Honeywell

SYSTEM METERING PROGRAM LOGIC MANUAL

SERIES 60 (LEVEL 68) MULTICS

SUBJECT:

Descriptions of Metering Interfaces for Use by Multics System Programmers.

SPECIAL INSTRUCTIONS:

This Program Logic Manual (PLM) describes certain internal modules constituting the Multics System. It is intended as a reference for only those who are thoroughly familiar with the implementation details of the Multics operating system; interfaces described herein should not be used by application programmers or subsystem writers; such programmers and writers are concerned with the external interfaces only. The external interfaces are described in the Multics Programmers' Manual, Commands and Active Functions (Order No. AG92), Subroutines (Order No. AG93), and Subsystem Writers' Guide (Order No. AK92).

As Multics evolves, Honeywell will add, delete, and modify module descriptions in subsequent PLM updates. Honeywell does not ensure that the internal functions and internal module interfaces will remain compatible with previous versions.

This PLM is one of a set, which when complete, will supersede the System Programmers' Supplement to the Multics Programmers' Manual (Order No. AK96).

DATE:

February 1975

ORDER NUMBER:

AN52, Rev. 0

PREFACE

Multics Program Logic Manuals (PLMs) are intended for use by Multics system maintenance personnel, development personnel, and others who are thoroughly familiar with Multics internal system operation. They are not intended for application programmers or subsystem writers.

The PLMs contain descriptions of modules that serve as internal interfaces and perform special system functions. These documents do not describe external interfaces, which are used by application and system programmers.

Since internal interfaces are added, deleted, and modified as design improvements are introduced, Honeywell does not ensure that the internal functions and internal module interfaces will remain compatible with previous versions. To help maintain accurate PLM documentation, Honeywell publishes a special status bulletin containing a list of the PLMs currently available and identifying updates to existing PLMs. This status bulletin is distributed automatically to all holders of the System Programmers' Supplement to the Multics Programmers' Manual (Order No. AK96) and to others on request. To get on the mailing list for this status bulletin, write to:

Large Systems Sales Support
Multics Project Office
Honeywell Information Systems Inc.
Post Office Box 6000 (MS A-85)
Phoenix, Arizona 85005

 File No.: 2L13

AN52

CONTENTS

SECTION I

INTRODUCTION

This Program Logic Manual describes the techniques and tools used to meter many of the Multics supervisor functions. Section II gives a detailed description of the metering data bases used. Section III briefly describes the general metering techniques used. Section IV contains usage information for the available metering commands and subroutines.

SECTION II

DATA BASES

To be supplied.

SECTION III

METERING DESIGN

This section gives a few recommendations for writing metering commands and subroutines. It also describes some of the conventions used by the standard metering commands.

THE USE OF meter_util_

The subroutine meter_util_, described in Section IV, can be used to extract the metering information from the hardcore data bases SST and TC_DATA. These data bases contain global metering data concerning the time the system has been up; the amount of memory configured; the number of processes (users) currently on the system; as well as the detailed metering data generated by the page control and traffic control programs of the supervisor. The meter_util_ procedure allows the caller to get this data easily and to reset the data being sampled. In particular the meter_util_$time entry prints the time the system has been up (or the time since the last reset call) in a standard format used by most of the metering commands.

THE VARIOUS TYPES OF METERING TIME

There are several different types of metering time that are of interest to users of metering tools. These are:

real time	the actual time period as measured by a normal clock.
virtual time	the actual CPU time spent in a process or in the system minus all CPU time spent in page fault and interrupt processing.

idle time — the amount of time the system was not running a program on behalf of a normal process. This time is partitionable into well-defined subsets. This partitioning is of interest to anyone measuring the efficiency of the system.

processor time — the amount of CPU-seconds the system has accumulated since it was bootloaded. This time is the same as real time for a system that has only one processor, but is guaranteed to be different if the system ever had more than one processor configured.

It is often interesting to print out the percentage of time a given program or programs spends in the system. To do this an appropriate base must be established against which to compare the given quantity. The standard base used by most metering commands is the processor time accumulated by the system. Some commands, however, feel it more appropriate to base their comparisons against nonidle processor time. Either of these methods is acceptable as long as it is clearly stated in some way what the meters represent.

THE RESET MECHANISM

Several metering commands allow the user to begin metering again at a specified time, usually the time of the call to the command with the -reset control argument specified. This type of metering is quite useful and is generally done with the use of internal static copies of the data bases containing the raw metering data. Although the internal static technique has proved to be very useful and easy to program, it does not allow the user to specify an arbitrary time interval during the day for which metering is desired. To do this, a periodic sampling of the metering data must be done. Such sampling is in fact done by the answering service at 15-minute intervals of the SST and TC_DATA segments (although this data is currently unavailable to normal users).

STANDARD CONTROL ARGUMENTS

The following metering control arguments function as described below. If any new metering command is written that performs one of the features controlled by these arguments, it should be designed so that it can use these same control arguments.

-reset, -rs	This control argument is used to indicate that a new metering interval should be defined, but that output of metering data is not necessarily desired.
-report_reset, -rr	This control argument is used to generate a full report and then reset the metering interval.
-brief, -bf	This control argument is used when an abbreviated report is desired. (The default report should print out all of the metering data normally available.)

SECTION IV

COMMANDS AND SUBROUTINES

This section contains command and subroutine descriptions of metering tools on the system.

Name: alarm_clock_meters, acm

The alarm_clock_meters command displays information about the behavior of the simulated alarm clock used within the Multics system.

Usage

alarm_clock_meters -control_arg-

where control_arg can be selected from the following list of control arguments:

-reset, -rs — resets for the invoking process those meters printed by alarm_clock_meters.

-report_reset, -rr — generates a report and then performs the reset operation.

If no control argument is given, a full report is generated.

Notes

The following are brief descriptions of the variables printed out by alarm_clock_meters.

Item	Meaning
No. alarm clock sims	is the total number of times that an alarm clock wakeup was generated.
Simulation lag	is the average value of the simulated clock delays (i.e., the time difference between when the alarm should have gone off and when it was actually processed).
Max. lag	is the largest of the simulated delays that has occurred since the time of bootload. This value is not affected by the -reset option.

alarm_clock_meters

Example

alarm_clock_meters

```
Total metering time     1:36:35

No. alarm clock sims.    2274
Simulation lag             91.676 msecs.
Max. lag                 2305.883 msecs.
Priority boosts             0
Priority elig. lost         0
```

Name: change_tuning_parameters, ctp

The change_tuning_parameters command is used to change several tuning parameters within the system.

