

# Manual De Comandos

Estudiante: Alvarez Poma Fabian Paulo

23 de febrero de 2026

## Índice

<b>1. Comandos Básicos (Navegación y Archivos)</b>	<b>2</b>
1.1. Gestión de Directorios . . . . .	2
1.2. Manipulación de Archivos . . . . .	2
1.3. Ayuda y Manuales . . . . .	3
<b>2. Comandos Avanzados (Sistema y Filtros)</b>	<b>3</b>
2.1. Ejemplos . . . . .	3
<b>3. Programación Shell (Bash Scripting)</b>	<b>5</b>
3.1. Estructura Básica . . . . .	5
3.2. Variables y Argumentos . . . . .	5
3.3. Estructuras de Control . . . . .	5
3.4. Script Funcional: Backup Automatizado . . . . .	5
<b>4. Guía de Diagnóstico (Errores Comunes)</b>	<b>6</b>
<b>5. Tuberías y Redireccionamientos</b>	<b>7</b>
5.1. Conceptos . . . . .	7
5.2. Operadores . . . . .	7
5.3. Ejemplos Obligatorios . . . . .	8

# 1. Comandos Básicos (Navegación y Archivos)

## 1.1. Gestión de Directorios

Comando	Descripción
ls	Lista el contenido de un directorio.
cd	Cambia de directorio.
mkdir	Crea un nuevo directorio.
pwd	Muestra la ruta actual.

### Ejemplos

#### ls

```
ls
ls -l
ls -a
```

#### cd

```
cd /home
cd ..
cd Documentos
```

#### mkdir

```
mkdir proyectos
mkdir -p cursos/linux
```

#### pwd

```
pwd
```

## 1.2. Manipulación de Archivos

Comando	Descripción
cp	Copia archivos o directorios.
mv	Mueve o renombra archivos.
rm	Elimina archivos o directorios.
touch	Crea un archivo vacío.
cat	Muestra el contenido de un archivo.

### Ejemplos

#### cp

```
cp archivo.txt copia.txt
cp -r carpeta respaldo/
```

#### mv

```
mv archivo.txt nuevo.txt
mv archivo.txt /tmp/
```

## rm

```
rm archivo.txt
rm -r carpeta
```

## touch

```
touch nuevo.txt
```

## cat

```
cat archivo.txt
cat archivo1.txt archivo2.txt
```

## 1.3. Ayuda y Manuales

Comando	Descripción
man	Muestra el manual de un comando.
help	Ayuda interna del shell.

```
man ls
help cd
```

## 2. Comandos Avanzados (Sistema y Filtros)

Comando	Descripción
chmod	Cambia permisos de archivos o directorios.
chown	Cambia el propietario o grupo de un archivo.
grep	Busca patrones de texto dentro de archivos.
find	Busca archivos o directorios en el sistema.
head	Muestra las primeras líneas de un archivo.
tail	Muestra las últimas líneas de un archivo.
sort	Ordena líneas de texto.
wc	Cuenta líneas, palabras y caracteres.
top	Muestra procesos en tiempo real.
ps	Lista procesos activos.
kill	Finaliza procesos por ID.

