11) Data Stacks

Instructions
- QUEUE,
- PUSH.

Built-in function QUEUED()

TSO commands and functions for work with stacks:
- DELSTACK,
- DROPBUF,
- MAKEBUF,
- NEWSTACK
- QBUF
- QELEM
- QSTACK

Resources: TSO REXX Reference
Chapter 10. TSO/E REXX Commands

PROPRIETARY AND CONFIDENTIAL INFORMATION

These education materials and related computer software program (hereinafter referred to as the "Education Materials") is for the end user's informational purposes only and is subject to change or withdrawal by CA, Inc. at any time.

These Education Materials may not be copied, transferred, reproduced, disclosed or distributed, in whole or in part, without the prior written consent of CA. These Education Materials are proprietary information and a trade secret of CA. Title to these Education Materials remains with CA, and these Education Materials are protected by the copyright laws of the United States and international treaties. All authorized reproductions must be marked with this legend.

RESTRICTED RIGHTS LEGEND

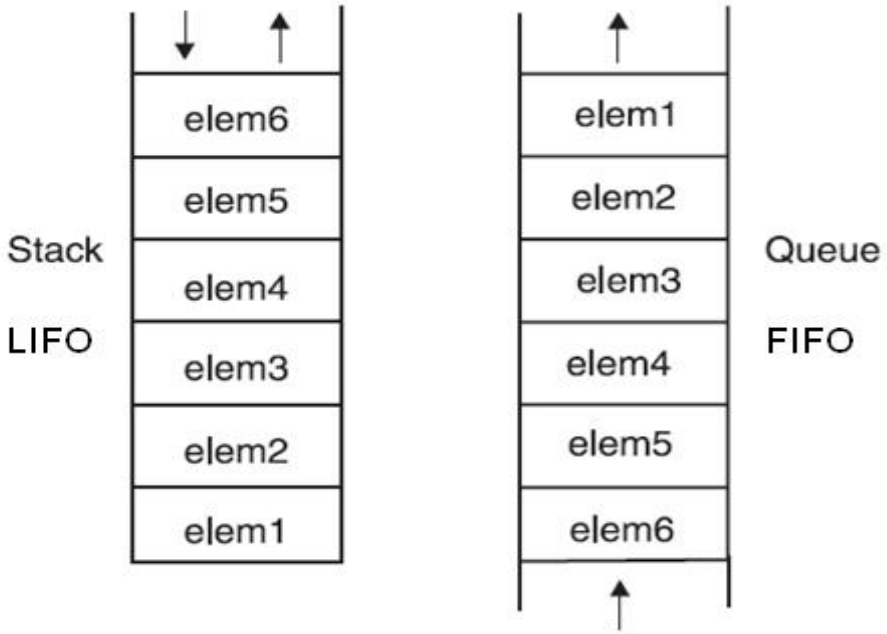
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

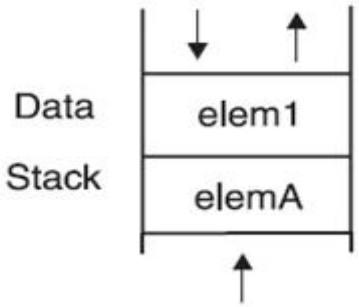
The manufacturer of this documentation is CA, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

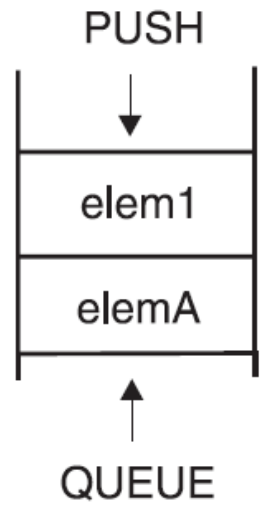
Data Stack



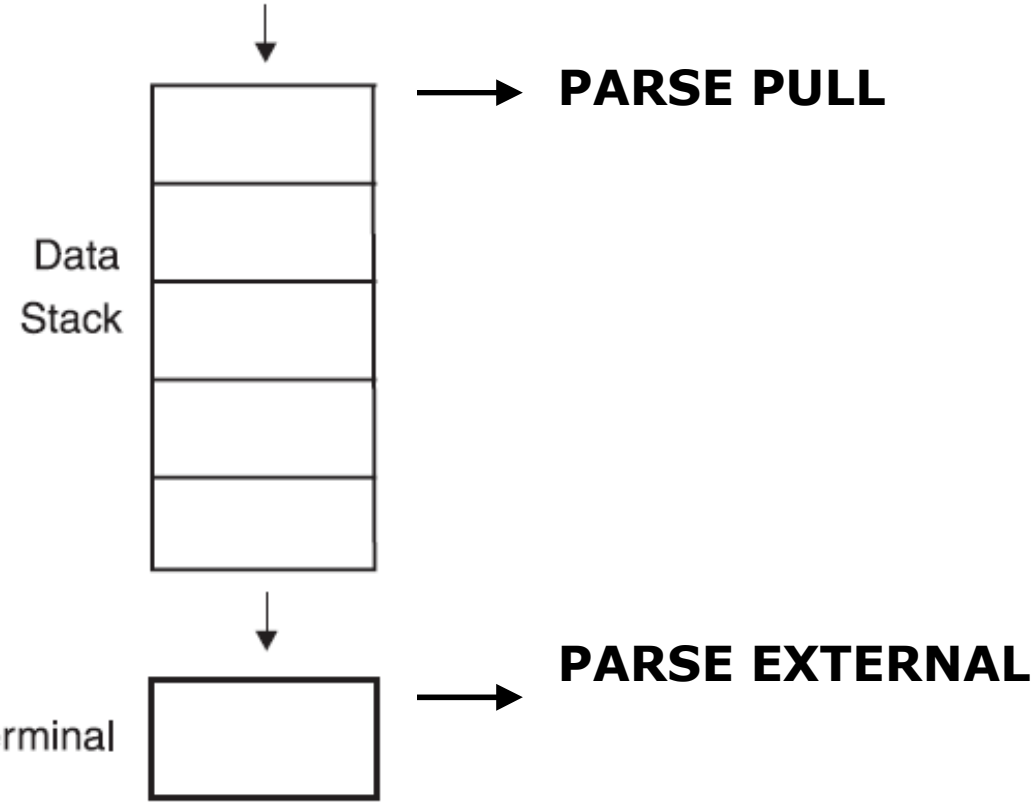
As shown in the following figure, the data stack that REXX uses combines the techniques used in adding elements to stacks and queues. Elements can be placed on the top or the bottom of a data stack. Removal of elements from the data stack, however, occurs from the top of the stack only.

