

DRAFT
12.8.69Identification

System Administration

Michael J. Spier

The system administrator has complete authority over the system and its resources. He has the ability to delegate such authority to projects and account group administrators which in turn have authority over users. The system administrator's identity (or at least one of his identities) must be "wired" into the system, i.e., the very initial system administrator identity may not be established dynamically, on line. Further system administrator identities (we talk of the system administrator, in fact that position may be held, for administrative, clerical or other reasons, by more than a single person) as well as all project administrator and user identities may then be dynamically defined, using the tools provided by the system- and user-control modules.

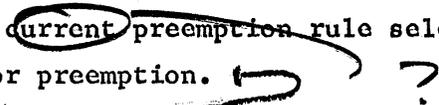
The system administrator defines projects, account groups and registered person identities.

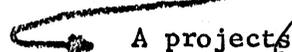
System Administration - Projects

All Multics projects must be defined and registered by the system administrator; he assigns them a unique project-ID, defines the project administrator's scope of privileges (e.g., right to delegate certain administrative authority to some user in his project), grants special privileges to the project as a whole (e.g., a party-group line), and establish^{es} the project's directory and the tree-name of its associated Project Master File (PMF) and Project Definition Table (PDT).

Other responsibilities of the system administrator are to establish the system's maximum capacity (in load units), establish the number of primary lines, and define the "weight" (in load units) of certain types of users.

A load unit corresponds to the burden placed upon the system by a user who engages in modest computation and who restricts himself to the usage of a small set of system-approved commands (in other words, in order to be able to get a minimum ^{amount} of useful service out of Multics, a user has to tax the system's capacity by one load-unit.) An example for a one-unit user is the user who confines himself to the "Basic" system. Normal Multics users, who are expected to make use of the entire spectrum of available commands, are rated as 2-unit users. Certain users, or perhaps certain system daemons may perhaps be rated at 3 units or more (the present implementation does not rate users at more than 2 units.) In order to give a user a 1-unit rating, it must be guaranteed that he will effectively be unable to become more burdensome than his rating implies; this is achieved by starting his process in a special-purpose listener which confines his ability to interact to a small set of approved commands. Thus, an easy and sure way to establish the user's "weight" is to look at his process' initial procedure; effectively, the System Administrator's Table (SAT) contains a list ^(known as user weight table) of procedure names which are known to be associated with non-standard weights. A logged-in user whose initial procedure is in the SAT's user weight table is rated at that specified weight. A user whose initial procedure does not figure in the user weight table is given a default rating of 2 load units. The system administrator defines the system's maximum load capacity in load units, a value which may sometimes correspond to the number of logged-in (lightweight) users, but which normally is not ^{the same as} indicative of the actual number of logged-in users.

The system administrator also specifies a number of "primary lines" which may be assigned to different projects. A primary line corresponds to a guaranteed user-session slot in the system, i.e., a project which possesses a single primary line has guaranteed access to the system for a single user-session, regardless of current system load. If the system is completely full, a login attempt by a user who has a right to a primary line causes the preemption of some logged-in user who does not enjoy this privilege (standby user). The current preemption rule selects the oldest standby user as candidate for preemption. 

 A project may be assigned one or more primary lines. This however does not imply that all users associated with that project are eligible for primary line privileges; rather, a subset of the users in the project is declared (by the project administrator) to have

the party-group line attribute, expressed by means of a state variable in their PDT entries, and only these users may make claims to the project's primary line(s). The primary lines of a project are assigned to that project's party-group line members on a first come, first served basis. When all the project's primary lines are in use, its remaining users having the party-group line attribute are logged in on a standby basis. When a primary line user logs out, his primary line is automatically awarded to the oldest logged-in user of that project belonging to the party line group (assuming that there exists such a logged-in user.) The system administrator defines the maximum number of primary lines in the system. This number must be kept lower than the maximum number of available user-session slots, and should therefore be less than one half of the maximum system capacity expressed in load units. *depend on # of logins versus given primary access.*

cc *predictory* The project administrator has the option of awarding certain privileges ("attributes") to the users in his project. Certain of these attributes are very special, e.g., the nobump attribute which makes a user invulnerable to preemption. There is nothing to stop a project administrator from awarding any number of attributes to his users, however the system does not respect those attributes unless approved by the system administrator. This is done by specifying an "attribute mask" per project which defines the attributes which that project's administrator may award his users.

