HONEYWELL

MULTICS SYSTEM METERING

SOFTWARE

MULTICS SYSTEM METERING
ADDENDUM B

SUBJECT

Additions and Changes to the Description of Metering Interfaces

SPECIAL INSTRUCTIONS

This is the second addendum to AN52, Revision 1, dated February 1979.

Insert the attached pages into the manual according to the collating instructions on the back of this cover.

Throughout the manual, change bars in the margins indicate technical additions and changes; asterisks denote deletions. These changes will be incorporated into the next revision of this manual.

Note:

Insert this cover after the manual cover to indicate the updating of the document with Addendum B.

SOFTWARE SUPPORTED

Multics Software Release 10.0

ORDER NUMBER

AN52-01B

November 1982

35823
5C1082
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
iii, iv	iii, iv
4-5, 4-6	4-5, 4-6
4-7, blank	4-7, 4-7.1
4-7.1, 4-8	4-7.2, 4-8
4-47.1, 4-47.2	4-47.1, 4-47.2
4-65 through 4-68	4-65 through 4-68
4-73, blank	4-73, blank

File No.: 1L13, 1U13

11/82

AN52B

LEVEL 68

MULTICS SYSTEM METERING ADDENDUM A

SUBJECT

Additions and Changes to the Description of Metering Interfaces

SPECIAL INSTRUCTIONS

This is the first addendum to AN52, Revision 1, dated February 1979.

Insert the attached pages into the manual according to the collating instructions on the back of this cover.

Throughout the manual, change bars in the margins indicate technical additions and changes; asterisks denote deletions. These changes will be incorporated into the next revision of this manual.

Note:

Insert this cover after the manual cover to indicate the updating of the document with Addendum A.

SOFTWARE SUPPORTED

Multics Software Release 9.0

ORDER NUMBER

AN52-01A

July 1981

32237
5C881
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
title page, preface	title page, preface
iii, iv	iii, iv
4-1 through 4-10	4-1, blank 4-1.1, 4-2 4-3 through 4-7, blank 4-7.1, 4-8 4-9, 4-10
4-17, 4-18	4-17, 4-18
4-21 through 4-28	4-21 through 4-26 4-26.1 through 4-26.4 4-27, 4-28
4-31, 4-32	4-31, 4-32 4-32.1, 4-32.2
4-37, 4-38	4-37, 4-38
4-41 through 4-58	4-41, 4-42 4-42.1, blank 4-43 through 4-46 4-47, blank 4-47.1, through 4-47.4 4-47.5, 4-48 4-49 through 4-58.1, blank
4-65 through 4-68	4-65 through 4-68
4-71 through 4-73, blank	4-71 through 4-73, blank
5-3 through 5-10	5-3 through 5-5, blank 5-5.1, 5-5.2 5-5.3, 5-6 5-7 through 5-10
i-1, i-2	i-1, i-2

File No.: 1L13

7/81

AN52A

MULTICS SYSTEM METERING

SUBJECT

Descriptions of Metering Interfaces for Use by Multics System Programmers

SPECIAL INSTRUCTIONS

This manual supersedes AN52, Rev. 0, dated February 1975. Since new metering tools of more general interest have been added, the manual is no longer restricted.

The manual has been extensively revised; change bars in the margins indicate technical changes and additions, and asterisks in the margins indicate deletions.

Section 2 is new, and does not contain change bars. The print_paging_histogram command is obsolete and has been removed.

SOFTWARE SUPPORTED

Multics Software Release 7.0

ORDER NUMBER

AN52, Rev. 1

February 1979

Honeywell

PREFACE

This manual describes certain internal modules constituting the Multics System. It is intended as a reference for 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 (MPM). For convenience, references to these manuals are as follows:

Document	Referred to in Text as:
Commands and Active Functions (Order No. AG92)	MPM Commands
Subroutines (Order No. AG93)	MPM Subroutines
Subsystem Writers' Guide (Order No. AK92)	MPM Subsystem Writers' Guide
Reference Guide (Order No. AG91)	MPM Reference Guide
Peripheral Input/Output (Order No. AX49)	MPM Peripheral I/O
Communications Input/Output (Order No. CC92)	MPM Communications I/O

The reader using the metering commands and tools which comprise most of this manual should be thoroughly familiar with the information in the MPM. For several of the commands, the user must have privileged access to the system for comprehensive use.

Changes in AN52, Revision 1

The following commands and subroutines are new to this manual and do not contain change bars:

command_usage_count	meter_rcp
define_work_classes	post_purge_meters
disk_meters	ring_zero_peek_
display_bulk_err	set_work_class
interrupt_meters	tune_work_class
iobm_meters	vtoc_buffer_meters
list_vols	work_class_meters

Section 2 is new, and also contains no change bars. The print_paging_histogram, link_meters, and meter_fim commands are obsolete, and have been removed.

The tty_meters and tty_lines commands have been moved to the Multics Administrators' Manual--Communications, Order No. CC75.

File No.: 1L13

AN52

SERIES 60 (LEVEL 68)

MULTICS SYSTEM METERING

SUBJECT

Descriptions of Metering Interfaces for Use by Multics System Programmers

SPECIAL INSTRUCTIONS

This manual supersedes AN52, Rev. 0, dated February 1975. Since new metering tools of more general interest have been added, the manual is no longer restricted.

The manual has been extensively revised; change bars in the margins indicate technical changes and additions, and asterisks in the margins indicate deletions.

Section 2 is new, and does not contain change bars. The print_paging_histogram command is obsolete and has been removed.

SOFTWARE SUPPORTED

Multics Software Release 7.0

ORDER NUMBER

AN52, Rev. 1

February 1979

Honeywell

PREFACE

This manual describes certain internal modules constituting the Multics System. It is intended as a reference for 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 (MPM). For convenience, references to these manuals are as follows:

Document	Referred to in Text as:
Commands and Active Functions (Order No. AG92)	MPM Commands
Subroutines (Order No. AG93)	MPM Subroutines
Subsystem Writers' Guide (Order No. AK92)	MPM Subsystem Writers' Guide
Reference Guide (Order No. AG91)	MPM Reference Guide
Peripheral Input/Output (Order No. AX49)	MPM Peripheral I/O
Communications Input/Output (Order No. CC92)	MPM Communications I/O

The reader using the metering commands and tools which comprise most of this manual should be thoroughly familiar with the information in the MPM. For several of the commands, the user must have privileged access to the system for comprehensive use.

File No.: 1L13

7/81

AN52A

Changes in AN52, Revision 1

The following commands and subroutines are new to this manual and do not contain change bars:

command_usage_count	meter_rcp
define_work_classes	post_purge_meters
disk_meters	ring_zero_peek_
display_bulk_err	set_work_class
interrupt_meters	tune_work_class
iobm_meters	vtoc_buffer_meters
list_vols	work_class_meters

Section 2 is new, and also contains no change bars. The print_paging_histogram and meter_fim commands are obsolete, and have been removed.

The tty_meters and tty_lines commands have been moved to the Multics Administrators' Manual--Communications, Order No. CC75.

Changes in AN52, Addendum A

The following commands and subroutines are new to this manual and do not contain change bars:

fim_meters
link_meters
metering_util_
response_meters

The meter_util_ subroutine has been superseded by the metering_util_ subroutine.

A table showing gate access requirements for the use of commands has been added at the beginning of Section 4.

Changes in AN52, Addendum B

This addendum documents new options to work class governing. The tune_work_class and work_class_meters commands have been updated to describe the use of the interactive queue by governed users. The change_tuning_parameters and print_tuning_parameters commands describe setting the integration interval for governing.

CONTENTS

SECTION 1

INTRODUCTION

This manual describes the techniques and tools used to meter many of the Multics supervisor functions. Section 2 gives a general explanation of the metering data bases used. Section 3 briefly describes the general metering techniques used. Section 4 contains usage information for the available metering commands, and Section 5 provides usage information for the metering subroutines.

The various metering tools provide information on how the Multics system works. As diagnostic aids, they can show how the users are actually using the system and what particular actions are being performed most frequently. This information can then be used to tune the system for maximum performance.

SECTION 2

DATA BASES

This section describes the two metering data bases, the system segment table (SST) and the traffic control data base (tc_data). The SST is an unpaged data base that resides in main memory at all times. It contains all data used by virtual memory management to manage the contents of main memory, and the page tables that describe segments. The subsystems of the supervisor known as page control and segment control maintain their data in the SST.

The tc_data segment is the unpaged data base of the Multics traffic controller, and also resides in main memory at all times. The Multics traffic controller is responsible for managing the assignment of physical processors to Multics processes, and all issues relating to the relative priorities of processes. The functions assumed by the traffic controller are often known as multiprogramming, multiprocessing, scheduling, dispatching, and processor management.

SYSTEM SEGMENT TABLE (SST) DATA BASE

The SST consists of four parts:

- SST header
- core map
- paging device map (which exists only if a bulk store paging device is in use)
- active segment table (AST)

The SST header contains meters, counters, list pointers, and pointers to other data bases in the SST. The core map is an array of table entries describing each frame of main memory in the system, and its contents. The paging device map is an array describing each frame of the bulk store paging device (if present), and its contents. The AST consists of entries, known as AST entries (each of which consists of 12 words of segment and page control data), and a page table. Every page table in the system is part of an AST entry; thus, the AST is that place where all page tables in a system reside.

The core map entries, paging device map entries, AST entries, and page table words contain numerous implicit and explicit threads and pointers linking data about a given page of a given segment, defining the structure of the storage system hierarchy among AST entries, and maintaining recency-of-use information for the algorithms that multiplex main memory and the AST. Tools are available for checking the consistency of these various threads.

In any given Multics configuration, a fixed number of frames of main memory are available: this is the amount of main memory configured. The task of page control is to satisfy the demand for main memory by multiplexing; i.e., controlling the sharing in an orderly fashion, of the main memory frames in response to page faults. A page fault is the hardware condition that occurs when an attempt is made to use a page that is not in main memory. The meters reported by the file_system_meters command when the -page control argument is specified (see the command description in Section 4) indicate the activity of, and load on, page control. These meters can be analyzed to determine if main memory is being used effectively, if a paging bottleneck exists, or to interpret other visible manifestations of paging behavior.

During any given bootload of Multics, a fixed number of AST entries are available; this number is thus the upper limit on the number of segments that can have page tables in main memory at any given time. Since a segment must have a page table in main memory in order to be used, the available page tables are multiplexed by segment control in response to segment faults. A segment fault is the hardware condition that occurs when an attempt is made to use a segment that does not have a page table in main memory. There are four pools of AST entries corresponding to those containing page tables describing 4, 16, 64, and 256 page segments. The number of entries to exist in each pool is specified at bootload time by the "SST" configuration card (see the Multics Operators' Handbook, Order No. AM81, for more information on configuration cards). The meters reported by the file_system_meters command indicate segment fault activity in the four AST pools. These meters can be analyzed to determine if the AST is being utilized effectively, if an AST bottleneck exists, or to interpret any other performance manifestations of segment fault activity. See the Multics Storage System manual, Order No. AN61, for more information on the format of these data bases and the meaning and interpretation of all the data contained in them.

TRAFFIC CONTROL (tc_data) DATA BASE

Four data bases reside in tc_data:

- tc_data header
- active process table (APT)
- work class table (WCT)
- interprocess transmission table (ITT)

The header of tc_data contains static meters, counters, list heads, and other information used by the traffic controller. Much of this data can be displayed via the traffic_control_meters command, described in Section 4.

By far the most important component of tc_data is the APT. The APT consists of APT entries. One APT entry exists for each process on the system. Some APT entries, e.g., those of the Initializer and the Syserr Logger daemon, exist for the duration of the bootload; most, however, are created and destroyed in response to login, logout, and new_proc commands. The APT entry of a process contains all information that the supervisor needs to know about a process when not actually running in that process. Typical of such information is the process identifier of the process, the CPU time and memory usage charged to that process, the descriptor segment base register (DSBR) value to be loaded to physically switch into that process, etc. The APT entry also contains scheduling parameters associated with the process, including dynamically computed statistics accumulated during recent running of that process. This information is used by the traffic controller to assign priorities and processor resources to the process.

APT entries can be in one of several lists, or in no lists at all. There is a list of processes ready to run, i.e., needing CPU time but not currently running, associated with each work class (see the Multics Administrators' Manual -- System, Order No. AK50, for more information). A list known as the eligible queue consists of processes to which the traffic controller has made a commitment of main memory resources. Only among these eligible processes are physical processors actually assigned; the decision to grant or revoke eligibility, i.e., place in or remove a process from this queue, is a major responsibility of the traffic controller. The order of processes in the eligible queue reflects the priority assigned to them by the traffic controller. Processes at the head of the queue are assigned a CPU before those lower down in the queue. Conversely, a process is assigned a CPU if, and only if, no process higher up in the eligibility queue can utilize a CPU.

For each processor configured on the system a special process known as the idle process of that processor exists. The idle process of a processor is a full-fledged process with an APT entry. Idle processes are run when no other work, i.e., ready process, can be found for a processor to run, or the traffic controller determines that no more processes should be granted eligibility, based on main memory sharing constraints. Time spent running idle processes is known as idle time. The total_time_meters command displays the accumulated amounts of the various types of idle time (see the command description in Section 4). The idle processes are always at the end of the eligible queue.

The work class table (WCT) consists of WCT entries defining the work classes known to the system. A WCT entry contains the parameters of the work class, recent statistics accumulated during running of processes in the work class, and the head of the queue of ready, non-eligible processes in the work class. Work class parameters and statistics can be displayed via the work_class_meters command, described in Section 4. The status of the eligible queue and work class queues can be displayed via the traffic_control_queue command, also described in Section 4.

The interprocess transmission table (ITT) consists of all interprocess messages associated with wakeups. There are two kinds of wakeups corresponding to the "fast" and "regular" event channels (see "Interprocess Communication" described in the MPM Subsystem Writers' Guide). Wakeups for "fast" event channels do not involve transmission of data between processes; rather, a bit is turned on in the APT entry of the target process, and that process is woken up. Wakeups for "regular" event channels involve copying an eight-word message from the sending process to the target process. This message includes both the sending and target process identifiers, the target event channel identifier, the validation level of the sender, and an arbitrary two-word datum supplied by the sender for receipt by the target process. This eight-word message is stored in the ITT between the time that the sending process calls the supervisor to send a wakeup, and the time that the target process, having been woken up, calls the supervisor to retrieve it. A list of these ITT messages queued to a given target process is maintained for each process; the head of this list is in the process APT entry.

In the current implementation, only a fixed number of ITT entries are available. The supervisor checks and returns an error indication to a caller if a call to wakeup might overflow the ITT.

SECTION 3

METERING DESIGN

This section gives a few recommendations for writing metering commands and subroutines. Also described here are some of the conventions used in the standard metering commands.

In general, a metering command must serve three functions: extracting the data, arranging it in a useful format, and printing it out. Each of these functions must be considered in writing metering commands.

EXTRACTING METERING INFORMATION

The meter_util_ subroutine, described in Section 5, can be used to extract the metering information from the system segment table (SST) and tc_data hardcore data bases. 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, and the detailed metering data generated by the page control and traffic control programs of the supervisor. The meter_util_ subroutine 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.

