APL-Journal
A Programming Language
1-2/2003

In dieser Ausgabe:

Reiner Nußbaum
Einfache Lösung: Eine Entwicklungsumgebung für IBM APL2 in einem allgemeinen Automationskonzept

Peter Antweiler
Kalender und ihre Berechnung mit APL2

David Liebtag
APL2 Calls to Java: GSE Fall 2003

Im Blickpunkt

Herbert Voß
Advanced Encryption Standard

APL-Germany e.V.
Jahrgang 22 Nr. 1-2 2003
ISSN - 1438-4531
RHOMBOS-VERLAG

www.apl-germany.de

Inhaltsverzeichnis

Reiner Nußbaum
Einfache Lösung
Eine Entwicklungsumgebung für IBM APL2 in einem allgemeinen Automationskonzept 4

Peter Antweiler
Kalender und ihre Berechnung mit APL2 7

Peter-Michael Hager
Dyalog Next Generation
Unnamed Window Refs 10

David Liebtag
APL2 Calls to Java
GSE Fall 2003 12

David Liebtag
APL2: What's New? 14

Nancy Wheeler
Storing APL Objects Outside the Workspace 15

Simon Garland
Matrikx reloaded
All work and no k makes jack a dull boy 17

Herbert Voß
Advanced Encryption Standard 19

Aufnahmeformular 24

Impressum 2

Editorial 3

Vorschau

Im nächsten APL-Journal lesen Sie unter anderem:

Geschichte der äußeren Ballistik in Europa

von Albert Badior

4 Einfache Lösung

Es werden die Vorteile gezeigt, wie unter Nutzung eines standardisierten Anwendungsumfeldes die Anwendungsentwicklung vereinfacht werden kann. Dabei muss für die Anwendungsentwicklung nur die Lösung für ein einzelnes Problem realisiert werden. In einem allgemeinen Automationskonzept kann die Anwendung dann unmittelbar für beliebig viele Benutzer / Mandanten eingesetzt werden. Soweit die Anwendung dies unterstützt, ist darüber hinaus auch der parallele Einsatz der Anwendung gegeben.

10 Dyalog Next Generation

Dyalog hatte mit der Version 9 einen weiteren eigenständigen Datentyp neben Variablen, Funktionen und Operatoren eingeführt, die Objekt-Referenz, auch kurz „Ref" benannt. Refs sind Pointer auf Namespaces und können unter anderem zugewiesen, verglichen und formatiert werden. Sie verhalten sich daher sehr ähnlich wie Variablen. Das Resultat der Formatierung ist der Name des Namespaces, auf den die Ref zeigt.

12 APL2 Calls to Java

The IBM APL group is working on an exciting new facility: Calls to Java. Through this facility, you will be able to call Java programs and have those programs call back to APL2. The heart of the calls to Java facility is a new associated processor 14.

17 Matrikx reloaded

Although the array language K is immediately accessible to programmers familiar with APL or J comparatively few have spent any time looking into it. For anyone who wants to add a new language to their toolkit this short article describes how to download and install a trial version - and points to further information for those who would like to learn more.

19 Advanced Encryption Standard

Seit Jahren versuchte man einen geeigneten Nachfolger für den bisherigen *A*dvanced *E*ncryption *S*tandard zu entwickeln. Unter der Federführung des amerikanischen NIST wurde 1997 ein entsprechendes formales Verfahren eingeleitet. Die öffentliche Analyse der einzelnen Verfahren war im Mai 2000 abgeschlossen und es begann die letzte Runde der Kandidatenauswahl, die Ende 2000 mit der Ernennung von Rijndael zum DES-Nachfolger abgeschlossen wurde.

Impressum APL - Journal ISSN 1438-4531

Herausgeber: Dipl.-Math. Dieter Lattermann, Walldorf

Redaktion: Dipl.-Volksw. Martin Barghoorn (verantwortlich)
Technische Universität Berlin
Franklinstr. 28
D-10587 Berlin
Tel. (030) 314 24392
Fax (030) 314 25901
eMail: barg@cs.tu-berlin.de

Erscheinungsweise: halbjährlich

Erscheinungsort: Berlin

Verlag: RHOMBOS-VERLAG, Berlin
Kurfürstenstr. 17
D-10785 Berlin
Tel. (030) 261 9461
Fax (030) 261 6300
eMail: verlag@rhombos.de,
Internet: www.rhombos.de

Gestaltung: RHOMBOS-VERLAG, Berlin

Druck: dbusiness.de GmbH, Berlin

Copyright: *APL Germany e.V.* (für alle Beiträge, die als Erstveröffentlichung erscheinen)

Fotos: NASA (Titel, Seite 4, 10, 12, 19)
M. Barghoorn (Seite 15, 17)

Homepage des APL-Germany e.V.: http://www.apl-germany.de
Homepage des APL-Journals beim Verlag: http://www.rhombos.de/shop/a/show/article/?99

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung von Redaktion und Herausgeber nicht übernommen werden. Sämtliche Veröffentlichungen im APL-Journal erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

Wie immer freuen wir uns auf Ihr Feedback zu diesem Heft. Schreiben Sie uns an barg@cs.tu-berlin.de, oder faxen Sie an: 030 - 314 25901.

