cd ~ # vaya a su directorio de trabajo

Compruebe que está en su directorio de trabajo

pwd # Muestra la ruta del directorio actual

whoami # Muestra el usuario conectado

echo $USER # Muestra el usuario conectado

echo $LOGNAME # Muestra el usuario conectado

cat /etc/passwd # Muestra la configuración de los usuarios. Buscar la línea del usuario conectado y comprobar que el penúltimo campo coincide con lo mostrado por el comando pwd anterior.

ls ../../bin/x???* # Con trayectoria relativa, liste los archivos de "/bin" que comiencen por "x" y que tengan 3 o más caracteres

ls ../../bin/[aeiou]*[aeiou] ../../bin/[aeiou] # Con trayectoria relativa, liste los archivos de "/bin" que comiencen y terminen por vocal

mkdir ejercicio # En su directorio de trabajo cree un directorio llamado ejercicio

cp ../../etc/group ejercicio/grupos # Copie con trayectorio relativa el archivo "/etc/group" en el directorio "ejercicio" con el nombre "grupos"

cd ejercicio # Muévase al directorio ejercicio

mkdir -p {2000..2100}/{01..12} # Cree un directorio para cada año entre el 2000 y el 2100 y que cada año tenga un subdirectorio para cada mes del 01 al 12

touch {2000..2100}/{01..12}/{01..31} # Cree un archivo del 01 al 31 que corresponde a los días de cada mes

rm {2000..2100}/{0{2,4,6,9},11}/31 # Borre los días 31 de los meses que no tengan 31 días

touch {2000,2004,2008,2010,..... completar....,2096}/02/29 # Añada los 29 de febrero

ls */*/31 # Mostrar todos los días 31 de todos los meses de 31 días de todos los años

¿Cuántas veces se repite la inicial del nombre de los usuarios que más común?

cat /etc/passwd | cut -c1 | sort | uniq -c | tr -s ' ' : | sort -t: -k2nr | head -1 | cut -f2 -d:

cat emails | grep -Eo "^[a-z0-9]+(\.+[a-z0-9]+)*@[a-z0-9]+(\.+[a-z0-9]+)*(\.+[a-z0-9]{2,})"

NOSE DE DONDE SON ////

cat /etc/passwd | cut -d: -f1-3 | sort -n | grep :[0-9][0-9]$ |cut -d: -f1

cat /etc/passwd | cut -f1,3,4 -d: | grep -Ev "^[^:]+:([^:]+):\1"

////

Mostrar cuántos caracteres tiene cada usuario

```
cat /etc/passwd | cut -f1 -d: | xargs -I{} bash -c "echo -n {}: ; echo {} | wc -c"
```

Nombre de usuario más largo:

```
cat /etc/passwd | cut -f1 -d: | xargs -Iuser bash -c 'echo -n user: ; echo -n user | wc -c' | sort -n -k2 -t: | tail -1 | cut -f1 -d:
```

```
cat users | cut -f1 -d: | xargs -Iuser bash -c 'echo -n user: ; echo -n user | wc -c' | grep ":$(cat users | cut -f1 -d: | xargs -Iuser bash -c 'echo -n user: ; echo -n user | wc -c' | sort -n -k2 -t: | tail -1 | cut -f2 -d:)$" | cut -f1 -d:
```

Sumar el tamaño de los ficheros de /etc/

```
echo $(ls -l /etc/ | tr -s ' ' | cut -f5 -d' ') | tr ' ' + | bc
```

Lo mismo pero sólo de ficheros regulares:

```
echo $(ls -la /etc/ | grep "^-" | tr -s ' ' | cut -f5 -d' ') | tr ' ' + | bc
```

```
echo $(find /etc -maxdepth 1 -type f -printf %s+)0 | bc
```

```
userlengths=$(cat users | cut -f1 -d: | xargs -Iuser bash -c 'echo -n user: ; echo -n user | wc -c')

maxlength=$(echo -e "$userlengths" | cut -f2 -d: | sort -n | tail -1)


echo -e "$userlengths" | grep ":$maxlength$" | cut -f1 -d:
```

NOSE DE DONDE SON /////

```
cat /etc/passwd | cut -d: -f1 | xargs -I{} bash -c 'groups {}'


paste -d: <(cut -d: -f1 /etc/passwd) <(id -Gn $(cut -d: -f1 /etc/passwd) | tr ' ' ,)


cut -d: -f1 /etc/passwd | xargs -I{} bash -c 'echo -n {}: ; echo $((cat /etc/group | grep -E "^[^:]+:[^:]:$(cat /etc/passwd | grep "^{}:" | cut -f3 -d:)" | cut -f1 -d: ; cat /etc/group | grep -E ":{}$|:{},|,{},|,{}$" | cut -f1 -d:) | sort | uniq) | tr " " ,'
```

///////

@gmail.com

.pepe@gmail.com

pepegmail.com

pepe@gmailcom

pepe.@gmail.com

pepe@.gmail.com

pepe@gmail@uk.com

pepe.perez@gmail.c

pepe@gmail@uk.

```
#!/bin/bash

cat /etc/passwd | cut -f1 -d: |

   while read user ; do

      first=$(echo $user | cut -c1)

      last=$(echo $user | rev | cut -c1)

      if [ $first = $last ] ; then

         echo $user

      fi

   done

cat /etc/passwd | cut -f1 -d: | xargs -I{} bash -c 'echo {} | grep -xE "(.).*\1"'
```

