Legac-E Education

Data Structures in REXX

Whilst any data item could be subdivided into elements or fields the concept is more common when processing records from a file. Records will comprise a number of fields. A personnel record for instance may contain such items as Forename, Surname, Date-of-birth, Postal address, Gender, Title, joining-date, Department, telephone number etc.

The problem in REXX is that it whilst it can process records to or from either the Data Stack (External Data Queue) or STEM variables, its only view of a given record is as a complete entity, there is no concept of data structures which may exist in other languages.

does not have any intrinsic file processing capability but relies on the functionality of the environment it runs in. For TSO this means the use of EXECIO to read or write records.

To cater for the lack of structures REXX provides two potential solutions for processing individual fields within a record, and these solutions are:

- Parsing
- Substring processing (SUBSTR Built-in Function)

Let us assume that the COBOL definition of the record looks like this.

COBOL Group Structure

```
01  LOCO-REC.
    03  LOCO-NO-S              PIC 9(5).
    03  FILLER                 PIC X.
    03  LOCO-DES-S             PIC X(10).
    03  LOCO-BLDA              REDEFINES LOCO-DES-S
        05  LOCO-FIRM          PIC X(10).
    03  FILLER                 PIC X.
    03  LOCO-POWER-S           PIC XXX.
    03  FILLER                 PIC X.
    03  LOCO-NAME-S            PIC X(29).
```

Data Structures in REXX

Assuming that EXECIO is used to read the records into a STEM called rr. and that 10 records were read.

To extract the LOCO-DES-S field from the fifth record then either the PARSE instruction or the SUBSTR built-in function can be used.

Parsing instruction

```
Parse var rr.5 . 7 loco_des_s 17 .
Say ‘Locomotive designer is: ’ loco_des_s
```

Here the parse instruction is using two place-holders so that the information not required can be thrown away, and the result should be that everything between positions 7 and 16 within the record is copied to the variable `loco_des_s`.

If the record had been read into the Data Stack rather than a STEM Variable the following code achieves the same result:

```
Parse Pull . 7 loco_des_s 17 .
Say ‘Locomotive designer is: ’ loco_des_s
```

SUBSTR Built-in Function

```
Loco_des_s = SUBSTR(rr.5,7,10)
Say ‘Locomotive designer is: ’ loco_des_s
```

Here a sub-string starting at position 7 for a length of 10 bytes is extracted from the string held in rr.5.