MULTICS

PROJECT ADMINISTRATOR'S MANUAL

SUBJECT

Information Needed by Project Administrators for the Management of Multics Projects

SPECIAL INSTRUCTIONS

This publication supersedes the Multics Administrator's Manual – Project, Order No. AK51-01, dated August 1976 and its associated addenda: Addendum A, dated April 1978, Addendum B, dated February 1979, Addendum C, dated September 1979, Addendum D, dated June 1981, Addendum E, dated July 1982, and Addendum F, dated February 1983. Effective with this edition the document is retitled Multics Project Administrator's Manual.

Change bars indicate new and changed information; asterisks denote deletions.

See the "Significant Changes" section in the Preface for a description of changed information.

SOFTWARE SUPPORTED

Multics Software Release 11.0

ORDER NUMBER

AK51-02

February 1985

Honeywell

PREFACE

Depending upon site policy, there can be four kinds of administrators who manage Multics system administration facilities: the project administrator, the registration and accounting administrator (referred to as the accounting administrator), the system security administrator, and the system administrator.

The project administrator manages Multics projects by controlling the resources allocated to the project and maintaining user accounts on the project.

This manual describes those aspects of system administration that are project specific. This includes:

- A complete description of the Project Master File (PMF) which contains the list of persons who can log in on a project and project specific "keywords" that assign to a project various system resources based upon the project's needs
- The procedures for petitioning the system administrator to register a new user on the project, and the procedures required to convert and install the PMF into its system usable binary version the Project Definition Table (PDT)
- The procedures required to delete a user from a project, or delete a project.
- Resource control limits which allows the project administrator to control the overall project budget
- Load control which controls a project's access to the system

File No.: 1L13, 1U13 AK51-02

- Notes on tailoring a user environment, allocation of volume quota, and control of master directories.
- All of the commands required by a project administrator.

Significant Changes in AK51-02

An AIM authorization range can now be specified in the PMF for a project.

CONTENTS

SECTION 1

INTRODUCTION

SYSTEM ADMINISTRATION

The overall function of Multics system administration is to provide an operating environment, control usage within that environment, and account for the use of system resources. The system administrator, the system security administrator, the accounting administrator, and the project administrator carry out the tasks necessary to system administration. The responsibilities of the project administrator are described in detail in this manual.

PROJECT ADMINISTRATION

A project is a set of users grouped together for resource and access control purposes. A project administrator is a registered Multics user in charge of one or more projects. He need not be registered on the project he administers. A single user may be project administrator for many projects and a single project may have up to four project administrators.

A project administrator logs in the same way as other users and, after he logs in, a process is created for him in the same way as for other users. A project administrator differs from other Multics users in that:

- A project administrator has access to certain segments to which other users do not. The most important of these are the ASCII project master file (PMF), the binary project definition table (PDT), and the project directory of the project.

- A project administrator uses special programs to manipulate the system-control data bases. (See Section 2, "Directory Structure and Data Bases.") These programs do not make privileged calls; they are ordinary PL/I programs that manipulate data in ordinary ways. However, the data for these programs are accounting records and control segments to which nonadministrative users have no access.

- The system grants certain requests for a project administrator that it does not grant for other users. In particular, the system control process installs new versions of the PDT of the project at the request of the project administrator.

A project may be "delegated"; this means that at least one user is listed in the system administrator's table (SAT) entry for the project as the project administrator. Or, a project may be "undelegated," meaning that the system administrator performs the project administration functions for a project. This manual is intended for a project administrator in charge of a project delegated to him by the system administrator.

A project administrator maintains the PMF of the project and can install a PDT. He adds and deletes users from his project. He also controls resource usage and generates usage reports showing month-to-date resource consumption for all users on his project. Section 3, "Project Master File," describes the PMF and PDT; Section 4, "Project and User Registration and Deletion," describes the creation and termination of a project and the addition and deletion of users to and from a project; and Section 5, "Resource Control," describes resource control and usage reporting.

A project administrator decides which users on his project are guaranteed access to the system at all times and which users are subject to preemption when the primary units of the load control group are full. He also determines the environment of the users registered on his project. Load control groups are described in Section 6, "Load Control and Preemption"; the man-machine interface is described in Section 7, "Tailoring the User Environment."

A project administrator may also be assigned the responsibilities of a logical volume administrator. A description of these responsibilities is in Section 8, "Logical Volume Quota Manipulation," and Section 9, "Master Directory Creation and Deletion."

GLOSSARY

accounting update
: The process of computing resource usage for each logged in user, saving it for later use by accounting routines, and logging out any user whose usage exceeds the limit specified by the project administrator. This function is performed by the answering service at an installation-specified interval (every 15 minutes, by default).

anonymous user
: A user who is not registered as an identifiable person on the Multics system. Projects that allow anonymous users have an asterisk (*) value for a Person_id keyword statement in the PMF. (See "Anonymous User Registration" in Section 4.)

answering service
: The software responsible for conducting the orderly login and logout of Multics users.

billing

The process of computing the total resource usage of each user and each project, preparing bills and other reports addressed to various administrative personnel, and then resetting the usage figures to zero. Billing is run by the accounting administrator. It is recommended that billing be run at the end of each month, on or before the last day of the month. Terminology in documentation and program output (e.g., "month-to-date charges") assumes this recommendation is followed.

crank

The absentee job that performs various accounting functions. The usage figures gathered by the accounting updates are copied, by the crank, into data bases that are used in billing. Month-to-date summaries are produced. Projects past their cutoff limits are cut off (users on these projects are no longer permitted to log in on these projects). The printing of cutoff warning messages is based on the usage figures computed daily by the crank. It is recommended that the crank be run daily near the end of operations. Terminology in documentation and program output (e.g., "daily report") assumes this recommendation is followed.

daemon

One of several system service processes that perform such tasks as process creation, backup, network control, and printing segments on the line printer.

home directory

The directory that is the working directory of a user when he first logs in to the system (also known as the initial working directory). Usually this directory has a pathname of:

```
>udd>Project_id>Person_id
```

initializer process

The first process (Initializer.SysDaemon.z) created during system initialization. Among other things, it runs the answering service, responds to operator commands, and installs system tables, such as the project's PDT.

Limited Service Subsystem (LSS)

An environment that restricts the set of commands and the amount of CPU time available to the user. The make_commands and enter_lss commands allow a project administrator to place users in an LSS without having to write a special process overseer. See Section 7 for more information.

load control group

A group of projects that share a guaranteed access to the system. See the *Multics System Administration Procedures* manual, Order No. AK50.

LSS

See Limited Service Subsystem.

logical volume

A set of physical volumes that are always mounted together. (See "Logical Volume" in Section 8.)

master directory
A directory whose segments reside on a different logical volume than the segments of its containing directory. (See Section 9, "Master Directory Creation and Deletion.")

master group table (MGT)
A segment maintained by the system administrator that contains information about the work classes and load control groups in use at the site. The work class and load control group membership of each user is determined from information in the SAT and PDT.

password
A character string of from one through eight ASCII printing characters including backspace, but excluding space and semicolon. "HELP", "help", "quit", and "?" are interpreted uniquely by the password processor and are therefore unacceptable as password specifications for an interactive login. A password is supplied by a user and known only to him and the software that controls access to the system. It is supplied with the user's Person_id at log-in time to validate the identity of the user.

PDT
See project definition table.

Person_id
A unique name assigned to each user of the system. It is usually some form of the user's name (usually his surname). The name must be from one through 20 characters in length, usually begins with a capital letter, and may not contain punctuation characters. A user has only one Person_id regardless of the number of projects on which he is registered. A password is associated with the Person_id.

PIT
See process initialization table.

PMF
See project master file.

process
A program or group of programs in execution: an address space and an execution point. Each logged-in user has his own process. (See "Process" in Section 1 of the *Multics Programmer's Reference Manual*, Order No. AG91.)

process directory
A directory containing those segments that are meaningful only during the life of a process. These segments include the stack(s), free storage, PIT, and various temporary segments.

process initialization table (PIT)
A segment containing information needed for a process to initialize itself, such as user_id, home directory, attributes, and accounting data.

project
A set of users grouped together for accounting and access control purposes.

project administrator
: A person who has the access to specify spending limits and other attributes for the users on a particular project.

project definition table (PDT)
: A compiled project master file (PMF).

project directory
: The directory inferior to the >user_dir_dir directory that usually contains the home directories for each user on the project. (See "Contents of the Project Directory" in Section 2.)

project master file (PMF)
: An ASCII segment giving the names, attributes, and account limits of the users of a particular project. It is created and modified using any Multics editor and is compiled into a project definition table (PDT) using the cv_pmf command.

Project_id
: The name assigned to a project. The name must be from one to nine characters long, must begin with a capital letter or a digit, and must be unique within the installation.

registration and accounting administrator
: A system administrator who has limited access to register users and run the billing software only.

ring
: A level of privilege at which programs may execute. Lower numbered rings are of higher privilege than higher numbered ones. The supervisor program runs in ring 0; most user programs run in ring 4. (See Section 6, "Access Control" in the *Multics Programmer's Reference Manual*, Order No. AG91 and "Rings" in Section 7 of this document.)

SAT
: See system administrator table.

secondary storage quota
: The amount of secondary storage that a project may occupy. The project administrator may subdivide this quota among directories inferior to the project directory or he may set the quota of directories inferior to the project directory to zero and allow users to charge their usage to the project directory quota. (See "Quota Limits" in Section 5.)

system administrator
: A highly privileged user who maintains system data bases, such as the system administrator table (SAT), that control when and by whom the system can be accessed. The system administrator has access to all Multics commands, has the ability to alter any operating parameter of the system, and may make emergency repairs. He is also concerned with the basic rules (and prices) for use of system resources.

system administrator table (SAT)
: A binary segment specifying the projects that use the system, the privileges delegated to these projects, and the project administrators.

system security administrator
: A system administrator whose primary responsibility is the integrity of the system and maintenance of the access control mechanisms.

system table installation
: The installation of a system table by the answering service at the request of a system, accounting, or project administrator. The install command sends a table installation request to the answering service. The answering service checks the contents of the new table for validity, then merges them with the contents of the current system copy of the table (if any), and finally replaces the current copy with the merged copy. (The current copy cannot simply be replaced by the new one, since the former contains information about the current state of the system not contained in the latter.) Changes implied by the contents of the new table take effect immediately.

time-record product (time-page product)
: The amount of time that a record is in storage. The time-record product is the basis for charging for disk storage.

user
: A person or logical entity, such as a daemon, who is registered on the Multics system and, therefore, has the ability to log in. Each user is associated with a project and is identified for access control purposes by the concatenation of his Person_id and Project_id. A person may be registered as a user on more than one project; thus one person can be two different users (since a user is identified by the combination of his Person_id and Project_id.)

User_id
: A character string representing a user or group of users. It consists of three components: Person_id.Project_id.tag. A User_id is often used as an argument to a command. Depending on the specific command, sometimes all the components are not specified (for example, the tag component is often omitted). The star convention may be used, also depending on the command being invoked. (Refer to the relevant command description in the *Multics Commands and Active Functions* manual, Order No. AG92 to see if the command in question accepts these conventions.)

volume administrator
: A user who is given "e" access to a logical volume by the owner of the logical volume and, thus, can control who can create master directories on that logical volume. (See "Logical Volume" in Section 8.)

work class
: A group of processes that are guaranteed a specified percentage of the available CPU time. (See "Work Class" in Section 6.)

SECTION 2

DIRECTORY STRUCTURE AND DATA BASES

The data bases that support the Multics system administration facilities are implemented as segments in the Multics storage system. These data bases are organized into a tree-structured directory hierarchy. This section provides a project administrator with all the information he needs to know about these data bases. However, for those interested in more complete information on these data bases, their location in the hierarchy, and their contents, refer to the *Multics System Administration Procedures* manual, Order No. AK50.

CONTENTS OF THE SYSTEM CONTROL DIRECTORY

The most important directory for system-administration programs is system_control_1. There is one important directory contained in >system_control_1 of interest to the project administrator. This directory, >system_control_1>pdt, contains a project definition table (PDT) for each registered project.

CONTENTS OF THE PROJECT DIRECTORY

A project directory, >user_dir_dir>Project_id, usually contains the home directories of all users registered on the project. The project administrator has sma permission on this directory and can, therefore, give himself access to all directories and segments inferior to it. Requests to give users other than the project administrator access to the project directory should be submitted to the system administrator. Requests to add names to the project directory should also be submitted to the system administrator. Names to be added cannot already exist in >user_dir_dir.

The project master file (PMF), described in Section 3, may exist anywhere in the storage system. However, it is usually placed in the project directory.