Eventually, tools will be available with which to update (i.e., add replace or delete) individual SAT entries. Currently the only available tool is that of complete replacement of the SAT. *so?*

Associated with the SAT is the System Master File (SMF) which is an ASCII file containing a section per project, each section corresponding to a project entry in the SAT; a SMF may be modified through the use of a context editor. A dedicated command named "cv_smf" is available for converting the symbolic SMF into a binary SAT. Conversion takes place in the system administrator's user-process; the source file is checked for errors (syntactical, logical or omission). If any error was found, it prints out adequate diagnostics (all errors are detected in a single run) then prints "FATAL ERROR. CONVERSION UNSUCCESSFUL." and terminates. If no error was detected, the command prints "CONVERSION SUCCESSFUL."

By convention, both the SAT and its associated SMF, as well as the newly-converted SAT (which is distinct from the current SAT) reside in directory >udd. The current SAT is protected by 0,0,0 ring brackets in all but the System Control process, making it impossible for anyone except the System Control process to replace an SAT. From access control's point of view, the system administrator is a user who is allowed to use directory >udd as his working directory; the system administration commands are designed around the assumption that the working directory is >udd. A special purpose command, "print_sat", is available to display the contents of an SAT. Command "ship" is available to ship a newly-converted sat to the system control process for on-line installation. By convention, the entry name of a newly-converted SAT is derived from the entry name of its source SMF suffixed by ".sat".

all of
it can

50?

System Administration - Persons

A person is a human being, known to the system by his unique person-ID (also known as "login-name") which is derived from the person's real-life name. A person is also associated with at least one, secret password known only to himself and to system control; the combination person-ID/password uniquely and positively identifies the person. A person's identity must be recorded in the system administrator's Person Name Table (PNT) either as a registered identity which is established by the system administrator, or as a non-registered temporary identity established by one or more project administrators (subject to the system administrator's approval). In order to be able to log into the system, a person must have a user identity, known by its unique user-ID, which implies that person's association with some project; the project is the only approved vehicle for system resource distribution, and consequently a person ^{who} has no user identity is effectively unable to perform any computation whatsoever on the system. For reasons of housekeeping, it is desirable to terminate those user identities which for some reason are no longer associated with any project. Such person identities may belong to transient users such as students, participants in some Multics seminar, etc. Some of these persons (e.g., seminar participants) are known to be one-shot short term users;

distinction
not clear

others (e.g., students) may be known to periodically assume short term user identities; hence the distinction between registered and non-registered person identities. A registered person identity may be established by the system administrator only, and may not be deleted from the system regardless of the person's project association, except by explicit system administrator command. A non-registered person identity automatically gets terminated as soon as it is no longer associated with any project. It may well prove to be an administrative bottleneck to force the system administrator to personally establish all user identity^{ies}, especially in cases such as^{for example,} the short term, large volume effort involved in giving seminar participants a one-week person identity. In such cases, it is desirable to let the project administrator who is in charge of these users ~~to~~ establish for them temporary non-registered user identities, and permit him to afterwards delete those identities without having to invoke the administrative overhead of official red tape. The deletion of a person identity implies the mandatory search of the entire file system hierarchy to reset all access privileges granted to that person. The reason is that in the future a different person may be registered with that same personid, and access files which by rights should be inaccessible to him. In addition to the concepts of "registered" and "non-registered" person identities, there also exists the concept of the "unauthenticated" person identity, whose identity has not been validated by login control, and is not even entered in the PNT. Such persons may still log in as users of specific projects, at the projects' cost and risk.

