OS Assignment-1 of Group No. 44(Section-A) , Prof. Vivek Kumar

Members:-

1) Athiyo Chakma
2) Yash Goyal

Git Clone this link for Access to GitHub Repository:-

https://github.com/Alpha-rgb-cell/OS_Assignments.git

Description:-

ELF Executable Loader Guide

This guide aims to clearly understand the ELF (Executable and Linkable Format) executable loader code. The purpose of this code is to enable the loading and execution of ELF executable files. ELF is a standard format for executables, libraries, and object code in Unix-like operating systems.

Introduction

The ELF Executable Loader is a program designed to facilitate the loading and running of ELF executable files. These files are the ones used for executing software programs. The loader code helps get the program into memory and start its execution.

Getting Familiar with the Code

The code uses global variables to keep track of important information:
- `ehdr`: This pointer stores details about the ELF header.
- `phdr`: Another pointer that stores information about the program header.
- `fd`: This is a unique number known as a file descriptor that helps interact with the ELF executable file.

Resource Cleanup with `loader_cleanup()`

One of the first functions in the code is `loader_cleanup()`. Its job is to ensure that everything is tidy after the code has done its job. This function:
- Checks if the file descriptor `fd` is open, and if so, closes it.
- Frees up memory used to store the ELF header and program header information.

Core Functionality with `load_and_run_elf(char **exe)`

The most important part of the code is the `load_and_run_elf()` function. This function:
- Opens the ELF executable file that's specified as a command-line argument.
- Allocates memory to store the ELF header and reads the header's content from the file.
- Allocates more memory for the program header table and reads its content.
- It goes through the program header entries to find a type known as PT_LOAD containing the information needed to load a segment into memory.
- Allocates a part of memory, similar to reserving space, using a function called `mmap()`. It then loads the segment's content from the file into this memory.
- Figures out where the actual start of the program is and executes a function called `_start` located there.
- Prints out what the `_start` function returns, which can be some value indicating success or an error.
- Cleans up by releasing the memory allocated using `mmap()` and closing the file.

How to Use It

To use this ELF Executable Loader:
- Compile the code using a C compiler.
- Run the compiled program and provide an ELF executable filename as an argument.

Important Note

- The code assumes that the ELF executable file is correctly formatted.
- It is essential to enhance the error handling in a real-world scenario.
- The loader only handles the first segment of a specific type, and real loaders are more sophisticated.

Conclusion

This guide provides a basic understanding of how the ELF Executable Loader code works. It is a practical example of how computers load and execute software programs.

Contributions:-

Athiyo Chakma:- Athiyo Chakma demonstrated proficiency in developing the core functionality of the ELF executable loader. Key areas of contribution include:

1) Code Logic and Implementation
2) Memory Management
3) Execution Flow

Yash Goyal:- Yash Goyal's expertise primarily contributed to robust error handling and code reliability. His contribution include:

1) Error Handling
2) Documentation
3) Quality Assurance