DRAFT: 1/1/68

Identification

Active Meter Table Management: Initial Implementation

Howard Greenbaum

Purpose

This section describes the hardcore ring subroutines for managing the Active Meter Table. Since all metering information will be accumulated in the AMT, there will be no accounting information in paged storage in the initial implementation.

Description of "start meter"

Two calls are provided for managing the AMT. The first is:

```
        start_meter (account, amtindex)
```

This entry is called whenever a process or a segment becomes active. Its purpose is to insure that an entry in the AMT exists corresponding the to the account number given as a parameter of the call. The call comes from either the Process Activation module or Segment Activation module at a time when page faults are permitted.

Implementation of"Start Meter"

In the initial implementation of the metering subsystem a very straightforward procedure will be used. Since all metering information will be accumulated in the AMT, entries will never be deleted, only added. Thus when a call to start_meter is made, the algorithm searches the AMT entries for a corresponding account number. If a match is found the user count is increased by one and the position of the entry in the AMT is returned as the"amtindex." parameter. Since, as we mentioned above, no entries are deleted; when the algoritms reaches the first empty entry, it notes that no entry exists corresponding to this account number. It then creates a new entry in the table for this account number, initializes portions of it, sets the user count to one (1) and returns the position of the entry in the AMT as the "amtindex "parameter. If the table is full, a call to the system "trouble" module is made.
The "amtindex", or relative pointer to the AMT entry, which is returned to the calling program, is stored in the AST entry or the process data block, depending on where the call originated. It is used by the metering calls to find the AMT entry in order to record resource-usage figures, without having to search the AMT each time.

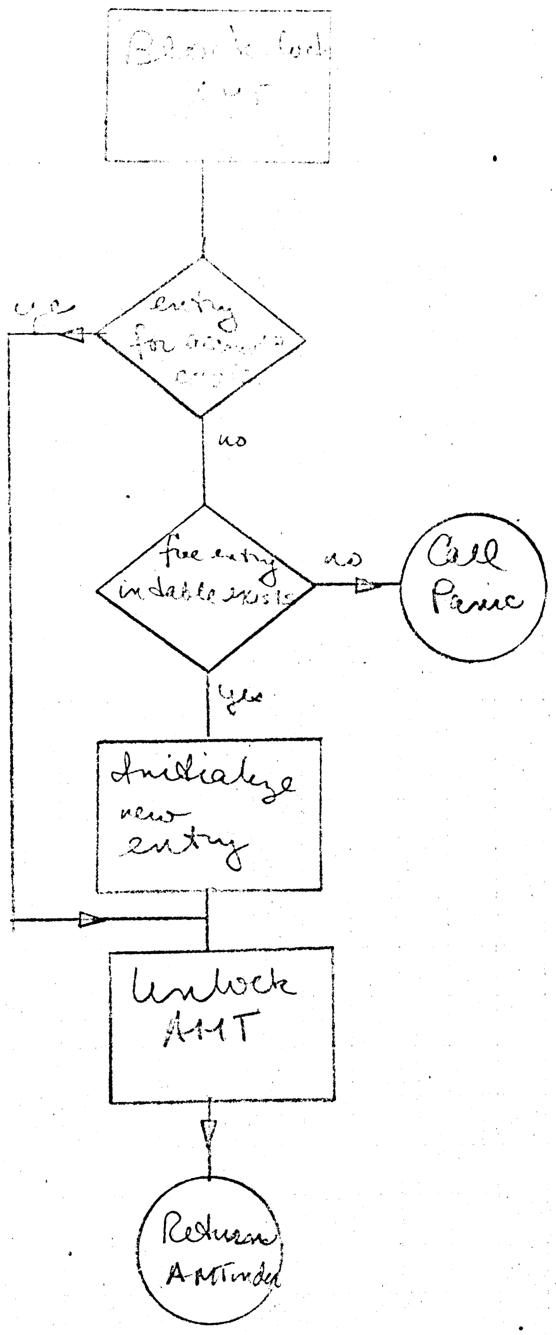
Description of "stop meter"

The second call provided for managing the AMT is:

stop_meter (amtindex, error_code)

Since metering information will always be kept in the AMT, the stop_meter procedure is a very simple one. Stop_meter is called whenever a process or segment becomes inactive. In the initial implementation, stop_meter will
of the AMT entry corresponding to
then merely decrease theuser-count/by one (1) and return the amtinde
to the caller. This is done to keep track of the number of users presently accessing this meter entry corresponding to the amtindex parameter for debugging purposes. In future implementations this call will also be used to free AMT entries which are no longer being used, and hence conserve table space in wired down memory.

start mode


Block lock AMT
yes
entry for access exists
no
free entry in table exists
no
Call Panic
yes
Initialize new entry
Unlock AMT
Return AMT index

START_METER

```
start_meter:      procedure(account, amtindex);
        /* Block-lock AMT */
    dcl (ind, already, locked, event) fixed bin 17,
          event_var bit(36);
    ind = 0
loop: call ulock$lock(addr(amt$amt.ulock), event,
          event_var, already, locked, ind);
    if locked ¬= 1 then do;
        call pwn$wait(event, ind);
          /* process wait and notify */
        go to loop;
        end;
```

(QUESTIONS:

① lock = bit(36) or bit(72)??

② locked ¬= 1 or locked = 1 above??

START_METER

```
start_meter :        procedure(account, amtindex);
        /* Block-lock AMT */
    dcl (ind, already, locked, event) fixed bin 17,
            event_var bit(36);
    ind=0
loop: call clock$lock(addr(amt$amt.clock), event,
            event_var, already, locked, ind);
    if locked 7=1 then do;
        call pwn$wait(event, ind);
            /* process wait and notify */
        go to loop;
        end;
```

(QUESTIONS:

① lock = bit (36) or bit (72) ??

② locked 7=1 or locked = 1 above ??

SM2

```
/* Linear search through table for
   either corresponding account number
   or vacant entry */

dcl i fixed, account bit(36);


search: DO i = 1 to amt$amt.entries;

          if amt$amt.amte(i).account = account
              then go to entry_exists;

          if amt$amt.amte(i).vacant = "0"B
              then go to create_entry;
          /* no match found so create a
             new amtentry */


       end;

       /* if neither a match nor a
          vacant entry found by now
          then table must be full */

       call panic;
```

SMB

```
entry-exists: amtindex = i ;
    unlock : call ilock$unlock(addr(amt$amt.lock),
                               event, event-var);
                return;


  create_entry: amtindex = i ;
        amt$amt.amte(i).account = account ;
           "            .processor = 0 ;
           "            .real_time = 0 ;
                        .nusers = 1 ;
           "            .ss.time = (?) ;
           "            ".trans =0 ;
                           .wds_used = 0 ;
                           .wd-secs = 0 ;
                      .vacant=1 ;

       go to unlock;

  end start_meter;
```