@gmail.com

.pepe@gmail.com

pepegmail.com

pepe@gmailcom

pepe.@gmail.com

pepe@.gmail.com

pepe@gmail@uk.com

pepe.perez@gmail.c

pepe@gmail@uk.

```
#!/bin/bash

cat /etc/passwd | cut -f1 -d: |

    while read user ; do

        first=$(echo $user | cut -c1)

        last=$(echo $user | rev | cut -c1)

        if [ $first = $last ] ; then

            echo $user

        fi

    done

cat /etc/passwd | cut -f1 -d: | xargs -I{} bash -c 'echo {} | grep -xE "(.).*\1"'
```

```
#!/bin/bash

function check_format {
    echo $1 | grep -Exo "[0-9]{1,2}/[0-9]{1,2}/([0-9]{2}|[0-9]{4})"
}

function map {
    command=$1
    while read x ; do
        $command $x
    done
}

function get_params {
    for param in $@ ; do
        echo $param
    done
}

function check_date {
    day=$1
    month=$2
    year=$3

    if [ $day -gt 0 ] && [ $day -le 31 ] &&
       [ $month -gt 0 ] && [ $month -le 12 ]; then
        if [ $day -le 28 ]; then
            echo $1 $2 $3
```

```
        elif [ $day -eq 31 ] &&
        ([ $month -eq 1 ] || [ $month -eq 3 ] || [ $month -eq 5 ] ||
        [ $month -eq 7 ] || [ $month -eq 8 ] || [ $month -eq 10 ] ||
        [ $month -eq 12 ]); then
            echo $1 $2 $3
        elif [ $day -eq 30 ] && [ $month -ne 2 ]; then
            echo $1 $2 $3
        elif [ $day -eq 29 ] && [ $month -eq 2 ]; then
            if [ $day -eq 29 ] && is_leap $year; then
                echo $1 $2 $3
            fi
        fi
    fi
}

function filter_bigger {
    sysdate=$(date +%F | tr -d '-')
    day=$(echo $1 + 0 | bc)
    month=$(echo $2 + 0 | bc)
    year=$3
    if [ $year -lt 100 ]; then
        year=20$year
    fi
    if [ $day -lt 10 ]; then
        day=0$day
    fi
    if [ $month -lt 10 ]; then
        month=0$month
```

```
    fi
    fecha=$year$month$day

    if [ $fecha -gt $sysdate ]; then
        echo $day/$month/$year
    fi
}

function is_leap {
    year=$1
    if [ $(echo $year % 4 | bc) -eq 0 ] && [ $(echo $year % 100 | bc) -ne 0 ] ||
    [ $(echo $year % 400 | bc) -eq 0 ]; then
        return 0
    else
        return 1
    fi
}

get_params $@ | map check_format | tr '/' ' ' | map check_date | map
filter_bigger


#!/bin/bash

function check_format {
    if (echo $* | grep -xvE "[0-9]{2}:[0-9]{2}:[0-9]{2}" >/dev/null) ||
        [ $(get_hour $1) -gt 23 ] ||
        [ $(get_min $1) -gt 59 ] ||
        [ $(get_sec $1) -gt 59 ] ; then
        return 1
```

```
    else
        return 0
    fi
}

function check_ge {
    time2=$1
    time1=$2
    check_format $time1 &&
    check_format $time2 &&
    [ $(transform_seconds $time1) -ge $(transform_seconds $time2) ]
}

function get_hour {
    echo ${1%%:*}
}

function get_min {
    echo $1 | cut -d: -f2
}

function get_sec {
    echo ${1##*:}
}

function transform_seconds {
    echo $(get_hour $1) '*' 3600 + $(get_min $1) '*' 60 + $(get_sec $1) | bc
}
```

```
function add_zero {
    if [ $1 -lt 10 ]; then
        echo -n 0$1
    else
        echo -n $1
    fi
}

function pluralize {
    number=$1
    word=$2
    echo -n $number $word
    if [ $number -ne 1 ]; then
        echo -n "s"
    fi
}

function format_seconds {
    days=$(($1 / 86400))
    hh=$((($1 - days * 86400) / 3600))
    mm=$((($1 - hh * 3600 - days * 86400)/60))
    ss=$(($1 - mm * 60 - hh * 3600 - days * 86400))
    if [ $days -gt 0 ]; then
        echo -n "$(pluralize $days día), "
    fi
    echo "$(pluralize $hh hora), $(pluralize $mm minuto) y $(pluralize $ss
segundo)"
}
```

```
function add {
    sum=0
    while read x ; do
        sum=$(($sum+x))
    done
    echo $sum
}

function filter {
    command=$*
    while read x ; do
        if $command $x ; then
            echo $x
        fi
    done
}

function map {
    command=$*
    while read x ; do
        $command $x
    done
}

function get_params {
    for param in $@ ; do
        echo $param
```

```
    done

    if [ ! -t 0 ] ; then # está abierta la entrada estándar por tubería
        cat
    fi
}

get_params $@                |
    filter check_ge 12:00:00  |
    map transform_seconds     |
    add                       |
    map format_seconds



#!/bin/bash

function check_format {
    if (echo $* | grep -xvE "[0-9]{2}:[0-9]{2}:[0-9]{2}" >/dev/null) ||
        [ $(get_hour $1) -gt 23 ] ||
        [ $(get_min $1) -gt 59 ] ||
        [ $(get_sec $1) -gt 59 ] ||
        [ $(get_hour $1) -lt 12 ]; then
        return 1
    else
        return 0
    fi
}
```

```
function get_hour {
    echo ${1%%:*}
}

function get_min {
    echo $1 | cut -d: -f2
}

function get_sec {
    echo ${1##*:}
}

function transform_seconds {
    echo $(get_hour $1) * 3600 + $(get_min $1) * 60 + $(get_sec $1) | bc
}

function add_zero {
    if [ $1 -lt 10 ]; then
        echo -n 0$1
    else
        echo -n $1
    fi
}

function check_plural {
    number=$1
    word=$2
    echo -n $number $word
```

```
    if [ $number -ne 1 ]; then
        echo -n "s"
    fi
}

function format_seconds {
    days=$(($1 / 86400))
    hh=$((($1 - days * 86400) / 3600))
    mm=$((($1 - hh * 3600 - days * 86400)/60))
    ss=$(($1 - mm * 60 - hh * 3600 - days * 86400))
    if [ $days -gt 0 ]; then
        echo -n "$(check_plural $days día), "
    fi
    echo "$(check_plural $hh hora), $(check_plural $mm minuto) y
$(check_plural $ss segundo)"
}

function add {
    sum=0
    while read x ; do
        sum=$(($sum+x))
    done
    echo $sum
}

function filter {
    command=$1
    while read x ; do
        if $command $x ; then
```

```
            echo $x
        fi
    done
}

function map {
    command=$1
    while read x ; do
        $command $x
    done
}

function get_params {
    for param in $@ ; do
        echo $param
    done

    if [ ! -t 0 ] ; then # está abierta la entrada estándar por tubería
        cat
    fi
}

get_params $@               |
    filter check_format     |
    map transform_seconds   |
    add                     |
    map format_seconds
  #!/bin/bash
```

```
function less_than_255 {
    read field
    if [ "$field" -le 255 ] ; then
       return 0
    else
       return 1
    fi
}

function filter {
    command=$1
    while read x ; do
        if $command $x ; then
            echo $x
        fi
    done
}

function map {
    command=$1
    while read x ; do
        $command $x
    done
}


function check_field {
```

```
    field=$1
    echo $field | less_than_255
}

function check_ip {
    ip=$1
    field_numbers=$(echo $ip | grep -oE "[0-9]+" | filter check_field | wc -l)
    if [ $field_numbers -ne 4 ] ; then
      return 1
    fi
}

function format_field {
    field=$1
    echo $field+0 | bc
}

function format_ip {
    ip=$1
    echo $(echo $ip | grep -oE "[0-9]+" | map format_field) | tr ' ' .
}

grep -xE "[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+" | filter check_ip | map format_ip

12.45.255.127
12.45.255
12.45.255.
12.45.255.256
```

