Timing Out

BY SAM GOLOB

Chances are, most of you haven't had the system programmer's "silver spoon" in your mouth for your entire data processing career. In other words, you probably haven't had "PC (Privileged Character)" status. One of the features of PC status is that your own TSO session does not "time out" and get logged off with a System 522 ABEND after a designated number of minutes of inactivity. You can leave the office and stay logged on all day.

From the administrator's point of view, this privilege of not timing out is undesirable for the "average" TSO user. Therefore, in most installations, such power is not given to these users. Why is it better that an average TSO user should not time out? First, any logged-on TSO user allocates a certain number of datasets and ties them up with a shared enqueue, so no one else can get an exclusive enqueue to these datasets if needed. Second, even a swapped-out TSO user is tying up an address space and some system resources; this user might even stop a new user from logging on, if the MAXUSERS number (of logged-on users) is being approached. Third, if that TSO session is being hung while no one skilled is around, it would be very difficult to free the session. You can probably think of several other reasons that an automatic session time-out would be helpful for most TSO users. Of course for us PCs, none of these reasons apply, since we can fix any problem we might cause. Yeah, right. Anyway, this month's column will examine this idea of timing out.

X22-TYPE ABENDS

My opinion is that this subject should really start with a discussion about purposely killing an address space. IBM has graciously provided us with several means of "chopping off" jobs, started tasks, and TSO sessions in the middle of their activity. We've all heard of canceling or forcing jobs and TSO sessions. When a job produces excessive output, we might want to chop it off before it floods our printing resources or JES spool space. CPU time excession ABENDs have occasionally been part of our experience. Sometimes, we run a big SMP job in the wrong class and it exceeds that class' allowable CPU time, getting cut short in the middle. All of these "cutoffs" are related by the fact that the resulting system ABENDs nearly all end with the characters '22'.

What are some of these System X22-type ABENDs?

- A System ABEND 122 occurs when the operator cancels a job, requesting a dump.
- A System ABEND 222 is an operator cancel without a dump, which is very common.
- A System ABEND 322 is a CPU time excession cancel of a job.
- A System ABEND 422 is new, and is similar to a 222, but it is done by an Open Systems MVS application.
- A System ABEND 522 results from excession of the Job Wait Time (JWT) parameter in the SMF parms in SYS1.PARMLIB. This is the "timing out"ABEND we will examine this month. A System 622 ABEND occurs when something goes wrong with TSO, or your session gets disconnected from the terminal and the reconnect limit time passes.
- A System 722 ABEND occurs if the job puts out a lot of output, exceeding the JES or JOB card output limits and the OUTLIM parameter was not coded in the job's JCL to override these limits.
- A System 822 ABEND occurs when the region requested for a job step cannot be obtained.
- A System 922 ABEND occurs if the INITIATOR is now in control of a job, before or after the job step has taken place, and an ABEND or other interruption occurs.
- A System A22 ABEND occurs after a FORCE operator command was issued, and the address space was "burned", or chopped off forcibly, no matter what it had been doing before.

We have faced nearly all of these ABENDs in our careers. Sometimes we are happy when one occurs. Other times, we can be very disappointed. This month, I'll talk about a few tricks we can do to avoid some of these disappointments.

Please remember that, by and large, we are PCs (Privileged Characters) in our profession. However, sometimes management feels even we PCs need to be restricted. Nevertheless, we still are the system doctors, and we still need our tools to do the job that management asks of us. This scenario is similar to what the parent demanded of the school bus driver: "Be careful and drive slowly, but get my child to school quickly." The bus driver still has to get that child to school, safely, despite the parent's protests. And we have to do the job that management requires of us, safely, but quickly too, without unnecessary hassles and interruptions.

So if management does not allow us to stop our sessions from timing out with a S522 ABEND, and the job requires that we have to run several parallel sessions on one or more systems for an extended period of time, without their timing out, what option do we have? How can we circumvent our management's restrictions and still do the job asked of us?

WAYS OF AVOIDING SYSTEM 522 ABENDS

A System 522 ABEND is often caused by a TSO session's inactivity. How long it takes to trigger the time-out depends on the setting of the Job Wait Time (JWT) parameter in the SMFPRMxx member of SYS1.PARMLIB (in minutes). All non-altered TSO sessions will time out in this determined number of minutes. How can we override this time-out in our own TSO session and not affect the average TSO user?

