Technical Report

Santa Teresa Laboratory
San Jose, CA

IBM

TR

Migrating Applications to APL2

by Michael T. Wheatley

October 1984

TR 03.266

October 1984
TR 03.266

MIGRATING APPLICATIONS TO APL2

by

M. T. Wheatley

International Business Machines Corporation
General Products Division
Santa Teresa Laboratory
San Jose, California

Migrating Applications to APL2

Michael T. Wheatley, APL Language Development, IBM Santa Teresa Laboratory, 555 Bailey Ave., P.O. Box 50020, San Jose, Ca., USA 95150.

Abstract

Recent announcement of APL2 as IBM's strategic APL offering has highlighted the need to migrate existing information center applications to an APL2 base. Issues involved in such a migration are discussed and the changes required to migrate two sample applications are shown.

Introduction

APL2 is an advanced implementation of APL which provides significant language and system extensions over previous IBM APL implementations. It is designed to be generally upward compatible from VSAPL and it has been our experience that most VSAPL applications can be migrated to APL2 with little or no change.

Some concern, nonetheless, has been expressed by a number of VSAPL users with regard to potential problems involved in migrating large and complex applications, such as those characterized by the information center products. In an attempt to quantify problems in this area, migration of two key information center products - ADRS-II and APLDI-II (VM/CMS versions) was undertaken and has been documented in this paper. These two products were chosen because of their complexity and because of their wide acceptance in the marketplace. It should be pointed out, however, that Info Center/1, which will operate under APL2, has been announced as a replacement for these two products.

Migration of the two products described here was undertaken by a competent APL programmer without any extensive experience with either of the products and without any detailed or special knowledge of their internal operation. The source material used included only the products themselves and their supporting documentation. Testing of the migrated code involved verifying the examples in the ADRS-II and APLDI-II Student Texts [1] and in the APLDI-II Program Description/Operations manual [2]. Subsequent casual use of the migrated applications by a number of users has uncovered no additional problems.

Less than one half day's effort was required to migrate each of the products to APL2 and to accomplish the described testing and verification! Furthermore, the changes were made in a fashion which allows the resulting code to run under either VSAPL or APL2.

The information contained in this document has not been submitted to any formal IBM test and is presented on an "as is" basis without any warranty either expressed or implied. The use of this information is a customer responsibility and depends on the customer's ability to evaluate and integrate it into the customer's operational environment.

Migration Aids

The primary migration aid provided with APL2 is the new)*MCOPY* command which allows an APL2 user to copy a VSAPL workspace. In addition to supporting all the things a normal)*COPY* command does, the)*MCOPY* command also:

- translates character data to APL2/EBCDIC encoding;
- inserts parentheses around indexed constant numeric vectors;
- copies the following system variables:
  - ⎕*LX*, ⎕*IO*, ⎕*RL*, ⎕*CT* and ⎕*PP*
- assigns function time stamps as required;
- deletes local names that match either argument of a function;
- replaces groups with character matrices containing the group name and the name of objects in the group.

In addition to the)*MCOPY* command, a workspace, 2 *TRANSFER*, is provided which contains a number of tools to aid migration. Chief among these is a function called *FLAG_* which allows the user to search for arbitrary character strings in defined functions.

ADRS-II Migration

The ADRS-II product is shipped with a VSAPL workspace called *ADRSFULL* which contains all of the functions and data for the VM/CMS version of the product. This workspace was used as the base for the migration which is described in the following steps:

- APL2 was invoked (in a 4 megabyte VM/CMS virtual machine) and the VSAPL workspace *ADRSFULL* was copied with the command:

```
)MCOPY ADRSFULL
```

  Since it was suspected that ADRS-II might have dependencies on the VSAPL □*AV*, uses of that system variable were isolated by using the *FLAG_* function from the APL2 workspace 2 *TRANSFER*:

```
)PCOPY 2 TRANSFER (GPFLAG_)
□PW←350
(⊂'□AV') FLAG_ □NL 3
```

  It was determined that the following functions referenced □*AV*:

CONPGMS	*GLOBS*	*BROWSE*
ECI	*II*	*CLEAR*
FI	*VNAME*	*CONVERT*
GE	*ZCI*	

  The functions *FI* and *II* involve legitimate use of □*AV* for floating point and integer conversion. □*AV* as used in the other functions, however, is dependent upon the VSAPL ordering of characters.