12.45.-25.254

12.45.34.25.254

12.45.34.320

12.A.23.23

12.12.12/12

12.45.255. 127

13.13.13.13

.45.255.12

014.00014.14.14

1.09.2.3

3.0.000.4

00.000.0.0

```
#!/bin/bash

function check_format {
    if (echo $* | grep -xvE "[0-9]{2}:[0-9]{2}:[0-9]{2}" >/dev/null) ||
        [ $(get_hour $1) -gt 23 ] ||
        [ $(get_min $1) -gt 59 ] ||
        [ $(get_sec $1) -gt 59 ] ; then
        return 1
    else
        return 0
    fi
}

function get_hour {
    echo ${1%%:*}
```

```
}

function get_min {
    echo $1 | cut -d: -f2
}

function get_sec {
    echo ${1##*:}
}

function transform_seconds {
    echo $(($(get_hour $1) * 3600 +
    $(get_min $1) * 60 + $(get_sec $1)))
}

function add_zero {
    if [ $1 -lt 10 ]; then
        echo -n 0$1
    else
        echo -n $1
    fi
}

function format_seconds {
    hh=$(($1 / 3600))
    mm=$((($1 - hh * 3600)/60))
    ss=$(($1 - mm * 60 - hh * 3600))
    echo "$(add_zero $hh):$(add_zero $mm):$(add_zero $ss)"
```

```
}

function first_last {
    read first
    [ -z "$first" ] && first=00:00:00

    while read next ; do
        last=$next
    done
    [ -z "$last" ] && last=$first

    echo $first
    echo $last
}

function filter {
    command=$1
    while read x ; do
        if $command $x ; then
            echo $x
        fi
    done
}

function map {
    command=$1
    while read x ; do
        $command $x
```

```
    done
}

function diff_time {
    read first
    read last
    echo $(($last-first))
}

function get_params {
    for param in $@ ; do
        echo $param
    done

    if [ ! -t 0 ] ; then # está abierta la entrada estándar por tubería
        cat
    fi
}

get_params $@               |
    filter check_format     |
    sort                    |
    first_last              |
    map transform_seconds   |
    diff_time               |
    map format_seconds
```

```
#!/bin/bash

function check_format {
    if (echo $* | grep -xvE "[0-9]{2}:[0-9]{2}:[0-9]{2}" >/dev/null) ||
        [ $(get_hour $1) -gt 23 ] ||
        [ $(get_min $1) -gt 59 ] ||
        [ $(get_sec $1) -gt 59 ] ; then
        return 1
    else
        return 0
    fi
}

function get_hour {
    echo ${1%%:*}
}

function get_min {
    echo $1 | cut -d: -f2
}

function get_sec {
    echo ${1##*:}
}

function transform_seconds {
    echo $(($(get_hour $1) * 3600 +
```

```
    $(get_min $1) * 60 + $(get_sec $1)))
}

function add_zero {
    if [ $1 -lt 10 ]; then
        echo -n 0$1
    else
        echo -n $1
    fi
}

function format_seconds {
    hh=$(($1 / 3600))
    mm=$((($1 - hh * 3600)/60))
    ss=$(($1 - mm * 60 - hh * 3600))
    echo "$(add_zero $hh):$(add_zero $mm):$(add_zero $ss)"
}

function valid_time {
    while read time ; do
        if check_format $time ; then
            echo $time
        fi
    done
}

function first_last {
    read first
```

```
    [ -z "$first" ] && first=00:00:00

    while read next ; do
        last=$next
    done
    [ -z "$last" ] && last=$first

    echo $first
    echo $last
}

function to_seconds {
    while read time ; do
        transform_seconds $time
    done
}

function from_seconds {
    while read time ; do
        format_seconds $time
    done
}

function diff_time {
    read first
    read last
    echo $(($last-first))
}
```

```
function get_params {
    if [ $# -gt 0 ] ; then
        echo $@ | tr ' ' '\n'
    fi
    if [ -p /dev/stdin ] ; then # está abierta la entrada estándar
        while read time ; do
            echo $time
        done
    fi
}

get_params $@ | valid_time | sort | first_last | to_seconds | diff_time |
from_seconds
```

```
#!/bin/bash

function check_format {
    if (echo $1 | grep -xvE "[0-9]{2}:[0-9]{2}:[0-9]{2}" >/dev/null) ||
        [ $(get_hour $1) -gt 23 ] ||
        [ $(get_min $1) -gt 59 ] ||
        [ $(get_sec $1) -gt 59 ] ; then
        return 1
    else
```

```
        return 0
    fi
}

function get_hour {
    echo ${1%%:*}
}

function get_min {
    echo $1 | cut -d: -f2
}

function get_sec {
    echo ${1##*:}
}

function transform_seconds {
    echo $(($(get_hour $1) * 3600 +
    $(get_min $1) * 60 + $(get_sec $1)))
}

function add_zero {
    if [ $1 -lt 10 ]; then
        echo -n 0$1
    else
        echo -n $1
    fi
}
```

```
function format_seconds {
    hh=$(($1 / 3600))
    mm=$((($1 - hh * 3600)/60))
    ss=$(($1 - mm * 60 - hh * 3600))
    echo "$(add_zero $hh):$(add_zero $mm):$(add_zero $ss)"
}

times=$(
    for time in $@; do
        if check_format $time ; then
           echo $time
        fi
    done | sort -n -t: -k1 -k2 -k3
)

earliest=$(echo -e "$times" | head -1)
latest=$(echo -e "$times" | tail -1)

secondsbetween=$(($(transform_seconds $latest) - $(transform_seconds $earliest)))
echo "$latest - $earliest = $(format_seconds $secondsbetween)"
```

```
#!/bin/bash

edad=$1
mayoria_edad=18
```

```
if [ $# -ne 1 ] || echo $edad | grep -xvE "[0-9]+" > /dev/null ; then
    >&2 echo "uso: kk edad (entero)"
    exit 1
fi



if [ $edad -lt $mayoria_edad ] ; then
    echo Eres menor de edad
else
    echo Eres mayor de edad
fi



root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin

nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin

systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin

systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin

messagebus:x:102:105::/nonexistent:/usr/sbin/nologin

systemd-timesync:x:103:106:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin

syslog:x:104:111::/home/syslog:/usr/sbin/nologin

_apt:x:105:65534::/nonexistent:/usr/sbin/nologin

tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false

rtkit:x:107:113:RealtimeKit,,,:/proc:/usr/sbin/nologin

systemd-coredump:x:108:114:systemd Core Dumper,,,:/run/systemd:/usr/sbin/nologin

kernoops:x:109:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin

uuidd:x:110:119::/run/uuidd:/usr/sbin/nologin

cups-pk-helper:x:111:115:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin

lightdm:x:112:120:Light Display Manager:/var/lib/lightdm:/bin/false

tcpdump:x:113:122::/nonexistent:/usr/sbin/nologin

avahi-autoipd:x:115:125:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin

usbmux:x:116:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin

nm-openvpn:x:117:126:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin

geoclue:x:118:127::/var/lib/geoclue:/usr/sbin/nologin

dnsmasq:x:119:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin

pulse:x:120:128:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin

_flatpak:x:121:131:Flatpak system-wide installation helper,,,:/nonexistent:/usr/sbin/nologin

avahi:x:122:132:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin

saned:x:123:133::/var/lib/saned:/usr/sbin/nologin

colord:x:124:134:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin

fwupd-refresh:x:125:135:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin

hplip:x:126:7:HPLIP system user,,,:/run/hplip:/bin/false

usuario:x:1000:1000:usuario,,,:/home/usuario:/bin/bash

sssd:x:127:137:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin

surtich:x:1001:1001:surtich,,,:/home/surtich:/bin/bash

p@gmail.com

pepe@gmail.com

pepe.perez@gmail.com

pepe.perez@gmail.uk.com

pepe88@gmail.com

pepe88.perez@gmail.com

pepe.perez.luis.lopez@gmail.com

88@gmail.com

pepe@gmail.c.com

pepe@gmail.c.uk.com

¡Ahora lo tengo claro! Aquí te paso la tabla completa con los códigos. Es una tabla organizada donde puedes ver el comando, la función y el código correspondiente.

Comando/Función	Descripción	Código
Verificar primer y último carácter de un usuario	Filtra usuarios con el mismo primer y último carácter.	```bash #!/bin/bash cat /etc/passwd
Verificar formato de fecha	Verifica si una fecha está en el formato `dd/mm/yyyy`.	`bash #!/bin/bash function check_format { if [[ "$1" =~ ^[0-9]{2}/[0-9]{2}/[0-9]{4}$ ]]; then echo "Fecha válida: $1" else echo "Fecha no válida: $1" fi } # Ejemplo de uso check_format "12/05/2025" check_format "2025-05-12"`
Verificar formato de hora	Verifica si una hora está en el formato `hh:mm:ss`.	`bash #!/bin/bash function check_format { if [[ "$1" =~ ^[0-9]{2}:[0-9]{2}:[0-9]{2}$ ]]; then echo "Hora válida: $1" else echo "Hora no válida: $1" fi } check_format "12:30:45"`

		`check_format "12-30-45"`
Sumar segundos de una hora	Convierte horas a segundos y realiza la suma.	```bash #!/bin/bash function transform_seconds { echo $(($(get_hour $1) * 3600 + $(get_min $1) * 60 + $(get_sec $1))) } function get_hour { echo ${1%%:*} } function get_min { echo $1
Filtrar por horas mayores que una dada	Filtra las horas que son mayores a una hora especificada.	`bash #!/bin/bash function check_ge { time1=$1 time2=$2 if [[ $(transform_seconds $time1) -ge $(transform_seconds $time2) ]]; then echo "$time1 es mayor o igual a $time2" fi } check_ge "12:30:45" "10:30:45"`
Sumar horas en formato `hh:mm:ss`	Suma horas en formato `hh:mm:ss` y retorna el resultado.	`bash #!/bin/bash function add_zero { if [ $1 -lt 10 ]; then echo -n 0$1 else echo -n $1 fi } function format_seconds { hh=$(($1 / 3600)) mm=$((($1 - hh * 3600)/60)) ss=$(($1 - mm * 60 - hh * 3600))`

		`echo "$(add_zero $hh):$(add_zero $mm):$(add_zero $ss)" } function sum_hours { total_seconds=0 for hour in $@; do total_seconds=$(($total_seconds + $(transform_seconds $hour))) done format_seconds $total_seconds } sum_hours "12:30:45" "1:30:30"`
Verificar si una dirección IP es válida	Verifica si una dirección IP es válida, considerando los valores de los campos.	```bash #!/bin/bash function check_ip { ip=$1 field_numbers=$(echo $ip
Formato de dirección IP	Formatea las direcciones IP para asegurar que los campos estén en el rango adecuado.	```bash #!/bin/bash function format_ip { ip=$1 echo $(echo $ip
Suma de edades	Calcula si una persona es mayor o menor de edad.	```bash #!/bin/bash edad=$1 mayoria_edad=18 if [$# -ne 1]

Funciones y Expresiones Regulares:

Función/Expresión	Descripción	Código/Expresión	Ejemplo de uso
`sumar_dias()`	Sumar días a una fecha	`bash sumar_dias() { fecha=$1; dias=$2; date -d "$fecha + $dias days" +%Y-%m-%d; }`	`sumar_dias "2024-02-20" 5`
`restar_dias()`	Restar días a una fecha	`bash restar_dias() { fecha=$1; dias=$2; date -d "$fecha - $dias days" +%Y-%m-%d; }`	`restar_dias "2024-02-20" 5`
`diferencia_dias()`	Diferencia de días entre dos fechas	`bash diferencia_dias() { fecha1=$1; fecha2=$2; echo $(( ( $(date -d "$fecha2" +%s) - $(date -d "$fecha1" +%s) ) / 86400 )); }`	`diferencia_dias "2024-02-15" "2024-02-20"`
