12/16/70
Bricklin

Limited Service Systems

It is often desirable to limit the resources available to Multics users. A small budget may have to be distributed evenly among many people, or students restricted to a subset of the Multics environment. One of the subsystems which is designed to limit the user is the Limited Service System (LSS). It limits both the rate at which cpu time is used and which commands are available. A description of how it works should help provide an insight into the design of limited systems in general.

When a Multics user gets to command level after logging in, he is pretty much free to use the system as he sees fit. Therefore, it is important that the limitation machinery be started before he gets to command level. This is done by providing a special login responder. A login responder is the subroutine called by the initializer after the user's process is created. The standard login responder sets up some condition handlers, then calls the listener. If a user is designated an LSS user, the LSS login responder is called instead of the standard one. It does the same setting up as the standard login responder, then starts the limitation machinery. It calls a special entry in the command processor telling it that it is to operate in the LSS mode and gives it a pointer to a table listing the commands to be allowed. It also starts a timer which will cause an interrupt after a few seconds of cpu time have passed. Then it calls the listener.

When the user types a command, the command processor, when in the LSS mode, calls a subroutine to check the legality of the command before executing it. If the command is not in the table provided by the login responder, an error message is printed and the command ignored. This method of limiting commands is very efficient and provides the user with the full power of the command processor without any duplication of code. A disadvantage is that it does not limit what the user's programs may do. If the user's program can manipulate the stack, it can modify the condition handlers, the limitation data bases, etc. Therefore, the commands allowed must not allow the user to do anything which could override the limitation machinery. Programs that the user can run written in basic can be allowed, while those written in pl1 may not.

When the cpu timer causes an interrupt, it is handled by the LSS cpu limitation subroutine. This subroutine checks the ratio

of cpu time to real time used since the timer was last set. If this ratio is within the range allowed in the LSS, the timer is set again, and the process is resumed. If too much cpu time was used, the process is blocked a sufficient amount of time to bring the ratio to an acceptable level before resuming. A special quit handler is provided during the time the process is blocked to inform the user how much time remains until the process will wake up and respond to the quit. This "percentage of cpu" limitation guarrantees that no more than a certain amount of money can be spent per hour. It also puts a maximum on the load the user can place on the system.

The techniques used by the LSS can be extended to provide limited systems with further capabilties. For example, while it is checking the cpu ratio it can be doing special accounting, logging out the user if he used up his allotment. This would be independent of any accounting done by the standard Multics machinery.

Sometimes it is desirable to interrupt periodically on a real time basis. Interrupting on cpu time, like the LSS does, means that checking is only done when the user actually is doing something, and has no overhead if he leaves the console unattended. Interrupting to check cpu usage on real time, say every minute, is a fixed overhead. It is often useful since there are things that should be done on a real time basis. There may be a limit set on the connect time, or perhaps a check should be made in a common data base for interprocess messages. This is useful when some connection is desired between users of a limited system.

The environment provided by the standard command processor is not always appropriate. The limited system designer can have his login responder use a different listener or command processor. There are many possibilities made possible by the flexibility of the Multics system.