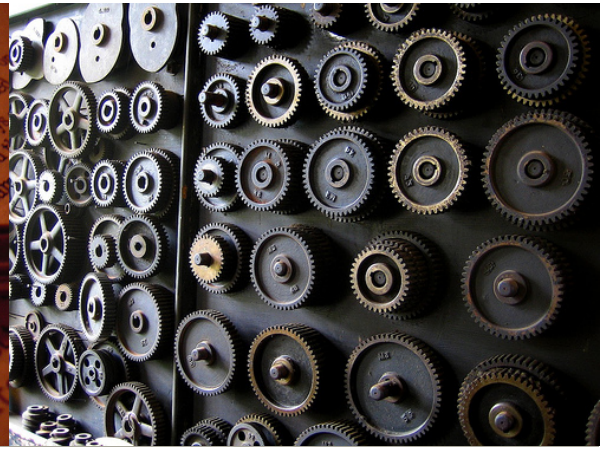
Freie Universität Berlin





# Systemverwaltung Sommersemester 2008
# am Fachbereich Mathematik und Informatik

## Automatisierung unter Windows

Daniel Bößwetter boesswet@inf.fu-berlin.de

August 2008

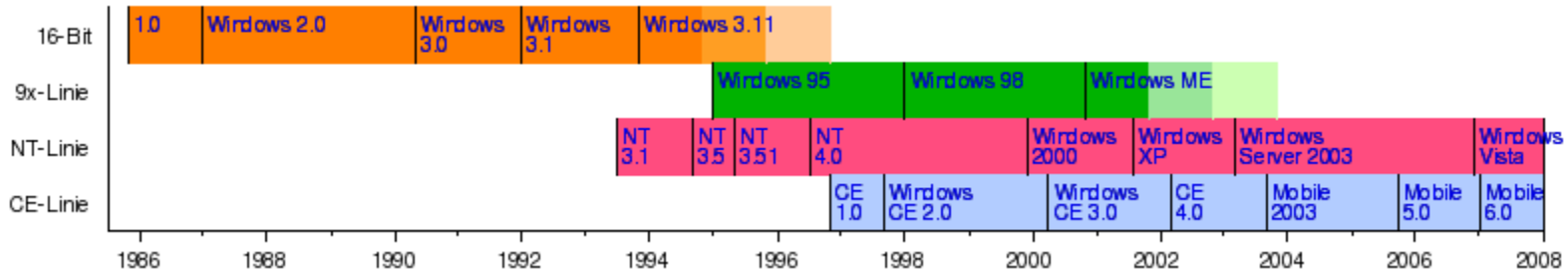
## Inhalt

Windows Historie (kurz)

Windows-Automatisierung mit Unix-Tools
-Cygwin
-Interix Services for Unix
-Perl, Python ...

Windows-Automatisierung mit Windows-Tools
-Batch-Files
-Windows-Scripting (VBS, JS) – nur kurz und knapp
-Powershell

# Windows Historie




-16bit Windows: Automatisierung durch Batch-Dateien

-Ab Windows 2000: Automatisierung durch Windows Scripting Host (WSH)

-Ab Windows Vista: Automasitierung durch Monad/Powershell

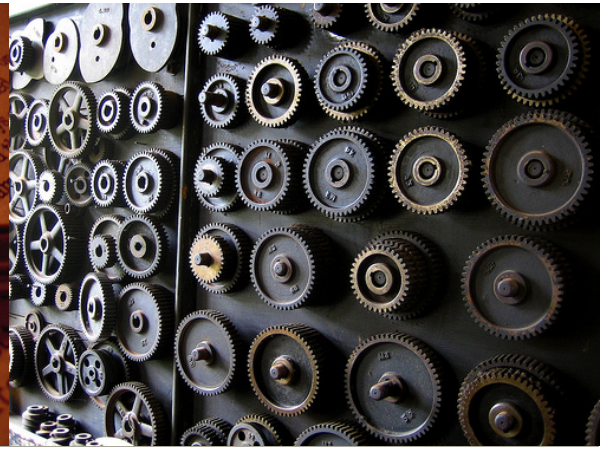
-(die jeweils älteren Mechanismen leben natürlich weiter!)

