UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD INGENIERIA

Sistemas Operativos I

Proyecto De Kernel

GRUPO: 5HW1

semestre: 5

Docente:

Ivan Miguel Jurado Chavero

Equipo:

Torres Cordova Diego #367587

Cervantes Valenzuela Joan Antonio #351737

Morales Schwartz Kevin Alejandro #367718

Chihuahua, Chih**. 24/10/24**

Este código en C recorre el directorio **`/proc`** de un sistema Linux para listar los procesos que están ejecutándose en el sistema.

\proc es un directorio en Linux que contiene información sobre el sistema y los procesos que se ejecutan en él.

1. **Verificación de si una cadena es numérica**:
   La función **`is_numeric`** revisa si una cadena dada está formada únicamente por caracteres numéricos, lo que es útil para identificar si un nombre de directorio en **`/proc`** corresponde a un ID de proceso (PID), ya que los directorios que contienen información de procesos tienen nombres numéricos.

Numérico: Un nombre de directorio que está formado exclusivamente por dígitos (por ejemplo, **1234, 567**). Estos representan los **PIDs de procesos** activos en el sistema.

No numérico: Un nombre de directorio que contiene letras o símbolos (por ejemplo, **`cpuinfo`, `meminfo`, `sys`)**. Estos representan **información del sistema** o configuraciones, no procesos específicos.

2. **Obtener el nombre de un proceso**:
   La función **`get_process_name`** toma la ruta de un archivo **`stat`** (que contiene estadísticas del proceso) y extrae el nombre del proceso de ahí. Esto se hace leyendo el nombre entre paréntesis en el archivo **`/proc/[PID]/stat`**.

   `/proc`: Es un sistema de archivos virtual que contiene información sobre los procesos y otros datos del kernel.

   `[PID]`: Es el identificador único del proceso que está en ejecución.

`stat`: Es el archivo que contiene estadísticas sobre el proceso, incluyendo información como su estado, uso de CPU, memoria, etc., además del **nombre del proceso**.

Declaración de variables:

`DIR *dir;`: Declara un puntero a una estructura de tipo DIR, que se utilizará para representar el directorio /proc.

`struct dirent *entry;`: Declara un puntero a una estructura dirent, la cual almacena información sobre cada entrada (archivo o subdirectorio) que se encuentre en el directorio /proc.

```
*** if ((dir = opendir(PROC_DIR)) == NULL) ***
```

`opendir(PROC_DIR)`: Abre el directorio /proc, que es un directorio del sistema de archivos en Linux que contiene información sobre los procesos en ejecución. Si opendir no puede abrir el directorio, devuelve NULL.

`if`: Si el directorio no se pudo abrir (es decir, si dir es NULL), se imprime un mensaje de error usando perror y se retorna un valor de salida de 1, lo que indica un fallo.

```
*** while ((entry = readdir(dir)) != NULL) ***
```

`readdir(dir)`: Esta función lee una entrada (archivo o subdirectorio) del directorio abierto y devuelve un puntero a la estructura dirent que contiene información sobre la entrada. El bucle while se ejecuta mientras haya entradas por leer.

```
*** if (entry->d_type == DT_DIR && is_numeric(entry->d_name)) ***
```

Este if verifica dos cosas:

`entry->d_type == DT_DIR`: Si la entrada es un directorio.

`is_numeric(entry->d_name)`: Si el nombre del directorio es completamente numérico, lo que en el contexto de /proc indicaría que corresponde a un ID de proceso (PID).

Declaración de buffers:

```
char cmdline_path[256];

char stat_path[256];
```

Se declaran dos buffers para almacenar las rutas de los archivos /cmdline y /stat de cada proceso.

```
*** snprintf(cmdline_path,sizeof(cmdline_path),"%s/%s%s",PROC_DIR,
                entry->d_name, CMDLINE_FILE); ***
```

`snprintf`: Construye la ruta completa hacia el archivo cmdline dentro del directorio del proceso en /proc/[PID]/cmdline, donde [PID] es el nombre del directorio que está siendo procesado.

```
*** snprintf(stat_path,sizeof(stat_path),"%s/%s%s",PROC_DIR,
                entry->d_name, STAT_FILE); ***
```

Similar a la línea anterior, construye la ruta hacia el archivo stat, ubicado en /proc/[PID]/stat.

```
FILE *cmdline_file = fopen(cmdline_path, "r");
     if (cmdline_file) {
          char cmdline[256];
          size_t length = fread(cmdline, 1,
          sizeof(cmdline), cmdline_file);
          fclose(cmdline_file);
```

Con un puntero declarado como '`*cmdline_fil`e' se abrirá los archivos "`cmdline_parth`". La función Fread lee los datos del archivo colocando el un arreglo y definiendo el tamaño para cada elemento en este caso de 1 byte (tamaño de un carácter) y '`sizeof(cmdline)`' Es el número total de bytes a leer retornando 256, que es el tamaño del arreglo por último '`fread`' devuelve el número real leido de bytes

```
                char process_name[256];
            get_process_name(stat_path, process_name);
```

Esta función se mandan los datos de `stat_path` y `process_name`

```
                    if (length == 0) {
                        printf("[%s] %s [proceso de kernel]\n",
                        entry->d_name, process_name);
                    }
                    else {
                        printf("[%s] %s proceso de usuario\n",
                        entry->d_name, process_name);
                    }
                }
            }
        }
        closedir(dir);
        return 0;
}
```

Por último se comprueba la longitud de los datos del archivo que se haya abierto en caso de que sea igual a 0 serán procesos del Kernel y en caso de que no serán procesos de Usuario y se cierra el puntero “`dir`”