Eventually, tools will be available with which to update (i.e., add, replace or delete) individual PNT entries. Currently the only available tool is that of complete replacement of the PNT. This implementation restriction implies that only the system administrator may change the PNT, and that the optional mechanism for the establishment of non-registered person identities may not currently be implemented.

Associated with the PNT is the system administrator's Person Master File (PERSMF) which is an ASCII file containing a section per person; a PERSMF may be modified through the use of a context editor. A dedicated command named "cv_persmf" is available for converting the symbolic PERSMF into a binary PNT. Conversion takes place in the system administrator's user-process; the source file is checked for possible errors (syntactical,

*also - does not use
to*

*It's non-rg. was known to
admins wanted? I think not.*

(6)

logical or omission). If any error was found, it prints out adequate diagnostics (all errors are detected in a single run) then prints "FATAL ERROR. CONVERSION UNSUCCESSFUL". If no error was detected, the command prints "CONVERSION SUCCESSFUL". By convention, the entry name of a newly-created PNT is derived from the entry name of its source PERSMF suffixed by ".pnt". By convention, both the PERSMF and its associated PNT, as well as the newly-converted PNT (which is distinct from the current PNT) reside in directory add. The current PNT is protected by 0,0,0 ring brackets in all but the System Control process, making it impossible for anyone but the System Control process to replace a PNT.

I would have included a new directory, e.g., add

System Administration - Account Groups

The Multics account group is a collection of Multics accounts which share some common denominator, where each individual account normally represents the financial resources allocated to some user (or group of users). Two possible approaches for ^{account} grouping are a) financially oriented, i.e., all accounts which draw on the same financial source (budget), and b) administratively oriented, i.e., all accounts of a specific group of users. Other approaches are conceptually possible, however only the two methods stated ^{above} may be meaningfully applied under the present Multics system control implementation. The normal Multics administrative usage is to establish an account group for all users of a single project (b), but as projects are normally independently financed, the same account group also corresponds to accounts drawing on a single financial source (a). Normally, then, the terms "project" and "account group" are in some way synonymous; the account group table (AGT) normally resides in the project directory, and the project administrator and account group administrator are normally one and the same person.

?)

Associated with the account group is a master account representing the total resources allotted to the account group; the sum of the resources of all component accounts within the group may not exceed the size of the master account. From the implementation point of view, a master account structurally resembles a normal account, the only difference is in the size of allocated resources. The resource allocated to an account is an amount of dollars which the user spends through system usage. System usage falls into two major categories,

(7)

oh?

a) charges for permanently allocated resources (registration fee, secondary storage^{etc.}) which are time-dependent and not directly related to any specific user session, and b) charges for resources which are temporarily allocated to a specific user session (cpu usage, core usage, I/O device usage, terminal connect time etc.). It follows that accounting for type (a) charges should be done periodically whereas accounting for type (b) charges must be done during the user session itself, and consequently both types of accounting need be handled separately. Therefore, an account consists of two major items, the "quota" which is the money allotted ^{for} to type (a) charges, and the "resource" which is the money allotted ^{for} to type (b) charges. Furthermore, type (b) charges vary, depending upon the time at which the user session is held, in order to entice people to use the system less during "rush hours" (e.g., workdays from 8am to 6pm) and more during the less popular "slack hours" (e.g., nights, weekends, holidays). The price discrepancy may be in the order of several hundred percent, and in addition the ^{absolute} price of execution (in terms of cpu time) may be lower during "slack hours" due to less competition among users (less ^{fewer} page/segment faults, less process unloading/deactivating etc.). As a result of this pricing policy, a given task (in the sense of "user-project") may cost n dollars if carried out during the most expensive shift, and only n/k dollars (where k may be non-trivial) if carried out during a less expensive shift. The thrifty account manager may well wish to capitalize on this and allocate to that task only the smaller amount of money; such allocation is, however, meaningless unless the user is restricted to using only the less expensive shift (if he were admitted into the most expensive shift, he would only accomplish $1/k$ of his task before his money ran out). Therefore, in the implementation, type (b) charge allocations are expressed in terms of per-shift sub-allocations.

