
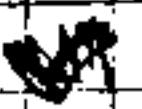





# Cameleonica

## safe cryptographic steganographic advanced filesystem

System engineering dictates a schedule:

• Mission statement is done     

• next phase?

logo is cool, too!

- Needs and premises, goals and objectives?

Conceptual design

- What to do?

- User interfaces

- Functionality

the user can use

specifies

Preliminary design

design/implementation

barrier

- How to do it?

Detailed design

- Internals of how

Construction and production

coding starts

only now

- Implementation

- Sprints no 1+ ?

## Conceptual design

(what does the user can do with it?)

- ① Run application that accesses files on the filesystem. System calls like `open()` `read()` `write()` `mkdir()` have to be implemented.
- ② Open a folder in Gnome/Nautilus. Item overlays, menus, property pages can be provided.

## Sample application:

```
int main():  
    int f  
    f = open("...", "w")  
    write(f, 1)  
    for (int i = 0; i < 3000; i++)  
        write(f, i)  
    close(f)
```

Filesystem obviously has  
to support these operations!

## Usage scenarios:

(contexts where ext4/btrfs can be compared with Camelotica)

These are qualities not functions.

## Synthetic tests

- sequential write then sequential read
- sequential write then random read
- random writes then sequential read
- random writes then random read
- list directory contents
- open/close files
- create/delete empty files
- file command on files
- move files around
- copy small and big files
- getting/setting extended properties
- chown and chown
- work under iocice?

## Utilities testing

- file command
- sha1sum command
- gzip command

## Applications testing

- Deluge (downloading torrents)
- LibreOffice writer and excel (opening and saving)
- Gimp and Inkscape and Blender (opening, saving and rendering)
- Ubuntu system (install and use dash, terminal, etc)
- Ubuntu packages (install and remove several packages)
- PDF Viewer or Image viewer
- Chromium/Firefox usage

## Manual management

- Reverting and reviewing changes to single file
- Reverting and reviewing changes to a folder
- Demanding immediate compression of several files
- Defragmenting on-demand?

## Usability testing

- Open a folder in Nautilus and produce all of thumbnails

## Needs specification:

- POSIX system calls / FUSE calls
  - Ultimately, a filesystem is there to provide user-authorized processes to create, open, read and write arbitrary files.
- Nautilus GUI
  - Browsing revisions, reverting changes, changing settings requires a graphical interface to input commands.
- Command-line control
  - All these control commands can be issued also through the terminal.
- Python code interface
  - shutil-like module for manipulating files including Cameleonica-specific operations
- ~~◦ Linux/POSIX semantics~~
  - ~~◦ Just how system call do things.~~



## Recovery scenario: #1

user starts a program like  
Photoshop

there was no manual snapshot  
made in advance

alternately a snapshot was made,  
manually or regularly or triggered  
by some event

an image gets opened  
for writing

image gets truncated  
down to zero bytes

this open-write-close cycle  
repeats many many times

image gets written with  
many many buffer writes

file gets flushed and/or  
closed

user decides to reverse  
changes to any of previous  
states (versions)



## Recovery scenario #2

user runs a program like shotwell  
or rename command or another that  
operates on many files in series

↓  
a snapshot of the folder/volume  
is/is not \* made

↓  
several images are rotated by shotwell

↓  
several images are renamed (moved really)

↓  
user browses history of entire folder,  
not of individual files

- selects files to be reverted
- \* (not all changes need to be undesired)
- selects a state (point in time) to  
which revert and no further
- can have several states for even  
a single file (several open-write-close cycles)

↓  
user recovers selected state to same folder  
(overriding current content) or to a new folder





### Recovery scenario #3

Cameleonica

↓  
user starts with 2 files  
any or both on Cameleonica-based  
filesystem

↓  
there is not a snapshot

↓  
user starts to copy one file  
onto another, overriding it

↓  
if both reside on same filesystem  
operation should complete immediately

↓  
power loss interrupts operation  
at any point in time

↓  
user browses versions of destination  
file or versions of the folder or  
is presented a notification of failure

↓  
user reverts the destination file  
if he chooses so

Nautilus - enabled

Recovery scenario GUI

for a single file!

General Opening Hashing Recovery

File name: /mydocuments/personal/something.pdf

Size: 130.1 KB Modified: 13 July 2015, 13:05:01

Available revisions:

① Current

○ Opened on (...) closed on (...) changed 57%

○ Opened on (...) closed on (...) changed 1%

○ Changed owner from (...) to (...)

○ Truncated to 0 bytes

○ Allocated 5000 more bytes

○ Created on (...)

○ Copied on (...) from (...)

○ Moved from (...)

○ Snapshotted on (...) in snapshot (...)

○ Reverted from history on (...)

} mutually exclusive  
and at bottom

Action to take:

○ Replace current version (preserves history)

○ Save to another location (zero-copy operation)

Snapshots...

Reload

Revert

Help

Close



Nautilus-enabled  
recovery scenario GUI

for an entire  
Folder



General

Opening

Mashing

Recovery

Directory path: /usr/documents/personal  
Snapshots contained: 8

Available revisions:

- Current
- File (...) opened and changed 31%
- Moved file (...) to (...)
- Copied file (...) from (...)
- Truncated file (...) to 0 bytes
- Snapshot created manually named snap#51
- Reverted from snapshot on (...)
- Snapshot created regularly named auto#50

Action to take:

- Replace current content (extends current history)
- Save to another location (zero-copy operation)

Snapshots...

Reload

Revert

Help

Close

- File moved into folder would be removed without recreating original file outside the folder. File moved outside the folder would be recreated without removing external one.
- Changes made after "Current" (last) checkpoint will be discarded as well as those listed.



# Nautilus-enabled recovery scenario GUI

## File Management of snapshots

General	Opening	Machlog	Snapshots
Create a new snapshot now			
List of available snapshots:			
<ul style="list-style-type: none"><li>● Current</li><li>○ Snapshot #50 manually on (...)</li><li>○ Snapshot #49 regularly on startup</li><li>○ Snapshot #48 automatically on apt-get operation</li><li>○ Snapshot of current directory /mydocs</li><li>○ Snapshot of parent path /</li><li>○ Snapshot of child path /mydocs/parent</li></ul>			
Rename		Remove...	
<ul style="list-style-type: none"><li>○ Replace current content</li><li>○ Save to another location</li></ul>			
Checkpoints...		Reload	Revert
Help		Close	

