

UNIVERSIDAD DE COSTA RICA

ESCUELA DE INGENIERÍA ELÉCTRICA

IE0117: PROGRAMACIÓN BAJO PLATAFORMAS ABIERTAS

---

# Reporte

## Laboratorio #

---

Prof. Carolina Trejos Quirós

Estudiante: David Abraham Vega Naranjo, C38344

1 de abril de 2025

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Implementación</b>	<b>2</b>
2.1. Scripting, usuarios y permisos . . . . .	2
2.1.1. Codigos . . . . .	2
2.2. Scripting y procesos . . . . .	4
2.2.1. Codigos . . . . .	4
2.3. Scripting y servicios . . . . .	6
2.3.1. Codigos . . . . .	6
<b>3. Resultados</b>	<b>8</b>
3.1. Scripting, usuarios y permisos . . . . .	8
3.2. Scripting y procesos . . . . .	9
3.3. Scripting y servicios . . . . .	10
<b>4. Conclusiones y recomendaciones</b>	<b>11</b>
<b>Referencias</b>	<b>12</b>

# 1. Introducción

En este laboratorio se llevó a cabo diversas tareas orientadas al procesamiento de grupos, usuarios y directorios, también control y monitorización de procesos y servicios. Como objetivo principal fue dar a conocer y usar nuevos comandos útiles para llevar a cabo dichas tareas.

Mi repositorio [link](#).

## 2. Implementación

### 2.1. Scripting, usuarios y permisos

#### 2.1.1. Códigos

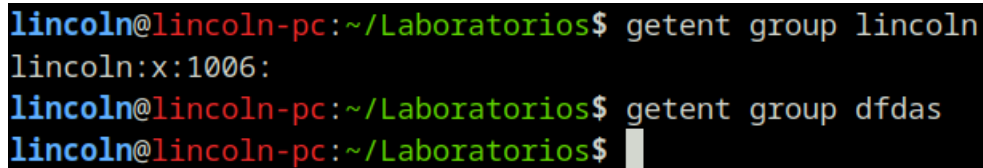
```
1      if [ "$(id -u)" -ne 0 ]; then
2          echo "Error: Este script debe ejecutarse como root."
3          exit
4      fi
```

Este código verifica que el script se ejecuta con el usuario root con el uso de "id -u", este comando imprime id de del usuario root, el id de root es 0, con eso se hace la comparativa.

```
1      if [ ! -e "$ruta" ]; then
2          echo "La ruta al archivo '$ruta' no existe." >> ar.log
3          exit
4      fi
```

Este código verifica que la ruta exista con el uso de condiciones de archivos, si no existe el archivo se redirige la salida del comando con ">>", el cual imprime un mensaje en "ar.log".

```
1     if getent group "$grupo"; then
2         echo "El grupo '$grupo' ya existe." >> ar.log
3     else
4         addgroup "$grupo"
5     fi
```

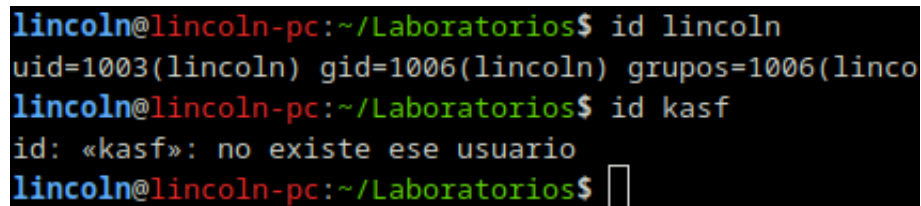


```
lincoln@lincoln-pc:~/Laboratorios$ getent group lincoln
lincoln:x:1006:
lincoln@lincoln-pc:~/Laboratorios$ getent group dfdas
lincoln@lincoln-pc:~/Laboratorios$
```

Figura 1: Ejemplo de imagen (getent)

Con ayuda del comando "getent" se verifica si existe el grupo, al ejecutarlo en la terminal si hay un grupo con ese nombre se imprime cierta informacion, sino no imprime la nada. Luego si no existe el grupo, se añade usando "addgroup".

```
1     if id "$usuario"; then
2         echo "El usuario '$usuario' ya existe." >> ar.log
3         usermod -m -G "$grupo" "$usuario"
4     else
5         adduser $usuario
6     fi
```



```
lincoln@lincoln-pc:~/Laboratorios$ id lincoln
uid=1003(lincoln) gid=1006(lincoln) grupos=1006(lincoln)
lincoln@lincoln-pc:~/Laboratorios$ id kasf
id: «kasf»: no existe ese usuario
lincoln@lincoln-pc:~/Laboratorios$
```