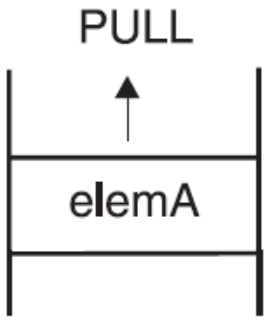


Data Stack - Manipulating with data

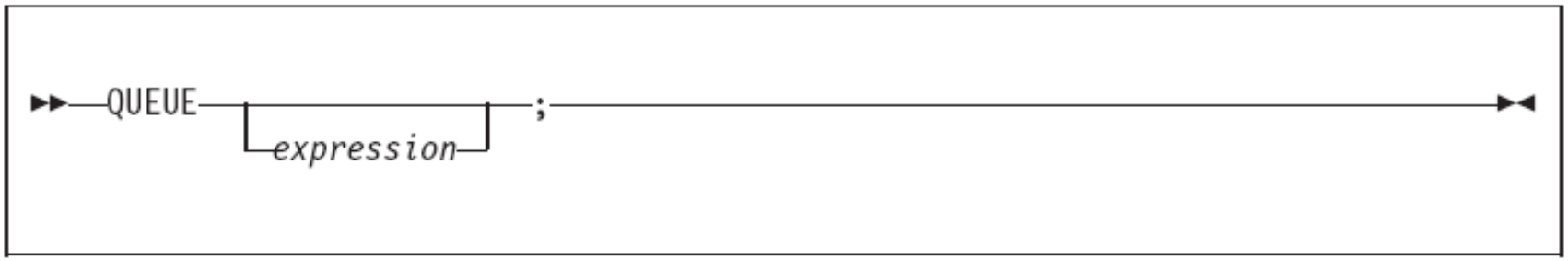


When an exec issues a PULL instruction, and when it issues an interactive TSO/E command, the data stack is searched first for information and if that is empty, information is retrieved from the terminal.



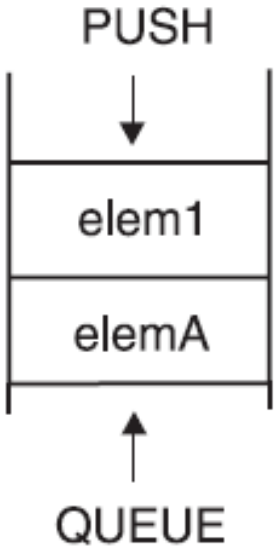


QUEUE



QUEUE appends the string resulting from *expression* to the tail of the external data queue. That is, it is added FIFO (First In, First Out).

If you do not specify *expression*, a null string is queued.



This course has been prepared by Milos Forman for MCoE needs only!

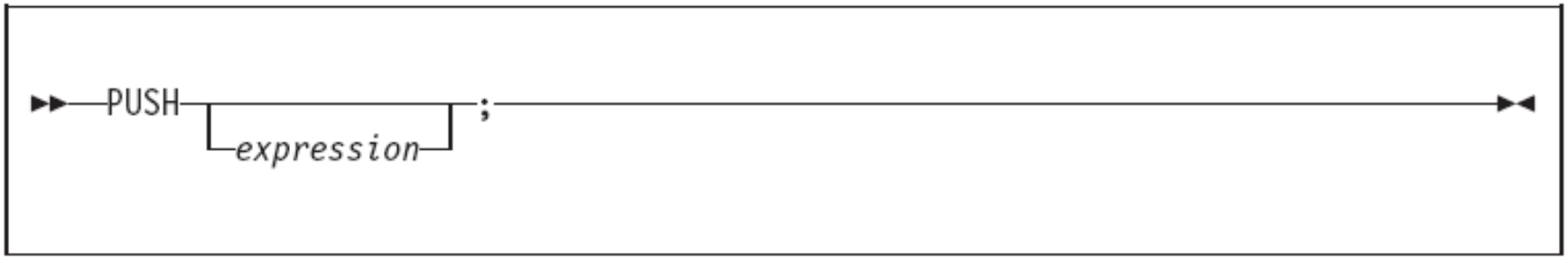
QUEUE example

```
ADDRESS "TSO"
"CLEAR"
SAY "Please enter your name :"
PARSE EXTERNAL full_name un_used
SAY "Please enter another name :"
PARSE PULL second_name un_used
QUEUE full_name
QUEUE second_name
PARSE PULL name
SAY "The first line off the stack was : "
SAY name
PARSE PULL name
SAY "The second line off the stack was : "
SAY name
```

```
 Please enter your name :
bob
 Please enter another name :
jane
 The first line off the stack was :
 bob
 The second line off the stack was :
 jane
 ***
```

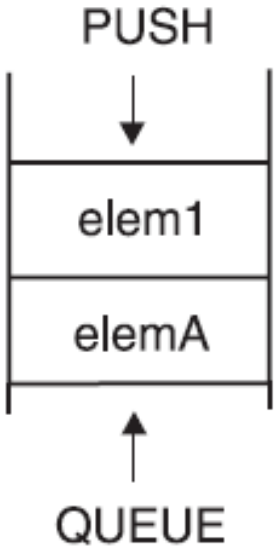
This course has been prepared by Milos Forman for MCoE needs only!

PUSH



PUSH stacks the string resulting from the evaluation of *expression* LIFO (Last In, First Out) onto the external data queue.

If you do not specify *expression*, a null string is stacked.



This course has been prepared by Milos Forman for MCoE needs only!