SECTION 3

PROJECT MASTER FILE

The list of persons who may log in on a project is contained in a binary table known as the project definition table (PDT), maintained by the system. There is one entry in the PDT for each user; the entry contains the user's attributes and resource usage information for the billing cycle. The PDT is created from an ASCII segment known as the project master file (PMF). A PMF exists for each registered project. It is the basic project-administration data base and contains the project's specification of user attributes.

The system references the PDT, not the PMF, when it decides if a user may log in. In order to make a change to the PDT, the project administrator must modify the PMF, convert the PMF into a binary copy of the PDT using the cv_pmf command, and request the system to modify its PDT according to the copy using the install command. (The cv_pmf and install commands are described in Section 10.)

FORMAT OF A PROJECT MASTER FILE

A PMF consists of lists of statements of the form:

```
<keyword>:      <value>;
```

A keyword designates the parameter and can be a global keyword, which specifies per project default parameters, or a user keyword, which specifies per user parameters. The keyword is followed by a colon (:). The value is either a character string, a pathname, a floating-point number, or a decimal integer, and is followed by a semicolon (;). The "end;" statement terminates the PMF. A sample PMF is found in "Sample Project Master File" below.

At the beginning of a PMF is a list of global keyword statements. Global keywords are spelled the same as user keywords, but begin with a capital letter. They specify default values that apply to all users listed in the PMF, unless overridden by user keywords, or by subsequent global keywords. There are global keywords for every user keyword except "personid."

Global keywords can be used between user entries as well as at the beginning of the PMF. When used in this way, the new default values apply only to subsequent user entries, since preceding user entries are processed using the previous default values.

The first global keyword in each PMF must be "Projectid." This keyword can occur only once per PMF. The value of "Projectid" is a character string beginning with a capital letter. This character string is the name of the project (Project_id). Projectid, plus the other global keywords and their values, are described in "Keywords and Their Values" below.

Following the list of global keyword statements is a user entry for every user registered on the project. The user entry is a "personid" statement followed by an optional list of other user keyword statements. This keyword statement, plus the other user statements and their values, are described in "Keywords and Their Values" below.

It should be noted that, for some PDT parameters, the system administrator can impose per-project limits on their values, or grant or deny a project permission to set their values. See "SAT Limits" below, for more details.

Keywords and Their Values

Listed below are the global and user keywords used in a PMF and their values. Global keywords begin with an uppercase letter; user keywords begin with a lowercase letter. Projectid and personid are the only required keywords. All other keywords have default values assigned by the system whenever they are omitted from a PMF. The default values are described in "Keyword Default Values" below.

Values for spending limit keywords include integer values and floating-point values. The floating-point_number value differs from the decimal_integer value only in that it (optionally) ends with a decimal point and a single-digit fraction (tenths), or a two-digit fraction (hundredths). For more information on spending limits, see Section 5, "Resource Control."

LOGIN AND LOAD CONTROL KEYWORDS

Projectid: character_string;

> where character_string is the name of the project (Project_id). This name must begin with a capital letter and must be the same as the project name in the SAT.

personid: character_string;

> where character_string is a Multics Person_id or an asterisk (*). An asterisk indicates an anonymous user. (See "Anonymous Users" in Section 4 for more information.) This keyword begins a new user entry.

password: character_string;

where character_string is the password used by anonymous users when they log in. This keyword can only be specified for an anonymous user entry and should not be confused with the password assigned to registered users. (See "Anonymous Users" in Section 4 for details.)

homedir: pathname;
Homedir: pathname;

where pathname is the absolute pathname of the user's home directory at login. If the user is to have no permanent storage in the storage system hierarchy, this pathname may have the form "[pd]>name", and the home directory is then created for the user at login as a subdirectory of his process directory.

initproc: pathname;
Initproc: pathname;

where pathname is the absolute or relative pathname of the user's process overseer procedure. If the pathname is followed by a comma and the string "direct", the overseer is not called by the initialize_process_ subroutine, but is invoked directly from ring 0 at process creation time; this usage requires very special programming in the overseer procedure, but is useful for finely tuned environments. (See Section 7, "Tailoring the User Environment," for more information.)

attributes: character_strings;
Attributes: character_strings;

where character_strings are attribute names separated by commas. The named attributes are turned on for the user(s). If the attribute name is preceded by a circumflex (^), the attribute is turned off for the user(s) even if a default attribute specifies it on. (See "Keyword Default Values" below.) Default attributes are completely replaced each time an Attributes statement is encountered. However, an attributes statement does not replace the default attributes by those specified in the statement; it assigns the user the logical OR of the default attributes with those specified in the statement. Thus, it is necessary to explicitly turn off any of the current default attributes that a particular user is not to have. Valid attributes are:

null, none
: may be used as the only value in an Attributes statement, to turn off all default attributes; illegal in an attributes statement.

nobump
: user not subject to preemption by anyone.

guaranteed_login
: user may use the -force control argument to the login.

guar
: command to bypass load control.

nopreempt
: user not subject to preemption by others in group.

nolist
: user should not be listed in whotab; others may nevertheless be able to deduce that a user having the nolist attribute is currently logged in.

dialok, dial
: user may accept dial requests.

multip, multi_login
: user may log in more than one process.

preempting, bumping
: user may preempt others in same load control group.

brief
: user is permanently in brief mode (as if -brief had been given in login command), and does not receive the messages associated with a successful login.

vinitproc, v_process_overseer
: user may specify a process overseer or outer module on the login command line; user may also replace his process overseer, outer module, or other procedures by placing a copy in his home directory, because the home directory is in the search path during process initialization for a user with this attribute.

vhomedir, v_home_dir
: user may specify home directory at login.

nostartup, no_start_up
: user may escape from using his start_up.ec.

no_secondary, no_sec
: user may not have secondary status.

no_primary, no_prime
: user may not have primary status.

op_login daemon
: user may be logged in by operator, via message coordinator.

no_warning, nowarn
: user is permanently in no_warning mode (as if -no_warning had been given in login command) and never receives system warning messages.

igroup
: user is in an individual load control group.

save_pdir
: save process directory after fatal process error; used for debugging purposes at development sites.

disconnect_ok
: user may have saved disconnected processes, i.e., user may use the -save argument to the login command. This attribute must also be on in the project's SAT entry, to give users on the project permission to use -save.

save_on_disconnect, save
: user's process is saved on disconnection, by default; relieves user from typing -save_on_disconnect at each login; can be overridden by typing -no_save_on_disconnect at login time. If this attribute is on (either individually or because of an Attributes statement) then the disconnect_ok attribute is forced on for the user (but the disconnect_ok attribute must also be on in the project's SAT entry to give users on the project permission to have saved disconnected processes). If the system administrator turns on this attribute in the project's SAT entry, then disconnected processes of all users on the project are saved by default, regardless of the setting of this attribute in the PDT.

grace: decimal_integer;
Grace: decimal_integer;
: where decimal_integer is the number of minutes after login for which the user is protected from preemption by other users.

group: character_string;
Group: character_string;
: where character_string is the name of a load control group. The group or Group statement is used in conjunction with the igroup attribute. (See "Work Class" in Section 6 for more details.)

outer_module: character_string;
Outer_module: character_string;
: where character_string is the name of the terminal device outer module of a user with an interactive process. (The statement has no effect for absentee processes.)

abs_foreground_cpu_limit: decimal_integer;
Abs_foreground_cpu_limit: decimal_integer;
: where decimal_integer is the upper CPU time limit in seconds for the user's foreground absentee jobs. A value of zero means no limit.

max_background: decimal_integer;
Max_background: decimal_integer;
: where decimal_integer is the maximum number of concurrent background processes that the user may have. A value of zero means no limit.

max_foreground: decimal_integer;
Max_foreground: decimal_integer;
: where decimal_integer is the maximum number of concurrent foreground processes that the user may have. A value of zero means no limit.

pdir_quota: decimal_integer;
Pdir_quota: decimal_integer;

where decimal_integer is the quota to be placed on the user's process directory. If the value is zero, or if no quota is specified, the system default process directory quota is placed on the user's process directory.

SPENDING LIMIT KEYWORDS

limit: floating-point_number or "open";
Limit: floating-point_number or "open";

where floating-point_number is the absolute dollar limit that the user may spend in any monthly billing period. If the user exceeds this limit, he is automatically logged out and is unable to log in until the end of the month, or until the limit is changed and a new PDT installed. If no limit value is imposed, the character_string "open" is used.

shift_limit: floating-point_numbers or "open";
Shift_limit: floating-point_numbers or "open";

where floating-point_numbers are the user's interactive resource limits, separated by commas, on shifts 1, 2, 3, 4, 5, 6, 7, and 0. Shifts not listed receive an "open" limit. Absentee and I/O charges, although counted against the total expenditure limit, are not counted against the shift limits. (Shift 0 is listed last, because most installations do not define a shift 0.)

cutoff: limit {,date {,increment}};
Cutoff: limit {,date {,increment}};

where:

limit
: is a floating-point number specifying a dollar value, or the word "open".

date
: is a date in a form acceptable to the convert_date_to_binary_ subroutine, or "now", "open", or "never".

increment
: is one of the following keywords:

never
: do not reset.

daily
: reset every day.

monthly
: reset to beginning of next month.

yearly
: reset to one year later.

cyear
: reset to beginning of next year.

fyear
: reset to July 1 of next year.

The cutoff statement is used to set a cutoff limit on the user; this is different from the monthly spending limit specified by the limit statement (above). The cutoff statement has three different interpretations, depending on whether one, two, or all three arguments are given.

If just the limit argument is given, then an absolute spending limit is imposed. When the user's spending reaches this limit, he is logged out and not allowed to log in again until a new PDT, containing a higher cutoff limit for that user, is installed. The cutoff limit is not reset by monthly billing, and is imposed in addition to the monthly limit. The word "open" causes no cutoff limit to be imposed; this is the default (in the absence of a Cutoff statement).

If just the limit and date arguments are given, then in addition to the absolute spending limit described above, a cutoff date is imposed. When that date and time arrives, the user is logged out and not allowed to log in again until a new PDT, containing a later cutoff date for that user, is installed. The word "open" causes no cutoff date to be imposed; this is the default (in the absence of a Cutoff statement).

If all three arguments are given, then the meanings of the first two arguments are changed, and a periodic spending limit is imposed. The increment argument specifies the time period after which the user's spending against this limit should be reset to zero. The limit argument specifies how much the user is allowed to spend during the time increment. The date argument specifies the date and time at which the first time period is to end. It is used only at the time the PDT is installed. Each time the cutoff date arrives, a new cutoff date is set, as specified by the increment argument, and the user's spending against the cutoff limit is reset to zero. When this third type of cutoff limit is imposed, the user is logged out as soon as his spending exceeds the cutoff limit, and he is not allowed to log in until the cutoff date arrives.

NOTE: It is possible to predate the date argument. When the user next logs in, the periodic spending limit is reset, and a new cutoff date is established based on log-in time and the specified increment.

warn_days: decimal_integer;
Warn_days: decimal_integer;

where decimal_integer is the day threshold for cutoff warning messages. When the project's account has fewer than this many days remaining until its cutoff date, the user receives a warning message to this effect at log-in time.

warn_percent: decimal_integer;
Warn_percent: decimal_integer;

where decimal_integer is a number in the range 0 through 100, specifying the percent threshold for cutoff warning messages. When the project's account has less than this percent of its funds remaining, the user receives a warning message to this effect at log-in time.

warn_dollars: floating-point_number;
Warn_dollars: floating-point_number;

where floating-point_number is the dollar threshold for cutoff warning messages. When the project's account has less than this amount remaining, the user receives a warning message to this effect at log-in time. This keyword applies to the project's account, not the individual user's spending limit.

user_warn_days: decimal_integer;
User_warn_days: decimal_integer;

where decimal_integer is the day threshold for cutoff warning messages. When the user's account has fewer than this many days remaining until absolute cutoff (cutoff increment is "never"), the user receives a warning message to this effect at log-in time.

user_warn_percent: decimal_integer;
User_warn_percent: decimal_integer;

where decimal_integer is a number in the range 0 through 100, specifying the percent threshold for cutoff warning messages. When the user's account has less than this percent of funds remaining, the user receives a warning message to this effect at log-in time. This warning may be triggered by any of the specified fields, limit, shift_limit, or cutoff.

user_warn_dollars: floating_point_number;
User_warn_dollars: floating_point_number;

where floating_point_number is the dollar threshold for user spending warning messages. When the user's account has less than this amount remaining, the user receives a warning message to this effect at log-in time. This warning may be triggered by any of the specified fields, limit, shift_limit, or cutoff.

