NonStop NET/MASTER Tips and Techniques

by John New

Gresham Software Labs

Email: jnew@greshamsoftwarelabs.com.au

This article originally appeared in *The Connection*, Volume 22, No. 6 - November/December 2001, and is reproduced with permission from the International Tandem Users' Group (ITUG).

Introduction

This is an ongoing column with NonStop NET/MASTER tips and techniques. Each column is also accessible from http://www.greshamsoftwarelabs.com.au/ (Gresham Software Labs). Please send all comments and suggestions to John New at jnew@greshamsoftwarelabs.com.au.

Biography

John New is a technical writer. He has written and updated various Tandem manuals. He currently writes hard-copy, online, and web documents for a variety of software products.

File Handling in NCL: Part 4 – Communicating with Guardian 90 Processes

This article continues the series on file handling in NCL. Part 1 discussed opening and closing files. Part 2 discussed adding, reading, updating, and deleting records. Part 3 discussed creating and using user-defined maps.

This article discusses communicating with Guardian 90 processes from NCL. It covers opening and closing a process, and sending and receiving messages. It shows a simple working example of NCL process/Guardian 90 process communication.

As you will see, there are many similarities in the way NonStop NET/MASTER and NCL handle Guardian 90 processes and entry-sequenced, key-sequenced, and edit files.

Opening and Closing Processes For Use by NonStop NET/MASTER

UDBCTL commands open and close processes for use by NonStop NET/MASTER.

A UDBCTL OPEN command specifies the name of the process and can also assign a required identifier. For example:

```
UDBCTL OPEN=$TCPG ID=*
```

The UDBCTL CLOSE command closes the process for use by NonStop NET/MASTER. For example:

```
UDBCTL CLOSE=$TCPG
```

Opening and Closing Processes For Access by NCL

After a process is opened for use by NonStop NET/MASTER and assigned an identifier, FILE statements in an NCL procedure are used to access the process.

The FILE OPEN statement opens a process and can specify the structure of messages sent to and from the process. Messages must have the same structure in each direction, that is, messages must be mapped in both directions or unmapped in both directions. They cannot be mapped in one direction and unmapped in the other direction. For example:

```
FILE OPEN ID=$TCPG FORMAT=UNMAPPED
```

The FILE CLOSE statement closes the process for use by NCL. For example:

```
FILE CLOSE ID=$TCPG
```

Sending and Receiving Messages

NCL regards a process as an entry-sequenced file in the sense that you can send and receive data as a message serially to and from the process.

The FILE ADD and FILE PUT statements do the same thing. They send a message to a process but do not receive a message back (except a return code that indicates the result of the operation). If the process sends a message back, the message is discarded.

The FILE PUTGET statement works in the same way as the Guardian 90 WRITEREAD procedure call. This statement sends a message to a process and receives both a message and a return code from the process.

Typically, the NCL process reads the message from the Guardian 90 process into one or more variables for subsequent analysis and further processing.

The &SYS.FILE.RC system variable contains a return code after the FILE statement completes. If an error occurs, the &SYS.FILE.ERROR system variable has more information.

Working Example

The following NCL procedure satisfies the essential requirements for communicating with a Guardian 90 process. The procedure:

- Makes a Guardian 90 process (named $TCPG) available from NonStop NET/MASTER and NCL
- Specifies the structure of messages sent to the process (unmapped)

- Sends a message to the process (the value of the system variable &SYS.ALLPARMS, which is obtained when you run the procedure)
- Reads the message returned from the process and places the message in a variable (&RESP)
- Displays the value of &RESP on the OCS window where the NCL procedure was executed
- Makes the process unavailable from NonStop NET/MASTER and NCL

Note that the NCL procedure is a minimal example. It specifies unmapped message structure, whereas you would typically specify mapped. It does not provide run-time error checking, for example to check the success or failure of making the process available, sending and receiving a message, or making the process unavailable.

```
PUTGET: PROCEDURE
   INTCMD 'UDBCTL OPEN=$TCPG ID=*'
   INTREAD
   FILE OPEN ID=$TCPG FORMAT=UNMAPPED
   FILE PUTGET DATA=&SYS.ALLPARMS TO VARS=&RESP
   WRITE DATA='RETURNED <' || &RESP || '>'
   FILE CLOSE ID=$TCPG
   INTCMD 'UDBCTL CLOSE=$TCPG'
   INTREAD
END PUTGET
```

The following C program is a minimal example that reads the contents of the input buffer, reverses the characters from the input buffer, and writes the result to the output buffer.

```
#include <stdio.h> nolist
#include <cextdecs( CLOSE, OPEN, READUPDATEX, \
    REPLYX ) > nolist
#include <tal.h> nolist
int main()
{
    short rcv_num;
    char input_buffer[4096];
    short input_count;
    char output_buffer[4096];
    int i;
    OPEN((short *)"$RECEIVE               ", &rcv_num, 0, 1);
    while (1)
        {
        READUPDATEX(rcv_num, input_buffer,
            sizeof(input_buffer), &input_count);
        if (input_count == 1 && input_buffer[0] == 'X')
            {
            REPLYX(,,,, 1);
            break;
            }
        for (i = 0; i < input_count; ++i)
            {
            output_buffer[input_count - i - 1] = input_buffer[i];
            }
        REPLYX(output_buffer, input_count,,, 0);
        }
    CLOSE(rcv_num);
    return 0;
}
```

Running the Example

To run the example, follow these steps:

1 Create an NCL procedure called PUTGET in your user procedure library from the NCL code above.

2 Create a C source program called TCPGC (or a name of your choice) in an appropriate subvolume from the C code above.

3 Compile the C program to create an executable object file called TCPGO (or a name of your choice).

4 Run the object specifying a process name of $TCPG. For example:

```
RUN TCPGO / NAME $TCPG, NOWAIT/
```

5 From OCS, run the NCL procedure, specifying some parameters on the OCS command line. The parameters are passed to the procedure and stored in the system variable &SYS.ALLPARMS. For example:

```
START PUTGET A B C
```

6 Observe the results on the OCS window. If successful, the parameters will be reversed:

```
Returned <C B A>
```

7 To terminate the process, run the NCL procedure, specifying X as the parameter:

```
START PUTGET X
```

8 Observe the results on the OCS window. If successful, no parameters will be displayed:

```
Returned <>
```

9 Confirm that the process has terminated by typing the following command from TACL:

```
PPD $TCPG
```

Conclusion

This article has described the basics of communicating with Guardian 90 processes. More information is available in the NonStop NET/MASTER documentation set.

The final article in this series on file handling in NCL will discuss using alternate keys in key-sequenced files.

Note The author would like to acknowledge the assistance given by Rod Falck in the preparation of this article.