Resource Control & Accounting on Cal TSS

I. Resources
   Some are controlled strictly on the basis of the user's ability to pay for them; there are artificial limits necessary for efficient operation of the system.
A) CPU time [W2]
B) Peripheral Equipment
   1) Fixed
   2) Swapped?
C) Disk space
   1) Permanent (words * ?)
   2) Temporary

D) ECS space
   1) Memory (words * 1024)

II. Accounting Considerations
A) User profile
   The user negotiates his profile with the CC business office, resulting in data known as the user profile. The profile contains:
   1) # of resources he may use
   2) $ spending limit
      a) written notification?
      b) warning at each login?
   3) Δ CP
      a) # users who can't use more in one session
         b) notification at Δ CP - may cream the stack!
   4) fixed ECS limit
      a) end limit
      b) default
II A) 5) **Employing ECS?**
   a) cell limit
   b) default
   c) time * cells

6) permanent disk space. Renegotiated with 20) Campbell
7) temporary disk space? a) limit
   b) default
8) access to peripherals
   a) line limit
   b) card limit
   c) tape mounting limit

II B) Nature of the user's accounting report

1) Accounting points. User interactions with the system which indicate changes in the resource nature and/or rate of his resource utilization. These interactions may be triggered by the user at a keyboard or by his running code. Accounting points are
   a) logon
   b) changes to ECS profile
   c) " " disk
   d) renewal of DCP?
   e) requests for peripheral service
   f) logoff
2. Dayfile. A file is maintained by the system into which entries relevant to system activities are made from time to time. An entry is made at each user accounting point, including the real time, a description of the AP.

3) Users report. At some time interval selected by the CC office (one month), the dayfile is processed to prepare reports for the individual user. The report can potentially be incredibly detailed, including every AP detail is left to the office to decide.

4) Actual charges are based on the report presented to the user.

5) Approximate account:

C) General philosophy of the Charging Algorithm. The system imposes a "pay in advance" attitude. That is, whenever a user is about to utilize a resource, he presents an upper limit for the task he is about to do. The system compares this limit to a $ charge and decrements the user's account. If his account is insufficient, the user is informed and may take appropriate action, i.e., curtail the request or obtaining more $. If the account is sufficient, the task is allowed to proceed until it terminates or reaches the limit. In the former case, the user's account is refunded the appropriate amount by $ in the latter case.
II C) 1) (cont.) If a user task reaches the limit he specified, he is so informed and is allowed to extend his limit whenever it is practical to do so. If it is impossible to communicate with the user (as in \* ?? editor), the task is, alas, terminated at the specified limit with some hopefully illuminating diagnostic.

Example:

a) ACP
b) Remote file print requests
c) Permanent disk storage
III. Mechanisms underlying Brief description of process allocation models.

A) Time

1) The 6400 clock attached to the 6400 is read periodically by a PP (process) in a cell. S.MAS.T (S.MAST) in central memory. S.MAST (S.MASTR) is the basis of all user deadlock avoidance. S.MASTR (S.MAST) 

2) Each partition of its own activities into 5 categories, each category being accumulated as it occurs in a certain cell in CM.

3) Each process in the system has a set of chores which round up the CPU time consumed by that particular process in various categories. An operation allows higher-level systems to display these chores of any process for which a suitable capability is possessed. The CP gives...
Timer enforcement. Each process contains a timer of a message channel

III A) 4) Process timer. Each process in the system contains a timer. Time
may be transferred to/from the process timers by allocation
mark to which the process is charged. Whenever a process is
swapped out, its timer is decremented by the amount of
CP time it has consumed; if its timer has run out, it is
suspended and the Process Manager is notified by the
swapper. Incoordinated deals with the Process Manager may
determine to reactivate a process descheduled for timeout. Other-
wise, it will remain dormant until it is destroyed or until some
other process bails it out. Retention operations:

a) more time to process

1) IP1  C: AB (OB.TIM)
   2  C: Process (OB.TIM)
   3  ST 30
   error if ST not available in AB;
   if process is currently descheduled for timeout, it is rescheduled;
   processes are created with timeout

ib) more time from process

1) IP1  C: AB (OB.TIM)
   2  C: Process (OB.TIM)
   3  ST 30
   if timer TST, process descheduled & caller notified of T (Preset?)

* The Process Manager has good vibes (small $n_i$) & runs next? ???
III A) 4) iii) move terminate AB
   I) C: AB (OB.TMF) from
   2) C: AB (OB.TMT) to
   3) DT
   error if DT not avail

note: perhaps time moves should be restricted to part of AB?

when a process is destroyed, its times are given to its owning AB.
when an AB is destroyed, all its processes revert to its owning AB.

III A) 5) process supervisor includes or process it may not receive request for prior process

Deals with the Process Manager:

i) send time
   C: AB
   D: DT

ii) send request
   C: process in question
   C: process to receive request
   D: subprocess
   D: request data

iii) send time & request
   C: process

note: Process supervisor forgets a request if it gets an error trying to comply (no AB, not enough, perhaps it makes auguay shuts?)

If its request queue over, or every DT in any case, DS probes its requests & deletes ones which are no longer relevant.

If the request queue really over, is it? If communicated with dupicate requests?
Tree structure on processes?