SPECIAL ENVIRONMENT KEYWORDS

subsystem: pathname;
Subsystem: pathname;

where pathname is the pathname of a Multics subsystem. (See "Limited Service Subsystems" in Section 7 for more information.)

ring: decimal_integer1, decimal_integer2 {,decimal_integer3};
Ring: decimal_integer1, decimal_integer2 {,decimal_integer3};

where decimal_integer1 and decimal_integer2 define the minimum and maximum values for the ring in which the user can create a process. decimal_integer3 specifies the default initial ring at login. All values must be in the range 1 through 7. decimal_integer1 must be less than or equal to decimal_integer2. decimal_integer3 must lie in the range defined by decimal_integer1 and decimal_integer2. If the value of decimal_integer3 is not specified, decimal_integer1 is used.

authorization: aim_range;
Authorization: aim_range;

This statement specifies the range of access authorizations at which the user may log in. An access authorization range is of the form:

```
low_auth:high_auth
```

If commas are used (to specify categories), or spaces (to improve readability) the entire range must be enclosed in quotation marks.

If neither an Authorization or any authorization statements are present in the PMF, the default authorization is the minimum login authorization allowed to the project by the System Administrator.

lot_size: decimal_integer;
Lot_size: decimal_integer;

where decimal_integer is the size of the user's linkage offset table (LOT). The LOT is a per-ring array of pointers that point to the linkage information for each procedure segment known in the given ring.

kst_size: decimal_integer;
Kst_size: decimal_integer;

where decimal_integer is the number of segment numbers that are allocated for the user's process.

cls_size: decimal_integer;
Cls_size: decimal_integer;

where decimal_integer is the size of the user's initial combined linkage region.

Keyword Default Values

The cv_pmf command supplies default values for any global keywords not listed in a PMF. These default values can be overridden by user keyword statements. The global keywords and their default values are:

```
Homedir:                    >user_dir_dir>Project_id>Person_id;
Initproc:                   process_overseer_;
Attributes:                 none;
Grace:                      (project max);
Group:                      (project default);
Outer_module:               tty_;
Limit:                      open;
Shift_limit:                open,open,open,open,open,open,open,open;
Cutoff:                     open, open, never;
Warn_days:                  10;
Warn_percent:               10;
Warn_dollars:               10;
User_warn_days:             10;
User_warn_percent:          10;
User_warn_dollars:          10;
Subsystem:                  none;
Ring:                       (project min), (project max);
Authorization:              "system_low";
Lot_size:                   (system default);
Kst_size:                   (system default);
Cls_size:                   (system default);
Abs_foreground_cpu_limit:   0;
Max_background:             0;
Max_foreground:             0;
Pdir_quota:                 0;
```

The user attributes statement either turns on attributes in addition to those turned on with the global Attributes statement or, for those attributes preceded by a circumflex (^), turns off attributes turned on with the global Attributes statement.

The system provides a number known as the maximum grace for each project. An attempt to set the grace to larger than this number, or to leave the grace unspecified for a user, results in the project maximum grace being given to the user. Currently, this maximum is 2880 minutes (48 hours) for all projects.

Similarly, the system maintains default values for a lowest initial ring and a highest maximum ring. The system default values are currently (4, 5). That is, no user may begin in any ring numbered less than 4, regardless of the ring brackets on his process overseer procedure. (He may make a call to a lower ring through an appropriate gate.) Since the maximum is 5, the user may execute programs in ring 4 or ring 5, either initially or by making a later outward call. Rings 6 and 7 are not normally used, since a user executing programs in these rings would have no access to any of the normal system library routines, including the operator segment for PL/I and the gates into the supervisor. If a project administrator attempts to violate the project ring limits, the system installs the PDT with a warning, and enforces the project ring limits at login time.

SAT Limits

Several of the PDT parameters whose values are set by the above keywords have corresponding parameters in the project's SAT entry (which is under the control of the system administrator. For most of these parameters, the value in the SAT entry imposes a limit on the values in the PDT. Violations of these limits are noted at PDT installation time, with a warning message, but the PDT is installed. If the SAT limits are not subsequently raised, then at login time the limits are enforced, a warning message is placed in the answering service log, and the user is logged in. (Note, however, that if a user violates a limit imposed by PDT or SAT parameters, by means of a control argument to the login command, then an error message is printed and the user is not logged in.)

The PDT parameters that have corresponding SAT parameters are listed below, along with the action taken for each pair of parameters when the PDT and SAT values are in disagreement.

abs_foreground_cpu_limit
: The smaller of the two values is used as the limit; except that a value of zero means no limit, so if a value is zero the other one is used, and if both are zero there is no limit.

attributes
: Except for the attributes listed below, each attribute is on for a user only if it is on in both the user's PDT entry and the project's SAT entry.

guaranteed_login
: On at login time if on in project's SAT entry and on in user's PDT entry and user gives -force control argument in the login command.

anonymous
: The anonymous attribute is set only in a project's SAT entry, to give the project permission to have anonymous users. Thus it does not appear in the above list of PDT attributes.

preempting
: On at login time if on in project's SAT entry and on in user's PDT entry and user does not give the -no_preempt control argument in the login command.

brief
: On at login time if on in project's SAT entry and on in user's PDT entry, or if user gives -brief control argument in login command (permission is not needed to use -brief).

nostartup
: On at login time if on in project's SAT entry and on in user's PDT entry and user gives -no_startup control argument in login command.

no_warning
: On at login time if on in project's SAT entry and on in user's PDT entry, or if user gives -no_warning control argument in login command (permission is not needed to use -no_warning)

save_on_disconnect
: On at login time if on in project's SAT entry, or if on in user's PDT entry, or if user gives -save_on_disconnect control argument in login command (but note requirement for the disconnect_ok attribute, described above).

authorization
: The user's maximum authorization is the lowest of: the value in the project's SAT entry, the value in the user's PDT entry, and the value in the person's PNT entry.

grace
: The user's grace time is the smaller of the values in the project's SAT entry and the user's PDT entry.

max_background
: The smaller of the SAT and PDT values is the user's limit, except that a value of zero means no limit, so if one value is zero the other one is used, and if both are zero there is no limit.

max_foreground
: The smaller of the SAT and PDT values is the user's limit, except that a value of zero means no limit, so if one value is zero the other one is used, and if both are zero there is no limit.

pdir_quota
: The smaller of the SAT and PDT values is the process directory quota assigned to the user, except that a value of zero in either the SAT entry or the PDT entry denies permission to the project or user (respectively) to use the variable size process directory quota feature, causing that user to be given the system default process directory quota (which is specified in the >sc1>communications segment).

ring
: The user's initial ring is the larger of the initial ring values in the project's SAT entry and the user's PDT entry. The user's maximum ring is the smaller of the maximum ring values in the project's SAT entry and the user's PDT entry.

SAMPLE PROJECT MASTER FILE

The following is an example of a PMF with five registered users and an anonymous user:

```
Projectid:          Alpha;
Grace:              30;
Attributes:         v_process_overseer, v_home_dir,
                    no_start_up;
Cutoff:             2000, 01/01/80;

personid:           Smith;
```

```
personid:           Brown;
grace:              2900;
attributes:         preempting;
cutoff:             open;

personid:           Black;
cutoff:             20.00, midnight, daily;

personid:           Green;
shift_limit:        50.00, 200.00, 200.00, 200.00;
limit:              800;

personid:           Johnson;
initproc:           >user_dir_dir>Alpha>student_overseer_;
personid:           *;
initproc:           >user_dir_dir>Alpha>student_overseer_;
homedir:            [pd]>home;
attributes:         brief, ^v_process_overseer, ^v_home_dir;
lot_size:           200;
kst_size:           200;
cls_size:           512, stack;
shift_limit:        0, open, open, open;

end;
```

User Brown might be the project administrator. He is never cut off by the system, he is the only user allowed to bump others in the load control group, and (because his grace is so high) he is never demoted from primary to secondary.

Users Smith and Black are permitted full system capabilities. Smith, and all the other users in the group, are subject to the default cutoff restriction that they may not spend more than a total of $2000 or log in after 12/31/79. User Black has his own cutoff restrictions.

User Green also has full access to the system, but has monthly expenditure limits. He may spend a total of $800 per month, and may only spend $50 on shift 1 and $200 on shifts 2, 3, and 4. (He is able to spend the remaining $150 on absentee jobs and I/O requests.) The monthly limit of $800 is in addition to the cutoff limit of $2000 set by the Cutoff default keyword; if either limit is exceeded, Green is logged out.

User Johnson might be the programmer in charge of checking out the system for student use. His default process overseer is the project's special program, student_overseer_; however, Johnson may log in and obtain the full system capabilities by typing:

```
login Johnson -po process_overseer_
```

since he has the v_process_overseer attribute by default. (See the login command in *Multics Commands and Active Functions* manual, Order No. AG92 for more details.)

The anonymous user entry is the last in this PMF. The attributes listed for anonymous users restrict them to the special overseer, suppress the log-in message, and deny them a permanent home directory. Since no password statement is specified for the anonymous users, they log in without typing a password. The anonymous users are unable to log in on shift 1, and are also subject to the group's standard cutoff of $2000. Because the special overseer for the students has very limited capabilities, the sizes of the system tables have been specified to be smaller than usual, in order to reduce the number of page faults taken by the student processes. The anonymous users have no permanent storage in >udd>Alpha, unless it is provided for them by the subsystem. Their home directories are created for them on each login as subdirectories of their process directories.

MODIFYING A PROJECT MASTER FILE

The following sequence of Multics commands are used by a project administrator to add the limit keyword statement to user Brown's entry in the PMF. This same procedure is used to change any user attribute or to add and delete users. (For more details on user registration and deletion, see Section 4, "Project and User Registration and Deletion.") In this example, an exclamation point indicates lines typed by the project administrator.

```
! qedx
! r Alpha
! /Brown/
  personid:          Brown;
! a
! limit:             800;
! \f
! w
! q
  r 1301 1.202 1.548 96

! cv_pmf Alpha
  r 1303 1.433  1.321 82

! install Alpha.pdt
  r 1305 1.702 1.024 31

  From Initializer.SysDaemon (install) 1306.0:
  installed Alpha.pdt for Brown.Alpha.a
```

In the example, the qedx command is used to add a limit statement for user Brown to the PMF named Alpha described earlier in this section. The cv_pmf command converts the PMF to a PDT named Alpha.pdt. Alpha.pdt is then installed using the install command.

SECTION 4

PROJECT AND USER REGISTRATION AND DELETION

NEW PROJECT REGISTRATION

The following steps outline the usual routine required to set up a new project. These steps may vary slightly at different Multics sites. Some sites may not require the project registration form to be filled out or may substitute a different type of form.

1. Arrange for funding.

2. Choose a Project_id and fill out a project registration form.

3. Decide on an initial list of users.

4. Decide how much disk space the project members must have available to them initially. The system administrator makes this amount of quota available for them to use.

5. Appoint a project administrator.

6. Submit the registration form and any pertinent information to the system administrator.

Disk Usage

Quota is an administrative limit on disk record consumption. Disk space is measured in pages; a page contains 1024 36-bit words (4096) characters. Charges are made according to quota used, not quota allocated. For more information, see the *Multics Programmer's Reference Manual*, Order No. AG91 and the *Multics System Administration Procedures* manual, Order No. AK50.

Funding

The system administrator needs a valid requisition or purchase order in order to set up a project. The requisition states the name of the principal investigator, the requisition amount and termination date, and should give a name and address of the person who receives the bills.

The requisition amount may be increased later by a change order; it provides a limit stop, checked once a day by the system, which prevents users on the project from logging in once the project funds are exhausted. The requisition amount may be specified as open if the project does not want to be cut off for overexpenditure. The requisition termination date also provides a stop on users of the project. It too may be changed by a change order. When a project reaches its termination date, as specified on the requisition, it continues to accumulate storage charges for disk and tape rental until the system administrator receives notification from the project administrator requesting cancellation of the project.

By default, when the requisition balance is within ten percent of the requisition amount or less than $10, or when the requisition termination date is less than ten days away, each user on the project receives a warning message each time he logs in. The project administrator can change these percentages and amounts by modifying the PMF.

The amount of funding required varies widely, depending on the number of users on the project, the kind of users, and the price. An occasional user of the system might find his charges running about $100/month. A full-time professional programmer can consume as much as $5000/month, but typically between $1000 and $2000, not including disk charges. Consult a knowledgeable person at the Multics site for an estimate of how much money to allocate.

Project Registration Form

Most Multics sites have a project registration form; a sample form is provided in Appendix A. The project registration form has spaces for a one-line title describing the project, the name and address of the principal investigator, the project supervisor, and the project administrator, and the estimated disk requirement and estimated duration of the project. Also, specify the Person_id and Project_id of the project administrators and the directory into which the new project's PMF should be moved.