VARIOUS TYPES OF METERING TIME

Several different types of metering time are of interest to users of metering tools. They 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, segment fault, bounds fault, process switching, and interrupt processing.

idle time
: the amount of time the system was not running a program on behalf of a normal process. This time can be partitioned 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 spend 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, 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.

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 general users).

STANDARD CONTROL ARGUMENTS

Some control arguments are standard for metering commands. They 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
: resets the metering interval for the invoking process so that the interval begins at the last call with -reset specified. If the -reset control argument has never been given in a process, it is equivalent to having been specified at system initialization time (i.e., the metering interval begins at system initialization time).

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

-brief, -bf
: generates an abbreviated report (i.e., some metered data is not printed on the report produced by invoking the command).

The default report, which is generated when any metering command is given with no control arguments, prints out all of the metering data normally available. The metering interval, by default, begins at system initialization time.

SECTION 4

COMMANDS

This section contains command descriptions of metering tools on the system. In these descriptions, the variables that are printed by the commands are generally listed as they would appear on actual reports, so capitalization and indentation attempt to reflect this. Examples of the reports that are printed by invoking the commands with no control arguments are included for most of the descriptions. Brief explanations of each variable are also provided. In the usage lines, optional arguments are enclosed in braces.

Many of the commands require access to one of the gates listed in the table below. For commands not listed in the table, no special access is required for use. Additional notes follow the table.

COMMAND	GATES				
	phcs_ or metering_gate_		mdc_	hphcs_	rcp_priv_
alarm_clock_meters	X				
change_tuning_parameters*				X	
device_meters	X				
disk_meters	X				
disk_queue	X				
file_system_meters	X				
fim_meters	X				
interrupt_meters	X				
iobm_meters	X				
link_meters	X				
list_vols	X	and	X		
meter_gate	X				
meter_rcp					X
post_purge_meters	X				
print_tuning_parameters**	X				
response_meters	X				
set_work_class***				X	
system_link_meters	X				
system_performance_graph	X				
total_time_meters	X				
traffic_control_meters	X				
traffic_control_queue	X				
tune_work_class				X	
vtoc_buffer_meters	X				
work_class_meters****	X				

Table 4-1. Gate Access Requirementes

* The change_tuning_parameters command requires access to hphcs_ and metering_gate_ (not phcs_). If the -silent control_argument is used, access to metering_gate_ and initializer_gate_ is required.

** The print_tuning_parameters command requires access to metering_gate_ (not phcs_).

*** Additionally for set_work_class, read access to one or more system tables is required if the Person_id.Project_id.tag option is used. If the tag specified is "a" or "*", access to the answer_table is required. If the tag specified is "m" or "*", access to the absentee_user_table is required. If the tag specified is "z" or "*", access to the daemon_user_table is required. All of these tables are located in the directory >system_control_1.

**** In order for work_class_meters to print the names of the work classes, access to both the Master Group Table (MGT) and the answer_table is required. These tables are located in the directory >system_control_1.

Name: alarm_clock_meters

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 one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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 process initialization time.

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

Notes

If the alarm_clock_meters command is given with no control argument, it prints a full report.

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

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 control argument.

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 waiting time before rescheduling set to 0. This variable is not printed if zero.

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. This variable is not printed if zero.

alarm_clock_meters

Example

The following is an example of the information printed when the alarm_clock_meters command is invoked with no control argument.

```
Total metering time 1:36:35

No. alarm clock sims.    2274
Simulation lag             91.676 msecs.
Max. lag                 2305.883 msecs.
Priority boosts             6
```

change_tuning_parameters

Name: change_tuning_parameters, ctp

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

Usage

change_tuning_parameters name1 value1 {... nameN valueN} {-control_arg}

where:

1. namei
   is the name of a tuning parameter whose value is to be changed. It can be either the long name or the short name of the parameter.

2. valuei
   is the representation of the value to which the tuning parameter is to be set. It typically can be an integer, a decimal number of seconds, either "on" or "off," a decimal number, or a fullword octal value. The data type of the value depends on the individual tuning parameter being set. The data type of each valid tuning parameter is indicated below.

3. control_arg can be:

   -silent
   causes the message normally printed on the operator's console to announce the change not to be printed, but only to be logged. This control argument can be used only in the Initializer's process.

Notes

The following is a list of tuning parameters, the units used for each, the data types used in the change_tuning_parameter command, and the default values. Note that the data types decimal number of seconds and decimal number can be used to express fractional values (e.g., 0.001 for one second, .5 for a multiplier of one-half, etc.). For each parameter, the short name, if any, is given in parentheses.

Parameter	Unit	Data Type	Default
max_eligible (maxe)	number of processes	integer	6
min_eligible (mine)	number of processes	integer	2
max_abs_eligible (maxabs)	number of processes	integer	0 (no limit)
tefirst	number of seconds	decimal number of seconds	2.0 sec.
telast	number of seconds	decimal number of seconds	2.0 sec.

change_tuning_parameters

timax	number of seconds	decimal number of seconds	8.0 sec.
priority_sched_inc (psi)	number of seconds	decimal number of seconds	80.0 sec.
working_set_addend (wsa)	number of pages	integer	0
working_set_factor (wsf)	fractional multiplier	decimal number	1.0
double_writes (dblw)	0 = no double writes 1 = all nonpd pages get written 2 = all directories get written	integer	0
post_purge (pp)	on/off	on/off	on
gp_at_ptlnotify (gpp)	on/off	on/off	off
gp_at_notify (gpn)	on/off	on/off	off
deadline_mode (dmode)	on/off	on/off	off
int_q_enabled (intq)	on/off	on/off	on
quit_priority (qp)	fractional multiplier	decimal number	0.0
nto_delta	number of seconds	decimal number of seconds	30.0 sec.
nto_severity	number of the severity	integer	3
write_limit	number of pages	integer	1/8 of pageable memory
pre_empt_sample_time (pest)	number of seconds	decimal number of seconds	.040 sec.
process_initial_quantum	number of seconds	decimal number of seconds	2.0 sec.
gv_integration	number of seconds	decimal number of seconds	4 times the value of telast

This command requires access to metering_gate_ and to hphcs_. If the -silent control argument is specified, the command requires access to metering_gate_ and to initializer_gate_.

Before making any change, the user is shown the change and asked if it is correct. The first pair of values represent the old and new values of the parameter, while the second pair of values (parenthesized) represent the octal contents of the word in the data base where that parameter is 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, described later in this section.

change_tuning_parameters

If the int_q_enabled parameter is set to off when the scheduler is in percent mode, the specified percentages are more closely observed, but response is degraded.

The max_eligible, min_eligible, post_purge, working_set_addend, and working_set_factor tuning parameters are used to control the number of processes that are multiprogrammed (i.e., allowed to compete for pages of memory). If post_purge is on, then each time a process loses eligibility, page control uses information about the process' recent paging behavior to estimate the size of the process' working set (paging behavior information is available to page control in the process definition segment). This estimate is then multiplied by the wsf, and the result is added to wsa to arrive at a final value for the working set of the process. When post_purge is off, the estimated working set is always zero.

The scheduler uses the following criteria to determine whether to make an additional process eligible:

1. Another process may be made eligible if fewer than min_eligible processes are currently eligible.

2. Another process may not be made eligible if more than max_eligible processes are currently eligible.

3. When the number of eligible processes is between mine and maxe, another process may be made eligible if the sum of the working sets of the eligible processes and the working set of the process being considered for eligibility is less than the number of pages of pageable (nonwired) main memory.

Checks 2 and 3 are ignored for a process in a realtime work class if that process' deadline has passed.

Thus, if one were to make any of the following changes:

increase min_eligible
increase max_eligible
decrease working_set_factor
decrease working_set_addend

it would tend to increase multiprogramming, and therefore reduce MP idle and increase page thrashing. Conversely, decreasing mine or maxe, and increasing wsf or wsa, would have the opposite effect.

The max_abs_eligible parameter provides the additional constraint that no absentee process is made eligible if the number already eligible equals max_abs_eligible. This parameter is ignored if the scheduler is in deadline mode or if the parameter is zero.

The max_max_eligible parameter determines the maximum to which max_eligible and max_abs_eligible can be set. It can only be changed by using a SCHD card in the config deck. (See the Multics Operators' Handbook, Order No. AM81, for more information on the config deck. A SCHD card can also be used to change wsf, mine, and maxe.) The max_max_eligible parameter defaults to ten processes more than maxe.

change_tuning_parameters

The nto_delta and nto_severity parameters control the action the system takes when a notify timeout occurs. (A notify timeout occurs when a process has been waiting for some event for longer than the nto_delta, and is generally symptomatic of some programming or hardware error.) With the tuning parameters set at their default values, when a notify timeout occurs the process is taken out of the waiting state and a message is printed on the operator's console. The nto_severity parameter controls the "severity" of the call (made by the system program syserr) that prints this message. Useful values for nto_severity are:

-1 no call to syserr is made, so no record of the notify timeout is printed or logged (useful for temporarily masking a problem).

1 the system is crashed after the message is printed (useful for debugging by system programmers).

3 the message is printed but no further action is taken.

The priority_sched_inc tuning parameter can be used to make processes that block always appear to be interactive (thereby getting favorable scheduling). Wakeups sent by the tape or tty interrupt handlers are interactive; wakeups caused by timers or explicit user action are not interactive unless the target process has been blocked for longer than priority_sched_inc. Thus, when priority_sched_inc is set to .125 second, an absentee process that pauses for longer than this appears to be interactive to the scheduler.

The gp_at_ptlnotify and gp_at_notify parameters are used to control the actions of the get_processor function of the scheduler. In all cases, get_processor attempts to find an idle processor on which to run the notified process. When notifying processes waiting for the page table lock, if gp_at_ptlnotify is on, then an attempt is made to preempt a lower priority process (one deeper in the eligible queue). When processes waiting for other events are notified, the attempt to preempt a lower priority process is made only if gp_at_notify is on. If gp_at_ptlnotify is set to "on," paging throughput increases at the cost of preemption overhead. If gp_at_notify is set to "on," I/O throughput increases at the cost of preemption overhead. The effect of setting these parameters "on" is improved performance for processes that take page faults frequently, at the expense of the considerable overhead of the preempt mechanism.

The pre_empt_sample_time parameter is used to set the maximum clock time between invocations of traffic control. This is implemented by limiting the value of the timer register when a process is given a CPU to:

pre_empt_sample_time X (number of CPUs online)

With parameters gp_at_notify and gp_at_ptlnotify set to "off" (the default value for both parameters), traffic control checks for higher priority processes to run at least once every pre_empt_sample_time.

change_tuning_parameters

If the deadline_mode tuning parameter is on, processes in non-realtime work classes are scheduled according to the deadlines and quanta specified for their work classes. If off (the default), then non-realtime work classes are scheduled according to the percentages specified for their work classes. The answering service sets this parameter at shift change time to the value specified by the master group table (see the ed_mgt command in the Multics Administrators' Manual--System, Order No. AK50).

The tefirst, telast, and timax tuning parameters are irrelevant in deadline mode. If the scheduler is in percent mode, however, the initial quantum awarded by the scheduler after an interaction is tefirst. At all other times, the scheduler awards a quantum of telast. In percent mode, processes within a work class are sorted either by the amount of CPU time used since the process has interacted, or by timax, whichever is less. Thus, timax sets a limit on the depth in its work class queue to which a process can sink. The following are examples of the use of various settings of tefirst and telast in percent mode:

tefirst	telast	MOTIVE
1.0 sec	0.5 sec	first trial setting
0.5	0.25	better trivial response
2.0	0.5	better response for commands using 1 to 2 seconds
1.0	2.0	better response for long-running commands

The tefirst, telast, and timax parameters can be set by using a SCHD card in the config deck (see the Multics Operators' Handbook, for more information).

The process_initial_quantum tuning parameter controls the first quantum awarded by the scheduler to a newly created process. Higher values of this parameter improve the responsiveness of the system to logins. For a newly created process, this parameter affects only processing until the first terminal interaction with the user (usually, the first interaction is the first command line input by the user). If improved login response is desired, this parameter should be set higher than the average virtual cpu time reported in the first "ready" message for a typical user.

The double_writes tuning parameter is useful only on systems with a paging device. It controls which pages are written to disk at the same time they are written to the paging device. Double writes are useful only when the paging device consistently has problems, in which case loss of data may be reduced, but at some cost in throughput.

change_tuning_parameters

The quit_priority parameter can be used administratively to control the action of the scheduler when a user quits. Interactive users sometimes quit and restart in order to appear interactive during a long-running command. When quit_priority is 0, users appear interactive. If the value is 1, the users' scheduling priority is unchanged, and if it is 2, the users' priority is actually degraded. This parameter is usually set at 0, but it can be increased if quit/restarts are a problem at a site.

The write_limit parameter determines the maximum number of pages that can be simultaneously queued for writing out. It can also be set by using a PARM card (see the Multics Operators' Handbook). The recommended value is N*44 + 14, where N is the number of disk subsystems configured. Thus, the value is 58 for a configuration with one disk subsystem and is increased by 44 for each additional disk subsystem.

The gv_integration parameter controls the granularity of control for work class governing. Approximately, it is the interval over which governing is enforced. For example, if it is set to 3600. (one hour), a work class with a limit of 10% is limited to 10% of available CPU time within any given one hour period. The default value of this parameter causes governing to be instantaneous; that is, the governing is 4 times the value of telast, and it is the smallest value to which gv_integration can be set.

Name: command_usage_count, cuc

The command_usage_count command provides a record of the number of times commands are used and the User_ids for each invocation of them. The commands to be metered in this way must be listed in a segment named command_usage_list_. Usage totals are stored in a segment named command_usage_totals_. This command actually performs three operations: it prints and clears the meters, adds commands to command_usage_list_, and deletes commands from command_usage_list_.

Usage

command_usage_count operation {command_names} {-control_args}

where:

1. operation can be one of the following:

   print, pr
   prints (and clears) the metered data (subject to any restrictions the specified control_args impose).

   add
   adds commands to the list (command_usage_list_) of commands to be metered. Commands added to the list in a single invocation of the "cuc add" command form a command "group", which can be manipulated as a whole.

   delete, dl
   deletes command groups (see above) from the list of commands to be metered.

2. command_names
   are long or short names of commands. If given with either the print or delete operation, only one command name from each group to be printed or deleted need be given, and all the commands in each group so represented are acted upon. If no names are given with the print or delete operation, all command groups are printed/deleted. Command names (long and/or short) must be given with the add operation, and all the names added in a single invocation are added as a single group to the list. Short names of commands can only be used with the print and delete operations if they were specified with the add operation.

3. control_args can be chosen from the following:

   -all, -a
   prints meters for all the command groups, or deletes all command groups from the list. This is the default for the print and delete operations if no command_names are given. This control_arg cannot be used with the add operation.

command_usage_count

-total, -tt
prints only the total use of the commands in the specified command groups, when used with the print operation. When used with the add operation, meters only the total usage of commands specified. The default with both of these operations is to print/meter the users of the commands as well as total usage. See "Notes" below. This control_arg cannot be used with the delete operation.

-brief, -bf
omits column headings from the printout (can only be used with the print operation). The default is to print column headings.

-clear, -cl
clears the usage counters and user list when meters are printed (can only be used with the print operation). It clears the user list even if the -total control_arg is also specified.