One way to avoid the time-out is to have your own TSO logon procedure in a PROC library. This would also require that SYS1.UADS or a security package (i.e., RACF, ACF2, TOP-SECRET, etc.) be set up so the use of your own procedure would be allowed when your ID is used to logon to TSO. Your own logon procedure, which, in effect, is JCL, can be then coded with TIME=1440 or TIME=NOLIMIT in the EXEC card which invokes the terminal monitor program (usually IKJEFT01 or ADFMDF03). This will override a short JWT setting and keep your own TSO session "on the air".

Some installations do not allow this, although nowadays it is unusual to find such "backward" places. These installations, while recognizing that systems programmers have different requirements in their TSO sessions than application programmers, will allow one logon PROC for application programmers and another one (or several) for systems programmers. They will not allow, however, a systems programmer to have a private logon PROC. And they will also not allow TIME=1440 or TIME=NO-LIMIT to be coded in any of these PROCs. In many places, the systems programmers are not the security administrators. What can we do now?

A common solution is to find (or write) a program which runs continuously in a TSO session if nothing else is running, and that has a built-in timer. For example, the timer will pop every five minutes. When the timer pops, the program triggers some small TSO activity to fool the session into thinking that it is not inactive. Therefore, the session will never time out.

There is a way to solve this problem with management's approval. If your installation sufficiently recognizes the importance of certain users not timing out, there is another possible approach via the SMF exit IEFUTL that can be used to control the effect of Job Wait Time on jobs, started tasks, or TSO sessions. An interesting and general solution that can be implemented with management approval operates through RACF or whatever security package you have. I'll show an example using RACF. You define an entity to RACF in the FACILITY class called TSOWAIT. Those users who will not time out are given READ access to this entity. Then an IEFUTL exit can be coded, which does a RACROUTE call for the TSO user to ask if READ access is granted to this entity, and if so, an infinite time extension is given to the TSO session. Management administers this solution officially, through the security administrators, and everything is above board.

One of the features of PC (Privileged Character) status is that your own TSO session does not "time out" and get logged off with a System 522 ABEND after a designated number of minutes of inactivity.

This IEFUTL exit can be quite easily coded. An example can be found on File 245 of the CBT MVS Utilities Tape, a huge, independently produced tape of MVS goodies available through NaSPA. The IEFUTL exit in File 245 can be simplified even further. Besides the RACROUTE call there is very little to it, just either a time extension or a bypass of the time extension, based on the result of the RACROUTE call. If you have other IEFUTL exits in use at your installation, don't worry. SMF exits can be easily piggybacked, so they all run in succession.

ANOTHER SOLUTION

Another clever solution can be found on File 183 of the CBT MVS Tape. This solution requires that ISPF be running in your TSO session. To implement this solution judiciously, not including too many people in the timeout exemption, you should have a restricted or a private load library to put into the ISPLLIB concatenation. This can be done dynamically in a CLIST using the ALLOCATE TSO command, even if your TSO session is using a public logon PROC. Just exit ISPF, FREE FILE(ISPLLIB), and issue an ALLOCATE command with the proper ISPLLIB libraries concatenated in the proper order. That can all be done in one CLIST. Finally, assemble and linkedit the source code for the ISPTASK module on File 183 (non-reentrant and non-reusable) into this private library. With ISPF running, your TSO session will not time out during weekdays, until 7 p.m. on Friday. Examine the source code of the ISPTASK module on File 183 to see how this works. You can alter the source code if you have different timeout exemption requirements.

The user version of the ISPTASK module from File 183 is designed to front-end IBM's ISPTASK module, which is either in the link list or in LPA. IBM's ISPTASK module is re-entrant; this user front-end is not. The user ISPTASK, as coded, works as follows:

1. Upon getting control, it pre-loads a bunch of ISPF modules into the Job Pack Queue for more efficient operation if these modules were not put in LPALIB, but were running from the link list. This is not important for our purpose, but I'm just mentioning it for completeness.
2. Afterwards, our user ISPTASK determines what day of the week it is (Monday through Friday) and conditionally passes control to the STIMER routine that jogs the TSO session into "activity" every 10 minutes. It only does this from Monday at midnight until Friday at 7 p.m.
3. Finally, it does a lot of searching through TCBs, looking for IBM's actual copy of ISPTASK, avoiding ISPLLIB and STEPLIB, looking for a re-entrant module.
4. Upon finding the proper copy of ISPTASK, it XCTLs to it. Meanwhile, the STIMER keeps popping itself every 10 minutes, and the TSO session does not time out, because of this "activity".

I hope you have found this month's discussion interesting and informative. Perhaps you will find alternative ways to do this job. I've always thought that systems programmers were a clever bunch. Good luck, and keep those timers popping. See you next month. ts

NaSPA member Sam Golob is a senior systems programmer.