Title:

MUS Operating System

Message Interface.

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 43-GL 7419
Edition: June 1978
Author: Dan Andersen

Keywords: RC3600, MUS, Operating System, Message Interface.

Abstract:
Description of the MUS Operating System message interface, i.e. internal commands and requests.

CONTENTS PAGE

1. PREFACE. 1.

This manual contains description of the MUS Operating System process 'S'.

The message interface design has been kept as close as possible to the interface given in the DOMUS Operating System. The DOMUS system is described in DOMUS Users Guide, Part I, by Phillippe Gauguin.

The main differences are a simpler command syntax, no parameter transfer, and no standard error text facilities.

2. INTRODUCTION.

The MUS-system contains the following components:

Monitor
Utility Procedures
Basic I/O
Character I/O
Record I/O
Primary input driver
Console driver
Operating System S

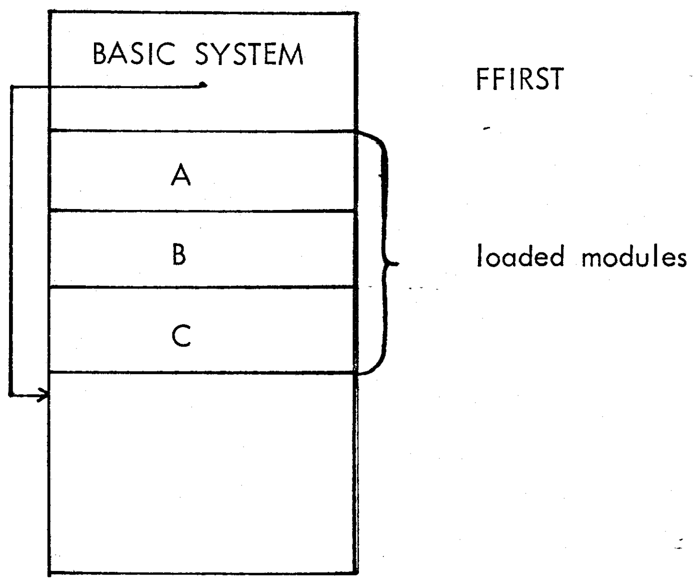
In almost all systems a MUSIL interpreter module is included too.

The operating system S is the manager of core storage resources, the process loader, and the process supervisor.

Commands are received either from the operator on the console input device or from internal processes which send messages requesting operations.

3. CORE STORAGE STRUCTURE. 3.

The core is administrated as a stack, and the first free location in core is found in the page zero variable FFIRST (First of free core) as a word address.



The variable is updated when a process is loaded or removed. The size of the free core-area is, however, only increased when the area occupied by the removed process is physical contiguous with the unoccupied core-area.

When C in the figure is removed FFIRST is updated to point after B. If B is removed before C the core-area occupied by B is not released.

4. COMMUNICATION with S.

After initialization the operating process is in the idle state, waiting for answer from the operator driver or for a message. When an event arrives the following types are treated:

1. Answer

The operator has entered commands on the console device. The S process will execute the command (s), and if necessary request more input.

Error messages are output on the console.

2. Internal command

An output message from a process is received with operation = 3. The data buffer contains commands as a text, with the same format as console entered commands.

3. Internal request

A control message from a process is received with operation = 1b8. The process is removed, and if the bytecount <> 0 the commands present are executed as for 2.

4. Other messages

The messages are dummy, and they are returned with zero status.

When answers are returned the MESS0 contains an error number * 256. The error codes are eqvivalent to error codes used by the DOMUS operating system.

5. FORMATS. 5.

5.1 Internal Command. 5.1

3
COUNT
ADDRESS
NAME ADDRESS

The count is the number of bytes to be interpreted by the operating system.

The address is the byte address of the first character in the data buffer.

Name address is a byte address giving a 6 bytes area, in which a name of the module involved in the current operation is placed in case of error. If the area is not present the value must be zero. The area is zeroized if no error is indicated.

If <commands> is the contents of the data buffer the message will request operations as if

BEGIN <NL> <commands> <NL> END <NL>

was keyed on the console.

The command string may contain an END command, but it need not to be present if bytecount is accurate.

Answer to internal commands:

RESULT * 256
COUNT
NUMBER
NAME ADDRESS

If result is zero the commands were executed succesfully. The count is equal to the count in the original message. The number is irrelevant if result = 0, else it contains a number describing the error i.e. status in case of device error, memory size in case of size error etc.

5.2 Internal Requests.

1b8
COUNT
ADDRESS
NAME ADDRESS

The count and addresses acts as for internal commands. The data buffer is interpreted as

BEGIN <NL> KILL <sender> <NL>

<commands> <NL> END <NL>

Answer:

RESULT * 256
COUNT
NUMBER
NAME ADDRESS

The answer format is eqvivalent to answers returned on internal commands.

If an error is found after removal of the sender-process, the text INTERNAL ERROR is output on the console device.

When the sender process is removed FINIS <sender name> is output on the console, if count is zero.

APPENDIX A - S COMMANDS

BEGIN	Read all commands until terminating END.
BREAK <process>	Break the process with name <process>
CLEAR	Remove all processes and modules in core.
END	Command terminator.
IN <process>	Select the process <process> as input device.
INT [<ident>]	Read a sequence of commands from the file with ident <ident>.
KILL <process>	Remove the process <process> from core.
LIST	List the processes loaded on the console (dummy in internal commands).
LOAD [<ident> *]	Load all the files with names given in ident list.
START <process>	Start the process with name <process>
STOP <process>	Stop the process with name <process>

APPENDIX B - ERROR MESSAGES AND NUMBERS.

Number	Text
0	BRK <break cause>
1	SYNTAX
5	TOO MANY COMMANDS
14	<device> ERROR <error code>
15	NOT ALLOWED
17	ILL
18	SIZE
19	SUM
21	UNKNOWN
22	INTERNAL ERROR

If a device error is found during execution of a console entered command the text

```
<device> ERROR <error code>
```

is output. <device> is the device name, and error code is 20 + number of the leftmost bit set in the device status word.

If operator enter STOP the load is skipped, and if START is entered the operation is repeated.

APPENDIX C - MUSIL EXAMPLE.

The use of internal command in MUSIL can be done as:

```
CONST

COM =   'KILL  PIP <10> LOAD  PAP <10>',
.
:
:
VAR

FC:   FILE 'S', 1, 1, 80, U OF STRING (1);


BEGIN
.
.
.
.
.
.
OPEN (FC, 3);            ! OPEN IN MODE = 3 !
OUTTEXT (FC, COM);       ! COMMAND OUTPUT   !
FC.ZBLOCK: = 0;          ! NO NAME ADDRESS  !
CLOSE (FC, 1);           ! SEND COMMAND     !
.
.
.
END;
```