Usage

change_tuning_parameters par1 val1 ... parn valn

where:

1. pari is taken from the following list:

   max_eligible (maxe)
   min_eligible (mine)
   tefirst
   telast
   timax
   working_set_addend (wsa)
   working_set_factor (wsf)
   double_write (dblw)
   post_purge (pp)
   quit_priority (qp)

2. vali specifies the new value to be assigned to the corresponding system parameter. The units used for the various parameters are given below:

max_eligible	number of processes
min_eligible	number of processes
tefirst	number of eighths of a second
telast	number of eighths of a second
timax	number of eighths of a second
working_set_addend	number of pages
working_set_factor	fractional multiplier
double_write	0 = no double writes 1 = all nonpd pages get written 2 = all directories get written
post_purge	on/off
quit_priority	fractional multiplier

Note

This command is highly privileged, requiring access to the hphcs_ gate. Before making any change, the user is shown the change and asked if it is correct. The first pair of values represents the old and new values of the parameter, while the second pair of values (parenthesized) represents the octal contents of the word in the data base where that parameter is

change_tuning_parameters

kept. The user must respond yes followed by a newline character for the change to be made. Invalid parameters are rejected. The current values of the parameters can be obtained by using the print_tuning_parameters command.

device_meters

Name: device_meters, dvm

The device_meters command prints out metering information for the page control devices. Depending on the system configuration, information can be printed for the bulk store, DSS190, DSS191, or DSS181.

Usage

device_meters -control_args-

where control_args can be selected from the following list:

-left	prints out information regarding available space on each device.
-io	prints I/O volume statistics for each device.
-latency, -lat	prints device latency (delay) statistics.
-error, -er	prints out error occurrence statistics for the device.
-reset, -rs	resets the metering interval to begin with the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.
-report_reset, -rr	generates a full report and then performs the reset operation.
	If no control argument is given, a full report is generated.

device_meters

Notes

The following are brief descriptions of the variables printed by device_meters.

The following variables are printed if -left is specified.

Item	Meaning
Records	is the number of Multics records on the device.
Unused	is the number of unused records left on the device.

The following variables are printed if -i/o is specified.

Item	Meaning
Reads	is the number of reads performed on each device.
ATB	is the average time between reads (in milliseconds).
Writes	is the number of writes performed on each device.
ATB	is the average time between writes (in milliseconds).
ATB I/O	is the average time between I/O being performed on each device.
% Capacity	is the percentage of total metering time spent performing I/O on each device.

The following variables are printed if -latency is specified.

Item	Meaning
Avg Page Wait	is the average number of milliseconds needed to complete the page transfer.
Avg Chan Time	is the average number of milliseconds spent waiting for a channel to become free so that I/O could be initiated.

device_meters

The following variables are printed if -error is specified.

Item	Meaning
EDAC Corr. Errs	is a count of the times a read was performed and an error occurred, but the EDAC (error-detection-and-correction) hardware was able to correct the error.
Recov. Errors	is a count of the times where an error occurred and the EDAC was unable to correct it, but a subsequent retry resulted in proper transmittal of the data.
Fatal Errors	is a count of the occurrences of nonrecoverable errors.

Example

```
device_meters

Total metering time    1:40:37

                          bulk         d191

Records                   1024       205000
Unused                       5        66866
Reads                   223810       102075
ATB                     26.975       59.145
Writes                  218089        59870
ATB                     27.682      100.839
ATB I/O                 13.662       37.279
% Capacity                   2          125
Avg. Page Wait            .652       38.416
Avg. Chan. Time           .051        0.000
EDAC Corr. Errs              0            0
Recov. Errors                0            0
Fatal Errors                 0            0
```

disk_queue

Name: disk_queue, dq

The disk_queue command prints out the waiting I/O requests queued for a given secondary storage device.

Usage

disk_queue devname

where devname is either -d181 for information about the DSS181 disk; -d190 or -e190 for information about the DSS190 disk; or -d191 or -e191 for information about the DSS191 disk.

Notes

The disk_queue command accepts only one device name at a time; therefore, multiple device names should be enclosed in parentheses. For the device specified, the number of connects on each of its channels is printed. For each waiting request, the following information is output.

Item	Meaning
P	is the priority of the request (1=high; 0=low).
RW	indicates the type of request (R=read; W=write).
D	represents the channel on the device to which the request is directed.
CORE	is the core address from or to which I/O is done.

Name: file_system_meters, fsm

The file_system_meters command is used to meter certain storage system variables and functions.

Usage

file_system_meters -control_args-

where control_args can be selected from the following list:

-ast	prints certain meters about Active Segment Table (AST) usage.
-page, -pg	prints certain meters about paging traffic.
-brief, -bf	causes a shortened report to be generated. Those meters not printed when -brief is specified are indicated by an asterisk (*) in "Notes" below.
-reset, -rs	resets the metering interval so it begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.
-report_reset, -rr	generates a report and then performs the reset operation. The report can be shortened by using the -brief option.
	If no control arguments are given, a full report is generated.

Notes

The following are brief descriptions of the variables printed by file_system_meters.

The following meters, which reflect the activity of the AST lists, are printed if -ast is specified. The two output columns contain the number of occurrences of the specified item and the average time between occurrences.

Item		Meaning
Deactivations		is the number of segment deactivations during the metering interval.
Seg Faults		is the number of segment faults during the metering interval.
Bound Faults		is the number of bound faults during the metering interval.
Setfaults (all)	(*)	is the number of setfaults performed during segment deactivation and during the handling of bound faults.
Setfaults (acc)	(*)	is the number of setfaults performed because the access was changed on a segment.
Updates	(*)	is the number of times branch information was updated from an AST entry.
Steps	(*)	is the number of steps taken through the AST lists searching for a free, usable AST entry.
Skips (ehs)	(*)	is the number of times an entry was skipped in the search for a free, usable entry because the entry-hold-switch was on. The entry-hold-switch is set for certain ring 0 segments that cannot be deactivated.
Skips (inf)	(*)	is the number of times an entry was skipped because there were active segments inferior to the (directory) entry.
Skips (level)	(*)	is the number of times an entry was skipped because another entry was available with fewer pages in core.
Skips (init)	(*)	is the number of times an entry was skipped to give it a grace lap after all of its pages were removed from core.
Skips (ring)	(*)	is the number of times the root directory was skipped. This entry is permanently in an entry-hold-state, therefore providing the number of laps in this list.

file_system_meters

Skips (lock) (*) is the number of times an entry was skipped because its parent could not be locked.

Skips (pc) (*) is the number of times an entry was skipped because page control could not clean all pages out of core.

The following items represent a table indexed by page table size and they show the activity and use of the four AST lists.

Item	Meaning
AST Sizes	indicates the page table sizes being used by the system (constant).
Number	is the number of entries of the specified size.
Need	is the number of entries of the specified size that were needed.
Steps	is the number of steps taken while scanning the specified list.
Ave Steps	is the average number of steps taken in the specified list to find a usable entry in the list.
Grace (sec)	is the average lap time for the specified list.

The following meters are printed if -page is specified. The two output columns contain the number of occurrences of the specified item and the average time between occurrences.