PUSH example

```
ADDRESS "TSO"
"CLEAR"
SAY "Please enter your name :"
PARSE EXTERNAL full_name un_used
SAY "Please enter another name :"
PARSE PULL second_name un_used
QUEUE full_name
QUEUE second_name
PARSE PULL name
SAY "The first line off the stack was : "
SAY name
PARSE PULL name
SAY "The second line off the stack was : "
SAY name
```

```
 Please enter your name :
bob
 Please enter another name :
jane
 The first line off the stack was :
 bob
 The second line off the stack was :
 jane
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!

QUEUED()

returns the number of lines remaining in the external data queue when the function is called.

The TSO/E implementation of the external data queue is the data stack.

This course has been prepared by Milos Forman for MCoE needs only!

QUEUED() Example

```
DO FOREVER
   SAY "Please enter your name :"
   PARSE UPPER EXTERNAL full_name un_used
   IF full_name = "" THEN DO
      LEAVE
   END
   ELSE DO
      QUEUE full_name
   END
END
no_in_stack = QUEUED()
DO no_in_stack
   PARSE PULL name
   SAY name
END
```

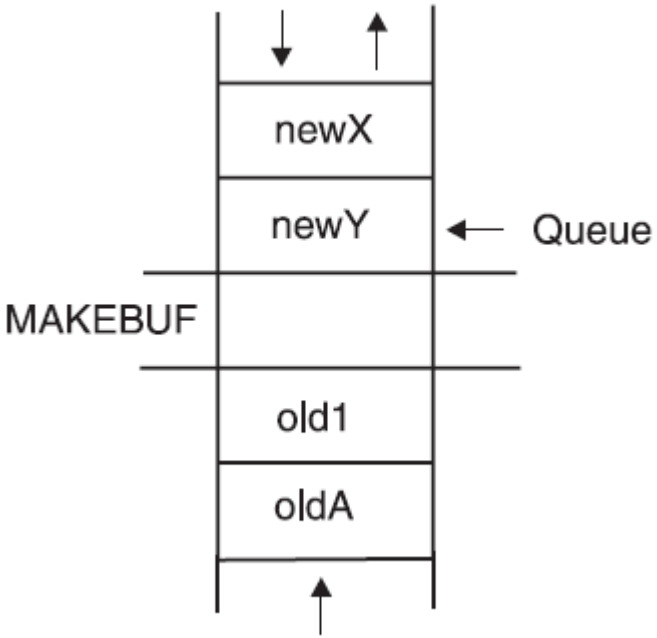
This course has been prepared by Milos Forman for MCoE needs only!

QUEUED() Example (cont.)

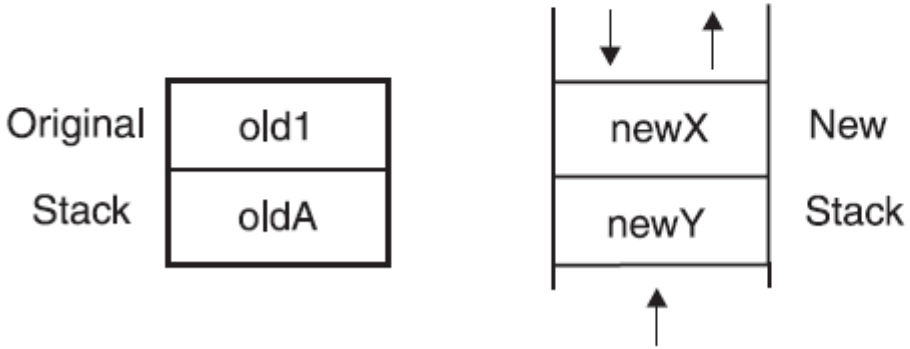
```
 Please enter your name :
bob
 Please enter your name :
jane
 Please enter your name :
jim
 Please enter your name :

 BOB
 JANE
 JIM
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!

Multiple Buffers and Stacks



All elements added to the data stack after the NEWSTACK command are placed in the new data stack. The original stack contains the elements placed on the stack before the NEWSTACK command.



MAKEBUF

Use the MAKEBUF command to create a new buffer on the data stack. The MAKEBUF command can be issued from REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

Initially, the data stack contains one buffer, which is known as buffer 0. You create additional buffers using the MAKEBUF command. MAKEBUF returns the number of the buffer it creates in the REXX special variable RC. For example, the first time an

Note: bThe TSO/E implementation of the external data queue is the data stack. The length of an element in the data stack can be up to one byte less than 16 megabytes. The data stack contains one buffer initially, but you can create additional buffers using the TSO/E REXX command MAKEBUF.

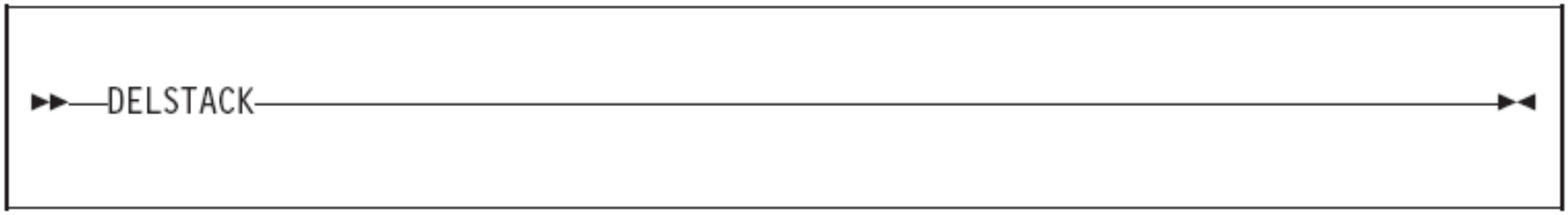
This course has been prepared by Milos Forman for MCoE needs only!

NEWSTACK

creates a new data stack and basically hides or isolates the current data stack. Elements on the previous data stack cannot be accessed until a DELSTACK command is issued to delete the new data stack and any elements remaining in it.

