

Tutorial 3 – /Free IO

You know what I want to know? When the guys at IBM were developing /Free did they make a lot of ‘free’ jokes. You know, yelling back and forth to each other ‘Hey, I’m working on the first version of RPG that is “Free”. And then the other ones would snicker because you never really get anything from IBM for free. I wonder. I would have. Kind of geeky, I know but . . .

Anyway, you won’t get far in programming if you can’t do file IO and /Free has some interesting (as in nice) twists on the traditional RPG IO procedures. I think it’s more straight forward and self documenting and is one of the big reasons why I now use /Free exclusively.

Basic Syntax

Some things in /Free have not changed from the fixed format, like the op code names for the IO; READ, CHAIN, SETLL, UPDATE, WRITE, DELETE.

And you still have that weird thing where READ, CHAIN, and SETLL use the file ID in the statement while UPDATE, WRITE, and DELETE use the record format. Although you can use the record format for the Read, Chain, and SETLL also but I never do that.

Of course the basic syntax is different, with the order being more English language and the semi colon at the end, plus you have some additional options with keys that I really like.

Reading with Keys

Speaking of keys, remember how in the old days you had to code up C-specs with the KLIST and KFLD junk? Well, that’s not supported in /Free. You do not need to set up key definitions in either D or C specs. Instead, you simply list the keys as arguments separated by colons in the CHAIN statement.

```
CHAIN (keyfld1:keyfld2:keyfld3) FileID;
```

I like this. I like seeing the list of keys that we are using right in the Chain, very self documenting. And the keyfld fields do not have to be the ones from the file you are reading. They should be whatever field currently holds the value you want to read by. It just has to be the same size and type as the actual key. It could even be a BIF statement, like %DEC(fielda). Don’t forget that’s just a variable like any other variable except that it will be transformed before it is used. Sort of like a caterpillar becoming a butterfly before it is eaten by a robin.

That is, the following two statements are identical, it just depends what looks more meaningful to you.

```

EVAL keyfld1 = file_field1;
CHAIN (keyfld1) FileID;

CHAIN (file_field1) FileID;

```

In some ways I like seeing a list in the Chain of exactly what the key fields are for this access, and other times I don't feel like coding the EVAL.

There is another way to do keys, but I don't like it and I'm not going to cover it.

Oh for goodness sake, stop looking like that. This is my tutorial and if I don't want to expose you to something I don't have to. I mean it's not like you paid money for . . . paid money for the tutorial. . . Hmmmm, I hadn't thought of that before. What an idea . . . hey, everybody who's reading this owes me \$299, no wait, \$399, no \$469. And I'd prefer cash from some of you . . . and no hundreds!

Of course, now that you're paying . . . OK, here's the deal. You set up a D-spec with a name like 'key1' or something. It's a DS with no picture but use the keyword 'Like-Rec' and then in parenthesis you put the file name and the verbiage '*key', separated by colons. Actually, just one colon. Then on the CHAIN you use the %KDS built in function. Like this.

```

D Key1           DS           Like-Rec(Filename:*keys)

```

Then in the code you go

```

Key1.CUSNO = cusno value or variable
Key1.WHSE = whse value or variable
CHAIN %KDS(Key1) File1

```

As you can see this is very similar to the fixed format way of doing it except that you don't have to put a picture to the key layout and it's in D-specs vice C-specs. That seems to be one thrust of /Free; put data in D and logic with the code.

Error Checking

The other thing I really like about /Free is the error checking and stuff. /Free does not support the old HI, EQ, LO scheme so you can't use indicators to check the results of your IO. Yes, let's all stand very solemnly and wave goodbye to the indicators as they drive out of sight. Then the party can begin.

Or maybe you don't feel that way. In which case, I am deeply and sincerely sorry. Hey, is that beer cold?

To do IO Checking, you need to rely on the built in functions. Namely %FOUND for CHAIN and %EOF for SETLL. To set a 'not' condition, simply code it as 'NOT %FOUND', etc.

```
DOW Not(%EOF);
```

```
IF %FOUND;
```

```
Etc.
```

I am assuming that you already know how to use the %EOF and the %FOUND, but there are some other error BIF's that might come in handy. Remember, these BIF's are not part of /Free per se. They are part of RPG and can be used in Fixed Format as well.

%ERROR

A handy BIF is the %ERROR. To get this to activate you need to use the 'e' extender on your CHAIN or READ (CHAIN(e), for example). It functions like the LO indicator in fixed format (returning a '0' or '1') and is the kind of thing you want to use if you have an error handling routine that you can call.

```
CHAIN(e) (key1) File1;  
IF %ERROR;  
    EXSR ERROR_HDLR;  
ENDIF;
```

%EQUAL

This is used with the SETLL op code to see if we found a record whose key equals the key we are pointing at. It can be used with multiple key field keys, and can be used with a partial key. If the value returned is '0', then it means that no records were found for whatever key (full or partial) you used. If the value is '1' then you got a hit. It can help eliminate the need to do an actual read and check the file field against some other field to see if anything's out there. You know, if you were just doing a CHAIN to verify the record exists, you could instead do a SETLL and check the value of %EQUAL. It's less overhead than a CHAIN which actually retrieves the data field by field. An 'IF %EQUAL' branch would be triggered by a hit on the key value.

Workstation IO

You will use the same op codes for workstation oriented IO that you use now (READ, WRITE, UPDATE, EXFMT, DELETE). The only difference is they will now use the /Free syntax of op code first, then the record you are processing. And, of course, the semicolon.

Printer IO

Again, this is basically the same as in fixed format RPG (WRITE and EXCEPT) except for the syntax change and the semicolon.

The End

IO in /Free is pretty easy, and I think a lot more intuitive than in fixed format. What you should do at this point is go back through the tutorial and, if you have not already done so, put some IO statements into your sample program. Kind of reinforce in your mind what the syntax is, and how to handle both keys as well as error checking. Don't think that just by reading this you know how to do it. You have to experiment a bit to really settle it in.

[Questions or Comments?](#)

For more information on how /Free can help you, contact Shirey Consulting Services at 616-452-6398 or support@shireyllc.com.