Unix & Shell-Programmierung SS21

Vorlesungswoche 3

Helga Karafiat

FH Wedel

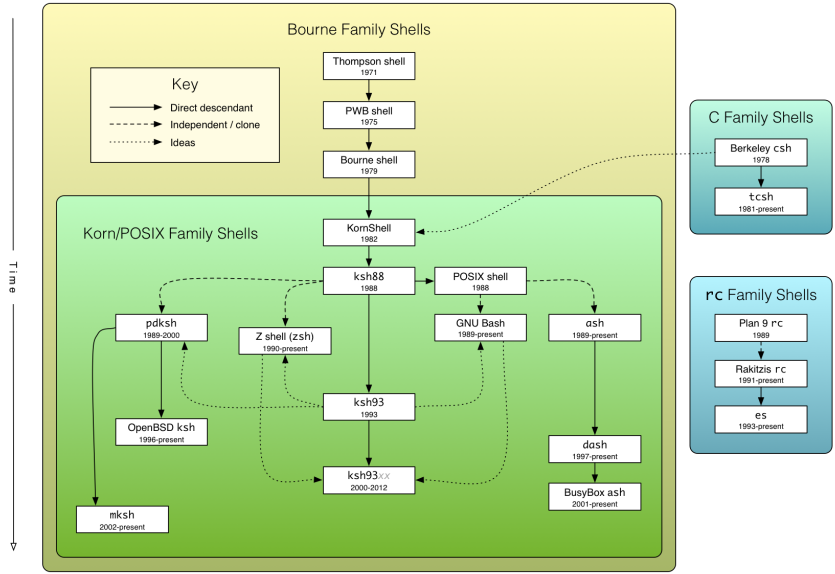
Einordnung von Shells

- Login-Shell:
  - wird zum Einloggen auf dem System verwendet
  - lädt komplexere Konfiguationsdatei(en)
    (üblicherweise erst systemweite Dateien, dann `.profile`, ...)
  - stellt die Umgebung für den Benutzer zur Verfügung
- Non-login Shell:
  - jede Shell, die von einer anderen Shell oder dem X aus gestartet wird
  - weniger Konfigurationsaufwand als Login-Shell (startet schneller)
- Modi:
  - Interactive Shell
    - Shell, die mit dem Benutzer interagiert
    - `stdin` ist üblicherweise mit der Tastatur verbunden
  - Non-interactive Shells
    - Keine direkte Interaktion mit dem Benutzer
    - häufig bei Skripten und automatisierten Abläufen
- Abgrenzung dieser Begriffe ist eher fließend

Welche Muschel?

- `/bin/sh`
  - `sh` ursprünglich Bezeichnung für die Thompson shell, dann für Bourne shell
  - Idee: Standard-Shellimplementierung auf dem System tauschen können, ohne alle Skripte ändern zu müssen (Abwärtskompatibilität)
  - Auf moderen Linux-Systemen per Definition erstmal Softlink auf irgendeine Shell (üblicherweise POSIX-konform)
    - Häufig `bash` oder `dash`
    - Anzeige über `ls -l /bin/sh` oder `file /bin/sh`
  - wird üblicherweise für Skripte verwendet (Shebang)
- `$SHELL`
  - bestimmt die bevorzugte Shell des Benutzers
  - wird als Login-Shell und interaktive Shell gestartet
  - Ausgabe mit `echo $SHELL`
  - Login-Shell kann in der Datei `/etc/passwd` für jeden Benutzer angepasst werden

Entwicklung der Shell-Familien



- Bourne Shell (sh):
  - Standard Unix Shell auf älteren Systemen (closed source)
  - abgelöst durch die KornShell
- KornShell (ksh*):
  - basiert auf und abwärtskompatibel zur Bourne Shell, viele zusätzliche Features aus der C-Shell übernommen
  - ursprünglich closed source, seit 2000 open source unter AT&T eigener Lizenz, seit 2005 unter Eclipse Public License
  - Auf den meisten Unix Systemen vorhanden
  - `oksh` als Port für Linux verfügbar (aber eher exotisch)
  - Funktionsumfang der original KornShell (ksh88) war Basis für den Standard "POSIX.2, Shell and Utilities"

- POSIX (Portable Operating System Interface - ursprünglich IEEE-IX)
  - keine Shell, sondern eine Reihe von Standards für die Gewährleitung von Kompatibilität zwischen verschiedenen Computer-Platformen (Portabilität)
  - für Betriebssystementwickler als auch für Anwendungsentwickler gedacht
  - erste Version: IEEE Std 1003.1-1988, aktuellste Fassung: IEEE Std 1003.1-2017
  - Seit 2001 Weiterentwicklung durch die Austin Group (Gemeinsame Arbeitsgruppe von IEEE, The Open Group und ISO/IEC JTC 1)
  - Unterteilung in 4 Abschnitte: Base Definitions, System Interfaces, Shell & Utilities, Rationale
  - definiert (unter anderem) verschiedene Werkzeugschnittstellen, Befehle und APIs wie:
    - Systemschnittstelle (Funktionen, Makros und externe Variablen)
    - Befehlsinterpreter / Shell (`sh`)
    - Dienstprogramme (wie z.B. `more`, `cat`, `ls`)
  - kleinster gemeinsamer Nenner aller "Unixe", aber auch darüber hinaus
  - Unterscheidung der Systeme in "POSIX-certified" (z.B. HP-UX, Solaris) und "Mostly POSIX-compliant" (die meisten Linuxe, FreeBSD, ...)