The NEWSTACK command can be used in REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

DELSTACK

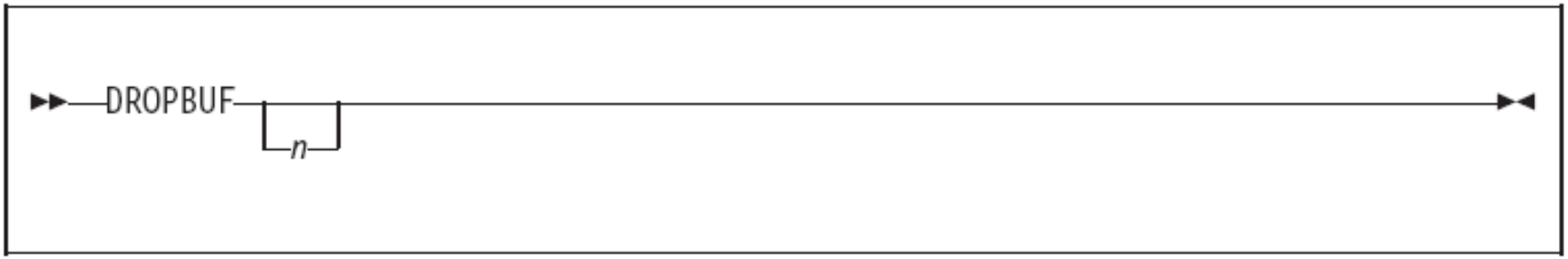


deletes the most recently created data stack that was created by the NEWSTACK command, and all elements on it. If a new data stack was not created, DELSTACK removes all the elements from the original data stack.

The DELSTACK command can be used in REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

This course has been prepared by Milos Forman for MCoE needs only!

DROPBUF



removes the most recently created data stack buffer that was created with the MAKEBUF command, and all elements on the data stack in the buffer. To remove a specific data stack buffer and all buffers created after it, issue the DROPBUF command with the number (n) of the buffer.

The DROPBUF command can be issued from REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

This course has been prepared by Milos Forman for MCoE needs only!

DROPBUF (cont.)

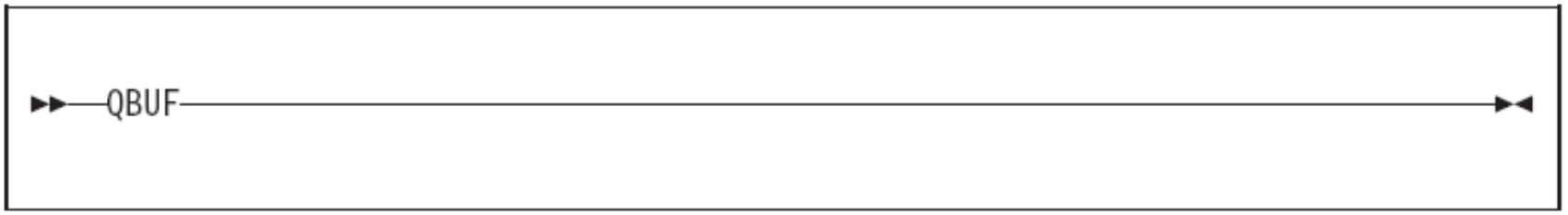
Operand: The operand for the DROPBUF command is:

n specifies the number of the first data stack buffer you want to drop. DROPBUF removes the specified buffer and all buffers created after it. Any elements that were placed on the data stack after the specified buffer was created are also removed. If *n* is not specified, only the most recently created buffer and its elements are removed.

The data stack initially contains one buffer, which is known as buffer 0. This buffer will never be removed, as it is not created by MAKEBUF. If you issue DROPBUF 0, all buffers that were created on the data stack with the MAKEBUF command and all elements that were put on the data stack are removed. DROPBUF 0 effectively clears the data stack including the elements on buffer 0.

This course has been prepared by Milos Forman for MCoE needs only!

QBUF



queries the number of buffers that were created on the data stack with the MAKEBUF command. The QBUF command returns the number of buffers in the REXX special variable RC. If you have not issued MAKEBUF to create any buffers on the data stack, QBUF sets the special variable RC to 0. In this case, 0 is the number of the buffer that is contained in every data stack.

You can use the QBUF command in REXX execs that run in both the TSO/E address space and non-TSO/E address spaces.

This course has been prepared by Milos Forman for MCoE needs only!

QBUF (cont.)

The following table shows how QBUF sets the REXX special variable RC.

Return Code	Meaning
0	Only buffer 0 exists on the data stack
1	One additional buffer exists on the data stack
2	Two additional buffers exist on the data stack
n	*n* additional buffers exist on the data stack

This course has been prepared by Milos Forman for MCoE needs only!

QELEM

queries the number of data stack elements that are in the most recently created data stack buffer (that is, in the buffer that was created by the MAKEBUF command). The number of elements is returned in the REXX special variable RC. When MAKEBUF has not been issued to create a buffer, QELEM returns the number 0 in the special variable RC, regardless of the number of elements on the data stack. Thus when QBUF returns 0, QELEM also returns 0.

The QELEM command can be issued from REXX execs that execute in both the TSO/E address space and in non-TSO/E address spaces.

This course has been prepared by Milos Forman for MCoE needs only!

QELEM

After the QELEM command processes, the REXX special variable RC contains one of the following return codes:

Return Code	Meaning
0	Either the MAKEBUF command has not been issued or the buffer that was most recently created by MAKEBUF contains no elements.
1	MAKEBUF has been issued and there is one element in the current buffer.
2	MAKEBUF has been issued and there are two elements in the current buffer.
3	MAKEBUF has been issued and there are three elements in the current buffer.
n	MAKEBUF has been issued and there are *n* elements in the current buffer.

This course has been prepared by Milos Forman for MCoE needs only!

QSTACK

queries the number of data stacks in existence for an exec that is running. QSTACK returns the number of data stacks in the REXX special variable RC. The value QSTACK returns indicates the total number of data stacks, including the original data stack. If you have not issued a NEWSTACK command to create a new data stack, QSTACK returns 1 in the special variable RC for the original data stack.

You can use the QSTACK command in REXX execs that run in both the TSO/E address space and in non-TSO/E address spaces.

This course has been prepared by Milos Forman for MCoE needs only!

QSTACK (cont.)

The following table shows how QSTACK sets the REXX special variable RC.