Technische Hinweise für Autorinnen und Autoren

Manuskripte werden von der Redaktion entgegengenommen. Sie müssen im Besitz der Nutzungsrechte für die von Ihnen eingereichten Texte, Fotos und Zeichnungen sein. Die Zustimmung zum Abdruck wird vorausgesetzt.

Manuskripte und Beiträge aller Art werden in jeder Form entgegengenommen. Es werden jedoch Beiträge in gespeicherter Form bevorzugt. Bitte speichern Sie Ihren Text im Format Ihrer Textverarbeitung und zusätzlich als RTF-Datei (Rich Text Format). Sie brauchen sich nicht die Mühe machen, Ihren Text aufwendig zu formatieren. Auszeichnungen im Text sollten sich auf „Kursiv“ beschränken. Ihre Datei schicken Sie der Redaktion bitte auf einem Datenträger (Diskette oder CD-ROM). Alternativ hierzu können Sie der Redaktion Ihre Daten auch per Email (barg@cs.tu-berlin.de) schicken.

Wenn Sie Ihr Manuskript einreichen, sollten Sie zusätzlich eine Gliederung und Zusammenfassung Ihrer Arbeit vorlegen.

Wichtig: In der Zusammenfassung (abstract) sollte in wenigen Sätzen die wichtigste Aussage Ihres Textes wiedergegeben sein.

Grafiken

Bitte senden Sie uns einen sauberen Ausdruck der Graphiken auf rein-weißem Papier. Linien (keine Haarlinien!) in den Graphiken sollten Schriften so groß gewählt sein, dass sie auch bei (stark) verkleinerter Wiedergabe im Heft noch zu erkennen sind; auf der Rückseite vermerken Sie bitte Ihren Namen und die Nummer der Abbildung; Zusätzlich wäre es schön, wenn Sie uns die mit einem gängigen Vektor-Grafikprogramm erstellten Zeichnungen und Grafiken auch als Datei liefern. Grafikdateien (Strichzeichnungen) sollten möglichst im Austauschformat EPS (skalierbar) mit 300 bis 600 dpi Auflösung in schwarz-weiß bzw. Graustufe abgespeichert werden. Nach Rücksprache können sie zusätzlich Ihre Grafiken in Ihrem Original-Dateiformat liefern oder in ihr Textdokument einbetten.

Digitale Fotos benötigen wir im TIF-Format mit einer maximalen Auflösung von 250 dpi. Nicht geeignet sind Abbildungen und Fotos, wenn Sie aus dem Internet stammen beziehungsweise im Original als Bildschirmdarstellung (Auflösung 75 - 150 dpi) vorliegen. Ebenfalls nicht geeignet sind Grafiken, die mit einem pixelorientierten Zeichenprogramm erstellt wurden.

EDITORIAL

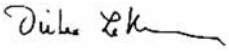
Liebe APL-Freundinnen und -Freunde,

was lange währt, kann eigentlich nur gut werden. Den Autoren möchte ich für ihre Beiträge danken und dem Editor für seine Mühe, sie zusammenzustellen. Es gibt wieder Interessantes zu lesen, diesmal hat fast alles etwas mit APL oder wenigstens Arrays zu tun.

Nichts zu lesen gibt es über eine diesjährige APL-Konferenz – meine Befürchtung, dass APL 2002 in Madrid die letzte „richtige" APL-Konferenz war und der APL-Teil der „Federated Computer Research Conference" im Juni 2003 in San Diego einen Ausklang darstellte, scheint sich zu bewahrheiten. Vielleicht sind solche Konferenzen im Internet-Zeitalter auch nicht mehr zu rechtfertigen, aber schade ist es doch, dass damit auch die persönlichen Kontakte stark leiden. Umso mehr freut es mich, dass sich die vereinten Kräfte des APL-Arbeitskreises in der GSE und von *APL Germany* weiterhin regelmäßig treffen.

Ich wünsche Ihnen viel Freude bei der Lektüre dieses *Journals,*

Ihr



Dieter Lattermann

Reiner Nußbaum

Einfache Lösung

Eine Entwicklungsumgebung für IBM APL2 in einem allgemeinen Automationskonzept

Es werden die Vorteile gezeigt, wie unter Nutzung eines standardisierten Anwendungsumfeldes die Anwendungsentwicklung vereinfacht werden kann. Dabei muss für die Anwendungsentwicklung nur die Lösung für ein einzelnes Problem realisiert werden. In einem allgemeinen Automationskonzept kann die Anwendung dann unmittelbar für beliebig viele Benutzer / Mandanten eingesetzt werden. Soweit die Anwendung dies unterstützt, ist darüber hinaus auch der parallele Einsatz der Anwendung gegeben.

Einleitung

Insbesondere bei der plattformübergreifenden Verarbeitung von Informationen ist in der Regel nicht bekannt,

- wann,
- für wen und
- welche Informationen vorliegen und
- was mit den Informationen zu geschehen hat.

