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 23, No. 1 - January/February 2002, and is reproduced with permission from the International Tandem Users' Group (ITUG).

Introduction

This is the final NonStop NET/MASTER tips and techniques column. This column and all past columns are accessible from http://www.greshamsoftwarelabs.com.au/ (Gresham Software Labs).

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 5 – Accessing Key-Sequenced Files With Alternate Keys

This article concludes 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. Part 4 discussed communicating with Guardian 90 processes from NCL.

This article discusses accessing key-sequenced files with alternate keys. Through a series of example NCL procedures (that you can type and try out yourself) the article explains how to:

- Create, open, and close files
- Add records
- Read records by primary and alternate key
- Delete records by primary and alternate key

A sample session shows how to run the NCL procedures from Operator Control Services (OCS).

Alternate Key Files, NonStop NET/MASTER, and NCL

NonStop NET/MASTER and NCL work with alternate key files in the same way as other Himalaya software.

Apart from when you create and name an alternate key file, for example using FUP from NonStop NET/MASTER and NCL, you do not refer to an alternate key file by name, only the primary key file. When you perform file operations from NonStop NET/MASTER and NCL (for example, open, close, add, read, delete, and update), access to the alternate key file is handled automatically.

Creating the Primary and Alternate Key Files

There are various ways to create a key-sequenced file with an alternate key. From an NCL procedure, it is often convenient to use the FUP CREATE command.

The following NCL procedure uses FUP to create:

- A primary key-sequenced file called PKFILE with a record length of 86 bytes and primary key length of 6 bytes from offset 0
- An alternate key-sequenced file called AKFILE with an alternate key specification of "AK" and alternate key length of 4 bytes from offset 6

The NCL procedure checks whether the files are created. If so, it displays the complete file specifications. (This is because, if you are trying out the examples, the files may not be created in the same directory as the NCL procedure and you need to know the complete file specifications to open the files using the UDBCTL OPEN command from OCS.) If the files already exist, an error message is displayed.

```
createpa: PROCEDURE
  /* Creates files */
  &filename = PKFILE
  /* Use FUP CREATE to create new files */
  SAY "Creating file "&filename
  INTCMD 'OPSYS SEND FUP CREATE '&filename', ',
    'TYPE K, ',
    'REC 86, ',
    'BLOCK 4096, ',
    'KEYLEN 6, ',
    'KEYOFF 0, ',
    'ALTKEY ("AK",FILE 0,KEYOFF 6,KEYLEN 4), ',
    'ALTFILE (0,AKFILE)'
  &need_filename = 1
  /* Check result of FUP CREATE command */
  DO UNTIL &msgno = NNM0999
    INTREAD VARS=(&msgno(7),*,&text) TYPE=ANY PARSE=NO
    IF POS("ERR 10",&text) > 0 THEN DO
      /* File exists, so flush NCL process */
      &char = ":"
      &filename = SELSTR(&char,WORD(&text,3))
      SAY &filename" already exists, stopping"
      FLUSH
```

```
      END /*do*/
    IF POS("CREATED",&text) > 0 THEN DO
      /* Files created, display file names */
      IF &need_filename = 1 THEN DO
        &need_filename = 2
        &filename = WORD(&text,3)
        SAY "The complete filename is "&filename
        END
      ELSE IF &need_filename = 2 THEN DO
        &need_filename = 0
        &ak_filename = WORD(&text,3)
        SAY "The alternate key filename is "&ak_filename
        END
      END
  END /*do until*/
  /* Secure files for unrestricted access */
  INTCMD 'OPSYS SEND FUP SECURE '&filename', "NNNN"'
  GOSUB read_replies
  INTCMD 'OPSYS SEND FUP SECURE '&ak_filename', "NNNN"'
  GOSUB read_replies
  INTCMD "OPSYS KILL FUP"
  GOSUB read_replies
  EXIT
  /* Check that command completes correctly */
  read_replies:
    &msgno = ""
    DO UNTIL &msgno = "NNM0999" OR &msgno = "NNM0995"
      INTREAD VARS=&msgno
    END
  RETSUB
END createpa
```

Opening the Files

Opening the primary and alternate key files requires two steps: opening the files for use by NonStop NET/MASTER; and opening the files for use by NCL.

The following UDBCTL OPEN command from OCS opens both PKFILE and AKFILE for use by NonStop NET/MASTER. It assigns PKFILE an identifier of PKFILE:

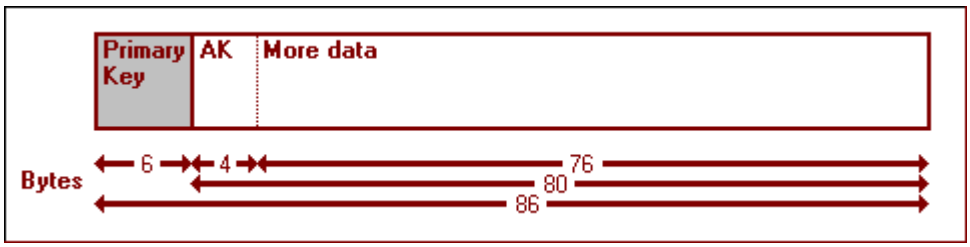
```
UDBCTL OPEN=$MYVOL.MYSUBVOL.PKFILE ID=PKFILE
```

The following FILE OPEN statement opens the files for use by NCL. Note that the record structure is unmapped (however you can also specify mapped or delimited alternate key files).

```
FILE OPEN ID=&id FORMAT=UNMAPPED
```

Figure 1 shows the structure of each record in the unmapped primary key-sequenced file.

