Jörg Schilling

Arbeit von Standardisierungsgremien am Beispiel POSIX Shell

Fokus Fraunhofer

Fraunhofer
FOKUS

Dieser Vortrag

- Erklärt nicht, wie z.B. DIN, ISO, IEC, IEEE oder Andere im Allgemeinen arbeiten
- Beschreibt die Arbeit der Austin Group
  - Die Austin Group entwickelt den POSIX Standard
- Die Austin Group ist offen, andere Gremien meist nicht
- Die T10 Standardisierung (SCSI) arbeitet ähnlich, wie man an ihrer offenen Mailingliste sehen kann
- Am Ende gibt es Ausblicke über zukünftige Entwicklungen im POSIX Shell

Wie kam es zur POSIX Standardisierung?

- Am Anfang war UNIX ein privates AT&T Projekt
- Dann wurde UNIX veröffentlicht und unabhängig von mehreren Seiten weiterentwickelt
  - → Divergenz der Eigenschaften
- Eine Konvergenz konnte also nur durch ein anerkanntes Gremium erreicht werden

Die Anfänge von UNIX

- 1969 Das Unics Projekt startet bei AT&T
- 1971 Unics wird innerhalb von AT&T verwendet
- 1972 pipes
- Mit troff → UNIX als „Kapitälchen“
- Um 1977 erste Versionen außerhalb von AT&T
- Frühe 1980er BSD UNIX dominiert an UNIs
- 1982 Sun zuerst mit AT&T UNIX wegen MC 68000 Bugs
- 1984 Sun-2 mit MC 68010 und BSD UNIX
- Andere Kommerzielle meist mit AT&T UNIX (USL)

Vorläufer der heutigen Standaridierung

- 1984 X/Open wird von europäischen Firmen gegründet
- Um 1986 wird IEEE-1003.1 begonnen und der damalige Standardentwurf als Buch herausgegeben
- Ende 1987 schließen Sun Microsystems und AT&T einen Vertrag über SVr4 ab (Basis: SVr3 und SunOS-4.0)
- 1988 OSF (Open Software Foundation) wird als Gegen-spieler zu Sun und AT&T geründet
  - Wichtigste Mitglieder: DEC, IBM, HP
- 1988 UNIX International (UI) wird um Sun und AT&T ge-gen die OSF gegründet

Streit und Annäherungen

- Resultate von UI und OSF: UNIX wars und Tar wars
- 1993 Unix International und OSF günden die COSE (Common Open Software Environment)
- Direkt danach: Zusammenschluß zu einer neuen OSF
- 1993 AT&T verkauft UNIX an Novell
- 1996 Die neue OSF und X/Open werden zur OpenGroup

Strukturen der POSIX Standardisierung

- Früher kam POSIX von der IEEE Institure of Electrical and Electronics Engineers
  - z.B. IEEE-1003.1POSIX 1988, mit wenigen Festlegungen
  - Auch bekannt als ISO/IEC 9945
  - Daher Verfeinerungen von X/Open, die von Behörden zusätzlich zu POSIX gefordert wurden
- Seit 1994 sind OSF und UNIX International vereint
- Seit 1996 sind X/Open und OSF zur OpenGroup vereint
  - The OpenGroup hält heute auch die Rechte am Namen UNIX
- Heute: Austin Group, eine gemeinsame Arbeitsgruppe von IEEE, OpenGroup und ISO/IEC JTC1

Arbeitsmittel der Austin Group

- Eine öffentliche/offene Mailingliste
- Eine private Mailingliste
- Ein öffentliches/offenes Bug Tracking System mit Mantis
- Ein Telekonferenzsystem für wöchentliche Konferenzen
- Ein Etherpad für die Telekonferenzen
- Treffen der Mitglieder (nicht aller Mitarbeiter)