Return Code	Meaning
0	No data stack exists. See “Data Stack Routine” on page 454.

Return Code	Meaning
1	Only the original data stack exists
2	One new data stack and the original data stack exist
3	Two new data stacks and the original data stack exist
n	*n - 1* new data stacks and the original data stack exist

This course has been prepared by Milos Forman for MCoE needs only!

Multiple Stacks example

```
SAY "Please enter your first name : "
PARSE UPPER EXTERNAL fore_name un_used
QUEUE fore_name
"NEWSTACK"
SAY "Please enter your surname : "
PARSE UPPER EXTERNAL sur_name un_used
QUEUE sur_name
SAY "Please enter your surname : "
PARSE UPPER EXTERNAL sur_name un_used
QUEUE sur_name
"QSTACK"
SAY "You have "||RC||" of stacks."
/*---- Empty stacks ----*/
PARSE PULL stuff
SAY stuff
PARSE PULL stuff
SAY stuff
"DELSTACK"
PARSE PULL stuff
SAY stuff
```

```
 Please enter your first name :

bob

 Please enter your surname :

smith
 Please enter your surname :

jones

 You have 2 of stacks.


 SMITH

 JONES


 BOB
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!

Unused Stack Data

```
tso_cmd = "LISTC LEVEL(IULC20)"
PUSH tso_cmd
EXIT
```

This course has been prepared by Milos Forman for MCoE needs only!

EXECIO and Stacks

```
dsn_name = "crone90.crone.rexx(test)"
dsn_check = SYSDSN("'"dsn_name"'")
IF dsn_check = "OK" THEN DO
   "ALLOC DD(ddname) DSN('"dsn_name"') SHR REU"
   "EXECIO 1 DISKR ddname (FINIS)"
   PARSE PULL line
   SAY "The first line of the dataset is : "
   SAY line
END
ELSE DO
   SAY dsn_check
END
```

Work Section 11.1

- Write a REXX program to accept any number of names to the screen and when they type "STOP", display all the names.
- Store the names in a stack.

```
 Please enter your name :
jane
 Please enter your name :
sue
 Please enter your name :
stop
 JANE
 SUE
 ***
```

Work Section 11.2

- Write a REXX program to accept a list of names and store in a stack, then create a new stack and accept a number of Date of births and store them in the new stack.
- Display the contents of each stack showing the names in reverse order.

```
 Please enter your name :
bob
 Please enter your name :
fred
 Please enter your name :
stop
 Please enter your DOB :
051276
 Please enter your DOB :
040303
 Please enter your DOB :
stop
 051276
 040303
 FRED
 BOB
 ***
```

Additional Program

- Using EXECIO read one of your members into a data stack and then display the contents one line at a time with the option to stop displaying the data.

```
 Do you wish to see a line from the stack (Yes/end) ?
y
 /* REXX */
 Do you wish to see a line from the stack (Yes/end) ?
y
 SAY 'Enter your name'
 Do you wish to see a line from the stack (Yes/end) ?
end
 ***
```

11) Data Stacks

Instructions
- QUEUE,
- PUSH.

Built-in function QUEUED()

TSO commands and functions for work with stacks:
- DELSTACK,
- DROPBUF,
- MAKEBUF,
- NEWSTACK
- QBUF
- QELEM
- QSTACK

Resources: TSO REXX Reference
Chapter 10. TSO/E REXX Commands

PROPRIETARY AND CONFIDENTIAL INFORMATION

These education materials and related computer software program (hereinafter referred to as the "Education Materials") is for the end user's informational purposes only and is subject to change or withdrawal by CA, Inc. at any time.

These Education Materials may not be copied, transferred, reproduced, disclosed or distributed, in whole or in part, without the prior written consent of CA. These Education Materials are proprietary information and a trade secret of CA. Title to these Education Materials remains with CA, and these Education Materials are protected by the copyright laws of the United States and international treaties. All authorized reproductions must be marked with this legend.

RESTRICTED RIGHTS LEGEND

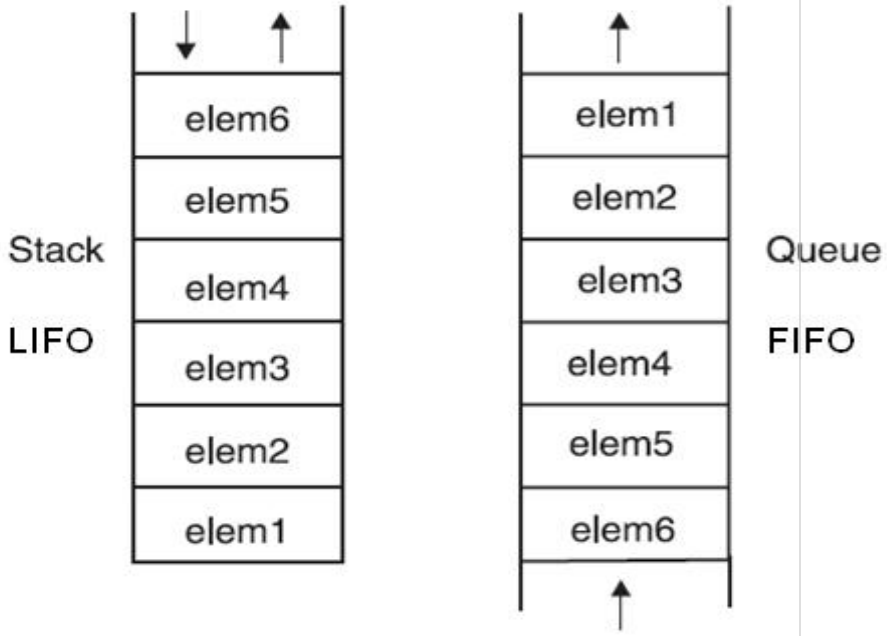
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is CA, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

Da

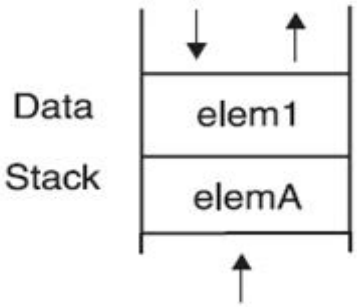


3

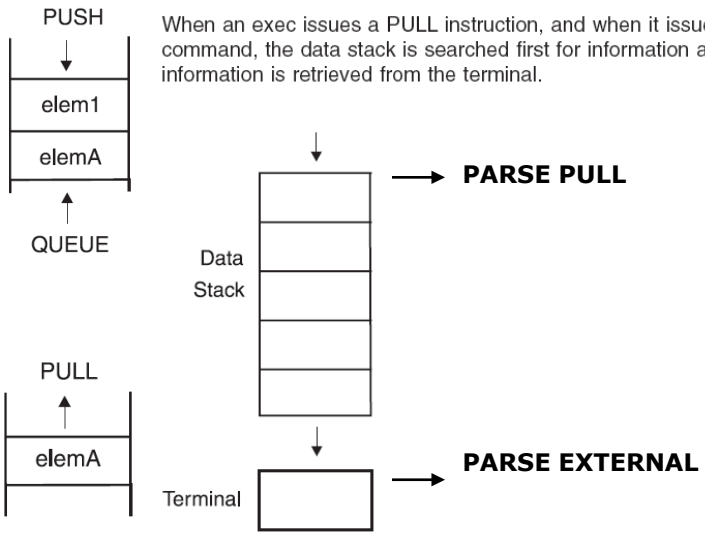
As shown in the following figure, the data stack that REXX u
techniques used in adding elements to stacks and queues.
on the top or the bottom of a data stack. Removal of eleme
however, occurs from the top of the stack only.

Stacks – a method to store data in memory without using variables.

Notice the difference between stack and queue: stack works as LIFO, queue works as FIFO.



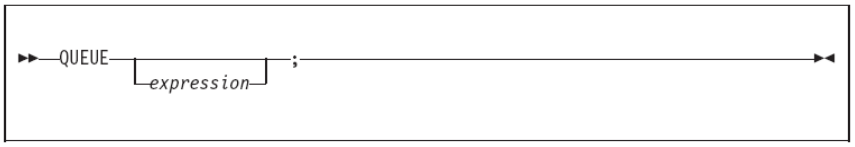
Data Stack - Manipulating with data



4

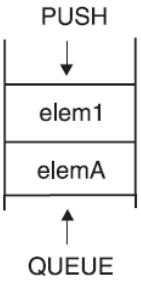
Use PARSE EXTERNAL to read from terminal.

QUEUE



QUEUE appends the string resulting from *expression* to the tail of the external data queue. That is, it is added FIFO (First In, First Out).

If you do not specify *expression*, a null string is queued.



This course has been prepared by Milos Forman for MCoE needs only!

Queue will put the data in the stack in FIFO order.

QUEUE example

