Setting Yourself Up

BY SAM GOLOB

Every experienced systems programmer wants to have as many tools as possible available and ready to use. If you've been at the same shop for a long time, you've probably already set yourself up with special programs and gadgets so you'll be able to do your work more quickly and easily. The whole "kit and caboodle" of your favorite tools has been installed, tested, and is accessible to your userid, ready to be deployed at a moment's notice. However, if you're new to a shop, or you're a consultant who travels from one shop to another, setting yourself up with an appropriate set of tools can be a chore. You might feel at a disadvantage because you can't use the wonderful work-saving tools you're familiar with.

This month, I'll present some tips about how to set yourself up quickly. I have my own experiences doing this, and I'd appreciate if you would email me your experiences, too. My teacher Jeff Broido has been a consultant ever since I've known him, and his clever inventions enable him to set himself up anywhere, at any shop, within one to several days.

When setting up your tools, there are several things you must consider: First, as a "system doctor," you need more privileges than the average user. Most times, the shop will give you a "systems programmer's id," which is a TSO userid that already has what management deems to be enough extra privileges built in, so you can do your special job. Other times, a shop will not "trust" a consultant, and they might even handicap him by giving him a userid with little more than an application programmer's power. Then they'll expect the consultant to do the system doctor's job just the same. Consultants often have to deal with such politics, under such conditions, and sometimes it's better to be able to do your job, quietly and efficiently, without butting heads with management more than necessary.

When "setting yourself up," there are various things in your TSO environment that you'll need to alter. Since you're special, you'll want to change the average TSO user's parameters as little as possible — preferably not at all. For example, you'll want to avoid public PARMLIB changes if you can. There are a few shops that require several "manager sign-offs" to change just one PARMLIB member. Often, if you know some tricks, you can quietly achieve the same result for yourself (on your own userid) as if you changed something in PARMLIB, but you don't have to do it "for the public." To me, this way is greatly preferred.

I'm a firm believer in removing as much JCL (that is, DDNAMEs) from the logon proc as possible, and instead, allocating files via the initial logon CLIST or REXX exec.

So, let's see what we need to do under various degrees of "management restraint." I can't say that I have the final word on the subject. I don't know everything — that's why I need your email input. However, I'll talk about the subject in general and discuss some pointers along the way. There are really two divisions to this subject: 1) How do you set up your own TSO environment and 2) What do you take with you and how do you package it?

Finally, no discussion of MVS tools is complete without mention of the wonderful world of free MVS tools and utilities. The granddaddy of free tool collections is the CBT MVS Utilities Tape, together with the CBT Overflow Tape. These tapes are independently produced and are not subject to the restrictions of any organization. Some of the tools found there have no equal, even among vendor products. I write a lot about materials from the CBT Tapes in this column because the tools are free and can prove invaluable to most shops.

Where can you get the CBT Tape materials? NaSPA incorporates a version of both CBT tapes into its annual CD-ROM. The CBT tape itself is updated more frequently. Over the years, NaSPA has played a key role in the distribution of CBT materials on both tape and CD-ROM. Now, NaSPA is also sponsoring a web site that has all the latest CBT materials in downloadable format for quick fixes.

While it's good to have as many mediums as possible available, I find the CD-ROM is especially useful because it's portable and can be taken anywhere. Additionally, I don't need Internet access to use it, and it doesn't require lengthy downloads. The CD-ROM source files are readable in ASCII on a PC. The tapes and the web site materials are in EBCDIC. On the web site, the files are PKZIPped and in TSO XMIT format. You upload a file to the MVS system after unzipping it and do a TSO RECEIVE on it so it's completely MVS-ready. Both resources have their own advantages, and it's nice to have them all. The NaSPA-sponsored web site is at http://www.cbttape.org.

