SDC

STORAGE DEVELOPER CONFERENCE

SNIA SANTA CLARA, 2017

Programming the Path

James Westland Cain, Ph.D
Snell Advanced Media

V 0.9.9 (beta)

Programming the Path – Agenda

- Introduction & Motivation
- What is a Virtual Filesystem (VFS)?
- Types of *Function Call* in a VFS
- Demos - Simple Parameterised Paths
- Path character constraints (in Windows)
- Python on the Path
- Demos - Simple Python Expressions

Programming the Path – Agenda

- Function Composition
- Demos - Complex & Compound Expressions
- Security (really!)
- Analysis & Problems
- Why Do This?
- Conclusions & Future Research

Introduction – James Westland Cain

- Principal Architect – Software @ Snell Advanced Media
- I code every day – as well as being arm waver in chief!
- Been coding for nearly 40 years, at SAM for nearly 20!
- My research interests include file systems innovation and browser based video production.
- PhD in Advanced Software Engineering from Reading University
- Visiting Research Fellow at Brunel University

Introduction – Snell Advanced Media

- SAM makes hardware and software that our customers use to make News & Sports TV & Feature Films
- Our customers include BBC, SKY, Fox Sports, Disney, ESPN, CNN, NBC, NEP, F1 & many others globally
- Most films have been touched by our technologies
- Nearly all 3D movies have been dimensionalised in retrospect using our equipment (sorry)!

Motivation – Programming the Path

- We build video effects & editors
- Mainly written in C++ & Cuda
- Offers a number of interfaces – including a Virtual File System (VFS)
- The VFS is an SMB2/3 server – running in user mode on Windows!
- Last year I embedded Python and added a rich API to the C++
- I have been trying to build a Domain Specific Language (DSL) to allow my Python API to express all the edits and effects we make

Motivation – Domain Specific Language

- The DSL text can be stored & run from files, but what if we could actually embed the language in the file paths we use to access the files in the VFS?
- Looking to build a DSL that could represent edit decisions and effects parameters in file paths
- I tried building a little parser in Python …
- Then I realised it was easier just to *use* Python!

Motivation – Context

- *'What if '*
  … are the most dangerous pair of words in English ☺
- ***It is better to ask for forgiveness, than permission.***

- This is a research idea – not in production!

- Questions & Feedback most welcome!

Virtual Filesystems

- SMB (along with NFS & other NAS protocols) prescribe a set of network function calls that a server needs to implement.
- The implied (assumed) model offers folders full of files, with CRUD semantics.
- As long as they honour the semantics of the model, sever implementers can do *anything they like* whilst answering a network request.

Virtual Filesystems

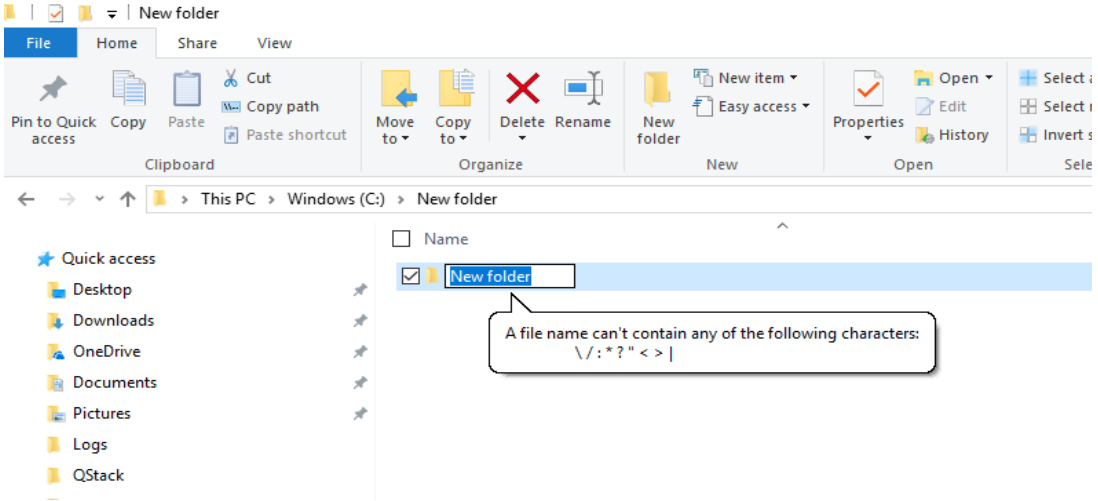
- The contents of a file do not need to be predetermined – as long as they appear stable.
  (C.f. https://en.wikipedia.org/wiki/Turing_test)
- This frees the server to interpret an SMB request as *a function call*.
- The SMB protocol can be seen as an API Gateway

Virtual Filesystems

- Example: On receipt of SMB2_Create, there is no need to be constrained to only offer the files listed in SMB2_Find.
- This frees the server to interpret the folder path – it can contain *state*.
  - Ideas brought from REST and HATEOAS - storing state in the URL.
    https://www.snia.org/sites/default/orig/sdc_archives/2010_presentations/thursday/JamesCain_RESTful_Filesystems.pdf
- So the strings used in the filepath *can have meaning*
- **Demo:** The Server can control a file's contents.