`validar_fecha()`	Validar fecha en formato YYYY-MM-DD	`bash validar_fecha() { if [[ "$1" =~`	`validar_fecha "2024-02-30"`

		`^[0-9]{4}-[0-9]{2}-[0-9]{2}$ ]]; then echo "Fecha válida"; else echo "Fecha inválida"; fi }`	
`validar_ip()`	Validar IP en formato estándar	`bash validar_ip() { if [[ "$1" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}$ ]]; then echo "IP válida"; else echo "IP inválida"; fi }`	`validar_ip "192.168.1.1"`
`validar_ip_completa()`	Validar IP con valores entre 0 y 255	```bash validar_ip_completa() { if [["$1" =~ ^((25[0-5]	2[0-4][0-9]
`sumar_valores()`	Sumar valores de un archivo con números	`bash sumar_valores() { awk '{s+=$1} END {print s}' $1; }`	`sumar_valores archivo.txt`
`contar_lineas()`	Contar el número de líneas de un archivo	`bash contar_lineas() { wc -l < $1; }`	`contar_lineas archivo.txt`

`eliminar_espacios()`	Eliminar espacios al inicio y al final de cada línea	`bash eliminar_espacios() { sed 's/^[ \t]*//;s/[ \t]*$//' $1; }`	`eliminar_espacios archivo.txt`
`reemplazar_texto()`	Reemplazar texto en un archivo	`bash reemplazar_texto() { sed 's/foo/bar/g' $1; }`	`reemplazar_texto archivo.txt`
`filtrar_archivo()`	Filtrar líneas de un archivo que contengan un patrón	`bash filtrar_archivo() { grep "$1" $2; }`	`filtrar_archivo "foo" archivo.txt`
`eliminar_lineas_duplicadas()`	Eliminar líneas duplicadas en un archivo	```bash eliminar_lineas_duplicadas() { sort $1	`uniq; } ````
`convertir_minusculas()`	Convertir texto a minúsculas	`bash convertir_minusculas() { tr '[:upper:]' '[:lower:]' < $1; }`	`convertir_minusculas archivo.txt`
`convertir_mayusculas()`	Convertir texto a mayúsculas	`bash convertir_mayusculas() { tr '[:lower:]' '[:upper:]' < $1; }`	`convertir_mayusculas archivo.txt`

`extraer_columna()`	Extraer una columna específica de un archivo	`bash extraer_columna() { awk -v col=$1 '{print $col}' $2; }`	`extraer_columna 2 archivo.txt`
`buscar_lineas()`	Buscar líneas que no contengan un patrón	`bash buscar_lineas() { grep -v "$1" $2; }`	`buscar_lineas "foo" archivo.txt`
`contar_palabras()`	Contar las palabras de un archivo	`bash contar_palabras() { wc -w < $1; }`	`contar_palabras archivo.txt`
`extraer_datos()`	Extraer datos con una expresión regular	`bash extraer_datos() { grep -oP "$1" $2; }`	`extraer_datos "\d+" archivo.txt`
`buscar_palabra()`	Buscar palabra específica en archivo	`bash buscar_palabra() { grep -w "$1" $2; }`	`buscar_palabra "error" archivo.txt`
`contar_lineas_con_patrón()`	Contar las líneas que contienen un patrón	`bash contar_lineas_con_patrón() { grep -c "$1" $2; }`	`contar_lineas_con_patrón "foo" archivo.txt`
`obtener_ip()`	Extraer dirección IP de un texto	`bash obtener_ip() { grep -oP "(\d{1,3}\.){3`	`obtener_ip archivo.txt`

		`}\d{1,3}" $1; }`	
`obtener_fecha()`	Extraer fechas en formato YYYY-MM-DD	`bash obtener_fecha() { grep -oP "\d{4}-\d{2}-\d{2}" $1; }`	`obtener_fecha archivo.txt`
`validar_email()`	Validar formato de email	`bash validar_email() { if [[ "$1" =~ ^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$ ]]; then echo "Email válido"; else echo "Email inválido"; fi }`	`validar_email "test@example.com"`
`validar_url()`	Validar formato de URL	`bash validar_url() { if [[ "$1" =~ ^https?://[a-zA-Z0-9.-]+(\.[a-zA-Z]{2,})+$ ]]; then echo "URL válida"; else echo "URL inválida"; fi }`	`validar_url "https://example.com"`

`remplazar_espacios()`	Eliminar espacios en un archivo	`bash remplazar_espacios() { sed 's/ //g' $1; }`	`remplazar_espacios archivo.txt`
`extraer_extension()`	Extraer la extensión de un archivo	```bash extraer_extension() { echo "$1"	rev

---

Sección	Descripción	Comando
Verificar directorio de trabajo	Muestra la ruta del directorio actual	`pwd`
Usuario actual	Muestra el usuario conectado	`whoami`
Usuario desde variable	Muestra el usuario actual usando variables de entorno	`echo $USER` o `echo $LOGNAME`
Listar archivos con patrones	Archivos en `/bin` que comiencen con 'x' y tengan al menos 3 caracteres	`ls ../../bin/x???*`
Listar archivos con vocales	Archivos en `/bin` que comiencen y terminen en vocal	`ls ../../bin/[aeiou]*[aeiou] ../../bin/[aeiou]`
Crear directorio	Crea un directorio llamado 'ejercicio'	`mkdir ejercicio`
Copiar archivo	Copia `/etc/group` en 'ejercicio'	`cp ../../etc/group ejercicio/grupos`
Crear subdirectorios	Crea subdirectorios por año y mes	`mkdir -p {2000..2100}/{01..12}`
Crear archivos por día	Genera archivos del 01 al 31 en cada mes y año	`touch {2000..2100}/{01..12}/{01..31}`
Eliminar días inexistentes	Elimina los días 31 de los meses sin 31 días	`rm {2000..2100}/{0{2,4,6,9},11}/31`

Agregar días bisiestos	Agrega el 29 de febrero a los años bisiestos	`touch {2000,2004,2008,2012,....,2096}/02/29`
Mostrar días 31	Lista todos los días 31 en todos los meses y años	`ls */*/31`
Contar iniciales de usuario	Cuenta cuántas veces se repite la inicial del nombre de usuario	`cat /etc/passwd
Validar correos	Extrae correos válidos de un archivo	`cat emails
Filtrar usuarios por UID	Lista usuarios con ID mayor que 9	`cat /etc/passwd
Usuarios con mismo UID	Filtra usuarios que tengan el mismo UID y GID	`cat /etc/passwd
Longitud de nombre de usuario	Muestra la cantidad de caracteres en cada nombre de usuario	`cat /etc/passwd
Usuario con nombre más largo	Encuentra el nombre de usuario más largo	`cat /etc/passwd
Tamaño total de archivos en /etc	Suma el tamaño de todos los ficheros en /etc	`echo $(ls -l /etc/
Tamaño de ficheros	Suma el tamaño de archivos regulares	`echo $(ls -la /etc/

regulares en /etc		
Tamaño con find	Calcula el tamaño de ficheros regulares en `/etc/` con `find`	`echo $(find /etc -maxdepth 1 -type f -printf %s+)0
Mostrar grupos de usuario	Lista los grupos a los que pertenece cada usuario	`cut -d: -f1 /etc/passwd
Validar direcciones IP	Verifica que una IP tenga formato correcto	`grep -xE "[0-9]+.[0-9]+.[0-9]+.[0-9]+"

Esta tabla resume los ejercicios y comandos esenciales del documento. Si necesitas otro formato o más detalles, dime.

---

1. Verificar el directorio de trabajo

cd ~ # Ir al directorio de trabajo del usuario

pwd # Mostrar la ruta del directorio actual

whoami # Mostrar el nombre del usuario conectado

echo $USER # Mostrar el nombre del usuario conectado

echo $LOGNAME # Mostrar el nombre del usuario conectado

cat /etc/passwd # Mostrar la configuración de usuarios en el sistema

---

2. Listar archivos con un patrón específico

ls ../../bin/x???* # Listar archivos que comienzan con "x" y tienen 3 o más caracteres

ls ../../bin/[aeiou]*[aeiou] ../../bin/[aeiou] # Listar archivos que comienzan y terminan con vocal

---

3. Crear y trabajar con directorios y archivos

mkdir ejercicio # Crear un directorio llamado "ejercicio"

cp ../../etc/group ejercicio/grupos # Copiar el archivo "group" al directorio "ejercicio"

cd ejercicio # Cambiar al directorio "ejercicio"

mkdir -p {2000..2100}/{01..12} # Crear subdirectorios para cada año entre 2000 y 2100 y meses del 01 al 12

touch {2000..2100}/{01..12}/{01..31} # Crear archivos del 01 al 31 para cada mes y año

rm {2000..2100}/{0{2,4,6,9},11}/31 # Eliminar los días 31 de los meses que no tienen 31 días

touch {2000,2004,2008,2012,.....,2096}/02/29 # Añadir el 29 de febrero para años bisiestos

ls */*/31 # Mostrar todos los días 31 de meses con 31 días

---

4. Contar la inicial más común entre los usuarios

cat /etc/passwd | cut -c1 | sort | uniq -c | tr -s ' ' : | sort -t: -k2nr | head -1 | cut -f2 -d: # Contar cuántas veces se repite la inicial de los usuarios

---

5. Validar y extraer correos electrónicos

cat emails | grep -Eo "^[a-z0-9]+(\.+[a-z0-9]+)*@[a-z0-9]+(\.+[a-z0-9]+)*(\.+[a-z0-9]{2,})" # Extraer correos válidos del archivo "emails"

---

6. Filtrar usuarios según su ID numérico

cat /etc/passwd | cut -d: -f1-3 | sort -n | grep :[0-9][0-9]$ |cut -d: -f1 # Filtrar usuarios con ID numérico mayor que 9

---

7. Listar usuarios con el mismo UID

cat /etc/passwd | cut -f1,3,4 -d: | grep -Ev "^[^:]+:([^:]+):\1" # Mostrar usuarios que no tengan el mismo UID

---

8. Contar los caracteres de los nombres de usuario

cat /etc/passwd | cut -f1 -d: | xargs -I{} bash -c "echo -n {}: ; echo {} | wc -c" # Contar cuántos caracteres tiene cada nombre de usuario

---

9. Buscar el nombre de usuario más largo

cat /etc/passwd | cut -f1 -d: | xargs -Iuser bash -c 'echo -n user: ; echo -n user | wc -c' | sort -n -k2 -t: | tail -1 | cut -f1 -d: # Buscar el nombre de usuario más largo

---

10. Sumar el tamaño de los ficheros en "/etc/"

echo $(ls -l /etc/ | tr -s ' ' | cut -f5 -d' ') | tr ' ' + | bc # Sumar el tamaño de todos los ficheros en "/etc/"

---

11. Sumar el tamaño de los ficheros regulares en "/etc/"

echo $(ls -la /etc/ | grep "^-" | tr -s ' ' | cut -f5 -d' ') | tr ' ' + | bc # Sumar el tamaño solo de los ficheros regulares en "/etc/"

---

12. Sumar tamaños con `find`

echo $(find /etc -maxdepth 1 -type f -printf %s+)0 | bc # Sumar los tamaños de ficheros regulares en "/etc/" usando find

---

13. Obtener usuarios con el nombre más largo

userlengths=$(cat users | cut -f1 -d: | xargs -Iuser bash -c 'echo -n user: ; echo -n user | wc -c')

maxlength=$(echo -e "$userlengths" | cut -f2 -d: | sort -n | tail -1)

echo -e "$userlengths" | grep ":$maxlength$" | cut -f1 -d: # Obtener el nombre de usuario más largo

---

14. Mostrar grupos a los que pertenece un usuario

cat /etc/passwd | cut -d: -f1 | xargs -I{} bash -c 'groups {}' # Mostrar los grupos de cada usuario

---

15. Mostrar grupos de usuarios con formato adecuado

paste -d: <(cut -d: -f1 /etc/passwd) <(id -Gn $(cut -d: -f1 /etc/passwd) | tr ' ' ,) # Mostrar los grupos de cada usuario

---

16. Verificar direcciones de correo electrónico con un formato específico

cat emails | grep -Eo "^[a-z0-9]+(\.+[a-z0-9]+)*@[a-z0-9]+(\.+[a-z0-9]+)*(\.+[a-z0-9]{2,})" # Verificar direcciones de correo electrónico válidas

---

17. Comprobar que un campo IP es válido

cat /etc/passwd | cut -f1 -d: | xargs -I{} bash -c 'echo {} | grep -xE "(.).*\1"' # Comprobar que un campo IP es válido

---

Claro, te proporcionaré algunas funciones de **filtro**, **mapeo**, **validación** y **cambio de formato** basadas en los ejercicios que me diste. Estas funciones pueden ser útiles para implementar en tu práctica de Bash. Las funciones serán lo suficientemente generales como para poder adaptarlas fácilmente a diferentes escenarios. Aquí tienes algunas:

Función de Filtro: Validar Correos Electrónicos