Item	Meaning
Needc	is the number of times a block of core was needed (for page faults, process loadings, etc.).
Ceiling	is the number of times too many write requests were queued at once. Not printed if zero.
Ring 0 faults	is the percentage of page faults that occurred while executing in ring 0.
PDIR faults	is the percentage of page faults that occurred on pages of segments in process directories.

file_system_meters

Level 2 faults		is the percentage of page faults on pages of segments in directories directly off of the root. This is a measure of the activity of the system libraries.
DIR faults		is the percentage of page faults on directory pages.
Laps		is the number of times the used pointer has gone around the used core map list in the search for a usable block of core.
Steps	(*)	is the number of steps taken around the core map list. A step consists of moving the used pointer to the next entry on the list.
Skip wired	(*)	is the number of times a page was skipped while searching the core map list because it was wired down.
Skip used	(*)	is the number of times a page was skipped because it was used in the last lap.
Skip mod	(*)	is the number of times a page was skipped because it had been modified.
Skip os	(*)	is the number of times a page was skipped because I/O was going on for that page (i.e., it was out of service). This meter is not printed if it equals 0.
Pages		is the number of pages available in the system. This is the total core minus the permanently wired down supervisor.
Wired		is the number of pages temporarily wired down. This includes descriptor segments and PDSs for loaded processes.
Average steps		is the average number of steps taken around the core map to find a usable block of core.

file_system_meters

Example

```
file_system_meters

Total metering time    0:40:38

                              #          ATB

Deactivations              2586         .943 sec.
Seg Faults                 6097         .400 sec.
Bound Faults                341        7.152 sec.
Setfaults (all)            5271      462.692 msec.
Setfaults (acc)             624        3.908 sec.
Updates                    4510      540.765 msec.
Steps                      8553      285.146 msec.
Skips (ehs)                 426        5.725 sec.
Skips (inf)                1295        1.883 sec.
Skips (level)               548        4.450 sec.
Skips (init)               2533         .963 sec.
Skips (ring)                  3      812.950 sec.
Skips (lock)                  6      406.475 sec.
Skips (pc)                    0        0.000 sec.

AST Sizes                     4           16          64         256
Number                      432          400         100          10
Need                       3025          523         176          18
Steps                      6542            0         167          17
Ave Steps                   2.2          0.0          .9          .9
Grace (sec)               161.0    975539.9      1460.4      1434.6

                              #          ATB

Needc                     71522      34.099 msec.
Ceiling                      98        .415 min.
Ring 0 faults                      35.747%
PDIR faults                        37.069%
Level 2 faults                     25.430%
DIR faults                         11.374%
Laps                        631       3.865 sec.
Steps                    277083       8.802 msec.
Skip wired                 1925    1266.935 msec.
Skip used                161728      15.080 msec.
Skip mod                  41908      58.195 msec.

408 pages, 25 wired.
Average steps             3.874
```

flush

Name: flush

The flush command is used to cause excessive paging activity in order to check out page control and to time the system. It is not to be used casually as it impairs service to all users of the system.

Usage

flush -control_arg-

where control_arg can be -check (-ck). If -check is specified, each of the 256 pages modified by flush are checked to see if the contents are the same as at the time of a normal flush call. Hence, with the control argument specified, only page reads are caused--no pages are modified causing the necessity of a write. If the data does not match (as is the case if flush with no control argument has never been called), the pages that do not match are printed out on the user's terminal. The format of the output is:

FLUSHi is 116 sb 136

where i is a number from 1 to 4 and specifies in which segment the bad page exists. The last number is the value that should be stored in the page and the first number is the value found there.

When the command is issued with no control argument specified, 256 pages (four segments in the process directory) are modified with a recognized value that can later be checked. With no control argument specified it is, therefore, equivalent to taking (generally at least) 256 page faults.

instr_speed

Name: instr_speed

The instr_speed command attempts to measure the speed of a processor by timing a series of code sequences. The code sequences are chosen to test the more common and important sequences of instruction as well as to get a general idea of the limiting speed that can be expected. It is generally useful to select a given processor (with the use of the set_proc_required command) before running timing tests.

Usage

instr_speed

Note

Code sequences that take longer than expected, probably due to interrupt or fault action, are factored out and not counted.

Name: link_meters, lkm

The link_meters command prints out per-process information regarding use of the Multics linker. The statistics are obtained from the Process Data Segment (PDS) of the process. System-wide linkage information can be obtained with the system_link_meters command.

Usage

link_meters -control_arg-

where control_arg can be selected from the following list:

-reset, -rs — resets the metering interval to begin at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at process initialization time.

-report_reset, -rr — generates a report and then performs the reset operation.

If no control argument is given, a full report is generated.

Notes

The following are brief descriptions of the variables printed by link_meters. The calls to the linker are broken down into four slots:

1. Calls completed in less than 64 milliseconds
2. Calls completed in between 64 and 128 milliseconds
3. Calls completed in between 128 and 192 milliseconds
4. Calls completed in greater than 192 milliseconds

link_meters

Item	Meaning
Calls	The number of calls to the linker that are completed in each time slot and the total number of calls made to the linker by the process.
avg time	The average time (in milliseconds) to completion for a call in each slot and the total average time to completion for a call to the linker made by the process.
avg pf	The average number of page faults for a call in that slot and for a call made by the process.
tot time	The total virtual time (in seconds) taken by calls in that slot and the total virtual time spent in the linker by the process. = calls * avg time
% time	Percentage of total linker time for the process that was taken by calls in each slot.

Example

```
linkage_meters

Linkage Meters:

slot     calls   avg time   avg pf   tot time   % time

<64        286     20.014      1.5      5.724     81.1
64-128      15     77.714      9.5      1.166     16.5
128-192      1    166.862     32.0       .167      2.4
>192         0      0.000      0.0      0.000      0.0
          -----  --------   -----    -------
Total      302     23.367      2.0      7.057
```

meter_fim

Name: meter_fim

The meter_fim command measures the amount of time spent in the Fault Intercept Module (FIM) by causing 100 each of the following faults, zerodivide, mme1, and simfault_000001, and displaying (the time in microseconds) required to handle the faults. The actual faults are generated by a routine called test_faults_, which meter_fim calls. The test_faults_ routine, in turn, calls meter_fim$interpret_results to generate an analysis and printout.

Usage

meter_fim

Note

For each of the three groups of 100 times, the minimum and mean time of the group (in microseconds) is displayed.

meter_gate

Name: meter_gate, mg

This command is used to interpret and print per-system metering information for entries in specified hardcore gates.

Usage

meter_gate gate_name -control_arg- -entryname-

where:

1. gate_name — is the name of the gate segment to be examined; i.e., hcs_, phcs_, etc.

2. control_arg — can be selected from the following list.

   -time, -tm — causes the output to be sorted on the total time spent in each entry.

   -average, -av — causes the output to be sorted on the average time spent in each entry.

   -call, -cl — causes the output to be sorted on total calls to each entry.

   -page, -pg — causes the output to be sorted on the average number of page faults in each entry.

   If control_arg is not specified, the output is not sorted.

3. entryname — is the name of a single entry in the specified gate. Only the information for that entry is printed. If entryname is not specified, information for all entries is printed.

Notes

The output header consists of the time the system was brought up, the current time and the total charge time (=total_cpu_time - idle_time). Also printed is the total number of calls to the gate, the amount of time spent in the entries that were called and the percentage of total charged time that was spent in the entries that were called.

The following is a brief description of the variables printed out by meter_gate.

Item	Meaning
calls	is the total number of times the gate entry point was called.
pcnt	is the percentage of total charge time spent in the called segment.
avg	is the average virtual time in milliseconds spent in the called segment.
pfault	is the average number of page faults incurred during a call to a segment through the specified entry.
entry name	is the name of an entry point to the gate.

Name: meter_gate_

The meter_gate_ subroutine is an entry point into the metering command meter_gate that returns data about specific gate entries to the caller.