Windows Reserved Characters

- < (less than)
- > (greater than)
- : (colon)
- " (double quote)
- / (forward slash)
- \ (backslash)
- | (vertical bar or pipe)
- ? (question mark)
- * (asterisk)



Non alphanumeric symbols in ascii

- (sp)!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
- Characters not allowed in a file path: \/:<>|*"?
- Allowed: (sp)!#$%&'()+,-.;=@[]^_`{}~
- In Python we can express quite a lot with these!

Example Allowed Python operators

- Function Parameters: () (round brackets)
- Function Parameter Separator: , (comma)
- Strings: ' ' (single quotes)
- Lists & comprehensions: [] (square brackets)
- Sets & comprehensions: () (round brackets)
- Dictionaries & comprehensions: {} (curly brackets)
- (Optional) Statement terminator: ; (semi-colon)
- Equality: == != (equals, not equals)

Demo: Calling Functions

- vlc.exe \\127.0.0.1\quantel\zone-1\clips\ports\compose;timeline()\essence.mxf
- Folder: *compose;timeline()*
- mspaint \\127.0.0.1\quantel\zone-1\clips\stills\compose;t()\1200.bmp
- Folder: *compose;t()*
- Both *timeline()* and *t()* are Python function calls

Folder syntax

- Idiomatic syntax – demonstrates the technique
- *<Python File Name>;<Python Code>*
- The VFS looks for *<Python File Name>.py*
- The VFS passes *<Python Code>* in a variable called *__params__*
- In the Python we call *eval(__params__)*
- **Demo:** open the code

More examples – function parameters

- These are valid Python Folder names:
  - seg(t(),1100,1200)
  - text(t(),'Hello James’)
  - sat(t(),0)

More examples – lists

- These are valid Python Folder names:
  - edit([t(),t()])
  - edit([seg(t(),1100,1200),seg(t(),1100,1200)])

More examples – comprehensions

- These are valid Python Folder names:
  - edit((seg(t(),1100,1200) for x in xrange(100)))
  - edit((seg(t(),800+x,900+x) for x in xrange(0,300,50)))

Security

- This technique can’t possible be secure – can it?
- Sign all the time – no man in the middle
- Encrypt – hide what you’re up to
- Know your clients!!!
  - We’re in a vertical niche market.
- Limit Python built ins …

Security - Making eval() safe

- In Python, eval() expects an expression
- Expressions are constrained so that no assignment is allowed
  - This prevents building up state between calls that might have side effects.
- eval() takes three parameters
  - *Expression* to run
  - Globals
  - Locals
- Limit access to builtins avoids clients running mischievous commands using ‘os’ etc.

Security - Making eval() safe

- #make a list of safe functions
- safe_list = ['text', 'sat', 'e', 'edit', 's', 'seg', 't', 'timeline', 'c', 'clip']
- #use the list to filter the local namespace
- safe_dict = dict([(k, locals().get(k, None)) for k in safe_list])
- #add any needed builtins back in.
- safe_dict['xrange'] = xrange
- # remove the nasty builtin functions
- safe_dict["__builtins__"] = None
- Clip = eval(__params__, safe_dict)

Security – Computer says ‘no’

- **Demo** – bad syntax!
- **Demo** – bad intentions!
- The VFS always has the right to halt proceedings and return an error on receipt of an SMB packet.
- We can even dump the python error stack in the VFS log to aid debugging ☺

Problems

- Disallowed Symbols
- Line Length
- Capitalisation
- Decimal Point has odd semantics (Mime types)

Problem: Disallowed Python operators

- End of if/while statement: : (colon)
  - Mitigation: loop using comprehensions
- Slicing: : (colon)
  - Mitigation: Make sub range function
- Control Blocks – no hard returns with whitespace layout
  - Possible Mitigation: Compound using ; (semi-colon)

Problem: Disallowed Python operators

- Comparison: <> (less or greater than)
  - Mitigation: Make comparison functions
- Strings containing paths: \/ (backslash or forward-slash)
  - Mitigation: (URL) character escaping: %47 %92

- Assignment: = (equals) (not Windows – due to using eval())
  - This is good, as it stops clients changing *persistent* server state.