```
# Función para validar correos electrónicos (acepta correos válidos según el patrón).

validar_correo() {

  echo "$1" | grep -Eq "^[a-z0-9]+(\.+[a-z0-9]+)*@[a-z0-9]+(\.+[a-z0-9]+)*(\.+[a-z0-9]{2,})$"

  if [ $? -eq 0 ]; then

    echo "Correo válido: $1"

  else
```

```
    echo "Correo inválido: $1"
  fi
}

# Ejemplo de uso
validar_correo "usuario@dominio.com"
validar_correo "usuario@dominio"
```

Función de Mapeo: Calcular Longitud de Nombres de Usuarios

```
# Función para mostrar el nombre de usuario y su longitud
mapear_longitud_usuario() {
  cat /etc/passwd | cut -f1 -d: | while read usuario; do
    echo "$usuario: $(echo -n $usuario | wc -c)"
  done
}

# Ejemplo de uso
mapear_longitud_usuario
```

Función de Validación: Verificar UID y GID Coincidentes

```
# Función para verificar si el UID y GID coinciden
validar_uid_gid() {
  cat /etc/passwd | cut -f1,3,4 -d: | while read linea; do
    usuario=$(echo $linea | cut -d: -f1)
```

```
    uid=$(echo $linea | cut -d: -f2)
    gid=$(echo $linea | cut -d: -f3)
    if [ "$uid" -eq "$gid" ]; then
      echo "$usuario tiene UID y GID coincidentes"
    fi
  done
}

# Ejemplo de uso
validar_uid_gid
```

Función de Cambio de Formato: Transformar Fecha de Formato DD/MM/YYYY a YYYY-MM-DD

```
# Función para cambiar el formato de fecha (DD/MM/YYYY a YYYY-MM-DD)
cambiar_formato_fecha() {
  echo "$1" | sed -E 's/([0-9]{2})\/([0-9]{2})\/([0-9]{4})/\3-\2-\1/'
}

# Ejemplo de uso
cambiar_formato_fecha "20/02/2025"
```

Función de Filtro: Filtrar Archivos con Nombre que Empiecen con 'x' y Tengan al Menos Tres Caracteres

```
# Función para filtrar archivos que empiecen con 'x' y tengan 3 o más caracteres
```

```
filtrar_archivos_x() {
  ls "$1" | grep -E "^x.{2,}$"
}

# Ejemplo de uso
filtrar_archivos_x "/bin"
```

Función de Mapeo: Sumar Tamaño de Archivos

```
# Función para sumar el tamaño de los archivos en un directorio dado
sumar_tamano_archivos() {
  total=0
  for archivo in "$1"/*; do
    if [ -f "$archivo" ]; then
      tamano=$(stat --format=%s "$archivo")
      total=$((total + tamano))
    fi
  done
  echo "El tamaño total de los archivos es: $total bytes"
}

# Ejemplo de uso
sumar_tamano_archivos "/etc"
```

Función de Validación: Verificar Archivos Regulares en un Directorio

```
# Función para verificar si un archivo es regular en un directorio dado
verificar_archivos_regulares() {
  find "$1" -maxdepth 1 -type f | while read archivo; do
    echo "$archivo es un archivo regular"
  done
}

# Ejemplo de uso
verificar_archivos_regulares "/etc"
```

Función de Filtro: Mostrar Archivos con Permisos de Solo Lectura

```
# Función para filtrar archivos con permisos de solo lectura
filtrar_solo_lectura() {
  ls -l "$1" | grep -E '^.r--'
}

# Ejemplo de uso
filtrar_solo_lectura "/etc"
```

Función de Mapeo: Extraer Solo Usuarios de /etc/passwd

```
# Función para extraer los usuarios del archivo /etc/passwd
extraer_usuarios() {
  cat /etc/passwd | cut -f1 -d:
}
```

```
# Ejemplo de uso
extraer_usuarios
```

Función de Cambio de Formato: Convertir Texto a Mayúsculas

