REXX

Scriptsprache für viele Plattformen
Erste Schritte
F. Hodel
www.anetgmbh.ch
os2.a-net.ch

A.NET Advanced Networking GmbH

Was ist REXX

- REXX ist eine Scriptsprache
  - Interaktiv, bei erster Ausführung wird automatisch kompiliert
  - mit Zusatzfunktionen beliebig erweiterbar (SysLoadFunc)
  - Syntax etwas ähnlich wie bei Pascal und PL1
- Geschichte
  - entwickelt von Mike Cowlishaw unter VM
  - portiert auf viele Plattformen (OS/2, PC DOS 7, CICS, TSO, AS/400, Windows NT bis 3.51)
  - Regina für andere Plattformen (Windows, Linux etc)
- Verwandte Produkte
  - GUI-Programme
    - VISpro REXX von Hookware
    - Dr. Dialog
  - Java-Programme generieren
    - NetREXX
  - HTML-Code generieren
    - ObjREXX (auch von M. Cowlishaw)

Was kann REXX?

- Alle Funktionen einer modernen Programmiersprache
  - Benutzereingaben
  - Parsing (Analyse) von Eingaben und Variablen
  - Anzeige am Bildschirm
  - Schleifen
  - Variabeln
  - Mathematische Funktionen (mit beliebiger Genauigkeit!)
  - Dateien lesen und schreiben
  - Subroutinen
- Alle Funktionen des jeweiligen Betriebssystems
  - Befehl unverändert zwischen ' (Hochkommata) stellen
- Erweiterungen (unter OS/2 mit DLLs)
  - WPS Objekte erstellen/verändern/löschen (RexxUtil)
  - SYSTEM.INI und SYS.INI auslesen/schreiben (RexxUtil)
  - wissenschaftliche Funktionen/Mathematik (RxMath)
  - Multimedia Funktionen
- API in vielen OS/2 Anwendungen

A.NET Advanced Networking GmbH

Wie starte ich ein REXX-Programm?

- OS/2
  - cmd-File mit Kommentar auf erster Zeile (z.B. hallo.cmd)
- PC DOS 7
  - bat-File mit Kommentar auf erster Zeile z.B. hallo.bat)
- VM
  - exec-File mit Kommentar auf erster Zeile (z.B. profile.exec)
- AS/400, i5
  - rxpgm lib/hallo
- Hilfe?
  - bei OS/2 ab 2.x immer dabei via Help Center (geeignet zum Lernen!) oder via Befehlszeile:
  - **view rexx.inf**

Mein erstes REXX Programm:

- Datei hallo.cmd erstellen

```
/* Kommentar zum Programm hallo.cmd */
SAY "Willkommen in REXX!"
SAY "Wie heißen Sie?"
PULL wer                /* warte auf Benutzereingabe */
 IF wer = ""            /* Hat er nichts eingegeben? */
  THEN
  SAY "Hallo Fremder"
 ELSE
  SAY "Hallo" wer
exit
```

  - Beachte: Der eingegebene Name wird in Grossbuchstaben ausgegeben (sonst hätte man `parse pull` schreiben müssen, s. nächste Seite)
- Ausführen mit:

```
hallo [Enter]
```

Erläuterungen 1

- Kommentar
  - Erste Zeile MUSS ein Kommentar sein, minimal also: /* */
  - Beginnt mit /*
  - Endet mit */
  - kann über mehrere Zeilen gehen
  - darf hinten am Befehl stehen
- Text-Anzeige am Bildschirm
  - `say 'Dieser Text wird angezeigt'`
  - Text zwischen Hochkommata ' oder "
  - gemischt mit Variablen:

```
say 'Die Variable "wer" hat den Wert: ' wer
```

- Eingabe vom Benutzer
  - `pull wer` /* wer enthält alles, was der Benutzer eingibt, bis zu [Enter] */
  - `pull wer` /* alles wird in Grossbuchstaben umgewandelt */
  - `parse upper pull wer` /* macht auch alles zu Grossbuchstaben */
  - `parse pull wer` /* belässt Gross-/Kleinbuchstaben wie eingegeben */
  - `pull .` /* wartet, bis Benutzer [Enter] eingibt (ähnlich wie pause) */

