



CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS
ASSOCIATE REGIONAL CENTER OF NEW-AQUITAINE

THESIS

defended in order to obtain

MASTER'S DEGREE from the CNAM

SPECIALIZATION: Computer Science

OPTION: Networks, Systems and Multimedia

by

Marc BENINCA

**Incremental Live Operating Systems
a reversal of conventional approaches**

Defended on Month DD, 2021

JURY

PRESIDENT:	Title	First	LAST	Role
				Organization
MEMBERS:	Title	First	LAST	Role
				Organization
	Title	First	LAST	Role
				Organization

Acknowledgements

Work conditions

Acronyms

OS Operating System. [6](#)

Glossary

expression english description. [6](#)

Contents

Acknowledgements	2
Acronyms	3
Glossary	4
Contents	5
Introduction	7
1 Problem: maintain operating systems	8
1.1 File systems, installed on partitions, with write access	8
1.1.1 Conventional file systems	8
1.1.2 File systems managed with configuration recipes	9
1.1.3 File systems supporting snapshots	9
1.2 Live images, without installation, with read access	10
1.2.1 Boot without persistent storage	10
1.2.2 Boot with persistent storage	10
1.3 Existing alternatives	10
1.3.1 Windows Unified Write Filter	10
2 Proposal: an incremental live workflow	11
2.1 Implement the workflow	11
2.1.1 Mirror official and third-party repositories	11
2.1.1.1 Synchronize local mirrors	12
2.1.1.2 Select useful architectures	12
2.1.1.3 Check integrity	12
2.1.2 Build a live file system	13
2.1.2.1 Install specific packages	13
2.1.2.2 Create a minimal file system base	13
2.1.2.3 Turn a system into a systems factory	13
2.1.2.4 Turn a file system into a live one	13
2.1.2.5 Install additional packages	13
2.1.2.6 Link specific data to persistent storage	13
2.1.3 Encapsulate in an image file	13

2.1.3.1	Use a format suited for read-only mounting	13
2.1.3.2	Choose a compression algorithm	14
2.1.3.3	Encapsulate in a hybrid image file	14
2.1.4	Secure the produced image file	14
2.1.4.1	Compute an integrity checksum	14
2.1.4.2	Sign to certify authenticity	15
2.1.5	Boot secure image files	15
2.1.5.1	Create standalone boot images	15
2.1.5.2	Create a boot menu	15
2.1.5.3	Check integrity and authenticity	15
2.1.5.4	Load images in random access memory	16
2.1.6	Incremental updating	16
2.1.6.1	Create a new image file	16
2.1.6.2	Avoid an unnecessary reboot	16
2.1.6.3	Reduce the duration of a mandatory reboot	16
2.2	Automate the workflow	16
2.2.1	Check integrity of local repositories	16
2.2.2	Build complete live file systems from scratch	16
2.2.3	Create new files by updating existing images	16
2.2.4	Generate a boot menu on-the-fly	16
3	Results	17
	Conclusion	18
A	Appendix	19
	References	20
	Figures	21
	Tables	22
	Summaries	23

Introduction

Purpose of this work, compared to existing ones in the same domain

[Operating System \(OS\)](#)

[Expressions](#)

[Debian\[1\]](#)

Chapter 1

Problem: maintain operating systems

Depending on the different use cases, maintaining operating systems leads to thinking about:

- updates
- unavailability
- backups policy
- testing backups
- snapshots
- restorations
- configuration recipes

1.1 File systems, installed on partitions, with write access

1.1.1 Conventional file systems

- ext2
- ext3
- ext4
- jfs

- xfs

Pros:

- TODO

Cons:

- TODO

1.1.2 File systems managed with configuration recipes

- ansible
- chef
- puppet

Pros:

- TODO

Cons:

- TODO

1.1.3 File systems supporting snapshots

- btrfs
- zfs

Pros:

- TODO

Cons:

- TODO

1.2 Live images, without installation, with read access

1.2.1 Boot without persistent storage

Pros:

- TODO

Cons:

- TODO

1.2.2 Boot with persistent storage

Pros:

- TODO

Cons:

- TODO

1.3 Existing alternatives

1.3.1 Windows Unified Write Filter

Chapter 2

Proposal: an incremental live workflow

Pros:

- reboot = restore
- update = backup
- separation of system and data

Cons:

- exhaustive manual procedure

2.1 Implement the workflow

2.1.1 Mirror official and third-party repositories

Pros:

- TODO

Cons:

- TODO

2.1.1.1 Synchronize local mirrors

apt-mirror

- translations (Translation-*.bz2)
- architecture independant contents (Contents-all.gz)
- Contents-*/InRelease with some third-party repositories

debmirror

ftpsync

2.1.1.2 Select useful architectures

amd64

arm64

armhf

i386

2.1.1.3 Check integrity

Pros:

- avoid errors during future package installations

Cons:

- no tool exists

2.1.2 Build a live file system

Debian GNU/Linux

2.1.2.1 Install specific packages

Bare metal

Virtual machine

Container

2.1.2.2 Create a minimal file system base

debootstrap

2.1.2.3 Turn a system into a systems factory

2.1.2.4 Turn a file system into a live one

live-boot

update-initramfs

2.1.2.5 Install additional packages

2.1.2.6 Link specific data to persistent storage

2.1.3 Encapsulate in an image file

2.1.3.1 Use a format suited for read-only mounting

SquashFS

2.1.3.2 Choose a compression algorithm

gzip

lzma

lzo

lz4

xz

zstd

2.1.3.3 Encapsulate in a hybrid image file

ISO

2.1.4 Secure the produced image file

2.1.4.1 Compute an integrity checksum

SHA-256

SHA-512

2.1.4.2 Sign to certify authenticity

2.1.5 Boot secure image files

2.1.5.1 Create standalone boot images

GRUB

BIOS

UEFI

Secure boot

2.1.5.2 Create a boot menu

grub.cfg

loopback

squash4

iso9660

2.1.5.3 Check integrity and authenticity

gcry_sha256

gcry_sha512

pgp

2.1.5.4 Load images in random access memory

overlayfs

2.1.6 Incremental updating

2.1.6.1 Create a new image file

2.1.6.2 Avoid an unnecessary reboot

Pros:

- no down time
- just replay the modifications on the system in memory

Cons:

- TODO

2.1.6.3 Reduce the duration of a mandatory reboot

kexec-tools

2.2 Automate the workflow

2.2.1 Check integrity of local repositories

2.2.2 Build complete live file systems from scratch

2.2.3 Create new files by updating existing images

2.2.4 Generate a boot menu on-the-fly

Chapter 3

Results

Conclusion

Appendix A

Appendix

References

- [1] Raphaël Hertzog and Roland Mas. *The Debian Administrator's Handbook*. Buster. 2020. URL: <https://debian-handbook.info>.

Figures

Tables



Incremental Live Operating Systems a reversal of conventional approaches

CNAM Master's Thesis,
Bordeaux 2021.

SUMMARY

Line 1.
Line 2.
Line 3.

Line 4.
Line 5.
Line 6...

**Key words: one, two, three, four,
five, six, seven, eight.**

RÉSUMÉ

Ligne 1.
Ligne 2.
Ligne 3.

Ligne 4.
Ligne 5.
Ligne 6...

**Mots clés : un, deux, trois, quatre,
cinq, six, sept, huit.**