RCSL No: 991-09840

Edition: June 1984

Author: Finn G. Strøbech

Title:

RC8000 Utility Program
Montest

RCSL Nr. 46-F 0087

A/S REGNECENTRALEN
af 1979

Keywords:

RC8000, monitor, monitor tables, process descriptions,
chain tables, message buffers, coredump
basic software, utility program.

Abstract: The montest program will display the contents of monitor tables, process descriptions, chaintables or message buffers taken from primary storage (running system) or a backing storage file (dumped system).
The program is intended to facilitate testing and trouble shooting.

(26 printed pages)

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL Nr. 46-F 0087

Foreword:

This manual describes the use and function of the program montest, used to display monitor tables and other data structures in the monitor.

Section 1 is a short introduction to the program.
Section 2 gives the call conventions and
Section 3 gives the function and the detailed conventions for the various commands to the program.
Section 4 gives the minimal resource requirements to run the program.

The manual is a rewriting of the manual RCSL No 31-D578, montest, mainly caused by a major renewal of the program.

Finn G. Strøbech
A/S Regnecentralen June 1984

TABLE OF CONTENTS PAGE

1. INTRODUCTION.

The program may execute in two different modes, either as
- a normal utility program taking its parameters from the call, or as
- a conversational program taking its parameter from current input zone.

The program may switch from one mode to the other any number of times, governed by commands in the call and in the current input zone.
The program may at any time transfer all of its segments to core and stay core resident during further execution, thereby being independent of the backing storage system and segment i/o.

The program may at any time switch to take the RC8000 monitor information from primary storage directly or from any backing storage file containing an RC8000 coredump (cf. the program movedump, (1).

During display in the conversational mode of certain data structures (internal, external and area process descriptions, message buffers and chaintables) it is possible to
- stop further execution while the latest data structure is inspected
- repeat the display of the latest structure
- direct current output to maybe continue writing in any file before the next structure or before a repeat at the latest one
- redirect current output to the primary output to continue inspection in the conversational mode
- leave the data structure to accept further commands.

The program is able to help the user by displaying the repertoire of commands or by displaying information concerning the single commands on request.

The program will interprete the commands, one by one, until the end of the parameterlist in the call is reached.

2. EXAMPLES.

2.1 Example 1.

the call

```
montest area user. myself
```

will display on current output all the area processes to which the internal process 'myself' is a user.

2.2 Example 2.

The call

```
montest buf sender. myself
```

will display all the message buffers in the message buffer pool in which the internal process 'myself' is recorded as sender.

2.3 Example 3.

The call

```
montest typein
```

will make the program enter the conversational mode and the program will take input from the terminal.

Now any command may be typed, e.g.

```
external devno.10.all buf sender. myself
```

The program will now display the process description for the external process with the device number 10. Following the display will appear the prompt character:

>

telling that the program is ready for a directive.
Now every empty directive (RETURN) will make the next external process appear on the screen followed by the prompt character, until all processes have been displayed.

The directive
> o lp
will make the program write on the terminal:
* o lp
connect current output to the printer and another prompt character will appear.

The directive
> r
will cause the last external process description seen on the screen be repeated, this time on the printer. When done the prompt character will appear on the screen again, ready for another directive.

The directive
> o c
will make the output reappear on the screen. If the file just used should be connected again, writing in the file will be continued, even in backing storage files.

The directive
> f
will make the program quit the command 'external' before all processes have been displayed.
When the command 'external' is left, either because all have been displayed or because of the directive 'f', another command is read from current input, here the command

    buf sender. myself

and the sequence starts all over until the command 'buf' is left.

When no more commands are found in the line, another line is read from current input. This time type:
end
and the program will leave the conversational mode and take the next parameter in the call.
Since no parameters appear after the 'typein', the program will terminate.

2.4 Example 4.

The command

commands

whether in conversational mode or not, will display the repertoire of commands on current output,

and the command

info <command>

<command> being any of these, will display a short description of its use.