Usage

```
declare meter_gate_ entry (char(*), ptr, fixed bin);

call meter_gate_ (gate_name, arrayp, code);
```

where:

1. gate_name — is the name of the gate whose entries are to be metered. (Input)
2. arrayp — is a pointer to an array described below. (Input)
3. code — is a standard Multics status code. (Output)

Notes

The second argument to meter_gate_ is a pointer to an array of entrynames to be metered. This array has the following format:

```
declare 1 arg_array aligned based (arrayp),
          2 num_ents fixed bin,
          2 info (0 refer (arg_array.num_ents)),
            3 name char(32),
            3 calls fixed bin,
            3 page_waits fixed bin,
            3 time fixed bin(71);
```

where:

1. num_ents — is the number of entries in the array info.
2. name — is the entryname.
3. calls — is the number of calls to that entry.
4. page_waits — is the number of page waits by that entry.
5. time — is the time in microseconds used by that entry.

Name: meter_signal

The meter_signal command is a metering tool used to measure the performance of the Multics signalling mechanism. It sets up an environment of condition handlers and stack frames, and then causes a specified number of zerodivide faults to occur. The calendar clock is read before each fault and again as the first operation in the zerodivide condition handler. The difference between these values is recorded and printed on the terminal. The mean and minimum value for all zerodivide faults caused in an invocation is computed.

Usage

meter_signal -control_arg1- value1...-control_argn- valuen

where control_argi is any of the following control arguments. Control arguments can be specified in any order, but each control argument must be followed by a decimal value.

-nfaults specifies how many zerodivide faults to cause. One is the default value.

-nframes specifies the number of stack frames to be established between the frame containing the zerodivide handler and the frame that causes the fault. A value of one implies that the fault occurs in the same frame that established the handler. One is the default value.

-nhandlers specifies the number of handlers for dummy conditions to be established in each stack frame. Handlers are established for the conditions meter_signal_1 through meter_signal_n where n is the value specified. Zero is the default value.

-unclaimed specifies that an unclaimed_signal handler should be established instead of the zerodivide handler. The unclaimed_signal handler is established in the nth frame where n is the value specified. Stack frames are numbered from 1 to p where p is the number in the -nframe option. If omitted, no unclaimed_signal handler is established.

meter_signal

Example

```
meter_signal -nfaults 25 -nframes 5 -nhandlers 2 -unclaimed 3
```

causes 25 faults. Five stack frames are laid down before any faults are caused. Each frame contains handlers for meter_signal_1 and meter_signal_2. An unclaimed_signal handler appears in the third frame.

Name: meter_util_

The meter_util_subroutine procedure contains several entry points used by many of the hardcore metering commands. It creates buffers, fills them with copies of hardcore data, and resets the buffers if required.

Entry: meter_util_$get_buffers

This entry is used to get some usable buffers for the calling procedure. meter_util_ allocates the buffers and returns a unique name of the buffers in the form of a nonzero integer. This number can then be used by the calling program in subsequent calls to meter_util_ for updating information in the buffers.

Usage

```
declare meter_util_$get_buffers entry (fixed bin, ptr, ptr,
     ptr, ptr, fixed bin);

call meter_util_$get_buffers (unique_index, sstp1, sstp2,
     tcdp1, tcdp2, code);
```

where:

1. unique_index — is a unique name of the buffers allocated on this call. (Output)

2. sstp1 — is a pointer to a buffer containing the older copy of data from the System Segment Table (SST) segment. Data is placed in this buffer only by a call to meter_util_$reset. (Output)

3. sstp2 — is a pointer to a buffer containing the most recent copy of data from the SST. Data is placed in this buffer only by a call to meter_util_$fill_buffers. (Output)

4. tcdp1 — is a pointer to a buffer containing the older copy of data from the tc_data segment. Data is placed in this buffer only by a call to meter_util_$reset. (Output)

5. tcdp2 — is a pointer to a buffer containing the most recent copy of data from tc_data. Data is placed in this buffer only by a call to meter_util_$fill_buffers. (Output)

6. code — is a standard Multics status code. (Output)

Note

The buffer sizes are 512 words and 256 words respectively for the SST and tc_data.

Entry: meter_util_$fill_buffers

This entry is used to copy the current contents of the SST and tc_data into the buffers selected. See meter_util_$get_buffers above.

Usage

```
declare meter_util_$fill_buffers entry (fixed bin);

call meter_util_$fill_buffers (unique_index);
```

where unique_index is the unique nonzero integer returned by meter_util_$get_buffers. (Input)

Entry: meter_util_$time

This entry point is called to print the total metering time (time since last reset call) and to return the same to the caller.

Usage

```
declare meter_util_$time entry (fixed bin, float bin);

call meter_util_$time (unique_index, mtime);
```

where:

1. unique_index is as above. (Input)
2. mtime is the total metering time in microseconds. (Output)

Note

The entry meter_util_$fill_buffers must be called at least once prior to this call.

Entry: meter_util_$reset

This entry point is called to reset the metering interval to the time of this call. This is done by copying the current buffers into the prior buffers. See meter_util_$get_buffers.

Usage

```
declare meter_util_$reset entry (fixed bin);

call meter_util_$reset (unique_index);
```

where unique_index is as above. (Input)

Name: page_multilevel_meters, pmlm

The page_multilevel_meters command prints information about the behavior of the page multilevel algorithm of Page Control.

Usage

page_multilevel_meters -control_args-

where control_args can be selected from the following list:

-brief, -bf — produces a shortened report, printing those variables marked below with an asterisk (*).

-reset, -rs — resets the metering interval so that it begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-report_reset, -rr — generates a full report and then performs the reset operation.

If no control arguments are given, a full report is generated.

Notes

The following is a list of the variables printed by page_multilevel_meters with a short description of the meaning of each variable.

Item		Meaning
PD records	(*)	is the number of available records on the paging device.
Pages moved to PD	(*)	is the number of times a page was moved from the disks to the paging device.
Core blocks needed	(*)	is the number of requests by the various parts of Page Control for a block of core.

page_multilevel_meters

New Pages	(*)	is the number of page faults that resulted in the creation of a new page (one that did not exist before).
Page faults from PD	(*)	is the number of page faults that occurred for pages that were found on the paging device (comparable to associative memory matches).
% faults from PD	(*)	is the percentage of faults that came from the paging device. Faults that did not come from the paging device either come from the disks or were new page faults.
Ratio PD to other	(*)	is the ratio of faults that come from the paging device to other faults.
Ratio of pd reads to disk reads		is the ratio of the number of paging device reads requested to disk reads requested.
Disk writes		is the total number of write requests to the disks.
Disk writes (RWS)		is the number of disk write requests queued to perform the write cycle of a read/write sequence (RWS). A RWS is used to move infrequently used and modified pages from the paging device back to its home address on a disk.
Disk writes (GTPD)		is the number of disk write requests queued when the page was forced to disk because the global transparent paging device (GTPD) flag was on for the segment. If this flag is on, no pages of the segment are moved to the paging device.

page_multilevel_meters

Disk writes (first)	is the number of disk write requests queued with the "first" switch on for a page being written from core. The "first" switch, when on, causes the first write for the page (since activation) to go to the disk. Subsequent page faults cause the page to be brought up to the paging device.
Disk writes (forced)	is the number of disk write requests queued because there were currently no free paging device records into which to move the page.
% PD disk writes forced	is the percentage of all normal writes forced to the disk when there are no free paging device records.
CPU time (start)	is the average CPU time in milliseconds to initiate a read/write sequence.
CPU time (done)	is the average CPU time in milliseconds to complete a read/write sequence (clean up after it, etc.).
Steps	is the total number of steps taken around the paging device used list in search of a paging device record that can be freed.
Ave steps	is the average number of steps taken to find a paging device record that can be freed.
Skips (incore)	is the number of times a paging device map entry for a page in core was encountered while searching for a record that can be freed.
RWS performed	is the number of read/write sequences performed.

page_multilevel_meters

PD write aborts — is the number of times a RWS was aborted in the write cycle. A RWS is aborted if a process takes a page fault on the page being moved from the paging device.