MODIFYING YOUR TSO ENVIRONMENT

Now, let's get started. I'd say that the one thing you should ask for when going to a new shop is whether you can use your own logon procedure in a system PROCLIB. If you can do that, and you have update access

to an authorized load library, you're in business. Copy the normal systems programmer logon procedure to your own named one. Carefully check what's already in the authorized library and if you don't see anything there that might interfere with your TSO session, add it as a STEPLIB ddname, with DISP=SHR of course, to your own logon procedure. You have to be very careful about who else uses the authorized library. You shouldn't put anything else in the library that would mess up someone's work. It's best if you have your own systems programmer authorized library so you stay out of everyone's way.

The IKJTABLS load module will completely override the IKJTSOxx parameters in PARMLIB if its load library is known to a TSO session as an authorized STEPLIB. Remember, even if this authorized library is used as an ISPLLIB by other people, the overriding of IKJTSOxx in PARMLIB will not occur for them. Authorization under STEPLIB will be ruined if there is a non-authorized library in the STEPLIB concatenation, so be careful that all the libraries in the STEPLIB of your logon proc are authorized, and omit the ones that aren't. If ISPF is involved in your work, consider whether the deleted libraries could be included in your ISPLLIB concatenation.

I'm a firm believer in removing as much JCL (that is, DDNAMEs) from the logon proc as possible, and instead, allocating files via the initial logon CLIST or REXX exec. If one of the datasets is gone from the system, if it's inaccessible or it was renamed, you can still have a TSO session, and you can make a different CLIST to recover and get to work. If the dataset that disappeared was part of your logon proc JCL, your TSO session will fail on a JCL error, and you won't have a session at all. It's much safer to get your files via the TSO ALLOC command rather than with logon proc JCL. So, get a sample logon CLIST ready for insertion into your SYSPROC concatenation under TSO. When you go to a new shop, just change the dataset names.

My teacher, Jeff Broido, carries a tape with him that contains all of his own stuff and personal libraries. After he loads these down to pdses on a new system, he customizes his TSO allocation CLIST to put them into his TSO session in the right places: ISPPLIB, SYSPROC, ISPLLIB, ISPMLIB, and so forth. It doesn't take too long.

Jeff doesn't like to use his personal programs if he hasn't assembled them at the new shop. For this purpose, he wrote a CLIST he calls A (to keep the name short), which assembles a program from one of his source libraries in the foreground under TSO. If the program isn't extremely long, the SYSTERM result comes back to his terminal in a relatively short time. In a couple of minutes, Jeff can reassemble and re-linkedit quite a few of the programs he relies upon for his daily work. This procedure saves a lot of time when he's setting up. I'm putting a copy of Jeff's old "A" CLIST on the web site http://members.aol.com/freepds.

With just one invocation of the SHOWMVS load module as a TSO command, you'll get so much information about your system and your own TSO session that you probably won't need much more.

SETTING UP YOUR OWN ISPF COMMAND TABLE ENTRIES

Your own personal ISPF command table entries can make life much easier. If you make a personal version of the system ISPF command tables, especially the ISPTABL or ISRTABL members, depending on how your shop is set up already, you can save yourself a lot of work later. Among other additions to the normally supplied table entries, I like to set up a way to instantly get any standard IBM ISPF panel. For example, I'll pick a little-used command prefix, say SP. Then, if I want an instant ISPF 3.1 screen, on top of anything I'm doing — say I'm in the middle of editing a file — I just enter SP31 on the command line and the ISPF 3.1 screen pops up on top of my edit. I use the ISPF 3.1 screen to do what I want and then I'll end the session and go back to editing my file. I like to set these up for all of the standard ISPF functions. How can we go about doing that?

