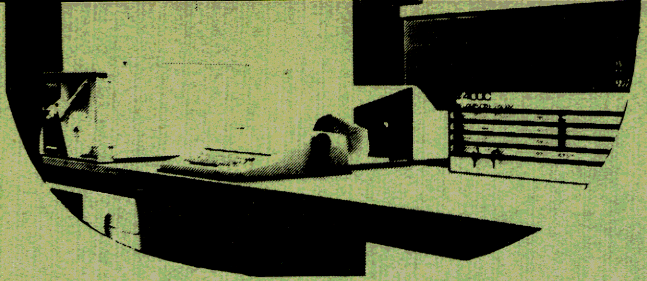

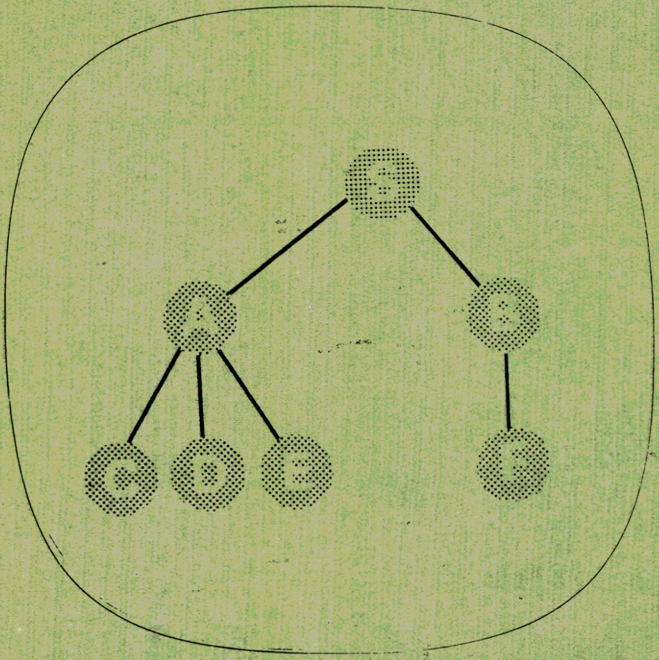

Monitor 3

rc 4000®
DATAMATICS

RCSL NO: 31-D300

Peter Lindblad Andersen

MONITOR 3

2nd edition

A/S Regnecentralen

1974

ISBN 87 7557 013 0

<u>CONTENTS</u> 1 page

Preface

The present report describes version 3 of the RC 4000 monitor. The report presupposes the manual 'Multiprogramming System' (RCSL No: 55-D140), but contains all formats of the calls of monitor procedures, the monitor tables, process descriptions, etc, so that it should be able to stand alone as an abbreviated reference manual.

The design of Monitor 3 was started in the autumn of 1970 by Jørn Jensen, Søren Lauesen, and the author. The reason was, that it had been discovered that Monitor 2 contained major weaknesses, which made it impossible to implement advanced operating systems using monitor 2. The purpose of monitor 3 is solely to remove these weaknesses.

The detailed design and the implementation was done by the author, the prototype version being ready in the summer of 1971.

Peter Lindblad Andersen
A/S Regnecentralen, february 1972.

1. EXTENSIONS COMPARED TO MONITOR 2

1.1. Name hierarchy, access rights

In monitor 3 the catalog and the table of process names have a common hierarchial structure and the search rules used by 'lookup entry', 'change entry' etc. and by 'send message', 'process description' etc. have been changed accordingly. The structure of the catalog (and the set of process names) may be visualized by regarding the catalog as divided into a nested set of subcatalogs, see fig. 1.

To each internal process is associated two subcatalogs, the catalog base and the max base. The initial value (i.e. the value when the process was created) of the catalog base is called the standard base.

The catalog base is the subcatalog in which new names are placed as well as the subcatalog in which a search for a given name is started. If a search in a subcatalog is unsuccessful it is continued in the smallest enclosing subcatalog. An internal process may change its catalog base but only to subcatalogs which are contained in or equal to the max base and which contains, are equal to or are contained in the standard base. Changes in the catalog base are made by calls of a new monitor procedure, set catalog base, which may also be used to change the catalog base of a child process.

An internal process may move a catalog entry from one subcatalog to another by means of a new monitor procedure, set entry base. The subcatalog to which the entry is moved must be one that would be an admissible catalog base for the process, that is, it must be contained in or equal to the max base and contain etc.

Protection is implemented by the rule that an internal process can only make changes in a catalog entry (change entry, rename entry, set entry base etc.) or the corresponding backing storage area (i.e. output data to it) if the entry is in a subcatalog contained in the max base of the process.

These rules mean that internal processes may use common names to gain access to common catalog entries while they still may introduce new names without interfering with each other if their max and standard bases are chosen properly. The rules also implement write as well as read protection and thus makes the use of catalog keys for this purpose superfluos. The reason for using two points in the catalog hierarchi (the max base and the standard base) to define access rights is that it should not be necessary to authorize a process to have access to all files in order to allow it to update global files such as systems programs.

Note that the catalog structure and the search rule closely corresponds to the algol block structure and the rules for defining the scope of an identifier.

