COMP 3980

Assignment 1
Design

Fereshteh Aghaarabi
A01426237
Sep 2025

Purpose

This program accepts one argument from the command line:

- `./elfinspect <filename>`

It inspects the given file and determines if it is a valid ELF binary.
If valid, it prints structured ELF header information (magic number, class, endianness, version, type, machine, entry point).
If invalid, it prints an error message.

Data Types

Field	Type	Description
argc	integer	Number of arguments passed to main()
argv	string[]	The list of arguments (filename)
program_name	string	Name of the program
filename	string	Path to the file being inspected

Settings

Field	type	Description
fd	int	File descriptor from open()
st	Struct stat	File descriptor from open()
ident	Unsigned char[16]	ELF identification bytes
hdr	Unsigned cher[64]	ELF header buffer
e_type	uint16_t	File type (executable, shared, etc.)
e_machine	uint16_t	Target machine architecture
e_version	uint32_t	ELF version
e_entry	uint64_t	Entry point address

Context

Field	Type	Description
arguments	arguments	Command line arguments
settings	Settings	File descriptor and parsed ELF header fields
exit_code	int	Program exit code
exit_message	string	Error message when validation fails

Functions

function	Description
main	Point of insertion
safe_read	Reads N bytes safely, handling short reads
type_to_string	Converts `e_type` numeric value to a string
machine _to_string	Converts `e_machine` numeric value to a string

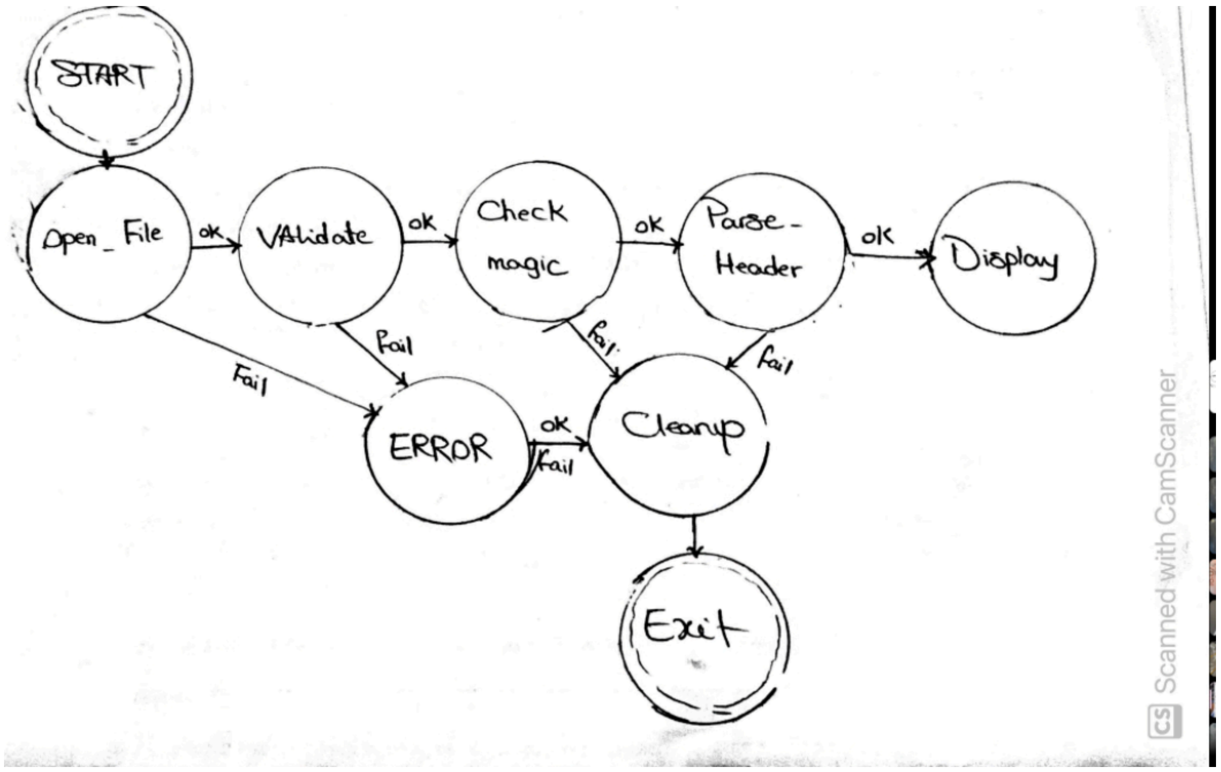
States

State	Description
START	Program begins
OPEN_FILE	Try to open the file
VALIDATE	Check file with fstat,size,and magic
PARSE_HEADER	Read ELF header
DISPLAY	Prints metadata
ERROR	Print error message and exit nonzero
CLEANUP	Close file

State table

From state	To state	Function
START	OPEN_FILE	open()
OPEN_FILE	VALIDATE	fstat()
VALIDATE	ERROR	Print error
VALIDATE	PARSE_HEADER	safe_read()
PARSE_HEADER	DISPLAY	Parse fields
DISPLAY	CLEANUP	printf()
ERROR	CLEANUP	printf()
CLEANUP	EXIT	close()

Diagram



Pseudocode

safe_read

Parameters:

- fd: file descriptor
- buf: pointer to buffer
- count: number of bytes

Return:

- number of bytes read, or -1 on error

```
function safe_read(fd, buffer, count):
    set offset = 0
    while offset < count:
        read some bytes into buffer
        if bytes read > 0:
            add to offset
        else if bytes read == 0:
            break (end of file)
        else if interrupted:
            try again
        else:
            return -1 (error)
    return offset
```

main

Parameters:

- argc: argument count

- argv: argument vector

Return:

- 0 on success, 1 on error

```
function main(argc, argv):
    if no filename given:
        print "Usage"
        return 1

    open the file
    if open fails:
        print error
        return 1

    get file info with fstat
    if fstat fails:
        print error
        close file
        return 1

    if file is not regular or too small:
        print error
        close file
        return 1

    read first 16 bytes into ident
    if magic number is not ELF:
        print error
        close file
        return 1

    check class (32 or 64 bit)
    check endianness
    read full ELF header (52 or 64 bytes)
    parse type, machine, version, entry point

    print all results
    close file
```

```
return 0
```