The disk requirement and estimated duration are included to help the installation plan the expansion of its storage facilities. At most sites, projects are given a small initial quota of 100 records. Then, if storage is available, the quota can be increased whenever the project gets low on disk.

List of Users

An initial list of users on the new project must be submitted to the system administrator. If these users are already registered on the system, their Person_ids and passwords are already in the system registration files. If any user is new to Multics, fill out a person registration form for him and supply an initial password. A sample initial user list and person registration form is included in Appendix A.

Submit all completed forms to the system administrator who then registers the project. At most installations a project can be registered in a few minutes. Unless there is some emergency, however, it is wise to allow at least one working day.

NEW USER REGISTRATION

The following sequence of steps describes the procedure for adding a new user to an already-existing project.

1. If the user is not personally registered on Multics, fill out a person registration form for the user and submit it to the system administrator. Clip a small piece of paper to the form giving the user's initial password.

2. Edit the PMF for the project and add the line:

```
personid:           Person_id;
```

3. Convert the PMF to a PDT by typing:

```
cv_pmf pmfname
```

4. Request installation of the resulting binary PDT by typing:

```
install pmfname.ptd
```

5. Wait. In a short time the system prints:

```
From Initializer.SysDaemon (install) 1527.0:
installed pmfname.pdt for Person_id.Project_id.tag
```

6. Delete the PDT by typing:

```
delete pmfname.pdt
```

When the PDT is installed, the new user's home directory is created.

ANONYMOUS USER REGISTRATION

Some projects need to allow users not registered on the system to log in under their project. This ability is provided by Multics through the "anonymous user" mechanism. Anonymous users are not distinguished by the system for the purposes of access control (which is the reason for the requirement that persons be registered) or billing.

Anonymous users log in to Multics with either the enter or enterp commands depending on whether or not the password statement is given after the anonymous user entry in the PMF. The enter command is used to log in an anonymous user without a password; the enterp command is used to login an anonymous user with a password. These commands are described in the *Multics Commands and Active Functions* manual, Order No. AG92. An anonymous user has a process created for him when he logs in. Depending on the process overseer specified with the initproc statement, the anonymous user may have access to all Multics facilities (if his process overseer procedure is process_overseer_), or some specialized subsystem (e.g., the Multics FAST subsystem or even a user-written environment specified by a pathname).

To set up an anonymous user in the PMF, the project administrator enters the following user keyword statement:

```
personid:          *;
```

This is the only required statement. However, all the user keyword statements, described in Section 3, are valid for anonymous users and can be specified after the personid statement. Global default values apply to anonymous users whenever a user keyword statement is omitted.

The cv_pmf command returns a warning message whenever default values are assumed for omitted homedir and initproc keywords. For example, if the project administrator does not specify the homedir statement for the anonymous user entry, a warning message is printed, and anonymous users are given the project directory as a home directory. If the initproc statement is omitted, a warning message is printed stating that the process overseer specified with the global Initproc statement is used by default.

Below is a sample anonymous user entry in a PMF:

```
personid:       *;
password:       anon;
initproc:       >udd>Project_id>login_proc;
homedir:        >udd>Project_id>users;
```

Since the password statement is given, all anonymous users who log in under this project must do so via the enterp command and give the "anon" password. These users are subject to a special process overseer. Also, they all share permanent storage in the specified home directory. Global keyword default values apply for the remaining unspecified user keywords.

USER DELETION

The following sequence of steps describes the procedure for deleting a user from a project:

1. Edit the PMF and delete the user's personid and any other keyword statements for that user.

2. Convert the PMF to a PDT by typing:

```
cv_pmf pmfname
```

3. Install the PDT by typing:

```
install pmfname.pdt
```

4. Wait. In a short time the system prints:

```
From Initializer.SysDaemon (install) 1539.1:
installed pmfname.pdt for Person_id.Project_id.tag
```

5. Delete the PDT by typing:

```
delete pmfname.pdt
```

6. Move segments to be saved to another directory and delete the user's home directory and segments as soon as possible since they continue to accumulate storage charges until deleted.

PROJECT TERMINATION

When a project is no longer to be used, the project administrator should request that the system administrator delete the project. The project is then removed from the system and all its segments are deleted. Any tapes or disk packs assigned to the project are reassigned. This means that before a project administrator requests the deletion of a project, he should arrange for the preservation of any programs and data that may be required later by punching the information on cards or writing it on a private volume.

The system administrator does not automatically delete a terminated project. A project is liable for storage charges until the system administrator receives notice to delete it, even if the project termination date has arrived.

SECTION 5

RESOURCE CONTROL

RESOURCE LIMITS

The resource limits provided for each user of a project enable the project administrator to control the overall consumption of his project's budget. Remember that the project as a whole is constrained in its total expenditures by the requisition amount. By limiting each user of his project to some fraction of the total budget, the project administrator ensures that each user gets a chance to consume some of these resources.

For instance, if a project has a budget of $1000 per month, and ten registered users, the project administrator might insert the following line at the top of his PMF:

```
Limit:              100.00;
```

so that no user could spend more than $100 per month.

The above example does not take into account the fact that the project as a whole, and not the individual user, is charged for storage and registration. It also ignores the possibility that one user might not need all of his $100 allocation while another might need more than the limit. Individual limit keyword statements in some users' entries might be necessary to override the global default value of $100.

Per-shift limits may be used to further control the resource consumption of users on a project. In order to save money, a project administrator might choose to lock his users out of shift 1 entirely by adding the statement:

```
Shift_limit:         0;
```

to the top of his PMF.

The cutoff keyword statement can be used to provide "absolute" limits, or to regulate the rate at which a user spends. For instance, the line:

```
cutoff:              100;
```

in a user entry sets a total expenditure limit of $100 for the user. He may spend this amount in a week, or over six months—but when he reaches this limit he is permanently cut off.

The statement:

```
cutoff:          100, 01/01/80;
```

establishes the same $100 maximum, and in addition cuts off the user after December 31, 1979, whether he has spent his $100 or not. On the other hand, the statement:

```
cutoff:          10,midnight,daily;
```

prevents the user from spending more than $10 in any one day. If the user overspends his limit, he is logged out automatically and is unable to log in again until midnight. (Since the limits are checked only at every 15 minute accounting update, and since I/O requests cannot be stopped by the limit, the user may be able to spend more than $10. When his limit is renewed, the system does not take the overrun into account; the user is given a full $10 for the next day.)

Checking Limits

The requisition balance, which functions as a per-project resource limit, is checked periodically by the crank; at most sites the crank is run once a day. Each users on the project is warned at login if the project is within the percentage of its limit specified in the warn_percent statement. When the limit is exceeded, users of the project are not able to log in until the limit is increased.

The per-user limits are checked at every login, every time the user creates a new process, and at every 15 minute accounting update. If a logged-in user exceeds his limit, he is bumped off the system (automatically logged out) with a few minutes warning. Absentee jobs are aborted before they get started if the user's total dollar limit is exceeded. Absentee jobs that have started are not stopped if they exceed the limit while running. I/O daemon requests are allowed to proceed even if the user has exceeded his dollar limit.

User Resource Usage Reporting

A user may discover how much of his resource limit he has consumed by invoking the resource_usage command described in the *Multics Commands and Active Functions* manual, Order No. AG92.

Project Resource Usage Reporting

Project administrators may generate a project usage report showing month-to-date resource consumption for all users on a project by typing:

```
proj_usage_report Project_id -bf
```

One line per user, plus a totals line, is printed.

The display_account_status command prints the latest accounting information for the project as a whole. Summarizations of all total dollar charges are printed, including disk. The display_account_status and proj_usage_report commands are described in detail in Section 10.

QUOTA LIMITS

The project administrator can limit secondary storage allocation consumed by users on his project. Since he has modify permission on the project directory, he can use the move_quota and get_quota commands to move quota from that directory to users' directories. The sweep and disk_stat_print commands can be used to determine the current distribution of quota. Both commands are described in Section 10. The move_quota and get_quota commands are described in Section 3 of the *Multics Commands and Active Functions* manual, Order No. AG92.

Requests for more disk quota should be submitted by the project administrator to the system administrator.

SECTION 6

LOAD CONTROL AND PREEMPTION

LOAD CONTROL GROUP

A load control group is a group of projects that share a guaranteed access to the system. Each load control group has a quota of primary load units, which represent a guaranteed number of users from the group who are able to be logged in at the same time. A user who attempts to log in is always able to do so if the group's primary quota is not filled. If, on the other hand, the group's primary quota is full but the system is not full, a user who attempts to log in is assigned secondary status. If the system becomes full, secondary users are preempted in order to log in primary users from other groups. If a group's primary quota is full and the system is full, a user from that group is refused login unless he can preempt some other primary user from his group.

Each project is assigned to a load control group by the system administrator. The project administrator can control who can have primary status and for what period of time, and who can preempt other primary users. Since several projects are often assigned to the same load control group, project administrators should work cooperatively to determine who from the group may have primary status and for how long.

A project administrator may decide that some users of his project should not be subjected to secondary status and its possible consequence of being preempted. To specify that a user should never be secondary—that is, that he should be refused login if he cannot be given primary status—the project administrator adds the following attributes statement to the user entry in the PMF:

```
attributes:          no_secondary;
```

This attribute may, of course, be combined with others.

The following attributes statement specifies that a user may not be given primary status:

```
attributes:          no_primary;
```

This statement ensures that the user is given secondary status only, and does not compete with other users of the group for primary status. He is only able to use the system when it is not full and may be preempted whenever it starts to become full.

If both the no_primary and no_secondary attributes are specified for a user, he is never able to log in. A warning message is printed by the cv_pmf command if such a PMF entry is found; but the PMF is still converted.

Preempting within a load control group, to prevent users who get primary status from holding onto the system all day, is controlled by two parameters: who can preempt, and who is preemptable. All users with the statement:

```
attributes:          preempting;
```

can preempt other primary users in their group.

Primary users, once they attain primary status, are immune from preemption for a period specified by the project administrator. This time period, the grace, controls how long users may monopolize primary status. Each project has a maximum grace, set by the system administrators. The project administrator can set the grace for a particular user with the statement:

```
grace:               n;
```

where n is the number of minutes the user may be logged in before he becomes subject to preemption. To set a default value for all users of a project, the project administrator can use the global statement:

```
Grace:               n;
```

If the project administrator does not specify either a global or user grace statement, or if he attempts to specify a grace in excess of his project's maximum, the project maximum grace set by the system administrator is used.

When a primary user logs out, a secondary user may be promoted to primary status. Whenever the system looks for a secondary user to promote, or for a preemptable user to preempt, it chooses the user who logged in first from the same load control group as the primary user who just logged out or, if that isn't possible, from the same load control group as the user who is doing the preempting.

WORK CLASS

A work class is a group of processes that are guaranteed a specified percentage of the available virtual CPU time. The work class membership of a process is determined by the load control group of the user owning the process and is under the control of the system administrator who estimates the needs of users is each load control group. Individual users working on high priority assignments may require a high percentage of CPU time. These users can be placed in private work classes. Placing individual users in private work classes is accomplished indirectly. They are first assigned to an individual load control group by the project administrator and then that group is placed in an private work class by the system administrator.

A project administrator who wants individual users in a private work class should request that the system administrator give the project the individual group (igroup) attribute and authorize one or more groups to which the project administrator can assign his users. The project administrator then assigns users to any of the authorized groups, using the Group and group keywords and the igroup attribute. When a group statement is specified for a user, the user is automatically assigned the igroup attribute by the cv_pmf command and placed in the specified load control group. When the Group statement is specified, any users who are given the igroup attribute (by an attributes or Attributes statement) are placed in the specified load control group. Users with neither a group statement nor the igroup attribute are not assigned an individual group (even if a Group statement is present) and are placed in the project's default group at log-in time. Users placed in individual groups are placed in the corresponding individual work classes, as specified by the system administrator.

SECTION 7

TAILORING THE USER ENVIRONMENT

This section describes the many means available to the project administrator for providing a different interface to some users on the project from that provided by the standard Multics system. There are many reasons why project administrators may wish to replace or modify the standard system interface. A project may need special facilities not provided by the normal interface, such as:

- Restriction of users to a limited set of system commands (a limited service subsystem).
- Restriction of users to a given rate of CPU consumption per hour logged in.
- Complete replacement of the standard command system interface with a set of interface programs simulating some other system.
- Isolation of users within a special-purpose subsystem (a closed subsystem).
- Separate accounting for anonymous users, who are lumped together by the system billing procedures.
- Additional user identification procedures, such as passwords for anonymous users or the maintenance of project numbers for subdivision of a user's resource accounts.
- Specialized accounting beyond those facilities provided by the system. For example, a project might wish to charge by the task completed rather than for the CPU time consumed.
- Use of special terminal devices with features not supported by the standard Multics I/O system.
- Extension of the user environment by making project-maintained programs appear to be part of the standard system.

