

Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

Programování v z/OS - Rexx

- jde o skriptovací jazyk, který se snaží být blízký běžnému jazyku
- má poměrně snadno naučitelnou syntaxi
- není háklivý na mezery jako JCL nebo HLASM

- komentáře se označují jako v C tj. pomocí
- první řádek vždy musí obsahovat komentář s textem REXX tj.

```
/****** REXX *****/
```

- příkaz SAY slouží k vypsání textu

```
SAY "Some text"
```

```
SAY "1234"
```

```
SAY 'abcde'
```

```
SAY abcde
```

- řetězec se ohraničuje uvozovkami nebo apostrofem
- při použití uvozovek lze uvnitř vypsát apostrof a naopak

- přidáním X nebo B za řetězec je ten chápan jako hexadecimální nebo binární číslo

```
SAY "1234"X
```

```
SAY ' abcde' X
```

```
SAY "010101010"B
```

- čísla se definují běžným způsobem

```
num_1 = 29
```

```
num_2 = -94.3
```

```
num_3 = 53.42E2
```

```
num_4 = -0.7E-3
```

- proměnné není potřeba inicializovat o udávat jejich typ
- vše je chápáno jako řetězec a konvertováno podle potřeby
- není-li proměnné přiřazena hodnota, nastaví se sama na své jméno zapsané velkými písmeny

SAY variable

- jména proměnných mohou obsahovat tečky, což je umožňuje slučovat do něčeho jako objektů – stemů

```
name. = "nobody"  
name.a = "joe"  
name.b = "jim"  
d=1  
name.d = "bill"  
say name.a name.b name.c name.d name.1
```

Rexx - navazování řetězců

```
a = "one"; b = "two"  
total = 75  
SAY a b  
SAY ab  
SAY a||b  
SAY total "%"  
SAY total"%"
```

Rexx - příkaz PARSE

- PARSE PULL - načítá řetězec z klávesnice
- PARSE VAR - slouží k parsování řetězců

Rexx - příkaz PARSE

```
PARSE PULL program  
SAY "Program name is: " program
```

Rexx - příkaz PARSE

ARG obsahuje argumenty procedury/příkazové řádky

```
ARG arguments
PARSE VAR arguments arg1 arg2 arg3 arg_leftf
SAY "Arg1 = " arg1
SAY "Arg2 = " arg2
SAY "Arg3 = " arg3
SAY "Left = " arg_left
```

Rexx - příkaz PARSE

```
SAY "Please tell me your name:"  
PARSE PULL first_name second_name  
SAY "Your name is " second_name first_name
```

Rexx - příkaz PARSE

```
new_date = "12/10/2013"  
PARSE VALUE new_date WITH mm "/" dd "/" yyyy  
SAY "Day is " dd  
SAY "Month is " mm  
SAY "Year is " yyyy
```

Rexx - příkaz PARSE

- části parsovaného textu, které nás nezajímají lze zahodit pomocí tečky

```
names = "Mike, Fred, Joe"  
PARSE VAR . ", " . ", " last_name  
SAY last_name
```


Rexx - příkaz PARSE

- text lze parsovat pomocí zadávání pozic v parsovaném textu

```
names = "Some text to be split up!"  
PARSE VAR names one 10 two 20 three  
SAY one  
SAY two  
SAY three
```

Rexx - příkaz PARSE

- pro relativní pohyb v parsovaném textu lze použít znak
+

```
number = "0987654321"  
PARSE VAR = number 3 one +2 two +4 three 2 four  
SAY one  
SAY two  
SAY three  
SAY four
```

Rexx - příkaz PARSE

```
data = "L/look for /1 10"  
PARSE VAR data verb 2 delim +1 string (delim)  
      rest  
  
SAY verb  
SAY delim  
SAY string  
SAY rest
```

Rexx - příkaz IF/THEN/ELSE

```
system = "UP"  
IF system = "UP" THEN  
    SAY "System is UP"  
ELSE DO  
    SAY "System is DOWN"  
    SAY "Turn the system on."  
END
```

Rexx - příkaz IF/THEN/ELSE

Lze použít tyto logické operace:

- = - rovná se
- \= - nerovná se
- > - větší než
- < - menší než
- >< - nerovná se
- >= - větší nebo rovno
- <= - menší nebo rovno
- \< - ne menší
- \> - ne větší

Rexx - příkaz IF/THEN/ELSE

Existují i striktní porovnání, kdy musí být např. čísla i stejně zapsána

- == - rovná se
- \== - nerovná se
- » - větší než
- « - menší než
- »= - větší nebo rovno
- «= - menší nebo rovno

Rexx - příkaz IF/THEN/ELSE

Platí:

```
"      REXX      " = "REXX"  
"00000012" = "12"
```

Ale neplatí:

```
"      REXX      " == "REXX"  
"00000012" == "12"
```

```
DO FOREVER
  SAY "Your name please:"
  PARSE UPPER PULL aux_name
  PARSE VAR aux_name name
  IF name = "BOB" THEN DO
    EXIT
  END
END
END
```



```
DO loop_counter = 1 TO 5 BY 1
  SAY loop_counter
END
```

```
loop_counter = 0
DO UNTIL loop_counter < 5
  loop_counter = loop_counter + 1
  SAY loop_counter
END
```

```
DO loop_counter = 1 TO 5
  IF loop_counter = 3 THEN DO
    ITERATE
  END
  SAY loop_counter
END
```

```
DO loop_counter = 1 TO 5
  IF loop_counter = 3 THEN DO
    LEAVE
  END
  SAY loop_counter
END
```