Universidad Autónoma de Chihuahua

Facultad de ingeniería

Ingeniería en sistemas computacionales

Sistemas Operativos

Clave de la materia: 643

Grupo: 5HW1

Semestre: AGO-DIC 2023

Profesor: CHAVERO JURADO IVAN MIGUEL

Proyecto medio semestre

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

Alumno: Alan Garcia (339112)

Fecha de entrega: 15 de noviembre del 2023

Introducción

Este programa en C está diseñado para obtener información sobre los procesos activos del sistema en un entorno basado en Unix, como Linux. Utiliza la biblioteca <dirent.h> para leer el contenido del directorio /proc, que contiene información sobre los procesos en ejecución.

El programa recorre los subdirectorios de /proc, identifica los procesos basados en su ID y obtiene información adicional, como el nombre del proceso y si es un proceso de kernel o de usuario. La información se extrae de archivos específicos en el directorio /proc, como comm para el nombre del proceso y status para el nombre de usuario y otros detalles.

La salida del programa incluye el ID del proceso, el nombre del proceso y se indica si es un proceso de kernel o de usuario. Este código proporciona una visión básica de la administración de procesos en un sistema operativo Unix, pero ten en cuenta que el acceso al directorio /proc puede requerir privilegios adecuados.

Código

```c
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>

struct ProcessInfo {
    char id[10];
    char name[100];
    int isKernel;
};
void displayProcessInfo(const struct ProcessInfo *process) {
    printf("ID de proceso: %s, Nombre del proceso: %s, Tipo de proceso: %s\n",
           process->id, process->name, process->isKernel ? "Kernel" : "Usuario");
}
int isKernelProcess(const char *username) {
    return (strcmp(username, "root") == 0);
}
void getProcessInfo() {
    DIR *dir = opendir("/proc");
    if (dir == NULL) {
        perror("Error al abrir el directorio /proc");
        exit(EXIT_FAILURE);
    }
    struct dirent *entry;
```

```c
    while ((entry = readdir(dir)) != NULL) {

        if (entry->d_type == DT_DIR && atoi(entry->d_name) != 0) {
            struct ProcessInfo process;
            strcpy(process.id, entry->d_name);
            char commPath[256];
            sprintf(commPath, "/proc/%s/comm", entry->d_name);
            FILE *commFile = fopen(commPath, "r");
            if (commFile != NULL) {
                fscanf(commFile, "%s", process.name);
                fclose(commFile);
            } else {
                perror("Error al abrir el archivo 'comm'");
                exit(EXIT_FAILURE);
            }
            char statusPath[256];
            sprintf(statusPath, "/proc/%s/status", entry->d_name);
            FILE *statusFile = fopen(statusPath, "r");
            if (statusFile != NULL) {
                char username[100];
                while (fscanf(statusFile, "%*s %s", username) == 1) {
                    if (strcmp(username, "Uid:") == 0) {
                        fscanf(statusFile, "%s", username);
                        process.isKernel = isKernelProcess(username);
                        break;
                    }
                }
                fclose(statusFile);
            } else {
                perror("Error al abrir el archivo 'status'");
                exit(EXIT_FAILURE);
            }
            displayProcessInfo(&process);
        }
    }
    closedir(dir);
}
int main() {
    printf("Procesos activos del sistema:\n");
    getProcessInfo();
    return 0;
}
```

Funcionamiento

1.-Definición de la Estructura ProcessInfo

Se define una estructura llamada ProcessInfo para almacenar la información de un proceso. Contiene tres campos: id para el ID del proceso, name para el nombre del proceso y isKernel para indicar si es un proceso de kernel.

2.-Función displayProcessInfo

Esta función toma un puntero a una estructura ProcessInfo e imprime la información del proceso en la consola.

3.-Función isKernelProcess

Esta función toma el nombre de usuario como argumento y devuelve 1 si es "root" (indicando un proceso de kernel) y 0 si no lo es.

4.-Función getProcessInfo

Abre el directorio /proc utilizando la función opendir. Si hay algún error al abrir el directorio, imprime un mensaje de error y sale del programa.

Itera sobre los contenidos del directorio /proc usando readdir. Por cada entrada en el directorio, verifica si es un subdirectorio y si su nombre es un número diferente de cero (indicando un proceso).

Para cada proceso, se inicializa una estructura ProcessInfo con el ID del proceso obtenido del nombre del directorio.

Luego, se construye la ruta al archivo comm del proceso y se lee el nombre del proceso desde ese archivo.

Después, se construye la ruta al archivo status del proceso y se lee el nombre de usuario. Se utiliza la función isKernelProcess para determinar si el proceso es de kernel.

La información del proceso se muestra llamando a la función displayProcessInfo.

Finalmente, el programa cierra el directorio /proc.

5.-Función main

En la función principal, se imprime un encabezado y se llama a la función getProcessInfo para obtener y mostrar la información sobre los procesos activos del sistema.