-**(Exkurs: Was bedeutet eigentlich 32bit?)**

(Bildquelle: Wikipedia)

Freie Universität Berlin





# Windows-Automatisierung mit Unix-Tools

Cygwin, Interix, Perl/Python ...

# Cygwin

-„Cygwin is a Linux-like environment for Windows.“ (cygwin.com)

-Ursprünglich „Minumum GNU for Windows“ (MinGW) entwickelt, um GCC unter Windows nutzen zu können.

- Ehem. Cygnus, heute Redhat

-Unix-API wird als Bibliothek (DLL) implementiert.

-(Fast) alles, was unter Linux läuft, kann durch Re-Kompilieren unter Cygwin ausgeführt werden.

-Viele Programmme verfügbar

- Shells
- Dateisystem-Tools
- X11
- Compiler und Scriptsprachen

-Frei verfügbar unter www.cygwin.com

# Interix Services for Unix (SFU 3.5)

-Ähnlich wie Cygwin, aber offiziell von Microsoft (ehemals von Interix)
- Ein „virtuelles“ Wurzelverzeichnis
- Unix-Rechtesemantik
- Pseudo-Prozessliste


-Unterschiede zu Cygwin
- Weniger GNU-Tools
- Mehr an kommerziellen Unix-Derivaten orientiert
- Näher am Systemkern:
  - NFS-Server und Client
  - Authentifikation

-Download unter
http://www.microsoft.com/germany/windowsserver2003/technologien/sfu/default.mspx

# Perl

-Scriptsprache aus dem Unix-Umfeld (basiert auf sed und awk)

-Gut geeignet für Textverarbeitung

-Im Kern Systemunabhängig

-11.000+ Module auf cpan.org für

- Wissenschaftliches Rechnen
- Datenverarbeitung (Text, XML, Grafik-Formate ...)
- Betriebssystem-Schnittstellen: POSIX, Win32 ...
- Und und und

-Gibt‘s für Cygwin oder nativ für Windows unter http://www.activestate.com


-Ähnliches gilt übrigens für Python und TCL (und vermutlich auch für Ruby, Groovy undsoweiter).

# Pros / Contras für Unix-Tools unter Windows

**Pro**

-Vereinheitlichung in heterogenen Netzen

-Knowhow-Transfer

**Contra**

-Es gibt trotzdem kleinere Differenzen

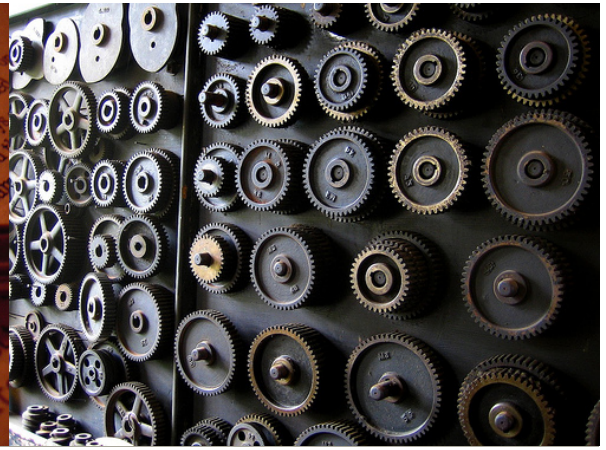
- Filenamen sind unter Windows case-insensitiv
- Abbildung von ACLs auf Berechtigungen ist nicht eindeutig ...

-Wenn die verantwortlichen Personen kein Unix-Knowhow haben

-Alle vorgestellten Lösungen müssen extra installiert werden und stehen daher nicht immer zur Verfügung

Freie Universität Berlin





# Windows-Grundlagen

Dateisystem, Prozesse ...