In Heft II/2002 des APL Journals wurde ein allgemeines Automationskonzept 'gift-ALU' vorgestellt, mit dem Informationen quasi ereignisgesteuert verarbeitet werden können. Dieses Konzept ist dadurch gekennzeichnet, daß

- die Verbindung zur Anwendung, Benutzern/Mandanten sowie Verarbeitungsregeln in Verzeichnisbäumen hinterlegt sind,
- individuelle Benutzersichten der Datenorganisation mittels Verzeichnisverweisen realisiert werden
  und
- das Verknüpfen von Arbeitsschritten (Jobabläufe) mittels Verzeichnisverweisen realisiert wird.

Daraus folgt: Die Existenz einer Datei in dem Verzeichnisbaum ist ausreichend, um zu ermitteln,

- was
- für wen
- wie zu erledigen ist.

Aus der Abbildung 1 ist ersichtlich, daß eine Datei, welche sich in dem Verzeichnis ‚empfaenger@de.ibm.com' befindet, für den Benutzer 'Reiner.Nußbaum@[10.73.152.250]' an den Mailempfänger ' empfaenger@de.ibm.com ' zu versenden ist.

Das Verbinden zur konkreten Anwendung geschieht über eine einheitliche Anwendungsumgebung. Diese kann auch für die Entwicklung von Anwendungen bzw. Anwendungsschnittstellen benutzt werden, wie in diesem Beitrag für IBM APL2 gezeigt wird. Eine analoge Vorgehensweise ist für beliebige andere Programmiersprachen (C, ...) möglich.

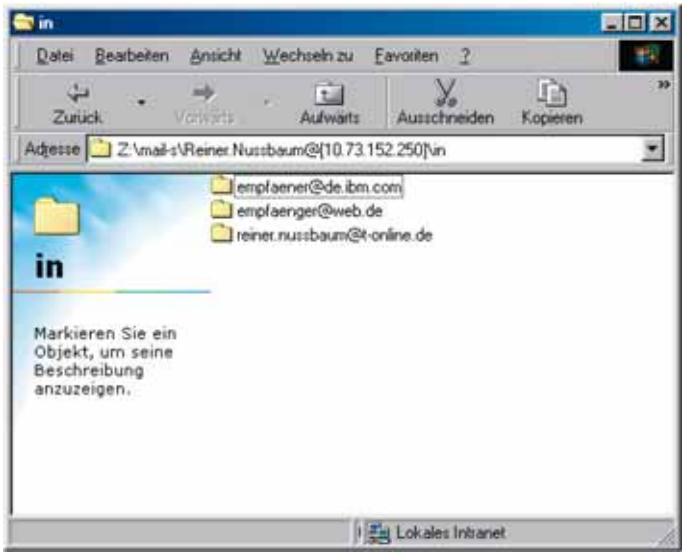


Abb. 1: gift-ALU Mailversand

Die APL2 Entwicklungsumgebung

In Abbildung 2 ist angegeben, wie die APL2 Entwicklungsumgebung in dem Verzeichnisbaum für verschiedene Entwickler dargestellt werden kann. Um die Entwicklungsumgebung zu aktivieren, ist es folgerichtig, daß dazu nur eine geeignete Datei (oder Datei-/Verzeichnisgruppe) als 'Entwicklungsauftrag' vorliegen muß.

Im Rahmen dieses Beitrages nehmen wir als Beispielaufgabe das Formatieren von Graphiken im jpg-Format (320 x 240) sowie das Erzeugen von mp3-Dateien ('low quality' 32 kHz, mono) für PDA-Computer.

Die erste Aufgabe soll darin bestehen, daß ein Bild von einer Digitalkamera von der Originalgröße 2048x1536 in eine PDA-gerechte Größe von 320x240 Bildpunkten umgewandelt wird. In der hier dargestellten Version wird ein Linux-Server benutzt, auf den über LAN-Verbindungen auf das Linux-Dateisystem von gift-

ALU zugegriffen wird. Die Linux Entwicklung erfolgt auf einem Windows Client.

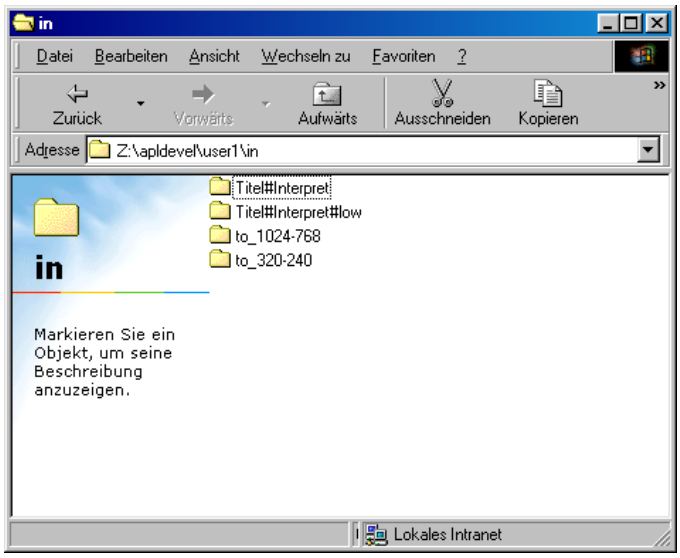


Abb. 2: gift-ALU Beispiel Entwicklungsumgebung

Soweit die in Abbildung 2 angegebene Verzeichnisstruktur vorliegt, ist es für einen ‚Entwicklungsauftrag' ausreichend, eine Bilddatei – hier: dscn0079.jpg – in das Verzeichnis to_320-240 zu kopieren. Sobald eine Datei vorliegt, wird eine IBM APL2 Session gestartet. Auf der Entwicklungsmaschine sind in der IBM APL2 Sitzung nachstehende Informationen sichtbar (vgl. Abbildung 3):