-first N, -ft N
prints only the N greatest users of the specified commands (can only be used with the print operation). This control_arg cannot be used in conjunction with the -total control_arg.

Notes

In order to add and delete commands, and to clear meters, the user must have rw access to the command_usage_list_ segment. Otherwise, all users should have r access to command_usage_list_, and rw access to command_usage_totals_. Both segments are found using object search rules and most commonly are in >sss (system_library_standard directory). If they are not in >sss, a link in >sss points to them.

For each group of commands added without the -total control argument, this command creates a segment named command_name.usage in >sss (or, if a link is there, wherever the link points). The user must put the link in >sss before the first usage of cuc add, since the metering program creates the command_name.usage segment in the same directory in which it finds command_usage_list_. The command_name.usage segment contains the list of User_ids for those using the commands in the group. User_ids are printed in the order of greatest usage. When the -first control argument is given, in addition to printing the user count and name for the N greatest users, this command prints an additional line giving the user count for "all others."

At sites using the access isolation mechanism (AIM), only the usage of system_low users is recorded.

Example

In the following example, assume that no commands are listed in the command_usage_list_ segment. The user adds two commands (in two separate command groups) by typing:

```
command_usage_count add set_search_rules ssr
cuc add enter_abs_request ear -total
```

command_usage_count

The next time a process is created, those commands can be metered by typing:

```
cuc print
```

The following lines are then displayed:

```
USAGE     COMMAND   USER     USER
COUNT     GROUP     COUNT    NAME
    3     set_search_rules
          ssr
                        2    Baker.Multics
                        1    Green.SysMaint

    1     enter_abs_request
          ear
```

Note that user count and user name are not provided for the command group added with the -total control_arg.

To delete these commands from the list, the user types:

```
cuc delete -all
```

or the equivalent:

```
cuc dl
```

define_work_classes

Name: define_work_classes, dwc

The define_work_classes command provides a means to assign percentages of CPU time to various work classes independently of what is specified in the master group table (MGT). The effect of this command is temporary, since the answering service defines the work classes specified in the MGT at next shift change or the next time the operator issues the "maxu auto" command (described in the Multics Operators' Handbook, Order No. AM81). The percentages are ignored if the scheduler is in deadline mode.

Usage

define_work_classes pct_wc1 {pct_wc2 ... pct_wcn}

where pct_wci is the percentage of CPU time to be assigned to the ith work class. If pct_wci is zero, the ith work class is not defined. If pct_wci is not specified (there is no ith argument) then the ith work class is not defined.

Notes

The percentages specified are normalized to a sum of 100% by the hardcore scheduler.

This command does not allow work classes defined in the MGT to be undefined. No changes made by this command are recorded in the MGT. Work classes created by the command may be undefined by this command. No work class may be undefined which contains active processes. All work classes are changed out of realtime mode by the define_work_classes command, although the tune_work_class command (described later in this section) may be used after define_work_classes to change some work classes into realtime work classes.

The define_work_classes command is useful in the following cases:

1. While testing or tuning the priority scheduler, define_work_classes may be used to rapidly change the percentages assigned to different work classes.

2. If the percentages specified in the MGT are temporarily out of balance, define_work_classes may be used to handle the transient condition gracefully without installing a new MGT.

3. Special or ad hoc work classes may be estalished, at the administrator's discretion, for the duration of a shift. The set_work_class command may then be used to move certain processes into the new work class.

define_work_classes

Example

The following three command lines are equivalent, and define work classes 1, 2, 3, 4, and 6, each with 20% of available CPU time.

```
dwc 20 20 20 20 0 20

dwc 20 20 20 20 0 20 0

dwc 1 1 1 1 0 1
```

Assume work classes 1, 2, and 3 already exist with 50%, 30%, and 20% of CPU time, respectively. To temporarily create a special work class and give it approximately 10% of available CPU time, while holding the relative amounts of CPU time received by the other work classes constant, type:

```
dwc 50 30 20 10
```

The actual effect is to give the four work classes 46%, 27%, 18%, and 9%, respectively.

device_meters

Name: device_meters, dvm

The device_meters command prints out metering information for the page control subsystems. Information is printed for the disk subsystems (dska, dskb, etc.). If the system includes a paging device (bulk store), information is printed for it also.

Usage

device_meters {-control_args}

where control_args can be chosen from the following:

-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 for the invoking process so that the interval 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.

Notes

If the device_meters command is given with no control arguments, it prints a full report.

The following variables are printed if the -i/o control argument is specified. All numbers are per subsystem.

Prior Page I/O
is the number of high priority page transfers (mostly reads) performed.

ATB
is the average time between high priority page transfers.

Other Page I/O
is the number of low priority page transfers (mostly writes) performed.

ATB
is the average time between low priority page transfers.

ATB I/O
is the average time between all page transfers.

Prior VTOCE I/O
is the number of volume table of contents (VTOC) entry transfers (all are high priority).

ATB
is the average time between VTOC entry transfers.

ATB I/O
is the average time between all transfers.

% Busy
is the percentage of time any logical channels are in use (maximum value = 100 * N, where N is the number of logical channels).

The following variables are printed if the -latency control argument is specified.

Avg. Page Wait
is the average delay between the queuing of a high priority page transfer and its completion.

Avg. Page ^Wait
is the average delay between the queuing of a low priority page transfer and its completion.

Avg. VTOCE Wait
is the average delay between the queuing of a VTOC entry transfer and its completion.

Avg. Page I/O T
is the average delay between the issuing of a page transfer and its completion.

Avg. VTOCE I/O T
is the average delay between the issuing of a VTOC entry transfer and its completion.

The following variables are printed if the -error control argument is specified.

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.

Errors
is a count of the times that 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.

device_meters

Example

The following is an example of the information printed when the device_meters command is invoked with no control arguments.

```
Total metering time   12:17:40

                          bulk          dska

Prior Page I/O         4716159        982744
ATB                       9.384        45.038
Other Page I/O         9581248        345374
ATB                       4.619       128.153
ATB Page I/O              3.095        33.326
Prior VTOCE I/O               0        384102
ATB                       0.000       115.232
ATB I/O                   3.095        25.850
% Busy                        0           128
Avg. Page Wait            1.510        49.023
Avg. Page ^Wait           0.000        79.164
Avg. VTOCE Wait           0.000        43.327
Avg. Page I/O T           0.000        39.717
Avg. VTOCE I/O T          0.000        31.716
EDAC Corr. Errs               0             2
Errors                        0             0
Fatal Errors                  0             0
```

Name: disk_meters

The disk_meters command prints statistics for each disk subsystem. A disk subsystem is a set of channels and devices where every channel can reach every device.

Usage

disk meters {subsystem_name} {-control_args}

where:

1. subsystem_name
   is the name of a disk subsystem (dska, dskb, dskc, etc.) for which information is to be printed. If not specified, information for all subsystems is printed.

2. control_args
   can be one of the following:

   -reset, -rs
   resets the metering interval for the invoking process so that the interval 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.

Notes

If the disk_meters command is given with no control arguments, it prints a full report.

If a reset is done, the meters for all subsystems are reset, whether or not subsystem_name is specified.

The following are brief descriptions of each of the variables printed by disk_meters.

call locks
are lockings of a disk subsystem in order to queue read or write operations. Information displayed includes the number of lockings (Count), the number of waits for the lock (Waits), the percentage of lockings that needed to wait (%Waits), and the average wait time (Avg. Wait (ms.)).

run locks
are lockings of a disk subsystem in order to check the status of all devices and channels. Run is called by page control on all subsystems when too many write transfers are pending. Information displayed is the same as for call locks. If the number of run lockings is greater than a tenth of the number of call lockings, a transfer bottleneck probably exists at the subsystem, channel, or device level.

interrupt locks
are lockings of a disk subsystem at interrupt time. Information displayed is the same as for call locks.

allocations
are metered whenever a transfer request is queued within a disk subsystem. Information displayed includes the number of allocations (Count), the number of times queueing of the request had to wait for the completion of a previous request, i.e., no free queue entry was available for use (Waits), the percentage of allocations which had to wait (%Waits), and the average time in milliseconds until a queue entry became available (Avg. Wait (ms.)). If the allocation %Waits is greater than 2%, a transfer bottleneck probably exists at the subsystem, channel, or device level.

For each drive in the subsystem, the following meters are displayed.

Reads
is the number of page or volume table of contents (VTOC) reads from the device.

Writes
is the number of page or VTOC writes from the device.

Seek Distance
is the average seek distance in cylinders on the device. If this average is greater than a third of the number of cylinders, then the pages are probably poorly distributed on the pack.

ATB Reads
is the average time between reads from the drive.

ATB Writes
is the average time between writes to the drive.

ATB I/O
is the average time between transfers on the drive. This number should generally be within 50% of the average ATB I/O. Any drive with an ATB I/O that is either less than half that of a typical drive, or less than 45-55 milliseconds, is probably a bottleneck.

Notes

Model 500 and 501 disk drives are configured as a pair of consecutively numbered physical devices. For these types of drives, only one set of ATB Reads, ATB Writes, and ATB I/Os are displayed for each pair.

Example

The following is an example of the information printed when the disk_meters command is invoked with no control argument, but the dska subsystem name is specified.

```
Total metering time   6:38:04

Subsystem dska          Count      Waits    %Waits    Avg. Wait(ms.)

call locks             626042       9941      1.59         0.281
run locks                3222         76      2.36         0.519
interrupt locks        625830      12203      1.95         0.290
allocations            625936          0      0.00         0.000

Drive      Reads     Writes     Seek      ATB       ATB       ATB
                              Distance   Reads    Writes      I/O

  1          416        382       11     57415     62525     29930
  2        18540      15868      115      1288      1505       694
  3        23485      19312       78      1017      1236       558
  4         3177       2275       37      7518     10498      4380
  5        22238      17779       86      1074      1343       596
  6          188         16       19    127047   1492805    117082
  7        23128      16808       71      1032      1421       598
  8        97537      21366       58       244      1117       200
  9        20771      15955       92      1149      1497       650
 10        52398       9912       43       455      2409       383
 11        55443       9824       54       430      2431       365
 12        22927      17714       61      1041      1348       587
 14        34519      28613      137       691       834       378
 15        12259       1346      141      1948     17745      1755
 16        35683      25760      107       669       927       388
```

disk_queue

Name: disk_queue, dq

The disk_queue command prints out the subsystem channel activity and the waiting I/O requests queued for a given secondary storage subsystem.

Usage

disk_queue subsytem_name

where subsytem_name is the subsystem name and can be of the form dska, dskb, dskc, etc.

Notes

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

P
: is the priority of the request (1=high; 0=low).

RW
: indicates the type of request (R=read; W=write).

VP
: indicates the type of request (P=page; V=volume table of contents entry).

DV
: represents the device to which the request is directed.

SECTOR
: is the sector to or from which input/output is done.

MEM
: is the main memory address to or from which input/output is done.

display_bulk_err

Name: display_bulk_err

The display_bulk_err command displays statistics about hardware errors on the bulk store paging device.

Usage

display_bulk_err {-control_arg}

where control_arg can be one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the display_bulk_err command is given with no control argument, it prints a full report.

The following are brief descriptions of the metering variables printed out by the display_bulk_err command.

bulk store errors
is the number of bulk store operations that encountered an uncorrectable error.

fatal errors
is the number of operations that could not be retried successfully, and thus caused data loss.

errors corrected by EDAC
is the number of errors detected and corrected by bulk store hardware.

Example

The following is an example of the information printed when the display_bulk_err command is invoked with no control argument.

```
Total metering time    1:15:42

0 bulk store errors
0 fatal errors
0 errors corrected by EDAC
```

file_system_meters

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 chosen from the following:

-ast
prints certain meters about active segment table (AST) usage.

-page, -pg
prints certain meters about paging. *

-brief, -bf
generates a shortened report. Those meters not printed if -brief is specified are indicated by a plus (+) in "Notes" below.

-reset, -rs
resets the metering interval for the invoking process so that the interval 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 control argument.

Notes

If the file_system_meters command is given with no control arguments, it prints a full report.

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

Activations
is the number of segment activations. *

segfault
is the number of activations caused by segment faults; also expressed as a percentage of all activations.

makeknown
is the number of activations resulting from explicit calls from the makeknown_ routine; also expressed as a percentage of all activations.

file_system_meters

backup
is the number of activations resulting from calls to activate$backup_activate; also expressed as a percentage of all activations.

directories
is the number of directories activated; also expressed as a percentage of all activations.

Deactivations
is the number of segment deactivations.

Demand deactivate
attempts
is the number of deactivations explicitly requested by users.

successes
is the number of demand deactivations which succeeded; also expressed as a percentage of attempts and as a percentage of all deactivations.

Seg Fault (flt)
is the number of segment faults.

Seg Fault (call)
is the number of calls to the segment fault handler to activate a segment without taking a segment fault.

Bound Faults
is the number of bound faults.

+ Setfaults (all)
is the number of setfaults performed during segment deactivation and during the handling of bound faults. Setfaults are segment faults forced when dynamic segment attributes are changed (e.g., access to a segment is revoked by another process).

+ 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
is the number of times an entry was skipped; also expressed as a percentage of Steps.

+ 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 segments that cannot be deactivated. Also expressed as a percentage of Skips.

+ mem
is the number of times an entry was skipped because it had pages in memory; also expressed as a percentage of Skips.

file_system_meters

+ 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; also expressed as a percentage of Skips.

+ Searches
  is the number of full AST searches required because no entry was readily available.

+ Avg. Cost
  is the average "cost" in I/Os of deactivations arising from full searches.

Cleanups
  is the number of calls to cleanup. The percentage of real time spent in cleanup is also given.

with any rws
  is the number of times any pages moved from bulk store to disk as a consequence of a cleanup operation (not printed if zero).

rws count
  is the actual number of pages moved (not printed if zero).

Force writes
  is the number of calls to force_write. The three following meters relating to force_writes are printed only if any force_writes occurred.

without pwrites
  is the number of times force_write wrote no pages.

pages written
  is the number of pages written by force_write.

force updatev
  is the number of calls to update_vtoce resulting from force_writes.

Lock AST
  is the number of lockings of the AST.

The following meters provide information about AST lock contention.

AST locked
  is the average real time during which the AST lock is held locked and the percentage of the metering interval during which the AST was locked. This percentage cannot exceed 100%, and the closer the 100% figure is approached, the more AST lock contention becomes the limiting function in system throughput.

AST lock waiting
  is the average real time delay between an attempt to lock the AST and successful locking of the AST. The total real time spent by all processes waiting for the AST lock, expressed as a percentage of the metering interval, is also given. This number may exceed 100% if, on the average, more than one process was waiting for the AST lock.

file_system_meters

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

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 the -page control argument is specified. The two columns printed by this command contain the number of occurrences of the specified item and the average time between occurrences.

Needc
is the number of times a frame of main memory 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.

Claim runs
is the number of times the page removal algorithm could not queue an additional write until a previous write was completed. If the average time between claim runs is less than .010 minutes, then an I/O bottleneck probably exists in the system.

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 occur on pages of segments in process directories.

Level 2 faults
is the percentage of page faults on pages of segments in directories directly off the root. This is a measure of the activity of the system libraries.

DIR faults
is the percentage of page faults on directory pages.

FSDCT
is the number of page faults taken on free storage maps.