# Windows Dateisystem

## Unterschiede zu Unix

-Es gibt kein eindeutiges Wurzelverzeichnis, sondern gleichberechtigte Laufwerke A:, C:, D: ... (Ausnahme „Mein Arbeitsplatz“)

-Dateinamen sind „case-insensitive“, d.h. Groß-/Kleinschreibung wird nicht unterschieden

-Es gibt Attribute und ACLs statt der Berechtigungs-Bits

-Pfad-Trenner ist der Backslash „\“

## Datei-Attribute

-A (archive) – wird durch Schreiboperationen gesetzt, dient der inkrementellen Datensicherung

-R (read-only) – markiert Datei als nur lesbar (unabhängig von Berechtigungen)

-H (hidden) – Markiert Datei als unsichtbar

-S (system) – verhindert das verschieben von Dateien auf der Festplatte

# Windows Dateisystem – ACLs

**Access Control Lists**

-Rechte können pro Datei (oder Verzeichnis) für Benutzer und Gruppen einzeln vergeben oder entzogen werden.

-Rechte

- F (full) – Vollzugriff
- R (read) – Lesen
- W (write) – Schreiben
- C (change) – ändernd schreiben
- Weitere Rechte nur mit Zusatz-Tools bearbeitbar (Eigentum übernehmen, Rechte ändern ...)

-Vererbung von ACLs (dynamisch ab W2k)

# Windows Dateisystem – wichtige Kommandos

| | |
|---|---|
| Laufwerksbuchstabe, z.B. C: | Wechselt das aktuelle Laufwerk |
| cd, chdir [/d] | Wechselt das Arbeitsverzeichnis auf dem aktuellen (oder im Pfad angegebenen) Laufwerk |
| dir | Listet Verzeichnisinhalt auf |
| attrib | Zeigt oder ändert Attribute |
| cacls | Zeigt ACLs an oder ändert sie |
| copy, xcopy | Datei(en) kopieren |
| del | Datei löschen |
| ren | Datei umbenennen |
| move | Datei verschieben |
| md | Verzeichnis anlegen |
| rd | Verzeichnis löschen |
| | |

# Windows Prozeßsystem

Eigenschaften eines Prozesses (u.a.):

```
class Win32_Process : CIM_Process {
        string CommandLine;
        string Name;
        datetime CreationDate;
        string ExecutablePath;
        uint16 ExecutionState;
        uint64 KernelModeTime;
        uint64 UserModeTime;
        uint32 ProcessId;
        uint32 ParentProcessId;
        uint32 Priority;
        uint32 SessionId;
        string Status;
        datetime TerminationDate;
        uint32 ThreadCount;
        uint64 UserModeTime;
};
```

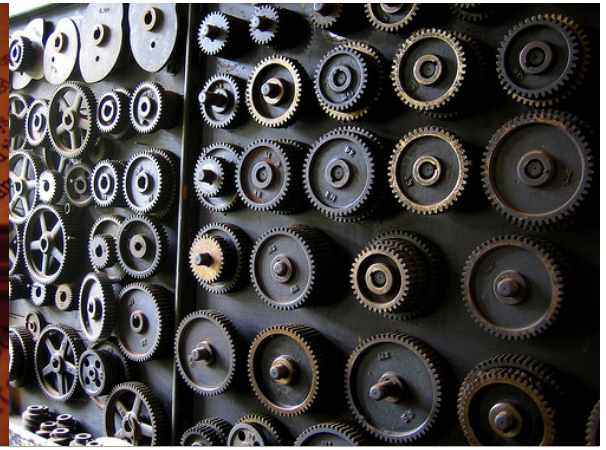
(Quelle: WMI-Dokumentation bei Microsoft)

# Windows Prozeßsystem - Kommandos

| | |
|---|---|
| tasklist | Prozessliste, ggfs. gefiltert |
| taskkill | Beendet einen Prozess anhand seiner ID |
| timethis | Misst Ausführungszeit eines Kommandos |
| at | Plant die Ausführung eines Hintergrund-Prozesses |

Freie Universität Berlin





# Automatisierung mit Windows „Bordmitteln“

Windows-Scripting, Batch-Files, Powershell ...