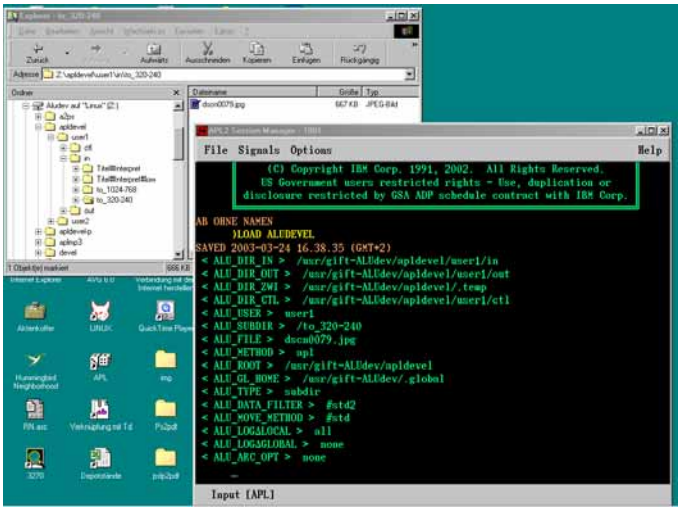


Abb. 3: Informationen in der Entwicklungsumgebung

Anhand der Umgebungsvariablen, welche der Entwicklungstask mitgegeben werden, kann die zu lösende Aufgabe unmittelbar erkannt werden. Es ist klar,

- wo die Datei abgelegt ist,
- wie der Name der Datei ist und
- was mit der Datei zu geschehen hat (in die Größe 320x240 zu transformieren).

Der routinemäßige –eigentlich problemunabhängige- Arbeitsablauf besteht darin, die zu verarbeitende Datei in einen Zwischenbereich zu verschieben, diese entsprechend der konkreten Aufgabe zu verarbeiten und zuletzt das Verarbeitungsergebnis in einem Ausgabebereich abzulegen. Die Routineaufgaben werden mit vorgefertigten Tools erledigt, so daß sich die Kernaufgabe weiter reduzieren läßt. Abbildung 4 zeigt die Routineaufgabe der Bildberarbeitung (ALUAPPL_IMG), Abbildung 5 die eigentliche Problemlösung (ALUAPPL_IMG_PROCESS).

```
EDIT
 [0] RC←ALUAPPL_IMG;CMD;MSG;ALBUM;INTERPRET;TITEL
 [1] ⍝ 00314RNUSS BILDVERARBEITUNG (.JPG DATEIEN RESIZE, ROTATE, FLIP/FLOP)
 [2] ⍝ AUFBEREITEN DER ZUSATZINFORMATIONEN AUS DER DIRECTORYSTRUKTUR
 [3] ⍝
 [4] →(0≠RC←ALU_MVINFI2ZWI '/temp')/ERR_01
 [5] →(0≠RC←ALUAPPL_IMG_PROCESS)/ERR_02
 [6] →(0≠RC←1 ALU_MVZWIFI2OUT((¯4↓ALU_FILE),'.jpg')'temp.jpg')/ERR_03
 [7] →(0≠RC←1 ALU_MVZWIFI2OUT((¯4↓ALU_FILE),'-tn.jpg')'temp-tn.jpg')/ERR_03
 [8] →0
 [9] ERR_01:.⍝FEHLER BEIM KOPIEREN AUS USER_INPUT NACH TEMP
[10] ERR_02:.⍝FEHLER IN DER VERARBEITUNG
[11] ERR_03:.⍝FEHLER BEI DEM UEBERTRAGEN IN DEN AUSGABEBEREICH
[12]
```