Laps
is the number of times the used pointer has gone around the main memory used list in the search for a usable block of main memory.

+ Steps
is the number of steps taken around the main memory used list. A step consists in moving the used pointer to the next entry on the list.

+ Skip
is the number of times a page was skipped; also expressed as a percentage of Steps.

+ wired
is the number of times a page was skipped while searching the main memory used list because it was wired down; also expressed as a percentage of Skip.

+ used
is the number of times a page was skipped because it was used in the last lap; also expressed as a percentage of Skip.

+ mod
is the number of times a page was skipped because it had been modified; also expressed as a percentage of Skip.

+ fc pin
is the number of times a page was skipped by find_core because it was pinned; also expressed as a percentage of Skip.

+ cl pin
is the number of times a page was skipped by claim_mode_core because it was pinned; also expressed as a percentage of Skip.

pages
is the number of pages available in the system. This is the total main memory minus the permanently wired down supervisor.

wired
is the number of pages temporarily wired down. This includes descriptor segments and process data segments (PDS) for loaded processes.

Average steps
is the average number of steps taken around the main memory used list to find a usable frame of main memory.

file_system_meters

Example

The following is an example of the information printed when the file_system_meters command is invoked with no control arguments.

```
Total metering time   1:33:17

                          #          ATB

Activations              1853       3.021 sec.
   segfault              1564       3.579 sec.      84.404% of all
   makeknown              289      19.370 sec.      15.596% of all
   backup                   0       0.000 sec.       0.000% of all
   directories            293      19.106 sec.      15.812% of all
Deactivations            1168       4.793 sec.
Demand deactivate
   attempts               317      17.659 sec.
   successes               79      70.861 sec.      24.921%, 6.764% of total
Seg Fault (flt)          4233       1.322 sec.
Seg Fault (call)         2482       2.255 sec.
Bound Faults              385      14.540 sec.
Setfaults (all)          4022    1391.842 msec.
Setfaults (acc)           107      52.318 sec.
Updates                   214      26.159 sec.
Steps                    5125    1092.290 msec.
Skips                    2898       1.932 sec.      56.546% of Steps
   ehs                    316      17.715 sec.      10.904% of Skips
   mem                   1043       5.367 sec.      35.990% of Skips
   init                  1539       3.637 sec.      53.106% of Skips
Searches                    7     329.293 sec.
Avg. Cost                 1.0
Cleanups                 1215       4.607 sec.      0.0 % of real time
   with any rws            76     267.095 sec.
   rws count              604     112.585 sec.
Force writes                0       0.000 sec.
   without pwrites          0       0.000 sec.
   pages written            0       0.000 sec.
   force updatev            0       0.000 sec.
Lock AST                22954       0.244 sec.

                      AVE/lock          %
AST locked          4.282 msec.       1.8
AST lock waiting    3.699 msec.       4.1

AST Sizes                 4          16          64         256
Number                  500         200          80          15
Need                   1312         530         337          65
Steps                  2592        1292        1076         165
Ave Steps               2.0         2.4         3.2         2.5
Grace (sec)          1079.1       866.6       416.2       508.9
```

file_system_meters

	#	ATB		
Needc	39367	142.200	msec.	
Ceiling	41	18.437	min.	
Claim runs	73	1.278	min.	
Ring 0 faults		26.565	%	
PDIR faults		21.048	%	
Level 2 faults		34.755	%	
DIR faults		9.711	%	
FSDCT	29	193034.027	msec.	
Laps	197	28.416	sec.	
Steps	236549	23.665	msec.	
Skip	215266	26.005	msec.	91.003% of Steps
wired	15730	355.880	msec.	7.307% of Skip
used	58940	94.978	msec.	27.380% of Skip
mod	66859	83.728	msec.	31.059% of Skip
fc pin	55653	100.587	msec.	25.853% of Skip
cl pin	18084	309.555	msec.	8.401% of Skip

298 pages, 51 wired.
Average steps 4.416

Name: fim_meters

The fim_meters command is used to interpret and print per-system metering information on central processor faults.

Usage

fim_meters {fault_name} {-control_args}

where:

1. fault_name
   is the name of a single fault type (valid fault types are listed under "Notes" below). Only the information for that fault type is printed. If fault_name is not specified, then information for all fault types is printed. No control argument can be given if a fault_name is specified.

2. control_arg
   can be chosen from the following if fault_name is not specified:

   -reset, -rs
   resets the metering interval for the invoking process so that the interval 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 a reset operation.

   -sort {STR}
   sorts the output as specified by STR, which can be either "count" or "number". If STR is not specified, the output is sorted by count. If this control argument is not specified, the output is sorted by hardware fault number.

   long, -lg
   prints information on software-interpreted subordinate faults with each associated hardware fault.

Notes

If both -long and -sort are used, subordinate faults are sorted within associated hardware fault.

The following is a brief description of the variables printed by the fim_meters command.

type
: is the name of the hardware fault.

count
: is the total number of times the fault occurred on any central processor configured.

The following are the valid names of hardware faults. These names are printed as "type" and can be used as the fault_name argument. Full descriptions of these faults can be found in Multics Processor Manual, Order Number AL39.

access_violation, acv
command, cmd
connect, con
derail, drl
directed_fault_2, df2
directed_fault_3, df3
divide_check, div
execute, exf
fault_tag_1, ft1
fault_tag_3, ft3
illegal_procedure, ipr
linkage_fault, ft2
lockup, luf
mme1, mme
mme2
mme3
mme4
op_not_complete, onc
overflow, ovf
page_fault, df1
parity, par
segment_fault, df0
shutdown, sdf
startup, suf
store, str
timer_ranout, tro
trouble, trb

Subordinate faults are recursive calls to the main fault processor that can be interpreted as subcases of hardware faults. For example, the hardware generated "command" fault may be interpreted as an isot_fault, a lot_fault, or a command_fault, depending on other conditions at the time of the fault. A complete description of subordinate faults would require a description of Multics fault processing, and is beyond the scope of this manual.

Example

The following is an example of the information printed when the fim_meters command is invoked with no control arguments.

```
fim_meters

Total metering time:                    0:10:11

                                            count
     shutdown                                   0
     store                                      0
     mme1                                       0
     fault_tag_1                                0
     timer_runout                             943
     command                                    0
     derail                                     0
     lockup                                     0
     connect                                 5283
     parity                                     0
     illegal_procedure                          0
     op_not_complete                            0
     startup                                    0
     overflow                                   0
     divide_check                               0
     execute                                    0
     segment_fault                           1153
     page_fault                              7320
     directed_fault_2                           0
     directed_fault_3                           0
     access_violation                        1255
     mme2                                       0
     mme3                                       0
     mme4                                       0
     linkage_fault                           3211
     fault_tag_3                                0
     trouble                                    0
```

flush

Name: flush

The flush command causes 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:

-temp_dir path, -td path
specifies that temporary segments used for flushing main memory are to be created in the directory identified by path (the default is to create them in the process directory).

Notes

In order for all pages in main memory to be flushed, the directory used for temporary segments must have sufficient quota for as many pages as there are in main memory.

instr_speed

Name: instr_speed

The instr_speed command measures 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 processor (with the use of the set_proc_required command described in Hardware Diagnostic Aids, Order No. AR97) before running timing tests.

Usage

```
instr_speed
```

Notes

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

interrupt_meters

Name: interrupt_meters, intm

The interrupt_meters command prints out metering information for input/output multiplexer (IOM) channel interrupts.

Usage

interrupt_meters {-control_args}

where control_args can be chosen from the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

-total, -tt
prints out only the total IOM and nonIOM interrupt metering information.

-iom N
prints out interrupt metering information only for those channels on IOM N.

-channel N
prints out interrupt metering information only for IOM channel N.

Notes

If the interrupt_meters command is given with no control arguments, it prints a full report.

The following are brief descriptions of the metering information printed out by interrupt_meters.

Int
is the number of interrupts which occurred.

Avg Time
is the average time (in milliseconds) needed to handle each interrupt.

% CPU
is the percentage of total CPU time needed to handle the interrupts.

Name
is the name of the device on the channel.

interrupt_meters

The following are descriptions of the totals printed by interrupt_meters.

Chan
is the total of all IOM channel interrupts. The times printed are based on the total time spent in the per-channel interrupt handlers.

Other
is the total of all IOM interrupts. Each IOM interrupt may cause the handling of several channel interrupts. The times printed include only that time in the common interrupt path and exclude time spent in the per-channel handlers.

Total
is the total of all interrupts handled by the system.

Example

The following is an example of the information printed when the interrupt_meters command is invoked with no control arguments.

```
Total metering time    26:23:30

IOM Ch      Int  Avg Time  % CPU  Name

 1   6      201    1.067     0.00  special
 1  10    78956    6.520     0.27  impr
 1  11    33928    1.478     0.03  impw
 1  15    37933    1.370     0.03  prta
 1  16       18    1.741     0.00  rdra
 1  17       56    1.269     0.00  puna
 1  20      343    1.065     0.00  opc
 1  21   628614    3.009     1.00  fnp a
 1  22    34731    1.306     0.02  tape
 1  23     3916    1.372     0.00  tape
 1  24   989785    0.813     0.42  dska
 1  25    23027    0.821     0.01  dska
 1  26   167782    0.819     0.07  dska
 1  27     1811    0.837     0.00  dska
 1  30   422948    0.816     0.18  dska
 1  31     7082    0.816     0.00  dska
 1  32    65616    0.816     0.03  dska
 1  33      409    0.856     0.00  dska

Chan    2497156    1.573     2.07
Other   2486450    0.454     0.59
Total   2486450    2.034     2.66
```

iobm_meters

Name: iobm_meters

The iobm_meters command prints out the metering information compiled in the hardcore ring by the input/output buffer manager (IOBM).

Usage

iobm_meters {-control_arg}

where control_arg can be one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the iobm_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of the metering variables printed out by the iobm_meters command.

Requests
is the number of requests made to IOBM to wire buffer pages, and the total number of pages requested to be wired.

Wirings
is the total number of times and the total number of pages that were wired as a result of requests to IOBM.

New Requests
is the number of requests to wire buffers not previously wired and the average real time spent processing each such request.

Matching Requests
is the number of requests to wire buffers already wired and the average real time spent processing each such request.

Unwire Calls
is the number of calls to release for unwiring a previously wired buffer, and the average real time spent processing each call.

Time-out Calls
is the number of times the traffic controller called IOBM to actually unwire previously released buffers, and the average real time spent processing each call.

Lock Loops
is the number of times IOBM had to try more than once to wire a buffer.

Inconsistent Calls
is the number of times IOBM was called to wire a buffer already wired and not yet released.

No Core Calls
is the number of times IOBM was unable to find enough memory to wire a caller's buffer.

Example

The following is an example of the information printed when the iobm_meters is invoked with no control argument.

```
Total metering time  12:02:10

                              #          PAGES

Requests                  42662         126268
Wirings                     127            355

                              #      Avg. Time

New Requests                127         15.832
Matching Requests         42535          0.268
Unwire Calls              42661          0.109
Time-out Calls             2683          0.270
Lock Loops                    0
Inconsistent Calls            0
No Core Calls                 0
```

Name: link_meters

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

Usage

link_meters {-control_arg}

where control_arg can be one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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 process initialization time.

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

Notes

If the link_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of the variables printed by the link_meters command.

slot
is a time slot into which the calls to the linker are broken down. The four slots are for calls completed in less than 25 milliseconds, calls completed in between 25 and 50 ms, calls completed in between 50 and 75 ms, and calls completed in more than 75 ms.

calls
is 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
is the average time (in milliseconds) to completion for a call in each slot and the average time to completion for all calls to the linker made by the process.

avg pf
is the average number of page faults for a call in each slot and the average number of page faults for all calls made by the process.

link_meters

tot time
is the total virtual time (in seconds) taken by calls in each slot and the total virtual time spent in the linker by the process. It equals calls times average time.

% time
is the percentage of total linker time for the process that was taken by calls in each slot and the percentage taken by all calls.

Example

The following is an example of the information printed when the link_meters command is invoked with no control argument.

```
Linkage Meters:

slot     calls  avg time  avg pf  tot time  % time

<25        286    20.014     1.5     5.724    81.1
25-50       15    77.714     9.5     1.166    16.5
50-75        1   166.862    32.0      .167     2.4
>75          0     0.000     0.0     0.000     0.0
         -----  --------  -----  --------
Total      302    23.367     2.0     7.057
```

list_vols

Name: list_vols

The list_vols command prints information about currently-mounted physical or logical volumes. Several of the items printed by list_vols can also be obtained as return values by invoking list_vols as an active function.

Usage

```
list_vols {-control_args}
```

where control args can be chosen from the following:

-lv name
: prints information about only the logical volume named. This is the default if a name is given without -lv or -pv preceding it.

-pv name
: prints information about only the physical volume named.

-totals, -tt
: specifies that information should not be printed for individual physical volumes, but rather should be totalled and printed for each logical volume.

-records
: prints only the number of records on the specified volume(s), exclusive of records occupied by partitions and the volume table of contents (VTOC). This is one of the items that can be obtained as an active function return value.

-records_left
: prints only the number of records on the specified volume(s) that are currently unused and are available to hold the pages of segments and directories. This is one of the items that can be obtained as an active function return value.

Notes

If no volume name is given, the list_vols command prints information about all mounted logical volumes.

If list_vols is used as an active function, either the -records or the -records_left control argument must be given.

If the -totals argument is given together with the name of a logical volume, a single line containing totals information for that logical volume is printed.

If physical volume information is being printed (-totals not given), the output lines contain the following items:

Drive flag Records Left VTOCEs Left PV Name PB/PD LV Name

If logical volume information is being printed (-totals given), the output lines contain the following items:

Records Left VTOCEs Left PB/PD Lv Name

The following are brief descriptions of the above variables.

Drive
is the name of the drive on which the physical volume is mounted.

flag
is the letter "X" if the drive is inoperative, or the letter "P" if the physical volume contains partitions that are described in the CONFIG deck.

Records
is the number of records not occupied by partitions or the VTOC, and therefore usable for the pages of segments and entries.

Left
is the number of records currently unused and therefore available for the pages of segments and directories.

VTOCEs
is the number of VTOC entries.

Left
is the number of unused VTOC directories.

PV Name
is the name of the physical volume.

PB/PD
contains "pb" if the logical volume is public, and "pd" if it has been designated as being available to hold the segments in process directories.

LV Name
is the name of the logical volume.

Example

The active function:

[list_vols -records_left root]

returns the number of records left on the root logical volume.

