CAGE – Grupo de Investigación

SC3 – Super Cómputo y Cálculo Científico – UIS

@sc3uis

Introducción a Linux

La terminal como herramienta de trabajo

Módulo 2 – Semillero CAGE

Luis Alejandro Torres Niño | sc3.operaciones@uis.edu.co

Objetivos del módulo

- ➔ Perder el miedo a la **terminal**: es la única forma de trabajar en el cluster.
- ➔ Entender la **estructura de directorios** de Linux y moverse con soltura en ella.
- ➔ Dominar los comandos esenciales de archivos, directorios y permisos.
- ➔ Combinar comandos con *pipes* (`|`) para resolver tareas reales.
- ➔ Escribir tus primeros **scripts de Bash** (variables, condicionales, ciclos).
- ➔ Conectarte al cluster por SSH y mover archivos entre tu equipo y `guane`.

Agenda

1. ¿Qué es Linux? Distribuciones
2. Estructura de directorios
3. La terminal: prompt y navegación
4. Archivos: crear, ver, copiar, mover, borrar
5. Directorios y comandos de utilidad
6. Editores de texto: `nano` y `vim`
7. Permisos de archivos y directorios
8. Pipes, `grep` y `sort`
9. **Laboratorio**: retos individuales
10. Instalación de software y gestión en el cluster
11. Introducción a **Bash scripting**
12. SSH, `scp` y `rsync`
13. **Reto grupal** de cierre

01
¿Qué es Linux?

Linux y sus distribuciones

- Linux es un **núcleo (kernel)** de sistema operativo, libre y de código abierto.
- Distintas organizaciones lo empaquetan junto a herramientas, gestores de paquetes y entornos gráficos: eso es una **distribución** (*distro*).
- Ejemplos: Ubuntu, Debian, Fedora, Arch, CentOS/Rocky Linux.
- **Casi todos los clusteres HPC del mundo corren Linux**: es estable, ligero y totalmente scriptable.

★ ¿Y en Guane?

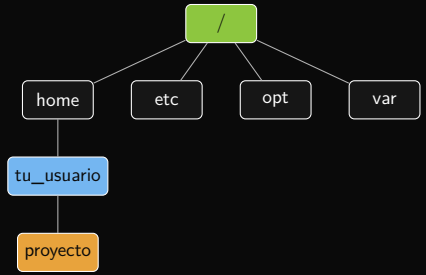
El frontend y los nodos de cómputo corren una distribución basada en **RHEL/Rocky Linux**, orientada a estabilidad y soporte de largo plazo – no a lucir bonita.

02
Estructura de directorios

El árbol de directorios

- Todo en Linux cuelga de una única raíz: `/`.
- No hay letras de unidad (`C:`, `D:`); los discos se **montan** dentro del árbol.
- Rutas **absolutas** empiezan en `/`; rutas **relativas** parten del directorio actual.

`/home/usuario`	tu directorio personal
`/etc`	configuración del sistema
`/tmp`	archivos temporales
`/opt`	software adicional (ej. `modules`)
`/var`	logs y datos variables



03
La terminal: prompt y navegación

El prompt

```
[user_name@guane ~]$
[root@guane /]#
```

- El símbolo $ indica que trabajas como **usuario regular**.
- El símbolo # indica que trabajas como **root** (superusuario) – en el cluster **nunca** tendrás este acceso, y está bien así.
- El texto antes de `@` es tu usuario; después, el nombre del host; `~` indica que estás en tu `home`.

Navegación básica

Comando	Qué hace
`pwd`	imprime el directorio actual (*print working directory*)
`cd ruta`	cambia al directorio `ruta`
`cd` o `cd ~`	vuelve a tu `home`
`cd /`	va a la raíz del sistema
`cd ..`	sube un nivel (directorio padre)
`cd -`	vuelve al directorio anterior

■ Mini-reto

Desde tu `home`, navega a `/tmp`, luego a `/`, y regresa a tu `home` usando `cd` sin argumentos. Verifica cada paso con `pwd`.