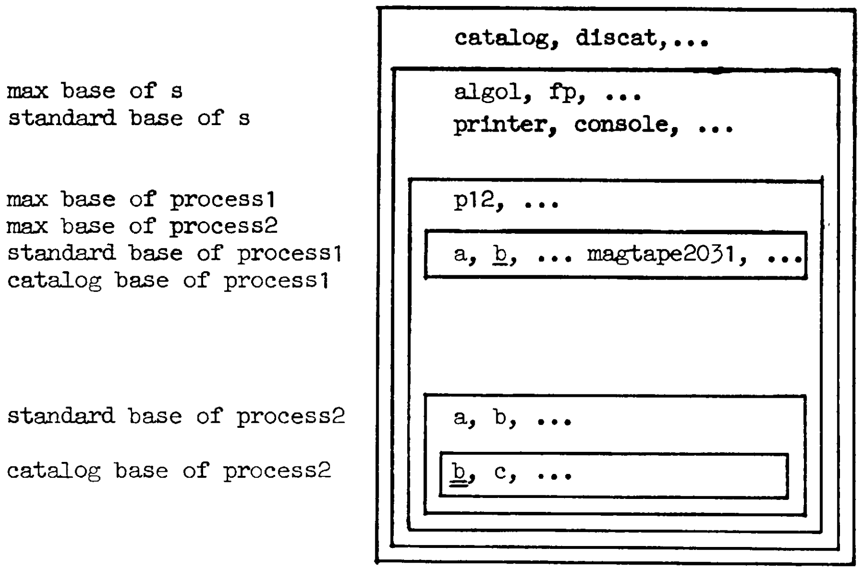

Fig. 1. Process 1 and process 2 both may read the area algol but cannot change it. Both processes may read and write the area p12. A call of the monitor procedure lookup entry(<:b:>, tail addresses, result) will yield the singly underlined version of b if performed by process 1 but the doubly underlined version if performed by process 2.

These concepts are implemented by representing the subcatalogs by pairs of integers (integer intervals) such that a subcatalog represented by the pair (a,b) is contained in a subcatalog represented by (a′,b′) if $a' \le a \le b \le b'$. (If $a=a'$ and $b=b'$ the subcatalogs are equal and if $b<a$ (a,b) does not represent a subcat). The subcatalog to which a name belongs is represented by associating with each name the interval representing the subcatalog. This interval is called the base of the name. In this way all the subcatalogs may be stored in an intermixed way and the hash-defined name key may still be used to speed up the search. The search rule now simply states that among all entries (or processes) with a given name one should choose the one with the smallest interval which contains the catalog base.

1.2. Catalog keys

As mentioned above the concept of catalog keys is no longer necessary for protection purposes. In monitor 3 the catalog keys are used to give a more flexible implementation of the temporary/permanent concept by considering the catalog keys as a set of tags which a process may use to inform the operating system of e.g. how long the catalog entries should remain in the catalog. The number of catalog keys have been reduced to at most 8 (the actual number is a monitor option). Usually only 4 keys are used. As an operating system may want to restrict the use of global subcatalogs to permanent entries, the convention has been adopted that a process may only place an entry in a subcatalog containing the standard base if the entry is tagged with a catalog key larger than or equal to a system parameter (monitor option) called min_global_key.

1.3. Types of backing stores, exchangeable disc packs

The backing storage consists of one or more drums and/or one or more discs. Each of these is identified by a document name. A user process creating a new backing storage area must either specify on which device the area should be allocated (which is done by placing the name of the drum or disc in the part of the entry tail reserved for a document name) or it must indicate whether a drum or a disc area is preferred by placing 0 or 1, respectively, in the first word of the document name part.

To facilitate the use of exchangeable disc packs, each disc pack may be equipped with an auxiliary catalog. The auxiliary catalog has the same form as the main catalog but contains only entries which have been tagged with a catalog key larger than or equal to a system parameter called min aux cat key. An entry describing an area is entered into the auxiliary catalog of the drum or disc on which the area is allocated. Other entries may be entered in an auxiliary catalog by means of a special version of permanent entry, permanent entry in auxiliary catalog.

By keeping an updated catalog on each device the risk of loosing data in connection with a system breakdown has been reduced, and by restricting this catalog to contain only the most permanent entries, the overhead of doing so has been reduced. Note that lookup entry, create area process etc. only search the main catalog so that these procedures will not be slowed down by the existence of an auxiliary catalog.

When a disc pack is removed, the corresponding entries should be removed from the main catalog. This is done by a new monitor procedure remove backing storage. As a complete scan of the main catalog would occupy the monitor for an unacceptable long period, this procedure has been designed so that it must be called once for each name key (= segment in the catalog). When all entries have been removed, the description of the discpack is removed. During removal the disc pack is in a state which will prevent creation of new entries etc. on it.

When a disc pack is mounted, a description of it must be entered into the monitor tables by calling create backing storage and after that the entries in the auxiliary catalog must be inserted into the main catalog by means of insert entry, which must be called once for each entry in the auxiliary catalog. Again, while the entries are being inserted changes in the auxiliary catalog are prevented. Insert entry will check that the areas corresponding to the inserted entries do not overlap.

1.4. Area allocation, slices

A new method of allocating backing storage area has been implemented. Each drum or disc is divided in a number of so called slices each consisting of an integral number of segments. To each slice corresponds a byte in core store in the so called chaintable. A backing storage area consists of an integral number of slices and the corresponding bytes of core store are linked together. In this way the segments in an area need not be physically consecutive and therefore it is always possible to allocate an area if the necessary number of slices are free. It is also possible to extend an area by a call of change entry. As the chaintable is essential for interpretation of the contents of a drum or disc, a copy of the chaintable is kept on the corresponding device. This copy is updated every time a catalog entry tagged with a key larger than or equal to min_aux_cat_key is changed. This and the corresponding rule concerning auxiliary catalog means that only the 'most permanent' files will survive a system breakdown or removal of a disc pack.