```
# Función para convertir el texto a mayúsculas
convertir_mayusculas() {
  echo "$1" | tr 'a-z' 'A-Z'
}

# Ejemplo de uso
convertir_mayusculas "hola mundo"
```

Función de Filtro: Mostrar Archivos con Expresión Regular

```
# Función para filtrar archivos con una expresión regular
filtrar_archivos_regex() {
  ls "$1" | grep -E "$2"
}

# Ejemplo de uso
filtrar_archivos_regex "/bin" "^x"
```

Con estas funciones puedes realizar tareas específicas de **filtro**, **mapeo**, **validación** y **cambio de formato** para trabajar con archivos, directorios y

cadenas. Son fácilmente adaptables a nuevos casos de uso, solo necesitas cambiar los parámetros de entrada o adaptar la lógica.

Función de Filtro: Filtrar Directorios Vacíos

```
# Función para filtrar directorios vacíos

filtrar_directorios_vacios() {

  find "$1" -type d -empty

}


# Ejemplo de uso

filtrar_directorios_vacios "/ruta/del/directorio"
```

Función de Mapeo: Convertir una Lista de Archivos a Formato CSV

```
# Función para convertir la lista de archivos en un directorio a formato CSV

convertir_a_csv() {

  ls "$1" | tr '\n' ',' | sed 's/,$/\n/'

}


# Ejemplo de uso

convertir_a_csv "/etc"
```

Función de Validación: Verificar si el Archivo es Ejecutable

```
# Función para verificar si un archivo es ejecutable

verificar_ejecutable() {

  if [ -x "$1" ]; then
```

```
    echo "$1 es ejecutable"
  else
    echo "$1 no es ejecutable"
  fi
}

# Ejemplo de uso
verificar_ejecutable "/bin/bash"
```

Función de Cambio de Formato: Reemplazar Espacios por Guiones

```
# Función para reemplazar espacios por guiones en una cadena
reemplazar_espacios() {
  echo "$1" | sed 's/ /-/g'
}

# Ejemplo de uso
reemplazar_espacios "nombre del archivo"
```

Función de Filtro: Filtrar Archivos Modificados Hoy

```
# Función para filtrar archivos modificados hoy
filtrar_archivos_modificados_hoy() {
  find "$1" -type f -mtime -1
}
```

```
# Ejemplo de uso
filtrar_archivos_modificados_hoy "/var/log"
```

Función de Mapeo: Crear un Resumen de Tamaño de Archivos y Directorios

```
# Función para crear un resumen del tamaño de archivos y directorios
resumen_tamano_directorio() {
  du -sh "$1"/* | sort -rh
}

# Ejemplo de uso
resumen_tamano_directorio "/home/usuario"
```

Función de Validación: Verificar si una Carpeta Existe

```
# Función para verificar si una carpeta existe
verificar_carpeta_existe() {
  if [ -d "$1" ]; then
    echo "La carpeta $1 existe"
  else
    echo "La carpeta $1 no existe"
  fi
}

# Ejemplo de uso
verificar_carpeta_existe "/home/usuario/documentos"
```

Función de Filtro: Filtrar Archivos Según su Extensión

```
# Función para filtrar archivos por su extensión
filtrar_por_extension() {
  find "$1" -type f -name "*.$2"
}

# Ejemplo de uso
filtrar_por_extension "/home/usuario" "txt"
```

Función de Mapeo: Contar el Número de Palabras en un Archivo

```
# Función para contar el número de palabras en un archivo
contar_palabras_archivo() {
  wc -w "$1"
}

# Ejemplo de uso
contar_palabras_archivo "/home/usuario/documento.txt"
```

Función de Validación: Verificar si un Archivo Existe y es Regular

```
# Función para verificar si un archivo existe y es regular
verificar_archivo_regular() {
  if [ -f "$1" ]; then
    echo "$1 es un archivo regular"
```

```
  else
    echo "$1 no es un archivo regular"
  fi
}

# Ejemplo de uso
verificar_archivo_regular "/etc/passwd"
```

Función de Cambio de Formato: Convertir Texto a Minúsculas

```
# Función para convertir el texto a minúsculas
convertir_minusculas() {
  echo "$1" | tr 'A-Z' 'a-z'
}

# Ejemplo de uso
convertir_minusculas "HOLA MUNDO"
```

Función de Filtro: Filtrar Directorios con Permisos de Escritura

```
# Función para filtrar directorios con permisos de escritura
filtrar_directorios_con_escritura() {
  find "$1" -type d -perm /u=w
}

# Ejemplo de uso
```

```
filtrar_directorios_con_escritura "/home/usuario"
```

Función de Mapeo: Mostrar el Número de Archivos en un Directorio

```
# Función para contar el número de archivos en un directorio
contar_archivos() {
  find "$1" -type f | wc -l
}

# Ejemplo de uso
contar_archivos "/etc"
```

Función de Validación: Comprobar si un Usuario Está Logueado

```
# Función para comprobar si un usuario está logueado
usuario_logueado() {
  who | grep -q "$1"
  if [ $? -eq 0 ]; then
    echo "El usuario $1 está logueado"
  else
    echo "El usuario $1 no está logueado"
  fi
}

# Ejemplo de uso
usuario_logueado "root"
```

Función de Filtro: Filtrar Archivos de un Usuario Específico

```
# Función para filtrar archivos creados por un usuario específico

filtrar_archivos_usuario() {

  find "$1" -type f -user "$2"

}


# Ejemplo de uso

filtrar_archivos_usuario "/home/usuario" "usuario1"
```

Consejos Adicionales:

- **Filtrado de directorios/archivos:** Utiliza `find` para localizar archivos con patrones específicos (como extensión o permisos) y `grep` para hacer búsquedas dentro de los archivos.
- **Mapeo:** Puedes usar comandos como `wc`, `cut`, `awk`, y `sed` para procesar y transformar información, como contar palabras, extraer partes de un archivo, etc.
- **Validación:** Es importante usar combinaciones de condicionales `if` y comandos como `test`, `-e`, `-d`, `-f`, y `-x` para verificar si archivos, directorios o usuarios existen, son regulares o tienen permisos específicos.
- **Cambio de formato:** `sed`, `tr`, y `awk` son tus aliados cuando necesitas modificar cadenas o formatos, como convertir texto a mayúsculas/minúsculas o reemplazar caracteres.

Con estas funciones adicionales, puedes estar mejor preparado para cualquier ejercicio que involucre manipulación de archivos y directorios, validación de datos, o transformación de texto en tu examen de Bash. Si necesitas ejemplos específicos de cómo integrarlas en un caso de examen o más detalles, ¡déjame saber!

📌 Generar URLs de imágenes desde claves

```
generate_urls() {
    local prefix="$1"
    local suffix="default.jpg"
    while read id; do
        echo "$prefix/$id/$suffix"
    done
}
```

- **Uso:**

```
cat claves | generate_urls "https://i.ytimg.com/vi"
```

📌 Extraer claves desde URLs

```
extract_keys() {
    while read url; do
        echo "$url" | cut -d'/' -f5
    done
}
```

- **Uso:**

```
cat urls | extract_keys
```

📌 Extraer extensión de archivos desde URLs

```
extract_extension() {
    while read url; do
        echo "$url" | rev | cut -d'.' -f1 | rev
    done
}
```

- **Uso:**

```
cat urls | extract_extension
```

📌 Validar direcciones IP (0-255 en cada octeto)

```
validate_ip() {
    while read ip; do
        echo "$ip" | grep -xE "([0-9]{1,3}\.){3}[0-9]{1,3}" | while read valid_ip; do
            IFS='.' read -r o1 o2 o3 o4 <<< "$valid_ip"
            if (( o1 <= 255 && o2 <= 255 && o3 <= 255 && o4 <= 255 )); then
                echo "$valid_ip"
            fi
        done
    done
}
```

- **Uso:**

```
cat ips | validate_ip
```

📌 Ordenar IPs eliminando ceros a la izquierda

```
format_and_sort_ips() {
    while read ip; do
        formatted_ip=$(echo "$ip" | awk -F '.' '{printf "%d.%d.%d.%d\n", $1, $2, $3, $4}')
        echo "$formatted_ip"
    done | sort -t. -k1,1n -k2,2n -k3,3n -k4,4n
}
```

- **Uso:**

```
cat ips_validas | format_and_sort_ips
```

📌 Expresiones Regulares Útiles

- **Extraer claves de URLs:**