```
ADDRESS "TSO"
"CLEAR"
SAY "Please enter your name :"
PARSE EXTERNAL full_name un_used
SAY "Please enter another name :"
PARSE PULL second_name un_used
QUEUE full_name
QUEUE second_name
PARSE PULL name
SAY "The first line off the stack was : "
SAY name
PARSE PULL name
SAY "The second line off the stack was : "
SAY name
```

```
 Please enter your name :
bob
 Please enter another name :
jane
 The first line off the stack was :
 bob
 The second line off the stack was :
 jane
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!

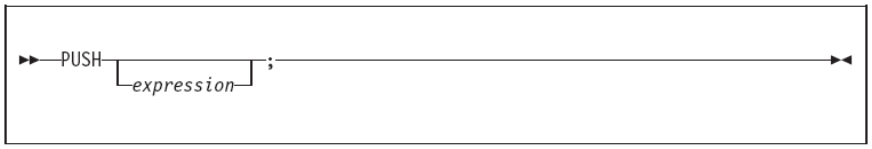
6

PARSE PULL will remove the data from the stack one line at a time. The data will be pulled into a variable.

It can be useful under TSO to only use EXTERNAL instead of PULL when working with stacks, PARSE PULL will go to a stack for data before the screen.

See ‘MCOE.REXA.REXX(RX201114)’

PUSH



PUSH stacks the string resulting from the evaluation of *expression* LIFO (Last In, First Out) onto the external data queue.

If you do not specify *expression*, a null string is stacked.

This course has been prepared by Milos Forman for MCoE needs only!

7

PUSH will put the data in the stack in LIFO order.

PUSH example

```
ADDRESS "TSO"
"CLEAR"
SAY "Please enter your name :"
PARSE EXTERNAL full_name un_used
SAY "Please enter another name :"
PARSE PULL second_name un_used
QUEUE full_name
QUEUE second_name
PARSE PULL name
SAY "The first line off the stack was : "
SAY name
PARSE PULL name
SAY "The second line off the stack was : "
SAY name
```

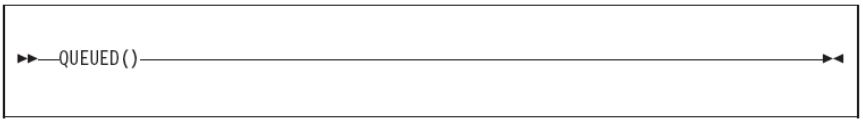
```
 Please enter your name :
bob
 Please enter another name :
jane
 The first line off the stack was :
 bob
 The second line off the stack was :
 jane
 ***
```


There is a mistake in this slide. It is just copy/paste from the previous one. Here should be PUSH instead of QUEUE.

I prefer to use QUEUE - see ‘MCOE.REXA.REXX(FTPCOM)’

See ‘MCOE.REXA.REXX(RX201116)’

QUEUED()



returns the number of lines remaining in the external data queue when the function is called.

The TSO/E implementation of the external data queue is the data stack.

QUEUED() Example

```
DO FOREVER
   SAY "Please enter your name :"
   PARSE UPPER EXTERNAL full_name un_used
   IF full_name = "" THEN DO
      LEAVE
   END
   ELSE DO
      QUEUE full_name
   END
END
no_in_stack = QUEUED()
DO no_in_stack
   PARSE PULL name
   SAY name
END
```

This course has been prepared by Milos Forman for MCoE needs only!

