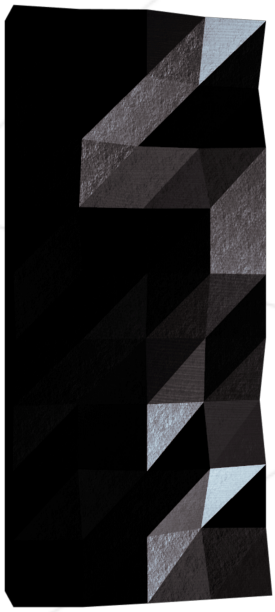

lk_filesystem

LK3: Wow file, such system.

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

Summary:

Version: 1

Contents

I	**Forewords**	**2**
II	**Introduction**	**4**
III	**Goals**	**5**
IV	**General instructions**	**6**
V	**Mandatory part**	**7**
VI	**Bonus part**	**8**
VII	**Turn-in and peer-evaluation**	**9**

Chapter I

Forewords

Genenis of the Bible, Sumerian-first edition:

Gur fabj tybjf juvgr ba gur zbhagnva gbavtug
Abg n sbbgcevag gb or frra.
N xvatqbz bs vfbyngvba,
naq vg ybbxf yvxr V'z gur Dhrra
Gur jvaq vf ubjyvat yvxr guvf fjveyvat fgbez vafvqr
Pbhyqa'g xrrc vg va;
Urnira xabjf V'ir gevrq1

Qba'g yrg gurz va,
qba'g yrg gurz frr
Or gur tbbq tvey lbh nyjnlf unir gb or
Pbaprny, qba'g srry,
qba'g yrg gurz xabj
Jryy abj gurl xabj1

Yrg vg tb, yrg vg tb
Pna'g ubyq vg onpx nalzber1

Yrg vg tb, yrg vg tb
Ghea njnl naq fynz gur qbbe
V qba'g pner
jung gurl'er tbvat gb fnl
Yrg gur fgbez entr ba.
Gur pbyq arire obgurerq zr naljnl3

Vg'f shaal ubj fbzr qvfgnapr
Znxrf rirelguvat frrz fznyy
Naq gur srnef gung bapr pbagebyyrq zr
Pna'g trg gb zr ng nyy

Vg'f gvzr gb frr jung V pna qb
Gb grfg gur yvzvgf naq oernx guebhtu
Ab evtug, ab jebat, ab ehyrf sbe zr,
V'z serr!

Yrg vg tb, yrg vg tb
V nz bar jvgu gur jvaq naq fxl
Yrg vg tb, yrg vg tb
Lbh'yy arire frr zr pel
Urer V fgnaq

Naq urer V'yy fgnl
Yrg gur fgbez entr ba1

Zl cbjre syheevrf guebhtu gur nve vagb gur tebhaq
Zl fbhy vf fcvenyvat va sebmra senpgnyf nyy nebhaq
Naq bar gubhtug pelfgnyyvmrf yvxr na vpl oynfg
V'z arire tbvat onpx, gur cnfg vf va gur cnfg

Yrg vg tb, yrg vg tb
Naq V'yy evfr yvxr gur oernx bs qnja
Yrg vg tb, yrg vg tb
Gung cresrpg tvey vf tbar
Urer V fgnaq
Va gur yvtug bs qnl
Yrg gur fgbez entr ba

Gur pbyq arire obgurerq zr naljnl!

Chapter II

Introduction

Filesystems ! Yay !

> *In computing, a file system (or filesystem) is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins.*
> *By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names is called a "file system".*

Filesystems are kernel-side structures who creates, stores and modifies files / directory. The main purpose of a filesystem is to interact with the hard drive and the system (you know... creating files, chmod, symbolic links, that sort of stuff). Today, filesystems are more considered as an Hard Drive / Kernel interface with 'new' technologies like SSH, Network file system, torrent, etc.
So yeah, filesystem are kinda important for an OS, because, you know... files.
Remember the intro of lk_driver_and_keyboard ?

> *Writing a kernel is only a part of writing an operating system, and to do so, one needs a robust and reliable interface between kernelspace and userspace. This interface is the syscalls.*

Lies, lies and lies. Kernel and Syscalls cannot be considered as an OS if there are no drivers. lk_process_and_mem author is a moron.

Pff, that's bullshit. How can you call something an Operating System if I can't save my sweet zsh config somewhere ? Those subjects authors obviously knows nothing about OS, I tell you.

Chapter III

Goals

Welcome to the marvellous world of filesystems !
Aaaah, those 600 lines headers files, 350-membered structures, quadruple linked-list, childs in childs in father of a non existing file...
Happiness.
Back to the serious stuff, if you play around with this subject, you should be able to:

- Create a new filesystem (from scratch !) in the Linux Kernel
- Understand and work with superblocks
- Understand and work with inodes
- Understand and work with rights, links and interact with other filesystems.
- Link this filesystem to the fabulous world of userspace

Pretty exciting, eh ?

Chapter IV

General instructions

- For this subject, you must use your custom linux distribution, the one you made with the ft_linux subject.
- You must use a kernel version 4.x. Stable or not, as long as it's a 4.x version.
- A Makefile must be turn in
- *ALL* the allocated memory must be properly released. Mind the *PROPERLY*
- *ALL* internal kernel function calls must be verified if needed. Don't want a Kernel Panic, do you ?
- *ALL* of the requests and registers declared in the Kernel must be properly destroyed when the module exits.

Chapter V

Mandatory part

You must create a new filesystem module.
The filesystem must be named "fortytwofs", and this command should work :

```
mount -o loop -t fortytwofs dir image
```

This filesystem should implement the following features (BOTH kernel-side and user-side):

- Directory creation
- File creation
- Owner, group of the entry
- Chmod of the entry
- Links (hard and symbolics)

In this module you MUST use:

- Superblocks. Both Linux and customs.
- Inodes. Both Linux and customs.

With those informations in mind, you have to program a binary that format a disk image for your filesystem. You can use any language you want to build it. This binary must take a disk image as a parameter and prepare superblocks, inodes and all the stuff needed by your filesystem for mounting and run.

Chapter VI

Bonus part

Be creative ! You could, for example, create syscalls / misc device for user-friendly information about your filesystem.
You can also make advanced stats with JSON output for example.
Be creative.

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.

Your code will be running on your custom linux distribution, keep it around for the evaluation.