The command:

```
list_vols -tt root
```

prints the following totals for the root logical volume.

```
Records   Left  VTOCEs   Left PB/PD LV Name
  50267   4026  20185    7874 pb    root
```

The following is an example of the information printed when the list_vols command is invoked with no control arguments.

```
Drive      Records   Left   VTOCEs   Left PV Name  PB/PD LV Name

dska_06      18504   1913     3085   2332 bdm01    pb    bdm
dska_11      37469   6157     3200   1604 ldd01    pb pd ldd
dska_12      36926   6357     5865   2299 nbdd02   pb    nbdd2
dska_15      36720   8042     7650   2259 nbdd1    pb pd nbdd
dska_01      18221    863     5200    589 pub04    pb pd public
dska_02      18221   1008     5200    602 pub06    pb pd public
dska_05      18221   1009     5200    570 pub05    pb pd public
dska_07      18221    983     5200    815 pub03    pb pd public
dska_10      18221    526     5200    539 pub07    pb pd public
dska_13      36069   2135    10200   1570 pub09    pb pd public
dska_14      36029   2104    10400   1365 pub01    pb pd public
dska_16      36209   2127    10200    852 pub08    pb pd public
dska_04 P    15905   1231     5500    965 root2    pb    root
dska_08 P    17027   1362     5050    835 rpv      pb    root
dska_09      17335   1373     9635   6069 root3    pb    root
```

Name: meter_gate, mg

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

Usage

meter_gate STR {entry_name} {-control_arg}

where:

1. STR
   is the name of the gate segment to be examined; i.e., hcs_, phcs_, hphcs_, ioi_, hc_backup_, etc.

2. entry_name
   is the name of a single entry in the specified gate. Only the information for that entry is printed. If entry_name is not specified, information for all entries is printed. No control argument can be given if an entry_name is specified.

3. control_arg
   can be one of the following:

   -reset, -rs
   resets the metering interval for the invoking process so that the interval 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.

   -time, -tm
   sorts the output on the total time spent in each entry.

   -average, -av
   sorts the output on the average time spent in each entry.

   -call, -cl
   sorts the output on total calls to each entry.

   -page, -pg
   sorts the output on the average number of page faults in each entry.

Notes

If the meter_gate command is given with no control argument, it does not sort the output.

The output header consists of the time the system was brought up, the current time, and the total charge time (which equals total_cpu_time minus 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.

Metering information is collected only for gate segments defined with the "hgate" macro, and only for those entries in the segment defined with the "gate" macro (refer to the gate_macros.incl.alm include file for these macros, and refer to the source listing of a particular gate to apply this principle). For example, some hardcore gate entries are defined with the "fgate" macro for efficiency or because ring 0 stack history is abandoned during the call (e.g., hcs_$block); such gate entries are not metered.

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

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_rcp

The meter_rcp command prints information about devices controlled by the resource control package (RCP).

Usage

meter_rcp {-control_args}

where control_args can be chosen from the following:

-all, -a
displays meters for both locks and devices.

-lock
displays only meters for locks. The default is to display meter information only for devices and not for locks.

-type STR, -tp STR
displays meters only for devices of the type specified by STR. This control argument cannot be used with the -lock or -device control arguments. STR can be tape, disk, console, printer, punch, reader, or special.

-device STR, -dv STR
displays meters only for the device named STR. This control argument cannot be used with the -lock or -type control arguments.

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

-long, -lg
displays additional information about devices/locks (as selected by the other control arguments) that is not otherwise printed (e.g., for -type, an assignment histogram; for -lock, four lines of totals).

Notes

If the meter_rcp command is given with no control arguments, it prints information for all devices only (no locks).

The following is a brief description of the variables printed if the -lock control argument is specified.

% time locked
is the percentage of time spent locked.

% time waiting
is the percentage of time waiting for locks.

% number of waits
is the percentage of the number of attempts to lock that required a wait.

If the -type control argument is specified, the following variables are printed for that type only. This same information is printed for all device types if no control argument is given, and for the named device if the -device control argument is given.

Total assignments
is the number of times the device has been assigned during the metering interval.

Total errors
is the number of I/O transfer errors for the device during the metering interval.

Total time assigned
is the time (in hours, minutes, and seconds) the device has been assigned during the metering interval.

% time assigned
is the percentage of the metering interval that the device has been assigned.

Example

The following is an example of the information printed if the meter_rcp command is specified with the -lock control argument.

```
Total time metered  =    1 hours, 29 minutes, 34 seconds

Lock meters for rcp_com_seg:
% time locked       =    0 %
% time waiting      =    0 %
% number of waits   =    0 %

Lock meters for rcp_data:
% time locked       =    0 %
% time waiting      =    0 %
% number of waits   =    2 %
```

The following shows a sample result of invoking the meter_rcp command with the -type control argument specifying tape.

```
Total time metered  =    1 hours, 28 minutes, 44 seconds

Meters for device type tape:
     Meters for tape_01
          Total assignments   =    1
          Total errors        =    0
          Total time assigned =    1 hours, 23 minutes, 36 seconds
          % time assigned     =   94 %
     Meters for tape_02
          Total assignments   =    4
          Total errors        =    0
          Total time assigned =    0 hours, 52 minutes, 14 seconds
          % time assigned     =   58 %
     Meters for tape_03
          Total assignments   =    1
          Total errors        =    2
          Total time assigned =    0 hours, 3 minutes, 27 seconds
          % time assigned     =    3 %
```

Finally, if the command:

```
meter_rcp -device dska_05
```

is given, the following information is printed. Giving the command with no control arguments produces a report that would include both this and the above example's information, as well as the same type of information for all other RCP devices.

```
Total time metered  =    0 hours, 5 minutes, 37 seconds
     Meters for dska_05
          Total assignments   =    1
          Total errors        =    0
          Total time assigned =    0 hours, 5 minutes, 37 seconds
          % time assigned     =   99 %
```

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 values for all zerodivide faults caused in an invocation are computed.

Usage

meter_signal {-control_args}

where control_args can be chosen from the following and can be specified in any order. Each control argument must include a decimal value (N).

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

-nframes N
specifies the number of stack frames to be established between the frame containing the zerodivide handler and the frame that causes the fault. The fault occurs in the same frame that established the handler if the value is one; this is the default.

-nhandlers N
specifies the number of handlers for dummy conditions to be established in each stack frame. Handlers are established for the conditions meter_signal1 through meter_signalN where N is the value specified. The default is that no dummy interrupt handler is established.

-unclaimed N
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 control argument. The default is that no unclaimed_signal handler is established.

Example

The command line:

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.

page_multilevel_meters

Name: page_multilevel_meters

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 chosen from the following:

-brief, -bf
produces an abbreviated report, not printing those variables marked below with a plus sign (+).

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

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

page_multilevel_meters

Notes

If the page_multilevel_meters command is given with no control arguments, it prints a full report.

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

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.

ATB
: is the average time between page moves.

Core blocks needed
: is the number of requests by the various parts of page control for a frame of main memory.

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).

PD desperations
: is the number of times an attempt is made, at page write time, to free a paging device frame. (This occurs if the pool of free-paging device frames is empty.)

page_multilevel_meters

% 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 are 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 moves 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.

+ 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 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 were 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.).

+ Overhead
is the total of start time and done time (above).

+ 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.

page_multilevel_meters

+ Skips (incore)
  is the number of times a paging device map entry for a page in main memory was encountered while searching for a record that could be freed.

+ RWS performed
  is the number of read/write sequences performed.

+ 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

The following is an example of the information printed when the page_multilevel_meters command is invoked with no control arguments.

```
Total metering time    0:34:03

PD records                1019
Pages moved to PD        11244
ATB                        .18 sec.
Core blocks needed       58430
New pages                 6567
Page faults from PD      27170
PD desperations            133
% 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.
```

post_purge_meters

Name: post_purge_meters

The post_purge_meters command displays information collected at post purge time, if post purging is enabled. The print_tuning_parameters and work_class_meters commands (described later in this section) are used to determine which work classes, if any, are being post purged.

Usage

post_purge_meters {-control_arg}

where control_arg can be one of the following:

-reset, -rs
: resets the metering interval for the invoking process so that the interval 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.

Notes

If the post_purge_meters command is given with no control argument, it prints a full report.

The following is a brief description of each of the variables printed out by the post_purge_meters command.

Post purge time
: is the average CPU time per post purge call.

Ave list size
: is the average number of page fault entries found in the per-process page trace list at post purge time.

Ave working set
: is the average estimated working set. The current estimated working set for each process is computed by the following formula:

```
working set = working_set_factor + raw_working_set
+ working_set_addend
```

The raw working set is estimated by page control at post purge time.

Working set factor
: is the current value of the wsf tuning parameter, and can be changed by the change_tuning_parameters command. Increasing the value tends to reduce page thrashing, but may increase multiprogramming idle. Decreasing the value has the opposite effects.

post_purge_meters

Working set addend
is the current value of the wsa tuning parameter, and can be changed by the change_tuning_parameters command. Increasing and decreasing this value has the same effects as noted above.

Thrashing percentage
is the percentage of page faults that were taken on pages faulted earlier in quantum.

Ave post in main memory
is the average number of entries in the trace list for which the page was still in main memory at post purge time, and the ratio of incore pages to faulted pages expressed as a percentage.

Example

The following is an example of the information printed when the post_purge_meters command is invoked with no control argument.

```
Total metering time   12:43:11
Post purge time         3.14 msec. (0.46% of system)
Ave list size          39.72 entries
Ave working set        15.14 pages
Working set factor      0.50
Working set addend         0
Thrashing percentage   12.07 %
Ave post in core       26.16        (65.85 %)
```

Name: print_tuning_parameters, ptp

The print_tuning_parameters command prints the current values of various tuning parameters within the system. The values of most of these tuning parameters can be changed by using the change_tuning_parameters command described earlier in this section.

Usage

print_tuning_parameters {name1 ... nameN} {-control_args}

where:

1. namei
   is the name of a tuning parameter whose value is to be printed. It can be either the long name or the short name of the parameter. If no names are supplied, all tuning parameters that can be changed while the system is running are printed.

2. control_arg can be:

   -all, -a
   if no names are specified, prints all tuning parameters, including those that are "special" and not alterable while the system is running (e.g., max_max_eligible, which can only be changed by means of a bootload).

   -long, -lg
   lists the short and long names of the parameter(s),as well as a pointer to the location of the tuning parameters in ring zero.

   -short, -sh
   prints only the long name and the value of the parameter(s) (default).

Notes

This command requires access to metering_gate_.

See the change_tuning_parameters command for explanations of the parameters.

print_tuning_parameters

Example

The following is an example of the information printed when the print_tuning_parameters command is invoked:

```
Current system tuning parameters:

tefirst                         0.5 seconds
telast                          1. seconds
timax                           8. seconds
priority_sched_inc              80. seconds
min_eligible                    2.
max_eligible                    20.
max_batch_elig                  0
working_set_factor              0.5
working_set_addend              0
deadline_mode                   off
int_q_enabled                   on
post_purge                      on
pre_empt_sample_time            0.04 seconds
gp_at_notify                    off
gp_at_ptlnotify                 off
process_initial_quantum         2. seconds
gv_integration                  4. seconds
quit_priority                   0
notify_timeout_interval         30. seconds
notify_timeout_severity         3
write_limit                     992
```

Name: response_meters

The response_meters command displays statistics on interactive response time.

Usage

response_meters {-control_args}

where control_args can be chosen from the following:

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

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

-work_class N, -wc N
prints information only for work class number N.

-total, -totals, -tt
prints summary information for all work classes.

Notes

If neither -reset nor -report_reset is specified, response_meters prints a report.

The control arguments -total and -work_class cannot be used together. If either of these is used, the remaining control arguments must be such that response_meters prints a report.

The fundamental concept needed to understand the output produced by the response_meters command is that of a terminal interaction. A terminal interaction is essentially an atomic unit of work by a Multics user at a remote terminal. A terminal interaction is triggered by a set of characters keyed by a user at a terminal, which are passed into the user ring as a unit. The activity between the receipt of the set of characters at the Multics mainframe and the next request by user-ring software for more input is a terminal interaction. For a typical user, a terminal interaction corresponds to a line of input. Some user-ring software, however, receives input as sets of characters rather than as lines (the Emacs editor is an example of such software). For software of this sort, a terminal interaction corresponds to the set of characters handed out by ring 0 TTY software at one time. If, in processing a set of characters keyed by a user at a terminal, the user-ring software blocks itself on anything other than terminal input, that activity is not considered a terminal interaction. For example, any command that results in tape I/O is not considered a terminal interaction.

response_meters

The output of response_meters is summarized by work class. Any work class for which no interactions were measured is not displayed. The following are brief descriptions of the variables printed out by response_meters for each work class and for the system:

WC
is the work class number. Work class "All" represents statistics for the entire system.

Thinks
is the number of times a process in the work class blocked itself awaiting terminal input. The term "Think" is used because the delay represented here corresponds to what is commonly called "Think Time" in models of interactive computer systems. This term is derived from the assumption that during this time the interactive user is cogitating about the response just received and is formulating an appropriate next input.

Avg Thinks
is the average time in seconds for all blocks awaiting terminal input. This is the average external delay between interactions to the same process in the work class. This external delay includes communications processing time (in the FNP), communications line delay, and user idle (or "Think") time.

Queues
is the number of times a process in the work class was queued for eligibility following receipt of terminal input. It may be less than the number of interactions because of type-ahead. That is, by keying input sufficiently rapidly, a user may cause several interactions to be processed during the same eligibility period. It may be larger than the number of Thinks because of perturbations associated with process initiation.

Avg Queues
is the average eligibility queue time (in seconds) following the receipt of terminal input. This is the average clock time between the receipt of the terminal input at the Multics mainframe and the awarding of eligibility to the target process by traffic control. The average queue time corresponds to what is called "response time" in the output of traffic_control_meters.

Load Control Group
is the set of Load Control Groups corresponding to the work class. Read access to the Answer Table and to the Master Group Table (MGT) is required for this information to be displayed. If the requesting process does not have this access, then this field is left blank.

response_meters

Response times by VCPU Range
Up to four detail lines and one summary line are printed, with each line corresponding to a set of interactions. Each interaction measured is categorized by the amount of virtual CPU time consumed by the interaction. Based on the virtual CPU time for the interaction, the interaction is recorded against one of four "buckets." Each of the detail lines represents all terminal interactions for processes in the work class that fell into the corresponding "bucket." A bucket is simply a range of virtual times. Intuitively, the first bucket represents trivial interactions, the last bucket represents exceptionally long interactions, and the remaining two buckets lie between these extremes. If there were no interactions recorded against a bucket, the detail line corresponding to that bucket is not printed. The summary line contains information on all terminal interactions for processes in the work class.

VCPU Range
From
To
is the range of virtual CPU times for interactions presented on that line, in seconds. "From" is the beginning of the range, and "To" is the end of the range. The summary line is identified by "-----" in both the "From" and the "To" fields.

Int
is the number of interactions in the bucket for the line. That is, this is the number of interactions whose virtual CPU time fell between the values of "From" and "To."

Avg VCPU
is the average virtual CPU time per interaction for the bucket.

Avg RT
is the average response time per interaction for the bucket. It is the average clock time required to process an interaction, including any eligibility queue time.

Resp Fact
is the response factor. This is defined as the ratio of the average response time per interaction to the average virtual CPU time per interaction. Intuitively, this ratio represents an "extension factor" for virtual CPU time. That is, it is a factor expressing the average amount of clock time required to process an interaction with a given virtual CPU time. This number can be used as a "quick" indicator of work class or system performance.

In addition to the per-work class data described above, the following summary information is displayed:

calls to meter_response_time
is the number of subroutine calls to the ring 0 routine that collects the raw response time data.

invalid transitions
is the number of invalid calls to this routine; meter_response_time implements a finite-state automaton that models a terminal user's interaction with the Multics system. The number of invalid transitions is the number of events that did not fit into this models. Typically, invalid transitions result from perturbations associated with process initiation.

response_meters

Overhead
is the CPU time consumed by the measurement routine, expressed as a percentage of total system CPU time, and as the average CPU time in milliseconds per call to this routine.

