



**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: 22/11/2023

Instrucciones:

Crear un programa que muestre:

- Procesos activos del sistema
 - ID de proceso
 - Nombre proceso
- Diferenciar entre procesos de kernel y procesos de usuario
- Total de memoria utilizada.

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
538 | /usr/lib/systemd/system-journald | Usuario
548 | /usr/lib/systemd/system-udev | Usuario
626 | N/A | Kernel
627 | N/A | Kernel
658 | /usr/lib/systemd/system-send | Usuario
659 | /usr/lib/systemd/system-resolved | Usuario
659 | /sbin/auditd | Usuario
657 | /usr/lib/systemd/system-userdbd | Usuario
678 | N/A | Kernel
688 | N/A | Kernel
688 | avahi-daemon: running [fedora.local] | Usuario
698 | /usr/libexec/low-memory-monitor | Usuario
699 | /usr/sbin/mclog | Usuario
699 | /usr/lib/polkit-1/polkitd | Usuario
691 | /usr/libexec/power-profiles-daemon | Usuario
692 | /usr/libexec/rtkit-daemon | Usuario
699 | /usr/libexec/accounts-daemon | Usuario
699 | /usr/libexec/switcheroo-control | Usuario
699 | /usr/lib/systemd/system-logind | Usuario
699 | /usr/lib/systemd/system-machine | Usuario
697 | /usr/libexec/pdsh/pdshd | Usuario
698 | /usr/libexec/tpowerd | Usuario
699 | /usr/sbin/WSX86Client | Usuario
792 | /usr/sbin/zdrd | Usuario
797 | /usr/sbin/WSXService | Usuario
798 | avahi-daemon: chreok helper | Usuario
712 | /usr/bin/dbus-broker-launch | Usuario
745 | /usr/sbin/chromyd | Usuario
747 | dbus-broker | Usuario
798 | N/A | Kernel
793 | /usr/bin/abrt-dump-journal-core | Usuario
799 | /usr/bin/abrt-dump-journal-ops | Usuario
771 | /usr/bin/abrt-dump-journal-xorg | Usuario
770 | /usr/bin/abrt1 | Usuario
786 | /usr/sbin/ModemManager | Usuario
787 | /usr/bin/python3 | Usuario
893 | /usr/sbin/NetworkManager | Usuario
899 | /usr/sbin/cupsd | Usuario
819 | /usr/sbin/gssproxy | Usuario
899 | /usr/sbin/gssproxy | Usuario

2122 | /usr/libexec/gd-printer | Usuario
2124 | /usr/libexec/ibus-engine-simple | Usuario
2366 | /usr/libexec/xdg-desktop-portal | Usuario
2321 | /usr/bin/abrt-dbus | Usuario
2339 | /usr/libexec/xdg-document-portal | Usuario
2349 | fusermount3 | Usuario
2349 | /usr/libexec/xdg-desktop-portal-gnome | Usuario
2359 | /usr/libexec/gd-swttings | Usuario
2408 | /usr/libexec/ibus-x11 | Usuario
2417 | /usr/libexec/mutter-x11-frames | Usuario
2431 | /usr/libexec/dconf-service | Usuario
2436 | /usr/bin/WSXClient | Usuario
2439 | /usr/bin/WSXClient | Usuario
2446 | /usr/libexec/xdg-desktop-portal-gtk | Usuario
2449 | /usr/bin/WSXClient | Usuario
2452 | /usr/bin/WSXClient | Usuario
2558 | /usr/bin/python3 | Usuario
2567 | /usr/libexec/gvfsd-trash | Usuario
2601 | /usr/libexec/tracker-miner-fs-3 | Usuario
2626 | /usr/libexec/gvfsd-recent | Usuario
2629 | /usr/libexec/gvfs-metadata | Usuario
2771 | N/A | Kernel
2801 | N/A | Kernel
2821 | N/A | Kernel
2839 | N/A | Kernel
2844 | /usr/libexec/flatpak-session-helper | Usuario
2849 | sssd | Usuario
2858 | N/A | Kernel
2859 | N/A | Kernel
2872 | N/A | Kernel
3169 | N/A | Kernel
3163 | N/A | Kernel
3176 | N/A | Kernel
3177 | N/A | Kernel
3178 | N/A | Kernel
3181 | N/A | Kernel
3177 | N/A | Kernel
3041 | N/A | Kernel
3062 | N/A | Kernel
3063 | N/A | Kernel
4506 | N/A | Kernel
4516 | /usr/bin/nautilus | Usuario
4547 | /usr/lib/systemd/system-hostnamed | Usuario
4559 | systemd-userwork: waiting... | Usuario
4561 | /usr/libexec/gnome-terminal-server | Usuario
4580 | bash | Usuario
4511 | systemd-userwork: waiting... | Usuario
4512 | systemd-userwork: waiting... | Usuario
4624 | /usr/libexec/tracker-extract-3 | Usuario
4639 | ./Proyecto | Usuario

