

ALUMNA: Priscila Haide Acosta Cano 361318. GRUPO: 5HW1

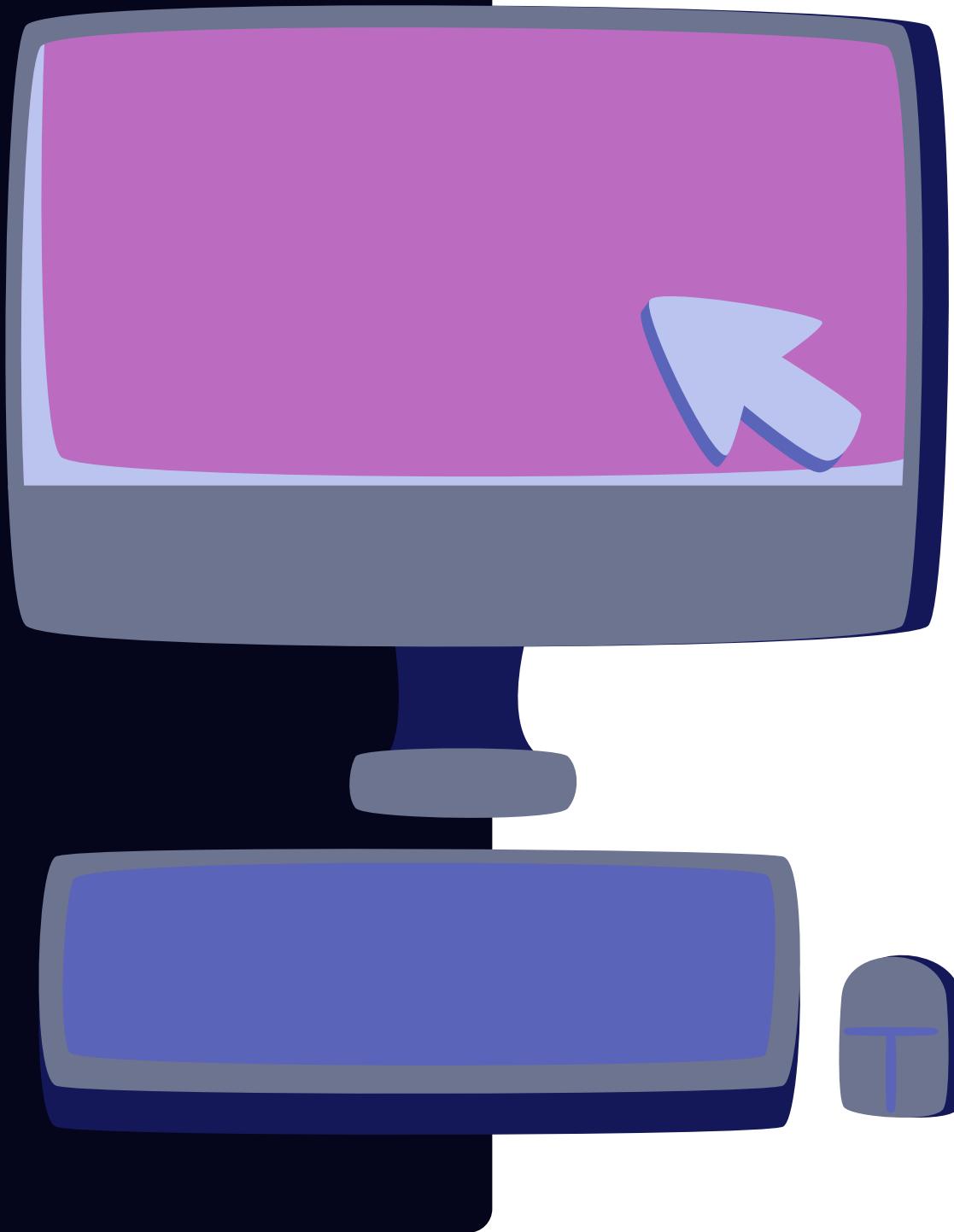
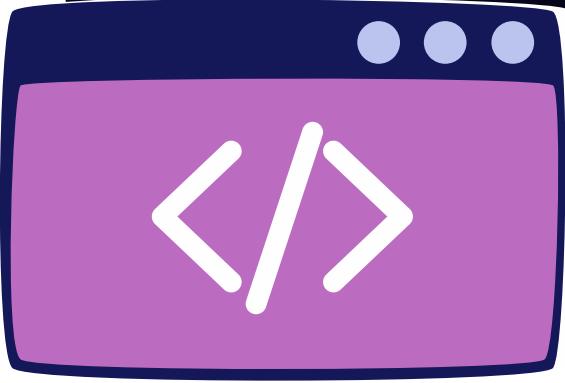
EXAMEN

SISTEMAS OPERATIVOS I

DOCENTE: Ivan Miguel Chavero Jurado.

