Header Inclusions:

The code includes header files for standard I/O operations, memory management, system calls, ELF header structures, and file operations.

Global Variables:

`Elf32\_Ehdr \*ehdr;` and `Elf32\_Phdr \*phdr;` are pointers to ELF header and program header structures, respectively. These have the info about the loaded ELF file.

`int fd;` represents the file descriptor used to open the ELF executable.

`loader\_cleanup` Function:

This function cleans up allocated memory and resources after the ELF execution is complete.

It checks if memory or file descriptors were allocated.

`load\_and\_run\_elf` Function:

This function loads and runs an ELF executable.

It takes an array of strings (`exe`) as input, where `exe[1]` is the path to the ELF executable.

The function goes through several key steps:

Open and Read ELF Header:

It opens the ELF executable using the provided path and reads the ELF header (`Elf32\_Ehdr` structure) into allocated memory.

Find Loadable Segment:

It iterates through the program header table to find a segment with the `PT\_LOAD` type, which contains the entry point method.

Allocate Memory using `mmap`:

It uses the `mmap` system call to allocate memory of the size specified by the loadable segment's `p\_memsz`.

The memory is set to be readable, writable, and executable.

Copy Segment Content:

It seeks to the offset of the loadable segment within the ELF file and reads its content into the allocated memory.

Calculate Entry Point:

It calculates the entry point address within the loaded segment by adding the offset (`e\_entry - p\_vaddr`).

Call `\_start` Method:

It casts the entry point address to a function pointer type compatible with the `\_start` method and calls it.

Print `\_start` Result:

It prints the return value of the `\_start` method.

`main` Function:

This is the entry point of the program.

It expects a single command line argument (the path to the ELF executable).

It performs necessary checks on the input, invokes the `load\_and\_run\_elf` function, and ensures cleanup with `loader\_cleanup`.

Vaibhav Tapariya: 1. Loading Binary content.

2. Identfying PT\_LOAD Section.

3. Memory Allocation through 'mmap'.

Ashu Kumar Jha : 4. Traversing to Entry Address.

5. Function Pointer Typecasting.

6. Calling \_start Function.