- VSAPL Release 4.0 was then invoked and the *ADRSFULL* workspace was loaded. After determining that the name Δ*AV* was not used in the workspace, a new variable was created with the statement:

```
ΔAV←□AV
```

  Then all references to □*AV* in the functions listed above, except those in functions *FI* and *II*, were changed to Δ*AV*.

- Later in the process of testing under APL2 it was discovered that lines 13 and 14 of the function *KEY2* contain indexed constant numeric vectors without parentheses. The parentheses are not inserted by)*MCOPY* in this case because the numeric vectors are part of executed strings. The function was changed as follows: follows:

```
[13]   ⍎(LI[1]=2)/'LI[1]←(2 9)[LK[LI[2]-3 2 1]∧.=''AND'']'
[14]   ⍎(LI[1]=2)/'LI[1]←(2 10)[LK[LI[2]-2 1]∧.=''OR'']'
```

- It was also discovered in later testing that function *FLAT* dynamically creates APL statements in which a numeric left argument is positioned without an intervening blank between it an the function. While this is overlooked in VSAPL, it will cause an error in APL2. This problem was corrected by modifying the function *FLAT* at lines 4 and 6 as shown:

```
[4]   ⍎¯3↓,'M',LN,((1↑⍴LN),7)⍴'←0⍴0 Δ '
[6]   ⍎¯2↓,'M',LC,'←','0',' ',(⍕LW),((1↑⍴LW),6)⍴'⍴''''  Δ '
```

- Finally, the workspace was saved and the installation was completed by executing the function *ADRSSET*. Execution of this function creates the *ADRS2* workspace and the associated function file. Note that because AP 124 is not supported in the APL2 environment, AP 126 must be chosen in the installation process if full screen processing is to be supported.

- APL2 was then invoked again and the two workspaces were copied and saved with the commands:

```
)MCOPY ADRS2
)SAVE ADRS2
)CLEAR
)MCOPY ADRSFULL
)SAVE ADRSFULL
```

Because certain functions with a dependence on the VSAPL □*AV* were changed on the ADRS-II function file, the Δ*AV* variable must be created in any previously saved ADRS-II workspaces. This can be done under VSAPL ans shown:

```
)LOAD MYWS
ΔAV←□AV
)SAVE
```

so that those workspaces will continue to work correctly in the VSAPL environment.

These workspaces can be migrated to APL2 using the following procedure under APL2:

```
)CLEAR
)MCOPY MYWS
)COPY ADRS2 CONPGMS GLOBS CLEAR CONVERT
)SAVE MYWS
```

APLDI-II Migration.

APLDI-II consists of 5 VSAPL workspaces,

APLDI DIAUX DIAUX126 DIUPDATE DICREATE

and a function file APLDIFUN FILE. There is no single workspace which contains all of the functions from the function file and which could be used as a source for)*MCOPY*. The first step of the migration outlined below, therefore, involves creation of a workspace containing all of these functions.

- VSAPL Release 4.0 was invoked in a 4 megabyte VM/CMS machine and the following statements entered to retrieve all of the functions from the function file:

```
)LOAD DIAUX
LIST←LIST
COPY LIST
```

- Since it was suspected that certain APLDI-II functions might have a dependence upon the VSAPL ⎕*AV*, after determining that the name was not used elsewhere in any APLDI-II workspace, the variable ⍙*AV* was created with the statement:

```
⍙AV←⎕AV
```

- It was discovered in later testing that APLDI-II determines the internal data type for APL objects through the use of ⎕*WA* in the function *DT*. The following lines were added to that function to cause it to work correctly in both VSAPL and APL2:

```
[3.5]  →(3=⎕NC '⎕AF')/L2
[8]    ⍝
[9]    ⍝ APL2 VERSION OF FUNCTION
[10]   L2:X←128⍴X
[11]    R←(1+4 ⎕AT 'X')÷128
[12]    →((R≠1)∨0≠1+0⍴X)/0
[13]   ⍝ FORCE BYTE INTEGERS TO FULLWORDS
[14]    R←4
```

and the resulting workspace was saved under the name *DIAUXFUL*. Note the use of ⎕*NC* '⎕*AF*' to determine whether the user is running under VSAPL or APL2.