These facilities and others like them can be provided by the project administrator for users on his project without the necessity of action by the system administrators. In most cases, the changes made by the project administrator take effect at the next login of a user.

The two most important "handles" for the project administrator in modifying the user environment are his ability to specify per-user attributes in the PMF, including the process overseer procedure; and his ability to modify the contents of the storage system hierarchy below the project directory. In order to provide the interface he desires for users on his project, a project administrator may need to be, or have the services of, a reasonably experienced Multics applications programmer.

PROCESS OVERSEERS

The process overseer sets up the environment during process initialization. initialize_process_ calls the listener to start reading commands, subject to the limitations set up by the procedures described below.

process_overseer_

The standard Multics process overseer procedure, process_overseer_, is the initial procedure assigned to a user unless the project administrator specifies otherwise by an initproc or Initproc statement in the project master file (PMF). The system process overseer searches for, and executes, the start_up.ec as described under "start_up exec_coms" later in this section (unless the project administrator has specified otherwise).

The process_overseer_ procedure is responsible for the following features of the default environment:

1. It creates an unclaimed signal handler that prints a message and creates a new listener level.

2. It explicitly allows the ".." escape to command_query_. This allows the user to execute any command before answering a question that is asked using the standard query routines.

3. It creates a handler for the mme2 handler that calls debug. This exists because debug breakpoints work by signalling the mme2 condition.

4. It finds a start_up.ec. It looks for start_up.ec in the home directory, then in the project directory, and finally in >sc1.

If a start_up.ec is found, then the listener is called with "exec_com Path login_type proctype" as an initial command line, where login_type is either "login" or "new_proc", and proctype is "interactive", "absentee", or "daemon".

If there was no start_up.ec, then the listener is called without an initial command line.

project_start_up_

Another installed process overseer, named project_start_up_, allows a project administrator to provide a project_start_up exec_com. It is identical in function to process_overseer_ except that it executes >udd>Project_id>project_start_up.ec before looking for start_up.ec.

A project_start_up exec_com allows the project administrator to specify a series of commands that some or all of the project's users will execute at the beginning of each process. The facility allows the administrator to guarantee that the users will execute the project_start_up exec_com, or they can be given the option of circumventing it (by being given the v_init_proc attribute as described below).

CREATING A project_start_up exec_com

To create a project_start_up exec_com, put the desired exec_com (or a link to it) into the project directory with the name project_start_up.ec. Thus if you are administering the Alpha project, the full pathname would be:

```
>udd>Alpha>project_start_up.ec
```

Give users on the project at least read access to the project_start_up exec_com. To make all of the users on your project execute the exec_com before their personal start_ups, put the following line in the project master file (PMF):

```
Initproc: project_start_up_;
```

ATTRIBUTES AND EXCEPTIONS

To except a user from executing the project_start_up exec_com, put the following line in the user's PMF entry:

```
initproc: process_overseer_;
```

If only a few of the users on the project are to execute the project_start_up exec_com, then omit the line "Initproc: project_start_up_;" from the PMF and insert the line "initproc: project_start_up_;" in the PMF entries of those users.

The ability of a user to substitute a process overseer for the one specified in the PMF is controlled by the v_init_proc attribute. A user with this attribute can change his process overseer by either specifying the -po argument to the login command, or putting a segment with the same name as the process overseer in the PMF in his home directory. It should be noted that a user with the v_init_proc attribute has almost complete control of his process environment, and can, among other things, avoid executing any of the exec_coms described above, even if he does not have the no_start_up attribute.

THE EXECUTION ENVIRONMENT

When the project_start_up exec_com is called, the environment is slightly different from the normal user environment. First, quits are disabled, so that the user cannot quit out of the start_up. Second, the working directory is set to the project directory, for users without v_init_proc. This prevents users from putting copies of programs called by the project_start_up in their home directories and thereby substituting them for the expected version. A side effect of this is that the project administrator can put commands and exec_coms called by the project_start_up in the project directory. An error handler is created so that errors in the exec_com cause the users to be logged out.

CONTENTS OF A project_start_up exec_com

The project_start_up exec_com allows the administrator to force the user to execute some commands at each login. Unless the project_start_up exec_com finishes up by putting the user into a restricted environment, however, the user can always undo whatever the exec_com does. The project_start_up exec_com should avoid doing things that the user could do for himself and might want to do differently, unless the user is being put in a limited environment in which he cannot do it for himself. For example, the project_start_up exec_com should not accept messages, since the user might prefer to use -hold or even -call. The proper place for such commands is the default start_up exec_com in the project directory which is executed only if the user lacks a personal start_up exec_com.

To enable quits during the execution of the project_start_up exec_com, insert a command line like:

```
io_call control user_i/o quit_enable
```

Limited Service Subsystem (LSS)

Project administrators can put their users in a limited environment, without having to write a special process overseer, by using the project_start_up_ process overseer and a project_start_up exec_com, together with the make_commands and enter_lss commands, described in Section 10. The LSS limits the commands and percentage of CPU time available to the user. The project administrator creates a control segment that contains allowable command names and their locations in the system with the make_commands command. Then the project administrator can insert the enter_lss command into the project_start_up.ec to specify the LSS control segment to be used by the command processor for the rest of the process. The command processor compares each command to those in the LSS control segment and accepts only those that are found. If a command is not found in the control segment, it is refused.

DEBUGGING

Since an error in the project_start_up.ec causes a logout, the exec_com should be debugged by explicitly specifying project_start_up_ to login with a command line like:

```
login Smith Multics -po project_start_up_
```

This requires the v_init_proc attribute.

Almost every feature of the standard Multics system interface can be replaced by providing a user with a specially tailored process overseer procedure in place of the standard version. A special process overseer can be used to inhibit quit signals (result of pressing the quit key on a terminal, usually marked ATTN, BRK, INTERRUPT, or QUIT). Also, all users can be forced to have an initial home directory of >udd>Project_id. A start_up.ec in this directory can do whatever is needed--including:

```
cwd path
```

The details of writing a process overseer are described in "Subsystem Programming Environment," in the MPM Subsystem Writers' Guide.

start_up exec_coms

The start_up exec_com facility is provided to allow users to write a script of commands that will be executed at the beginning of each new process. Start_ups are used to tailor the environment: users can accept messages, reset their search rules, check their mail, and the like. Many projects and sites find that the default environment for a user with no start_up exec_com is deficient. For example, a site or project might believe that all users should accept messages by default, so that consultants or other people can communicate with them. To this end, if the user does not have a start_up exec_com, the system searches for one first in the project directory (>udd>Project_id) and then in >sc1 for default start_up exec_coms. Thus if a project wishes to provide a default start_up exec_com, the project administrator needs only to create a segment named start_up.ec in the project directory, and users on the project without start_up exec_coms in their home directories will execute this exec_com instead. Similarly, the site administrator can create a segment named start_up.ec in >sc1, and users who lack start_up exec_coms in both their home directory and project directory will execute it. Users must, of course, be given at least read access to the default start_up exec_coms.

In addition to default start_up exec_coms a facility is provided for supplying a mandatory exec_com which is always executed before the user's start_up exec_com. The search mechanism described above is still used to find a personal start_up. Thus a project can supply both a mandatory start_up exec_com for some or all of its users, as well as a default start_up exec_com for users without personal start_up exec_coms.

CLOSED SUBSYSTEMS

Multics supports two subsystems: FAST and DFAST. To set up a Multics FAST user, the project administrator must specify the following keyword statements. Global statements can be used if the entire project is restricted to this subsystem.

```
initproc:       fst_process_overseer_;
attributes:     ^vinitproc, ^vdim;
```

To set up a Multics DFAST user, the project administrator must specify the following keyword statements. Global statements can be used if the entire project is restricted to this subsystem.

```
initproc:       dfast_process_overseer_;
attributes:     ^vinitproc, ^vdim;
```

RINGS

The Multics ring structure can be used to provide user environments that even a user with the ability to write and execute arbitrary PL/I programs cannot break out of. By placing the user in an outer ring, and interposing the project subsystem residing in a lower ring between the user and the system, the project administrator can provide a completely controllable interface which still allows the user to use general-purpose compilers. See "Access Control," of the *Multics Programmer's Reference Guide*, Order No. AG91 for more information concerning the ring mechanism. Also see "Keywords and Their Values" in Section 3 of this book for information on the ring and Ring keyword statements under "Special Environment Keywords".

SECTION 8

LOGICAL VOLUME QUOTA MANIPULATION

LOGICAL VOLUME

Segments in the storage system hierarchy are stored on disk volumes. These disk volumes are organized into groups called logical volumes. A logical volume consists of one or more disk volumes used by the storage system to contain segments. Storage is allocated on logical volumes according to the rules described in the *Multics Programmer's Reference Manual*, Order No. AG91.

When a logical volume is created, a registration record is created for it. This record contains the following information:

- list of the physical disk volumes comprising the logical volume
- owner identification
- switch setting, either public or private
- list of master directories
- list of users with quota accounts

The owner of the logical volume controls the access control segment (ACS) of the volume. Any user who is given "e" access to the logical volume is a volume administrator. Usually, system administrators are volume owners and project administrators are volume administrators.

LOGICAL VOLUME QUOTA

A volume administrator allocates disk quota on a logical volume to users who need space. The set_volume_quota command, described in Section 10, is used to set a user's quota account on a logical volume. Users with quota accounts on logical volumes can create master directories on these volumes as described in Section 9, "Master Directory Creation and Deletion." The list_mdir command, described in Section 10, is used to print a list of users with logical volume quota accounts.

SECTION 9

MASTER DIRECTORY CREATION AND DELETION

A master directory is a directory whose segments reside on a different logical volume than the segments of its containing directory. When a new directory is created, its segments, by default, reside on the same logical volume as the segments of its containing directory. If the segments in the new directory are to reside on some other logical volume, a master directory must be created. (See the *Multics Programmer's Reference Manual*, Order No. AG91 for more details.)

Master directories are created by using the -logical_volume control argument to the create_dir command. Only users with logical volume and master directory quota accounts can create a master directory. The set_mdir_owner command sets the owner of a master directory and the set_mdir_account command sets the quota account of a master directory. Both of these commands can only be issued by a volume administrator. Once the owner and quota account of the master directory are set, either the volume administrator, the user with a quota account, or the owner of the master directory can set the quota on the master directory with the set_mdir_quota command. The list_mdir command is used to list users with master directory quota accounts and owners of master directories. All of these commands are described in Section 10.

SECTION 10

COMMANDS USED BY A PROJECT ADMINISTRATOR

COMMAND DESCRIPTIONS

This section contains descriptions of the Multics commands used by project administrators. Each description contains the name of the command (including the abbreviated form, if any), discusses the purpose of the command, and shows the correct usage. Notes and examples are included when deemed necessary for clarity. The discussion below briefly describes the content of the various divisions of the command descriptions.

Name

The "Name" heading lists the full command name and its abbreviated form. The name is usually followed by a discussion of the purpose and function of the command and the expected results from the invocation.

Usage

This part of the command description first shows a single line that demonstrates the proper format to use when invoking the command and then explains each element in the line. The single line contains the full command name (or its abbreviated form) followed by the valid arguments. Some commands have required arguments; some commands have optional arguments. Most commands have both required and optional arguments; in general, the required arguments precede the optional arguments.

Any argument preceded and followed by a curly brace ({}) is an optional argument. Any other argument is a required argument. Anything specifically identified as "control_arg" in the usage line must be preceded by a minus sign in the actual invocation of the command. For example, the usage line:

```
commandname path {-control_arg} {xxx}
```

means that the command has one required argument and two optional arguments. Therefore, any of the following command lines are valid:

```
commandname path
commandname path -control_arg
commandname path xxx
commandname path -control_arg xxx
```

If a command accepts more than one of a specific type of argument, an "s" is added to the argument name. For example, the usage line:

```
commandname paths {-control_args}
```

means that the user must specify at least one pathname and may specify none, one, or several control arguments.

If a command accepts multiple arguments that must be in a specific order, the usage line is as follows:

```
commandname xxx1 yyy1 ... xxxn yyyn
```

to show that although several xxx and yyy arguments can be given, they must be given in pairs.

Notes

Comments or clarifications that relate to the command as a whole are given under the "Notes" heading. Also, where applicable, the required access modes, default condition (invoking the command without any arguments), and any special case information are included.

Examples

The examples show different valid invocations of the command. The results of each example command line are either shown or explained.

Other_Headings

Additional headings are used in some descriptions, particularly the more lengthy ones, to introduce specific subject matter. These additional headings may appear in place of, or in addition to, the notes.