Variable und Rechengenauigkeit

- Name muss mit einem Buchstaben beginnen
  - kein Punkt im Namen (sonst ist es ein Array)
- automatische Typ-Konversion (Text - Zahl)
  - keine Deklaration notwendig
  - wird initialisiert mit dem eigenen Namen in Grossbuchstaben
  - `say 'die Variable name enthält. ' name`
    (Anzeige ergibt bei undefinierter Variablen "name": NAME)
- Zahlen
  - Standardgenauigkeit 15 Stellen (unabhängig vom Betriebssystem/Hardware)
  - kann angepasst werden mit
    `numeric digits = 1000` (auf tausend Stellen genau)

```
/* Rechenbeispiel mit REXX */
  say 'Auf wieviele Stellen genau?'
  pull stellen
  numeric digits stellen
  result = 13/31097
  say '13/31097 ist genau: ' result
```

A.NET Advanced Networking GmbH

Zerlegen einer Variablen

- Der Befehl `parse var` dient dem Zerlegen komplexer Variablen
- Praktisch beim Prüfen von Eingaben
- Ein Beispiel: Analyse der IP-Adresse (Trennen beim .)

```
/* IP Adresse zerlegen in die vier Elemente */
ipaddresse = '192.168.112.30'
say 'ipaddr ist: ' ipaddresse
parse var ipaddresse  r1'.'r2'.'r3'.'r4
say 'Teil 1 ist: ' r1
say 'Teil 2 ist: ' r2
say 'Teil 3 ist: ' r3
say 'Teil 4 ist: ' r4
```

- Zerlegen in einzelne Worte (Trennen beim Leerzeichen)

```
/* Jedes Wort einzeln */
say 'Bitte geben Sie drei Wörter ein: '
parse pull antwort
parse var antwort wort1 wort2 wort3
say 'erstes Wort: ' wort1
say 'zweites Wort: ' wort2
say 'drittes Wort: ' wort3
```

Schleifen

- Eine bestimmte Anzahl mal ausführen

```
/* genau fünfmal */
anzahl = 5
do anzahl
   say 'Hallo REXX'
end /* do */
```

- bis eine Bedingung erfüllt ist

```
/* genau fünfmal mit Zähler */
x = 1
do z = x to 5
   say z
end /* do */
```

- unedlich (fast) mit do forever

```
/* fast ewig */
z = 0
do forever
   z = z + 1
   say z
   if z > 5 then leave /* Abbruchkriterium, sonst eben "for ever" */
end /* do */
```

A.NET Advanced Networking GmbH

Eingebaute Funktionen

- Es gibt viele eingebaute Funktionen
  - `xxxxxxx()` Wert in Klammer = Parameter
- weitere können mit SysLoadFunc hinzugefügt werden
- Beispiel: Datum und Zeit

```
/* Datum und Zeit angeben */
say date()
now1 = time('N') /* Normal, gemäss Country Setting  */
now2 = time('L') /* lang, mit 1000-tel Sekunden     */
now3 = time('C') /* Englische Art mit am / pm       */
say now1
say now2
say now3
```

- Beispiel: Leerzeichen entfernen

```
text = strip(text)  /* alle Leerzeichen der Variablen text werden entfernt */
```

WPS Objekte erstellen

- SysCreateObject um Folder und Programm-Objekte zu erstellen
- Zusatzfunktion notwendig (RexxUtil)

```
/* Erstellen eines Folders und Program Objects */
  call RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
  call SysLoadFuncs

/* Folder erstellen */
  If SysCreateObject("WPFolder","Test Folder FHO","<WP_DESKTOP>","OBJECTID=<TESTFHO>")
    Then Do
       Say 'Folder erstellt'

      /* Programm 1 erstellen */
      If SysCreateObject("WPProgram","Test Programm1","<TESTFHO>",,
         "OBJECTID=<PROGRAM1>;"||,
         "EXENAME=D:\utl\browse2.exe;"||,
         "PROGTYPE=PM;")
       Then Say '"Test Programm 1" erstellt'
       Else Say '"Test Programm 1" nicht erstellt'
      /* Ende Programm 1 */
    End
  Else Say 'Folder und Programm nicht erstellt'
```

- Beachte: SysCreateObject wird mit If abgefragt.
  - Wert ist "true", falls Befehl erfolgreich war
  - Wert ist "false", wenn der Befehl nicht erfolgreich war
- Jedes WPS-Objekt muss einen eindeutigen Namen haben (hier TESTFHO für den Ordner und PROGRAM1 für das Programm)