



KFS_10

The END

Louis Solofrizzo louis@ne02ptzero.me
42 Staff pedago@42.fr

Summary: The End of the world. Or of this series of projects. Can't decide.

Contents

I	Foreword	2
II	Introduction	3
III	Goals	4
IV	General instructions	5
IV.1	Code and Execution	5
IV.1.1	Emulation	5
IV.1.2	Language	5
IV.2	Compilation	6
IV.2.1	Compilers	6
IV.2.2	Flags	6
IV.3	Linking	6
IV.4	Architecture	6
IV.5	Documentation	6
IV.6	Base code	6
V	Mandatory part	7
VI	Bonus part	8
VII	Turn-in and peer-evaluation	9

Chapter I

Foreword

Chapter II

Introduction

This is the last KFS project. You may shed manly tears.
This subject is about making your kernel a complete Unix system.
Nothing specific here.

Chapter III

Goals

At the end of this project, you will have a complete OS. It's about god damn time.

- Fullu functional basic binaries `/bin/*`.
- Libc.
- A Posix Shell.

Chapter IV

General instructions

IV.1 Code and Execution

IV.1.1 Emulation

The following part is not mandatory, you're free to use any virtual manager you want; however, I suggest you use KVM. It's a **Kernel Virtual Manager** with advanced execution and debug functions. All of the examples below will use KVM.

IV.1.2 Language

The C language is not mandatory, you can use any language you want for this series of projects.

Keep in mind that not all languages are kernel friendly though, you could code a kernel in **Javascript**, but are you sure it's a good idea?

Also, most of the documentation is written in C, you will have to 'translate' the code all along if you choose a different language.

Furthermore, not all the features of a given language can be used in a basic kernel. Let's take an example with **C++**:

this language uses 'new' to make allocations, classes and structures declarations. But in your kernel you don't have a memory interface (yet), so you can't use any of these features.

Many languages can be used instead of C, like **C++**, **Rust**, **Go**, etc. You can even code your entire kernel in **ASM**!

So yes, you may choose a language. But choose wisely.



IV.2 Compilation

IV.2.1 Compilers

You can choose any compiler you want. I personally use `gcc` and `nasm`. A Makefile must be turned-in as well.

IV.2.2 Flags

In order to boot your kernel without any dependency, you must compile your code with the following flags (adapt the flags for your language, these are `C++` examples):

- `-fno-builtin`
- `-fno-exception`
- `-fno-stack-protector`
- `-fno-rtti`
- `-nostdlib`
- `-nodefaultlibs`

You might have noticed these two flags: `-nodefaultlibs` and `-nostdlib`. Your Kernel will be compiled on a host system, that's true, but it cannot be linked to any existing library on that host, otherwise it will not be executed.

IV.3 Linking

You cannot use an existing linker in order to link your kernel. As mentionned above, your kernel would not be initialized. So you must create a linker for your kernel.

Be careful, you **CAN** use the `'ld'` binary available on your host, but you **CANNOT** use the `.ld` file of your host.

IV.4 Architecture

The i386 (x86) architecture is mandatory (you can thank me later).

IV.5 Documentation

There is a lot of documentation available, good and bad. I personally think the [OSDev](#) wiki is one of the best.

IV.6 Base code

In this subject, you have to take your previous `KFS` code, and work from it! Or don't. And rewrite everything from scratch. Your call!

Chapter V

Mandatory part

You must install the following:

- A POSIX shell. sh will do.
- The complete libc.
- Basic Unix binaries:
 - cat
 - chmod
 - cp
 - date
 - dd
 - df
 - echo
 - hostname
 - kill
 - ln
 - ls
 - mkdir
 - mv
 - ps
 - pwd
 - rm
 - rmdir
 - sleep

Chapter VI

Bonus part

Install whatever you want. No really, I mean it, make your OS yours.
Have fun :)

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `Git` repository, as usual. Only the work that's in your repository will be graded during the evaluation.

You must turn in your code, a `Makefile` and a basic virtual image for your kernel. Careful about that image, your kernel does nothing with it yet, **SO THERE IS NO NEED TO BE BUILT LIKE AN ELEPHANT**. (More than 10Mo is waaay too much.)