# Windows Scripting (more to come)

Windows Scripting Host

-Framework mit austauschbaren Scriptsprachen

- VBScript
- Jscript
- PerlScript

-... und einem gemeinsamen Objekt-Modell (COM-Komponenten)

- Office-Automatisierung
- Dateisystem
- Datenbank-Zugriffe
- Windows Management Instrumentation (WMI)

-Web-Applikationen (ASP – nicht meht ganz aktuell)

-Ein WMI-Beispiel in VBS (Liste aller at-Jobs):

```
set service = GetObject("winmgmts:")
set jobs = service.ExecQuery("SELECT * FROM Win32_ScheduledJob")
for each s in jobs
    wscript.echo( s.JobId & " """ & s.Command & """")
next
```

# Windows Batch Programmierung

-Batch („Stapel-“) Verarbeitung gibt es seit MS-DOS

-Ein Batchfile (.bat) ist eine Textdatei mit Kommandos, die nacheinander ausgeführt werden.

-Kommando-Interpreter

- früher COMMAND.COM (unter DOS und 16bit-Windows)
- heute CMD.EXE (seit Windows NT)

-Hilfe zu den wichtigsten Befehlen liefert „help“
bzw. „help <befehlsname>“

-Es gibt inzwischen sogar Kontrollstrukturen ...

- **if** für bedingte Ausführung
- **for** für Schleifen (und vieles mehr!)

-... und ein relativ mächtiges Variablen-Handling (siehe „help set“)

- Rechnen
- Textersetzung
- Substrings

# Windows Batch Programmierung vs. Unix

## Ähnlichkeiten zu Unix Shells

-IO-Umleitung mit <, >, >>, 2>

-Pipes mit |

-Konjunktion mit &&, Disjunktion mit ||

-Kommando-Vervollständigung mit TAB (seit XP aktiviert)

## Unterschiede zu Unix-Shells

-Wildcards (*,?) werden nicht von der Shell interpretiert, sondern nur von ausgewählten Kommandos.

-Kommandosubstitution mit `Backticks` funktioniert nur in for-Schleifen.

-Optionen werden meistens mit „/“ statt mit „-“ eingeleitet.

-Variablen-Ersetzung mit %var% statt $var.

-Maskierung mit „^“ statt mit „\“.

-Quoting?

# Batch - Wichtige Kommandos

| | |
|---|---|
| findstr | Sucht Strings (oder reguläre Ausdrücke) in Textdateien (oder stdin) |
| more | Pager, analog zum Unix-more |
| sort | Sortiert Textdateien (oder stdin) |
| tree | Listet ein Verzeichnis rekursiv als ASCII-Text-Baum auf |
| net send | Sendet kurze Textnachricht an einen eingeloggten User im Netz |
| net use | Verbindet Netz-Laufwerk mit Laufwerksbuchstaben (bei neueren Versionen auch WebDAV) |
| net user | Legt User an oder löscht diese |
| net ... | Weitere Netzwerk-Konfiguration (siehe „net help“) |
| | |

# Zusätzliche Tools für Batch-Files

**Microsoft Support Tools**

-Auf der Windows-CD unter

- \Support\Tools
- \ValueAdd

**Microsoft Resource-Kit**

-eigentlich als Buch erhältlich ...

-... aber seit W2K3 sind die Tools bei Microsoft downloadbar

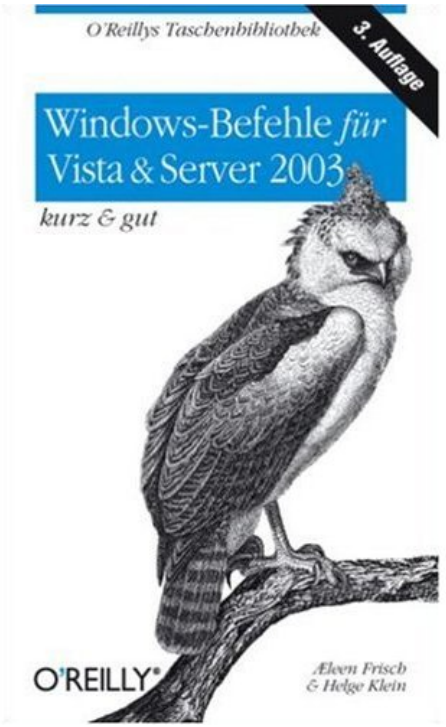
**Sysinternals**

-www.sysinternals.com (inzwischen von Microsoft gekauft)

