Peter Ardivson requests

RetroX name matching
RetroX cloud savings
RetroX network storage
...

BitaBit Extra
bitabit.catrinlabs.cl

RetroX games names matching

How a ROM name is matched in the games database?

- RetroX uses a technique called **Normalization**
- Each name is transformed to a normalized code
- The code is built from all the letters and numbers with no spaces
- All extra marks like (JP) (!) etc, are removed

```
Alien vs Predator     => alienvspredator
Alien vs. Predator    => alienvspredator

007 GoldenEye         => 007goldeneye
007: GoldenEye        => 007goldeneye
007 - GoldenEye       => 007goldeneye

Atari Tetris [!] (EU) => ataritetris
Atari Tetris (B)      => ataritetris
```

How a ROM name is matched in the games database?

- There are a few exceptions
  - **MAME** and **FBNeo** already use encoded names like **mslug**.zip
  - **ScummVM** has codes for all the supported games (retrox.tv/scummvm-games)

Conquests of the Longbow – The Legend of Robin Hood (Floppy DOS)	longbow
Cruise for a Corpse (VGA DOS)	cruise
Darby the Dragon	darby
Day of the tentacle	tentacle
Discworld	dw
Discworld II: Missing Presumed...!?	dw2
Dr. Seuss's ABC	seussabc
Dragon History (Floppy DOS)	draci
Drascula: The Vampire Strikes Back	drascula

How a ROM name is matched in the games database?

- The database only uses the codes and not the original names

```
007goldeneye        for Golden Eye
alienvspredator     for Alien vs Predator
ataritetris         for Atari Tetris
mslug               for Metal Slug (MAME/FBA)
dott                for Day of the Tentacle (ScummVM)
```

How is the game info pulled from the database?

- There are three sources:
  - **RetroX server** (automatic)
    - Artwork: Cover, Screenshots
    - Metadata: Game Description, Rating, Developer, etc
  - **EmuMovies** (downloads to local storage)
    - Artwork: Cover, Screenshots, 3D boxes, Game logos
    - Videos
  - **Local Storage** (manually added or downloaded through EmuMovies)
    - Same as EmuMovies
- RetroX always tries local storage before pulling from the server
- EmuMovies has its own matching methods.
  - The game name is sent “as is”, with no normalization

Possible Bug: Why a game could end at the top of the list?

- The list of games use the file names
- There is no relationship between the codes or the database and the name on the list
- Computers don’t see letters, digits, symbols, they transform everything to bytes
- A byte is just a number
- There are two common ways to transform text to bytes
  - ASCII : 127 codes for all the western text
  - UTF : “a lot of codes” for worldwide text. It’s a bigger ASCII

ASCII Table

Ascii	Char	Ascii	Char	Ascii	Char	Ascii	Char
0	Null	32	Space	64	@	96	`
1	Start of heading	33	!	65	A	97	a
2	Start of text	34	"	66	B	98	b
3	End of text	35	#	67	C	99	c
4	End of transmit	36	$	68	D	100	d
5	Enquiry	37	%	69	E	101	e
6	Acknowledge	38	&	70	F	102	f
7	Audible bell	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Horizontal tab	41	)	73	I	105	i
10	Line feed	42	*	74	J	106	j
11	Vertical tab	43	+	75	K	107	k
12	Form feed	44	,	76	L	108	l
13	Carriage return	45	-	77	M	109	m
14	Shift in	46	.	78	N	110	n
15	Shift out	47	/	79	O	111	o
16	Data link escape	48	0	80	P	112	p
17	Device control 1	49	1	81	Q	113	q
18	Device control 2	50	2	82	R	114	r
19	Device control 3	51	3	83	S	115	s
20	Device control 4	52	4	84	T	116	t
21	Neg. acknowledge	53	5	85	U	117	u
22	Synchronous idle	54	6	86	V	118	v
23	End trans. block	55	7	87	W	119	w
24	Cancel	56	8	88	X	120	x
25	End of medium	57	9	89	Y	121	y
26	Substitution	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	File separator	60	<	92	\	124	|
29	Group separator	61	=	93	]	125	}
30	Record separator	62	>	94	^	126	~
31	Unit separator	63	?	95	_	127	Forward del.

How the game list is sorted?