```
https?:\/\/[^\/]+\/vi\/([^\/]+)\/
```

- **Validar IPs de 0-255:**

```
^([0-9]{1,3}\.){3}[0-9]{1,3}$
```

- **Extraer extensiones de archivos:**

```
\.[^.]+$
```

📌 Manipulación de Archivos y Texto

📂 Listar archivos de un tipo específico

```
list_files_by_extension() {
    local ext="$1"
    find . -type f -name "*.$ext"
}
```

- **Uso:**

```
list_files_by_extension "txt"
```

📂 Contar líneas, palabras y caracteres en un archivo

```
count_file_content() {
    local file="$1"
    wc "$file"
}
```

- **Uso:**

```
count_file_content "archivo.txt"
```

📝 Convertir un texto a mayúsculas o minúsculas

```
to_uppercase() {
    tr '[:lower:]' '[:upper:]'
}

to_lowercase() {
    tr '[:upper:]' '[:lower:]'
```

```
}
```

🔹 **Uso:**

```
echo "Hola Mundo" | to_uppercase   # HOLA MUNDO
echo "Hola Mundo" | to_lowercase   # hola mundo
```

📂 Reemplazar texto en un archivo

```
replace_text() {
    local old="$1"
    local new="$2"
    local file="$3"
    sed -i "s/$old/$new/g" "$file"
}
```

🔹 **Uso:**

```
replace_text "error" "solución" "log.txt"
```

📂 Buscar líneas que contienen una palabra en un archivo

```
search_word() {
    grep -i "$1" "$2"
}
```

🔹 **Uso:**

```
search_word "error" "log.txt"
```

📂 Extraer solo los nombres de archivos sin la ruta

```
extract_filenames() {
    ls -1 | awk -F '/' '{print $NF}'
}
```

📌 Validaciones en Bash

🔢 Validar si una variable es un número

```
is_number() {
    [[ "$1" =~ ^[0-9]+$ ]]
}
```

- **Uso:**

```
is_number "123" && echo "Es número" || echo "No es número"
```

📅 Validar formato de fecha (YYYY-MM-DD)

```
is_valid_date() {
    [[ "$1" =~ ^[0-9]{4}-[0-9]{2}-[0-9]{2}$ ]]
}
```

- **Uso:**

```
is_valid_date "2024-02-19" && echo "Fecha válida" || echo "Fecha inválida"
```

📂 Validar si un archivo existe

```
file_exists() {
    [[ -f "$1" ]]
}
```

- 🔹 **Uso:**

```
file_exists "archivo.txt" && echo "Existe" || echo "No existe"
```

📌 Formatos y Filtros Útiles

📂 Formatear números con separadores de miles

```
format_number() {
    printf "%'d\n" "$1"
}
```

- 🔹 **Uso:**

```
format_number 1000000  # 1,000,000
```

📅 Obtener la fecha en diferentes formatos

```
get_date() {
    date "+%Y-%m-%d"
}

get_time() {
    date "+%H:%M:%S"
}
```

🔹 **Uso:**

```
get_date  # 2024-02-19
get_time  # 14:45:30
```

📂 Filtrar solo líneas con números en un archivo

```
filter_numbers() {
    grep -E "^[0-9]+$"
}
```

🔹 **Uso:**

```
cat archivo.txt | filter_numbers
```

📂 Ordenar líneas de un archivo alfabéticamente y eliminar duplicados

```
sort_unique() {
    sort | uniq
}
```

🔹 **Uso:**

```
cat archivo.txt | sort_unique
```

📂 Extraer direcciones de correo electrónico

```
extract_emails() {
```

```
    grep -Eo "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"
}
```

🔹 **Uso:**

```
cat correos.txt | extract_emails
```

📂 Extraer números de teléfono

```
extract_phone_numbers() {
    grep -Eo "[0-9]{3}-[0-9]{3}-[0-9]{4}"
}
```

🔹 **Uso:**

```
cat agenda.txt | extract_phone_numbers
```

📌 Expresiones Regulares Útiles

🔸 **Validar dirección IP**

```
^([0-9]{1,3}\.){3}[0-9]{1,3}$
```

🔸 **Validar fecha (YYYY-MM-DD)**

```
^[0-9]{4}-[0-9]{2}-[0-9]{2}$
```

🔸 **Extraer extensión de archivo**

```
\.[^.]+$
```

🔸 **Buscar palabras en un archivo ignorando mayúsculas/minúsculas**

(?i)palabra

¡Por supuesto! Aquí tienes un **resumen de la sintaxis** de los comandos, estructuras y operadores más usados en Bash, para que no cometas errores en el examen.

📌 Sintaxis en Bash

🔹 Estructura de un script en Bash

#!/bin/bash

Comentario en Bash

echo "Hola, mundo" # Imprimir en pantalla

🔹 Variables

nombre="Juan"   # Asignar variable (sin espacios)

echo "$nombre"   # Mostrar valor de la variable

numero=10

echo $((numero + 5)) # Operaciones aritméticas

📌 **IMPORTANTE:** No usar `$` al asignar valores.

🔹 Leer entrada del usuario

```
read -p "Introduce tu nombre: " nombre
echo "Hola, $nombre"
```

🔹 Condiciones (if, elif, else)

```
if [ "$nombre" == "Juan" ]; then
   echo "Hola, Juan"
elif [ "$nombre" == "Maria" ]; then
   echo "Hola, Maria"
else
   echo "No te conozco"
fi
```

📌 **IMPORTANTE:**

- **Espacios obligatorios** dentro de `[ ]`
- **Comparaciones correctas:**
  - Cadenas: `==`, `!=`
  - Números: `-eq`, `-ne`, `-lt`, `-le`, `-gt`, `-ge`
  - Archivos: `-f` (existe), `-d` (es directorio)

🔹 Bucles

Bucle `for`

```
for i in {1..5}; do
    echo "Número $i"
done
```

Bucle `while`

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

📌 **IMPORTANTE:**

- `$((contador++))` ✅ **Correcto**
- `contador++` ❌ **Error en Bash**

🔹 Funciones

```
mi_funcion() {
    echo "Hola desde la función"
}
mi_funcion  # Llamar a la función
```

📌 **IMPORTANTE:**

- Se define sin `()` en los parámetros.
- Se llama sin `()` (diferente de otros lenguajes).

🔹 Operaciones con archivos

Verificar si un archivo existe

```
if [ -f "archivo.txt" ]; then
    echo "El archivo existe"
fi
```

Leer un archivo línea por línea

```
while read linea; do
    echo "Línea: $linea"
done < archivo.txt
```

🔹 Redirecciones y Tuberías

Redirigir salida a un archivo

```
echo "Hola" > archivo.txt  # Sobreescribe
echo "Mundo" >> archivo.txt  # Añade
```

Redirigir entrada desde un archivo

```
sort < archivo.txt  # Ordena contenido del archivo
```

Encadenar comandos con tuberías

```
cat archivo.txt | grep "error" | sort
```

📌 **IMPORTANTE:** Se recomienda usar

```
grep "error" archivo.txt | sort
```

en lugar de `cat archivo.txt | grep "error"`.

🔹 Expresiones regulares con `grep`, `sed` y `awk`

Buscar texto en un archivo

```
grep "error" archivo.txt  # Busca "error" en el archivo
grep -i "error" archivo.txt  # Ignora mayúsculas/minúsculas
grep -E "[0-9]{3}-[0-9]{3}-[0-9]{4}" archivo.txt  # Expresión regular
```

Reemplazar texto en un archivo

```
sed -i 's/error/solución/g' archivo.txt  # Reemplaza "error" por "solución"
```

Extraer columna de un archivo

```
awk '{print $2}' archivo.txt  # Muestra la segunda columna
```

🔹 Manipulación de Fechas

Obtener la fecha y hora actual

```
date "+%Y-%m-%d %H:%M:%S"
```

Sumar días a la fecha actual

```
date -d "+3 days" "+%Y-%m-%d"
```

Restar días a la fecha actual

```
date -d "-3 days" "+%Y-%m-%d"
```

🔹 Operaciones con Números

Suma y Resta

```
echo $((5 + 3))  # 8
```

```
echo $((5 - 3))  # 2
```

Multiplicación y División

```
echo $((5 * 3))  # 15
```

```
echo $((10 / 2))  # 5
```

🔹 Ordenar y Filtrar Datos

Ordenar alfabéticamente

```
sort archivo.txt
```

Ordenar numéricamente

```
sort -n archivo.txt
```

Eliminar duplicados

```
sort archivo.txt | uniq
```

🔹 Expresiones Regulares Útiles

📌 Validar direcciones IP