04

Archivos: crear, ver, copiar, mover, borrar

Listar archivos: `ls`

```
[user_name@guane ~]$ ls
datos.txt  proyecto  scripts

[user_name@guane ~]$ ls -al
drwxr-xr-x 5 usuario grupo 4096 Oct  5 10:12 .
drwxr-xr-x 8 root    root  4096 Sep  1 08:00 ..
-rw-r--r-- 1 usuario grupo  128 Oct  5 10:12 datos.txt
drwxr-xr-x 2 usuario grupo 4096 Oct  5 10:10 proyecto
drwxr-xr-x 2 usuario grupo 4096 Oct  5 10:11 scripts
```

Col.	Descripción
1	Tipo y permisos
2	*Hard links*
3 / 4	Propietario / grupo
5	Tamaño en bytes
6 / 7	Fecha y nombre

★ Tip

Crear y ver contenido de archivos

```
# Crear un archivo vacio
touch archivo.txt
# Crear y escribir directamente (Ctrl+D para cerrar)
cat > archivo.txt
  contenido del archivo...
  (Ctrl+D)
# Ver el contenido completo
cat archivo.txt
# Combinar dos archivos en uno nuevo
cat file1 file2 > nuevo_archivo
```

Mini-reto

Investiga la diferencia entre `cat`, `less`, `head` y `tail`. ¿Cuál usarías para ver solo las primeras 10 líneas de un log de 2GB sin abrirlo completo?

Copiar, mover y borrar

Comando	Qué hace
`cp origen destino`	copia un archivo
`cp -r dir1 dir2`	copia un directorio completo
`mv origen destino`	mueve **o renombra** un archivo
`rm archivo`	borra un archivo **sin confirmar**
`rm -r directorio`	borra un directorio y su contenido

❄ `rm` no tiene papelera de reciclaje

Un `rm -rf` mal dirigido (por ejemplo `rm -rf` o `rm -rf *` en la carpeta equivocada) borra todo tu `home`. **Siempre** revisa dos veces la ruta antes de un `rm -r`, especialmente con comodines (`*`).

■ Reto individual

Crea un archivo con `touch`, cópialo con un nuevo nombre, mueve la copia a `/tmp`, y luego bórrala. Verifica cada paso con `ls`.

05
Directorios y comandos de utilidad

Manipulación de directorios

```
mkdir directorio                    # crear un directorio
mkdir dir1 dir2 dir3                 # crear varios a la vez
mkdir -p dir1/subdir/subsubdir       # crear con padres intermedios
rmdir directorio                     # borrar SOLO si esta vacio
rm -r directorio                     # borrar con contenido
```

Reto individual

Crea un directorio `prueba`, adentro crea un archivo, e intenta borrar `prueba` con `rmdir`. ¿Qué mensaje aparece? ¿Por qué existe esa protección?

Comandos de utilidad

Comando	Qué hace
`man comando`	abre el manual de referencia
`history`	muestra el historial de comandos numerado
`!numero`	repite el comando `numero` del historial
`clear`	limpia la pantalla de la terminal
`whoami`	muestra tu usuario actual
`which comando`	muestra la ruta del ejecutable de un comando

★ Tip

`man` es tu mejor amigo: casi todo comando de Linux tiene su manual. Presiona `q` para salir.

06
Editores de texto en la terminal

nano vs vim

nano – simple, atajos visibles

```
nano archivo.slurm
```

- ➔ `Ctrl+O` guarda, `Ctrl+X` sale.
- ➔ Ideal para ediciones rápidas.

vim – modal, muy potente

```
vim archivo.slurm
```

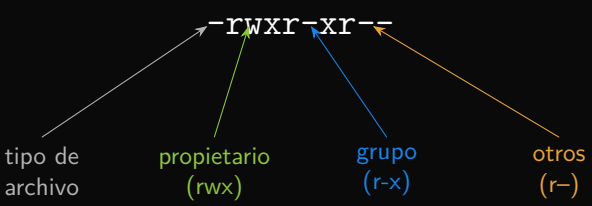
- ➔ `i` entra a modo inserción, `Esc` sale de él.
- ➔ `:wq` guarda y sale, `:q!` sale sin guardar.

★ Recomendación

Empieza con `nano` para editar tus scripts `.slurm`; con el tiempo aprende también `vim`. Ten presente que `vi` está disponible en prácticamente cualquier Linux, mientras que `nano` o `vim` completo pueden no estar instalados – y en un servidor remoto sin permisos de `sudo` no podrás instalarlos.