3. CALL

$$\left\{\text{<outfile> =}\right\}_{0}^{1} \quad = \quad \text{montest} \quad \left\{\text{<command>}\right\}_{0}^{*}$$

<command> =

commands	
info	<param>
typein	
end	
dump	<param>
core	
lock	
internal	<param>
external	<param>
area	<param>
chain	<param>
buf	<param>
veri	<param>
format	<param>
extra	<param>
lines	<param>
0	<param>

4. FUNCTION.

If <outfile> is specified, current output is connected to <outfile>.

Now the program gets the next command with optional parameters and executes it, until the end of the parameter list in the call is met.

4.1 Commands.

The command is

commands

with no parameters.

The program lists on current output the repertoire of commands.

4.2 Info.

The command is

info <command>

where <command> is the name of one of the commands listed in section 3.

The program displays on current output a short explanation of the parameter syntax and the function of the command.

4.3 Typein.

The command is

typein

with no parameters.

If the program is in the conversational mode, the command has no effect.
If the program is not in the conversational mode, it enters the conversational mode, i.e. reads the next line from current input zone and gets the first command from the line. The program prepares a return to the next command in the call to be executed when conversational mode is left again.

In conversational mode, the display of records from each of the data structures internal, external and area process descriptions, message buffers and chaintables will be followed by the output of a prompt character (the character >) on current input.

The first letter of the first parameter in next line in current input is interpreted as one of the following directives:

r : Skip the rest of the line in input and repeat the latest display unchanged.

o : The next parameter in current input will be handled as a file name, the rest of the line after the filename will be skipped.
Current output will be stacked for later continuation and connected to the file specified. If the file does not exist, a backing storage area of that name will be created. If the file has been connected before, it is reconnected for continued writing. (a maximum of 10 different files ready for reconnection is allowed).
Now a prompt is output and the next directive may be given.

f : The rest of the input line is skipped and the display of records stops. The next command will be executed.

anything else or empty line:
The rest of the input line is skipped. The next record is displayed. If the data structure is emptied, the next command is executed.

4.4 End

The command is

end

with no parameters.

If the program is not in conversational mode, the command has no effect.
If the program is in conversational mode, the mode is left and the next command in the call after the command 'typein' is executed.

4.5 Dump

The command is

dump <dumparea>

where <dumparea> is the name of a backing storage file containing an RC8000 core dump (cf. movedump (1)).

The program sets the coredump mode, disconnects from a possible file, connects to the file specified and reads the monitor key variables from the file. Any succeeding command concerning data structures in the monitor will concern the data structures in the coredump contained in the file until the coredump mode is left again or another file connected.

4.6 Core

The command is

core

with no parameters.

The program will set the core mode, and get the monitor key variables from locations in the primary storage.
Any succeeding command concerning data structures in the monitor will concern the data structures in the primary storage until the coredump mode is entered.

4.7 Lock

The command is

lock

with no parameters.

The program transfers all of its segments to core and locks them, i.e. the entire program stays core resident until it terminates.

4.8 Internal

The command is

internal {all/name.<name>}

where

<name> ::= name of an internal process.

The program will display a number of lines from all internal process descriptions or the one specified by name.
The number of lines is by default all, but may be set otherwise by the command 'lines' (4.16).

4.9 External

The command is

external {all /
user.<user> /
reserver.<reserver> /
name.<name> /
devno.<devno> /
devno.<devno>.all }

main (name)

<user> ::=

<reserver>::=

<name> ::= name of an internal process

<devno> ::= positive integer

The program will display the external process descriptions specified.

all	:	all external processes
user.<user>	:	all external processes with <user> as user of the process
reserver<reserver>	:	all with <reserver> as reserver of the process
name.<name>	:	the one with that name
devno.<devno>	:	the one corresponding to the device number
devno.<devno>.all	:	the same and all succeeding

A number of lines in extension to the process descriptions may be displayed, the number controlled by the command 'extra' (4.15) and the display format controlled by the command 'format' (4.14).

