Universidad Autónoma de Chihuahua

Facultad de ingeniería

Sistemas Operativos

Procesos del kernel

Alumnos:

Diego Perez Prieto 365341

Jared Alejandro Rosas Molina 365337

Braulio Sebastian Porras Olivas 344175

Fecha de elaboración: 19/10/2024

Introducción

Este proyecto implementa un programa en lenguaje C que explora el sistema de archivos `/proc`, una interfaz virtual del kernel de Linux que proporciona información sobre los procesos en ejecución. El programa obtiene y procesa datos tanto de procesos de usuario como del kernel, identificando su existencia y nombre mediante la lectura de archivos específicos como `cmdline` y `stat`.

El objetivo principal es demostrar la capacidad de interacción con el sistema de archivos del kernel, manejar diferentes tipos de procesos, y proporcionar un mecanismo robusto para verificar la existencia de archivos en el sistema.

Funcionamiento

El programa cuenta con las siguientes características:

1. Generación de un PID aleatorio: Se genera un número de proceso aleatorio entre 1 y 5000, simulando la selección de un proceso existente en el sistema.
2. Exploración del directorio `/proc`: Se accede al directorio `/proc`, que contiene información sobre todos los procesos en ejecución. El programa verifica si la entrada del directorio es un número válido (que corresponde a un PID).
3. Obtención de información del proceso:
    - Si el proceso es de usuario, se extrae el nombre del proceso y su información desde el archivo `/proc/[PID]/cmdline` y `/proc/[PID]/stat`.
    - Si el proceso es del kernel, se procesa de manera diferente, obteniendo la información desde `/proc/[PID]/stat`.
4. Verificación de la existencia de archivos: Se incluye una función que verifica si el archivo del proceso existe antes de intentar abrirlo, previniendo errores en la ejecución del programa.

Ejecución

El programa se puede compilar y ejecutar de la siguiente manera:

```
gcc -o proc_info proc_info.c
./proc_info
```

Código

```
#include <stdio.h>
#include <dirent.h>
#include <ctype.h>
#include <stdlib.h>
#include <time.h>

void get_process(char *filename, char *procname, int pid);
void archivo_existe(FILE *fp, char *filename);

int main (int argc, char *argv[]) {
        struct dirent *entry;
        DIR *pDir;
        char proc_dir[] = "/proc";
        char proc_id_dir[255];
        char cmdline[255];

        // Genera PID aleatorio entre 1 y 5000
         srand(time(NULL));
         int pid = rand() % 5000 + 1;

    // Intenta abrir el directorio /proc
   pDir = opendir (proc_dir);
    if (pDir == NULL) {
        printf ("Cannot open directory UFF '%s'\n", argv[1]);
        return 1;
    }

    // Procesa la entrada si es que existe
            if((entry = readdir(pDir)) != NULL) {
              if(! isdigit(entry->d_name[0])){
                        printf("El PID es: %i\n", pid);
                        sprintf(proc_id_dir, "/proc/%i/cmdline", pid);
                        get_process(proc_id_dir, cmdline, pid);
              }
            }

            // Cierra el directorio
    closedir (pDir);
    return 0;
}
```

```
// Funcion para obtener y mostrar informacion del proceso
void get_process(char *filename, char *procname, int pid) {
	FILE *fp;
	char *line = NULL;
	size_t len = 0;
	procname[0] = '\0';
	int unused;
	char comm[1000];

	// Abrir archivo cmdline del proceso
	fp = fopen(filename,"r");
	archivo_existe(fp, filename);

	//Sacar la información del proceso de Usuario.
	if (getline(&line, &len, fp) != -1) {
		printf("El PID: %i, es un proceso de Usuario\n", pid);
		sprintf(procname, "%s%s", procname, line);
		printf ("%i:%s\n", pid, procname);
		fclose(fp);

		// Obtiene mas detalles del proceso desde /stat
		sprintf(filename, "/proc/%i/stat", pid);
		fp = fopen(filename, "r");
		archivo_existe(fp, filename);
		fscanf(fp, "%d %s", &unused, comm);
		printf("Nombre del proceso de Usuario: %s\n", comm);
	//Sacar la información del proceso de Kernel.
	} else {
		//Se cierra el anterior File con filename ./cmdline
		fclose(fp);

		printf("El PID: %i, es un proceso de Kernel\n", pid);
		sprintf(filename, "/proc/%i/stat", pid);

		fp = fopen(filename, "r");
		archivo_existe(fp, filename);
		fscanf(fp, "%d %s", &unused, comm);
		printf("Nombre del proceso de Kernel: %s\n", comm);
	}

	//Libera memoria y cierra el archivo
```

```
        fclose(fp);
        free(line);
}

// Función para verificar la existencia del archivo
void archivo_existe(FILE *fp, char *filename){
   if( fp == NULL ) {
     printf("Error abriendo archivo %s\n", filename);
     exit(2);
   }
}
```

Conclusiones

Este programa es una excelente demostración de cómo interactuar con el sistema operativo Linux a nivel del kernel y obtener información detallada sobre los procesos. Además, es una base sólida para construir aplicaciones más avanzadas que requieran la monitorización y análisis de procesos en tiempo real.