Abbildung 4: Routineaufgaben der Bildverarbeitung

In Abbildung 5 wurde in der Zeile 12 der Befehl eingefügt, um das vorliegende Bild in die PDA-gerechte Größe umzuwandeln. Die übrige Funktionalität der bisherigen Bildverarbeitung, wie das Rotieren, Invertieren bzw. Spiegeln von Bildern bleibt dabei erhalten. Benutzt wird hierbei das Linux Paket ‚convert'.

```
EDIT
 [0] ERRSW←ALUAPPL_IMG_PROCESS;X;CMD;PARMR;PARMS
 [1] ⍝ 030324RNUSS: IMG - (CONVERT TO JPG) - THE PROCESSING
 [2] ⍝ ALL IMAGES GO TO THE SAME OUTPUT DIRECTORY
 [3] ⍝ WE KNOW ROOT,USER, DIR_IN,DIR_ZWI,DIR_OUT,SUBDIR, FILE
 [4] ⍝
 [5] PARMR←PARMS←''
 [6] ⍎(∨/'r090'∊ALU_SUBDIR)/'PARMR←'' -rotate 90 '''
 [7] ⍎(∨/'r180'∊ALU_SUBDIR)/'PARMR←'' -rotate 180 '''
 [8] ⍎(∨/'r270'∊ALU_SUBDIR)/'PARMR←'' -rotate 270 '''
 [9] ⍎(∨/'-flip'∊ALU_SUBDIR)/'PARMR←PARMR,'' -flip '''
[10] ⍎(∨/'-flop'∊ALU_SUBDIR)/'PARMR←PARMR,'' -flop '''
[11] ⍎(∨/'-invert'∊ALU_SUBDIR)/'PARMR←PARMR,'' -negate '''
[12] ⍎(∨/'-240'∊ALU_SUBDIR)/'PARMS←'' -scale  320 '''
[13] ⍎(∨/'-480'∊ALU_SUBDIR)/'PARMS←'' -scale  640 '''
[14] ⍎(∨/'-600'∊ALU_SUBDIR)/'PARMS←'' -scale  800 '''
[15] ⍎(∨/'-768'∊ALU_SUBDIR)/'PARMS←'' -scale 1024 '''
[16] ⍎(∨/'-1200'∊ALU_SUBDIR)/'PARMS←'' -scale 1600 '''
[17] ⍎(∨/'-1536'∊ALU_SUBDIR)/'PARMS←'' -scale 2048 '''
[18] CMD←'convert ',PARMS,PARMR,ALU_DIR_ZWI,'/temp ',ALU_DIR_ZWI,'/temp.jpg' ⍝
[19] →(0≠⍴X←HOSTΔEXEC CMD)/ERR,ERRSW←2   ⍝ NO RESULT ≡ OK
[20] CMD←'convert -scale 120 ',PARMR,ALU_DIR_ZWI,'/temp ',ALU_DIR_ZWI,'/temp-t
[21] →(0≠⍴X←HOSTΔEXEC CMD)/ERR,ERRSW←2.1   ⍝ NO RESULT ≡ OK
[22] →ERRSW←0                              ⍝ FINITO
[23] ERR:CMD←'IMG IN:',DIR_IN,' SUB:',SUBDIR,' FILE:',FILE,' RC=',⍕ERRSW
[24]
```

Abb. 5: Bildverarbeitung im Detail.

Wenn abschließend nicht vergessen wird, den APL Arbeitsbereich zu speichern, sind die Änderungen permanent verfügbar.

Die IBM APL2 Entwicklungsumgebung im parallelen Betrieb

Nachfolgend wird der parallele Betrieb von Anwendungen dargestellt. Hierzu wird ein analoger Verzeichnisbaum wie in Abbildung 2 eingerichtet. Um die Entwicklungsumgebung zu aktivieren, muß folgerichtig nur eine geeignete Dateien (oder Datei-/Verzeichnisgruppen) als 'Entwicklungsaufträge' vorliegen.

Dazu greifen wir erstens die oben bereits beschriebene Aufgabe der Bildverarbeitung für PDAs auf und nehmen das Erstellen von mp3-Dateien in niedriger Qualität hinzu. In der hier dargestellten Version wird ein Linux-Server benutzt, auf den über LAN-Verbindungen auf das Linux-Dateisystem von gift-ALU zugegriffen wird. Die Linux-Entwicklung erfolgt auf einem Windows-Client. Selbstverständlich könnten die Terminal-Sessions für die Entwicklung auch auf unterschiedlichen Clients ablaufen.

Nachdem die Bilddatei und die Musikdatei in ein geeignetes Verzeichnis kopiert wurden, sind auf der Entwicklungsmaschine nachstehende IBM APL2-Sitzungen aufgebaut.