Lap time estimate (*) — is an estimate of the lap time for the paging device used list.

Example

```
Total metering time     0:34:03

PD records                1019
Pages moved to PD        11244
Core blocks needed       58430
New pages                 6567
Page faults from PD      27170
% faults from PD          52.4
Ratio PD to other         1.1:1

Ratio of PD reads to
   disk reads             1.2:1
Disk writes               7769
Disk writes (RWS)         7021
Disk writes (GTPD)          64
Disk writes (first)          0
Disk writes (forced)       684
% PD disk writes forced   1.99

Overhead to perform read-write sequences:
CPU time (start)    Ave =  2.4 msec.,   .83 % of system.
CPU time (done)     Ave =  2.1 msec.,   .73 % of system.
Overhead                              1.56 % of system.

Steps                    17209
Ave steps                1.531
Skips (incore)            1359
RWS performed             7021
PD write aborts              5

Lap time estimate        2.017 min.
```

print_gen_info_

Name: print_gen_info_

The print_gen_info_ subroutine is used to print out general information about an object segment. The format of the output is (as might be expected) the same as the print_gen_info command.

Usage

```
declare print_gen_info_ entry (ptr, fixed bin(24),
     char(*), fixed bin(35));

call print_gen_info_ (p, bc, stream, code);
```

where:

1. p — is a pointer to the object segment of interest. (Input)
2. bc — is the bit count of the object segment. (Input)
3. stream — is the name of the I/O switch over which the information is to be output. (Input)
4. code — is a standard Multics status code. (Output)

Entry: print_gen_info_$component

This entry prints the information about a particular component of a bound segment over the I/O switch specified.

Usage

```
declare print_gen_info_$component entry (ptr,
     fixed bin(24), char(*), fixed bin(35), char(*));

call print_gen_info_$component (p, hc, stream, code,  name);
```

1-4. are as in the print_gen_info_ entry.

5. name — is the name of the component within the bound segment for which information is to be printed. (Input)

Name: print_paging_histogram, pph

The print_paging_histogram command can be used to display, in graphical form, a histogram of the interarrival times of system-wide page faults. This information is retrieved from the hardcore data base page_fault_histogram.

The data is displayed on 62 lines, corresponding to interarrival intervals of up to 62 milliseconds. The last line represents all interarrival intervals greater than or equal to 62 milliseconds. Each line gives the number of interarrival intervals observed in its range, and a line of asterisks scaled to the largest line.

Usage

print_paging_histogram -control_arg-

where control_arg can be taken from the following list:

-reset, -rs — resets the metering interval to begin with the time of the last call with -reset or -report_reset specified. If -reset has never been given in a process it is equivalent to having been given at system initialization time.

-report_reset, -rr — generates a full report and then performs the reset operation.

If no control argument is given a full report is generated and a reset is performed (equivalent to a -report_reset).

print_tuning_parameters

Name: print_tuning_parameters, ptp

The print_tuning_parameters command is used to print the current values of various tuning parameters within the system. The values of these tuning parameters can be changed by using the change_tuning_parameters command.

Usage

print_tuning_parameters

Notes

The values of tefirst, telast, and timax are output in terms of seconds, even though their values are changed in change_tuning_parameters by entering units that represent eighths of a second.

Example

```
min_eligible              2
max_eligible              6
tefirst                .500 sec.
telast                1.000 sec.
timax                 8.000 sec.
working_set_addend        0
working_set_factor        1
post_purge              off
double_writes             0
quit_priority         0.000
```

spg_ring_0_info_

Name: spg_ring_0_info_

The spg_ring_0_info_ subroutine returns information about the virtual CPU time spent in the three main gates into ring zero. The three gates are hcs_, phcs_, and hphcs_.

Usage

```
declare spg_ring_0_info_ entry (fixed bin(71));

call spg_ring_0_info_ (time_rz);
```

where time_rz is the cumulative time, in microseconds, spent in ring zero. (Output)

Name: spg_util_

The spg_util_ subroutine collects metering information from the Multics supervisor and subtracts it from the previous sample taken. It is normally called by system_performance_graph.

Usage

```
declare spg_util_ entry (float, float, float, float, float,
     float, float, float, float, (110),
     fixed bin, fixed bin);

call spg_util_ (pzi, pnmpi, pmpi, pint, ptc, ppf, psf,
     puse_rz, px, string, length, chsw);
```

where:

1. pzi — is the percentage of zero idle time. (Output)
2. pnmpi — is the percentage of nonmultiprogramming idle time. (Output)
3. pmpi — is the percentage of multiprogramming idle time. (Output)
4. pint — is the percentage of time in interrupts. (Output)
5. ptc — is the percentage of time in the traffic controller. (Output)
6. ppf — is the percentage of time in page control. (Output)
7. psf — is the percentage of time in segment control. (Output)
8. puse_rz — is the percentage of time executing nonsupervisor code spent in ring zero.
9. px — is no longer used. A value of 0.0 is returned. (Output)
10. string — if the variable chsw is nonzero, string contains upon output, a character string that describes a new configuration or a new setting of the scheduler tuning parameters. (Output)

11. length is the length of the character string "string". (Output)

12. chsw is a switch that, if zero, indicates normal output; if nonzero, it indicates that string and length are valid and should be output. (Output)

Entry: spg_util_$reset

The effect of this call is to reset the internal initialization switch of the subroutine.

Usage

```
declare spg_util_$reset entry;

call spg_util_$reset;
```

There are no arguments.

Name: system_link_meters, slm

The system_link_meters command prints out system-wide statistics regarding usage of the Multics linker. Information is obtained from the data bases: active_hardcore_data and tc_data. Per-process linkage information is available from the link_meters command.

Usage

system_link_meters -control_arg-

where control_arg can be selected from the following list:

-reset, -rs — resets the metering interval to begin at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-report_reset, -rr — generates a full report and then performs the reset operation.

If no control arguments are given, a full report is generated.

