Automatisierung von SAM-FS Abläufen mit Python

Python Binding für die SAM-FS API

Carsten Grzemba
IT-Consultant
contac Datentechnik GmbH
Auf dem Steine 1
98693 Ilmenau
http://www.contac-dt.de

contac

Wer sind wir?

- Systemhaus: 1992 in Thüringen gegründet
- Kunden aus
  - *öffentlichen Bereich,*
  - *Forschungseinrichtungen, Hochschulen,*
  - *Gesundheitswesen,*
  - *Banken & Versicherung,*
  - *Medienhäuser,*
  - *Mittelstand*
- Lösungen:
  - *Netzwerk (Firewall, VPN),*
  - *Storage (SAN,NAS,NFS,CIFS),*
  - *Identitymanagement & Collaboration*
- Partner: Oracle, Telekom, Dell, NetApp, SGI, ...
- betreuen mehrere Jahre verschiedene SAM-FS Installationen

contac

Motivation

- Automatisierung und Integration mit Unix Shell Skripten
- Orchestrierung und Instrumentierung
- keine Datentypen: Ergebnis nur als String, selten Integer
- Verarbeitung durch String-Zerlegung
- System LOCALE-Einstellung abhängig
- keine strukturierten Datentypen oder Datencontainer

sls -D

```
grzemba@samhost$ sls -D SAM-FS-Konferenz.odp
SAM-FS-Konferenz.odp:
  mode: -rw-r--r--  links:   1  owner: grzemba   group: staff
  length:     18821  admin id:      0  inode:    73506.15
  project: system(0)
  copy 2: S----- May 29 07:20    47350e.a2c27 li COS104
  access:      May 29 08:17  modification: May 29 08:17
  changed:     May 29 08:17  attributes:   May 28 09:16
  creation:    May 28 09:16  residence:    May 28 09:16
```

Welche anderen Möglichkeiten haben wir?

- SAM-FS in C geschrieben und hat eine API für ausgewählte Funktionen
- wenig Dokumentation: nur MAN Pages, einige Beispiele
- FS - Funktionen:
  - *stat (sls)*
  - *sam_ssum (Checksum)*
  - *archive, unarchive, stage, release, ...*
- Management-/Wartungs-funktionen
  - *devstat*
  - *getcatalog*
  - *sam_readrminfo*
  - *sam_undamage*
- Migration
- keine Funktionen wie, *showqueue* oder *chmed*
- RPC aufrufbar

Wie mit der C-API arbeiten?

- wenig Dokumentation
- SAM-FS V5.0 2009 von Sun unter der CDDL als OSS freigegeben
- kann nach Anpassung kompiliert werden (ohne Tivoli, STK ACSLS, OSD)
- getestet auf OpenIndiana, ohne BUI (Java-Code ist enthalten)
- statt Dokumentation:
  - ➔ *Sourcecode lesen*
- im kompilierten SAM-FS:
  - ➔ *Funktionen können mit C-Debugger analysiert werden*

Welche weitergehende Möglichkeiten durch OSS?

- durch OSS weitergehende Möglichkeiten
- Problem: private API kann sich in der nächsten Version ändern
- positives Beispiel: NAGIOS NRPE Plugin
  - ➔ *ließt Fault-log und meldet offene Alarme an NAGIOS*
- zufinden: https://github.com/cgrzemba/check_samfs
- DTrace: unabhängig von System Provider samfs (Solaris11)

Wie kann ich die C-API verwenden?

- eigene C-Programme, umfangreiche Entwicklungsumgebung
- besser: höhere Script-Sprachen wie Python oder Perl
- Python ist eine Objektorientierte Skriptsprache
- dynamische Variablen
- Jython als Java-Implementierung
- viele System-Tools werden in Python implementiert, Solaris11:
  - *IPS,*
  - *Timeslider, ZFS Python-Binding*
- Python Funktionen liefern Datentypen/Objekte:
  - *einfachere Verarbeitung der Ergebnisse als in String parsing in Shell Skript*
- bessere Performance

Wie kann ich die C-API mit Python verwenden?

- Python Binding
- verschiedene Tools: Cython (top-down), SWIG, ...
- Cython:
  - ➔ *Erweitern der Möglichkeiten des Python Programm durch Einbinden von C-Code*
- andere Ansatz:
  - ➔ *C-Code ist bereits da, soll eingebunden werden*
- SWIG erscheint für bottom-up Design am geeignetsten

SWIG

- SWIG ist ein Software Development Tool welches C/C++-Programme mit verschiedenen High-level Programmiersprachen verbindet wie:
  - *Perl, PHP, Python, Tcl and Ruby. (CLISP, Allegro CL, CFFI, UFFI), D, Go language, Java including Android, Lua, Modula-3, OCAML, Octave and R*
- 1995 Alamos
- 1996 in Utha überarbeitet
- jetzt Version 2.0.10
- Paket verfügbar über http://www.opencsw.org

Wie kann man ein Python-Binding erzeugen ?

- Anpassen der C-Funktionen in Python Funktionen
- Interface-Beschreibung auf Basis der C-Header
- einfache Typen werden automatisch konvertiert
- C struct's als Named Tuples implementieren -> Templates
- C-API Returnwerte meist Erfolg-Fehler -> Python Exception
- Funktion-Returnwert damit frei für Rückgabewerte (out)
- kompakter gut lesbare Programmcode

Weitere Aufgaben

- wenn Funktion nicht in der API:
  - ➔ *Aufruf der CLI-Tools*
  - ➔ *auch durch den FileManager (Java Webconsole)*
- zusätzliche Funktionalität in nativen Python für Standardaufgaben
- aktueller Stand: https://github.com/cgrzemba/py_samfs
- Dokumentation

Beispiel stage

```
from os import walk
from os.path import join
from samfs import *

basedir = '/samtest'

for root, dirs, files in walk(basedir):
    for file in ((join(root, name)) for name in files):
        if not isOffline(file) or isArchDone(file):
            print "%s %x %d\n" %
              (file,sam_stat(file).attr,sam_stat(file).copies
            sam_stage(file ,'i')
```

Beispiel VSN Migration

```
from subprocess import *
from samfs import *

vsn = 'li:D10001'
path = '/samtest'

for f in getAllFilesForVSN(path,vsn).keys():
    sam_rearch(f,'2','o','c%d' % getAllFilesForVSN(path,vsn)[f])

call(['/opt/SUNWsamfs/sbin/samcmd','arrun'])

while len(getAllFilesForRearch(path,vsn)) > 0:
    sleep(60)

call(['/opt/SUNWsamfs/sbin/sam-recycler'])
```

Fragen ?