b2d

To summarize, an account (be ~~it~~ a master-account or a regular account) represents a certain amount of allocated money (say \$1000) which is paid for by some financial source. The account's administrator first divides the money into (a) and (b) charge allocations (say \$500 for permanently-allocated resources and \$500 for temporarily-allocated resources), then proceeds to subdivide the (b) charge allocation into per-shift sub-allocations (say \$200 for shift-1, \$200 for shift-2, and \$50 each for shifts 3 and 4). In the present accounting control implementation, an account consists of 5 allotments: a quota and four resource allotments corresponding to shifts 1-4 as indicated by the example above.

Temporarily-allocated (type (b)) resources are dynamically managed by the system, on line, and awarded to users on a first-come first-served basis. Permanently-allocated resources (type (a)) must be distributed among users in advance, regardless of actual usage. Of these resources, secondary storage record capacity is finite, and the amount distributed must not exceed the system's capacity. Therefore, associated with an account's quota (dollars) there is a record quota (number of records) allocated to that account. Record usage is charged against the money quota, and a special charge is made for the record quota reservation, regardless of whether or not the entire quota is actually used. *Less than use, I presume.*

Accounts are further divided into two parts: a) the allotment, and b) metered usage. All system usage is metered and charged, as explained above, against the account's allotment. When usage equals or exceeds the allotment, an error condition is established which results in one of the two following actions:

a) if the ^{excess} was in the record quota, i.e., user is still financed and is only of trying to access more than his allocated share of the system' resources, then an error condition is signalled within his process, but b) if the excess occurred in an actual money allotment, then the user is no longer financed and an "out-of-funds" condition is signalled to the user overseer which terminates the user session. The out-of-funds condition applies only to the actually depleted allotment, i.e., if a user's shift-1 allotment ran out and his session has been automatically logged-out because of it, he may still login later in the day on his valid shift-2 allotment.

The account allotment and the account usage are kept in separate tables. This allows the administrator to modify allotments and shift resource allocations within the domain of his privileges, without affecting the already accumulated usage data. All allocation tables may be modified on-line, and ^{modifications} take immediate effect without affecting any currently logged in user session, except by forcing a premature out-of-funds condition.

There is a single per-system master account table (MAT) with an entry per account group. It is located in >udd and is accessible to the system administrator only. It has no corresponding usage table. There is an account group table (AGT) per account group, to specify allotment, and an account usage table (AUT) to record usage in that account group. Normally, both the AGT and the corresponding AUT reside in the associated project's project-directory. Tools are provided for the dynamic on line updating of the MAT and the AGTs.

A master account represents the resources, in terms of dollars and secondary storage record quotas, allocated to (or rather, purchased in behalf of) an account group. A master account is an entry in the master account table (MAT) which describes the available system resources and their distribution among account groups. The MAT resides in `>udd` and is accessible to the system administrator only. It contains two major types of allocation, money allocations which allow users to request system services on a competitive basis, and permanent allocations of finite system resources (secondary storage record quotas) which are preallocated by the system administrator and which may not be exceeded through user competition.

Associated with the MAT is the system administrator's Master Account Master File (MAMF) which is an ASCII file containing a section per master account. A MAMF may be modified through the use of a context editor. A dedicated command named `"cv_mamf"` is available for converting the symbolic MAMF into a binary MAT. Conversion takes place in the system administrator's user-process; the source MAMF is checked for possible errors (syntactical, logical or omission). If any error was found, it prints out adequate diagnostics (all errors are detected in a single pass), then prints `"FATAL ERROR. CONVERSION UNSUCCESSFUL."` and terminates. If no error was detected, the command prints `"CONVERSION SUCCESSFUL."`