Article: WHOHAS the File I Need?

(Author Note: This is a pre-edit version of the whohas article that appeared in the January Netware Technical Journal. It is, however, the only documentation written for this program, so here it is.)

In a recent article in LAN Times, a survey was done of software packages that are written specifically to Netware. In this survey of features, one capability was mentioned that very few of the programs had. It was the ability for a user to find out quickly and easily who else on the LAN might be using a particular file. With an increase in the usage of networks for file sharing comes an associated increase in the probability of a collision, or concurrent attempted file access. For this issue's slice o' code I have written a small utility to provide users quickly with information about which other users are accessing a file.

This utility, named "WhoHas", tells you exactly what its name implies. It gives you the user name associated with every connection at the file server that currently has the file open. For instance if you try to edit a file only to have your word processor report back that the file is "locked" or "in use", then you could use this utility to find out exactly who was using the file. As an added extra, I included the capability to send a short message to everyone using the file. Something like "Get out or else!" can be sent to let other users know you need to access the file as soon as possible.

The program consists of four main sections: parsing the filename to prepare for the informational call, the actual loop using a virtual console call to get the connection numbers, displaying the user names for the connections, and finally sending a message if requested to. The only call which may involve a little explanation is the GetConnectionsUsingFile function. (This is a new function in the recently released Network C libraries.) It returns a structure (specified by CONN_USING_FILE) with both summary information and connection-specific information. Multiple calls will return additional specific information, but the summary information will remain the same (unless, of course, the file usage changes). The two variables lastRecord and taskID are used internally by the function to determine whether subsequent requests need to be made to the server, and which buffer number inside a particular reply to return to the caller. Setting them to zero initially is all that need be done by the user program.

This is just one example of a short utility to provide added information about the network environment. I'm sure that there are many other value-added utilities waiting to be written. With the inclusion in the C libraries of support for more of the virtual console and extended function calls, hopefully the network environment can be more easily enriched with programs and utilites that tap the information and services offered by Novell LANs.