Example

The following is an example of the information printed when the response_meters command is invoked with no control argument.

```
Total metering time      1:11:13

WC  --Thinks/--   ----Response Times by VCPU Range---   Load Control Group
    --Queues---   -VCPU Range-    #  Avg   Avg   Resp
       #   Avg     From    To    Int  VCPU   RT   Fact

 0     3  2.77    0.00  0.50      7  0.04  0.44  10.40  Init
       4  0.01    0.50  1.00      2  0.52  3.76   7.18
                 10.00 99.99      1 11.34 65.38   5.77
                 ----- -----     10  1.27  7.60   5.99

 3  1354  4.30    0.00  0.50   1390  0.07  0.28   4.00  SysProg
    1391  0.05    0.50  1.00     34  0.71  2.67   3.78
                  1.00 10.00     24  2.53 11.22   4.43
                 10.00 99.99     13 29.15 53.24   1.83
                 ----- -----   1461  0.38  0.99   2.57

All 1357  4.29    0.00  0.50   1397  0.07  0.28   4.02
    1395  0.05    0.50  1.00     36  0.70  2.74   3.93
                  1.00 10.00     24  2.53 11.22   4.43
                 10.00 99.99     14 27.88 54.11   1.94
                 ----- -----   1471  0.39  1.03   2.65

   22556 calls to meter_response_time        50 invalid transitions.
         Overhead =   0.03% (  0.047 ms./call)
```

Name: set_work_class, swc

The set_work_class command moves a process or set of processes from one work class to another without installing a revised master group table (MGT). The effect of this command is temporary since the answering service recomputes and resets a process' work class if the shift changes, a new MGT is installed, the user issues a new_proc command, or the operator issues the "maxu auto" command (described in the Multics Operators' Handbook, Order No. AM81).

Usage

```
set_work_class wc_number {id}
```

where:

1. wc_number
   is the number of the work class to which processes are to be moved.

2. id
   may be a User_id or a process identifier. If a User_id is given it must be of the form Person.Project.tag, and any or all components may be "*". If a process identifier is given it must be an octal number. If this argument is not given, only the process executing the command is moved to the specified work class.

Notes

In order to prevent severe errors, set_work_class never matches any User_id to the Initializer process. If for some reason it is necessary to move the Initializer out of work class zero, this must be done by specifying the Initializer's process identifier.

Example

The following are examples of valid command lines:

```
swc 1 *.*.*
```
moves all processes to work class 1.

```
swc 2 *.SysDaemon.z
```
moves all processes with the SysDaemon Project_id and the tag of "z" to work class 2.

```
swc 3 *.*.m
```
moves all absentee process to work class 3.

```
swc 4
```
moves the executing process to work class 4.

system_link_meters

Name: system_link_meters

The system_link_meters command prints out system-wide statistics regarding usage of the Multics linker. Information is obtained from the active_hardcore_data and tc_data data bases.

Usage

system_link_meters {-control_arg}

where control_arg can be one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the system_link_meters command is given with no control argument, it prints a full report.

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.

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

CPU Metering time
is the amount of time for which the processor was busy. It equals total processor time minus idle time.

Total time in linker
is the total amount of CPU time spent in the linker, expressed as hh:mm:ss.

system_link_meters

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 into which calls to the linker are broken down. The four slots are for calls completed in less than 25 milliseconds, calls completed in between 25 and 50 ms, calls completed in between 50 and 75 ms, and calls completed in more than 75 ms.

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.

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: definition search, segment search, and get linkage.

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 categories.

system_link_meters

Example

The following is an example of the information printed when the system_link_meters command is invoked with no control argument.

```
Linkage Meters:

Total Metering time                          7:16:29
CPU Metering time                           12:22:38

Total time in linker                         0:12:32
Average time per link                       5.49 msec.
Percentage of real time in linker           2.88
Percentage of CPU time in linker            1.69

Time slot (msec)              <25          25-50          50-75          >75

Calls                       136483            410             16          286
Total time in slot          0:11:27        0:00:12        0:00:00      0:00:51
Percent total time            91.35           1.69           0.12         6.85
Percent total calls           99.48           0.30           0.01         0.21
Average time                   5.04          31.00          54.87       180.29
Average page faults            4.42          40.00          66.88        82.38

Segment Search
 Average time                  2.65          24.47          43.72         5.99
 Average page faults           1.22          20.81          50.33         3.01
 Percent time in slot         57.15          79.80          86.94         3.31

Get Linkage
 Average time                  0.67           4.49           5.20       173.65
 Average page faults           0.95          11.41          10.67        76.52
 Percent time in slot         14.42          14.66          10.35        95.93

Definition Search
 Average time                  0.26           0.59           0.24         0.25
 Average page faults           1.17           1.62           0.67         0.88
 Percent time in slot          5.67           1.91           0.48         0.14
```

system_performance_graph

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. Periodically, metering information is 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 switch spg_output_.

Usage

system_performance_graph 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:

   -halt, -ht
   terminates plotting.

   -output_file {path}, -of {path}
   directs output to a file called spg_output, or if a path is supplied, directs output the the file specified by path.

   -short
   compresses the width of the meter display lines (see "Notes" below).

Notes

Description of the output pattern:

1. An initial line gives 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.

system_performance_graph

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 that the time given exceeds the minute represents the response time. Strictly interpreted, the time 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. When the -short control argument is given, the total width of the grid is only 50 units, so all individual components are correspondingly compressed. The "Example" section below shows the output when the command is invoked with the -short control argument. The wider display would, of course, be easier to read. The 100-unit 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 order shown below. When the superimposition is complete, the resultant line containing only the last characters superimposed is printed.

1) Time Usage Percentages

blank located to the right of y to right margin
user processing not in ring 0. (The position of y is an estimate; it is a figure that divides user processing into ring 0 and non-ring 0 sections.)

blank located to the right of s to left of y
user processing in ring 0.

s
time spent handling segment faults.

p
time spent handling page faults.

t
time spent in the traffic controller.

i
interrupt processing.

multiprogramming idle.

*
nonmultiprogramming idle.

blank located from the left margin to left of *'s
zero idle.

system_performance_graph

2) Other Values

The current average is determined from samples taken at one-second intervals weighted backwards in time exponentially, with a smoothing constant of 1/64. The effect is to average over roughly the last minute.

q relative to the left margin
current average of the ready list length.

e relative to the left margin
current average of the number of eligible processes.

r relative to the left margin
current average of the response time in seconds, for trivial requests.

Q relative to the left margin
average over a sample of quits/minute.

S relative to the left margin
average over a sample of schedulings/(10 seconds).

D relative to the right margin
average over a sample of disk read and write traffic in units of pages/(.1 seconds). Full scale equals 1000/sec.

P relative to the right margin
average over a sample of all read and write traffic (bulk store and disk) in units of pages/(.1 seconds). Full scale equals 1000/sec.

V relative to the right margin
average over a sample of VTOC entry read and write traffic in units of VTOCES/(.1 seconds). Full scale equals 1000 VTOCES transferred per second.

\- relative to the left margin
number of load units at the time of the sample. If this number is greater than 100, 100 is assumed.

\+ relative to the left margin
number of users at the time of the sample. If this number is greater than 100, 100 is assumed.

system_performance_graph

Example

The following example was generated by the command line:

```
spg -short 1
```

Note that the grid marks are not overwritten if the character that would appear is the same as both adjacent characters.

```
                                    12/27/78  1639.2 est Wed
up= 12/26/78  0732.1 est, sys_hours= 33.1, cpu_hours= 66.2
      cpu= 2, pages= 297, min_e= 2, max_e= 6, wsa= 0,
      wsf= 0.50, tefirst=  1.00, telast=  0.75, timax= 32.00.
1639.30 Qqe   .    *¦*****S**+*itpppp    ¦     .     ¦   p.    V
1640.00 Qqe******S**iitppp+p¦p    .     ¦     .     ¦P   .    Vy
1641.00 Qqe********iStpppp+ps     .     ¦     .    P     .    Vy
1642.01 Qqe****iitSppppppp+ ¦     .     ¦     .  P  ¦     .    Vy
1643.00 Qqe*******Stppppppp+ ¦     .     ¦     .     ¦P   .    Vy
1644.00 Qr*****ittppSppp  + ¦     .     ¦     .    P     .    DV
1645.00 Q*e*****iiSttppppp+pps    .     ¦    .P    ¦     .    DVy
1646.00 Qe****S***¦*******+*¦*****mittpppp .     ¦    P.    Vy
1647.00 Qe****S***¦*******+*¦**********itppps     ¦     P    Vy
1648.00 Qe********¦*S*****+*¦***iiittp¦ppppp   P¦     .     Vy
1649.01 re****Q***S*******+*¦*********iitpppps    ¦   P .    Vy
1650.00 Qe********¦*S*****+i¦itppppsss¦     .     ¦   P .   VDy
1651.01 Qre*iitpppps S .  + ¦     .     ¦     .     ¦   P .    DV
1652.01 Q*e********Sittppp+pps    .     ¦     .    P     .    DVy
1653.00 Qe********¦**S****+iittppppppps¦     .     ¦P   .    Vy
1654.00 Qe********S*******+*¦********iittpppps    ¦  P   .    Vy
1655.04 *Qe*******¦**S****+*miiittpppp¦pps . P    ¦     .    DVy
1656.02 Q*reiittttppppSppp+pp     .     ¦  P  .    ¦     .    DVy
```

total_time_meters

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 one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the total_time_meters command is given with no control argument, it prints a full report.

The following are brief descriptions of each of the variables printed out by total_time_meters. Average CPU times are given in microseconds. In the description below, system CPU time is the total amount of CPU time generated by all configured CPUs. Idle time is CPU time consumed by an idle process; an idle process is given a CPU only if no other (nonidle) process can be given that CPU. System nonidle time is the difference between system CPU time and the aggregate idle time. In this computation, MP idle time and loading idle time are considered as overhead time and are included in system nonidle time.

Page Faults
is the percentage of system CPU time spent handling page faults, the percentage of system nonidle time spent handling page faults, and the average time spent per page fault.

PC Loop Locks
is the percentage of system CPU time spent looping on the page table lock, the percentage of system nonidle time spent looping on the page table lock, and the average time spent per looplocking. This number will be nonzero only on a multiprocessor system.

RWS Overhead
is the percentage of system CPU time spent in read-write sequences transferring modified pages from the paging device back to disk, the percentage of system nonidle time spent for this function, and the average time spent per read-write sequence. This number is nonzero only on a system with a paging device configured.

total_time_meters

PC Queue
is the percentage of system CPU time spent processing the core queue, the percentage of system nonidle time spent processing the core queue, and the average time spent per core queue processing. The core queue is used to prevent loop looks in page control on interrupt side. If an interrupt for a page I/O is received when the page table is locked, an entry is made into the core queue. When the page table is next unlocked, the core queue is processed.

Seg Faults
is the percentage of system CPU time spent handling segment faults, the percentage of system nonidle time spent handling segment faults, and the average time spent per segment fault. These values do not include 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 the percentage of system nonidle time spent handling bound faults, and the average time spent per bound fault. These values do not include time spent handling page faults that occurred during bound fault processing.

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

Other Fault
is the percentage of system CPU time spent handling certain other faults and the percentage of system nonidle time spent handling these faults. The fault processing time included is fault handling time that is not changed to the user process as virtual CPU time and that does not appear elsewhere in the total_time_meters output (i.e., it is not page fault, segment fault, or bound fault processing). The vast majority of the time included as Other Fault processing is related to the processing of connect faults and timer_runout faults.

Getwork
is the percentage of system CPU time spent in the getwork function of traffic control, the percentage of system nonidle time spent in this function, and the average time spent per pass through getwork. The getwork routine is used to select a process to run on a CPU and to switch address spaces to that process.

TC Loop Locks
is the percentage of system CPU time spent looping on a traffic control lock, the percentage of system nonidle time spend looping on a traffic control lock, and the average time spent per looplocking. The locks included in this category are the global traffic control lock and the individual Active Process Table Entry (APTE) locks. This time is nonzero only on a multiprocessor system.

total_time_meters

Post Purging
is the percentage of system CPU time spent in post purging processes that have lost eligibility, the percentage of system nonidle time spent in this function, and the average time spent per post purge. Post purging a process involves moving all of its per-process pages that are in main memory into the "most recently used" position in the core map and computing the working set of the process. This time is nonzero only if the "post_purge" tuning parameter is set to "on."

MP Idle
is the multiprogramming idle. This is the percentage of system CPU time and the percentage of system nonidle time that is idle time when processes are contending for eligibility, but not all contending processes are eligible. This occurs because some site-defined or system limit on eligibility has been reached--e.g., maximum number of eligible processes (tuning parameter "max_eligible"), maximum number of ring 0 stacks (tuning parameter "max_max_eligible"), per-work-class maximum number of eligible processes, working set limit, etc.

Loading Idle
is the percentage of system CPU time and the percentage of system nonidle time that is idle time when processes are contending for eligibility, not all contending processes can be made eligible, and some eligible processes are being loaded. Being loaded means wiring the two per-process pages that must be in main memory in order for a process to run--the first page of the descriptor segment (DSEG) and the first page of the process descriptor segment (PDS).

NMP Idle
Is the nonmultiprogramming idle--the percentage of system CPU time that is idle time when all processes contending for eligibility are eligible.

Zero Idle
is the percentage of system CPU time that is idle time when no process is waiting to be run.

Other Overhead
is the percentage of system CPU time that is overhead but cannot be attributed to any of the above categories of overhead. This is almost entirely instrumentation artifact, due to a small but indeterminable amount of time between the occurrence of a fault or interrupt and the reading of the system clock (which begins the charging of time to some overhead function). Due to hardware features such as cache memory and associative memory, this time is not constant per fault, even though the same instruction sequence is executed each time. Other Overhead represents the effect of this nondeterminism.

total_time_meters

Example

The following is an example of the information printed when the total_time_meters command is invoked with no control argument.

```
Total metering time   91:33:53

                        %       %NI        AVE

Page Faults           1.49     4.28     2301.466
 PC Loop Locks        0.14     0.41     3439.733
 RWS Overhead         0.00     0.00        0.000
PC Queue              0.17     0.49      306.381
Seg Faults            0.84     2.40     9628.827
Bound Faults          0.05     0.14    15850.365
Interrupts            2.66     7.61     1959.442
Other Fault           3.17     9.07
 Getwork              1.49     4.27      638.160
 TC Loop Locks        0.08     0.24      309.842
 Post Purging         0.09     0.25      790.584
MP Idle               0.20     0.58
Loading idle          0.02     0.05
NMP Idle             36.36
Zero Idle            28.72
Other Overhead        0.10     0.29
Virtual CPU Time     26.22    75.10
```

traffic_control_meters

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:

-gen
prints out general traffic control information.

-counters, ct
prints out the number and frequency of certain paths through the traffic controller.

-queue, -qu
prints out certain resource usage as a function of depth in the eligible queue.

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the traffic_control_meters command is given with no control arguments, it prints a full report.

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

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.

Response time
is the average time between a process' receiving an interactive wakeup and the awarding of eligibility to the process. The response time seen by the user is larger than this meter.

traffic_control_meters

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

Interactions
is a count of, and the average time between, terminal interactions.