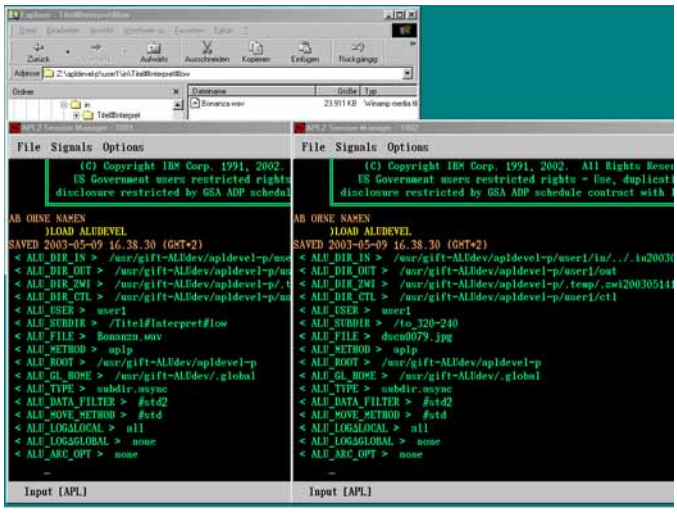


Abb. 6: Informationen in der parallelen Entwicklungsumgebung

Wir sehen hieraus, daß das parallele Konzept von gift-ALU den entscheidenden Vorteil hat, daß sich die Anwendung um den Parallelbetrieb nicht zu kümmern hat. Selbst wenn zwei verschiedene Dateien in dem gleichen Verzeichnis wären, so wird von gift-ALU für die Verarbeitung jeder einzelnen Datei ein nach gleichen Regeln aufgebautes Anwendungsumfeld aufgebaut. Da wir uns in dem ersten Anwendungsbeispiel (Bildverarbeitung) auf die Variablen des Anwendungsumfeldes bezogen haben ist klar, daß die Bildverarbeitung auch im parallelen Betrieb lauffähig ist. Daher können wir uns nun dem Erzeugen von mp3-Dateien für PDA Computer zuwenden. Aus Speicherplatzgründen und der Tatsache, daß der PDA ohnehin nur einen Lautsprecher besitzt, wollen wir eine mp3-Datei erzeugen, welche nur Monoinformationen enthält und in der Bitstreamrate auf 32kb reduziert ist. Die Anwendung wird, wie bei der Bildverarbeitung, über den Verzeichnisnamen gesteuert, in dem die ursprüngliche Musikdatei abgelegt ist. Nachstehende Regeln sind vorgesehen:

- Der Verzeichnisname besteht aus Album [#Interpret[#highqual|#low]]

Bei dem Erzeugen der mp3-Datei sollen als beschreibende Informationen der Name des Albums sowie des Interpreten, getrennt durch ‚#', mit in die Headerinformation aufgenommen werden. Die Steuerung der Qualität kann durch ein angehängtes Schlüssenwort ‚highqual' bzw. ‚low', ebenfalls durch ‚#' getrennt, zum Ausdruck gebracht werden.

- Für den Titel selbst, kann der Dateiname der Eingabedatei verwendet werden.

Wie bei der Bildverarbeitung bereits beschrieben, kann die gesamte Verarbeitung getrennt werden in routinemäßige Arbeiten und die spezielle Problemlösung. Die routinemäßigen Aufgaben bei dem Erstellen einer mp3-Datei aus Abbildung 7 ersichtlich.

Zum Erstellen von mp3-Dateien wird das Linux Paket lame benutzt. Für die Problemlösung, das Erstellen von .mp3 Dateien für PDA's, muß der Qualitätsparameter (#low) am Ende des Verzeichnisnamens erkannt und in den entsprechenden Parameter von ‚lame' umgesetzt werden.

Die Aufgabe wird in Abbildung 8 mit dem Einfügen von Zeile 17 erledigt.

Als Konsequenz der Verarbeitung ergibt sich, daß eine wave Datei von etwa 23MB auf eine PDA gerechte Größe von 500kB reduziert wird.

```
EDIT
[0] RC←ALUAPPL_WAV2MP3;CMD;MSG;ALBUM;INTERPRET;TITEL
[1] ⍝ 00314RNUSS WANDLE .WAV DATEI IN MP3 DATEI UM
[2] ⍝ AUFBEREITEN DER ZUSATZINFORMATIONEN AUS DER DIRECTORYSTRUKTUR
[3] ⍝
[4] →(0≠RC←ALU_MVINFI2ZWI '')/ERR_01
[5] →(0≠RC←ALUAPPL_WAV2MP3_PROCESS)/ERR_02
[6] →(0≠RC←ALU_MVZWIFI2OUT((¯4↓ALU_FILE),'.mp3')'lameout.mp3')/ERR_03
[7] →0
[8] ERR_01:.⍝FEHLER BEIM KOPIEREN AUS USER_INPUT NACH TEMP
[9] ERR_02:.⍝FEHLER IN DER VERARBEITUNG
[10] ERR_03:.⍝FEHLER BEI DEM UEBERTRAGEN IN DEN AUSGABEBEREICH
```