(interrupt father on timeouts? Xfer time to descendants + ancestors?)
III A) 6) Allocation blocks. Allocation mechanisms serve several purposes:

a) control the amount of space available for different users

b) accumulate information on charges for usage

c) provide a structure on objects occupying ECS so that which will allow cleanup of objects (even when the creator of the object has run out of problems). HeadTail

d) serve as a repository for CP time

e) operations (in addition to III A 2 a i iii)

f) AB dispatch

1) create

2) destroy

3) display

4) read charges

v) request reserved space to son

C: data AB

C: son of data AB

D: space

demands reserved + charging

movements reserved in path

vi create capability for new object

vii movement changed space (or steal reserved space?)
It is hoped that ECS is cheap enough that spying on ABs at an interval of say 3-10 mins is feasible. In any case, ABs run at regular intervals, say 3T_{ABs}, and store the prescribed interrupt for each one that has passed its limit to delete the request. No duplicate requests? (Then user can't spy on his AB.)
System use of control and accounting features. When a user logs on at
system A or a TTY, the LTP software creates a user's profile. It uses datamart to create

- An AB with released fixed ECS
  $CP_u = \Delta CP_E$
- How wrapped ECS ?!
- A process with times $\Delta CP_E$ taken from the AB

1) The system deletes $\$ from the user account for:
   a) $\Delta CP$
   b) $3^{\text{rd}} $ (fixed ECS * $\Omega_{AB}$)
   c) wrapped ECS?
   d) Temporary disk space?

2) If the account is sufficient, the user is system
   a) creates an AB, the USER AB, containing
      i) $\text{Charged} = \text{Reserved} = \text{fixed ECS from the profile}$
      ii) $CP_u = \Delta CP_E$
   b) creates a process, the USER PROCESS, charged to the USER AB,
      sets the process times to $\Delta CP_E$ from taken from USER AB
   c) the process then starts. Constructs its system sub-process trees
      iii) makes a request on the AB tag
         iv) makes a request on the AB tag

Fixed ECS AB?
III A) 3) If the account is insufficient the user is informed. The system may or may not offer a barter with the user to let him make another request to use his last small amount of time.

IV b) SCP enforcement:

Whenever a timeout interrupt arrives, the system sub-process responsible for accounting first checks whether the user has requested local control (see below). Actions taken when local control is in effect are described below. When not under local control, the user profile and/or keyboard requests determine what happens.

1) Automatic renewal (requested from TTY, but possibly not allowed by profile). Adjustments are made to the user account by charging for another SCP, modified by any remainders of the previous SCP not actually consumed.

IIIIIIIIIIIIIII

IV B) 2) Optional renewal: The user is informed of expiration of CP two
7 allowed to renew, as in 1, or log off.
3) Forced logoff: Some users may only be allowed to use SCP
A one sitting?

IV C) ECS trigger space enforcement. The DTS field of the AB is
read & the user account is charged for the amount read.
1) If the account is sufficient, a new request is posted
with the AB supervisor & processing resumes (thought of as
automatic, although user options in IV B 2 are possible).
2) If account not sufficient, the user is informed that he
was DT AB5 to terminate gracefully & a request is
posted with the AB supervisor. If the user fails to
logout before the interrupt arrives, he is involuntarily
logged out.

D) Strapped ECS?
E) Disk space being cheap, it can merely be charged for
by some crude algorithm. Permanent disk space can
be charged for X months in advance. Then when the
user's account is depleted, he can replenish his account
or preserve his files on tape within X months without
fear of any later dire consequences. The code doing the charging need only
inform the CC business office when the account runs dry.
2) Temporary disk space may be charged in any format in advance
in terms of any convenient time interval \[ \Delta T_{ABS} \].
The code then need not detect account deficiency, as SCI or
OCS will inevitably handle the problem shortly.

F) Logout. The objects attached to the user AB are deleted
from the system and any resources refunded is made to
the user account for any of the resources charged
for under IV A 1 but not yet consumed.

G) Peripheral usage:
1) Remote printing, etc.
2) Use of printers, tape, etc.
User access to control and accounting features. Facility for the user to control the CPU time and/or ECS and/or disk space used by some subtask initiated during his run is fairly general, hopefully convenient. The mechanisms described in III are available to him with only modest restrictions.

A) CPU time

1) Suspicious subtask usage. If about to call a subsystem which he does not trust, he may request an interrupt after $\Delta t$ ms of processing, either from the TTY or by a system call from his running code. The system will thereupon stop the timer, move all but $\Delta t$ of the USER process times back to the USER AB. *Note that the process is under local control.*

*If* when the interrupt arrives, the process times is restored to the user informed & presented with the option of continuing or terminating.

(with or without a wait-st)

2) Satellite processes

* Local control - CPU time in USER AB?
process

The next level is to define the necessary allocation between the processes. Each process is assigned a specific role and resources. This allocation is critical to the success of the system.

In smaller systems, this allocation can be done manually. However, in larger systems, automated processes are necessary. Also, it must be defined how resources are shared among processes.

They are chained together.
1. process

The next
find 1st unused allocation
block that is "special"
lead in front to designated
process & suspends

describes the run out process.

in an allocation block that is special
a process & a cross over on interrupt status
also some allocation blocks are "natural"
They are chained.