Memoria Total: 2002116 kB
[tutu@fedora FedoraLovers]$ cat /proc/meminfo
bash: cat /proc/meminfo: No existe el fichero o el directorio
[tutu@fedora FedoraLovers]$
```

Explicación

Función ObtenerMemoriaTotal:

Crea un archivo que tome la dirección de memoria del directorio /proc/meminfo y usamos fopen para leer lo que hay en el directorio /proc.

Utilizamos fgets para leer el contenido del archivo y en caso de que sea diferente de NULL va almacenando la memoria en la variable memoria_total.

Cierra el archivo y regresa el valor de la variable memoria_total.

Función ObtenerNombrePrograma:

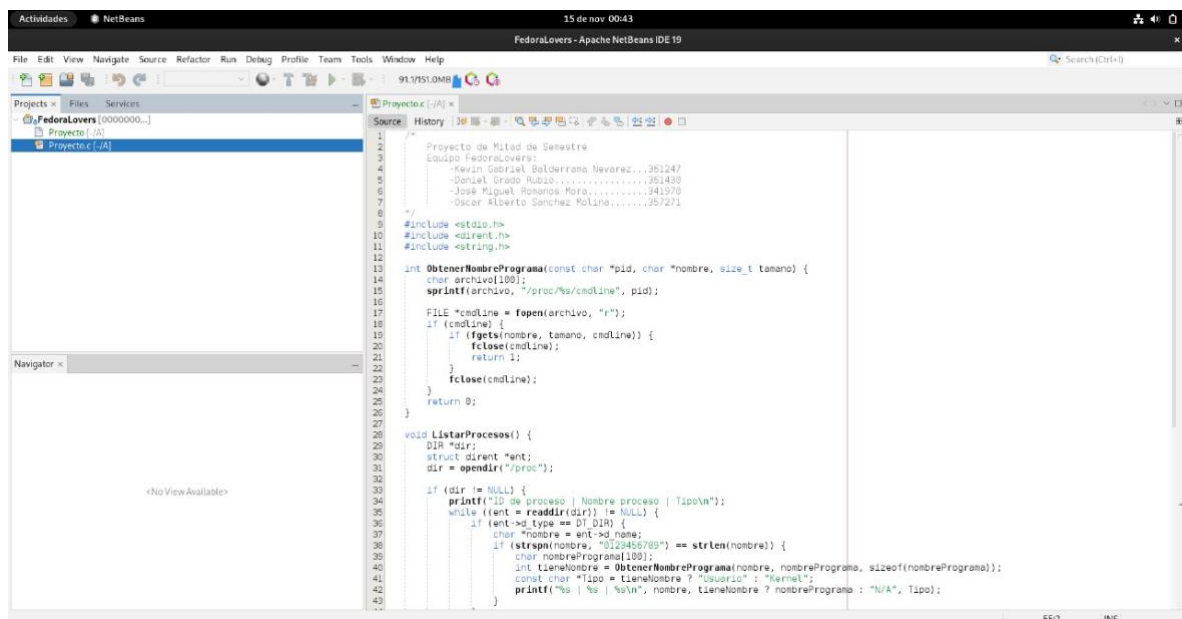
Recibe el ID de un proceso (pid), un buffer para almacenar el nombre del programa (nombre), y el tamaño del buffer (tamano).

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  3  Proyecto de Mitad de Semestre
4  5  Equipo FederaLovers:
6  7  -Kevin Gabriel Balderrama Nevarez...351247
8  9  -Daniel Grado Rubio.....351489
10 11 -Jose Miguel Romeros Mora.....341375
12 13 -Oscar Alberto Sanchez Molina.....357271
14 15 */
16 #include <stdio.h>
17 #include <dirent.h>
18 #include <string.h>
19
20 int ObtenerNombrePrograma(const char *pid, char *nombre, size_t tamano) {
21     char archivo[100];
22     sprintf(archivo, "/proc/%s/cmdline", pid);
23
24     FILE *cmdLine = fopen(archivo, "r");
25     if (cmdLine) {
26         if (fgets(nombre, tamano, cmdLine)) {
27             fclose(cmdLine);
28             return 1;
29         }
30         fclose(cmdLine);
31     }
32     return 0;
33 }
34
35 void ListarProcesos() {
36     DIR *dir;
37     struct dirent *ent;
38     dir = opendir("/proc");
39
40     if (dir != NULL) {
41         printf("ID de proceso | Nombre proceso | Tipo\n");
42         while ((ent = readdir(dir)) != NULL) {
43             if (ent->d_type == DT_DIR) {
44                 char *nombre = ent->d_name;
45                 if (strcmp(nombre, "0123456789") == strlen(nombre)) {
46                     char nombrePrograma[100];
47                     int tieneNombre = ObtenerNombrePrograma(nombre, nombrePrograma, sizeof(nombrePrograma));
48                     const char *Tipo = tieneNombre ? "user" : "kernel";
49                     printf("%s | %s | %s\n", nombre, tieneNombre ? nombrePrograma : "N/A", Tipo);
50                 }
51             }
52         }
53     }
54 }
```