Figure 1. Alternate key file with unmapped record structure



Adding Records

Both the FILE ADD and FILE PUT statements add records to a key-sequenced file. The FILE PUT statement replaces a record if the primary key you specify matches an existing primary key.

The following NCL procedure adds records to PKFILE (and automatically add records to AKFILE). Each primary key happens to be less than the maximum primary key length of six bytes, for example, John, Rod, and Bo. The alternate key for each record is the first four bytes of the data (three digits plus one space), for example, "555 ", "220 ", and "111 ".

The &SYS.FILE.RC system variable contains a return code after a FILE statement finishes an operation. The NCL procedure checks this variable to determine whether the FILE OPEN statement completes successfully. Other system variables that FILE statements can check are

&SYS.FILE.ERROR, &SYS.FILE.ID, &SYS.FILE.KEY, &SYS.FILE.RCNT, &SYSMSG, and &SYS.VARCNT.

```
loadpa: procedure
  /* Loads records */
  &id=PKFILE
  file open id=&id format=unmapped
  select &sys.file.rc
    when 0  then say "0 read-only access"
    when 4  then say "4 read and write access"
    when 8  then say "8 read, write, delete access"
    when 12 then say "12 no access"
    when 16 then say "16 "&sysmsg
    otherwise
      say unexpected error
  end /*select*/
  say "Loading "&id
  file put key="Adam" data="223 Adam's record"
  file put key="Tony" data="111 Tony's record"
  file put key="Andy" data="111 Andy's record"
  file put key="Rod"  data="220 Rod's record"
  file put key="Jim"  data="444 Jim's record"
  file put key="Alan" data="222 Alan's record"
  file put key="John" data="555 John's record"
  file put key="Bo"   data="111 Bo's record"
  file put key="LynB" data="221 LynB's record"
  file put key="LynF" data="212 LynF's record"
  file put key="Ken"  data="333 Ken's record"
  file close id=&id
  select &sys.file.rc
    when 16 then say "16 closed"
    otherwise
      say unexpected error
  end /*select*/
end loadpa
```

Reading Records

The FILE GET statement reads a record from a file and places the data in one or more variables. When you sequentially read all records from a key-sequenced file (from the first or last record) you do not need to specify a record key. Otherwise, you must specify a key that exactly matches an existing key to retrieve a specific record (exact retrieval) or a key that partially matches existing keys to retrieve a group of records (generic retrieval).

To retrieve records using the primary key, use FILE GET with the KEY keyword. To retrieve records using the alternate key, use FILE GET with the ALTKEY keyword in addition to the KEY keyword.

When you read records, you can use the KEYEXTR keyword. This specifies whether the NCL regards the record key as part of the data in a record (KEYEXTR=NO) or not (KEYEXTR=YES). The setting of KEYEXTR determines what parts of a record are placed into variables. Figure 2 shows the structure of a typical unmapped record. Table 1 shows the effect of KEYEXTR when NCL reads the record into a variable (a caret (^) indicates a space).

Figure 2. A typical unmapped record

Table 1. The effect of the KEYEXTR keyword

FILE SET	KEYEXTR=NO	KEYEXTR=YES
KEY=ABC	Variable contains: ABC^^^123^THE QUICK BROWN FOX ...	Variable contains: 123^THE QUICK BROWN FOX ...
ALTKEY=123	Variable contains: ABC^^^123^THE QUICK BROWN FOX ...	Variable contains: ABC^^^THE QUICK BROWN FOX ...

The following NCL procedure retrieves records from PKFILE using the primary key. When you run the NCL procedure you can specify a key. If you do, all records with a partially matching key are retrieved and displayed; otherwise, all records are retrieved and displayed.

```
pkread: procedure nofold
  /* Reads records by primary key */
```

```
  &id=PKFILE
  file open id=&id format=unmapped
  if &1 = "" then
    /* Sets key to read all records */
    do
      &opt = seq
      file set keyextr=no
    end
  else
    /* Sets key to read matching records */
    do
      &opt = keq
      file set key=&1 keyextr=no
    end
  file set keyextr=no
  file get opt=&opt vars=&rec
  do while &sys.file.rc = 0
    say &rec
    file get opt=&opt vars=&rec
  end
  file close id=&id
end pkread
```

The following NCL procedure retrieves records from PKFILE using its alternate key. The ALTKEY keyword specifies the two-character identifier (AK) that specifies the record's alternate key. When you run the NCL procedure you can specify a key. If you do, all records with a partially matching key are retrieved and displayed; otherwise, all records are retrieved and displayed.

```
akread: procedure nofold
  /* Reads records by alternate key */
  &id=PKFILE
  file open id=&id format=unmapped
  if &1 = "" then
    /* Sets key to read all records */
    do
      &opt = kge
      file set altkey="AK" key=" " keyextr=no
    end
  else
    /* Sets key to read matching records */
    do
      &opt = keq
      file set altkey="AK" key=&1 keyextr=no
    end
  file get opt=&opt vars=&rec
  do while &sys.file.rc = 0
    say &rec
      file get opt=&opt vars=&rec
    end
  file close id=&id
end akread
```

Deleting Records

The FILE DEL statement deletes records. To delete records using the primary key, use FILE DEL with the KEY keyword. To delete records using the alternate key, use FILE DEL with the ALTKEY keyword in addition to the KEY keyword.