- The name is treated as a set of UTF bytes
- For western text, it’s the same than ASCII:
  - `spot 23 = 115, 112, 111, 116,  32, 50, 51`
  - `eye 23  = 101, 121, 101, 32, 50, 51`
- Then the everything is sorted using those numbers
  - **101 < 115** so “eye” goes before “spot”

What could go wrong?

- Not all bytes have a visual interpretation
  - ASCII 7 = Rings a bell
  - ASCII 8 = Backspace
- You can’t see them in the file names, but they may be there
- They can make a text be sorted at the top
  - [7]spot : begins with 7
  - eye : begins with e = 101
  - Internally 7 is lower than 101, so spot goes first

How do cloud saves work?

First, how do saves work?

- An emulator is a software that implements hardware
- Hardware has registers and memory
- Registers and memory are just “data” for the emulator
- If you save that data, you save the running state of the machine
  - That’s why it’s called : saved state
- Not all emulators in RetroX support saving the state (MAME, DOS)
- Some system / games can also save the current progress
  - They use their own formats and media (memory cards)
  - They require less storage space than saving a full machine state
  - Those are called in RetroX “saved files”

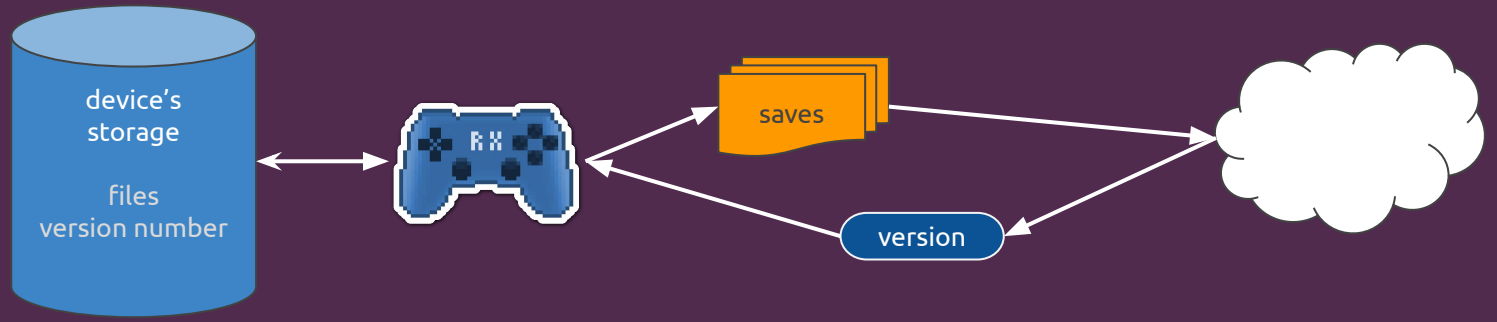
How do cloud saves work?

- Given these saves are files, they can be stored in cloud services
- The server keeps a list of save files for the user
- Each file has a unique incremental number associated to it
  - For RetroX this is the “version number” of the users saves
  - In database lingo, this is called a **sequence**

```
1368 | /savefiles/pcengine/libretro/Falcon/Falcon (USA).srm
1369 | /savefiles/pcengine/libretro/Fighting Run/Fighting Run (Japan).srm
1370 | /savefiles/pcengine/libretro/Final Match Tennis/Final Match Tennis (Japan).srm
1371 | /savefiles/pcengine/libretro/King of Casino/King of Casino (USA).srm
1372 | /savefiles/pcengine/libretro/naxatopen/Naxat Open (Japan).srm
1373 | /savefiles/pcengine/libretro/powergolf/Power Golf (USA).srm
1374 | /savefiles/pcengine/libretro/powertennis/Power Tennis (Japan).srm
1375 | /savefiles/pcengine/libretro/Silent Debuggers/Silent Debuggers (USA).srm
1376 | /savefiles/pcengine/libretro/TV Sports Basketball/TV Sports Basketball (USA).srm
1377 | /savefiles/pcengine/libretro/turrican/Turrican (USA).srm
1378 | /savefiles/pcengine/libretro/World Court Tennis/World Court Tennis (USA).srm
```