The amount of backing storage that an internal process uses is a resource that should be within the control of the operating system. This is achieved in a similar way as the control of the usage of message buffers, area processes etc.: by associating with each internal process the maximal amount of backing storage it may use. These claims specifies for each backing storage document and each catalog key the number of catalog entries and the number of slices the process may tag with this key or a larger key. Use of entries are registered as use of main catalog entries for keys less than min_aux_cat_key and as use of entries in the auxiliary catalog for keys larger than or equal to min_aux_cat_key. The backing storage claims of an internal process may be changed (increased or decreased) by the parent by means of a monitor procedure set backing storage claims.

1.5. Indivisible entry access

For the catalog maintenance performed by an operating system it is essential that the inspection and change (e.g. removal) of a catalog entry may be done in an indivisible way; that is, it should be possible to prevent other processes to change the entry in the time between inspection and change of the entry. This is implemented for area describing entries by the rule that an entry may not be changed if another process has an area process corresponding to it. For all entries, area describing or not, the same effect may be achieved by means of a new monitor procedure, create entry lock process, which will create an (area process((that may have a negative size) and supply the name table address of this process in the same way as send message. Another procedure for catalog maintanance is lookup head and tail, which will do the same work as look up entry except that it will return the head of the entry as well as the tail.

2. DEFINITION OF MONITOR PROCEDURES.

The following defines the monitor procedures in monitor 3 by giving for each procedure the parameters, the error reactions and, if the procedure is changed relative to monitor 1, the changes. In describing the error reactions the following terminology is used:

Document not mounted: The discpack named in the document name of an area entry is unknown, i.e. no chaintable corresponding to it is found.

Document not ready: The document named in the document name of an area entry is not in a state allowing the called monitor procedure, i.e. a mounting or dismounting procedure has been initiated but is not completed.

Name conflict: It is attempted to introduce a name twice in the catalog with base intervals (a,b) and (a',b') such that either a=a' and b=b' or a<a'<b<b'.

Catalog inconsistent: An entry which according to its catalog key should appear in the auxiliary catalog is not found. This error may only occur after an error in either the hardware or the monitor code.

Set interrupt

```
w0  interrupt mask (call)
w1
w2
w3  interrupt address (call)
jd 1<11+ 0
```

Parameter error: interrupt address or return address outside calling process.

No changes.

<u>Process description</u>

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 4
result   = 0 process does not exist
         > 0 process description address
```

Parameter error: process name or return address outside calling process.
No changes.

<u>Initialize process</u>

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 6
result = 0 process initialized
         1 reserved by another process
         2 calling process is not a user
         3 process does not exist
```

Parameter error: process name or return address outside calling process.
Changes: Area process: if the calling process is not a user, result 3 (= process does not exist) is returned.

<u>Reserve process</u>

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 8
result = 0 process reserved
         1 reserved by another process
```

```
        2 calling process is not a user;
          process cannot be reserved
        3 process does not exist
```

Parameter error: process name or return address outside calling process.

Changes: Area process: if the calling process is not a user, result 3 (= process does not exist) is returned. Protection against the calling process is defined by comparing the base interval of the area process with the max base of the calling process. Typewriter (kind 8, 36 or 46): the process may be reserved, and after reservation, the operator request key will send an attention message to the reserver without asking the 'att'-question.

Release process

```
w0
w1
w2
w3  name address (call)
jd 1<11+ 10
```

Parameter error: process name or return address outside calling process.

No changes.

Include user

```
w0  result (return)
w1  device number (call)
w2
w3  name address (call)
jd 1<11+ 12
result = 0 child included as a user
         2 calling process is not a user
         3 described process is not a child
         4 device number does not exist
```

Parameter error: process name or return address outside calling process.

No changes.

Exclude user

```
w0  result (return)
w1  device number (call)
w2
w3  name address (call)
jd 1<11+ 14
result = 0 child excluded as a user
         2 calling process is not a user
         3 described process is not a child
         4 device number does not exist
```

Parameter error: process name or return address outside calling process.
No changes.

Send message

```
w0
w1  message address (call)
w2  buffer address (return)
w3  name address (call)
jd 1<11+ 16
buffer address = 0 buffer claim exceeded
               > 0 selected buffer address
```

Parameter error: process name, message, or return address outside calling process.
No changes, but note that the name table address has priority over the catalog hierarchy in determining to which process the message is sent. This means that in certain cases the destruction of the name table address will not only slow down the call of send message but may also lead to another result (because the hierarchy may have changed).

Wait answer

```
w0  result (return)
w1  answer address (call)
w2  buffer address (call)
w3
jd 1<11+ 18
```

```
result = 1 normal answer
         2 dummy answer, message rejected
         3 dummy answer, message unintelligible
         4 dummy answer, receiver malfunction
         5 dummy answer, receiver does not exist
Answer after i/o operations with result 1:
answer + 0 status word
       + 2 bytes transferred
       + 4 characters transferred
```

Parameter error: buffer address does not point at a message buffer assigned to calling process; answer area or return address outside calling process.

No changes, but note that an answer with bytes transferred = 0 may be returned in case the sender was stopped temporarily after sending the message. In this case, the sender should repeat the message.

Wait message