Notes

The following are brief descriptions of the variables printed out by system_link_meters. The calls to the linker are broken down into four slots:

1. Calls completed in less than 64 milliseconds
2. Calls completed in between 64 and 128 milliseconds
3. Calls completed in between 128 and 192 milliseconds
4. Calls completed in greater than 192 milliseconds

system_link_meters

Statistics are given for overall use of the linker, and are also broken down by task. The three major tasks of the linker are:

1. searching the definition section of the object segment for the symbolic name of the referenced segment
2. searching for the segment using the standard search rules
3. getting the linkage to the referenced segment

Item	Meaning
Total Metering time	is the amount of time over which the metering has occurred, expressed as hh:mm:ss.
CPU Metering time	is the amount of time for which the processor was busy. =(total processor time)-(idle time).
Total time in linker	is the total amount of CPU time spent in the linker, expressed as hh:mm:ss.
Average time per link	is the average time to completion (in milliseconds) for a call to the linker.
Percentage of real time in linker	is the percentage of total metering time that was spent in the linker.
Percentage of CPU	is the percentage of virtual CPU metering time that was spent in the linker.
Time Slot	are the time slots referred to at the beginning of this section.
Calls	is the number of calls that were completed in each time slot.
Total time in slot	is the total amount of virtual CPU time taken by calls in each time slot.

system_link_meters

Percent total time — is the percentage of the virtual CPU time spent in the linker that was taken by calls in each slot.

Percent total calls — is the percentage of calls to the linker that fell into each time slot.

Average time — is The average time (in milliseconds) to complete a call to the linker that ended up in each time slot.

Average page faults — is the average number of page faults for a call in each slot.

The following statistics are given for each of the three major tasks of the Multics linker; segment search, get linkage, and definition search.

Average time — is the average time (in milliseconds), for a call in each slot, spent on that particular function of the linker.

Average page faults — is the average number of page faults for a call in each slot, which occurred during that particular task of the linker.

Percent time in slot — is the percentage of the total time spent in the slot that was taken up by that particular task. These percentages do not add up to 100% because some time used by the linker does not fit into any of the three task categories.

Name: system_performance_graph, spg

The system_performance_graph command is used to generate a system of graphs that meter information concerning system performance and operation. The output can be directed to a file or to the controlling terminal. Metering information is periodically, incrementally presented in an output line. The initial line contains the cumulative values since system initialization. Whenever there is a change in system configuration or any of several parameters affecting system performance, an additional line noting the change is issued before the sample line. In this way, a system of graphs is developed where various metered quantities are plotted against time. Because the sampling is implemented by means of an event call channel, it is possible to use the terminal in a restricted way for other purposes while metering is in progress. All output is produced on the I/O stream spg_output_.

Usage

spg sample_time -control_args-

where:

1. sample_time — is a decimal integer giving the time in minutes, desired between meter display lines.

2. control_args — can be chosen from the following list:

   -halt, -ht — terminates plotting.

   -output_file, -of — directs output to a file called spg_output.

Notes

Description of the output pattern:

1. There is an initial line giving the date and time that metering sampling began.

2. A line is given describing configuration and scheduling parameter settings.

3. The current state of the meters since system initialization is on the next line where the sample time is replaced by the system initialization line.

4. Each subsequent meter display line gives the incremental status of the meters since the previous line. In addition, whenever the configuration or scheduling parameter settings change, a notification line is interspersed.

Description of the meter display line:

Each line contains, in the left margin, the time that the sample was taken. Each sample is scheduled to be taken at an exact minute so that the amount the time given that exceeds the minute represents a sample of the response time. Strictly, the discrepancy is the response time of a trivial request only if the metering computation is less than the quantum and if the command argument sample_time is greater than one minute so that interactive scheduling occurs.

The remainder of the meter display line consists of a sequence of superimpositions over a grid 100 units wide. The grid is created by vertical bars every 10 spaces with periods at the intervening midpoints between the bars. Over this grid, various metering quantities are superimposed in the following order. When the superimposition is complete, the resultant line containing only the last characters superimposed is printed.

1. Time Usage Percentages

Symbol	Location	Meaning
blank	right of y to right margin	user processing not in ring 0. (position of y is an estimate)
blank	right of s to left of y	user processing in ring 0.
s		time spent handling segment or bounds faults.
p		time spent handling page faults.
t		time spent in the traffic controller.
i		interrupt processing

system_performance_graph

blank	right of *'s to i	multiprogramming idle.
*		nonmultiprogramming idle.
blank	left margin to left of *'s	zero idle.

2. Other Values

The current average is determined from samples taken at one second intervals weighted backwards in time by increasing powers of 63/64. The effect is to average over roughly the last minute.

Symbol	Relative to	Meaning
q	left margin	current average of the ready list length.
e	left margin	current average of the number of eligible processes.
r	left margin	current average of the response time in seconds, for trivial requests.
Q	left margin	average over a sample of quits/minute.
S	left margin	average over a sample of schedulings/10 seconds.
D	right margin	average over a sample of DSS190 read and write traffic in units of pages/(.5 seconds).
P	right margin	average over a sample of all read and write traffic (bulk store, DSS181 and DSS190) in units of pages/(.5 seconds).

-	left margin	number of load units at the time of the sample.
+	left margin	number of users at the time of the sample.

Entry: system_performance_graph$line

This entry is called each time a wakeup is received to print a line of metering.

Usage

```
declare system_performance_graph$line entry;

call system_performance_graph$line;
```

There are no arguments.

Name: total_time_meters, ttm

The total_time_meters command prints out the CPU time percentage and average CPU time spent doing various tasks.

Usage

total_time_meters -control_arg-

where control_arg can be selected from the following list:

-reset, -rs — resets for the invoking process, and the metering interval so that it begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-report_reset, -rr — generates a full report and then performs the reset operation.

If no control argument is given, a full report is generated.

Notes

The following are brief descriptions of each of the variables printed out by total_time_meters. Average CPU times are given in microseconds.

Item	Meaning
Page Faults	is the percentage of system CPU time spent handling page faults and the average time spent per page fault.
Getwork	is the percentage of system CPU time spent in the getwork function and the average time spent per pass through getwork.
Seg Faults	is the percentage of system CPU time spent handling segment faults and the average time spent per segment fault. These values are exclusive of the time spent handling page faults that occurred during the segment fault handling.
Bound Faults	is the percentage of system CPU time spent handling bound faults and the average time spent per bound fault. This value is

total_time_meters

exclusive of time spent handling page faults that occurred during bound fault processing.

Interrupts — is the percentage of system CPU time spent handling interrupts and the average time spent per interrupt.

MP Idle — is the multiprogramming idle--the percentage of time all eligible processes were waiting, but the maximum number of eligible processes had already been reached.

Loading Idle — is the percentage of time all but the last eligible processes were waiting, these processes were being loaded, and the maximum number of eligible processes had been reached.

NMP — is the nonmultiprogramming idle--the percentage of time all eligible processes were waiting and there were no processes contending for eligibility.

Zero Idle — is the percentage of time that no process was waiting to be run.

Other — is the percentage of system CPU time and the percentage of nonidle CPU time spent executing user processes or other overhead functions.

Example

```
Total metering time    1:54:32

                          %        AVE

Page Faults           15.31    5182.804
Getwork                1.23     642.514
Seg Faults              .88    8719.190
Bound Faults            .22   23444.484
Interrupts             5.04    3158.481
MP Idle                 .55
Loading idle            .05
NMP Idle                .08
Zero idle               .03
Other                 76.61       77.16 % of non-idle
```

Name: traffic_control_meters, tcm

The traffic_control_meters command prints out the values of various traffic control meters.