07
Permisos de archivos y directorios

Anatomía de los permisos



Permiso	Letra	Valor octal
Lectura (*read*)	r	4
Escritura (*write*)	w	2
Ejecución (*execute*)	x	1

Cambiar permisos: modo absoluto

Se suman los valores octales de cada permiso deseado por cada categoría (propietario, grupo, otros).

```
chmod 750 script.sh
# 7 = rwx (propietario: 4+2+1)
# 5 = r-x (grupo: 4+0+1)
# 0 = ---  (otros: sin permisos)

chmod 644 datos.txt    # rw-r--r--  (lectura/escritura dueno, solo lectura resto)
chmod +x script.sh     # agrega ejecucion sin tocar el resto (modo simbolico)
```

★ Modo simbólico

También puedes usar letras: `chmod u+x,g-w,o=r archivo` (u=usuario, g=grupo, o=otros; +/-/= agregan, quitan o fijan permisos).

Propietario y grupo: chown / chgrp

```
chown usuario archivo           # cambia solo el propietario
chown usuario:grupo archivo     # cambia propietario y grupo
chgrp grupo archivo             # cambia solo el grupo

groups                          # muestra los grupos a los que perteneces
```

★ Tip

Los grupos de Linux están definidos en `/etc/group`. En el cluster, tu grupo de investigación suele tener permisos compartidos sobre el directorio del proyecto.

■ Reto individual

Crea un archivo y asígnale: propietario con lectura/escritura/ejecución, grupo solo lectura, otros sin permisos. Verifica con `ls -l`. Luego cambia el permiso del grupo a lectura+escritura+ejecución.

08
Pipes, `grep` y `sort`

El operador *pipe* (|)

- ➔ Conecta la **salida** de un comando con la **entrada** del siguiente.
- ➔ Permite construir tuberías de procesamiento de texto sin archivos intermedios.




```
1 ls -l | grep '\.txt' | sort
```

Mini-reto

Crea un archivo con una lista de objetos (uno por línea) y ejecuta `cat archivo.txt | sort`.

grep – búsqueda de texto

grep busca líneas que coincidan con un patrón, en un archivo o en la salida de otro comando.

```
grep "error" logfile.txt          # lineas que contienen "error"
grep -i "error" logfile.txt       # busqueda sin distinguir mayus/minus
grep -r "TODO" ./proyecto         # busqueda recursiva en un directorio
grep -c "error" logfile.txt       # cuenta las coincidencias
cat logfile.txt | grep "error"    # el mismo patron, via pipe
```

■ Reto individual

Del archivo de objetos que creaste, usa grep para buscar uno de ellos. Luego cuenta cuántas líneas contienen una letra específica con grep -c.

`sort` – ordenar salidas

```
sort archivo.txt          # orden alfabetico ascendente
sort -r archivo.txt        # orden alfabetico inverso
sort -n numeros.txt        # orden NUMERICO (no alfabetico)
sort archivo.txt | uniq    # elimina lineas duplicadas consecutivas
```

Mini-reto

Ordena tu archivo de objetos alfabéticamente en orden inverso. Luego prueba `sort archivo.txt | uniq -c`: ¿qué información nueva te da la columna de la izquierda?

Laboratorio: retos individuales

Reto – Pipeline de texto

Individual

1. Descarga o crea un archivo `palabras.txt` con al menos 30 palabras (una por línea), con repeticiones.
2. Con un solo pipeline (`sort`, `uniq -c`, `sort -rn`, `head -5`) obtén las 5 palabras más frecuentes.
3. Guarda el resultado en `top5.txt` usando redirección (`>`).
4. Dale permisos de solo lectura a `top5.txt` para todos (`444`) y verifica con `ls -l`.

09
Instalación de software

Gestores de paquetes

- Cada distribución tiene su propio gestor de paquetes: `apt` (Debian/Ubuntu), `dnf`/`yum` (RHEL/Rocky/Fedora), `pacman` (Arch).
- En **tu propio Linux** (con permisos root):

```
sudo apt update && sudo apt upgrade   # actualizar el sistema
sudo apt search mailutils             # buscar un paquete
sudo apt install mailutils            # instalar
sudo apt remove mailutils             # desinstalar
```

❄ En el cluster no tienes `sudo`

No puedes instalar paquetes del sistema en guane. Para eso están los **módulos de ambiente** (Módulo 1) y los gestores de entornos como `conda` o `pip install --user`.

10
Introducción a Bash scripting

Tu primer script

```
#!/bin/bash
# Este es un comentario
echo "Hola desde $(hostname)"
echo "Hoy es $(date)"
```

```
chmod +x saludo.sh     # dale permiso de ejecucion
./saludo.sh             # ejecutalo (./ indica directorio actual)
```

★ `#!/bin/bash` – el *shebang*

Le dice al sistema qué intérprete usar para ejecutar el script. Siempre debe ser la primera línea.