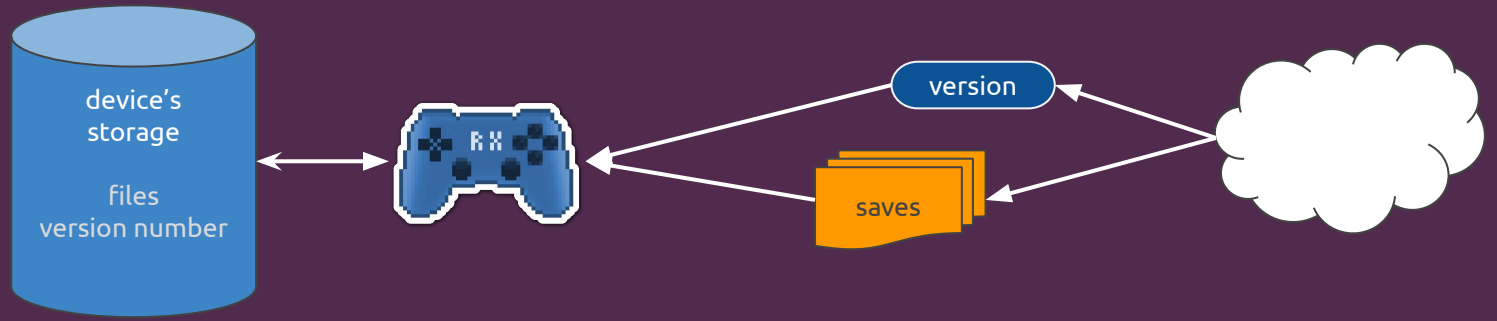
How the saves version number is created?

- RetroX checks for new saves when the game is running
  - If a new save is found, it's added to a list of files to upload
  - As soon as possible, the file is uploaded
- Once finished, a new version number is received from the server
- This version number is stored in the device



How the saves version number is used?

- The last known version number is checked against the one in the server
  - On RetroX boot
  - Before starting any game
- If the server version is higher
  - RetroX downloads all the new files
  - The new version number is stored in the device