Zeittafel einer neuen POSIX Release

- In der Austin Group laufen „Bug Reports“ an (via Mantis)
- In wöchentlichen Telefonkonferenzen (1,5 Stunden) werden die Bug Reports behandelt und Lösungen erarbeitet
  - Übliche Teilnehmer:
    - Eric Blake, Red Hat
    - Don Cragun, IEEE – früher Sun
    - Geoff Clare, The Open Group, Conformance Test Skripte
    - David Clissold, IBM
    - Roger Faulkner, Oracle
    - Richard Hansen, BBN
    - Andrew Josey, The Open Group, Chair
    - Jörg Schilling, Fraunhofer FOKUS
    - Nick Stoughton, USENIX, ISO/IEC JTC 1
    - Mark Ziegast, Shware Systems

Zeittafel einer neuen POSIX Release

- Die gefundenen Lösungen werden klassifiziert
  - z.B. TC1 (Technical Corrigendum 1) oder neue Release
- Die Lösungen existieren erstmal nur als Deltas
- In regelmäßigen Abständen werden die Ergebnisse abstimmberechtigten Mitgliedern zu Abstimmung vorgelegt
  - Dazu wird ein vorläufiger Komplett-Text produziert
  - Der Komplett-Text ist dem Kern-Team zugänglich
- Ein verabschiedeter Standard der OpenGroup wird dann dem IEEE zur Abstimmung vorgelegt
- Nach Verabschiedung durch IEEE wird der neue gemeinsame Standard herausgegeben

Wer kann bei POSIX mitmachen?

- Jeder kann mitmachen, die OpenGroup ist offen
- Für allgemeine Fragen und Antworten gibt es eine Mailing Liste: austin-group-l@opengroup.org
  - Anmeldung mit Selbstbedienung
- Für das Kernteam aus der Telekonferenz gibt es eine geschlossene Mailing Liste
- Für Bug Reports und Feature Requests gibt es eine Mantis-Installation: http://austingroupbugs.net/
  - Anmeldung auch hier mit Selbstbedienung
- Abstimmungsberechtigung aber nur für Zahlende

Arbeitsweise der POSIX Standardisierung

- Entwicklung der UNIX Varianten beobachten
- Bugreports bearbeiten
- Tendenzen bei diversen Implementierungen erkennen
  - Also auch Programme, die nicht in zertifiziertem UNIX verwendet werden
- Vorhandene Implementierungen dokumentieren
- Features, die im Widerspruch zu vorher Vorhandenem sind, dürfen nicht standardisiert werden
- Bisherige Ausnahmen:
  - POSIX: getpgrp() / setpgrp() ist in Konflikt zu BSD
  - Drepper: fexec() und getline() sind in Konflikt zum älteren UNOS und zur älteren lib-schily

Es gab letztens viele Bugreports zum Shell

- **Allein im letzten Jahr gab es viele Shell Bug-Meldungen**
  - **Einige bitten um neue Features, z.B. 32-Bit Exit codes:**
    http://austingroupbugs.net/view.php?id=1026
  - **Einige beschreiben Defekte im Standard Text, z.B. `alias` unterspezif.**
    http://austingroupbugs.net/view.php?id=953
  - **Einige bitten um eine Lockerung der Syntax, z.B. `for` Syntax:**
    http://austingroupbugs.net/view.php?id=581
- **Im Sommer 2015 wurde festgestellt, daß diverse Shell Implementierungen immer stärker divergieren und es immer schwerer wird, Gemeinsamkeiten zu finden**
- **Ohne Gemeinsamkeiten aber gibt es keine Fortentwicklung in POSIX**

Probleme mit POSIX und dem Shell

- Normalerweise standardisiert POSIX das was am Markt ausreichend einheitlich verfügbar ist
- Es gibt aber viele Shell Implementierungen mit vielen zu einander inkompatiblen Erweiterungen
- Daher muß hier die Initiative von der Standardisierung ausgehen und Einfluß auf alle Shells nehmen
- Die normale Methode, Alles teilweise Inkompatible als „unspecified“ zu bezeichnen wäre unbenutzbar
- Daher werden sinnvolle Shell Erweiterungen aufgegriffen und durch das POSIX Komitee neu formuliert