### 2.1. Ejemplos

#### chmod

```
chmod 755 script.sh
chmod +x programa.sh
chmod 644 archivo.txt
```

#### chown

```
chown usuario archivo.txt
chown usuario:grupo archivo.txt
sudo chown -R usuario carpeta/
```

### grep

```
grep "error" log.txt
grep -i "linux" archivo.txt
grep -r "config" /etc/
```

### find

```
find /home -name "*.txt"
find . -type d
find / -size +100M
```

### head

```
head archivo.txt
head -n 5 archivo.txt
head -c 20 archivo.txt
```

### tail

```
tail archivo.txt
tail -n 10 archivo.txt
tail -f log.txt
```

### sort

```
sort nombres.txt
sort -r numeros.txt
sort -n numeros.txt
```

### wc

```
wc archivo.txt
wc -l archivo.txt
wc -w archivo.txt
```

### top

```
top
top -u usuario
top -n 1
```

### ps

```
ps
ps aux
ps -ef
```

### kill

```
kill 1234
kill -9 1234
killall firefox
```

## 3. Programación Shell (Bash Scripting)

### 3.1. Estructura Básica

```
#!/bin/bash
echo "Hola Mundo"
```

### 3.2. Variables y Argumentos

```
#!/bin/bash
nombre=$1
echo "Hola $nombre"
```

### 3.3. Estructuras de Control

#### If-Else

```
if [ -f archivo.txt ]; then
    echo "Existe"
else
    echo "No existe"
fi
```

#### For

```
for archivo in *.txt; do
    echo $archivo
done
```

#### While

```
contador=1
while [ $contador -le 5 ]; do
    echo $contador
    contador=$((contador+1))
done
```

### 3.4. Script Funcional: Backup Automatizado

```
#!/bin/bash

# Variables
origen="/home/usuario/documentos"
destino="/home/usuario/respaldo"
fecha=$(date +%Y-%m-%d)

# Crear carpeta destino con fecha
mkdir -p "$destino/$fecha"

# Copiar archivos
```

```
cp -r "$origen/"* "$destino/$fecha/"

# Verificar si el backup fue exitoso
if [ $? -eq 0 ]; then
    echo "Backup completado correctamente en $destino/$fecha"
else
    echo "Error al realizar el backup"
fi
```

## 4. Guía de Diagnóstico (Errores Comunes)

En Linux, los errores son mensajes generados por el sistema cuando una operación no puede ejecutarse correctamente. A continuación se presentan errores comunes, cómo identificarlos y cómo solucionarlos.

### 1. Permission denied

**Mensaje típico:**

```
bash: ./script.sh: Permission denied
```

**Causa:** El usuario no tiene permisos de ejecución o acceso al archivo.

**Solución:**

```
chmod +x script.sh
sudo ./script.sh
```

**Explicación:** Linux controla el acceso mediante permisos (lectura, escritura y ejecución). Si no se tiene permiso, el sistema bloquea la acción.

### 2. Command not found

**Mensaje típico:**

```
comando_incorrecto: command not found
```

**Causa:** El comando está mal escrito o no está instalado.

**Solución:**

```
which nombre_comando
sudo apt install nombre_paquete
```

**Explicación:** El sistema busca el comando en las rutas definidas en la variable \$PATH.

### 3. No such file or directory

**Mensaje típico:**

```
ls: cannot access 'archivo.txt': No such file or directory
```

**Causa:** La ruta es incorrecta o el archivo no existe.

**Solución:**

```
ls
pwd
ls /ruta/completa/archivo.txt
```

**Explicación:** Linux distingue entre rutas relativas y absolutas. Si la ubicación es incorrecta, el sistema no encuentra el archivo.

## 4. Directory not empty

**Mensaje típico:**

```
rmdir: failed to remove 'carpeta': Directory not empty
```

**Causa:** Se intenta eliminar un directorio que contiene archivos.

**Solución:**

```
rm -r carpeta
```

**Explicación:** El comando rmdir solo elimina directorios vacíos.

## 5. Syntax error

**Mensaje típico:**

```
./script.sh: line 5: syntax error near unexpected token
```

**Causa:** Error en la estructura del script (espacios incorrectos, falta de fi, done, etc.).

**Solución:**

```
bash -n script.sh
```

**Explicación:** El comando bash -n permite verificar errores de sintaxis sin ejecutar el script.

## 5. Tuberías y Redireccionamientos

### 5.1. Conceptos

- stdin: Entrada estándar.
- stdout: Salida estándar.
- stderr: Salida de errores.

### 5.2. Operadores

- | : Envía la salida de un comando como entrada de otro (pipe).
- >: Redirige la salida estándar y sobrescribe el archivo.
- >>: Redirige la salida estándar y añade al final del archivo.
- 2>: Redirige la salida de errores (stderr) a un archivo.

## 5.3. Ejemplos Obligatorios

### 1. Filtro combinado

```
ls | grep archivo.txt
```

### 2. Conteo y ordenamiento

```
cat texto.txt | sort | uniq | wc -l
```

### 3. Registro de errores

```
ls /directorio_inexistente 2> errores.log
```

## Declaración de uso de inteligencia artificial

El presente trabajo fue desarrollado con el apoyo de la herramienta de inteligencia artificial **ChatGPT**, la cual fue utilizada como recurso complementario para la orientación en la redacción y estructuración del contenido. El autor es responsable de la selección, verificación y adecuación final de la información presentada, garantizando su originalidad y validez.