



**UNIVERSIDAD AUTONOMA DE
CHIHUAHUA**

FACULTAD DE INGENIERIA

SISTEMAS OPERATIVOS I

GRUPO: 5HW1

**NOMBRE: OSCAR ALBERTO SANCHEZ MOLINA,
KEVIN GABRIEL BALDERRAMA NEVAREZ, DANIEL
GRADO RUBIO, JOSE MIGUEL ROMANOS MORA**

MATRICULA: 357271, 361247, 361430, 341970

Chihuahua, Chih. FECHA DE ENTREGA: 15/11/2023

Instrucciones:

Crear un programa que muestre:

- Procesos activos del sistema
 - ID de proceso
 - Nombre proceso
- Diferenciar entre procesos de kernel y procesos de usuario

Toda esta información se encuentra dentro del directorio */proc*

El proyecto se deberá de subir a la carpeta correspondiente al equipo en github.

Se debe de crear un documento que explique lo que hace el programa desde la perspectiva de subsistemas del kernel y el funcionamiento del código.

Código en terminal

```
434 | N/A | Kernel
435 | N/A | Kernel
436 | N/A | Kernel
437 | N/A | Kernel
438 | N/A | Kernel
439 | N/A | Kernel
440 | N/A | Kernel
441 | N/A | Kernel
442 | N/A | Kernel
443 | N/A | Kernel
444 | N/A | Kernel
445 | N/A | Kernel
446 | N/A | Kernel
447 | N/A | Kernel
448 | N/A | Kernel
449 | N/A | Kernel
533 | /usr/lib/systemd/systemd-journald | Usuario
548 | /usr/lib/systemd/systemd-udevd | Usuario
616 | N/A | Kernel
627 | N/A | Kernel
628 | /usr/lib/systemd/systemd-conda | Usuario
635 | /usr/lib/systemd/systemd-resolved | Usuario
636 | /sbin/auditd | Usuario
637 | /usr/lib/systemd/systemd-userdbd | Usuario
638 | N/A | Kernel
639 | N/A | Kernel
683 | avahi-daemon: running [fedora.local] | Usuario
688 | /usr/libexec/lttng-monitor | Usuario
689 | /usr/sbin/mclog | Usuario
696 | /usr/lib/polkit-1/polkitd | Usuario
691 | /usr/libexec/powershell-daemon | Usuario
692 | /usr/libexec/rlkit-daemon | Usuario
693 | /usr/libexec/accounts-daemon | Usuario
694 | /usr/libexec/switcheroo-control | Usuario
695 | /usr/lib/systemd/systemd-logind | Usuario
696 | /usr/lib/systemd/systemd-machined | Usuario
697 | /usr/libexec/udisks2/udisksd | Usuario
698 | /usr/libexec/upowerd | Usuario
699 | /usr/bin/usbmuxd | Usuario
702 | /usr/sbin/abrt | Usuario
707 | /usr/sbin/ibus-service | Usuario
708 | avahi-daemon: chroot helper | Usuario
712 | /usr/bin/dbus-broker-launch | Usuario
745 | /usr/sbin/chronyd | Usuario
747 | dbus-broker | Usuario
766 | N/A | Kernel
768 | /usr/bin/abrt-dump-journal-core | Usuario
769 | /usr/bin/abrt-dump-journal-ops | Usuario
771 | /usr/bin/abrt-dump-journal-klog | Usuario
776 | /usr/sbin/alsactl | Usuario
786 | /usr/sbin/MidnightManager | Usuario
787 | /usr/bin/python3 | Usuario
801 | /usr/sbin/NetworkManager | Usuario
806 | /usr/sbin/cupsd | Usuario
819 | /usr/sbin/gssproxy | Usuario
```

Explicación

Función ObtenerNombrePrograma:

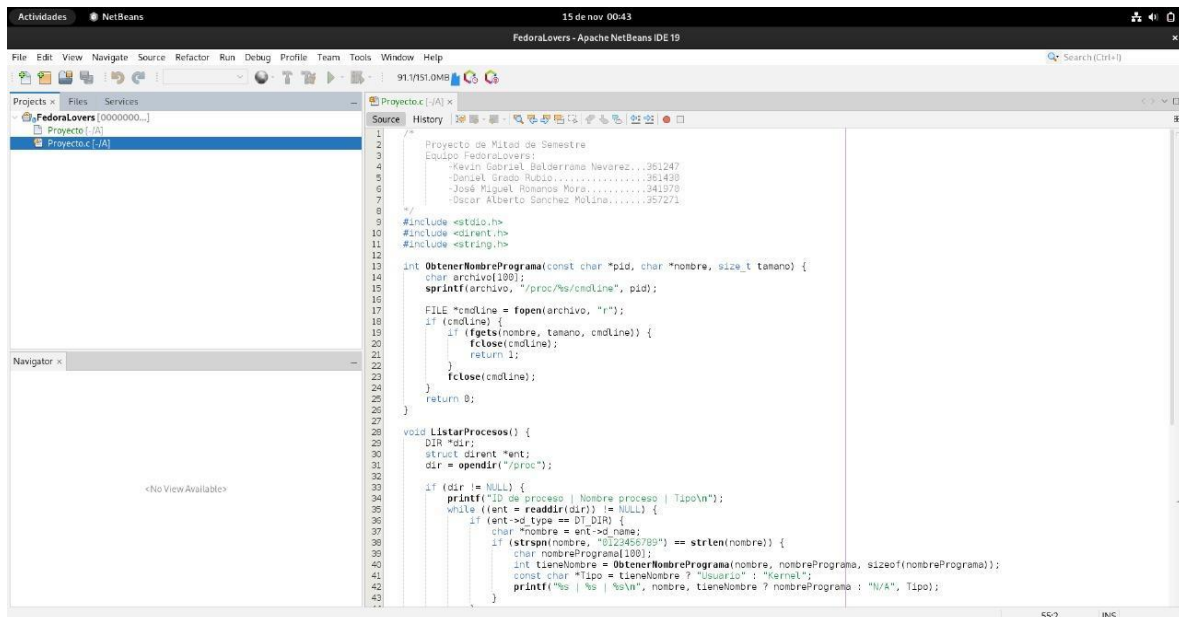
Recibe el ID de un proceso (pid), un buffer para almacenar el nombre del programa (nombre), y el tamaño del buffer (tamano).

Construye la ruta al archivo cmdline correspondiente al proceso en el directorio /proc.

Abre el archivo cmdline en modo lectura.

Utiliza fgets para leer el contenido del archivo (que contiene el nombre del programa) en el buffer nombre.

Si la lectura es exitosa, cierra el archivo y devuelve 1 para indicar que se obtuvo el nombre del programa. Si no se pudo obtener, devuelve 0.



```
1  /*
2  * Proyecto de Mitad de Semestre
3  * Equipo FedorLovers:
4  * Kevin Gabriel Balderrama Navarez...361247
5  * Daniel Grado Rubio...361438
6  * José Miguel Romero Mora...341379
7  * Oscar Alberto Sanchez Molina...357271
8  */
9
10 #include <stdio.h>
11 #include <dirent.h>
12 #include <string.h>
13
14 int ObtenerNombrePrograma(const char *pid, char *nombre, size_t tamano) {
15     char archivo[100];
16     sprintf(archivo, "/proc/%s/cmdline", pid);
17
18     FILE *cmdline = fopen(archivo, "r");
19     if (cmdline) {
20         if (fgets(nombre, tamano, cmdline)) {
21             fclose(cmdline);
22             return 1;
23         }
24         fclose(cmdline);
25     }
26     return 0;
27 }
28
29 void ListarProcesos() {
30     DIR *dir;
31     struct dirent *ent;
32     dir = opendir("/proc");
33
34     if (dir != NULL) {
35         printf("ID de proceso | Nombre proceso | Tipo\n");
36         while ((ent = readdir(dir)) != NULL) {
37             if (ent->d_type == DT_DIR) {
38                 char *nombre = ent->d_name;
39                 if (strcmp(nombre, ".") == 0 || strcmp(nombre, "..") == 0) continue;
40                 char nombrePrograma[100];
41                 int tieneNombre = ObtenerNombrePrograma(nombre, nombrePrograma, sizeof(nombrePrograma));
42                 const char *Tipo = tieneNombre ? "Usuario" : "Kernel";
43                 printf("%s | %s | %s\n", nombre, tieneNombre ? nombrePrograma : "N/A", Tipo);
44             }
45         }
46     }
47 }
```

Función ListarProcesos:

Abre el directorio /proc utilizando opendir.