The following NCL procedure deletes all records with a primary key that matches the key you specify:

```
pkdel: procedure nofold
  /* Deletes records using primary key */
  &id=PKFILE
  file open id=&id format=unmapped
  file set key=&1
  file del key=&1 opt=keqall
  file close id=&id
end pkdel
```

The following NCL procedure deletes all records with an alternate key that matches the key you specify:

```
akdel: procedure nofold
  /* Deletes records using alternate key */
  &id=PKFILE
```

```
  file open id=&id format=unmapped
  file del altkey="AK" key=&1 opt=keqall
  file close id=&id
end akdel
```

Closing the Files

Closing the primary and alternate key files requires two steps: closing the files for use by NCL and closing the files for use by NonStop NET/MASTER.

The FILE CLOSE statement closes a file for use by NCL:

```
FILE CLOSE ID=&id
```

The UDBCTL CLOSE command closes a file for use by NonStop NET/MASTER:

```
UDBCTL CLOSE=$MYVOL.MYSUBVOL.PKFILE
```

Sample Session

The following sample session uses the NCL procedures described above. Lines in **bold** are commands that you type at the OCS command input line. Other lines are the responses to commands, which are displayed in the OCS window.

```
   START CREATEPA
   Creating file PKFILE
   The complete filename is $MYVOL.MYSUBVOL.PKFILE
   The alternate key filename is $MYVOL.MYSUBVOL.AKFILE
   NNM1005 START CREATEPA PROCESSING COMPLETE. NCLID 000374
   UDBCTL OPEN=$MYVOL.MYSUBVOL.PKFILE ID=PKFILE
   NNM0300 OPEN REQUEST COMPLETE
   START LOADPA
   8 read, write, delete access
   Loading PKFILE
   16 closed
   NNM1005 START LOADPA PROCESSING COMPLETE. NCLID 000379
   START PKREAD
   Adam  223 Adam's record
   Alan  222 Alan's record
   Andy  111 Andy's record
   Bo    111 Bo's record
   Jim   444 Jim's record
   John  555 John's record
   Ken   333 Ken's record
   LynB  221 LynB's record
   LynF  212 LynF's record
   Rod   220 Rod's record
   Tony  111 Tony's record
   NNM1005 START PKREAD PROCESSING COMPLETE. NCLID 000380
   START AKREAD
   Andy  111 Andy's record
   Bo    111 Bo's record
   Tony  111 Tony's record
   LynF  212 LynF's record
   Rod   220 Rod's record
   LynB  221 LynB's record
   Alan  222 Alan's record
   Adam  223 Adam's record
   Ken   333 Ken's record
   Jim   444 Jim's record
   John  555 John's record
   NNM1005 START AKREAD PROCESSING COMPLETE. NCLID 000381
   START PKREAD A
   Adam  223 Adam's record
   Alan  222 Alan's record
   Andy  111 Andy's record
   NNM1005 START PKREAD PROCESSING COMPLETE. NCLID 000385
   START AKREAD 22
   Rod   220 Rod's record
   LynB  221 LynB's record
   Alan  222 Alan's record
   Adam  223 Adam's record
   NNM1005 START AKREAD PROCESSING COMPLETE. NCLID 000386
   START PKDEL L
   NNM1005 START PKDEL PROCESSING COMPLETE. NCLID 000387
   START PKREAD
   Adam  223 Adam's record
   Alan  222 Alan's record
```

```
Andy  111 Andy's record
Bo    111 Bo's record
Jim   444 Jim's record
John  555 John's record
Ken   333 Ken's record
Rod   220 Rod's record
Tony  111 Tony's record
NNM1005 START PKREAD PROCESSING COMPLETE. NCLID 000388
START AKDEL 1
NNM1005 START AKDEL PROCESSING COMPLETE. NCLID 000389
START AKREAD
Rod   220 Rod's record
Alan  222 Alan's record
Adam  223 Adam's record
Ken   333 Ken's record
Jim   444 Jim's record
John  555 John's record
NNM1005 START AKREAD PROCESSING COMPLETE. NCLID 000391
UDBCTL CLOSE=$MYVOL.MYSUBVOL.PKFILE
NNM0300 CLOSE REQUEST COMPLETE
```

Conclusion

This article on file handling in NCL has discussed using alternate keys in key-sequenced files and concludes the series on file handling.

You can find more information on file handling in the *NonStop NET/MASTER NCL Programmer's Guide* and *NonStop NET/MASTER NCL Reference Manual*.

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