```
^([0-9]{1,3}\.){3}[0-9]{1,3}$
```

📌 Validar fechas (YYYY-MM-DD)

^[0-9]{4}-[0-9]{2}-[0-9]{2}$

📌 Extraer extensión de archivo

\.[^.]+$

📌 Validar correos electrónicos

[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}

¡Por supuesto! Aquí tienes funciones en **Bash** para **sumar y restar fechas**, además de validaciones y filtros útiles que pueden servirte en el examen.

📌 Funciones para Fechas en Bash

🔹 Sumar días a una fecha

```
sumar_dias() {
    fecha="$1"   # Fecha en formato YYYY-MM-DD
    dias="$2"    # Número de días a sumar
    date -d "$fecha +$dias days" "+%Y-%m-%d"
}

# Ejemplo:
```

```
sumar_dias "2024-02-20" 5  # Salida: 2024-02-25
```

🔹 Restar días a una fecha

```
restar_dias() {
    fecha="$1"
    dias="$2"
    date -d "$fecha -$dias days" "+%Y-%m-%d"
}

# Ejemplo:
restar_dias "2024-02-20" 5  # Salida: 2024-02-15
```

🔹 Diferencia de días entre dos fechas

```
diferencia_dias() {
    fecha1=$(date -d "$1" "+%s")
    fecha2=$(date -d "$2" "+%s")
    echo $(( (fecha2 - fecha1) / 86400 ))  # 86400 segundos en un día
}

# Ejemplo:
diferencia_dias "2024-02-15" "2024-02-20"  # Salida: 5
```

🔹 Validar formato de fecha (YYYY-MM-DD)

```
validar_fecha() {
```

```
    if [[ "$1" =~ ^[0-9]{4}-[0-9]{2}-[0-9]{2}$ ]]; then
        date -d "$1" "+%Y-%m-%d" &>/dev/null
        if [ $? -eq 0 ]; then
            echo "Fecha válida"
        else
            echo "Fecha inválida"
        fi
    else
        echo "Formato incorrecto. Use YYYY-MM-DD"
    fi
}

# Ejemplo:
validar_fecha "2024-02-30"  # Salida: Fecha inválida
validar_fecha "2024-02-28"  # Salida: Fecha válida
```

📌 Filtros y Validaciones Útiles

🔹 Filtrar solo fechas válidas de un archivo

```
grep -E "^[0-9]{4}-[0-9]{2}-[0-9]{2}$" archivo.txt | while read fecha; do
    date -d "$fecha" "+%Y-%m-%d" &>/dev/null
    if [ $? -eq 0 ]; then
        echo "$fecha"
    fi
done
```

🔹 Filtrar IPs válidas de un archivo

grep -E "^([0-9]{1,3}\.){3}[0-9]{1,3}$" archivo.txt | awk -F. '$1<=255 && $2<=255 && $3<=255 && $4<=255'

🔹 Filtrar líneas con números y ordenarlos numéricamente

grep -E "^[0-9]+$" archivo.txt | sort -n

🔹 Eliminar líneas duplicadas en un archivo

sort archivo.txt | uniq

¡Claro! Aquí tienes una lista de **expresiones regulares útiles** que te pueden servir en tu examen de **Bash** para validaciones, filtros y manipulaciones de texto.

🔹 EXPRESIONES REGULARES ÚTILES EN BASH

1️⃣ Validaciones de formatos

Expresión Regular	Descripción	Ejemplo de uso
`^[0-9]+$`	Solo números enteros positivos	`grep -E "^[0-9]+$" archivo.txt`

`^-?[0-9]+$`	Números enteros (positivos y negativos)	`grep -E "^-?[0-9]+$" archivo.txt`
`^[0-9]+\.[0-9]+$`	Números decimales	`grep -E "^[0-9]+\.[0-9]+$" archivo.txt`
`^[a-zA-Z]+$`	Solo letras (mayúsculas y minúsculas)	`grep -E "^[a-zA-Z]+$" archivo.txt`
`^[a-zA-Z0-9]+$`	Solo letras y números (sin espacios ni símbolos)	`grep -E "^[a-zA-Z0-9]+$" archivo.txt`
`^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$`	Validar email	`grep -E "^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$" archivo.txt`

2️⃣ Validación de Fechas y Horas

Expresión Regular	Descripción	Ejemplo de uso
`^[0-9]{4}-[0-9]{2}-[0-9]{2}$`	Fecha en formato YYYY-MM-DD	`grep -E "^[0-9]{4}-[0-9]{2}-[0-9]{2}$" archivo.txt`

`^[0-9]{2}/[0-9]{2}/[0-9]{4}$`	Fecha en formato DD/MM/YYYY	`grep -E "^[0-9]{2}/[0-9]{2}/[0-9]{4}$" archivo.txt`
`^([01][0-9]	2[0-3]):[0-5][0-9]$`	Hora en formato 24h (HH:MM)

3️⃣ Validación de Direcciones IP

Expresión Regular	Descripción	Ejemplo de uso
`^([0-9]{1,3}\.){3}[0-9]{1,3}$`	Formato básico de IP (sin validar valores)	`grep -E "^([0-9]{1,3}\.){3}[0-9]{1,3}$" archivo.txt`
`^((25[0-5]	2[0-4][0-9]	[01]?[0-9][0-9]?).){3}(25[0-5]

4️⃣ Manipulación de Archivos y Rutas

Expresión Regular	Descripción	Ejemplo de uso
`^/([^/\0]+/)*[^/\0]+$`	Ruta absoluta de archivo o carpeta	`grep -E "^/([^/\0]+/)*[^/\0]+$" archivo.txt`
`^([a-zA-Z]:\\)?(\\[a-zA-Z0-9_.-]+)+\\?$`	Ruta en Windows	`grep -E "^([a-zA-Z]:\\)?(\\[a-zA`

		`-Z0-9_.-]+)+\\?$" archivo.txt`
`^[a-zA-Z0-9_.-]+\.[a-zA-Z0-9]+$`	Nombre de archivo con extensión	`grep -E "^[a-zA-Z0-9_.-]+\.[a-zA-Z0-9]+$" archivo.txt`

5️⃣ Filtrado de Palabras y Caracteres

Expresión Regular	Descripción	Ejemplo de uso
`\bpalabra\b`	Buscar una palabra exacta	`grep -E "\berror\b" archivo.txt`
`[^a-zA-Z0-9]`	Caracteres especiales (no alfanuméricos)	`grep -E "[^a-zA-Z0-9]" archivo.txt`
`^$`	Líneas en blanco	`grep -E "^$" archivo.txt`
`\s+`	Espacios en blanco (uno o más)	`grep -E "\s+" archivo.txt`
`^\s+	\s+$`	Espacios al inicio o final de línea	

6️⃣ Expresiones para Comandos `sed`, `awk`, `grep`

- **Eliminar espacios en blanco al inicio y final**

```
sed 's/^[ \t]*//;s/[ \t]*$//'
```

- **Reemplazar todas las ocurrencias de "foo" por "bar"**

```
sed 's/foo/bar/g'
```

- **Extraer la primera columna de un archivo separado por espacios**

```
awk '{print $1}'
```

- **Eliminar líneas duplicadas**

```
sort archivo.txt | uniq
```

- **Mostrar líneas que no contengan un patrón**

```
grep -v "patrón" archivo.txt
```

7️⃣ Expresiones para Formatos Específicos

Expresión Regular	Descripción	Ejemplo de uso
`^[A-Z][a-z]+$`	Nombre propio (primera mayúscula, resto minúsculas)	`grep -E "^[A-Z][a-z]+$" archivo.txt`

`^\+?[0-9]{1,3}-?[0-9]{6,10}$`	Número de teléfono internacional	`grep -E "^\+?[0-9]{1,3}-?[0-9]{6,10}$" archivo.txt`
`^#[0-9A-Fa-f]{6}$`	Código de color HEX (ej. `#FFA07A`)	`grep -E "^#[0-9A-Fa-f]{6}$" archivo.txt`

📌 CONSEJOS PARA EL EXAMEN

1. **Usa `grep -E`** para aplicar expresiones regulares extendidas.
2. **Usa `sed` y `awk`** para manipular texto de manera eficiente.
3. **Combina con `sort`, `uniq`, `cut`, `tr`** para procesar archivos de texto.

Prueba las expresiones con `echo` antes de usarlas en archivos reales:
echo "Texto a probar" | grep -E "expresión"

4.
5. **Si tienes dudas en el examen, usa `man comando`** para ver la documentación.