Loadings
is a count of, the average time between, and the 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.

* Schedulings
is a count of, the average time between, and the number per interaction of trips through the scheduler/rescheduler function that caused priorities to be changed.

Lost priority
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 eligible time exceeded the CPU quantum that the process remains in the queue. The process reenters the traffic controller to be rescheduled.

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 waiting time before rescheduling set to 0. The process is then resorted into the ready list with its new, higher priority.

Wait Page
is a count of, the average time between, and the number per interaction of calls to force some process to a wait state in order to wait for page transfer.

Wait PTL
is a count of, the average time between, and the number per interaction of calls to force some process to a wait state in order to wait for the page table lock.

Wait Other
is a count of, the average time between, and the number per interaction of calls to force some process to a wait state in order to wait for events other than page control events.

Total Waits
is a count of, the average time between, and the number per interaction of calls to force some process to a wait state.

Notify Page
is the number of, and average time between, calls to notify processes waiting for page transfer events.

traffic_control_meters

Notify PTL
is the number of, and average time between, calls to notify processes waiting for page table unlockings.

Notify Other
is the number of, and average time between, calls to notify processes waiting for all other events.

Total Notifies
is the number of, and the average time between, notify calls (i.e., returning a waiting process to the ready state).

Get Processor
is the number of, and average time between, calls to get_processor. Get_processor is called at notify time to find a CPU on which to run the notified process. An idle process or lower priority running process is preempted.

Pre-empts
is a count of, average time between, and the number per interaction of process preemptions and timer runout faults.

Getwork
is the number of, and average time between, calls to getwork. Getwork is the dispatcher portion of the scheduler; it finds a process to run on the executing CPU.

Retry getwork
is the number of, and average time between, retries of the getwork function.

Extra notifies
is the number of, and average time between, notify calls that found no process waiting on the notified event.

Last EN event
is the last notified event for which no process was waiting.

Notify timeout
is the number of times a notify was not received by a waiting process within notify_timeout_interval (a tuning parameter). This is printed only if the count is nonzero.

Last NTO event
is the last event on which a notify timeout occurred.

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

Depth
is the depth of the process within the eligible queue. A process deep in the eligible queue is run only if processes above it cannot run.

%PF
is the percentage of page faults that occurred from processes at this depth.

TBPF
is the average time between page faults at this depth.

traffic_control_meters

%GTW
is the percentage of getwork calls being made when a member of this priority relinquishes control.

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

The following is an example of the information printed when the traffic_control_meters command is invoked with no control arguments.

```
Total metering time   12:10:33


Ave queue length     11.35
Ave eligible          5.78
Response time         0.639 sec

  COUNTER            TOTAL          ATR        #/INT

Interactions          98304        0.446 sec
Loadings             144460        0.303 sec     1.470
Blocks               131632        0.333 sec
Wakeups              148256        0.296 sec
Schedulings          150711        0.291 sec     1.533
Lost priority          5108        8.581 sec
Priority boosts           3    14611.130 sec
Wait Page            947446       46.265 msec    9.638
Wait PTL            5382614        8.144 msec   54.755
Wait Other           324631      135.025 msec    3.302
Total Waits         6654691        6.587 msec   67.695
Notify Page         1042316       42.054 msec
Notify PTL          5382614        8.144 msec
Notify Other         562303       77.953 msec
Total Notifies      6987233        6.273 msec
Get Processor       6763776        6.481 msec
Pre-empts           4801938        9.128 msec   48.848
Getwork            11561090        3.791 msec
Retry getwork         68867        0.636 sec
Extra notifies       344940        0.127 sec
 Last EN event    000000000075

 DEPTH   %PF    TRPF   %GTW    TBS   %CPU

   1     34.6   51.3   19.9    9.7   25.5
   2     26.9   66.7   20.1    9.7   25.8
   3     18.0   77.5   20.4    7.4   20.0
   4     10.9   87.4   17.5    5.9   13.7
   5      6.2  101.5   13.0    5.3    9.1
   6      3.2  124.6    8.6    5.0    5.7
   7      0.2  107.9    0.6    4.1    0.3
```

traffic_control_queue

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 {-control_arg}

where control_arg can be -all to print information about all processes. The default is to print information only for processes in ready queues.

Notes

The following items are printed out by the traffic_control_queue command.

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 it 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.

flags
are one-bit indicators in the active process table (APT) entry for the user. The following flags are printed:

E process is eligible
W Interprocess Communication (IPC) wakeup pending
S stop pending
P process being preempted
L process is loaded
D process has descriptor base register loaded
H process is a hardcore process
I process is an idle process

traffic_control_queue

The flags are preceded by a letter indicating the state of the process. The allowed states are:

e empty or unused
x running
r ready
w waiting
b blocked
s stopped
p waiting for page table lock

If the flag is followed by a parenthesized letter, the letter is the CPU tag of the processor on which that process must be run.

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

dpf
is the incremental number of page faults the process has taken since the tcq command was last called.

temax
is the value (in milliseconds) of temax of the process. Temax is the maximum amount of CPU time the process may use in the current eligibility quantum.

te
is the value (in milliseconds) of te of the process. Te is the amount of CPU time used in the current eligibility quantum.

ts
is the value (in milliseconds) of ts of the process. Ts is the amount of CPU time used since scheduling priority changed.

ti
is the value (in milliseconds) of ti of the process. Ti is the amount of CPU time used since the process interacted, or the tuning parameter timax, whichever is less.

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

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

d
is the device identifier of the device containing the page, if the process is waiting for a page. This is not currently used.

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

wc
is the number of the work class to which the process belongs.

process
i the name of the user who owns the process.

traffic_control_queue

Example

The following is an example of the information printed when the traffic_control_queue command is invoked with no control argument.

```
avq = 19, elapsed time = 108 sec, 28 active last 15 sec.
flags     dtu     dpf temax    te     ts     ti     tssc event d  ws wc process
wWLE        4     760  2097  1273      0      0    0.032 63336 0   0  3 LEJohnson
xLED        3     555  1000   162      0      0    0.000     0 0  51  3 Mullen
wWLE        2     253  2097   543      0      0    0.107 10623 0   0  3 Paradise
xWLED       0      74  2097   106      0      0    0.004     0 0   0  1 Stefanick
wLE         3     539  1000    30      0      0    0.027 61200 0  28  3 Stern

REALTIME QUEUE:

INTERACTIVE QUEUE:
rW          1     129  1000     0      0      0    2.137     0 0  27  3 Bensoussan
rW          3     670  1000     0      0      0    2.040     0 0  17  3 JRDavis
rW          5    1093  1000     0      0      0    1.645     0 0  30  1 Kolb
rW          2     303  1000     0      0      0    0.999     0 0  37  4 Cooke
rW          1      96  1000     0      0      0    0.948     0 0  28  1 Strayhorn
rW         13    2786  1000     0      0      0    0.920     0 0  75  3 Webber
rW          2     503  1000     0      0      0    0.459     0 0  24  4 Hornig
rW          3     485  1000     0      0      0    0.444     0 0  26  1 YSChiang

WORKCLASS  1 QUEUE: credits = 126 ms.
r          15    2260   750     0   2263   3258   28.436     0 0  47  1 Yip

WORKCLASS  2 QUEUE: credits = 143 ms.
rW          1     189   750     0    911      0   33.195     0 0  56  2 Roach
r          10    1665   750     0   1503   1000   21.679     0 0  44  2 RHart

WORKCLASS  3 QUEUE: credits = 1618 ms.
rW          5     694   750     0      0   1000    8.309     0 0  31  3 Chittenden
r           4     433   750     0   3759   3260   35.030     0 0  42  3 Herbst
rW          8    1672   750     0   2257   8000   23.642     0 0  50  3 Whitmore
rW          6    1311   750     0    753   8000   18.021     0 0  78  3 Seaman

WORKCLASS  4 QUEUE: credits = 152 ms.
rW          1     292   750     0   1764   1082   33.277     0 0  15  4 JSLove
```

Name: tune_work_class, twc

The tune_work_class command sets or changes the scheduling parameters for a single work class.

Usage

tune_work_class -work_class N -control_args

where:

1. -work_class N, -wc N
   specifies the work class for which scheduling parameters are to be set.

2. control_args
   are the parameters to be set, and can be chosen from the following (at least one must be specified):

   -governed STR, -gv STR
   controls the limitation of CPU resources to the work class. STR can be "off," in which case there is no limitation for the work class; or STR can be a number between one and 100, which represents a percentage of total system CPU time. In this case, the work class is limited to the specified percentage of total system CPU time.

   -int_response N, -ir N
   is the desired response time, in decimal seconds, after an interaction.

   -int_quantum N, -iq N
   is the quantum (time slice), in decimal seconds, given after an interaction.

   -init_queue STR
   controls the use of the interactive scheduler queue by users in the work class. STR can be "on", in which case users in the work class who have interacted recently are given priority over users in all work classes who have not interacted recently. STR can also be "off", in which case users in the work class who have interacted recently do not receive priority. The default is "off" for governed work classes and "on" for all other work classes.

   -response N, -r N
   is the time, in decimal seconds, between successive quanta.

   -quantum N, -q N
   is the quantum, in decimal seconds, given when an interaction has not just occurred.

-pin_weight N, -pw N
sets the pin weight of the work class to N. The default is 3 for the Initializer, and 0 for all other work classes.

-post_purge STR, -pp STR
controls post purging of processes in the work class, where STR can be "on" or "off." If on, processes are post purged if post purging is enabled for the system; if off, processes are never post purged.

-realtime STR, -realt STR
places the work class in realtime mode if STR is "on"; removes the work class from realtime mode if STR is "off."

-wc_max_eligible N
applies eligibility constraints to processes in the work class, where N is an integer. If N is nonzero, no more than N processes are eligible at one time; if N is zero, only system-wide eligibility constraints are applied.

Notes

If the system scheduler is in percent mode and the specified work class is not in realtime mode, then the values of int_response, int_quantum, response, quantum, and wc_max_eligible have no effect on the system's operation.

If the system scheduler is in deadline mode or the specified work class is in realtime mode, then the values of governed have no effect on the system's operation.

This command is useful for setting scheduler parameters on a temporary basis. Parameters set by this command are overridden by the values in the master group table (MGT) at shift change time, if a new MGT is installed, or if the operator issues the command line "maxu auto" (the maxu command is described in the Multics Operator's Handbook, Order No. AM81).

Name: vtoc_buffer_meters

The vtoc_buffer_meters command provides information about the utilization of volume table of contents (VTOC) buffers.

Usage

vtoc_buffer_meters {-control_arg}

where control_arg can be one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the vtoc_buffer_meters command is given with no control argument, it prints a full report.

Three buffers are needed to hold a complete VTOC entry, but most operations do not require all three parts of a VTOC entry. The first part contains general information needed to activate segments and the device addresses of the first 96 pages of the segment. The second part contains the next 128 device addresses. The third part contains the last 32 device addresses and trailer information that is not needed for activation.

The following are brief descriptions of the metering variables printed out by the vtoc_buffer_meters command.

buffers
is the number of VTOC buffers allocated at system bootload time. The default number is 30, but this may be overridden by a config card.

In the following, the first three variables are column headings that give the distribution of requests made as a function of which VTOC parts were involved.

Pattern
is a bit string specifying the VTOC parts involved. For each bit that is on (numeral 1), the corresponding part of the VTOC is involved (e.g., 011 means parts 2 and 3 are involved). Activation of a segment requires reading part 1 to determine the current length, then reading parts 2 and 3, if necessary.

Read
is the number of reads for the given pattern.

Write
is the number of writes for the given pattern.

Calls to get
is the number of calls to read parts of a VTOC entry specified by the caller (used, for example, by segment activation).

Calls to put
is the number of calls to write parts of a VTOC entry specified by the caller (used, for example, by segment deactivation).

Calls to alloc
is the number of calls to allocate (or use) a free VTOC (used, for example, by segment creation). A read of pattern 100 and a write of pattern 111 are generated.

Calls to free
is the number of calls to release a VTOC to the free list (used, for example, by segment deletion). A write of pattern 100 is generated.

Calls to await
is the number of calls to wait for all I/O transfers for a given VTOC to complete. It is called after a call to put if the caller wants to wait for the I/O transfer to complete.

Loops in alloc
Loops in GET_READ
Loops in GET_WRITE
Loops in WAIT
Loops in SELECT
are internal meters of various internal procedures of the VTOC manager, vtoc_man.pl1. They are incremented when the primary operation performed must be retried because of potential asynchronous changes in the states of the VTOC buffers.

Number disk reads
is the actual number of read calls made by vtoc_man to disk I/O software.

Number disk writes
is the actual number of write calls made by vtoc_man to disk I/O software.

Number wait os
is the number of times a process called pxss$wait to wait for a VTOC I/O operation to complete. All disk reads and most calls to await result in waiting for an I/O transfer to complete. This is done by waiting for an out-of-service (os) bit to be turned off.

vtoc_buffer_meters

Example

The following is an example of the information printed when the vtoc_buffer_meters command is invoked with no control argument.

```
Total metering time:  8:40:25

30 buffers


Pattern         Read          Write
    000            0              0
    001         2914           5534
    010          538              0
    011          187              0
    100        95412         103478
    101         8860              0
    110            0             16
    111            1          12167

Calls to get                  96406
Calls to put                  99629
Calls to alloc                11477
Calls to free                 10089
Calls to await                22718


Loops in alloc                   29
Loops in GET_READ                 0
Loops in GET_WRITE                0
Loops in WAIT                 27987
Loops in SELECT                   0

Number disk reads             79710
Number disk writes           145545
Number wait os               105550
```

work_class_meters

Name: work_class_meters, wcm

The work_class_meters command prints certain information from the tc_data segment about each work class currently defined.

Usage

work_class_meters {-control_arg}

where control_arg can be one of the following:

-reset, -rs
resets the metering interval for the invoking process so that the interval 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.

Notes

If the work_class_meters command is given with no control argument, it prints a full report.

When the scheduler is operating in percent mode, percentages are computed against two different base CPU quantities. It is necessary to understand the differences between these quantities in order to interpret the output of work_class_meters.

One base quantity is the total system CPU time. This is simply the total realtime all CPUs have been active doing anything (including running an idle process). In any interval of time when there was no reconfiguration of CPUs, the total system CPU time is the product of the length of the interval and the number of CPUs. Another base quantity is nonidle CPU time. This is the total CPU time expended by all CPUs except when running an idle process. It is given by the total system CPU time minus the sum of all idle time reported by total_time_meters (MP Idle, Non-MP Idle, and Zero Idle).

When the scheduler is operating in percent mode, it distributes CPU resources among contending work classes according to their guaranteed percentages. These percentages are percentages of total nonidle CPU time. So, if there are two work classes, each with a guarantee of 50 percent, and the system is 50 percent idle, each work class gets 25 percent of total system CPU time (assuming that there is enough demand for this to be possible). In this example, each work class is getting 50 percent of the nonidle CPU time, but only 25 percent of the total system CPU time. Another way of viewing this is that the guaranteed percentages define a relationship among work classes according to the ratio of percentages. That is, a work class with a guaranteed percentage of 10 percent gets about half as much CPU time as a work class with a guaranteed percentage of

work_class_meters

20 percent, assuming sufficient demand by both. Further, this ratio is independent of the system load.