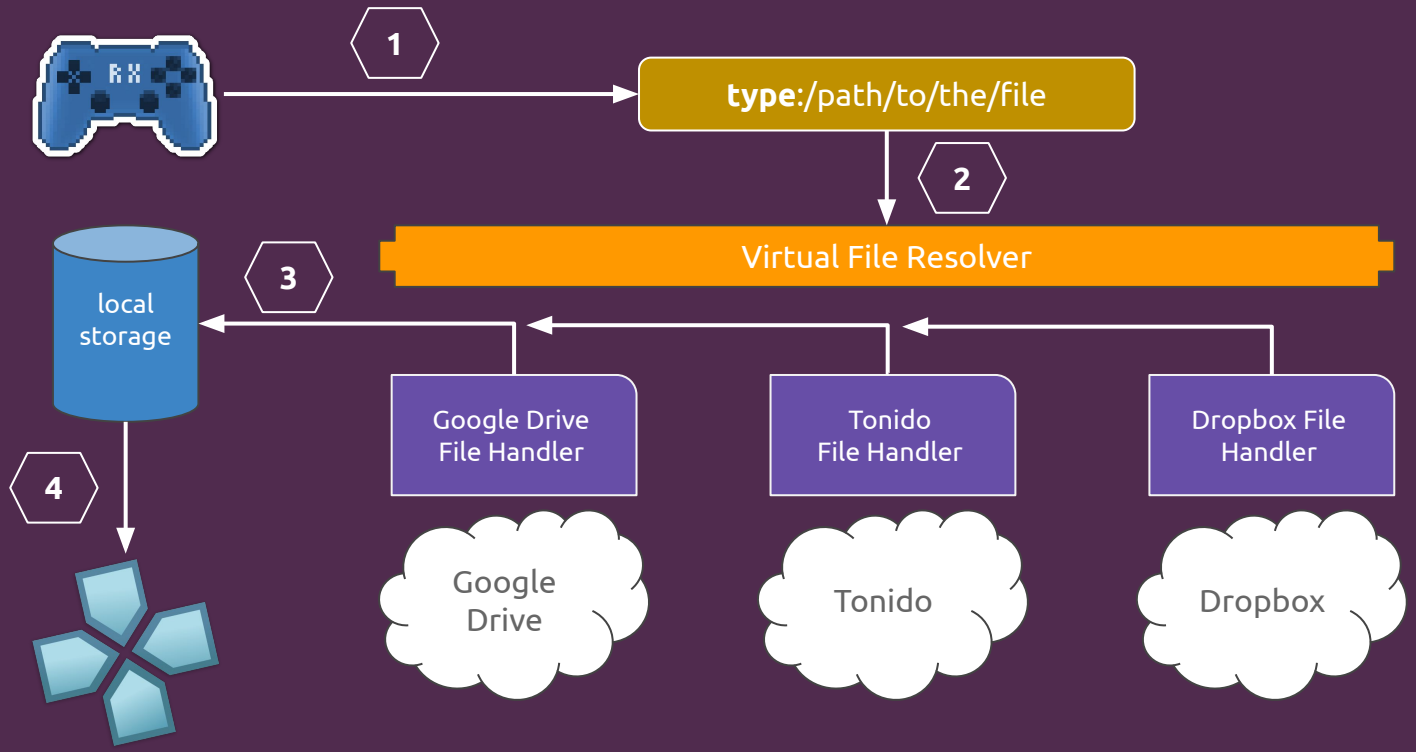


How does network storage work?

Virtual filesystems

- RetroX doesn't know exactly how to handle storage
- Each file has a "location" of the form type:/path
  - **local**:/storage/favorites/genesis/bob.smd
  - **gdrive**:/mygames/amiga/battelsquadron.adf
  - **tonido**:/raspberry/games/pcengine/bomberman.pce
- A Virtual File Handler piece of code is registered to resolve each type
  - `local  -> LocalFileHandler class`
  - `gdrive -> CloudFileHandler class`
  - `tonido -> TonidoFileHandler class`
- This is called an **abstraction**

Resolving a Virtual File Handler



Real code from RetroX

```
private void setupFileHandlers() {
    VirtualFile.addHandler(SystemRootHandler.ROOT_SYS,      new SystemRootHandler());
    VirtualFile.addHandler(SystemRootHandler.ROOT_CLOUD,    new CloudRootHandler());
    VirtualFile.addHandler(SystemRootHandler.ROOT_LAN,      new LANRootHandler());
    VirtualFile.addHandler(SystemRootHandler.ROOT_LOCAL,    new LocalFileHandler(context));
    VirtualFile.addHandler(SevenZFileHandler.ROOT_7Z,       new SevenZFileHandler());
    VirtualFile.addHandler(ZipFileHandler.ROOT_ZIP,         new ZipFileHandler());
    VirtualFile.addHandler(RarFileHandler.ROOT_RAR,         new RarFileHandler(context));
    VirtualFile.addHandler(CloudRootHandler.ROOT_DROPBOX,   new CloudFileHandler(context, ServiceType.DROPBOX, handler));
    VirtualFile.addHandler(CloudRootHandler.ROOT_GDRIVE,    new CloudFileHandler(context, ServiceType.GOOGLE_DRIVE, handler));
    VirtualFile.addHandler(CloudRootHandler.ROOT_ONEDRIVE, new CloudFileHandler(context, ServiceType.ONE_DRIVE, handler));

    VirtualFile.addHandler(TonidoRootHandler.ROOT_TONIDO,            new TonidoRootHandler(context));
    VirtualFile.addHandler(TonidoRootHandler.ROOT_TONIDO_SERVER,     new TonidoFileHandler());
    VirtualFile.addHandler(TonidoRootHandler.ROOT_TONIDO_NEW_SERVER, new TonidoFileHandler());

    VirtualFile.addHandler(LANRootHandler.ROOT_DOMAIN, new DomainFileHandler());
    VirtualFile.addHandler(LANRootHandler.ROOT_SERVER, new ServerFileHandler());
    VirtualFile.addHandler(LANRootHandler.ROOT_SHARE,  new ShareFileHandler());
    VirtualFile.addHandler(LANRootHandler.ROOT_SHARED, new SharedFileHandler());

    CloudServices.getInstance().prepare(activity);
    VirtualFile.setIconResourceIdDefault(R.drawable.ic_insert_drive_file_white_36dp);
}
```

Network storage: The good, the bad and the ugly

- The Good
  - Adding a new source is a matter of writing a new Virtual File Handler
  - Most sources are compatible with this approach
- The Bad
  - External sources are always slower than local storage
  - Emulators need the files to be local, so they must be downloaded to the device
- The Ugly
  - Third parties sometimes break things and handlers must be rewritten
  - Some protocols are horrible, like the ones used in NAS devices

Honorable mention: Nvidia Shield

- The Shield can mount a NAS filesystem at the operating system level
- Apps see these files as local
- The Shield streams the data from the network automatically
- Games reading huge files like CD may experience latency issues
- Note: Most if not all cloud services are not streamables

Thanks!

if you liked this video
join
patreon.com/fcatrin