Figura 2: Ejemplo de imagen (id)

Con "id" solo si existe el usuario con ese nombre se imprime información, esto se verifica con el "if". Si el usuario existe se agrega al grupo nuevo creado anteriormente con "usermod -m -G", sino se agrega el nuevo usuario con "adduser".

```
1      chown $usuario:$grupo $ruta
2
3      chmod u+rw,g=r,o= $ruta
```

Por ultimo se ejecuta "chown" para cambiar el propietario del archivo al nuevo usuario y grupo, y "chmod" para cambiar los permisos; "u+rw" significa que el usuario podrá leer, escribir y ejecutar; "g=r" el grupo podrá solo leer y "o=" quiere decir que los demás no podrán realizar ninguna acción.

## 2.2. Scripting y procesos

### 2.2.1. Códigos

```
1      segundos_time_real() {
2          echo $(( (10#$(date +%M) * 60) + 10#$(date +%S) ))
3
4      }
```

Función para obtener los segundos en tiempo real, se convierten los minutos a segundos, para tener un margen de datos de una hora como mucho, se usa "10#" para convertir los números a base decimal y no tener problema con números que empiecen con 0.

```
1 inicio=$(segundos_time_real)
2 while [ "$intentos" -lt 10 ];do
3     echo "$((($segundos_time_real) - $inicio)) $(ps -C "$proceso" -o %cpu,%mem --
4     no-headers)" >> ar.log
5     intentos=$((intentos + 1))
6     sleep 2
7 done
```

Se inicia por así decir el contador de segundos. Se condiciona el while a un límite de intentos para que no sea infinito y así por poder graficar. Se hace el cálculo de los segundos transcurridos como dato para el eje x en la gráfica, y con el uso de "ps" se extrae el uso de memoria y cpu, se agrega "--no-headers" para eliminar el encabezado default de ps. Se hace el reporte en el archivo "ar.log". En "sleep" se puede cambiar para así no obtener datos constantes con respecto al tiempo.

```
1      cat > "grafica.gnuplot" <<EOF
2      set terminal jpeg
3      set output "grafica.jpeg"
4      set title "Uso de CPU y RAM"
5      set xlabel "Tiempo (s)"
6      set ylabel "Uso CPU (%)"
7      set y2label "Uso RAM (%)"
8      set ytics nomirror
9      set y2tics
10     set grid
11     plot "ar.log" using 1:2 title "CPU" with lines, \
12           "ar.log" using 1:3 title "RAM" with lines axes x1y2
13     EOF
14
15     gnuplot "grafica.gnuplot"
```

- # 1. "cat > "grafica.gnuplot" <<EOF"; con el uso de "cat" se escribe en el archivo de configuracion de gnuplot "grafica.gnuplot". Con "<<EOF" y "EOF" se marca el inicio y final.
- # 2. "set terminal jpeg" escoge el formato de salida de la imagen, en este caso es jpeg.
- # 3. "set output "grafica.jpeg"" se guarda la imagen.
- # 4. "set title "Uso de CPU y RAM"" se nombra el titulo de la grafica.
- # 5. "set xlabel "Tiempo (s)"" se nombra el eje x.
- # 6. "set ylabel "Uso CPU (%)"" y set y2label "Uso RAM (%)" nombran el eje y.
- # 7. "set ytics nomirror y set y2tics" permite que los ejes de "y" CPU y RAM tengan escalas separadas.
- # 8. "set grid" cuadricula la grafica.
- # 9. "plot" se usa para extraer los datos guardados en columnas en "ar.log" y "axes x1y2" indica que esos datos son para que el segundo eje de "y".

## 2.3. Scripting y servicios

### 2.3.1. Codigos

```
1 directorio="/home/lincoln/Desktop/scripts"
2
3 inotifywait -m -e modify,create,delete "$directorio" | while read path caso
archivo
4 do
```

Se utiliza el comando "inotifywait" con "-e" para monitorear eventos específicos como modificar, crear, eliminar. Con "While read" se extrae la información que imprime el comando.

```
1 if [[ "$archivo" != "ar.log" ]]; then
2     case $caso in
3         MODIFY)
4             echo "$(date +%e/%m/%Y\(%H:%M\)): Se modifiko el archivo '
$archivo' en $path" >> ar.log
5             ;;
6         CREATE)
7             echo "$(date +%e/%m/%Y\(%H:%M\)): Se creo el archivo '$archivo'
en $path" >> ar.log
8             ;;
9         DELETE)
10            echo "$(date +%e/%m/%Y\(%H:%M\)): Se elimino el archivo '$archivo
' en $path" >> ar.log
11            ;;
12        esac
13    fi
14 done
```