Name: as_who

SYNTAX AS A COMMAND

```
as_who {-control_args} {User_ids}
```

FUNCTION

lists the contents of the answering service's user tables: answer_table, absentee_user_table, and daemon_user_table, in the directory >sc1. Access to these segments is usually restricted, preventing most users from using the as_who command. Users lacking access to these tables can use the who command (see the Commands manual) to list the whotab segment in >sc1, which is a public list of logged-in users.

When invoked as an active function, as_who returns Person_id.Project_id for the processes selected for output, separated by spaces.

ARGUMENTS

User_ids
are access control names of the form:

Person_id.Project_id
lists all users logged in with the specified Person_id and Project_id. The star convention is allowed in both Person_id and Project_id.

Person_id
lists all users logged in with the specified Person_id. The star convention is allowed.

Project_id
lists all users logged in with the specified Project_id. The star convention is allowed.

CONTROL ARGUMENTS

-absentee, -as
prints the ratio of absentee users logged in to the number of absentee slots currently available, and then lists the absentee users.

-channel chn_id, -chn chn_id
lists only interactive users whose tty ID matches chn_id, or daemon users whose source name (e.g., prta, vinc, etc) matches chn_id, or absentee users whose absentee name (e.g., abs1) matches chn_id. The chn_id argument may be a starname to cause several users to be listed.

-cpu
shows CPU usage in seconds for the listed processes. Use of this control argument required access to metering_gate_ or phcs_.

-daemon, -dmn
prints the number of currently active daemon processes and then lists them.

-disconnected, -disc
prints a list of disconnected processes only.

-group {name}, -gp {name}
prints a list of users that fall under the specified load control group (see "Notes" below).

-idle
shows how long in seconds the listed processes have been idle. Use of this control argument requires access to metering_page_ or phcs_.

-interactive, -ia
prints a list of all users having current interactive processes.

-long, -lg
prints the long form of output including log-in time and tty ID. If this control argument is not given, the command prints the User_id (Person_id.Project_id) and flag for each user (see "Notes" below).

-name, -nm
sorts the users by name.

-no_header, -nhe
suppresses column headings and load control heading from the printed output.

-pdir_volume {lv_name}, -pdv {lv_name}
either includes in the output the name of the logical volume containing the user's process directory segments (if no lv_name argument is given) or prints information about only those users whose process directory segments are on the volume specified by the lv_name argument.

-project, -pj
sorts the users by project.

-secondary, -sc
prints a list of all users having currently active secondary user processes (see "Notes" below).

NOTES

The default sort is by login time.

Anonymous users' true log-in names are shown, preceded by a *.

Specification of the -long control argument returns time of login, tty ID, weight, device channel, load control group, flags indicating special variables, and a User_id for each user.

Included in the output next to the User_id are flags, indicating preemption attributes (primary or secondary status), and the current status of that user's process (these attributes vary according to the login instance and the discretion of the project administrator). When this command is invoked with the -long control argument, the column listing these flags has the heading "PNDS." The flags in these four column positions indicate the user's status with respect to: Preemption, Nolist, Disconnected, and Suspended status. Additionally, if the -idle control argument is specified, R and W flags can occur. An R flag indicates the process is ready; a W flag indicates the process is waiting for another event.

Possible preemption attribute flags are:

<blank>
indicates that the user has primary status.

S
indicates that the user has secondary status.

+
the user has the nobump attribute (cannot be preempted).

>
the user is subject to bumping (preemption) by other members of his project, whose grace period has not yet run out.

X
the user has been bumped but has not yet logged out.

D
identifies a daemon user.

Possible nolist flags are:

<"blank>"
the user does not have the nolist attribute.

N
the user has the nolist attribute.

Possible disconnected flags are:

<blank>
the user is not disconnected.

D
the user is disconnected.

Possible suspended flags are:

<blank>
the user is not suspended.

S
the user is suspended.

EXAMPLES

In the following example, the as_who command is invoked with the -long control argument, and the resulting output is printed:

```
! as_who -long

Multics mpp03221; Development Machine.
Load = 11.0 out of 70.0 units; users = 11 out of 70
Absentee users = 1 background, 1 foreground; Max background absentee users = 3
Daemon users = 7
System up since 04/23/84   1308.3
Scheduled shutdown at 04/23/84  2300.0
Last shutdown was at 04/22/84  2307.7

Login at      TTY  Load   Chan      Group    PNDS  User ID

      2119.1  none  1.0   a.h018    SysProg  > D  Smith.Multics
      2150.3  001   1.0   a.h019    Admin    >    Jones.SysAdmin
      2200.1  Q 1   1.0   abs1      Admin         Jones.SysAdmin (crank)
      2207.4  Q FG  1.0   abs2      SysProg       Doe.Multics (load_tape)
      1308.4  cord  1.0   cord      IO       D    IO.SysDaemon
      1308.6  bk    1.0   bk        System   D    Backup.SysDaemon
      1308.6  prta  1.0   prta      IO       D    IO.SysDaemon
      1308.6  puna  1.0   puna      IO       D    IO.SysDaemon
      1308.7  vinc  1.0   vinc      System   D    Volume_Dumper.SysDaemon
      1308.7  nw    1.0   nw        System   D    Network_Daemon.Daemon
      1308.7  ns    1.0   ns                 D    Network_Server.Daemon
```

The following example invokes the as_who command to list all users on projects that begin with "Mi".

```
! as_who .Mi*
```

The following example invokes the as_who command to list all users on the CompBar project whose names begin with "A".

```
! as_who A*.CompBar
```

Name: cv__pmf

SYNTAX AS A COMMAND

```
cv_pmf pmf_path {-control_args}
```

FUNCTION

converts an ASCII project master file (PMF) into a binary project definition table (PDT).

ARGUMENTS

pmf_path
: is the pathname of the PMF. If path does not have a suffix of pmf, one is assumed; however, the suffix pmf must be the last component of the name of the source.

CONTROL ARGUMENTS

-brief, -bf
: prints error messages in the short format.

-long, -lg
: prints error messages in the long format.

-severity N, -sv N
: causes error messages whose severity is less than N (where N is 0, 1, 2, 3, or 4) not to be written to the user_output switch. If not specified, a severity level of 0 is assumed; i.e., all error messages are written to the user_output switch.

NOTES

The newly converted PDT is placed in the current working directory. The entryname of the new PDT is the same as the entryname of its source PMF, with the pmf suffix replaced by pdt. This command associates the following severity values to be used by the severity active function:

cv_pmf

Value	Meaning
0	No compilation yet or no error.
1	Warning.
2	Correctable error.
3	Fatal error.
4	Unrecoverable error.
5	Could not find source.

EXAMPLES

```
cv_pmf >udd>Project_id>projfile
```

converts the PMF >udd>Project_id>projfile.pmt and creates a PDT named projfile.pdt in the project administrator's working directory.

delete_volume_quota

Name: delete_volume_quota, dlvq

SYNTAX AS A COMMAND

```
dlvq logical_volume account
```

FUNCTION

is used to delete a quota account for a logical volume. This command is to be used by volume executives.

ARGUMENTS

logical_volume
is the name of the logical volume from which quota is to be deleted.

account
is the name of the quota account (in the form Person_id.Project_id.tag) to be deleted.

NOTES

The quota account cannot be deleted if there are still master directories whose quotas are charged against the account to be deleted. Such directories must either be deleted or transferred to another account (see the set_mdir_account command).

ACCESS REQUIRED

To use this command, the user must have execute access to the logical volume. It is not necessary that the volume be mounted.

Name: disk_stat_print

SYNTAX AS A COMMAND

```
disk_stat_print {path} {-control_args}
```

FUNCTION

prints the disk_stat segment that is created by the sweep command. Optional control arguments cause the information to be presented in a variety of ways, to facilitate analysis of disk usage patterns. By default, a header is printed, followed by one line for each directory in the disk_stat segment, and a totals line.

ARGUMENTS

path
: is the pathname of the disk_stat file to be printed. If path is not given, the disk_stat segment in the working directory is assumed.

CONTROL ARGUMENTS

-level L, -lev L
: summarizes each subtree that begins at level L; prints no lines with level numbers greater than L. The effect of this argument is to make the output appear as if no directories with quotas existed below level L in the hierarchy since the directories below level L have their usage figures included in those of whatever level L directory they are inferior to.

  The default value for L is 16, causing all directories in disk_stat to be printed individually. The root's level is 0; a directory is said to be below level L if its level number is greater than L. A value of 2 for L causes the disk usage of each project to be displayed in a single line; a value of 3 causes the usage of each user to be displayed (provided that the user directories have quotas).

-logical_volume, -lv
: prints the name of the logical volume on which segments contained in each directory reside, and the usage figures for each logical volume. An extra column is printed in the directory line, giving the logical volume index (lvix). This is merely the line number in the table of logical volume totals that is printed after the regular totals lines. An lvix of zero indicates that the subtree summarized by the line contains segments that reside on more than one logical volume; an lvix of -1 indicates a logical volume not known to the system.

-subtotal, -stt
: prints subtotal lines giving the totals for each subtree. Each time a directory is encountered whose level number is less than that of the preceding directory, one or more subtotal lines are printed. This argument causes the maximum amount of subtotal information to be printed. Some users may find the resulting output too cluttered to be easily read. The -level control argument provides an alternative, producing less information, but in an easier to read format.

-total, -tt
does not print a line for each directory; rather prints a totals line (plus any other lines specified by other arguments).

NOTES

The first date printed in the header is the date of the last billing. It is filled in by charge_disk if the disk_stat file is the one used by the daily disk accounting job. This date is zero in files created separately (e.g., for use in disk usage analysis).

Although the output of disk_stat_print is designed to be printed on a line printer (using the file_output and dprint commands, described in the *Multics Commands and Active Functions Manual*, Order No. AG92), it can be printed on a terminal if the carriage is wide enough. Each line consists of approximately 70 characters followed by a pathname. Pathnames can be up to 168 characters long but are typically less than 50 characters long.

EXAMPLES

```
disk_stat_print Alpha.disk_stat -level 3
```

prints one disk usage line for each user on the Alpha project, assuming that the segment Alpha.disk_stat was created as shown in the example under the sweep command.

```
disk_stat_print Alpha.disk_stat -total
```

prints a single line giving the disk usage for the entire Alpha project.

display_account_status

Name: display_account_status, das

SYNTAX AS A COMMAND

```
das {Project_id} {-control_args}
```

FUNCTION

prints the latest accounting information for a project. The information is stored in the PDT of that project and is correct as of the last time the daily accounting job was run; it is usually run every night.

ARGUMENTS

Project_id
: is the Project_id of the project. If this argument is not given, the project under which the project administrator is currently logged in is assumed.

CONTROL ARGUMENTS

-brief, -bf
: prints a one-line summary of the account information.

-long, -lg
: prints all information found in the projfile (project registration segment) entry and the reqfile (requisition segment) entry.

-no_header, -nhe
: suppresses printing of the header.

ACCESS REQUIRED

The user must have read access to the PDT to use this command; usually only project administrators have such access.

NOTES

If neither the -brief nor -long control argument is given, all information about charges is printed.

See also the proj_usage_report command to get a brief summary of each user's resource consumption and the print_pdt command to get more detailed information about each user.

EXAMPLES

The following example illustrates the output of the das command when it is used with the -long control argument:

```
das Multics.pdt -long


                    Multics.pdt                      12/04/84  0855.3

                    Account information copied       12/03/84  2130.7

Projectid:          Multics;

REQFILE
 account:           SYSPROG
 reqno:             Multics
 rate structure:    default
 qflag:
 date on:           03/13/75  1254.4 est Thu
 date off:
 billing name:      System Administration
 billing addr:      Accounting Office
 charge this month: $     57285.74
 charge this req:   $   2414583.92
 req amount:        $         0.00      balance:  OPEN
 cutoff date:       01/01/99  1254.4 est Fri

PROJFILE
 title:             Worker Bees
 investigator:      John Gintell
 inv address:       Right Here
 supervisor:        CTC
 sup addr:          There
 sup phone:         phone_no
 date on:           03/13/75  1254.5 est Thu
 date off:
 disk page-months:  29296,  $ 87.89
 disk quota:        67538
 disk use:          27329
 dir disk use:      2214
 misc charges:      $ 0.00
 # misc charges:    0
```

display_account_status

```
SAT
 attributes:        primary_line, guaranteed_login, anonymous, nopreempt,
                    dialok, multip, bumping, brief, vinitproc, vhomedir,
                    nostartup, daemon, igroup, save_pdir, disconnect_ok;
 ring:              1,5
 alias:             m
 default group:     SysProg
 authorized groups: *,Admin
 max grace:         2880
 audit flags:       seg_init, dir_init, mc_seg_init, no_access, ipr_fault,
                    acv_mode, acv_ring, no_wakeup, sys_priv
 authorization:     system_low : system_high
 days to cutoff:    5143
 percent balance:   100%
 dollars to cutoff: $ 0.00
```

The following example illustrates the output of the das command without the -long argument:

```
das Multics.pdt

                    Multics.pdt                      12/04/84  0857.7

                    Account information copied       12/03/84  2130.7

Projectid:          Multics;
 charge this month: $     57285.74
 charge this req:   $   2414583.92
 req amount:        $         0.00      balance:  OPEN
 cutoff date:       01/01/99  1254.4 est Fri
 disk page-months:  29296,  $ 87.89
 disk quota:        67538
 disk use:          27329
 dir disk use:      2214
 misc charges:      $ 0.00
 # misc charges:    0
```

Name: enter_lss

SYNTAX AS A COMMAND

```
enter_lss path
```

FUNCTION

causes the command processor to compare each command to a supplied Limited Service Subsystem (LSS) control segment, which was created using the make_commands command. Any command not found in the control segment is refused. Those found are mapped into the command specified by the control segment.