# Literatur zu Batch-Programmierung

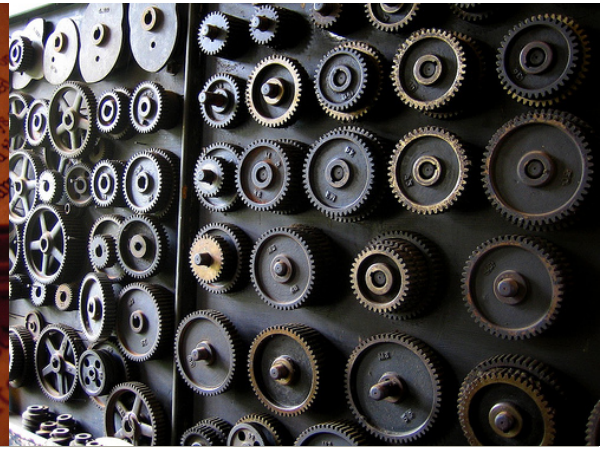
**Windows Befehle für Vista und Server 2003 - kurz und gut (Broschiert)**

von Aeleen Frisch (Autor), Helge Klein (Autor)



Freie Universität Berlin





# Windows Powershell

# Windows Powershell

-neue Shell für Systemadministration (ab Windows Vista)

- interaktiv
- nicht interaktiv

-basiert auf .NET

-orientiert sich an früheren Shells, ist aber Objekt-orientiert (statt Text-orientiert)

-weitere Infos unter
http://www.microsoft.com/germany/technet/scriptcenter/hubs/msh.mspx

-beginnend mit Ausgabe 08/2007 gibt es ein Powershell-Tutorial in der iX (Online verfügbar über OPAC der FUB!)

# Windows Powershell: Cmdlets

-Cmdlets: grundlegende Kommandos, per Konvention immer in der Form <verb>-<substantiv> benannt.

- z.B. Get-Command

-Online-Hilfe ähnlich wie Unix-Manpages

-Pipelines können wie gewohnt mit „|“ konstruiert werden

- Objekt-Pipelines (.NET Objekte)

-Es gibt Aliase für gängige Kommandos (z.B. dir, ls, man, echo, ps…)

-Variable haben immer (!) ein $ davor, z.B.

```
$x=1
echo $x
```

-Arrays

- Konstruktion $a = 1, 2, 3
- Zugriff $a[0]

-Kommandosubstitution durch ()

-Funktionen

| | |
|---|---|
| Get-Command | Listet alle Cmdlets (z.B. 129) |
| Get-Help<br>man | Hilfe zu einem Kommando |
| Get-Member | Zeigt die Meta-Daten der Objekte in der Pipeline |
| Object-Where | Selektiert Objekte mit bestimmten Eigenschaften |
| Select-Object | Projiziert Eigenschaften |

# Powershell: Flusskontrolle

-Bedingungen werden mit if in einer C-ähnlichen Syntax geschrieben:

```
if ( $x -eq 1 ) { … }
```

(Weitere Prädikate –lt, -gt, -le, -ge)

-switch-Statements haben eine etwas gewöhnungsbedürftige Syntax

```
switch ( $var ) {
        1 { … }
        2 { … }
        default { … }
}
```

-for-Schleifen:

```
for ( $i = 0 ; $i –lt 100 ; $i++ ) { }
```

(while- und until-Schleifen analog)

-Foreach-Schleifen werden mit einem Cmdlet realisiert:

```
… | Foreach-Object –process { … $_ … }
```

# Powershell: Zugriffe auf das System

-Dateisystem
- Festplattenpartitionen C:, …
- „virtuelle“ Laufwerke für
  - Registry (HKLM:, HKCU:)
  - Zertifikate (cert:)
  - Variable (variable:)
  - Umgebungsvariable (env:)
  - Funktionen (function:)
- Navigation …