Usage

traffic_control_meters -control_args-

where control_args can be chosen from the following list:

-gen	prints out general traffic control information and parameters.
-counters, -ct	prints out the number and frequency of certain paths through the traffic controller.
-idle	prints out the time spent in the various idle states.
-queue, -qu	prints out certain resource usage as a function of depth in the eligible queue.
-reset, -rs	resets the metering interval to begin at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.
-report_reset, -rr	generates a full report and then performs the reset operation.
	If no control arguments are given, a full report is generated.

Notes

The following are brief descriptions of the variables printed out by traffic_control_meters.

The following meters reflect activity of the traffic controller, and some constants used therein. They are printed if -gen is specified.

Item	Meaning
Ave queue length	is the average number of processes in the eligible and priority queues. This is the average number of ready, waiting, or running processes.
Ave eligible	is a recent average of the number of eligible processes.
Working set factor	is the system constant used to modify the working set estimate.
Working set addend	is the system constant used to modify the working set estimate.
Te first (seconds)	is a scheduling parameter. The amount of CPU time for which a process is guaranteed to remain eligible (if it needs to) the first time it runs after an interaction.
Te last (seconds)	is a scheduling parameter. The amount of CPU time for which a process is guaranteed to remain eligible when it is in the last scheduling queue.
Ti max (seconds)	is a scheduling parameter. The default amount of CPU time a process remains in the last scheduling queue before being rescheduled (at the end of that queue).

The following meters subdivide the system idle time. The values displayed are the CPU time lost and the percentage of CPU time lost. They are printed if the -idle option is specified.

Item	Meaning
Total idle	is the total system idle time; the sum of the following four meters.

traffic_control_meters

Multi prog idle	is the amount of time consumed when all eligible processes were waiting, the maximum number of processes allowed were eligible, and further processes await eligibility.
Loading idle	is the amount of time consumed when all but a few eligible processes were waiting, these last eligible processes were being loaded, and the maximum number of eligible processes had been reached.
Non multi prog idle	is the amount of time consumed when all eligible processes were waiting and there were no processes contending for eligibility.
Zero idle	is the amount of time that the CPU had no processes available to run.

The following meters pertain to the number and frequency of certain paths through the traffic controller. They are printed if -ct is specified.

Item	Meaning
Interactions	is a count of, and the average time between terminal interactions.
Loadings	is a count of the average time between, and number per interaction of process loadings.
Blocks	is a count of, and the average time between calls to "block" to block some process.
Wakeups	is a count of, and the average time between wakeup signals being sent.
Waits	is the number, average time between, and the number per interaction of calls to force some process to a wait state.
Notifies	is the number of, and the average time between, notify calls (i.e., returning a waiting process to the ready state).

Schedulings	is the number of, and average time between trips through the scheduler/rescheduler function that caused priorities to be changed.
Pre-empt	is the number of average time between, and number per interaction of process preemptions and timer runout faults.
Priority boosts	is the number of times the alarm clock went off indicating a priority scheduling process on the ready list should be granted high priority; i.e., have its ti set to 0. The process is then resorted into the ready list with its new, higher priority.
Priority elig. lost	is the number of times the alarm clock went off indicating a priority process that had been running lost its eligibility because it had used up its eligible time; i.e. its te was larger than timax. The process reenters the traffic controller to be rescheduled.

The following meters pertain to the eligible queue. They are printed if -qu is specified.

Item	Meaning
Depth	is the priority level of the process within the eligible queue.
%PF	is the percentage of page faults that occurred from processes at this priority level.
TBPF	is the average time between page faults at this priority level.
%GTW	is the percentage of getwork calls being made when a member of this priority relinquishes control.

traffic_control_meters

TBS — is the average time between getwork calls at this priority level.

%CPU — is the percentage of CPU time consumed by members of this priority.

Example

```
Total metering time    2:06:07


Ave queue length     24.84
Ave eligible          4.26
Working-set factor     .50
Working-set addend       0
Te first (seconds)       0
Te last  (seconds)       1
Ti max (seconds)         8

    IDLE TYPE              TIME        %

Total idle                0:01:44      .70
Multi-prog idle           0:01:21      .55
Loading idle              0:00:07      .05
Non-multi-prog idle       0:00:10      .07
Zero idle                 0:00:04      .03

  COUNTER        TOTAL          ATB          #/INT

Interactions       806       9.389 sec
Loadings         14576         .519 sec     18.084
Blocks            3717       2.036 sec
Wakeups           3992       1.896 sec
Waits           129462      58.451 msec   160.623
Notifies        946969       7.991 msec
Schedulings      17418         .434 sec     21.610
Pre-empts       147781      51.205 msec   183.351

 DEPTH    %PF    TBPF    %GTW    TBS    %CPU

   1      38.5   52.7    22.9    40.5   17.7
   2      31.7  177.8    21.0   122.7   49.2
   3      17.1  145.5    27.9    40.8   21.7
   4       7.8  105.4    16.1    23.5    7.2
   5       3.4  100.3     8.2    19.2    3.0
   6       1.4  101.4     3.9    17.0    1.2
   7       0.0    0.0     0.0     0.0    0.0
```

Name: traffic_control_queue, tcq

The traffic_control_queue command prints out the state of the traffic control queue at the time of the call.

Usage

traffic_control_queue

Notes

The following items are printed out by traffic_control_queue.

Item	Meaning
avq	is the average number of processes in the eligible and priority queues. This is the average number of ready, waiting, or running processes.
elapsed time	is The time since traffic_control_queue was last called. This equals 0 if this is the first time the program was called for the given process.
active last 15 sec.	is the number of processes that changed state during the last 15 seconds.

The following items are printed out for each user presently in the ready queue.

1. flags are one bit indicators in the Active Process Table (APT) entry for the user. The following flags are printed:

   W Interprocess Communication (IPC) wakeup pending
   S stop pending
   P process being preempted
   N process is noninterruptable
   L process is loaded
   R process is running
   I process has interaction switch on

2. tu is the total CPU time the process has used (in seconds).

traffic_control_queue

3. dtu — is the incremental CPU time (in seconds) the process has used since tcq was last called.

4. te — is the value (in milliseconds) of te of the process.

5. ts — is the value (in milliseconds) of ts of the process.

6. ti — is the value (in milliseconds) of ti of the process.

7. tssc — is the real time (in seconds) since the state change of the process.

8. event — is the event for which the process is waiting. If this value is 0, the process is not waiting.

9. d — if the process is waiting for a page, this is the device ID of the device containing the page.

10. ws — is the modified value of the working set estimate being used for the process.

11. process — is the name of the user who owns the process.