ARGUMENTS

path
is the pathname of the LSS control segment.

NOTES

The LSS control segment must be previously created with the make_commands command. The command line as shown in the example above is usually included in the project_start_up.ec. See the make_commands command for more details on the LSS facility and the use of the project_start_up exec_com.

EXAMPLES

To limit users on a project to a subsystem as specified in an LSS control segment named student_commands, place the following command line in their project_start_up.ec:

```
enter_lss >udd>Students>student_commands
```

Name: get_dir_quota

SYNTAX AS A COMMAND

```
get_dir_quota paths {-control_arg}
```

FUNCTION

returns information about the directory quota and pages used by inferior directories. For more information on directory quota see the *Multics Programmer's Reference Manual*, Order No. AG91.

ARGUMENTS

paths
: are the names of the directories for which directory quota information is desired. If one of the paths is -wd or -wdir, the working directory is used. If no paths are given, the working directory is assumed. The star convention can be used to obtain directory quota information about several directories.

CONTROL ARGUMENTS

-long -lg
: specifies that the long form of output is to be used. This control argument may appear anywhere on the command line.

NOTES

The short form of output (the default case) prints the number of pages of directory quota assigned to the directory and the number of pages used by the directories contained in that directory and any other inferior directories that are charging against that quota. The output is prepared in tabular format, with a total, when more than one pathname is specified. When only one pathname is specified, a single line of output is printed.

The long form of output gives the directory quota and records-used information provided in the short output. In addition, it prints information about the number of immediately inferior directories with nonzero quotas. It also shows the time-record product in units of record-days, along with the date that this number was last updated. Thus, a user can see what secondary storage charges his accounts are accumulating. If the user has inferior directories with nonzero quotas, he has to print this product for all these directories in order to obtain the charge.

Name: install

SYNTAX AS A COMMAND

```
install path {-control_args}
```

FUNCTION

requests installation of a system control table. The request is transmitted to the system control process which validates the request and performs the installation. A message from the system control process indicates successful installation or rejection of the table.

A project administrator can install a PDT only; a system administrator can also install a number of additional tables.

ARGUMENTS

path
: is the relative or absolute pathname of the PDT to be installed. The PDT must have previously been produced by the cv_pmf command. The pdt suffix must be given.

CONTROL ARGUMENTS

-all, -a
: installs all attributes.

-attributes, -attr
: installs only nonsecurity related attributes. This is the default if no control arguments are specified.

-authorization, -auth
: installs only security related attributes.

NOTES

The install command reports PDT parameters that exceed limits specified for the project in the SAT, but it allows the PDT to be installed. If the SAT limits are not subsequently raised, they are enforced at login time and a message to that effect is logged. This is done for the initial ring, max ring, grace time, and pdir quota parameters.

EXAMPLES

```
install alpha.pdt -auth
```

installs the security related attributes in the PDT named alpha.pdt found in the project administrator's working directory.

```
install >udd>Demo>demo.pdt
```

installs the PDT named demo.pdt found in the Demo project directory.

A message of the following form is sent to the project administrator when the installation has been completed:

```
From Initializer.SysDaemon.z (install) 1321.0:
installed Demo.pdt for Jones.Demo.a.
```

Name: list_mdir, lmd

SYNTAX AS A COMMAND

```
lmd volume_names {-control_args}
```

FUNCTION

prints logical volume and master directory quotas.

ARGUMENTS

volume_names
are the names of the logical volumes.

CONTROL ARGUMENTS

-account User_ids
specifies a list of quota account names for which information is desired, where each User_id is of the form Person_id.Project_id. Asterisks (*) may not be used when specifying quota account names. Asterisks in account names only match quota account names that contain asterisks.

-all, -a
prints information about all users of the logical volume.

-brief, -bf
suppresses header and shortens the output lines.

-directory, -dr
prints only master directory information.

-long, -lg
prints additional information, including the quota account for each directory.

-owner User_ids
specifies a list of directory owners for which information is desired, where each User_id is of the form Person_id.Project_id. (An asterisk (*) can be used when specifying either component of an owner name.) If this control argument is omitted, information is printed only for directories owned by the user issuing the command.

-quota
prints only quota information. (See "Examples" below.)

ACCESS REQUIRED

The user must have "e" access to the logical volume to use the -owner, -account, and -all control arguments.

NOTES

If neither the -quota nor the -directory control argument is specified, information about both quotas and directories is printed. It is not necessary that the logical volume be mounted to use this command.

If the -all control argument is specified, the -owner and -account control arguments cannot be given. If both the -owner and -account control arguments are specified, information is printed only for directories that match both conditions.

EXAMPLES

In the example below, the user with Person_id Jones on the Fed project requests information on the logical volume named work. The system responds with information about Jones' master directories. In this example and the examples that follow, an exclamation point indicates lines typed by the user.

```
!  lmd work
       QUOTA        PATHNAME
       100          >udd>Fed>Jones>sub
       250          >udd>Fed>Jones>sub1>sub2
       350          records of quota assigned, 500 available.
```

In the example below, Jones requests brief information on the logical volume named work. The system responds with information about Jones' master directories.

```
!  lmd work -bf
       >udd>Fed>Jones>sub
       >udd>Fed>Jones>sub1>sub2
       Quota=500, used=350.
```

In the example below, Jones requests brief, quota information about the logical volume named work.

```
!  lmd work -quota -bf
       Quota=500, used=350.
```

In the example below, Jones requests information about all users of the logical volume named work. The use of the -all control argument requires "e" access on the logical volume.

list_mdir

```
!  lmd work -all
       OWNER          QUOTA      PATHNAME
       Jones.Fed      100        >udd>Fed>Jones>sub
       Jones.Fed      250        >udd>Fed>Jones>sub1>sub2
       Smith.Fed      900        >udd>Fed>Smith>work

       ACCOUNT        USED       VOLUME QUOTA
       Smith.Fed      900        1000
       *.Fed          350        500

       Total volume quota: 1500
       Total quota used: 1250
```

Name: load_ctl_status

SYNTAX AS A COMMAND

```
load_ctl_status {group} {-control_args}
```

FUNCTION

prints the current status of the system load control groups. It does this by printing selected items from the system copy of the master group table (MGT).

ARGUMENTS

group
: if specified, prints only the header and the line for the group named; otherwise, prints one line for each group in the MGT. Each line gives the maximum number of primary load units, the current number of primary load units, the current number of secondary load units, and, if the group has an absolute maximum, the total and maximum number of units. Also, the group's current load, as a percent of the total allowable system load, is given.

CONTROL ARGUMENTS

-long, -lg
: requests a long format header.

-total, -tt
: requests that only a header be printed.

NOTES

If the priority scheduler is enabled, then each line gives, in addition, the interactive and absentee work class of the group, and the header contains two additional lines giving the defined work classes and their percents, for the current shift.

Name: make_commands

SYNTAX AS A COMMAND

```
make_commands path_name
```

FUNCTION

creates a Limited Service Subsystem (LSS) control segment from an ASCII input segment. The control segment is referenced by the Limited Service System (LSS) when it is limiting the commands and percentage of CPU time of a user (see also the enter_lss command).

The input segment consists of a series of statements. Each statement is composed of two parts. The first part is the name of the command to be transformed; i.e., the command that is to be typed by the user in a limited system. If there is more than one name for the command, they should all be enclosed in parentheses and separated from each other by one or more blanks. The name field is terminated by a colon preceded by any number of blanks.

The second part of each statement is the pathname (which may be a relative pathname) of the command to be executed when the user types one of the names in the first part. If a relative pathname is used, it is relative to the current working directory. If only an entryname is given, the standard system search rules are applied. It is followed by any number of blanks and terminated by a semicolon. If the pathname is omitted (semicolon still required), it is assumed to be the same as the last name in the name field.

The first and second parts of each statement may be separated from each other by any number of blanks or tabs. Newlines are ignored and are allowed anywhere. Comments enclosed between "/*" and "*/" are allowed and are treated as blanks.

If the first two statements have as their first part the names "ratio" and "interval", respectively, the second parts of the two statements are assumed to be decimal integers to be assigned to the ratio and interval_length variables of the LSS control segment. Otherwise, the two variables are set to zero.

The ratio and interval variables control the amount of CPU time used by the process. The LSS forces the process to use no more than (interval/ratio) virtual CPU seconds in each (interval) real second(s) (see "Examples" below). If it attempts to do so, the process is rendered inactive for the remainder of the interval.

ARGUMENTS

path_name
: is the pathname of an ASCII input segment that has the name path_name.ct. (The .ct suffix is assumed if it is not included.) The output segment has the same entryname as the input segment with the .ct suffix removed, and is placed in the working directory.

EXAMPLES

A project administrator wants to create a limited command environment for a project consisting of student users. The first step is to create the segment >udd>Students>student_commands.ct, with the following contents:

```
/* set the ratio and interval */

ratio:    60;
interval: 120;

/* define the commands */

(list ls):          >abc>special$list;
logout:   ;
edit:     bsys;
start:    ;
hold:     ;
(pr print):         ;
```

Then, to convert this segment to an LSS control segment, type:

```
make_commands student_commands.ct
```

The command then creates an output segment named student_commands, which can be put to use with the enter_lss command.

In the above example, the ratio is set to 60, and the interval is set to 120, indicating that no more than 2 virtual CPU seconds are allowed per 120 real seconds.

NOTES

See the enter_lss command for additional information.

move_dir_quota

Name: move_dir_quota

SYNTAX AS A COMMAND

```
move_dir_quota path1 quota_change1 ...  {pathN quota_changeN}
```

FUNCTION

allows a user to move records of directory quota between two directories, one immediately inferior to the other.

ARGUMENTS

pathi
: is the pathname of a directory. The quota change takes place between this directory and its containing directory. A pathi of -wd or -wdir specifies the working directory. The star convention cannot be used.

quota_changei
: is the number of records to be subtracted from the containing directory's directory quota and added to the directory quota on pathi. If this number is negative, records are added to the containing directory's directory quota and subtracted from the directory quota on pathi.

ACCESS REQUIRED

The user must have modify permission on both the directory specified by pathi and its containing directory.

NOTES

After the change, the directory quota must be greater than or equal to the number of records used by directories in pathi unless the change would make the quota zero.

If the change would make the directory quota on pathi zero, there must be no immediately inferior directory with nonzero quota. When the directory quota is changed to zero, the records used and the time-record product for pathi is reflected up to the superior directory.

EXAMPLES

```
move_dir_quota >udd>Demo>Smith>1_dir 1000
>udd>Demo>Smith>1_dir>2_dir -50
```

adds 1000 records to the directory quota on >udd>Demo>Smith>1_dir and tracts 1000 records from the directory quota on >udd>Demo>Smith. It then tracts 50 records from the directory quota on >udd>Demo>Smith>1_dir>2_dir and adds 50 records to the directory quota on >udd>Demo>Smith>1_dir.

Name: print_pdt

SYNTAX AS A COMMAND

```
print_pdt path {Person_ids} {-control_args}
```

FUNCTION

The print_pdt command prints a listing of a project definition table (PDT).

ARGUMENTS

path
is the pathname of the PDT segment to be printed. If the pdt suffix is not given, it is assumed. If the pathname given does not start with a greater-than or less-than character, it is interpreted as a project name and the PDT in the directory containing PDTs (>sc1>pdt) is used.

Person_ids
are the Person_ids about whom information is desired. If this argument is omitted, information is printed for all users listed in the PDT.

CONTROL ARGUMENTS

-brief, -bf
prints small amount of information about each user.

-long, -lg
prints all data items in the PDT.

-no_header, -nhe
suppresses printing of the header.