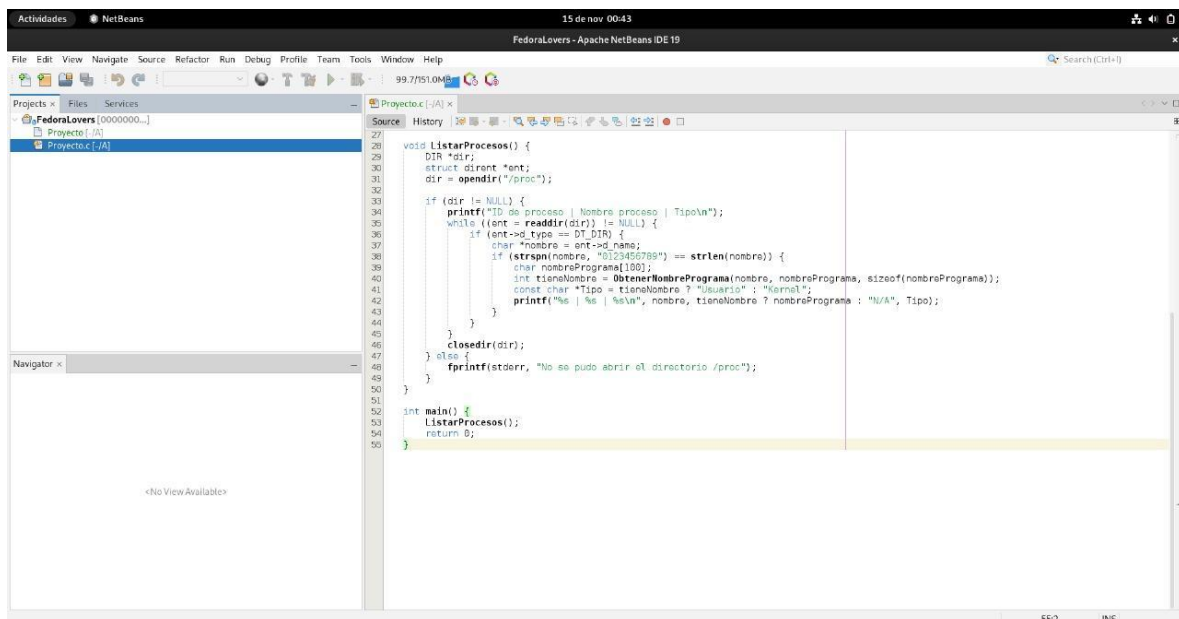
Imprime una cabecera para la salida.

Itera sobre las entradas del directorio usando readdir.

Verifica si la entrada es un directorio y si su nombre consiste solo en dígitos (asumiendo que es un ID de proceso).

Llama a ObtenerNombrePrograma para obtener el nombre del programa y determinar si es un proceso de usuario o del kernel.

Imprime el ID del proceso, el nombre del programa (o "N/A" si no se puede obtener), y el tipo de proceso (Usuario o Kernel).



```
27 void ListarProcesos() {
28     DIR *dir;
29     struct dirent *ent;
30     dir = opendir("/proc");
31
32     if (dir != NULL) {
33         printf("ID de proceso | Nombre proceso | Tipo\n");
34         while ((ent = readdir(dir)) != NULL) {
35             if (ent->d_type == DT_DIR) {
36                 char *nombre = ent->d_name;
37                 if (strlen(nombre, "0123456789") == strlen(nombre)) {
38                     char nombrePrograma[100];
39                     int tieneNombre = ObtenerNombrePrograma(nombre, nombrePrograma, sizeof(nombrePrograma));
40                     const char *Tipo = tieneNombre ? "Usuario" : "Kernel";
41                     printf("%s | %s | %s\n", nombre, tieneNombre ? nombrePrograma : "N/A", Tipo);
42                 }
43             }
44         }
45         closedir(dir);
46     } else {
47         fprintf(stderr, "No se pudo abrir el directorio /proc");
48     }
49 }
50
51 int main() {
52     ListarProcesos();
53     return 0;
54 }
```

Función main:

Llama a ListarProcesos desde la función principal.

Devuelve 0, indicando que el programa se ejecutó correctamente.

En resumen, el programa abre el directorio /proc para iterar a través de los procesos. Para cada proceso, intenta obtener el nombre del programa desde el archivo /proc/pid/cmdline. Luego, imprime el ID del proceso, el nombre del programa (o "N/A" si no se puede obtener), y el tipo de proceso (Usuario o Kernel).



```
51
52
53 int main() {
54     ListarProcesos();
55     return 0;
56 }
```

Subsistemas de Kernel

Utilizamos el archivo de encabezado "dirent.h" para poder acceder a la interfaz de directorios y explorar los procesos "/proc" obtenemos el "pid" o process ID (ID de proceso) y el cmdline.

Para esto se leen los archivos y en caso de que se obtenga un nombre de proceso, este es un proceso de usuario, pero si el nombre de proceso está vacío, entonces es de Kernel.