Example

```
avg = 18, elapsed time = 0 sec, 15 active last 15 sec.
flags       tu  dtu   te    ts    ti     tssc event d  ws process
NLEQI      151  151   70    99     0   -.003     0 0   0 IO
NLERI       90   90  658     0  1007    .117     0 0   0 Filichia
WNLEI      444  444  289  8009 16000    .008 46164 2   0 LTD
LEQI       379  379   39     0     0    .001 62145 2   0 Jones
WNLEI      452  453    8  1003 16000   -.005 66557 2   0 LTD
WNI        463  463    0  9104 16000    .104     0 0   0 LTD
WNI        451  452    0  1000 16000   5.385     0 0   0 LTD
WNI        443  444    0  9018 16000   8.072     0 0   0 LTD
WNI        450  450    1     0 16000 129.644     0 0   0 LTD
```

Name: tty_lines, tln

The tty_lines command prints certain information from the answer_table segment about each channel that is attached to Multics.

Usage

tty_lines -control_arg-

where control_arg is of the form xxyyyy. Usage of this option allows one to obtain selected subsets of the information in the answer table. If no control argument is specified, information regarding all lines is output. The following table describes the permitted values of xx and the interpretation of the yyyy part.

Value of "xx"	Meaning of "yyyy"
tt	Teletype channel name (letter "y" followed by 2 or 3 digits; i.e. tty102).
id	Teletype ID code; this value is set for each teletype. The default value is "none".
ct	Channels with experiment-count ≥ yyyy since channel was last initialized.
dl	Channels with dialup-count ≥ yyyy since answer table was last truncated.
st	Channels in state yyyy (possible values 1-6).
wp	Channels at wait-point yyyy (1-10).
ac	Channels with activity yyyy (0-6).
ty	Channels last reached by device of type yyyy. (yyyy = 1 to 11) 1 = 1050 5 = ARDS 9 = MDS2400 2 = 2741 6 = IBM2741 10 = PDP8 3 = TT37 7 = TT33/35 11 = ASCII 4 = TN300 8 = TT38
sl	Information for slot yyyy (1-9999). Blank or 0 means entire table.

c= Channels for which experiment-count equals yyyy.

d= Channels for which disk-count equals yyyy.

Notice that a null part of the option is equivalent to a value of 0 in the selection mechanism, except for the tt and id options, where yyyy is alphanumeric.

Notes

The following is a description of the information printed by tty_lines.

Item	Meaning
Attached lines =	is the number of tty channels currently known to Multics. A one line entry is printed for each channel. Lines consisting of a parenthesized numerical value indicate the number of entries skipped over either because of selection criteria or because the lines have been disabled by the answering service.
size =	is the maximum number of tty channels allowed in the answer table.
at	is the date and time of print out.
Name	is the name of the tty channel.
Type	is the type of the last device to dialup this channel. Possible device types are 1050, 2741, TTY37, TN300, ARDS, IBM2741, TTY33, TTY38, MDS2400, PDP8, ASCII. The characters (NU) in this position indicate that the channel has not been successfully dialed up since the last time the answering service was initialized.
no.	is the number of dialups received since the answer_table was last truncated.

S — is the channel state returned the last time the answering service called the tty_dim for this channel. The possible state values are:

0: not a valid state
1: channel is hungup
2: dialups will be accepted
3: (no longer used)
4: (no longer used)
5: channel in use

WP — is the location in the dialup procedure where processing of this tty channel is currently blocked. The possible values are:

1: Waiting for a dialup or doing experiment
3: Waiting for first Multics header line to be transmitted
4: Waiting for login line to be received
5: Waiting for "password:" to be transmitted
6: Waiting for user's password to be received
7: Waiting for result of login attempt message to be transmitted
8: User's process is running. Waiting for bump, detach, new_proc, logout, etc., to be signalled
9: Waiting for the message following the event listed in 8 (above) to be transmitted
10: Waiting for hangup to occur

A — is the activity value for the channel. The possible values are:

1: Channel is hungup
2: Channel is being listened to
3: Channel is dialed up
4: User is logged in
5: User has process

User — is the user and project name if activity value is 4 or 5; followed by terminal ID.

tty_lines

The tty_lines command can be used for diagnosing problems with the answering service, dataphones, and channels. In particular, the following combinations of channel state, wait-point, and user state are worth noting.

S	WP	A	
2	1	2	Channel waiting for dialup
5	4	3	User going through
5	6	3	login ritual
5	8	5	User logged in and has a process
1	1	0	Channel has been disconnected (indicated as a skip in a normal printout)
x	10	x	A hangup is pending on the channel; should not be observed for more than 10-15 seconds

Name: tty_meters, ttym

The tty_meters command prints out metering information for the ttydim.

Usage

tty_meters -control_args-

where control_args are chosen from the following:

-reset, -rs — resets the metering interval so that it begins at the last call with -reset specified. If -reset has never been given in a process, it is equivalent to having been specified at system initialization time.

-long, -lg — prints the number of buffers containing type ahead data before printing the ttydim meters.

Notes

The following are brief descriptions of the variables printed by tty_meters.

Item	Meaning
min output msg size to cause block	is the minimum output message size for which the sending process had to go blocked due to lack of output buffers.
ave output msg size to cause block	is the average output message size for which the sending process had to go blocked due to lack of output buffers.
% status queued	is the percentage of status words that could not be processed immediately.
output buffer eff.	is the average percentage of an output buffer filled with characters.
ave interrupt time	is the average time in milliseconds spent in handling an interrupt by the ttydim.

tty_meters

max interrupt time	is the maximum time in milliseconds spent in handling an interrupt by the ttydim.
ave input message	is the average size in characters of an input message.
ATB reads	is the average time between reads.
ave output message	is the average size of an output message.
ave r-a message	is the average number of characters found in a read-ahead message.
complete input messages in buffer	is the current and average number of read-ahead messages in the system.
chars input in read-ahead	is the percentage of all input characters typed in read-ahead mode.
consoles reading	is the current number of consoles in read mode and the current number blocked on read.
consoles writing	is the current number of consoles in write mode and the current number blocked on write.
consoles in read-ahead	is the current number of consoles doing read-aheads.
chars output	is the total number of characters output.
chars input	is the total number of characters input.
ATB output blocks	is the average time between going blocked for output.
ATB status	is the average time between the arrivals of status words.
ATB quit	is the average time between quits.
ATB dialups	is the average time between dialups.

ATB cycle	is the average time between input/output cycling.
ATB writes	is the average time between writes.
ATB read-ahead	is the average time between read-aheads.
output conversion factor	is the ratio of final output characters to original output message.
input conversion factor	is the ratio of final input characters to original input message.
cur # dialed	is the current number of each terminal type dialed up.
ave # dialed	is the average number of each terminal type dialed up.
input rate	is the average input rate from each terminal type.
output rate	is the average output rate for each terminal type.

INDEX

HONEYWELL INFORMATION SYSTEMS

Publications Remarks Form*

TITLE: SERIES 60 (LEVEL 68) MULTICS SYSTEM METERING PROGRAM LOGIC MANUAL

ORDER No.: AN52, REV. 0

DATED: FEBRUARY 1975

ERRORS IN PUBLICATION:

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

(Please Print)

FROM: NAME _____ DATE: _____

COMPANY _____

TITLE _____

_____

_____

ALONG LINE

*Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here. ☐

CUT ALONG

FOLD ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WELLESLEY HILLS,
MASS. 02181

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

POSTAGE WILL BE PAID BY:

HONEYWELL INFORMATION SYSTEMS
60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATTN: PUBLICATIONS, MS 050

FOLD ALONG LINE

Honeywell