The system administrator can limit the CPU resources consumed by a work class to a fixed percentage of the total system CPU time. The scheduler enforces this limitation, even at the expense of going idle. That is, a work class with a maximum percentage of 10 percent gets no more than 10 percent of the total CPU time in any interval, regardless of load. Excess CPU time is distributed among work classes with no maximum percentage, according to their guaranteed percentages. If this cannot be done, the excess CPU time becomes idle time.

At any time one or more work classes may be a realtime work class with specified response time and quanta. A process in such a work class is low priority until its deadline arrives, at which time it is made eligible regardless of any other constraints. The remainder of the work classes are scheduled either by percentage of CPU time (percentage mode) or by soft deadlines (deadline mode).

The following parameters are always displayed for each work class.

WC
: is the number of the work class.

%GUAR
: is the percentage of nonidle CPU time guaranteed to the work class if the scheduler is being operated in percent mode and if there is sufficient demand by the work class for this to be possible.

%MAX
: is the maximum percentage of total CPU time allowed by the system administrator to be consumed by this work class. This field is blank if the work class has no limitation on CPU consumption.

%TCPU
: is the percentage of total CPU time actually received by this work class in the metering interval.

V/ELIG
: is the average amount of CPU time used per eligibility quantum.

PW
: is the pin weight, or number of free laps for pages brought into memory.

The following parameters are always displayed for realtime work classes, and are displayed for other work classes only if the scheduler is operating in deadline mode.

IRESP
: is the response time (in seconds) specified for the work class after an interaction.

IQUANT
: is the initial quantum (in seconds) for the work class after an interaction.

RESP
: is the specified delay (in seconds) between subsequent quanta.

work_class_meters

QUANT
is the value (in seconds) of subsequent quanta.

The following parameters are displayed when the scheduler is operating in either deadline or percentage mode.

P
if printed, members of the work class are post purged.

M
is the max_eligible limit per work class. A zero means the work class has no particular limit.

R
if printed, the members of the work class are scheduled in realtime mode. They are made eligible at or before their deadlines.

I
if printed, members of the work class are given scheduler priority after interactions.

LCG
are the load control groups that are placed in the work class. If the LCG name is parenthesized, only the absentee processes in the LCG are placed in the work class.

Example

The following is an example of the information printed when the work_class_meters command is invoked with no control argument. The scheduler is operating in percentage mode.

```
Total metering time     1:49:41

WC  %GUAR  %MAX  %TCP  V/ELIG  PW  IRESP  IQUANT  RESP  QUANT  P  M  R  I LCG

 0                3.    0.30   3  0.26    2.10   0.26   2.10  P  0  R  I Init
 1                2.    0.08   0  0.25    0.75   0.50   1.00  P  0  R  I RTime
 2     8.        10.    0.06   0                              P  0     I SysAd
 3    36.        59.    0.53   0                              P  0     I SysEn
 4     9.   14.   7.    0.23   0                              P  0       SEngr
 5     6.   12.   3.    0.39   0                              P  0       HEng
 6    13.   20.   6.    0.36   0                              P  0       MktUS
                                                                        MktFo
 7     3.    5.   3.    0.41   0                              P  0       DS-CC
 8    12.        2.    0.22   0                              P  0     I OffAu
 9     5.    8.   5.    0.74   0                              P  0       MiscM
10     3.         1.    0.23   0                              P  0     I Other
11     5.         1.    0.42   0                              P  0     I Spec.

TCPU percents (%GUAR) control non-realtime work_classes.
```

SECTION 5

SUBROUTINES

This section contains descriptions of subroutines used in metering the system. Sample declare and call lines are given in the "Usage" sections; the "Notes" provide additional information where required.

Name: meter_gate_

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

Usage

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

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

where:

1. gate_name (Input)
   is the name of the gate whose entries are to be metered.

2. array_ptr (Input)
   is a pointer to an array described in "Notes" below.

3. code (Output)
   is a standard status code.

Notes

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

```
dcl 1 arg_array aligned based (array_ptr),
    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 CPU time in (microseconds) used by that entry.

The meter_util_ subroutine is obsolete and should not be used. It has been replaced by the metering_util_ subroutine, described on the following pages. ✱

Name: metering_util_

The metering_util_ subroutine contains several entry points useful for collecting hardcore metering data. In general, hardcore metering data elements can be categorized as samples, cumulative times, or cumulative counts (the latter two being cumulative since system initialization). Samples are snapshots of variables that describe the state of some system object (e.g, number of processes eligible at this instant). An example of a cumulative count is the total number of read I/Os issued to a particular disk device since system initialization, while an example of a cumulative time is the total busy time of a particular disk device while processing read I/Os. It is easy to compute average I/O time for a read to a particular device from these last two items. If a given set of metering data is sampled periodically, then more interesting, time-varying data can be computed. For example, the average I/O time for a read during a certain time interval can be computed. That interval is the time between any two samples of the data; subtracting the earlier cumulative count of I/Os from the later count yields the incremental count (i.e., the count of I/Os during the interval).

Multics metering commands are designed for interactive use, with the interval boundaries defined by the user in real time. Typically, metering commands support the following control arguments:

-report
prints a report of activity since the last interval boundary (or since system initialization, if no boundary has been defined).

-reset
defines an interval boundary for this metering program; all further invocations of this command display data reflecting activity since this boundary.

-report_reset
reports and then resets.

Under this scheme, each display of data, establishment of an interval boundary, etc., is done in a separate invocation of the same metering program. This allows the user to establish an interval boundary, exercise the system in some fashion, and then print data describing the system performance while it was being exercised. Additionally, a user can run any number of metering programs, each with independent interval boundaries. These considerations imply that metering data collection (which is sampling of hardcore data bases) should be global to the process (in order to exist through multiple invocations of the same metering command) and be distinguished among different metering programs.

The metering_util_ subroutine satisfies the above requirements in the following manner. On the first invocation of a metering program, the program calls metering_util_$define_region to define the hardcore data of interest; the collection of such data can be an arbitrary number of contiguous regions in an arbitrary number of hardcore data bases. On this first invocation, metering_util_ allocates sufficient static storage to maintain two copies of each hardcore region. This storage is allocated in a system free area in the process directory. A unique identifier in the form of a nonzero integer is assigned and returned to the invoker. This unique identifier is used in all further communications with metering_util_ by that metering program to identify the set of hardcore regions defined in this first call. Current buffers are filled by calls to metering_util_$fill_buffers, at which time the hardcore regions specified in the call to metering_util_$define_region are copied into the corresponding current buffers. Previous buffers are initially set to binary zeros. On calls to metering_util_$reset, the current buffers are copied into the previous buffers. On calls to metering_util_$fill_buffers, pointers to the current and previous buffers for each hardcore region are returned.

To use this subroutine, sufficient access to copy all hardcore regions specified is required. Access to the phcs_ gate is sufficient. If all hardcore regions specified are defined in >sl1>ring_zero_meters_limits.ascii, then access to metering_gate_ is sufficient.

Entry: metering_util_$define_regions

This entry is used to define a set of sections of hardcore data bases which are of interest to the invoker. Upon return, sufficient static storage has been allocated to contain two copies of each hardcore region specified in the call; this storage has also been initialized to zero.

Usage

```
declare code fixed bin (35);
declare unique_index fixed bin;
declare metering_util_$define_regions entry options (variable);

call metering_util_$define_regions (unique_index, code,
     hardcore_seg_1, begin_region_1, end_region_1,
     ...           , ...           , ...
     hardcore_seg_n, begin_region_n, end_region_n);
```

where:

1. unique_index (Output)
   is a unique identifier for the set of regions. This identifier must be used in calls to other metering_util_ entry points.

2. code (Output)
   is a standard status code. The code error_table_$wrong_no_of_args is returned if the number of arguments remaining is not modulo 3.

The remaining arguments must be in groups of three, as shown in the calling sequence above. Each such group defines a hardcore region by specifying a hardcore segment and a contiguous region within the segment. The arguments in each group, in order, are the following:

hardcore_seg_i (Input)
identifies the ring 0 data base. It may be of the form char (*), in which case it is assumed to be the name of a ring 0 segment; or of the form ptr aligned, in which case it is assumed to be a pointer to the segment. In the latter case, only the segment number is significant.

begin_region_i (Input)
identifies the beginning of the region in the ring 0 data base. It may be of the form char (*), in which case it is assumed to be the name of an external symbol in hardcore_seg_i; or of the form fixed bin, in which case it is assumed to be a word offset into hardcore_seg_i.

end_region_i (Input)
identifies the end of the region in the ring 0 data base. It may be of the form char (*), in which case it is assumed to be the name of an external symbol in hardcore_seg_i that refers to the next word beyond the end of the region; or of the form fixed bin, in which case it is assumed to be the length of the region in words.

Notes

Any errors encountered by this entry point are reported to the user by means of the sub_err_ subroutine. Examples of such errors are invalid segment names or symbol names, or invalid region specification (e.g., nonpositive length). Errors of this sort are always programming errors, and are not external circumstances from which the calling program can be expected to recover.

Entry: metering_util_$fill_buffers

This entry is used to copy the current contents of all regions defined for the specified unique identifier into the current buffers for that unique identifier, and to return pointers to the current and previous buffers for these regions.

Usage

```
declare metering_util_$fill_buffers entry (fixed bin, fixed bin(71),
     char(10), (*) ptr, (*) ptr, fixed bin(35));

call metering_util_$fill_buffers (unique_index, meter_time, formatted_time,
     current_ptrs, previous_ptrs, code);
```

where:

1. unique_index (Input)
   is the unique identifier returned by metering_util_$define_regions (above).

2. meter_time (Output)
   is the total metering time in microseconds. The total metering time is defined as the time between the last call to metering_util_$reset and this call. If metering_util_$reset has not been called, the total metering time is defined as the time between the last system bootload and this call.

3. formatted_time (Output)
   is the total metering time in a format suitable for printing. This format is
   HHHH:MM:SS
   where this represents the decomposition of total metering time into hours (HH), minutes (MM), and seconds (SS).

4. current_ptrs (Output)
   is an array of pointers that, on return, contain pointers to the current buffers for the hardcore regions defined in the call to metering_util_$define_regions. The number of elements in this array must be equal to the number of hardcore regions defined in the call to metering_util$define regions. The elements of this array are pointers to the current buffers for the corresponding hardcore regions. Specifically, current_ptrs (i) contains on return a pointer to the current buffer for hardcore_seg_i (defined above).

5. previous_ptrs (Output)
   is an array of pointers which, on return, contain pointers to the previous buffers for the hardcore regions defined in the call to metering_util_$define_regions. The number of elements in this array must be equal to the number of hardcore regions defined in the call to metering_util_$define regions. The elements of this array are pointers to the previous buffers for the corresponding hardcore regions. Specifically, previous_ptrs (i) contains on return a pointer to the previous buffer for hardcore_seg_i (defined above).

6. code (Output)
   is a standard status code. If either the array current_ptrs or the array previous_ptrs does not have the proper number of elements (see above), the code error_table_$invalid_array_size is returned, and no action is performed.

Entry: metering_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 previous buffers for all regions defined for the unique index specified.

Usage

```
declare metering_util_$reset entry (fixed bin, fixed bin(35));

call metering_util_$reset (unique_index, code);
```

where:

1. unique_index (Input)
   is as above.

2. code (Output)
   is as above.

Name: print_gen_info_

The print_gen_info_ subroutine is used to print out general information about an object segment. The information is the same as that printed in a listing created when a system tape is generated.

Usage

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

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

where:

1. p (Input)
   is a pointer to the object segment of interest.

2. bc (Input)
   is the bit count of the object segment.

3. switch (Input)
   is the name of the I/O switch over which the information is to be output.

4. code (Output)
   is a standard status code.

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, bc, switch, code, name);
```

where p, bc, switch, and code are as described above, and name (Input) is the name of the component within the bound segment for which information is to be printed.

Name: ring_zero_peek_

The ring_zero_peek_ subroutine is used to extract data from the hardcore supervisor. Data that is not generally available is returned only to privileged users.

Access to the phcs_ gate allows this subroutine to be used to copy any portion of any hardcore segment. Access to the metering_gate_ gate allows only certain portions or certain hardcore segments to be copied. The segments that can be copied via metering_gate_ are defined in >sl1>ring_zero_meter_limits.ascii.

Usage

```
declare ring_zero_peek_ entry (ptr, ptr, fixed!bin(18), fixed!bin(35));

call ring_zero_peek_ (ptr0, ptr_user, nwords, code);
```

where:

1. ptr0 (Input)
   is a pointer to the data in ring 0 that is to be copied out.

2. ptr_user (Input)
   is a pointer to the region in the user's address space where the data is to be copied.

3. nwords (Input)
   is the number of words to be copied.

4. code (Output)
   is a standard status code that is nonzero if the user did not have access to the requested data.

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_. To use this subroutine, access to either the phcs_ or the metering_gate_ gate is required.

Usage

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

call spg_ring_0_info_ (time_rz);
```

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

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 the system_performance_graph command, described in section 4. To use this subroutine, access to either the phcs_ or the metering_gate_ gate is required.

Usage

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

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

where:

1. pzi (Output)
   is the percentage of zero idle time.

2. pnmpi (Output)
   is the percentage of nonmultiprogramming idle time.

3. pmpi (Output)
   is the percentage of multiprogramming idle time.

4. pint (Output)
   is the percentage of time in interrupts.

5. ptc (Output)
   is the percentage of time in the traffic controller.

6. ppf (Output)
   is the percentage of time in page control.

7. psf (Output)
   is the percentage of time in segment control.

8. puse_rz (Output)
   is the percentage of time executing nonsupervisor code spent in ring zero.

9. px (Output)
   is no longer used. A value of 0.0 is returned.

10. string (Output)
    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.

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

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

spg_util_

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.

system_performance_graph$line

Name: system_performance_graph$line

The system_performance_graph$line subroutine is called each time a wakeup is received to print a line of metering for the system_performance_graph command (described in Section 4).

Usage

```
declare system_performance_graph$line entry;

call system_performance_graph$line;
```

There are no arguments.

MISCELLANEOUS

A

B

C

D

F

G

I

L

M

P

Q

R

S

T

U

V

W

Z

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE: LEVEL 68
MULTICS SYSTEM METERING
ADDENDUM A

ORDER NO. AN52-01A

DATED JULY 1981

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE—
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

CUT ALONG LINE

FOLD ALONG LINE

FOLD ALONG LINE

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE MULTICS SYSTEM METERING
ADDENDUM B

ORDER NO. AN52-01B

DATED NOVEMBER 1982

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

Together, we can find the answers.

Honeywell

Honeywell Information Systems
U.S.A.: 200 Smith St., MS 486, Waltham, MA 02154
Canada: 155 Gordon Baker Rd., Willowdale, ON M2H 3N7
U.K.: Great West Rd., Brentford, Middlesex TW8 9DH **Italy:** 32 Via Pirelli, 20124 Milano
Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F. **Japan:** 2-2 Jinbou-Cho Kanda, Chiyoda-Ku Tokyo
Australia: 124 Walker St., North Sydney, N.S.W. 2060 **S.E. Asia:** Mandarin Plaza, Tsimshatsui East, H.K.

33730, 1C182, Printed in U.S.A.

AN52-01