INTRODUCCION

En este proyecto se tiene como objetivo crear un programa en C, que lee e imprime un archivo utilizando un buffer de tamaño definido. Tambien que de un lista de los procesos activos en el sistema, indicando si son de usuario o del kernel.



DESCRIPCION DEL PROGRAMA

Para la lectura de un archivo se usa fread para leer y fwrite para imprimir.

Para el listado de procesos activos se accede a /proc, ya que identifica procesos por sus UID.

CODIGO FUENTE

```
#INCLUDE <STUDIO.H>
#INCLUDE <STDLIB.H>
#INCLUDE <DIRENT.H>
#INCLUDE <CTYPE.H>

VOID READFILE(CONST CHAR *FILENAME, SIZE.T BUFFER_SIZE) {
    FILE *FILE = FOPEN(FILENAME, "R");
    IF (!FILE) {
        PERROR("ERROR AL ABRIR EL ARCHIVO");
        EXIT(EXIT_FAILURE);
    }

    CHAR BUFFER[BUFFER_SIZE];
    SIZE.T BYTESREAD;
    WHILE ((BYTESREAD = FREAD(BUFFER, 1, BUFFER_SIZE, FILE)) > 0) {
        FWRITE(BUFFER, 1, BYTESREAD, STDOUT);
    }

    FCLOSE(FILE);
}

VOID LIST_PROCESSES() {
    DIR *DIR;
    STRUCT DIRENT *ENTRY;
    DIR = OPENDIR("/PROC");

    IF (!DIR) {
        PERROR("NO SE PUDO ABRIR /PROC");
        EXIT(EXIT_FAILURE);
    }
}
```

```
PRINTF("\NPROCESOS ACTIVOS:\N");

WHILE ((ENTRY = REaddir(DIR)) != NULL) {
    IF (ISDIGIT(ENTRY->D_NAME[0])) {
        PRINTF("PID: %S\n", ENTRY->D_NAME);
    }
}

CLOSEDIR(DIR);

INT MAIN(INT argc, CHAR *argv[]) {
    IF (argc != 3) {
        PRINTF("USO: %S <ARCHIVO> <TAMANO_BUFFER>\N", argv[0]);
        RETURN 1;
    }

    CONST CHAR *FILENAME = argv[1];
    SIZE.T BUFFER_SIZE = atoi(argv[2]);

    PRINTF("LEYENDO ARCHIVO:\N");
    READFILE(FILENAME, BUFFER_SIZE);

    LIST_PROCESSES();

    RETURN 0;
}
```



EXPLICACION DEL CODIGO FUENTE

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

Son las librerías usadas para el manejo de archivos, control de errores, navegación de directorios y manipulación de caracteres.

```
void read_file(const char *filename, size_t buffer_size)
```

Son funciones para leer archivos:

- fopen: abre el archivo.
- Se crea un buffer de tamaño indicado por el usuario.
- fread: lee el archivo en bloques.
- fwrite: imprime cada bloque en pantalla.
- fclose: cierra el archivo al terminar.

```
void list_processes()
```

Son funciones para listar procesos activos:

- opendir: abre el directorio /proc.
- readdir: recorre los elementos dentro de /proc.
- isdigit: identifica si el nombre corresponde a un PID (un número).
- Se imprime el PID.
- closedir: cierra el directorio.



EXPLICACION DEL CODIGO FUENTE

```
int main(int argc, char *argv[])
```

Su función principal es:

- Verificar que el usuario pase los argumentos correctos.
- Llama a read_file y list_processes.

EXPLICACION DE OTRAS FUNCIONES:

- perror: imprime mensajes de error automáticamente.
- atoi: convierte el argumento del buffer de texto a número.
- /proc: sistema de archivos virtual que representa información del sistema en tiempo real.