-Prozesse durch Get-Process

-Dienste durch Get-Services

| | |
|---|---|
| Set-Location<br>cd | Wechselt in Verzeichnis |
| Get-Location<br>pwd | Gibt aktuelles Verzeichnis aus |
| Get-ChildItem<br>ls | Listet Verzeichnis-Inhalt |
| Get-Item | Liefert Objekt anhand des Pfades |
| Get-Process | Listet alle laufenden Prozesse |
| Get-Service | Liefert alle Dienste |

# Ausgabe: Formatierung und Umleitung

| | |
|---|---|
| Format-List | Ausgabe als Liste (Eigenschaften untereinander) |
| Format-Table | Ausgabe als Tabelle (Eigenschaften nebeneinander) |
| Out-Host | Ausgabe auf die Konsole (inkl. Paging) |
| Out-File | Ausgabe in Datei umleiten |
| Out-Null | Ausgabe gar nicht anzeigen |
| Write-Host | Ausgabe auf die Konsole |
| Read-Host | Eingabe von der Konsole |
| Write-Output<br>echo | Ausgabe auf die Konsole |

# Powershell: Zugriff auf Komponenten

-Komponenten
- .NET
  - z.B. ADO.NET
- COM
  - Word, Excel, …

-Windows Management Instrumentation
- Beispiel
  - Win32_Process
  - Win32_Processor
  - Win32_DiskDrive

| | |
|---|---|
| Get-Object | Zugriff auf .Net oder COM-Komponenten |
| Get-WMIObject | Zugriff auf WMI-Objekte |

# Übung

Suche mit einem Powershell-Kommando rekursiv in lokalen (Festplatten-)Dateisystemen nach Dateien mit dem Namen *virus.exe* und geben deren Pfad, Größe und Erstelldatum aus.

**Kommandos:**

Get-ChildItem

Foreach-Object

Get-PSProvider

# Übung II

Schreibe ein Powershell-Script, welches ähnlich dem Unix top-Kommando eine Liste laufender Prozesse ausgibt, die in konfigurierbaren Abständen neu aufgebaut wird (Default 5sec). Dabei sollen die Prozesse entweder nach CPU-Zeit (Default) oder nach Speicherverbrauch absteigend sortiert werden. Es werden höchstens so viele Prozesse ausgegeben wie die Konsole Zeilen hat.

Es sollen die Optionen *–t <Anzahl Sekunden>* und *–m* (sortiere nach Speicherverbrauch) verstanden werden.

**Kommandos:**

Get-Process

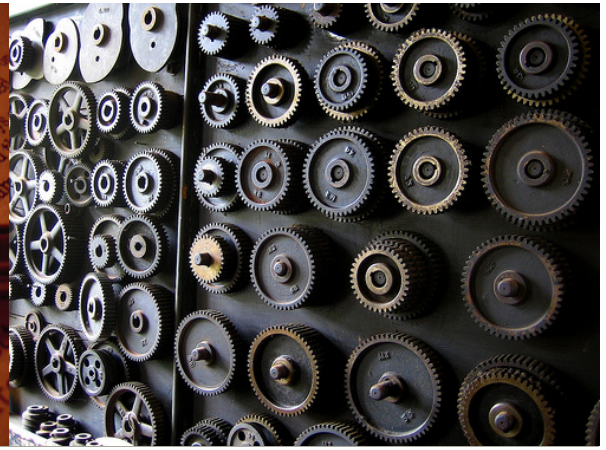
Sort-Object

Clear-Host

Start-Sleep

Freie Universität Berlin





# Ausblick

Wie kann man den Systembetrieb noch automatisieren?

# Ausblick

- „unattended“ Installation
  - Unix
  - Windows
- Monitoring
  - Verfügbarkeit von Diensten
    - Nagios (http://www.nagios.org), Netsaint ...
  - Performance-Überwachung
    - SNMP
    - MRTG (http://oss.oetiker.ch/mrtg/), Orca (http://www.orcaware.com/orca/) ...
  - Intrusion Detection / Intrusion Prevention
    - Tripwire (http://www.tripwire.com/products/enterprise/ost/)
    - Snort (http://www.snort.org/)
    - ...
- ...