Función ListarProcesos:

Abre el directorio /proc utilizando opendir.

Imprime una cabecera para la salida.

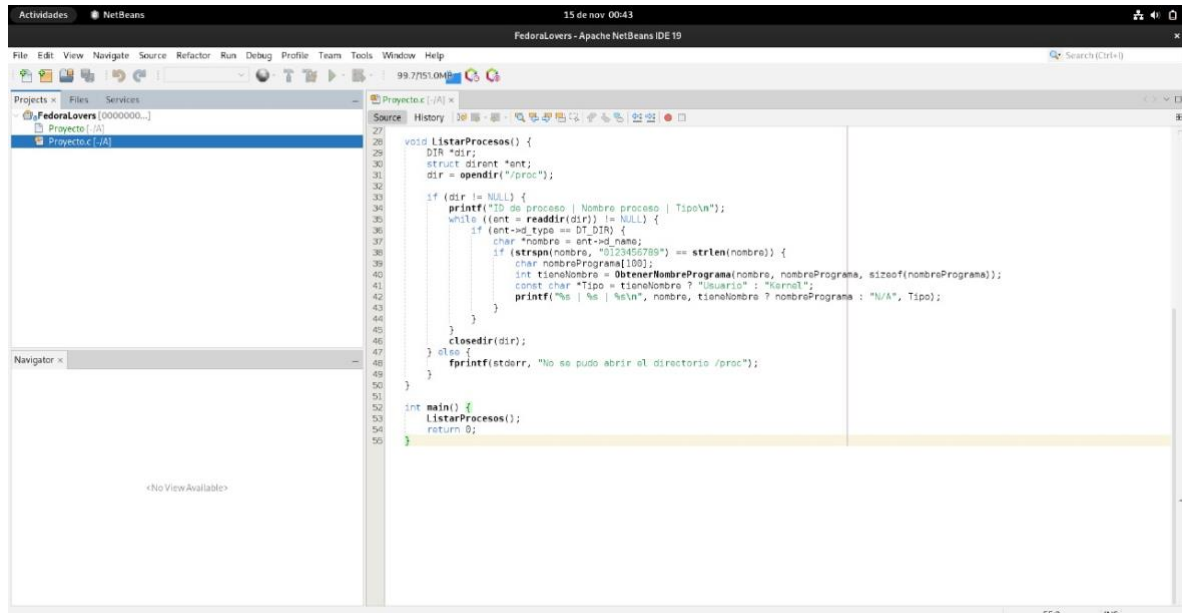
Itera sobre las entradas del directorio usando readdir.

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.

Llama a ObtenerMemoriaTotal para obtener la memoria total y después la muestra.

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 (strcmp(nombre, ".") != 0) {
38                     int tieneNombre = ObtenerNombrePrograma(nombre, nombrePrograma, sizeof(nombrePrograma));
39                     const char *Tipo = tieneNombre ? "Usuario" : "Kernel";
40                     printf("%s | %s | %s\n", nombre, tieneNombre ? nombrePrograma : "N/A", Tipo);
41                 }
42             }
43         }
44         closedir(dir);
45     } else {
46         fprintf(stderr, "No se pudo abrir el directorio /proc");
47     }
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 int main() {
53     ListarProcesos();
54     return 0;
55 }
```

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.



```
1 Proyecto de Mitad de Semestre
2 Equipo FedoraLovers:
3 -Kevin Gabriel Balderrama-Navarez...361247
4 -Daniel Grado Rubio...361439
5 -José Miguel Romanos Mora...341978
6 -Oscar Alberto Sanchez-Rolón...357271
7
8
9 #include <stdio.h>
10 #include <dirent.h>
11 #include <string.h>
12
```