Solaris Service Management Facility

Dipl.-Chem. Rainer Orth
Technische Fakultät
Universität Bielefeld
ro@TechFak.Uni-Bielefeld.DE

Übersicht

- Definition Service
- Probleme bisherigen Service-Mangagements
- SMF-Komponenten: Service-Beschreibung, -Management
- Implementierung: Restarter, Repository, Contract

Services auf Unix-Systemen

- üblicherweise langlaufende (Netzwerk-)Dienste
- werden i.d.R. von einem oder mehreren Dämonen bereitgestellt
- aber teilweise auch nur einmalige Kommandos (z.B. `dumpadm`)

Probleme des bisherigen Service-Managements

- Start und Stop sequentiell (`init.d`-Skripte)
- keine echten Abhängigkeiten (nur über `rc?.d`-Skripte)
- keine Übersicht über bereitgestellte Services
- kein Neustart bei Ausfall/Crash
- kein zuverlässiges Logging
- keine zuverlässige Deaktivierung, Konfigurationsänderungen
- keine Delegation der Administration
- kein einfacher Start mit anderen Privilegien

Grundideen von SMF

- eingeführt in Solaris 10
- Service-Beschreibungen mit vollständigen Abhängigkeiten
- Service-Start, sobald Abhängigkeiten erfüllt sind: Parallelisierung
- Fehlerbehandlung:
  - bei Service-Fehler kein Start abhängiger Services
  - separate Logs pro Service
  - automatischer Neustart möglich
- Kompatibilität: Dämonen unverändert, `init.d`-Skripten funktionieren weiter

SMF für Anwender

- `svcs`: Status-Informationen
- `svcadm`: Service-Management: aktivieren, deaktivieren
- `svccfg`: Import, Löschen, Konfigurationsänderungen
- `svcprop`: Anzeige von Properties
- für `inetd`-Services:
  - `inetadm`: Service-Statusabfrage und -Management
  - `inetconv`: Konversion von `inetd.conf`-Einträgen (die einzige inkompatible Änderung durch SMF)

Service-Bezeichner: FMRIs

- Fault Managed Resource Identifier
- allgemein: `svc://localhost/system/cron:default`
- diverse Kurzformen möglich
- bisher nur Host-lokal
- mehrere Instanzen möglich (z.B. mehrere Webserver)

Service-Zustände

- `uninitialized`: Konfiguration noch nicht gelesen
- `disabled`: Service abgeschaltet
- `offline`: Service wartet auf Abhängigkeiten
- `online`: Service läuft
- `degraded`: Service läuft nur teilweise
- `maintenance`: Service konnte nicht gestartet werden, Admin-Eingriff erforderlich
- `legacy_run`: altes `rc?.d`-Skript

Runlevel NG: Milestones

- Milestones hängen von allen Services ab, die in einem Runlevel laufen sollen
- Runlevel `S`, `2` und `3` entsprechen `single-user`, `multi-user` und `multi-user-server`
- dazu noch `none` und `all`
- erreichbar mit `svcadm milestone` *`milestone`* und `boot -m milestone=`*`milestone`*
- daneben noch einige andere, nützlich als Abhängigkeiten: z.B. `name-services`, `network`

Bestandteile eines Service

- Manifest: XML-Beschreibung des Service (in `/var/svc/manifest`)
- Methoden: Start, Stop und optional Refresh des Service, sehr ähnlich den alten `init.d`-Skripten, häufig in `/lib/svc/method`
- Properties: Konfigurationsparameter, Timeouts, Privilegien
- Logfile: alle Ausgaben des Service (in `/var/svc/log/`*`fmri`*`.log`, cf. `svcs -l`)

Beispiel-Manifest: utmpd

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type='manifest' name='SUNWcsr:utmpd'>
<service
        name='system/utmp'
        type='service'
        version='1'>