```
w0  result (return)
w1  message address (call)
w2  buffer address (return)
w3  name address (call)
jd 1<11+ 20
result = 0 buffer claim exceeded
       > 0 process description address
buffer address = 0 buffer claim exceeded
               > 0 selected buffer address
```

Parameter error: name area, message area, or return address outside calling process.

Changes: The calling process must supply a message buffer if it has not already done so (if the buffer address is equal to the buffer address returned by wait event, the calling process supplied the buffer when it called wait event). It is no longer possible to receive a message from a process which does not exist.

Send answer

```
w0  result (call)
w1  answer address (call)
w2  buffer address (call)
w3
jd 1<11+ 22
result = 1 normal answer
         2 dummy answer, message rejected
```

3 dummy answer, message unintelligible
4 dummy answer, receiver malfunction
5 dummy answer, receiver does not exist

Parameter error: buffer address does not point at message buffer received by calling process; answer area or return address outside calling process; illegal result value (note: result 5 = receiver does not exist is legal for simulation purposes).
No changes.

Wait event

```
w0  result (return)
w1
w2  last buffer address (call)
    next buffer address (return)
w3
jd 1<11+ 24
result = 0 message;
             next buffer address = 0  buffer claim exceeded
         1 answer
```

Parameter error: last buffer address does not point at message buffer in the queue of the calling process, return address outside calling process.
Changes: The calling process must supply a message buffer if next buffer address points to a message. If the queue of the calling process contains a message for which the calling process has already supplied a buffer (in a previous call of wait event) the buffer supplied by the calling process is released.

Get event

```
w0
w1
w2  buffer address (call)
w3
jd 1<11+ 26
```

Parameter error: buffer address does not point at message buffer in the queue of the calling process; return address outside calling process.

Get clock

```
w0  clock(0:23) (return)
w1  clock(24:47) (return)
w2
w3
jd 1<11+ 36
```

Parameter error: return address outside calling process.
No changes.

Set clock

```
w0  clock(0:23) (call)
w1  clock(24:47) (call)
w2
w3
jd 1<11+ 38
```

Parameter error: return address outside calling process;
function forbidden in calling process.
No changes.

Create entry

```
w0  result (return)
w1  tail address (call)
w2
w3  name address (call)
jd 1<11+ 40
result = 0 entry created
         2 catalog i/o error;
           document not mounted or document not ready
         3 name conflict
         4 claims exceeded
         5 catalog base of calling process does not allow
           creation of entry
         6 name format illegal
```

Parameter error: name, tail area, or return address outside
calling process.

Changes: Area entry, i.e. the first word in the tail is greater than or equal to 0: Either the second word of the tail must be 0 or 1 or else word 2 to word 5 must contain the document name corresponding to a drum or a disc pack; in the first case, the area is allocated on a document on which the calling process has sufficient claims (preferring a drum if word 2 is 0 and a discpack if word 2 is 1); in the second case, the area is allocated on the document with the given name. If the size specified by the tail is not an integral number of slices, the size is increased to an integral number of slices. After the creation, the tail area is changed to contain the actual number of segments in the area and the document name of the document on which it was allocated. The head of a catalog entry has been changed and the concept of creation number is no longer implemented. The procedure is not privileged.

Lookup entry

```
w0  result (return)
w1  tail address (call)
w2
w3  name address (call)
jd 1<11+ 42
result = 0 entry looked up
         2 catalog i/o error
         3 entry not found
         6 name format illegal
```

Parameter error: name, tail area, or return address outside calling process.
No changes, but cf. lookup head and tail, jd 1<11+ 76.

Change entry

```
w0  result (return)
w1  tail address (call)
w2
w3  name address (call)
jd 1<11+ 44
result = 0 entry changed
```

```
          2 catalog i/o error;
            document not mounted or document not ready
          3 entry not found
          4 entry protected, i.e. base of entry name not
            contained in max base of calling process
          5 area process (or entry lock process) used by
            another process
          6 name format or new size illegal; claims exceeded
          7 catalog inconsistent
```

Parameter error: name, tail area, or return address outside calling process.

Changes: Area entry: the size may be increased or decreased. The document name in the tail is not changed and has no influence. If the area is used as an area process, the size of the process is changed in accordance with the new size. The tail area is changed to contain the actual new size. The procedure is not privileged.

Rename entry

```
w0  result (return)
w1  new name address (call)
w2
w3  name address (call)
jd 1<11+ 46
result = 0 entry renamed
         2 catalog i/o error;
           document not mounted or document not ready
         3 entry not found
         4 entry protected, i.e. base of entry name not
           contained in max base of calling process
         5 area process (or entry lock process) used by
           another process
         6 name format (old or new name) illegal
         7 catalog inconsistent
```

Parameter error: old name, new name, or return address outside calling process.

Changes: If the entry is used as a process, the calling process must be the only user. The name of the process is changed to the new name. The procedure is not privileged.

Remove entry

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 48
result = 0 entry removed
         2 catalog i/o error;
           document not mounted or document not ready
         3 entry not found
         4 entry protected, i.e. base of entry name not
           contained in max base of calling process
         5 area process (or entry lock process) used by
           another process
         6 name format illegal
         7 catalog inconsistent
```

Parameter error: name or return address outside calling process.
No changes, except that the procedure is not privileged.