Probleme diverser Shell Implementierungen

- Eine verwirrende Anzahl von Builtin Kommandos
  - Teilweise statt Standardprogrammen (ksh, busybox)
  - Builtins haben Vorrang und keinen Suchpfad
- Unabgestimmte „set“ Parameter
- Mangelnde Skriptportabilität durch Nutzung v. Features
- Umgang mit Sonderzeichen in Kommandos / Variablen
- Nur 8 Bit vom exit() Code in $?
- $? nicht getrennt vom Child-Ende Code
- Trap ohne Informationen zum Signal

Planungen für POSIX Issue 8

- Unterstützung für die ksh $'...' Expansion
  - C-ähnliche String Escape Darstellung, z.B. $'\n'
  - Unicode Zeichen mit $'\uxxxx'
- Reservierter neuer POSIX Namensraum
  - Auflisten der inkompatiblen Builtins
  - POSIX Flags für set(1)
  - Prüfen oder aktivieren von erweiterten Features
- 32 Bit exit Codes mittels waitid()
  - Vermutlich auch mit Hilfe einer neuen Variablen $/
- siginfo_t Werte in „trap“ Kommandos verfügbar machen

Planungen für POSIX Issue 8

- Set -o pipefail
- Verhalten des Shells mit „trap … EXIT“
- Evt. Korn Shell „[[“ hinzufügen

Vorschlag zu 32-Bit Exit Code im Shell

- **Austin Group Mantis Bug: http://austingroupbugs.net/view.php?id=1026**
- **${.sh.code} Numerischer Grund für Kind-Ende**
- **${.sh.codename} Text aus signal.h für Kind-Ende**
- **${.sh.pid} Prozess ID des Kindes**
- **${.sh.signame} Signalname (bei wait SIGCHLD)**
- **${.sh.status} 32-Bit exit Code**
- **${.sh.termsig} Signalname bei Kind-Tod durch Signal**
- **$/ mit Dezimalzahl bei exit() und Text bei abnormen Ende**

Suchpfade für eingebaute Kommandos

- Alle Shells haben eigene Builtin-Kommandos
- Viele der Kommandos entsprechen normalen UNIX Kommandos mit leichten Abweichungen
- Normale Shell-Regel: ein Builtin hat immer Vorrang
- PATH ermöglicht bei separaten Binaries eine Auswahl
- Für die ksh93-Integration in OpenSolaris wurde PATH für Builtins gefordert und implementiert
- Bei ksh93 mit dem Kommando „builtin“:
  - PATH Zuordnung für jedes Builtin ändern
  - Einzelne Builtins deaktivieren oder reaktivieren

Suchpfade für eingebaute Kommandos

- Die meisten ksh93 Builtins sind dem Pfad `/usr/ast/bin` zugeordnet.
- `/usr/ast/bin` ist nicht im normalen Such-PATH
- Einige wenige Kommandos sind `/usr/bin` zugeordnet
  - Allerdings nur auf Solaris
  - Ein solches Kommando auf Solaris ist `getconf` wegen POSIX `getconf PATH`
- Ein unbedarfter Nutzer sieht diese eingebauten Kommandos nicht, da er seinen PATH nicht modifiziert

Wichtige Shell Implementierungen

- Ksh88 (Closed Source aber Zertifiziert)
- Ksh93 (OpenSource aber nicht komplett konform)
- Bash (OSS aber viele Abweichungen)
- Dash (OSS aber ohne Multi-Byte Support → kein POSIX)
- Mksh (OSS Auf Basis von pdksh – guter ksh „Klon“)
- Bosh (OSS Bourne Shell, fast POSIX konform)
- Andere: (auch zsh) zu viele Abweichungen von POSIX

Danke!

URL: http://cdrtools.sf.net/Files/

Shell URL Sammlung: http://schilytools.sf.net/bosh.html

Fragen?