Abb. 7: Routineaufgaben zum Erzeugen von mp3-Dateien

```
EDIT
[0] RC←ALUAPPL_WAV2MP3_PROCESS;CMD;MSG;ALBUM;INTERPRET;TITEL;LOW_W_SW;SW_QUAL
[1] ⍝ 00314RNUSS WANDLE .WAV DATEI IN MP3 DATEI UM
[2] ⍝ AUFBEREITEN DER ZUSATZINFORMATIONEN AUS DER DIRECTORYSTRUKTUR
[3] ⍝
[4] ⍝ KOSMETIK - ALBUM,INTERPRET (SUBDIR ohne / bis #), SONG NAME
[5] ⍝ 'LOW QUALITY' FUER LANGSAME INTERNETVERBINDUNGEN NACH 2TEM #
[6] ⍝
[7] ⍝ HIGH QUALITY PARMS -q 0 -b 256
[8] ⍝
[9]
[10] TITEL←'''',((30⌊⍴TITEL)↑TITEL←¯4↓ALU_FILE),''''
[11] ALBUM←'''',((30⌊⍴ALBUM)↑ALBUM←(¯1+ALBUM⍳'#')↑ALBUM←1↓ALU_SUBDIR),''''
[12] INTERPRET←'''',((30⌊⍴INTERPRET)↑INTERPRET←(INTERPRET⍳'#')↓INTERPRET←1↓ALU
[13] LOW_W_SW←'#'∊INTERPRET
[14] ⍎LOW_W_SW/X←'INTERPRET←((¯1+INTERPRET⍳''#'')↑INTERPRET),''''''''
[15] SW_QUAL←''
[16] ⍎('#highqual'≡¯9↑ALU_SUBDIR)/'SW_QUAL←'' -q 0 -b 256 '''
[17] ⍎('#low'≡¯4↑ALU_SUBDIR)/'SW_QUAL←'' -b 32 -m m  ''' ⍝ EHNINGEN DEMO
[18] ⍝
[19] ⍝ AUFBAU DES PROGRAMMAUFRUFES
[20] ⍝
[21] CMD←'cd ''',ALU_DIR_ZWI,''' ; nice -n 19 lame --silent ',SW_QUAL
[22] CMD←CMD,' --tc ''<gift-ALU> using <lame>'''
[23] ⍎(2<⍴ALBUM)/'CMD←CMD,'' --tl '',ALBUM '
[24] ⍎(2<⍴INTERPRET)/'CMD←CMD,'' --ta '',INTERPRET '
[25] ⍎(2<⍴TITEL)/'CMD←CMD,'' --tt '',TITEL '
[26] CMD←CMD,' ''',ALU_DIR_ZWI,'/',ALU_FILE,''' ''',ALU_DIR_ZWI,'/lameout.mp3'
[27] ⍝
[28] ⍝ DOIT.
[29] ⍝
[30] →(RC←0≠⍴MSG←HOST⍎EXEC 'WAV2MP3' ALU_TRC2 CMD)/ERR01
[31] →0
[32] ERR01:.
[33]
F3:quit 4:file  5:top 6:bot  7:line  8:ins s-8:del  9:exec  10:copy   Esc:undo
```

Abbildung 8: Erzeugen von mp3-Dateien im Detail

Zusammenfassung

Abschließend sei erwähnt, daß für einen Produktivbetrieb die IBM APL2 Entwicklungsumgebung nicht vorhanden sein muß. Wenn aus den entwickelten Anwendungen ein APL2 Namespace erstellt wird, dann kann der Betrieb mit der IBM APL2 runtime - Umgebung erfolgen.

Das Heranziehen einer standardisierten Anwendungsumgebung kann die Entwicklungsarbeit erheblich vereinfachen. Dies gilt besonders, weil auf einfache Weise standardisierte Werkzeuge zur Verfügung gestellt werden können, welche auf den Umgebungsvariablen basieren. Damit wird der Routineteil bei der Anwendungsentwicklung erleichtert. Bei der Anwendungsentwicklung ist jeweils nur ein einfaches Problem zu lösen, wie hier bei der Bildverarbeitung und dem Erstellen von mp3 Dateien gezeigt wurde. Als Ergebnis ist die so erstellte Anwendung allgemein nutzbar. Sie kann unverändert für viele Benutzer und viele (unterschiedliche) Detailprobleme eingesetzt werden. Gleichzeitig ist auch der parallele Einsatz für einen oder beliebig viele Benutzer möglich.

Kontakt:

Reiner Nußbaum,
eMail: reiner.nussbaum@web.de

I. Astronomische Grundlagen von Kalendern

Die wichtigsten astronomischen Zyklen sind der Tag, das Jahr und der Monat. Der Tag basiert auf der Umdrehung der Erde um ihre Achse, das Jahr auf der Bewegung der Erde um die Sonne und der Monat auf der Umdrehung des Mondes um die Erde. Kalender sind kompliziert, weil diese Zyklen nicht aus einer ganzzahligen Anzahl von Tagen bestehen und weil die Zyklen weder konstant noch exakt kommensurabel sind.

Das tropische Jahr ist definiert als die mittlere Länge zwischen den Frühlings-Tag- und Nachtgleichen und bestimmt somit den Lauf der Jahreszeiten.

Der synodische Monat ist die mittlere Länge zwischen den Neumonden. Und bestimmt damit die zeitliche Abfolge der Mondphasen.

Ist Jd die aktuelle Julianische Tagesnummer(siehe unten) und das Zeitargument T = (Jd-2451545)/36525, dann gilt bis auf kleine Korrekturglieder in höheren Potenzen von T:

Länge tropisches Jahr:	365.2421896698	-	T×0.00000615359	Tage
Länge synodischer Monat:	29.5305888531	+	T×0.00000021621	Tage

Diese mittleren Längen können bei der Sonne um mehrere Minuten, beim Mond um mehrere Stunden von den wahren Längen abweichen. Aus den Formeln erkennt man, dass sich die Zyklen langsam mit der Zeit verändern. Außerdem sind die Formeln nicht exakt, sondern sind nur die heute am besten verfügbaren.

Ein Kalenderjahr mit einer ganzen Zahl von Tagen kann nie mit dem tropischen Jahr synchronisiert werden. Eine ungefähre Anpassung der Kalendermonate an die Mondphasen erfordert eine komplizierte Folge von Monaten mit 29 und 30 Tagen. Ein Mondjahr mit zwölf synodischen Monaten hat eine ungefähre Länge von 35.436.707 Tagen.