- Almquist shell (ash)
  - Open Source Bourne / POSIX Shell Klon für BSD-Systeme
  - minimale POSIX-konforme Shell
- GNU Bourne-again shell (bash)
  - basiert auf der Bourne Shell, ist weitgehend kompatibel zur Kornshell
  - viele bash-eigene built-in Kommandos
    (Erweiterte Bedingungen für bedingte Anweisungen, Brace Expansion, ...)
  - zudem viele nützliche Erweiterungen als interaktive Shell
  - kann im POSIX-Kompatibiltätsmodus betrieben werden (reduzierter Funktionsumfang)
    `bash --posix` (häufig vom OS implizit so aufgerufen, wenn als `/bin/sh` definiert)
  - Standard Login-Shell auf den meisten Linux-Systemen
  - auf vielen Linux-Systemen bis heute auch `/bin/sh`

- Z shell (zsh)
  - weitere moderne auf der KornShell basierende Shell
  - viele eigene Erweiterungen, teilweise ähnlich zu bash, ksh und tcsh, z.B.:
    - Rechtschreibkorrektur
    - invidualisierbarer Promt mit Möglichkeit für extra Infos am rechten Bildschirmrand
    - Laden von Modulen zur Funktionserweiterung (TCP/IP, FTP)
    - weitreichende Hilfe zur Konfiguration
- Debian Almquist shell (dash)
  - Portierung der Almquist shell auf Linux-Systeme
  - einige Erweiterungen zur ursprünglichen ash
  - seit 2006 Standard-Shell unter `/bin/sh` für Debian-Systeme

Vorsicht

- Verwendung von bash, zsh und ksh93*
  - alle drei haben viele (nützliche) Erweiterungen, die in den anderen Shells nicht existieren
  - alle drei sind außerhalb des POSIX-Standards nicht untereinander kompatibel
- Häufiger Fehler: `#!/bin/sh` als Shebang mit KornShell oder bash-Erweiterungen innerhalb des Skriptes
  - kann klappen, muss es aber nicht - je nachdem auf welche Shell `/bin/sh` zeigt!
  - Besser: wenn explizite Shell gewünscht, dann auch angeben, z.B. `#!/bin/bash`
- POSIX konforme Programmierung garantiert bestmögliche Portabilität, aber nicht unbedingt immer den besten Programmierkomfort

Unterschiede zwischen den Shells

- Einige Unterschiede zwischen häufig verwendeten Shells (unvollständige Auflistung)

	sh	csh	ksh	bash	tcsh	zsh
Job control	N	Y	Y	Y	Y	Y
Aliases	N	Y	Y	Y	Y	Y
Shell functions	Y	N	Y	Y	N	Y
Command history	N	Y	Y	Y	Y	Y
Command line editing	N	N	Y	Y	Y	Y
Login/Logout watching	N	N	N	N	Y	Y
Filename completion	N	Y	Y	Y	Y	Y
Username completion	N	Y	Y	Y	Y	Y
History completion	N	N	N	Y	Y	Y
Builtin artithmetic evaluation	N	Y	Y	Y	Y	Y
Custom Prompt (easily)	N	N	Y	Y	Y	Y
Spelling Correction	N	N	N	N	Y	Y
Freely Available	N	N	N	Y	Y	Y
Can cope with large argument lists	Y	N	Y	Y	Y	Y
List Variables	N	Y	Y	N	Y	Y
Local variables	N	N	Y	Y	N	Y

Verwendung von Shells für Skripte (bash vs dash)

- bash
  - Vorteile:
    - viele built-in Funktionen, die die Arbeit erleichtern können
    - eleganteres Programmieren
    - teilweise Vermeidung von Subshells durch built-in Funktionen
  - Nachteile:
    - groß und unhandlich (Größe 1.1 MB)
    - nur Grundfunktionalität ist POSIX-konform
- dash
  - Vorteile
    - klein und schnell (Größe 119 KB)
    - deutlicher Geschwindigkeitsvorteil beim Öffnen (vieler) Subshells
    - POSIX-konform & unterstützt nur POSIX-konforme Befehle, dadurch recht hohe Portabilität
  - Nachteile
    - Eingeschränkter Umfang
    - teilweise weniger elegante Lösungen

Weiterführende Links

POSIX-Standard für Shells & Utilities
https://pubs.opengroup.org/onlinepubs/9699919799/utilities/contents.html

Kurzer Überblick über POSIX
https://stackoverflow.com/questions/1780599/what-is-the-meaning-of-posix

POSIX-Kompatibilität von Shells (und deren Geschichte)
https://unix.stackexchange.com/questions/145522/what-does-it-mean-to-be-sh-compatible

Unterschiede zwischen den Shells
https://stackoverflow.com/questions/5725296/difference-between-sh-and-bash