Variables y argumentos

```bash
#!/bin/bash
NOMBRE="CAGE"
echo "Hola, $NOMBRE"

# Argumentos de linea de comandos
echo "Primer argumento: $1"
echo "Numero de argumentos: $#"
echo "Todos los argumentos: $@"

# Capturar la salida de un comando
FECHA=$(date +%Y-%m-%d)
echo "Hoy es $FECHA"
```

Mini-reto

Escribe un script `saludo.sh` que reciba tu nombre como argumento e imprima "Hola, <nombre>, hoy es <fecha>".

Condicionales

```bash
#!/bin/bash
if [ -f "datos.txt" ]; then
    echo "El archivo existe"
elif [ -d "datos.txt" ]; then
    echo "Es un directorio, no un archivo"
else
    echo "No existe"
fi

if [ "${1:-0}" -gt 10 ]; then
    echo "Mayor que 10"
fi
```

`-f archivo`	existe y es un archivo regular
`-d archivo`	existe y es un directorio
`-eq, -gt, -lt`	comparación numérica (igual, mayor, menor)

Ciclos: `for` y `while`

```bash
for i in 1 2 3 4 5; do
    echo "Iteracion $i"
done

for f in *.txt; do
    echo "Archivo: $f"
done
```

```bash
contador=0
while [ $contador -lt 5 ]; do
    echo "Contador: $contador"
    contador=$((contador + 1))
done
```

Reto individual

Escribe un script que reciba un número *N* como argumento y use un `for` para crear *N* directorios llamados `dato_1`, `dato_2`, ..., `dato_N`.

11
SSH, `scp` y `rsync`

Repaso: SSH al cluster

Ya lo hiciste en el Módulo 1 – aquí por qué funciona:

```
ssh user_name@toctoc.sc3.uis.edu.co    # 1. bastion (red publica)
ssh guane                               # 2. frontend (red interna UIS)
```

- ➔ **SSH** (*Secure Shell*) cifra toda la comunicación entre tu equipo y el servidor remoto.
- ➔ Sin esto, cualquiera en la red podría leer tu contraseña y tus comandos en texto plano.

Transferir archivos: scp y rsync

```
# 'guane' es el alias con ProxyJump del Modulo 1
# scp: copia simple, un archivo o carpeta (con -r)
scp archivo.txt guane:~/destino/
scp -r carpeta/ guane:~/destino/

# Descargar del cluster a tu equipo (sentido inverso)
scp guane:~/resultados.csv .

# rsync: sincroniza, solo copia lo que cambio (mas eficiente)
rsync -avz carpeta/ guane:~/destino/
```

★ ¿Por qué rsync y no scp para proyectos grandes?

rsync compara archivos y transfiere **solo las diferencias**: ideal para sincronizar código o resultados que cambian poco a poco.

Reto grupal de cierre

Reto grupal – Auditor de proyecto

★ Grupos de 2-3 personas

Escriban un script `auditar.sh` que reciba como argumento la ruta de un directorio (por defecto, el actual) y genere un reporte `reporte.txt` con:

1. Cantidad total de archivos y de subdirectorios (usen `ls`, `grep` y `wc -l`).
2. Las 3 extensiones de archivo más comunes (`.txt`, `.c`, `.slurm`, etc.) usando `grep`, `sort` y `uniq -c`.
3. El archivo más grande del directorio (pista: `ls -la` + `sort` numérico).
4. Una lista de archivos sin permiso de lectura para el grupo.

El script debe validar con un `if` que la ruta recibida exista antes de continuar.

Reto grupal – Entregables

Al final de la sesión, cada grupo debe mostrar:

- ➔ El script `auditar.sh` corriendo sobre su propio `home` en el cluster.
- ➔ El `reporte.txt` generado, con permisos `644`.
- ➔ Explicar qué pipeline (combinación de comandos con `|`) usaron para cada parte del reporte.
- ➔ **Bono**: subir el script a su `home` en `guane` con `scp` y ejecutarlo ahí vía SSH.

Cierre

★ Lo esencial de Linux

La terminal, los *pipes* y los scripts de Bash son tu herramienta diaria en el cluster: **automatiza lo repetitivo** y revisa dos veces antes de borrar.

¡Gracias!

sc3.operaciones@uis.edu.co