There are two table library DDNAMEs in your ISPF dataset concatenation. The ISPTLIB concatenation contains ISPF tables that are read-only. Your ISPF session will recognize and use these tables. The updates to tables are done in a pds that is in the ISPTABL DDNAME concatenation. If you want to update a table using ISPF 3.9, the table must be a member in a library that is allocated to the DDNAME of ISPTABL. Suppose you want to update the system member ISPTABL which everyone uses. You want to add command table entries to it that only you want to have. What do you do? You shouldn't or (depending on the version of ISPF you have) can't edit the system ISPTABL or ISRTABL member directly. You can, however, make a copy of the system ISPTABL member for example, into your own table library pds, and call it some non-system name, ending in ..TABL, such as XSPTABL. Then you do ALLOC FILE(ISPTABL) DA(your.table.dataset) SH REUSE, and afterwards get into ISPF 3.9. When the screen asks you which table you want to edit, you say XSP. The system will supply the suffix of ..TABL for you, and you'll be able to see the system table entries, one by one.

Now, we want to be able to specify special commands to invoke standard IBM ISPF screens, as we said before. With ISPF 3.9, editing XSP on one split, invoke the main menu on the other split, and enter the PANELID command on the command line of the second split. You'll instantly see the member name of the displayed ISPF panel in the upper left-hand corner of the second split screen. Go browse the panel library where this member is contained and look at the member. Toward the bottom of the member, you'll see all of the SELECT statements that invoke the different ISPF functions. Then you copy them as additional entries to the XSPTABL command table, on the ISPF 3.9 screen that's in the first split of your session. For example, to add a command called SP0, which invokes the ISPF 0 screen, look at the select statement of the ISR@PRIM panel (or whatever the name of the main panel is at your site) that specifies the command "0". Copy the name SP0 into a new line of your ISPF 3.9 split, specify 0 characters so the command can't be abbreviated, and then copy the SELECT parameters exactly from the ISR@PRIM panel into the ISPF 3.9 screen, except for the enclosing quotes. Do the same for all the other ISR@PRIM screen options. For 3.1, 3.2, etc., look at the ISRUTIL panel (or whatever name that PANELID specifies for ISPF 3) and get the select statements from

there to make commands SP31, SP32, and so forth, all the way up to SP314.

When all of the additional entries are complete, get out of ISPF 3.9 with PF3 (which automatically updates and saves the member XSPTABL). Copy XSPTABL to your own partitioned dataset as ISPTABL. Then, allocate this personal table library dataset to the top of your ISPTLIB DDNAME concatenation. Get out of ISPF and back into ISPF. Lo and behold, all of your new table entries will be usable.

If you don't want to go to all of this trouble every time you change shops, Gilbert Saint-flour might have just the solution for you on File 183 of the CBT MVS Tape, which contains a lot of his programs. The program is called FASTPATH. Gilbert made the FASTPATH program, which need not be authorized, to do all of the setups necessary to customize most of his own ISPF session to his own taste. Your requirements may be different, so you can go into the source code for FASTPATH, customize it to your own needs, and behold! You'll have your own customized ISPF session initializer.

Documentation about the CBT MVS Tapes can be found on the web at http://members.aol.com/cbttape. The free PDS package can be obtained online at http://members.aol.com/freepds. The NaSPA-sponsored CBT web site is at http://www.cbttape.org.

In conclusion, I'd like to mention one of Gilbert's other programs from File 183 of the CBT Tape, SHOWMVS; it's a systems programmer's (and EDP auditor's) dream. With just one invocation of the SHOWMVS load module as a TSO command, you'll get so much information about your system and your own TSO session that you probably won't need much more. There's a load module for SHOWMVS on File 135 of the CBT Tape that you can pop into one of your ISPLLIB libraries and try out. If you haven't seen it yet (or even if you have), you'll be amazed at how it's been improved. SHOWMVS will be able to show you so much about your system and your TSO session, as it currently is, that when you walk into a new shop, it'll be just that much easier to further customize your session and set yourself up. Good luck. See you next month. *ts*

NaSPA member Sam Golob is a senior systems programmer. He also participates in library tours and book signings with his wife, author Courtney Taylor. Sam can be contacted at sbgolob@aol.com or sbgolob@ibm.net.