Permanent entry

```
w0  result (return)
w1  catalog key (call)
w2
w3  name address (call)
jd 1<11+ 50
result = 0 entry permanent
         2 catalog i/o error;
           document not mounted or document not ready
         3 entry not found; name conflict (in auxiliary catalog)
         4 entry protected, i.e. base of entry name not con-
           tained in max base of calling process; key illegal
         6 name format illegal; claims exceeded
         7 catalog inconsistent
```

Parameter error: name or return address outside calling process.
Changes: Permanent entry does no longer protect the entry, except by describing it in the auxiliary catalog if the key is not less than min aux cat key. If the catalog key is less than min global key, the base of the entry name must be contained in the standard base of the calling process, otherwise error 4 (= key illegal) is returned. Result = 3, name conflict, is possible

because the auxiliary catalog may contain an entry which has not been succesfully inserted into the main catalog. The procedure is not privileged. Cf. the procedure permanent entry in auxiliary catalog, jd 1<11+ 90.

Create area process

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 52
result = 0 area process created
         1 area claims exceeded
         2 catalog i/o error
         3 entry not found
         4 entry does not describe an area
         6 name format illegal
```

Parameter error: name address or return address outside calling process.
No changes, but cf. create entry lock process, jd 1<11+ 92.

Create peripheral process

```
w0  result (return)
w1  device number (call)
w2
w3  name address (call)
jd 1<11+ 54
result = 0 peripheral process created
         1 function forbidden in calling process
         2 calling process is not a user; catalog i/o error
         3 name conflict
         4 device number does not exist
         5 device is reserved by another user
         6 name format illegal
```

Parameter error: name or return address outside calling process.
Changes: Name conflict includes name conflict in the catalog as well as in the monitor name table.

Create internal process

```
w0  result (return)
w1  parameter address (call)
w2
w3  name address (call)
jd 1<11+ 56
result = 0 internal process created
         1 storage area outside calling process;
           claims exceeded; illegal protection;
           max base or standard base not contained
           in corresponding base of calling process
         2 catalog i/o error
         3 name conflict
         6 name format illegal
```

Parameter error: name, parameters, or return address outside calling process.

Changes: The parameterlist has the following format:

```
+ 0 first storage address
+ 2 top storage address
+ 4 buffer claim, area claim
+ 6 internal claim, function mask
+ 8 protection register, protection key
+12 max base (double word)
+16 standard base = catalog base (double word).
```

Name conflict includes name conflict in the catalog as well as in the monitor name table. Note that the internal process is not given claims on the backing store by create internal process.

Start internal process

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 58
result = 0 internal process started
         2 state of process does not permit start
         3 described process is not a child
         6 name format illegal
```

Parameter error: name or return address outside calling process.

No changes.

Stop internal process

```
w0  result (return)
w1
w2  buffer address (return)
w3  name address (call)
jd 1<11+ 60
result = 0 stop initiated
         3 described process is not a child
         6 name format illegal
buffer address = 0 buffer claim exceeded
               > 0 selected buffer address
```

Parameter error: name or return address outside calling process.

No changes, except that the protection of the child's core area is changed to the key of the calling process.

Modify internal process

```
w0  result (return)
w1  register address (call)
w2
w3  name address (call)
jd 1<11+ 62
result = 0 internal process modified
         2 state of process does not permit modification
         3 described process is not a child
         6 name format illegal
```

Parametererror: name, registers, or return address outside calling process; instruction counter outside child process.

```
Register address: working register 0
               + 2 working register 1
               + 4 working register 2
               + 6 working register 3
               + 8 exception register
               +10 instruction counter
```

No changes.

Remove process

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 64
result = 0 process removed
         2 state of internal process does not permit removal;
           calling process is not a user of external process
         3 described process is not a child or does not exist
         5 peripheral process reserved by another process
         6 name format illegal
```

Parameter error: name or return address outside calling process.
Changes: Entry lock process: treated like an area process; pseudo process: if the pseudo process is a child of calling process, it is removed and the area claim of the calling process is increased by one; Internal process: protection keys are not changed. The procedure is not privileged.

Generate name

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 68
result = 0 name generated
         2 catalog i/o error
```

Parameter error: name or return address outside calling process.
No changes, the generated name does not exist with any base.

Copy

```
w0  result (return)
w1  first address (call)\ bytes moved (return)
w2  buffer address (call)/
w3  last address (call), characters moved (return)
jd 1<11+ 70
```

```
result = 0 area copied
         2 sender of buffer is stopped  (w1 and w3  undefined at
           return)
         3 buffer  describes  input  or  output  outside senders
           area; message regretted;  operation or mode in buffer
           illegal (w1 and w3 undefined at return)
```

Parameter error: as send answer; area or return address outside calling process.

The buffer must define input from or output to the calling process. The procedure will move the area defined by first address and top address to or from the area defined by the buffer (according to the standard format of input/output messages) according to the operation in the buffer. The mode of the operation must be 0.

Set catalog base

```
w0  base (lower limit of the interval) (call)
    result (return)
w1  base (upper limit of the interval) (call)
w2
w3  name address (call)
jd 1<11+ 72
result = 0 catalog base set
         2 state of internal process does not permit
           modification
         3 described process is not a child
         4 new base illegal
         6 name format illegal
```

Parameter error: name or return address outside calling process.

Changes the catalog base of an internal process. The process will be the calling process when the first word of the name contains a 0 and it will be a child process otherwise. If it is a child process it must be in the state 'waiting for start by parent'. The new catalog base must be contained in the max base of the calling process and it must contain, be equal to or be contained in the standard base of the calling process.