        <create_default_instance enabled='true' />

        <single_instance/>

        <dependency
                name='milestone'
                grouping='require_all'
                restart_on='none'
                type='service'>
                <service_fmri value='svc:/milestone/sysconfig' />
        </dependency>
```

```
<dependent
        name='utmpd_multi-user'
        grouping='optional_all'
        restart_on='none'>
        <service_fmri value='svc:/milestone/multi-user' />
</dependent>

<exec_method
        type='method'
        name='start'
        exec='/lib/svc/method/svc-utmpd'
        timeout_seconds='60' />

<exec_method
        type='method'
        name='stop'
        exec=':kill'
        timeout_seconds='60' />

<stability value='Unstable' />
```

```
        <template>
                <common_name>
                        <loctext xml:lang='C'> utmpx monitoring
                        </loctext>
                </common_name>
                <documentation>
                        <manpage title='utmpd' section='1M'
                                manpath='/usr/share/man' />
                        <manpage title='utmpx' section='4'
                                manpath='/usr/share/man' />
                </documentation>
        </template>
</service>

</service_bundle>
```

Service-Abhängigkeiten

- Typen (`grouping`): `require_all`, `require_any`, `optional_all`, `exclude_all`
- Fehlerbehandlung (`restart_on`): `none`, `error`, `restart`, `refresh`
- auch umgekehrte Abhängigkeiten möglich (mit `dependent`), auch zur Eingliederung in Milestones

Kontext für Service-Methoden

- Credentials (`user`, `group`, `supp_groups`)
- Privilegien (`privileges`, `limit_privileges`)
- Directory (`working_directory`)
- Ressource-Management (`project`, `resource_pool`)

Delegation der Service-Administration

- möglich über RBAC-Privilegien
- entweder global (`solaris.smf.modify`, `solaris.smf.manage`)
- . . . oder pro Service (`modify_authorization`, `value_authorization`, `read_authorization`, `action_authorization`)

Konfiguration mit Profilen

- XML-Dokumente in `/var/svc/profile`
- momentan nur Aktivierung oder Deaktivierung von Services
- drei vorgegebene: `generic`, `platform`, `site`
- Anwendung mit `svccfg apply`
- in Arbeit: Enhanced Profiles, u.a. Setzen von Properties

Fehlersuche

- schneller Einstieg mit `svcs -x` oder `svcs -xv`
- neben service-spezifischen Logs werden `stdout` und `stderr` im Service-Log (unter `/var/svc/log`) gesammelt
- vermeidet gemischte Ausgabe auf der Konsole

Implementierung

- Kernel startet `init`
- `init` liest `/etc/inittab`, startet `svc.startd` (Master Restarter, startet und überwacht Services)
- `svc.startd` started `svc.configd`, verwaltet SMF-Konfiguration in Repository (`sqlite`-Datenbank in `/etc/svc/repository.db`)
- `svc.startd` startet Services für Ziel-Milestone
- `inetd` als Delegated Restarter für manche Netzwerk-Dienste

Prozeß-Kontrakte

- neue Abstraktion zur Überwachung von Prozessen
- ermöglicht Benachrichtigung bei leerem Kontract, `fork`, `exit`, Coredumps, fatalen Signalen oder Hardware-Fehler
- unabhängig von Eltern-Kind-Beziehung zwischen Prozessen
- implementiert als Filesystem (`/system/contract/process`)
- cf. `ptree -c`, `ctstat`

Weitere Informationen

- OpenSolaris-SMF-Community:
  `http://www.opensolaris.org/os/community/smf/`
- Adams, Bustos, Hahn, Powell, Praza, Solaris Service Management Facility: Modern System Startup and Administration, 19th Large Installation System Administration Conference (LISA '05), p. 225–236
- Praza, Solaris 10 Service Management Facility, SANE 2006
  (`http://mediacast.sun.com/share/lianep/t-smf-sane-may-2006.pdf`)
- Ley, Service Management Facility (smf), Sun System Manager Tagung 2006