SIMPLE_SMART_LOADER

Functions:

Includes and Global Variables:
The code includes necessary header files like stdio.h, stdlib.h, unistd.h, sys/types.h, and more to access various system calls and data structures.
Several global variables are declared to manage file descriptors, memory allocation, and statistical information.

segv_handler Function:
This function is a signal handler for handling segmentation faults (SIGSEGV).
When a segmentation fault occurs, it increments the pageFaults counter.
It calculates the total pages required for memory allocation based on a program header's p_memsz field, allocates virtual memory, and calculates internal fragmentation.

cleanTheResources Function:
This function is responsible for cleaning up the allocated resources.
It frees memory allocated for the ELF header (ehdr), program header array (phdrArr), and virtual memory (virtual_mem).
It also closes the file descriptor (fd).

readArgs Function:
This function checks the command line arguments to ensure that the correct number of arguments is provided (exactly 1 argument, the ELF file to be executed).

openFileDescriptor Function:
This function attempts to open the ELF file specified in the command line argument.
It stores the file descriptor in the global variable fd for later use.

readEhdr Function:
This function reads the ELF header (Elf32_Ehdr) from the opened file.
It allocates memory for the header and checks for read errors.

main Function:
The program's entry point.

It checks command line arguments, opens the ELF file, and registers a signal handler to capture segmentation faults.
It reads the ELF header and program headers, allocates memory for the program headers, and sets up the program's entry point.
It then executes the program and captures its return value.
Finally, it prints various statistics, including the number of page faults, page allocations, and internal fragmentation.
It calls cleanTheResources to release allocated memory and close the file descriptor.

Contribution Report Contributions:

1. Anish: ● Contribution Percentage: 50% Details: ➔ Made a basic structure of the code, laying down the logic and components. ➔ Focused on the code implementation, from defining global variables to drifting the initial versions of the primary functions.

2. Arav: ● Contribution Percentage: 50% Details: ➔ Worked intensively on error analysis. ➔ Dedicated efforts towards the optimization of the code, making it efficient, and more user friendly.

Final Breakdown:

- Anish: 50%
- Arav: 50% GITHUB :

https://github.com/ianishdev/OS-Assignment.gi