- The revised *DT* function and the new ⍙*AV* variable were copied to the workspaces *APLDI* and *DIAUX* where they are also required.

- By)*MCOPY*'ing *DIAUXFUL* and the other APLDI-II workspaces to APL2 and running the *FLAG_* function from 2 *TRANSFER* against them, it was determined that the functions *SORT* (in workspaces *DIAUX* and *DIAUXFUL*) and *QFUNCT* (only in workspace *DIAUXFUL* contain dependencies on the VSAPL ordering of characters in ⎕*AV*. All occurrences of ⎕*AV* in these functions were changed in the VSAPL workspaces to ⍙*AV*.

- In subsequent testing it was determined that the following functions create executed expressions containing indexed constant numeric vectors without parentheses:

```
FFN     in workspaces:  APLDI, DIAUXFUL, DIUPDATE
CODER   in workspace:   DIAUXFUL
DAYS    in workspace:   DIAUXFIL
```

Parentheses were inserted as shown:

```
FFN[24]     ⍎(0=⍴DFW)/'DFW←(W←(⍴PH)⌈⌈DEF[J;2]×
            (0 1 2.5 1 2 1 2 2)[DEF[J;0]])⍴DEF[J;5]'

CODER[24]   SEB←0⍴SEE←'R←(T1←1=+⌿R)/R←(T1,(⍴R)÷T1←',
            (⍕1↑⍴T),')⍴R∩R←T1\R←(1+⍳1↑⍴R)+.×R∩',
            (0≠''⍴⍴Y)/'R←(',(⍕Y),')[R]∩'

DAYS[10]    SEB←0⍴SEE←'R←','- '[PO],Y,'-R[2;]+(',
            (⍕CMO),')[R[1;]]+(365×R[0;])+(0=4|R[0;])∧
            2<(R←(3⍴100)⊤,R)[1;]∩'
```

- Certain APLDI-II functions depend upon the fact that in VSAPL, a pendant function cannot be expunged. In APL2, this is not the case and the following changes must be made in workspace *DIAUXFUL*:

```
∇SUB[2.5]     EXFUN←''∇
∇SUBDET[.5]   EXFUN←''∇
∇SUBDET3[.5]  EXFUN←''∇
∇DISUM[.5]    EXFUN←''∇
∇TAGN[.5]     EXFUN←''∇
∇UPDATE[1.5]  EXFUN←''∇
```

  These modifications do not affect operation under VSAPL, but correct the problem when operating under APL2.

- After the changes indicated above were made, the resulting VSAPL workspaces were saved.

- A new function file was created by first renaming the old one using the CMS command:

```
RENAME APLDIFUN FILE A = OLDFILE =
```

  and then entering the following statements under VSAPL:

```
)LOAD DIAUXFUL
INITIAL
200
APLDIFUN FILE
SAVE LIST
```

- Then, as per the APLDI-II installation instructions, the system was customized by entering the following statements under VSAPL:

```
)LOAD APLDI
)COPY DIAUX126 DISPGRP
)SAVE FADI
```

  Note that it is not possible to use AP 124 with APL2 since that AP is not supported in the APL2 environments. AP126 is used instead.

- Finally, the resulting VSAPL workspaces,

APLDI, *DIAUX*, *DIAUX*126, *DIUPDATE*, *DICREATE*, *DIAUXFUL* and *FADI*

were)*MCOPY*'ed to APL2 and saved there under the same names. This step completes the migration process.

Because changes were made to certain functions on the APLDI-II function file, the following change must be made to any APLDI workspaces previously saved under VSAPL:

```
)LOAD MYWS
)COPY APLDI ΔAV DT FFN
)SAVE
```

The resulting workspaces should run correctly under VSAPL and may be)*MCOPY*'ed to APL2 where they should also run correctly if no changes to user written functions are required.