Problem: Line Length

- Classic Windows limits a filepath to MAX_PATH characters
  - MAX_PATH is defined as 260
- You can add a prefix \\?\UNC\, that extends this to 32,767!
  - **Demo** – long paths.
- Windows 10, version 1607, MAX_PATH limitations have been removed from common Win32 file and directory functions.
- You must opt-in to the new behavior
  https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx
  - Either use the registry to opt in
  - Add a manifest to your program

Problem: Line Length - Bugs

- This data-structure was at the heart of some core processing in our code base.
- Note MAX_PATH in `cFileName.`

```
typedef struct _WIN32_FIND_DATAW {
    DWORD dwFileAttributes;
    FILETIME ftCreationTime;
    FILETIME ftLastAccessTime;
    FILETIME ftLastWriteTime;
    DWORD nFileSizeHigh;
    DWORD nFileSizeLow;
    DWORD dwReserved0;
    DWORD dwReserved1;
    WCHAR cFileName[ MAX_PATH ];
    WCHAR cAlternateFileName[ 14 ];
} WIN32_FIND_DATAW, *PWIN32_FIND_DATAW,
*LPWIN32_FIND_DATAW;
```


Problem: Capitalisation

- Windows NTFS is (generally) case insensitive, but case preserving
  - When used in Linux it is case sensitive too
- Therefore SMB3 honours case as proffered to CreateFile etc
  - Thus strings such as ‘James’, can be passed correctly – *in theory*!
- Not all Windows applications do
  - For example IIS can lower case some URLs before passing them to the filesystem

Problem: Decimal Point semantics

- What does this string mean: (2.5)
  - Is it two and a half in brackets
  - Is it a file called *(2*, with a mime type of *5)*?
- SMB sees it as the first interpretation.
- IIS sees it as the second.

Why do this?

- File systems are pervasive
- We can add *new* functionality to (legacy) applications *without their permission*
- We can be lazy – we can make files *just in time* – rather than *just in case*
- We can be speedy – we can appear to make files instantly, without having to wait for one job to finish before the next can start
- We can start to expose the recipe and ingredients rather than just the pre-baked result
- It enables interesting workflows by empowering the client …
  … breaking the hegemony of the file server!

Future Research

- File systems are destructive
  - The processes & recipes used to build the contents of a file are not stored by normal filesystems.
  - Rendering video is lossy
- Provenance aware filesystems have been discussed in academia.
- Consider how to make *recipe folders* from internal settings?
  - If files are built using recipes, can recipes also be exposed as folders (ie can this be a duplex connection)?

Future Research

- What about other Client OSes: OS-X, Linux?
- Do these folders surprise other client applications?
- What does it mean to *write* through a *recipe folder*?

- I have yet to consider how this technique might be used in production – and what customers might say … ;-)

Conclusions

- Once you have a VFS – many assumptions about file system constraints can be questioned
- I previously (SDC: 2015) talked about Delaying (even Stopping) the VFS server and what benefits that can bring
  http://www.snia.org/sites/default/files/SDC15_presentations/file_sys/JamesCain_A_Pausable_File_System.pdf
- Having programming text in a folder name allows a client to inform a VFS of intent, context, requirements, or even recipes
- The technique does rely on having an underlying model and API which can be manipulated

Questions?

- james.cain@s-a-m.com

Appendix: Example Command lines

- http://127.0.0.1/quantel/zone-1/clips/stills/compose;timeline()/1330.800.jpg
- http://127.0.0.1/quantel/zone-1/clips/stills/compose;sat(timeline(),0)/1330.800.jpg

- mspaint \\127.0.0.1\quantel\zone-1\clips\stills\compose;timeline()\1220.bmp
- mspaint \\127.0.0.1\quantel\zone-1\clips\stills\compose;sat(timeline(),0)\1210.bmp

- mspaint "\\127.0.0.1\quantel\zone-1\clips\stills\compose;text(t(),'hello there')\1110.bmp"
- mspaint "\\127.0.0.1\quantel\zone-1\clips\stills\compose;text(t(),'Hello James')\1120.bmp"

- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;timeline()\essence.mxf
- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;t()\essence.mxf
- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;seg(t(),1100,1300)\essence.mxf
- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;s(t(),1100,1300)\essence.mxf
- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;e(s(t(),1100,1200),s(t(),1100,1200))\essence.mxf
- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;e([s(t(),1100,1150),s(t(),1100,1150),s(t(),1100,1150),s(t(),1100,1150)])\essence.mxf

- "C:\Program Files\VideoLAN\VLC\vlc.exe" "\\127.0.0.1\quantel\zone-1\clips\ports\compose;edit((seg(timeline(),1100,1200) for x in xrange(100)))\essence.mxf"
- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\127.0.0.1\quantel\zone-1\clips\ports\compose;edit((seg(timeline(),800+x,900+x) for x in xrange(0,300,50)))\essence.mxf

- "C:\Program Files\VideoLAN\VLC\vlc.exe" \\?\UNC\127.0.0.1\quantel\zone-1\clips\ports\compose;edit([seg(t(),1100,1200),seg(t(),1100,1200)])\essence.mov