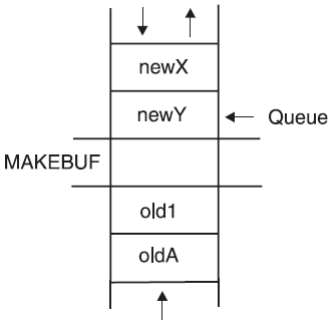
The QUEUED() function will return how many items are in the stack, and can be used to loop until the stack is empty.

See ‘MCOE.REXA.REXX(RX201117)’

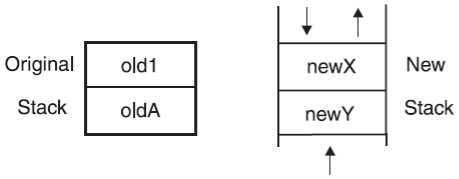
QUEUED() Example (cont.)

```
 Please enter your name :
bob
 Please enter your name :
jane
 Please enter your name :
jim
 Please enter your name :

 BOB
 JANE
 JIM
 ***
```

Multiple Buffers and Stacks

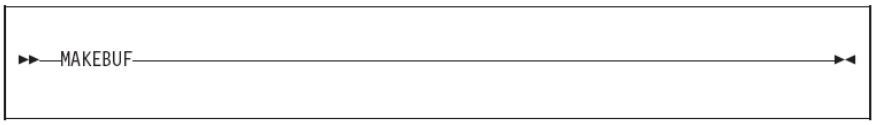


All elements added to the data stack after the NEWSTACK command are placed in the new data stack. The original stack contains the elements placed on the stack before the NEWSTACK command.



More than one stack can be used at any time, but you can only access the current stack. The current stack has to be deleted to access the stacks behind it.

MAKEBUF



Use the MAKEBUF command to create a new buffer on the data stack. The MAKEBUF command can be issued from REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

Initially, the data stack contains one buffer, which is known as buffer 0. You create additional buffers using the MAKEBUF command. MAKEBUF returns the number of the buffer it creates in the REXX special variable RC. For example, the first time an

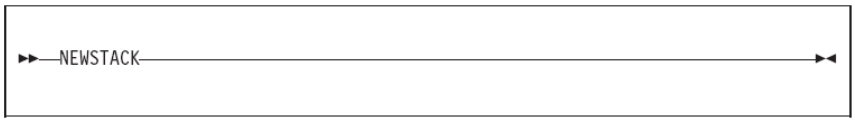
Note: bThe TSO/E implementation of the external data queue is the data stack. The length of an element in the data stack can be up to one byte less than 16 megabytes. The data stack contains one buffer initially, but you can create additional buffers using the TSO/E REXX command MAKEBUF.

This course has been prepared by Milos Forman for MCoE needs only!

13

Let us look to some TSO REXX commands working with stacks.

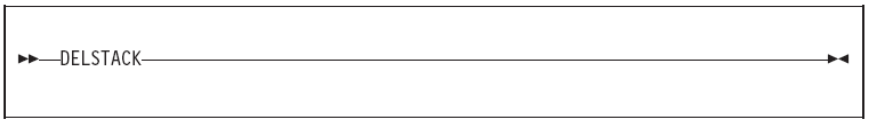
NEWSTACK



creates a new data stack and basically hides or isolates the current data stack. Elements on the previous data stack cannot be accessed until a DELSTACK command is issued to delete the new data stack and any elements remaining in it.

The NEWSTACK command can be used in REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

DELSTACK



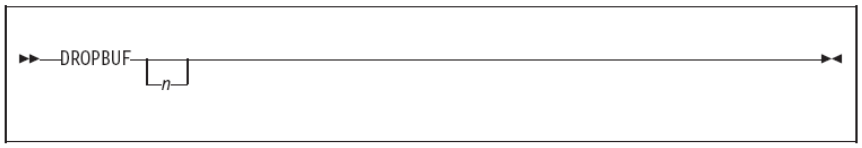
deletes the most recently created data stack that was created by the NEWSTACK command, and all elements on it. If a new data stack was not created, DELSTACK removes all the elements from the original data stack.

The DELSTACK command can be used in REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

This course has been prepared by Milos Forman for MCoE needs only!

After the DELSTACK the previous stack is accessed.

DROPBUF



removes the most recently created data stack buffer that was created with the MAKEBUF command, and all elements on the data stack in the buffer. To remove a specific data stack buffer and all buffers created after it, issue the DROPBUF command with the number (n) of the buffer.

The DROPBUF command can be issued from REXX execs that execute in both the TSO/E address space and non-TSO/E address spaces.

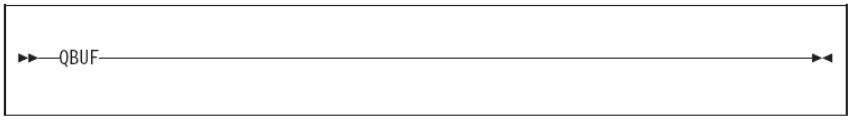
DROPBUF (cont.)

Operand: The operand for the DROPBUF command is:

n specifies the number of the first data stack buffer you want to drop. DROPBUF removes the specified buffer and all buffers created after it. Any elements that were placed on the data stack after the specified buffer was created are also removed. If *n* is not specified, only the most recently created buffer and its elements are removed.

The data stack initially contains one buffer, which is known as buffer 0. This buffer will never be removed, as it is not created by MAKEBUF. If you issue DROPBUF 0, all buffers that were created on the data stack with the MAKEBUF command and all elements that were put on the data stack are removed. DROPBUF 0 effectively clears the data stack including the elements on buffer 0.

QBUF



queries the number of buffers that were created on the data stack with the MAKEBUF command. The QBUF command returns the number of buffers in the REXX special variable RC. If you have not issued MAKEBUF to create any buffers on the data stack, QBUF sets the special variable RC to 0. In this case, 0 is the number of the buffer that is contained in every data stack.