Conclusions

Migration of these two large applications to APL2 proved to be surprisingly simple. After converting □*AV* dependencies, it was found that the simplest method for isolating problems involved)*MCOPY*'ing the application and testing it in the APL2 environment. As problems were encountered they were fixed and testing proceeded.

The *FLAG_* function in the 2 *TRANSFER* workspace also proved to be a valuable tool. When a problem was encountered during testing, *FLAG_* was used to search for other occurrences of similar problems.

Changes to the code were made in such a way that the resulting application would run under either VSAPL or APL2. This approach is useful for a number of reasons:

- It allows substantial additional testing of the resulting code against production applications before actually migrating those applications to APL2.
- User modified ADRS-II and APLDI-II workspaces, previously saved under VSAPL should continue to run correctly with the new function file. These workspaces could then be migrated in a orderly fashion to APL2.
- Should problems occur in early use of APL2, the applications could be run under VSAPL until those problems were corrected.
- No attempt was made to use the new facilities in APL2. Thus, no substantial education is required to accomplish the migration. Further, when and if it was decided to change the applications to capitalize upon new APL2 facilities, stable underlying code will reduce potential problems in the process.

It was felt that access to a competent APL programmer to assist in the debugging process contributed greatly to the rapid success achieved. This programmer need not be an APL wizard, but should be experienced with VSAPL and should be familiar with the incompatibilities between VSAPL and APL2 as outlined in the APL2 Migration Guide [3].

No great understanding of the application or its internal design was needed to accomplish the migration. Some familiarity with the function file design of APLDI-II (as described in the APLDI-II Systems Guide [4]) proved useful. Much more important, however, was access to a set of test cases (sample problems) which tested substantial portions of the code.

Finally, it should be pointed out that the two applications described in this paper are substantially larger and more complex than typical user written APL applications. Migration of less complex user written applications may be considerably simpler, with many requiring no changes at all.

References

1. A Departmental Reporting System II Student Text,
   IBM Corporation, 1980, SC20-1893-0.

   APL Data Interface II Student Text,
   IBM Corporation, 1981, SC20-1891-1

2. APL Data Interface-II Program Description/Operations
   Manual, IBM Corporation, 1981, SH20-6147

3. APL2 Migration Guide,
   IBM Corporation, 1984, SH20-9215

4. APL Data Interface-II Systems Guide
   IBM Corporation, 1980, LY20-9007-0

APL2 Migration Aids

)MCOPY

- translates character data
- inserts parentheses for indexed numeric vector constants
- copies ⎕LX, ⎕IO, ⎕RL, ⎕CT, ⎕PP
- assigns function time stamps
- deletes local names which duplicate argument names
- replaces groups with lists

2 TRANSFER FLAG_

- used to search for arbitrary character strings:

```
'⎕AV' '⎕WA' FLAG_ ⎕NL 3
```

ADRS–II Migration

1. Under APL2:

```
)MCOPY ADRSFULL
)PCOPY 2 TRANSFER (GPFLAG_)
⎕PW←350
(⊂'⎕AV') FLAG_ ⎕NL 3
```

2. In ADRSFULL under VSAPL, change all occurances of ⎕AV to ⍙AV in:

CONPGMS	GLOBS	BROWSE
ECI	VNAME	CLEAR
GE	ZCI	CONVERT

3. In ADRSFULL under VSAPL, modify functions KEY2 and ELAT as shown:

```
KEY2[13] ±(LI[1]=2)/'LI[1]←(2 9)[LK[LI[2]-3 2 1]
         ∧.=''AND'']'
KEY2[14] ±(LI[1]=2)/'LI[1]←(2 10)[LK[LI[2]-2 1]
         ∧.=''OR'']'

ELAT[4]  ±¯3↓,'M',LN,((1↑⍴LN),7)⍴'←0⍴0 ⍙ '
ELAT[6]  ±¯2↓,'M',LC,'←','0',' ',(⍕LW),((1↑⍴LW),6)
         ⍴'⍴''''' ⍙ '
```