Set entry base

```
w0  base (lower limit of the interval) (call), result (return)
w1  base (upper limit of the interval) (call)
w2
w3  name address (call)
jd 1<11+ 74
result = 0 entry interval set
         2 catalog i/o error:
           document not mounted or document not ready
         3 entry not found; name conflict (at the new base)
         4 entry protected, i.e. old base of entry name not con-
           tained in max base of calling process;
           key, new base combination illegal
         5 area process (or entry lock process) used by another
           process
         6 name format illegal
         7 catalog inconsistent
```

Parameter error: name or return address outside calling process.
The procedure will set the base of the catalog entry specified by the name at name address. If the catalog key of the entry is less than min global key the new base must be contained in the standard base of the calling process, otherwise the new base must be contained in the max base of the calling process, and must contain, be equal to or be contained in the standard base of the calling process. If the entry is used as a process the name base of the process is changed to the new base.

Lookup head and tail

```
w0  result (return
w1  entry address (call)
w2
w3  name address (call)
jd 1<11+ 76
result = 0 entry looked up
         2 catalog i/o error
         3 entry not found
         6 name format illegal
```

Parameter error: name, entry area, or return address outside calling process.
The procedure corresponds to lookup entry, except that it moves both head and tail (34 bytes) to the entry area.

Set backing storage claims

```
w0  result (return)
w1  claim list address (call)
w2  document name address (call)
w3  name address (call)
jd 1<11+ 78
result = 0 backing storage claims set
         1 claims exceeded
         2 backing storage document with given name not found
         3 process defined by name at name address is not child
           of calling process
         6 name format illegal
```

Parameter error: claim list, document name area, name area, or return address outside calling process.

Format of the claim list:

```
         claim list address + 0 : entry claim key = 0
                            + 2 : segment claim key = 0
                            + 4 : entry claim key = 1
                            ...
         claim list address + 4 × max key
                            + 0 : entry claim key = max key
                            + 2 : segment claim key = max key
```

The procedure will transfer claims on the backing storage document specified by the document name between the calling process and the process specified by the name (which must be a child of the calling process). The number (negative or positive) of entries and segments specified by the claim list are subtracted from the claims of the calling process and added to the claims of the child process. If the number of segments specified is not an integral number of slices, it is increased to an integral number of slices. After the call, the claim list has been changed to the number of segments actually used (an integral number of slices). Result 1 (= claims exceeded) is returned if either the claims of the calling process or the claims of the child would become negative by the transfer; in this case nothing is changed in the claims.

Create pseudo process

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 80
result = 0 pseudo process created
         1 (area) claims exceeded
         2 catalog i/o error
         3 name conflict
         6 name format illegal
```

Parameter error: name or return address outside calling process.

The procedure will create a pseudo process with the given name and with the calling process as parent. Any messages sent to the pseudo process will be linked to the queue of the parent. The area claim of the calling process is decreased by one.

Regret message

```
w0
w1
w2  buffer address (call)
w3
jd 1<11+ 82
```

Parameter error: buffer address does not point at message buffer sent by calling process; the buffer defines an operation which cannot be regretted i.e. the operation is uneven.

The buffer claim of the calling process will be increased by one. If the buffer has not yet been received it will be released, otherwise it will be released when the receiver sends an answer.

Create backing storage

```
w0  result (return)
w1  device number (call)
w2
w3  chaintable address (call)
jd 1<11+ 84
result = 0 backing storage created
         2 catalog i/o error
         3 name conflict
         4 device number does not exist;
           device is not drum or disc;
           the calling process is not a user of the device;
           the device is reserved by another process
         5 core storage for the chaintable is not available in
           the monitor; the chaintable is inconsistent
         6 name format illegal (entry name or document name in
           chaintable)
```

Parameter error: chaintable area or return address outside calling process.

The procedure will include the device in the backing storage system by allocating a chaintable in the monitor, inserting the description of the auxiliary catalog in the main catalog, reserving the device for procfunc (= the part of the monitor implementing the interruptable monitor functions), and giving the calling process all claims on the new backing storage device. The chaintable in the monitor is initialized by means of the core area pointed to by w3, so that it describes only the area holding a copy of the chaintable and the auxiliary catalog. Unless the auxiliary catalog is empty, all entries in it must be inserted in the main catalog before any catalog operations are attempted on the new backing storage device (any such operation will give the result 'document not ready'). The format of the chaintable is given in the appendix.

Insert entry

```
w0  result (return)
w1  entry address (call)
w2
w3  chaintable address (call)
jd 1<11+ 86
```

```
result = 0 entry inserted
         2 catalog i/o error; document not mounted
         3 name conflict
         4 claims exceeded
         5 entry not consistent with chaintable; base of entry
           is not contained in max base of the calling process;
           catalog key of entry is less than min global key and
           base of entry is not contained in standard base of
           the calling process.
         6 name format illegal
```

Parameter error: entry area, chain table area, or return address outside calling process.

The procedure will check that the entry and the chain table presented in the call are consistent with the chain table initialized by create backing storage and will then insert the entry in the main catalog. When all entries have been inserted (or rejected on account of name conflict or base violation) the backing storage document is released for general use.

Remove backing storage

```
w0  result (return)
w1  name key (call)
w2
w3  document name address (call)
jd 1<11+ 88
result = 0 all entries corresponding to name key have been
           removed
         2 catalog i/o error; document not mounted
         3 name key out of range (negative or > size of catalog)
         4 the calling process has not all claims on the
           to be removed
         5 it is attempted to remove an entry which is used as
           an area (or entry lock) process by another process
         7 catalog inconsistent
```

Parameter error: document name area or return address outside calling process.

The procedure will remove all entries in the main catalog which have names that hashes to the value of name key (p 43 in

'Multiprogramming system') and which are permanent in the auxiliary catalog of the document or describe areas on the document. The calling process must have all claims on the backing storage document. When all entries permanent in the auxiliary catalog have been removed, the chaintable in the monitor is released and the device is no longer reserved by procfunc.

Permanent entry in auxiliary catalog

```
w0  result (return)
w1  catalog key (call)
w2  document name (call)
w3  name address (call)
jd 1<11+ 90
result = 0 entry permanent
         2 catalog i/o error;
           document not mounted or document not ready
         3 entry not found; name conflict (in auxiliary catalog)
         4 entry protected, i.e. base of entry name not
           contained in max base of calling process; key illegal
         5 area (or entry lock) process used by another process
           entry is already permanent in another auxiliary
           catalog
         6 name format illegal; claims exceeded
         7 catalog inconsistent
```

Parameter error: name area, document name area, or return address outside calling process.

The procedure corresponds to the procedure permanent entry, except that an entry which does not describe an area may be inserted in the auxiliary catalog on the document. If the procedure is applied to an entry which describes an area, the document name in the entry and in the call must be the same.

Create entry lock process

```
w0  result (return)
w1
w2
w3  name address (call)
jd 1<11+ 92
```

result =	0	entry lock process created
	1	area claims exceeded
	2	catalog i/o error
	3	entry not found
	6	name format illegal

Parameter error: name area or return address outside calling process.

The procedure will do the same as create area process, except that the entry need not describe an area and that the name table address (cf. send message) corresponding to the process created will be stored in the word with address 'name address + 8'. The procedure is used to implement indivisibility between several accesses to the same catalog entry.

3. DEFINITION OF CONSOLE COMMANDS

The basic operating system, s, has been changed slightly to make it possible to utilize the new monitor facilities. The full utilization of these facilities will, however, only be possible under a more advanced operating system such as Boss 2. One of the console parameters, catalog mask, has been removed and a few new console parameters have been added. The new console parameters (cf. Multiprogramming system, p 129) are:

textstrings:	backing storage document
integers:	max base, lower and upper limit
	standard base, lower and upper limit
	aux. catalog entry claims
	backing storage segment claims
	main catalog entry claims
booleans:	all backing storage resources

When a process is created, it will be given all the backing storage resources available if the boolean 'all backing storage resources' is true; otherwise the process will be given the main catalog entry claims on the main catalog, and the aux. catalog entry claims and the backing storage segment claims on the backing storage document with name 'backing storage document'. Note, that in the latter case the process will be given the same claim on all catalog keys. The integers max base and standard base will be used for the corresponding parameters of the process. All names introduced by s will have the same base, that is the catalog base of s cannot be changed.

New and changed commands:

catalog command	has been removed
date command	has been removed
newdate command	has been removed

<u>All command</u>

```
        all <process name>
```

Assigns the name to the console parameter process name, sets the boolean all backing storage resources to true and sets all claims defining parameters to the maximum value, i.e. the value that would be displayed by a max command.

<u>Base command</u>

```
        base <ll standard base> <ul standard base>  or
        base <ll standard base> <ul standard base> <ll max base>
            <ul max base>
```

Assigns the integers to the console parameter(s) standard base (and max base) (ll = lower limit, ul = upper limit). Note that the input mechanism of s will only read positive integers. A negative number, n, may be entered as 2**24 + n (2-complement representation).

<u>Work command</u>

```
        work <main entry> <aux entry> <segments> <document name>
```

Assigns the 3 numbers and the string to the console parameters main catalog entry claim, aux. catalog entry claim, backing storage segment claim, and backing storage document, respectively.

<u>Bs command</u>

```
        bs <entries> <segments> <document name> ...
```

Increases the claims of the process defined by the console parameter process name by the amount given by entries and segments for all keys on the backing storage document defined by document name.

<u>Print command</u>

```
        print <from> <to>
```

Prints the contents of the core store from the address <from> to the address <to> for each word as a signed number, the two bytes as two unsigned numbers, the word as an octal number and as a 3 character textstring. The printing is performed on the device with the name 'printer'. Only implemented in debugging versions.

<u>Replace command</u>

```
        replace <area name>
```

The area must contain an independant program (content key = 8). This program is loaded replacing the code of s.

4. FORMAT OF MONITOR TABLES, PROCESS DESCRIPTIONS ETC.

Monitor table (absolute addresses)

8	interrupt number
10	saved instruction counter
12	interrupt response
14	start key response
16	interrupt 0 service routine
...	
64	interrupt 24 service routine (24 = non-existing interrupt)
66	current process
68	next process in time slice queue
70	last process in time slice queue
72	name table start
74	first device in name table
76	first area in name table
78	first internal in name table
80	end of process part of name table
82	next message buffer
84	last message buffer
86	message pool start
88	message pool end
90	message buffer size
92	first drum chain in name table
94	first disc chain in name table
96	top chain table part of name table, i.e. the first unused entry
98	chain table corresponding to main catalog
100	max catalog key, relative address of last byte in internal process description
102	maximum time slice
104	time slice
106	
108	} time in units of 0.1 msec
110	} since dec 31, 1967.
112	clock value sensed
114	
116	number of storage bytes
118	min global key, min aux cat key
120ff	entry points of monitor procedures

Message buffers (relative addresses)

+0	next message buffer
+2	last message buffer
+4	receiver or answer type
+6	sender
+8 to + 22	message or answer

input/output messages have a standard format:

	message	answer
+8	operation, mode	status word
+10	first storage address	number of bytes
+12	last storage address	number of characters
+14	segment number	

The possible states of a message buffer is defined by the sign of the receiver and sender address:

sender	receiver	state
+	+	message pending (i.e. buffer claim of receiver has not been decreased on account of this message)
+	-	message received (i.e. buffer claim of receiver has been decreased on account of this message)
-	+	not possible
-	-	message received, but regretted by sender

The answer type is coded as follows:

1	normal answer
2	dummy answer, message rejected
3	dummy answer, message unintelligible
4	dummy answer, receiver malfunction
5	dummy answer, receiver does not exist

Format of a message sent from an attention key on a typewriter terminal: all words of message contain 0.

Internal process descriptions (relative addresses)

-4, -2 }	base of process name (double word)
0	kind
+2 to +8	process name
+10	stop count, state
+12	identification bit
+14	next message buffer in event queue
+16	last message buffer in event queue
+18	next process in time slice queue

+20	last process in time slice queue		
+22	first address of core area		
+24	top address of core area		
+26	buffer claim, area claim		
+28	internal claim, function mask		
+30	not used		
+32	protection register, protection key		
+34	interrupt mask		
+36	interrupt address		
+38	working register 0		
+40	working register 1		
+42	working register 2		
+44	working register 3		
+46	exception register		
+48	instruction counter		
+50	parent description address		
+52	time quantum		
+54 +56	run time (double word)		
+58 +60	start run (double word)		
+62 +64	start waiting time (double word)		
+66	wait address		
+68 +70	catalog base (double word)		
+72 +74	max base (double word)		
+76 +78	standard base (double word)		
+80	entry claim, slice claim	(key = 0 on first	backing storage document
+82		(key = 1 on first	
...			
+80 + 2×max key		(key = max key on first	
+80 + 2×max key +2		(key = 0 on second	
etc.			

<u>Peripheral process description</u> (relative addresses)

-4 -2	base of process name (double word)
0	kind
+2 to +8	name
+10	device_number × 64

+12	reserver
+14	users
+16	next message in event queue
+18	last message in event queue
+20	interrupt address

The following words are used as working storage by the peripheral process, the number of words depends on the kind of the process.

Area process description (relative addresses)

-4	
-2	base of process name
0	kind (= 4)
+2 to +8	name
+10	process description address of peripheral process
+12	reserver
+14	users
+16	first slice
+18	number of segments
+20 to +26	document name

Lock entry process
Same as area process, number of segments may be negative.

Pseudo process description (relative addresses)

-4 } -2 }	base of process name (double word)
0	kind (= 64)
+2 to +8	name
+10	process description address of parent process

A pseudo process description is allocated among the area process descriptions, i.e. the name table entry of a pseudo process is found in the part of the name table defined by the contents of absolute addresses 76 and 78.

Catalog entry (relative addresses)

0	first slice, name_key × 8 + catalog_key
+2 } +4 }	base of entry name (double word)
+6 to +12	name
+14	size
+16 to +22	document name
+24 to +33	optional

The interpretation of a catalog entry depends on the value of the size field:

size ≥ 0 first slice is the relative address of the first slice in the chain table corresponding to document name. Size is the length of the area in segments.

size < 0 first slice is either 0 or equal to $2 \times n - 2048$ if the entry is described in the auxiliary catalog corresponding to chain table number n (the address of this chain table may be found as the content of (content of(92) + 2xn)). If size is negative, the document name is not used by the monitor.

Chaintable (relative addresses)

-36	relative address of claims on this document in an internal process description
-34	first slice of catalog, name_key × 8 + catalog_key
-32 } -30 }	base of catalog name (double word)
-28 to -22	name of catalog on this document
-20	size of catalog
-18 to -12	document name
-10	not used
- 8	slicelength
-6	last slice, first slice in chaintable area (relative addresses)
-4	auxiliary catalog, not used
-2	entries in catalog, entries in catalog not in main catalog
0	slice link, slice link,
etc.	

Note that the part of the chain table from relative address -34 to -1 is the catalog entry describing the auxiliary catalog. The byte 'auxiliary catalog' is -1 if the entry describes an auxiliary catalog, 0 otherwise. The byte 'entries in catalog not in main catalog' is 0 unless the document is being mounted or dismounted. When this is the case, it gives the number of entries which have not yet been inserted in the main catalog or the number of entries which have been removed from the main catalog. The slicelink is the value to be added to the address of this slice to obtain the address of the next slice of a chain, the last slice of a chain has slicelink = 0, a free slice has slicelink = -2048.

A/S REGNECENTRALEN

HEADQUARTERS: FALKONER ALLÉ 1 · DK-2000 COPENHAGEN F · DENMARK
PHONE: (01) 10 53 66 · TELEX: 16282 RCHQ DK · CABLES: REGNECENTRALEN

AUSTRIA
DENMARK
FINLAND
GERMANY
HOLLAND
NORWAY
SWEDEN