Para evitar hacer registros "irrelevantes" en "ar.log", se evitan los cambios en este archivo, ya que se encuentra en el directorio que se monitoria.

```
1  [Unit]
2  Description=Servicio de Monitoreo de Cambios en Directorios
3  After=network.target
4
5  [Service]
6  Type=simple
7  ExecStart=/bin/bash /home/lincoln/Desktop/scripts/2ejercicio3.sh
8  ExecStop=/usr/bin/pkill -f /home/lincoln/Desktop/scripts/2ejercicio3.sh
9  WorkingDirectory=/home/lincoln/Desktop/scripts
10 KillMode=control-group
11
12 [Install]
13 WantedBy=multi-user.target
```

Figura 3: Ejemplo de imagen de servicio

Se utiliza configuracion basica para el el servicio, con dos lineas agregas. "ExecStop" se configura para detener el script de bash con "pkill" y "KillMode" para detener todos los procesos generados por ese servicio.



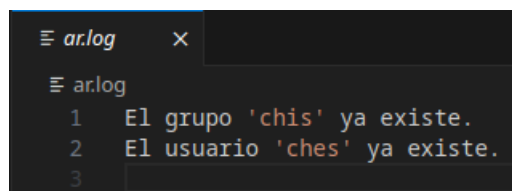
### 3. Resultados

#### 3.1. Scripting, usuarios y permisos

```
root@lincoln-pc:/home/lincoln/Desktop/scripts# ./2ejercicio1.sh ches chis /home/lincoln/Desktop/scripts/prueba.sh
-rw-r--r-- 1 lincoln lincoln 0 mar 31 13:03 /home/lincoln/Desktop/scripts/prueba.sh
Añadiendo el grupo 'chis' (GID 1000) ...
Hecho.
id: «ches»: no existe ese usuario
Añadiendo el usuario 'ches' ...
Añadiendo el nuevo grupo 'ches' (1004) ...
Adding new user 'ches' (1004) with group 'ches (1004)' ...
adduser: The home directory '/home/ches' already exists. Not touching this directory.
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para ches
Introduzca el nuevo valor, o pulse INTRO para usar el valor predeterminado
    Nombre completo []:
    Número de habitación []:
    Teléfono del trabajo []:
    Teléfono de casa []:
    Otro []:
¿Es correcta la información? [S/n] s
Adding new user 'ches' to supplemental / extra groups 'users' ...
Añadiendo al usuario 'ches' al grupo 'users' ...
-rwxr----- 1 ches chis 0 mar 31 13:03 /home/lincoln/Desktop/scripts/prueba.sh
root@lincoln-pc:/home/lincoln/Desktop/scripts#
```

Figura 4: output terminal

Este seria el resultado en la terminal, del caso donde el usuario, grupo y ruta existe, no se genera ningun mensaje en el "ar.log".



```
ar.log x
ar.log
1 El grupo 'chis' ya existe.
2 El usuario 'ches' ya existe.
3
```

Figura 5: output (ar.log)

En el caso de que el grupo y usuario existan, se imprimirían estos mensajes en el archivo "ar.log".