ADRS–II Migration

4. Under VSAPL:

```
ΔAV←⎕AV
)SAVE ADRSFULL
ADRSSET
```

5. Under APL2:

```
)MCOPY ADRSFULL
)SAVE ADRSFULL
)CLEAR
)MCOPY ADRS2
)SAVE ADRS2
```

6. Under VSAPL:

```
)LOAD MYWS
)COPY ADRS2 ΔAV ⍝CONPGMS ⍝GLOBS
)COPY ADRS2 CLEAR CONVERT
)SAVE
```

APLDI–II Migration

Under VSAPL:

1.)LOAD DIAUX
   LISI←LIST
   COPY LISI
   ΔAV←⎕AV

2. **Change all occurances of ⎕AV to ΔAV in SORT, QFUNCT**

3. **Modify function DT:**

```
[3.5]  →(3=⎕NC '⎕AF')/L2
[8]    ⍝
[9]    ⍝ APL2 VERSION OF FUNCTION
[10]   L2:X←128⍴X
[11]    R←(1↓4 ⎕AT 'X')÷128
[12]    →((R≠1)∨0≠1↑0⍴X)/0
[13]   ⍝ FORCE BYTE INTEGERS TO FULLWORDS
[14]    R←4
```

APLDI–II Migration

4. Modify functions FFN, CODER, DAYS:

```
FFN[24]   ⍎(0=⍴DFW)/'DFW←(W+(⍴PH)⌈⌈DEF[J;2]×
          (0 1 2.5 1 2 1 2 2)[DEF[J;0]]),DEF[J;5]'

CODER[24] SEB←0⍴SEE←'R←(I1←1=+⌿R)/R←(I1,(⍴R)÷I1←',
          (⍕1↑⍴T),')⍴R∩R←I1\R←(1+⍳1↑⍴R)+.×R∩',
          (0≠''⍴⍴Y)/'R←(',(⍕Y),')[R]∩'

DAYS[10]  SEB←0⍴SEE←'R←',',- '[P0],Y,'-R[2;]+('
          (⍕CMO),')[R[1;]]+(365×R[0;])+(0=4|R[0;])
          ∧2<(R←(3⍴100)⊤,R)[1;]∩'
```

5. Modify functions:

```
∇SUB[2.5]       EXFUN←''∇
∇SUBDEI[.5]     EXFUN←''∇
∇SUBDEI3[.5]    EXFUN←''∇
∇DISUM[.5]      EXFUN←''∇
∇IAGN[.5]       EXFUN←''∇
∇UPDAIE[1.5]    EXFUN←''∇
```

APLDI–II Migration

6.)SAVE DIAUXFUL

)LOAD APLDI
)COPY DIAUXFUL ⍙AV DT FFN
)SAVE

)LOAD DIAUX
)COPY DIAUXFUL ⍙AV DT SORT
)SAVE

)LOAD DIUPDATE
)COPY DIAUXFUL FFN
)SAVE

APLDI–II Migration

7. **Under CMS, rename the existing APLDI–II function file:**

```
RENAME APLDIFUN FILE A = OLDFILE =
```

8. **Under VSAPL:**

```
)LOAD DIAUXFUL
INITIAL
200
APLDIFUN FILE
SAVE LISI
)LOAD APLDI
)COPY DIAUX126 DISPGRP
)SAVE FADI
```

APLDI–II Migration

9. **Under APL2,)MCOPY workspaces APLDI, FADI, DIAUX, DIAUX126, DIUPDATE, DICREATE, DIAUXFUL and)SAVE under the same names.**

10. **Modify user APLDI–II workspaces:**

```
)LOAD MYWS
)COPY APLDI ⍙AV DT FFN
)SAVE
```

Conclusions

- Use)MCOPY to convert WS's to APL2
- Use FLAG_ to locate ⎕AV dependencies
- Create bilingual code
  APL2 exploitation later
- Isolate other changes by testing
- APL programmer required
- Read APL2 Migration Guide