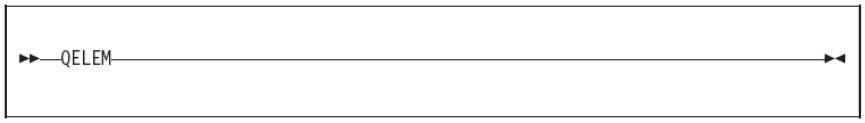
You can use the QBUF command in REXX execs that run in both the TSO/E address space and non-TSO/E address spaces.

QBUF (cont.)

The following table shows how QBUF sets the REXX special variable RC.

Return Code	Meaning
0	Only buffer 0 exists on the data stack
1	One additional buffer exists on the data stack
2	Two additional buffers exist on the data stack
n	*n* additional buffers exist on the data stack

QELEM



queries the number of data stack elements that are in the most recently created data stack buffer (that is, in the buffer that was created by the MAKEBUF command). The number of elements is returned in the REXX special variable RC. When MAKEBUF has not been issued to create a buffer, QELEM returns the number 0 in the special variable RC, regardless of the number of elements on the data stack. Thus when QBUF returns 0, QELEM also returns 0.

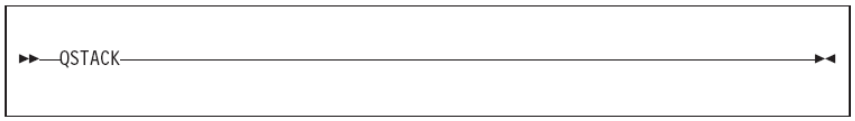
The QELEM command can be issued from REXX execs that execute in both the TSO/E address space and in non-TSO/E address spaces.

QELEM

After the QELEM command processes, the REXX special variable RC contains one of the following return codes:

Return Code	Meaning
0	Either the MAKEBUF command has not been issued or the buffer that was most recently created by MAKEBUF contains no elements.
1	MAKEBUF has been issued and there is one element in the current buffer.
2	MAKEBUF has been issued and there are two elements in the current buffer.
3	MAKEBUF has been issued and there are three elements in the current buffer.
n	MAKEBUF has been issued and there are *n* elements in the current buffer.

QSTACK



queries the number of data stacks in existence for an exec that is running. QSTACK returns the number of data stacks in the REXX special variable RC. The value QSTACK returns indicates the total number of data stacks, including the original data stack. If you have not issued a NEWSTACK command to create a new data stack, QSTACK returns 1 in the special variable RC for the original data stack.

You can use the QSTACK command in REXX execs that run in both the TSO/E address space and in non-TSO/E address spaces.

QSTACK (cont.)

The following table shows how QSTACK sets the REXX special variable RC.

Return Code	Meaning
0	No data stack exists. See “Data Stack Routine” on page 454.

Return Code	Meaning
1	Only the original data stack exists
2	One new data stack and the original data stack exist
3	Two new data stacks and the original data stack exist
n	*n - 1* new data stacks and the original data stack exist

Multiple Stacks example

```
SAY "Please enter your first name : "
PARSE UPPER EXTERNAL fore_name un_used
QUEUE fore_name
"NEWSTACK"
SAY "Please enter your surname : "
PARSE UPPER EXTERNAL sur_name un_used
QUEUE sur_name
SAY "Please enter your surname : "
PARSE UPPER EXTERNAL sur_name un_used
QUEUE sur_name
"QSTACK"
SAY "You have "||RC||" of stacks."
/*---- Empty stacks ----*/
PARSE PULL stuff
SAY stuff
PARSE PULL stuff
SAY stuff
"DELSTACK"
PARSE PULL stuff
SAY stuff
```

```
 Please enter your first name :
bob
 Please enter your surname :
smith
 Please enter your surname :
jones
 You have 2 of stacks.


 SMITH
 JONES


 BOB
 ***
```


"NEWSTACK" creates an additional stack in front of the current stack. The newstack has to be removed before the data on the original stack can be retrieved.

"DELSTACK" deletes the current stack.

"QSTACK" returns the number of stacks into RC.

See 'MCOE.REXA.REXX(RX20111A)'

Unused Stack Data

```
tso_cmd = "LISTC LEVEL(IULC20)"
PUSH tso_cmd
EXIT
```

This course has been prepared by Milos Forman for MCoE needs only!

25

Putting TSO commands on the stack allows them to be executed on completion of the REXX program.

See ‘MCOE.REXA.REXX(RX20111B)’

EXECIO and Stacks

```
dsn_name = "crone90.crone.rexx(test)"
dsn_check = SYSDSN("'"dsn_name"'")
IF dsn_check = "OK" THEN DO
   "ALLOC DD(ddname) DSN('"dsn_name"') SHR REU"
   "EXECIO 1 DISKR ddname (FINIS)"
   PARSE PULL line
   SAY "The first line of the dataset is : "
   SAY line
END
ELSE DO
   SAY dsn_check
END
```

26

The data stack can be used with EXECIO instead of using STEM variables by removing the STEM option and variable.

See ‘MCOE.REXA.REXX(RX20111C)’

Work Section 11.1

- Write a REXX program to accept any number of names to the screen and when they type "STOP", display all the names.
- Store the names in a stack.

```
 Please enter your name :
jane
 Please enter your name :
sue
 Please enter your name :
stop
 JANE
 SUE
 ***
```

27

Write it and test it.

Work Section 11.2

- Write a REXX program to accept a list of names and store in a stack, then create a new stack and accept a number of Date of births and store them in the new stack.
- Display the contents of each stack showing the names in reverse order.

```
 Please enter your name :
bob
 Please enter your name :
fred
 Please enter your name :
stop
 Please enter your DOB :
051276
 Please enter your DOB :
040303
 Please enter your DOB :
stop
 051276
 040303
 FRED
 BOB
 ***
```

28

Write it and test it.

Additional Program

- Using EXECIO read one of your members into a data stack and then display the contents one line at a time with the option to stop displaying the data.

```
 Do you wish to see a line from the stack (Yes/end) ?
y
 /* REXX */
 Do you wish to see a line from the stack (Yes/end) ?
y
 SAY 'Enter your name'
 Do you wish to see a line from the stack (Yes/end) ?
end
 ***
```

29

Write it and test it.