediante letras que representan al usuario (u), grupo (g) y otros (o).

- Quita todos los permisos al archivo.

Para quitar todos los permisos al archivo se usó el comando:

chmod 000 dormir.sh

- Da permiso de lectura, escritura y ejecución a tu usuario.

Para dar permiso de lectura, escritura y ejecución al usuario se usó el comando:

chmod 700 dormir.sh

- Da permisos de lectura y escritura a todos los usuarios.

Para dar permiso de lectura, escritura a todos los usuarios se usó el comando:

chmod u=rw dormir.sh

- Da solo permisos de ejecución al grupo de usuarios.

Para dar solo permiso de ejecución al grupo de usuarios se usó el comando:

chmod g+x dormir.sh

después de usar el comando *chmd 000 dormir.sh* para quitar todos los demás permisos.

- Quita todos los permisos al grupo y a todos. Solo tu usuario es el que debe tener permisos.

Se usó el comando *chmod 000 dormir.sh* para quitar todos los permisos del archivo dormir.sh y después se usó el comando:

chmod 700 dormir.sh

para dar permisos de escritura, lectura y ejecución a mi usuario.

Para verificar se los comandos mencionados en cada una de las alineas funcionaban y se cambiaban los permisos correctamente se usó nuevamente el comando *ls -l*.

15. **Ejecuta el `Shell` *script* en segundo plano**. Comprueba que se está ejecutando. ¿Qué identificador tiene asociado? Elimina el proceso en ejecución.

Para ejecutar el script dormir.sh en segundo plano se usó el comando:

./dormir.sh &

En este comando se utiliza el operador & que permite lanzar el proceso sin bloquear la terminal.

Una vez ejecutado, se comprueba que el proceso está en ejecución mediante el comando:

ps

El comando *ps* muestra los procesos activos del usuario junto con su identificador de proceso (PID). El identificador del proceso corresponde al valor PID asignado por el sistema en el momento de la ejecución.

Para finalizar el proceso en ejecución se utiliza el comando:

kill 3527

en el que 3527 es el PID del proceso de ejecución del script dormir.sh.

para verificar que el proceso ha terminado se vuelve a usar el comando *ps* que devuelve el siguiente mensaje relativo a la ejecución del archivo dormir.sh:

[1]+ Terminated ./dormir.sh

Indicando que el proceso ha sido terminado correctamente.

Para enviar el Shell script como pedido en el enunciado de la actividad se cambió el nombre de dormir.sh a Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej13.sh mediante el comando:

mv dormir.sh Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej13.sh

16. **Escribe un `Shell` *script* que se llame con un argumento**. Ese argumento será el nombre X de una persona, y será usado para mostrar por pantalla: «Soy X, y me gusta el máster de bioinformática»

Para resolver esta pregunta se creó un Shell script que recibe un argumento por la línea de comandos. El Shell script se creó usando el comando:

nano Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej16.sh

Dentro de este Shell script se usa el primero argumento recibido representado por la variable NOMBRE=$1, para almacenar el nombre de una persona. A continuación, mediante el comando:

echo "Soy $NOMBRE, y me gusta el máster de bioinformática"

el script muestra por pantalla el mensaje «Soy X, y me gusta el máster de bioinformática».

Para comprobar que el script funcionaba correctamente primero se cambiaron los permisos del usuario para poder ejecutar el script mediante el comando:

chmod u+x Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej16.sh

y seguidamente se ejecutó el programa usando el comando:

./Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej16.sh Ana

Ya que para que el script funcione correctamente es necesario pasar el nombre (en este caso Ana) como primer argumento en el momento de su ejecución. Este comando devolvió lo siguiente en pantalla:

Soy Ana, y me gusta el máster de bioinformática

17. **Escribe un `Shell` *script* que lea como argumento una nota numérica y muestre la correspondiente alfabética**. Es decir, N<5 es suspenso, 5<=N<7 es aprobado, 7<=N<9 es notable y 9<=N<=10 es sobresaliente.

Se creó un Shell script que recibe una clasificación numérica de una asignatura (de 0 a 10) como argumento por la línea de comandos. El Shell script se creó usando el comando:

nano Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej17.sh

El script utiliza el primer argumento recibido, almacenado en la NOTA=$1, para evaluar la nota introducida. Mediante una estructura condicional if, elif y else, el script compara el valor de la nota con distintos rangos numéricos. Para garantizar la precisión y permitir el uso de decimales (como 6.5), se integró el comando bc (Basic Calculator) en las evaluaciones lógicas.

Para comprobar que el script funcionaba correctamente primero se cambiaron los permisos del usuario para poder ejecutar el script mediante el comando:

chmod u+x Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej17.sh

y seguidamente se ejecutó el programa usando el comando:

./Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej17.sh 6.5

Ya que para que el script funcione correctamente es necesario colocar una clasificación numérica (en este caso 6.5) como primer argumento en el momento de su ejecución. Este comando devolvió lo siguiente en pantalla:

Aprobado

18. **Escribe un S `Shell` *script* que compruebe si un fichero introducido por el usuario existe**. La ruta de dicho fichero se debe definir dentro de una variable llamada FILE al comienzo del *script*. Primero debes comprobar si el fichero existe, si es así, debes imprimir el mensaje «El fichero X existe» (siendo X la ruta al fichero según FILE); si no existe, debes decir que «el fichero X no existe».

Se creó un Shell script para que se compruebe si existe un fichero introducido por el usuario. El Shell script se creó usando el comando:

nano Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej18.sh

En este script se define al comienzo una variable llamada FILE (FILE="/home/anaptedim/curso2024/plasmids.txt") que contiene la ruta completa del fichero a comprobar. Mediante una estructura condicional if y el operador -e, el script verifica si el fichero existe en la ruta indicada y muestra un mensaje informando del resultado.

Para comprobar que el script funcionaba correctamente primero se cambiaron los permisos del usuario para poder ejecutar el script mediante el comando:

chmod u+x Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej18.sh

y seguidamente se ejecutó el programa usando el comando:

./Santos-Tedim-Sousa-Pedrosa_Ana-Sofia_Act2_Ej18.sh

Este comando devolvió lo siguiente en pantalla:

El fichero /home/anaptedim/curso2024/plasmids.txt existe