4.10 Area

The command is

area {all / user.<user> / reserver.<reserver> / name.<name>}

The program will display the area processes specified.
The specifikations equal the specifications for the command 'external'.
A number of lines in extension to the process descriptions may be displayed as for external processes.

4.11 Chain

The command is

```
chain {all                /
       docname.<name>}
```

where

<name> ::= an RC8000 name.

The program will display the chaintables specified.

```
all             : all chaintables
docname.<name>  : the one with the document name specified.
```

4.12 Buf

The command is

```
buf {all                                /
     sender.<name>                      /
     receiver.<name>                    /
     sender.<name1>.receiver.<name2>    /
     receiver.<name2>.sender.<name1>    /
     addr.<integer>                     /
     addr.<integer>.all                 }
```

```
<name>    ::=
<name>    ::=
<name>    ::= name of a process
<integer  ::= positive integer
```

The program will scan the message buffer pool and display the message buffers specified.

```
all                 : all message buffers in the entire pool
sender.<name>       : all the ones with <name> as sender
receiver.<name>     : all the ones with <name> as receiver

sender.<name1>.receiver.<name2>
receiver.<name2>.sender.<name1>
                    : all the ones with <name1> as sender and
                      <name2> as receiver
addr.<integer>      : the one with the address specified
addr.<integer>.all: the same and all succeeding
```

4.13 Veri

The command is

veri <first addr> $\left[.\text{<no of halfs>} \right]_{0}^{1}$

<first addr> ::=

<no of halfs> ::= positive integer

The program will display the contents of the words specified in a format controlled by the command 'format' (4.14).

<first addr> : the word addressed by <first addr>, i.e. if odd then <first addr>-1

<no of halfs> : as many words as given by <no of halfs>//2

The words specified should be kept below 'first free', i.e. the first address of the first process created by s and certainly below cpa.

4.14 Format

The command is

format <format> $\left[.\text{<format>} \right]_{0}^{*}$

where

<format> ::= { integer/
octal /
half /
byte /
bit /
text /
all /
code }

The programs sets a format used in the display by the commands 'veri' (4.13) and 'external' (4.9) or 'area' (4.10) in connection with 'extra' (4.15).

The format is valid until changed by another format command.
Default is all except code and bit.
All means all except code.

4.15 Extra

The command is

extra <integer>

where

<integer> ::= positive integer

The program sets an internal value controlling the number of lines displayed in the extension of external and area process descriptions (4.9 and 4.10).
The value is valid until changed by another 'extra' command.
Default is zero.

4.16 Lines

The command is

lines <first line> $\left\{ .\text{<last line>} \right\}_{0}^{1}$

where

<first line> ::=

<last line> ::= positive integer

The program will set the line interval displayed in internal process descriptions (4.8).
The lines are numbered 1,2,3..., line no of last line of bs claims.

First line will be set no lower than 1.
Last line will be set no higher than line no of last line of bs claims.
The values of first and last line are valid until changed by another 'lines' command.
Default is (1, max integer).

4.17 o

The command is

o <name>

where

<name> ::= an RC8000 name.

The program will stack current output for later reconnection and connect it to the file with the specified name.
If the file does not exist, a backing storage area of that name will be created.
If the file has been connected before, it will be reconnected for continued writing (a maximum of 10 different files ready for reconnection may exist).
The command is a command equivalent to the directive of the same name described in 4.3, typein.

5. RESOURCE REQUIREMENTS.

The program may run in a process of only 11000 halfwords, conversational mode or not.
In case the lock command (4.7) is used, the size must be at least 79700 halfwords.
The process will need only 3 message buffers and only 3 area processes, unless the dump command (4.5) is used, in which case the process needs 4 area processes.

6. ERROR MESSAGES.

The messages from the program should be self explaining.

7. REFERENCES.

(1) RCSL No. 31-D661, Movedump.

RETURN LETTER

Title: RC8000 Utility program Montest

RCSL No.: 991-09840

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

_____

_____

_____

_____

Do you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

RCSL Nr. 46-F 0093

. . . . . Fold here . . . . .

. . . . . Do not tear - Fold here and staple . . . . .

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark