

DRAFT
DRAFT 3/15/77Identification

Active Meter Table Management

T.H. Van Vleck

date _____

Purpose

This section describes the hardcore ring subroutines for managing the Active Meter Table (AMT), and the Account Update Process, which updates resource usage figures from the hardcore ring to the Account Data Segments in the administrative ring.

Calls

Two calls are provided for managing the AMT:

- 1) start_meter (account, amti, err, code)

This entry is called whenever a process or a segment becomes active. Its purpose is to insure that a wired-down "scratchpad" will be available to accumulate usage figures generated when the system cannot tolerate page faults.

The call comes from either the Process Activation module or the Segment Activation module at a time when page faults are permitted.

"Start_meter" uses the AMT hash table to search for an entry for account. If an entry is found, the use count in the entry is increased by one. If no AMT entry exists, one is created and initialized with a use count of 1.

"Start_meter" returns a relative pointer, amti, to the entry in the Active Meter Table. This index will be used by the metering calls to find the AMT

entry in order to record resource-usage figures. The index is stored in the AST entry or process data block.

2) stop-meter (amt¹, err, code)

This entry is called whenever a process or segment becomes inactive, to indicate that the AMT entry may possibly be freed, if no other process or segment is using the entry.

"Stop_meter" decreases the use count in the AMT entry specified by amti by one. If the count is non-zero, the program returns. If the count becomes zero, then the entry can be freed; "stop_meter" copies the entry onto the AMT update list, removes the AMT entry and its hash table entry, and signals a wakeup to the AMT update process so that the copied entry will be updated to the account's Account Data Segment.

The Account Update Process

The Account Update Process is responsible for copying entries from the Account Update List to the Account Data Segments.

The main routine of the process operates in the administrative ring, calling hardcore-ring subroutines when it is necessary to access the Account Update List. The administrative-ring procedure is called by

```
call acct_update_procs,
```

when the accounting subsystem is initialized. It does not return until system shutdown time.

The account Update Process is normally blocked, waiting for a wakeup from subroutine "stop_meter", signifying that an entry has been made in the Account Update List. When such a wakeup is received, it calls into the hardcore ring as follows:

```
call get_acc_list$start (ent, flg);
```


This call locks the Account Update List (a "block-lock") and returns the first entry. If the list is empty, as a result of "start_meter" having deleted all entries, flg is set to non-zero.

For each entry received from the Account Update List, the process attempts to lock the Account Data Segment and insert the information. If successful, it calls

```
call get_acc_list$continue (ent, flg)
```


this call deletes the entry from the Update List and returns the next entry, if any. If the process cannot lock the Account Data Segment, it calls as follows:

```
call get_acc_list$skip (ent, flg);
```

which doesn't delete the entry, but advanced  down the list.

When flg is returned non-zero, the Account Update List is empty, and the process calls

```
call get_acc_list$finish;
```

to unlock the list. It then calls the wait-coordinator  to wait for more wakeups.

Interlocking

In order to prevent confusion in the use of the AMT, both "start_meter" and "stop_meter" must lock the AMT Hash Table before referencing the AMT. This is a "block-lock", which prevents an entry for an account from being added and deleted simultaneously.

In initializing an AMT entry, "start_meter" must fill in the current residence on each secondary storage device. The correct value is either in the Account Data Segment or the Account Update List. "Start_meter" must therefore lock both of these segments, again with block-lock calls, before creating an AMT entry, and must search the update list for an entry for the account. If an entry exists on the Update List, its values are the correct ones, and the Update List entry should be deleted. If no entry is found, the Account Data Segment is up-to-date, and its secondary storage residence figures are to be used in initializing the AMT entry.

"Stop_meter" never references the ADS, and so doesn't need to lock it. It must lock the Update List.

The Account Update Process must lock the Update List and the target Account Data Segment, in order to avoid conflicts with "start_meter" and "stop_meter".

start_meter

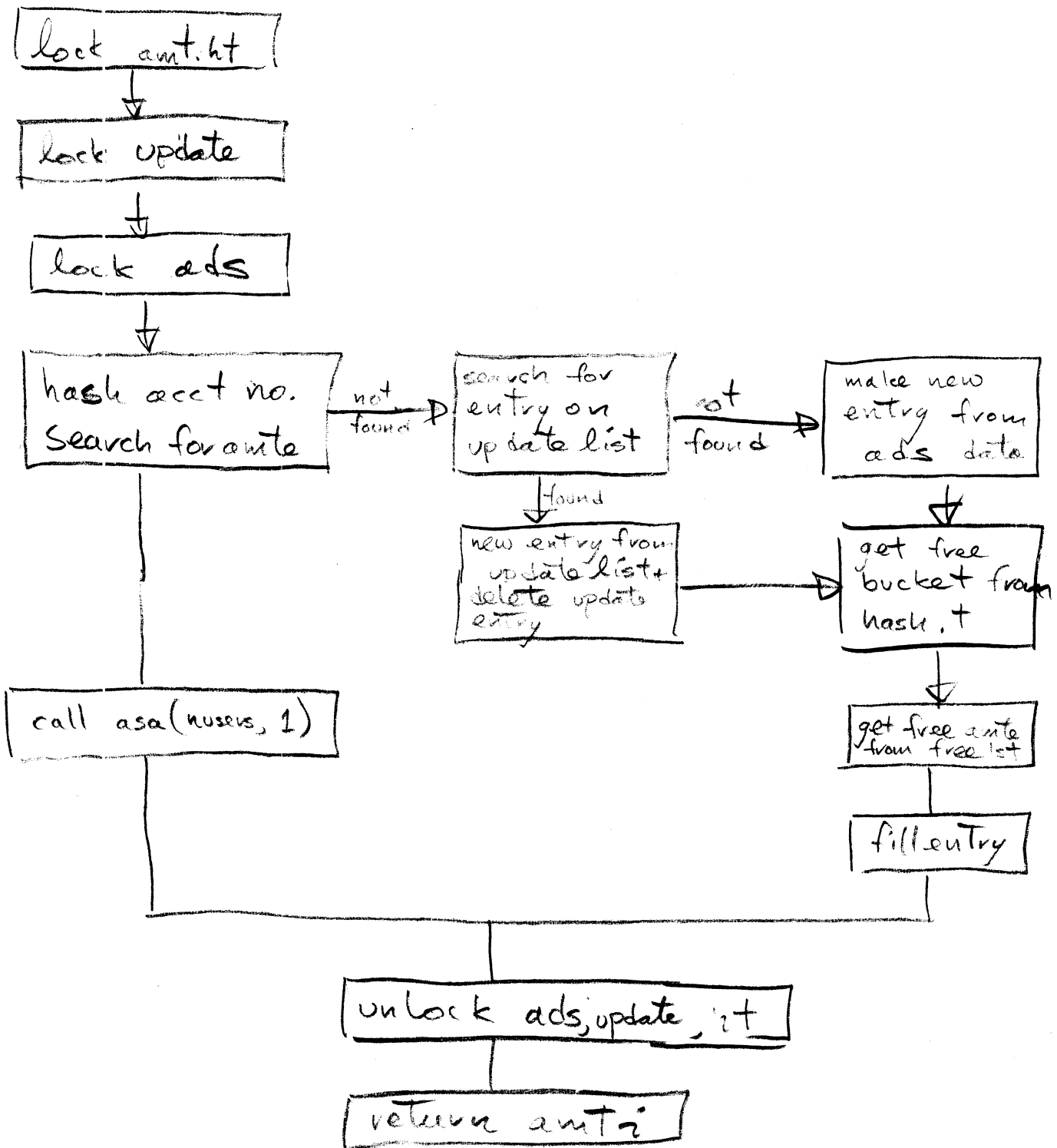


Fig. 1 : Start_meter

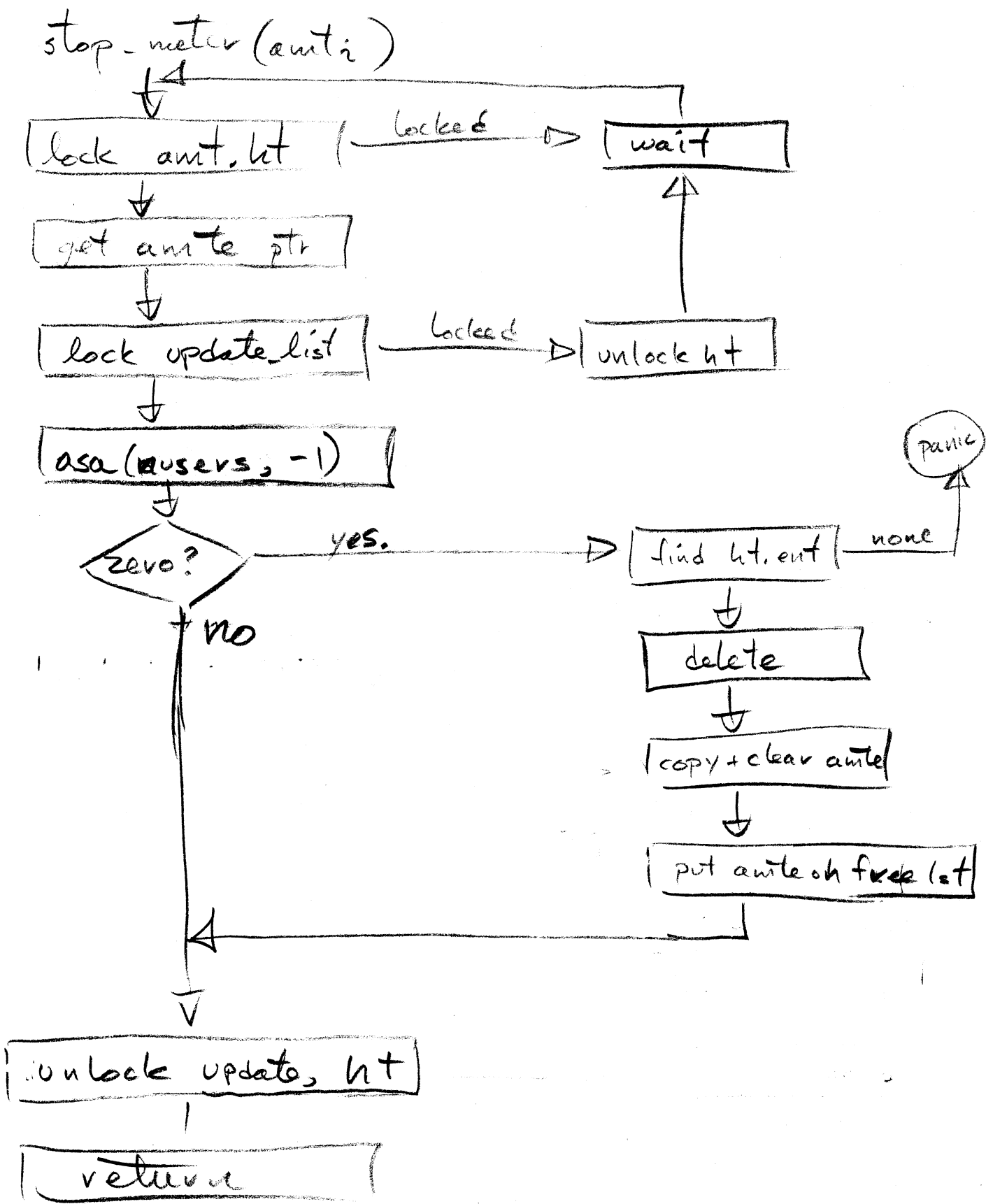


Fig. 2: Stop-meter

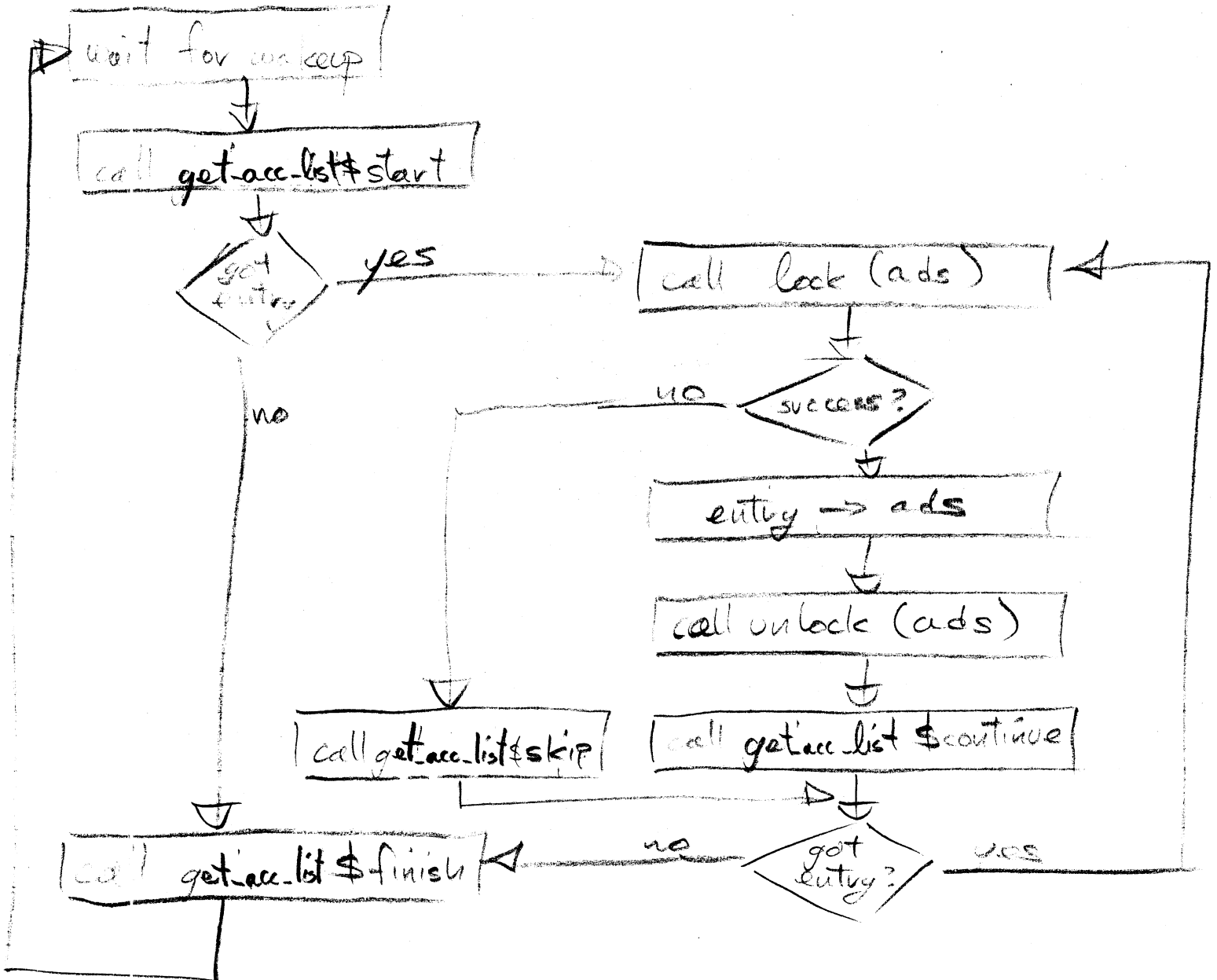


Fig. 3:
The account update process