Portabel Programmieren

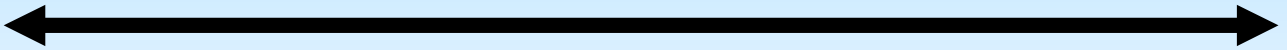
- Dr. Olaf Flebbe
- o.flebbe@science-computing.de

Einführung

- Erfahrung im Portieren und Reengeneering
- Systemnahes Programmierung im CAx Umfeld
  - Was heißt da portabel?
  - Wohin portabel?
  - Wie portabel?
  - Was portabel?
  - Warum portabel?

Was heißt da portabel?

- Programm so zu schreiben, dass die Chance besteht, es mit vertretbarem Aufwand auf einem anderen System benutzbar zu machen.
- Optimal: Das Programm auf System kopieren, tut.
- Murx: Das Programm wegschmeissen und neu schreiben.



Optimal Murx

Warum portabel?

- Warum nicht? Killer-Applikation
  - Catia: AIX in CAD Abteilungen
  - MS Word, MS Excel auf Windos
  - gimp für Linux/Unix?
- Erweitert die Einsatzmöglichkeiten
- Verlängert die Lebenszeit

Wohin portabel?

- Linux i386 (Debian, Ret Hat, Suse)
- Linux (m68k, alpha, sparc, IA64...)
- Unix (AIX, HP-UX, IRIX, Solaris, Unicos, Super-UX...)
- Echtzeit/Embedded systeme (Lynx, QNX,...)
- Windos
- Palmtops (EPOC, PalmOS)

Was ist portabel?

- Software, die auf vielen verschiedenen Systemen erhältlich ist:
  - UNIX
  - Linux
  - TeX
  - X11
  - perl
  - GNU emacs

Wie programmiert man portabel?

- Wie haben es die Autoren erreicht, die Software auf vielen verschiedenen Systemen zu unterstützen?
- Viele Strategien werden verwendet:
  - #ifdef Wüste && Automatisches Portieren
  - Reduktion
  - Abstraktion

#ifdef Wüste

- emacs
- perl
- Lässt sich manchmal nicht vermeiden

Automatisches Portieren

- Generieren von Kommandozeilen für bedingte Kompilierung
  - perl: MetaConf
  - GNU: autoconf
- Für UNIX ein Muss
- generieren von Makefiles
  - X11: imake
  - perl: MakeMaker
- Für X11 bzw. perl ein Muss

Zusammenfassung

- Das Feature testen, nicht das System!
- Autoconf und imake
  - Autoconf testet dynamisch
  - Imake verwendet statische Informationen
  - imake ausserhalb von X11 nicht durchgesetzt:
    Viele UNIX Hersteller liefern falsche config Dateien.

Reduktion

- Verwenden von Standards:
- tex als batch prozessor in PASCAL, keine dynamische Speicherverwaltung, keine Grafik, kaum Benutzerinteraktion

C: Standards I

- ANSI C
  - Der C Standard, räumt den Pseudo K&R Standard auf:
    - function prototypes
    - void
    - I/O funktionen: fopen, fread, printf, rename, remove
    - String fuktionen: strcpy, strchr...
    - signal

C: Standards II

- POSIX 1003.1b
  - Umfasst ANSI-C
  - UNIX Schnittstellen
    - I/O: open, creat, mkdir, link,
    - KEINE symbolischen Links (-> Windos NT)
    - Genauere Signale: sigaction etc
    - Processe: exec, fork
  - Weitere Posix Standards: pthreads, Echtzeit, IPC

C: Standards III

- System V (SVID)
  - Posix 1003.1b
  - IPC: Semaphore, Shared Memory
  - TLI (Das hat Linux nicht!)
- BSD 4.x
  - Networking: sockets
- X/Open, XPG4 und Unix95
  - termios
  - curses

C: fremde Welten

- Win32
- Palm
- EPOC

- ANSI gibt es praktisch immer, meist auch POSIX Unterstützung

C: Zusammenfassung

- Die unter Linux vorhandenen Funktionen sind ein bunter Topf von UNIX (z.T. Pseudo) Standards
- Die man pages sind gute Information, zu Standards
- Mit POSIX API fehlt Netzwerk, Grafik und alles was Spass macht.

Abstraktion

- Neue Programmiersprachen
- Virtuelle Maschinen
- Abstrakte Schnittstellen und ihre Implementation

Neue Programmiersprache

- UNIX: Erfindung von C, um von Assembler wegzukommen
- TeX: web: Literarisches Programmieren.
- Nicht mehr aktuell

Virtuelle Maschinen

- scheme, modula, oberon,...
- GNU emacs: Grosse Teile des Editors sind in lisp geschrieben: emacs ist eigentlich ein Lisp Interpreter
- TeX: LaTeX und plain TeX ist auf ein absolut minimales System aufgebaut
- java: Definition eines virtuellen Processors
- Weiterhin aktuell.

Abstraktion

- AWT
- SGI Open GL: Die Funktionalität ist in einer shared Library untergebracht, die zur Laufzeit geladen wird.

Skript Applikationen

- Verwendung einer Skriptsprache macht alleine noch keine portable Applikation
- csh ist generell zu vermeiden, da es keine Funktionen bietet
- sh ist sehr schwierig zu maintainen, läuft dafür aber auf jedem UNIX System (Autoconf zusammen mit m4)

Do It Yourself

- Wenn es keinen Standard gibt, dann beschränkt man sich nicht auf das Portable, sondern legt eben selbst Hand an:
  - java SWING / java Allgemein
  - qt Bibliothek
  - StarOffice (StarView)
- Kann Performance stark beeinträchtigen

Was kann denn da noch schief gehen?

- Compiliert ≠ funktioniert
  - Structure alignment
  - pointer aliasing
  - Unklare Standards

Ratschläge I

- Die richtige Sprache für das richtige Problem finden!
- Informieren zu vorhandenen Werkzeugen sammeln
- Portable Toolkits verwenden, soweit es geht.
- Verwendung von Standard Bibliotheken, soweit es geht.

Ratschläge II

- Systemabhängige Teile in einzelnen Modulen isolieren
- Abstraktion hierfür entwickeln
- Auf so vielen Plattformen wie möglich compilieren und testen!
- Immer alle Warnings ansehen. Optionen: (-Wall -Weffc++)