COMPILACION Y EJECUCION

COMPIALAR:

```
gcc -o read_file read_file.c
```

EJECUTAR:

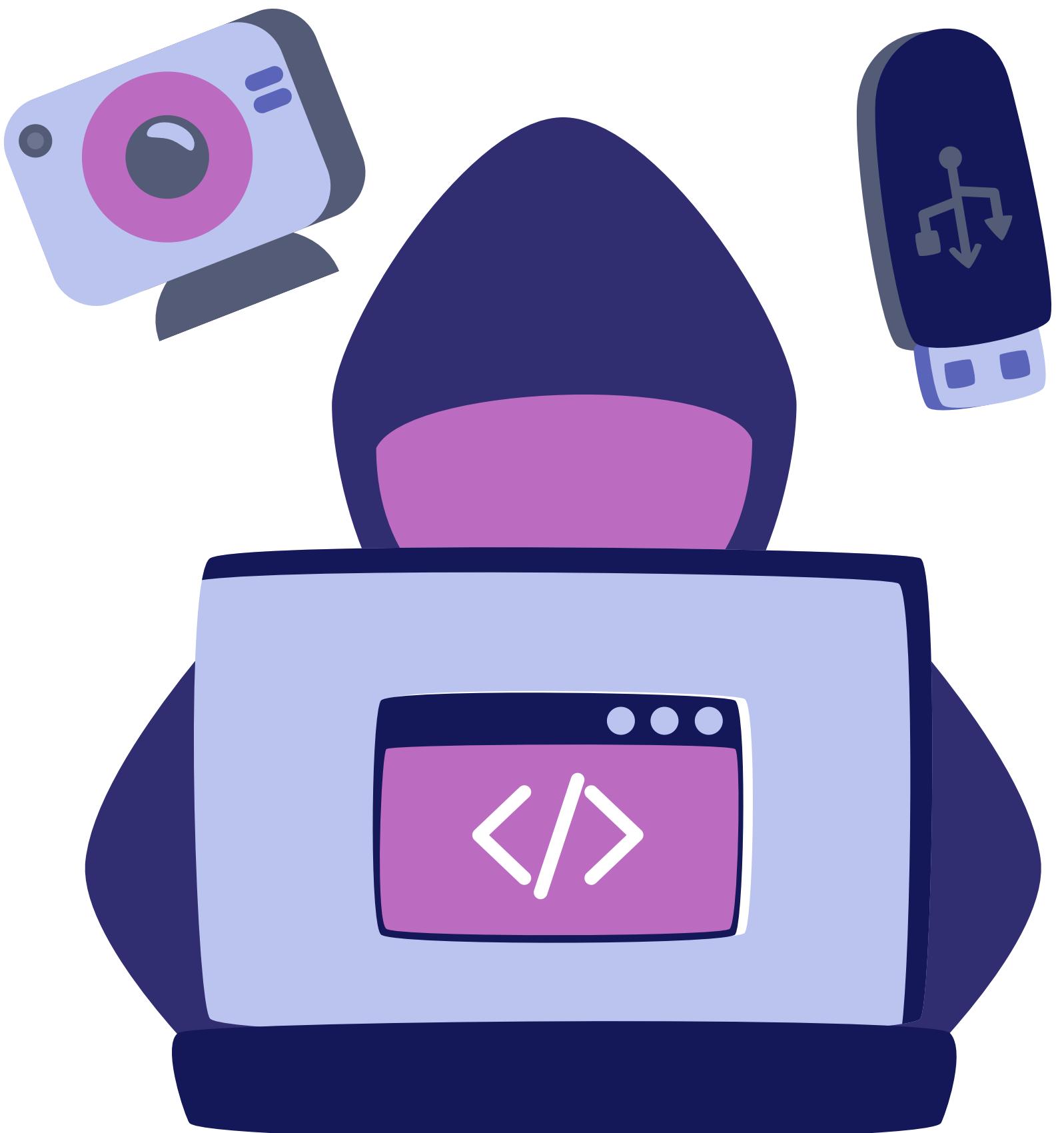
```
./read_file archivo.txt 200
```

EXPLICACION:

archivo.txt es el archivo a leer y 200 es el tamaño del buffer en bytes.

CONCLUSION

En este programa se logro mostrar de manera sencilla como leer archivos usando buffers, como interactuar com el sistema /proc. Tambien permite entender conceptos como el manejo de memoria, la entrada y la salida de datos y la exploracion del sistema operativo a nivel de procesos, ya que como el codigo si funciono logramos aprender todos esos procesos y conceptos.



MUCHAS GRACIAS