Backup Software for the Linux Filesystem

A project talk by Martin Sova

project

- Backup software application for a filesystem
- Distributed with the **Linux OS** as a desktop application
- Utilizes **full and incremental** backup schemes
- Chronological access to **snapshots** of older version of filesystem via **GUI**
- Backup made both **locally** and **remotely**

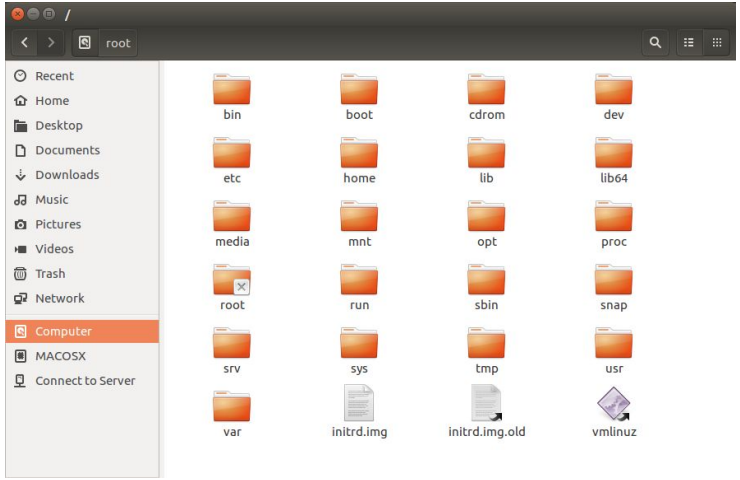
  → connected storage disks *or* to a remote machine

The importance of filesystem backup

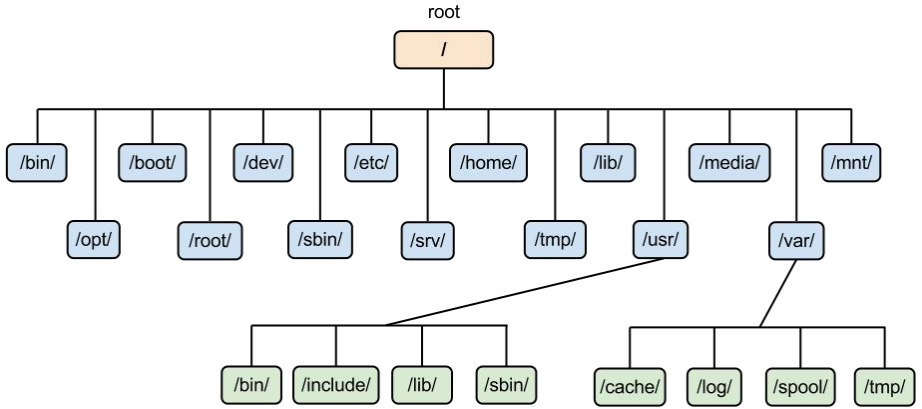
filesystems

- A central component of an operating system
- Enables storage control
- Provides intuitive access for end-users

the linux filesystems



A screenshot of the root directory on my Linux machine.



Typical filesystem hierarchy at root the root directory (Linux example).

A hierarchical filesystem is a tree structure of directories and files.

filesystems expand

Increasingly more difficult to ensure all data is safely stored.

DISK DRIVES

20% increase in **access times and data rates** per year

vs.

50% increase in **capacity** per year

the importance of backup software

- Reading contents from a disk drive and writing to a device takes increasingly longer
- Traditional backup schemes will prove to be too slow
- Developing new software that protects filesystems is a growing challenge
- We observe increased attention from the industrial community

My project

Backend

1 System service

backend

- Exists as a background process (in the Linux OS known as **daemon**)
- Backups can occur even when user logs off
- Initial backup settings read from a configuration file
- User-interaction only via configuration file (**manually** editable or via **GUI**)

2 Hard links

backend

- Copy **only modified files** between backup cycles
- **Hard links** made for **non-modified files** between backup cycles

  → New file with link to the underlying inode

  → Saves disk spaces

  → Recreates **identical** version of filesystem **as viewed by the end-user**

2 Hard links

backend

**insert image* of example hard links from my own computer time machine*

3 Inotify subsystem

backend

- Inode notify (Inotify) is a **Linux kernel subsystem**
- Backend implements **event listener** for **inotify signals** of filesystem changes:

  → File *modification*, *deletion*, *and creation*

- **No need** to **scan entire filesystem** for changes for each backup cycle:

  → Low cost

  → Saves time and resources

- **Modified** files copied & **non-modified** files hard-linked on storage medium

file level storage

backend

- File level (or logical) system **interprets** filesystem metadata and structure
- Writes to device in a **canonical representation** (highly desired for end-users)
- Files stored **contiguously** to enable **rapid single file recovery**

  → Easy user error recovery

- Can backup only a **subset** of data and **omit** certain files:

  → Saves backup time and storage device space

- Highly **resilient** to corruption of disk - files are **self-contained**, unlike disk blocks

full & incremental strategies

backend

- Utilizes a combination of full and incremental backup schemes

  → **satisfy** system requirements and **optimise** backup at a lower cost

- First backup is full backup

- Incremental at prescheduled intervals:

  → Copy only directory structure and modified files

  → Hard link non-modified files

local backup

backend

- Backup to locally connected storage devices
- Registered devices stored in configuration file
- Disk image created on locally connected drive

remote backup

backend

- Use of **rsync shell script**

  → `rsync -a user@remote-host:/source/folder/ /destination_folder`

- **SHH** to connect user to remote-hots

- GNU cp command **--link** argument for hard links

  → Very efficient

  → Net result: series of backups that share common data (desired)

My project

Frontend

end-user

frontend

- **Browsing** through versions of filesystem via **animated UI**
- Recovery of **single** or **multiple** files
- **Entire recovery** of Linux machine

  → Using *dd low level byte-for-byte copy utility*
- Editable **settings**

  → *Backup cycle frequency, storage medium registration, file omission*

design

frontend

- Launched manually or when registered device connect (*event listener in backend service*)
- Settings modification written to configuration file (*read by backend when file modified*)
- Developed using **Qt** widget toolkit