### 3.2. Scripting y procesos

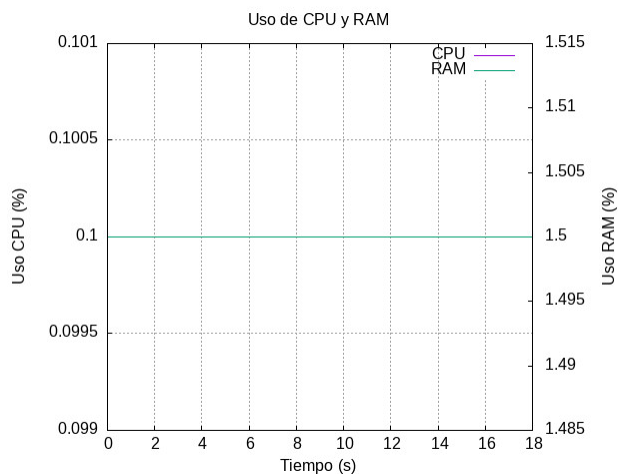


Figura 6: Caso 1

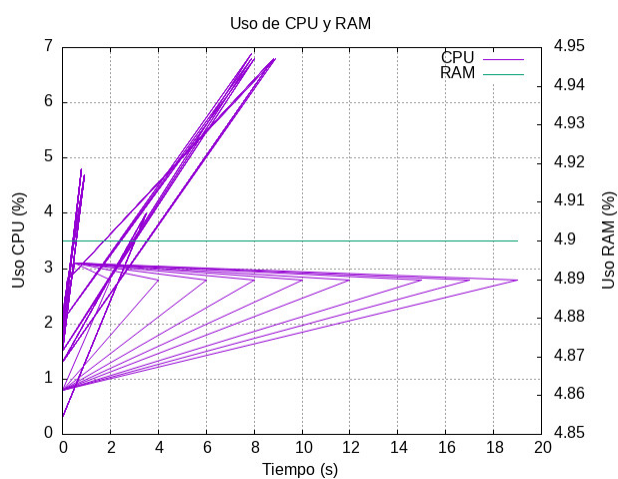
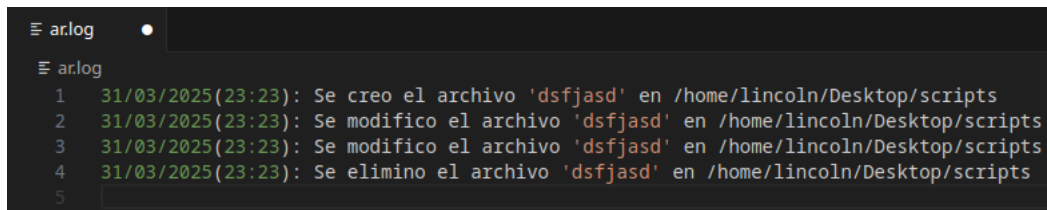


Figura 7: Caso 2

En el caso 1 se grafica un proceso unitario, y por así decirlo, no pertenece a un árbol de procesos. En cambio el caso 2 se grafica un árbol de procesos, por ende hay mayor cantidad de datos a graficar, entonces por eso se ve en la imagen esos picos de datos.

### 3.3. Scripting y servicios



```
ar.log
ar.log
1 31/03/2025(23:23): Se creo el archivo 'dsfjasd' en /home/lincoln/Desktop/scripts
2 31/03/2025(23:23): Se modifiko el archivo 'dsfjasd' en /home/lincoln/Desktop/scripts
3 31/03/2025(23:23): Se modifiko el archivo 'dsfjasd' en /home/lincoln/Desktop/scripts
4 31/03/2025(23:23): Se elimino el archivo 'dsfjasd' en /home/lincoln/Desktop/scripts
5
```

Figura 8: output (ar.log)

En la imagen se aprecia la salida esperada al ejecutar el script, el monitoreo del directorio con acciones como modificacion, creacion, eliminacion.

## 4. Conclusiones y recomendaciones

Como conclusion de este laboratorio es una buena base para tener en cuenta como prodrian ser las aplicaciones de cada tema, la herramienta gnuplot muy util para procesar datos, o ver un enfasis directo de los datos. El comando inotifywait tambien para tener control de los directorios o archivo, y ver exactamente que ocurren en ellos. La parte de los servicios muy interesante, a la hora de crear un nuevo servicios para ejecutar de manera mas tecnico un script, ya de manera mas automatica.

En el segundo ejercicio, seria util implementar, preguntar al usuario si desea graficar datos de un solo proceso, o y si fuera el caso de un arbol de procesos sumar el consumo de porcentaje de cpu y memoria, para asi que gnuplot grafique ya una funcion como tal de ese proceso en conjunto.

## Referencias

@manualid, title = GNU Coreutils - id, author = Free Software Foundation, year = 2025, url = <https://www.gnu.org/software/coreutils/id>, note = Accedido: 31 de marzo de 2025

@manualgetent, title = getent(1) - Linux manual page, author = die.net, year = 2025, url = <https://linux.die.net/man/1/getent>, note = Accedido: 31 de marzo de 2025

@manualgroupadd, title = groupadd(8) - Linux manual page, author = die.net, year = 2025, url = <https://linux.die.net/man/8/groupadd>, note = Accedido: 31 de marzo de 2025

@manualusermod, title = usermod(8) - Linux manual page, author = die.net, year = 2025, url = <https://linux.die.net/man/8/usermod>, note = Accedido: 31 de marzo de 2025

@manualps, title = ps(1) - Linux manual page, author = die.net, year = 2025, url = <https://linux.die.net/man/1/ps>, note = Accedido: 31 de marzo de 2025

@manualps, title = inotifywait(1) - Linux manual page, author = die.net, year = 2025, url = <https://linux.die.net/man/1/inotifywait>, note = Accedido: 31 de marzo de 2025