-pmf
prints the PDT in project master file (PMF) format. The file_output command (described in the *Multics Commands and Active Functions* manual, Order No. AG92) can be used to place the printed PDT in a segment for daemon printing or for subsequent use as a PMF (see "Notes" below).

NOTES

If no control arguments are given with this command, all PMF-specifiable attributes and the total amount spent are printed. The user must have read access to the PDT; usually only project administrators have such access. The following command line is recommended to make a PMF from a PDT:

```
fo Project_id;print_pdt Project_id -pmf;ro
```

See also the proj_usage_report command to get a brief summary of each user's resource consumption and the display_account_status command to obtain the charges accrued to the account.

proj_usage_report

Name: proj_usage_report, pur

SYNTAX AS A COMMAND

```
pur {Project_id} {-control_args}
```

FUNCTION

prints a project usage report for the current billing period.

ARGUMENTS

Project_id
is the Project_id of the project. If this argument is not given, the project under which the project administrator is currently logged in is assumed.

CONTROL ARGUMENTS

-brief, -bf
prints reports in one short line per user.

-long, -lg
prints detailed information about per shift, per absentee, per device, and I/O daemon queue usage.

-no_header, -nhe
suppresses printing of the header.

-pathname path, -pn path
is the pathname of a PDT. The pdt suffix must be given. This control argument is used to print a PDT not currently being used by the answering service. If this control argument is specified, the Project_id argument may not be given.

-reverse, -rev
reverses the order of the sort.

-sort XX
sorts output according to XX, where XX can be the string:

```
name
usage
rem
limit
fraction_used
```

to specify users' names, usage, remainder, limits, or entries in order of ratio between usage and limit. Only one string may be specified. The default prints the PDT as is.

-total, -tt
    does not print a line for each user; rather prints a totals line (plus any other lines specified by other arguments).

-user Person_id
    prints information on only the user specified by Person_id.

ACCESS REQUIRED

The user must have read access on the PDT; usually only project administrators have such access.

NOTES

If neither the -brief nor -long control argument is given, the report printed contains one detail summary line for each user.

See also the print_pdt command to get more detailed information about each user and the display_account_status command to obtain a summary of the charges accrued to the project.

reconnect_ec_disable

Name: reconnect_ec_disable

SYNTAX AS A COMMAND

```
reconnect_ec_disable
```

FUNCTION

suppresses any attempt to find or invoke the reconnect.ec segment upon reconnection to a disconnected process.

NOTES

The reconnect_ec_disable command reverses the effect of the reconnect_ec_enable command, which is the default setting when the standard process overseer procedure process_overseer_ is used. See Section 7 of the MAM -- Project Administrator Manual, Order No. AK51 for a discussion of process overseers and their use.

reconnect_ec_enable

Name: reconnect_ec_enable

SYNTAX AS A COMMAND

```
reconnect_ec_enable
```

FUNCTION

invokes a search for the reconnect.ec segment upon reconnection to a disconnected process. The search begins in the home directory, continues through the project directory, and then through >scl until the segment is located, at which time the command "exec_com >DIRECTORY_NAME>reconnect.ec" is executed.

NOTES

The reconnect_ec_enable command reverses the effect of the reconnect_ec_disable command.

Use of reconnect.ec is enabled automatically by the standard process overseer procedure process_overseer_. Invocation of reconnect.ec is not automatically enabled by the project_start_up_ process overseer (see Section 7 of the MAM -- Project Administrator Manual, Order No. AK51). Thus, when using project_start_up_, the project administrator may enable invocation of reconnect.ec at any point in the project_start_up.ec.

The current command processor is used to execute the command. Thus, if the user is using the abbrev command processor, any applicable abbreviation will be executed.

set_mdir_account

Name: set__mdir__account, smda

SYNTAX AS A COMMAND

```
smda path {User_id}
```

FUNCTION

sets the quota account of a master directory. (See the *Multics Programmer's Reference Manual*, Order No. AG91 for an explanation of master directories.)

ARGUMENTS

path
: is the pathname of the master directory whose quota account is to be changed.

User_id
: is the name of the new quota account of the master directory, specified as Person_id.Project_id. If omitted, the Person_id and Project_id of the user issuing the command are used.

ACCESS REQUIRED

The user must have "e" access on the logical volume containing the master directory. The volume need not be mounted.

NOTES

The quota for the master directory is returned to the old quota account and is withdrawn from the new quota account. The new quota account must have sufficient quota to allow this.

set_mdir_owner

Name: set__mdir__owner, smdo

SYNTAX AS A COMMAND

```
smdo path {User_id}
```

FUNCTION

sets the owner of a master directory. (See the *Multics Programmer's Reference Manual*, Order No. AG91 for an explanation of master directories.)

ARGUMENTS

path
is the pathname of the master directory to be changed.

User_id
is the Person_id.Project_id of the new owner of the master directory. If omitted, the Person_id and Project_id of the user issuing the command are used.

ACCESS REQUIRED

The user must have "e" access on the logical volume containing the master directory. The volume need not be mounted.

Name: set_mdir_quota, smdq

SYNTAX AS A COMMAND

```
smdq path1 change1 ...  {pathN changeN}
```

FUNCTION

sets the quota on a master directory.

ARGUMENTS

pathi
: is the pathname of a master directory whose quota is to be changed.

changei
: is the amount of quota, or amount of quota change, and can be specified as follows:

```
+n  add n records of quota to pathi
-n  subtract n records of quota from pathi
 n  set the quota on pathi to n records
```

ACCESS REQUIRED

The user must have modify permission on the master directory, and must be the owner of the master directory, be a volume administrator, or have the same quota account as the master directory.

NOTES

If the quota is being increased, the master directory's quota account must have sufficient volume quota to satisfy the request.

The quota of a master directory can never be zero, and it can never be set less than the current number of records being charged against the master directory.

Name: set_volume_quota, svq

SYNTAX AS A COMMAND

```
svq logical_volume change {account}
```

FUNCTION

sets a quota account's volume quota on a logical volume. This command is to be used by volume administrators.

ARGUMENTS

logical_volume
is the name of the logical volume for which quota is to be set.

change
is the amount of quota, or the amount of quota change, and can be specified as follows:

```
+n  add n records to the quota
-n  subtract n records from the quota
 n  set the quota to n records
```

account
is the Person_id.Project_id of the user with a quota account for the logical volume. If omitted, the Person_id and Project_id of the user issuing the command are used.

ACCESS REQUIRED

To use this command, the user must have "e" access to the logical volume. It is not necessary that the volume be mounted.

NOTES

If the volume quota is set less than the quota account's current quota used, the quota is changed as directed, but a warning message is printed.

Name: sweep

SYNTAX AS A COMMAND

```
sweep {path} {-control_args}
```

FUNCTION

obtains the disk usage figures for the hierarchy, or any specified subtree, and places them in a segment for subsequent use by accounting and disk usage analysis tools. By default, the segment is named disk_stat. It is not an ASCII segment, and the disk_stat_print command must be used to print its contents.

ARGUMENTS

path
: is the pathname of the directory at which the sweep is to start. The usage figures for this directory, and for the entire subtree beneath it are recorded. If path is omitted, the sweep begins at the root directory. See "Notes" below.

CONTROL ARGUMENTS

-brief, -bf
: does not print an error message when unable to force access to a directory to read its usage figures. This is the default.

-long, -lg
: prints an error message if unable to force access to a directory.

-output_file XX, -of XX
: places the output in the specified segment XX rather than in the default segment, named disk_stat, in the working directory.

-pdd
: do not exlude >pdd from the sweep. By default, a sweep that starts at the root deliberately omits >pdd. This argument is not used in the accounting application of this program, but it may be used during an analysis of disk space usage, when complete and accurate total page usage figures are desired. The initializer process must first be used to give sma access to >pdd to the process that will run the sweep.

NOTES

The figures recorded in disk_stat are the quota, records used, and time-record product (trp). These figures are recorded separately for segment pages and directory pages.

Since directories with zero quotas have their record usage and trp figures included in those of the first superior directory with nonzero quota, it is not necessary for sweep to walk the entire hierarchy. Instead, whenever it encounters a directory with a zero quota, it goes no further down in that particular branch, but instead goes on to the next directory at the same level. This applies separately to segment and directory quotas.

Therefore, any directory whose usage figures are to be recorded separately for purposes of accounting or disk usage analysis must be given nonzero segment and directory quotas. It is recommended that at least all directories immediately inferior to the root and all project directories under >udd be given nonzero segment and directory quotas.

Project administrators may use this command to obtain detailed information about the disk usage of a project's users by executing sweep with the project directory pathname as the first argument. This is possible, however, only if the users' home directories have nonzero quotas.

EXAMPLES

```
sweep >udd>Alpha -output_file Alpha.disk_stat -long
```

places disk usage information for the Alpha project directory and all directories beneath it with quotas in the segment Alpha.disk_stat, and prints an error message if unable to force access to any directory. Alpha.disk_stat can then be printed using the disk_stat_print command.

Name: work__class__meters, wcm

SYNTAX AS A COMMAND

```
work_class_meters {-control_arg}
```

FUNCTION

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

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.

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).

work_class_meters

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 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.

work_class_meters

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.

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.

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.

work_class_meters

EXAMPLES

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     0:35:59

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

 0                3.    0.63   3  0.26    2.10    0.26  2.10   P  0  R  Init
 1     1.         0.    0.00   0                               P  0     C1 C2
 2                0.    0.00   0  0.30    0.10    0.30  0.10   P  0  R  IO
 3    31.         5.    0.34   0                               P  0     SysProg
 4     5.         0.    0.00   0                               P  0     Other
 5                0.    0.00   0  0.00    0.50    2.00  1.00   P  0  R  Admin
 6     5.         0.    0.00   0                               P  0     Abs1
 7     5.        10.    1.19   0                               P  0     Abs234
 8    14.         0.    0.00   0                               P  0     System
 9     5.   20.   9.    0.18   0                               P  0
10    10.   13.   6.    0.43   0                               P  0
11     5.         0.    0.00   0                               P  0
12    10.         0.    0.00   0                               P  0
16                0.    0.00   0  0.05    0.05    0.05  0.05   P  0  R

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

APPENDIX A

SAMPLE FORMS

The three forms included in this appendix are samples of forms that have been used at various Multics sites. They are:

```
Project Registration Form
Initial List of Users Form
Person Registration Form
```

Persons requesting registration of new projects on Multics may be required to fill out forms similar to these.

PROJECT REGISTRATION FOR MULTICS SYSTEM

Project Title _____

_____

Principal Investigator _____

Address _____

Project Supervisor _____

Address _____

Phone _____

Each project is assigned a Project_id (1-9 characters long, beginning with a capital letter) for identification and access control purposes. Please suggest a Project_id for your project:

_____

Initial disk quota _____records (default 25)

Acct No. _____ Requisition or P.O. _____

If you wish to administer your project online, please supply the following information:

Directory for PMF _____

Project Administrator (Person_id.Project_id) _____

Project_id assigned _____Date _____

By _____

INITIAL LIST OF USERS ON NEW PROJECT

Please give the full name of all users to be registered on your project. If these users are already registered on the Multics system, give their assigned Person_ids.

Project_id _____

List of Users

Last	First	Middle	Person_id

PERSON REGISTRATION FOR MULTICS SYSTEM

This form is only needed for people who have never been registered on Multics before.

Name _____
Last First Middle

Mailing address _____

Telephone _____

Programmer number _____ (if any)

Each person registered on the Multics system is assigned a "Person_id" (1-22 characters long), which is unique at this installation. The Person_id is usually the last name (beginning with a capital letter) if possible; if someone else with the same name is already registered, the Person_id will be the last name with initials prefixed (e.g., Smith, JSmith, JRSmith).

Default Project_id _____

Registered persons must also be authorized to use a particular project by the project administrator for the project.

Please attach a slip of paper with a personal password. The password may be 1-8 characters long (letters and digits only).

You may change your default Project_id or your password online any time you wish; consult the login command description in the MPM Commands.

Please return this form to:

(Supply appropriate accounting address)

If you have any problems with your registration, please contact (Supply appropriate accounting phone number).

Person_id assigned _____ Date _____

By _____

INDEX

S

T

U

W

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE: MULTICS
PROJECT ADMINISTRATOR'S MANUAL

ORDER NO. AK51-02

DATED FEBRUARY 1985

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

FOLD ALONG LINE

FOLD ALONG LINE

CUT ALONG

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 Kanda Jimbo-cho, Chiyoda-ku, Tokyo
Australia: 124 Walker St., North Sydney, N.S.W. 2060 **S.E. Asia**: Mandarin Plaza, Tsimshatsui East, H.K.

42807, 2.5C585, Printed in U.S.A.

AK51-02