Diese Situation führt zu einer Unterscheidung von drei Kalendertypen:

1. Der Sonnenkalender: Er entsteht durch eine Anpassung des Kalender-Jahres an das tropische Jahr. Im Schaltjahr wird ein Tag eingeschoben. Beispiele sind der gregorianische, der persische und der französische Revolutionskalender.
2. Der Mondkalender: Beispiel ist der islamische Kalender. Er folgt den Mondphasen ohne Anpassung an das tropische Jahr. Daher verschieben sich die Monate bezogen auf zum Beispiel den gregorianischen Kalender.
3. Der Mond/Sonnenkalender: Beispiele sind der chinesische und der Jüdische Kalender. Diese Kalender haben eine Monatsfolge, die sich nach den Mondphasen richtet. Aber alle paar Jahre wird ein Schaltmonat eingefügt, um den Kalender dem tropischen Jahr Anzupassen.

Hier kann eine weitere Unterscheidung der Kalender erfolgen:

4. Arithmetische Kalender: Bei diesen erfolgt die Bestimmung von Jahreslängen und Mondphasen mit Algorithmen, die die wahren astronomischen Ereignisse mehr oder weniger genau abbilden. Beispiele sind der Julia-Nische, gregorianische, jüdische und islamische Kalender. Stellen sich diese Algorithmen als in ihrer Genauigkeit unzureichend heraus, können sie durch verbesserte ersetzt werden. Dies führte im Jahr 1582 zur gregorianischen Kalenderreform. Die römisch-katholische Kirche war damals die mächtigste politische Kraft und konnte diese Reform auch durchsetzen. Heute ist der gregorianische Kalender weltweiter Standard.

5. Astronomische Kalender: Beispiele sind der chinesische, persische und französische Revolutionskalender. Jahreslängen und Mondphasen werden mit astronomischen Rechenverfahren bestimmt. Diese Methoden sind heute so genau, dass sie für Jahrhunderte in der Vergangenheit und Zukunft präzise Ergebnisse liefern. Sie erfordern andererseits hohe Rechenzeiten und sind am bequemsten mit Computern zu erstellen. Hinzu kommt aber eine andere Schwierigkeit: astronomische Berechnungen erfolgen in terrestrischer dynamischer Zeit (Tdt). Sie ist eine gleichförmig verlaufende Zeit und wird durch Atomuhren kontrolliert. Kalender laufen aber in Weltzeit (Ut). Diese ist von der Rotation der Erde um ihre Achse abhängig und unterliegt unvorhersagbaren Schwankungen. Zur Zeit beträgt die Differenz Tdt - Ut etwa eine Minute. In historischen Zeiträumen kann diese Differenz auf Stunden anwachsen und daher bei Kalendern zur Abweichung von dem exakten Eintreffen astronomischer Ereignisse führen.

Aus Punkt 4 und 5 folgt, dass es weder arithmetische noch astronomische Kalender geben kann, die über Zehntausende von Jahren ihre Gültigkeit behalten.

Kalender können ganzzahlige Zyklen enthalten, die keine astronomischen Grundlagen haben. Dazu gehört die Woche mit ihrer 7-Tage-Folge. Ihr Ursprung ist unsicher. Die Zahl 7 hat in vielen Kulturen eine mystische, meist glückbringende Bedeutung. Es ist kein Zufall, dass die Bezeichnungen von Boeing-Flugzeugen immer die 7 enthalten. Andererseits ist die Zahl 7 nützlich (wenn auch nicht exakt), um die Tage zwischen vier Mondphasen zu zählen.

II. Der julianische und gregorianische Kalender

Der julianische Kalender wurde -45 von Julius Caesar eingeführt. Er ist ein Sonnenkalender mit jeweils fester Monatslänge. Jedes vierte Jahr wurde ein Schalttag eingefügt, um das Kalenderjahr Dem tropischen Jahr anzugleichen. Ein julianisches Jahr (Jj) ist Schaltjahr, wenn gilt:

```
---> ((Jj Mod 4) = 0) <---
```

Das mittlere Jahr bekommt damit eine Läge von 365,25 Tagen. Die Länge des tropischen Jahres beträgt jedoch 365,2422 Tage, so dass das julianische Jahr sich langsam durch die Jahreszeiten bewegt. Das Konzil von Nicaia hat 325 den Kalender für die Christenheit anerkannt.

Dehnt man die Regeln des julianischen Kalenders auf den Zeitraum vor Caesars Reform aus, so erhält man den sogenannten ‚Proleptischen' julianischen Kalender. Er bietet ein einfaches chronologisches System für Umrechnung von anderen Kalendern und ist Basis für die julianischen Tagesnummern. Teile der orthodoxen Kirche benutzen noch heute den Julianischen Kalender für die Festlegung kirchlicher Feiertage.

Im 16. Jahrhundert war die Abweichung zum tropischen Jahr auf 10 Tage angewachsen. Nachtgleichen und Neumonde traten zu immer früheren Kalenderdaten ein. Dies wirkte sich auch auf das Osterdatum aus, das vom Datum des ersten Vollmonds an oder nach der Frühlings-Nachtgleiche abhängig ist. Diese Abweichung konnte nicht mehr ignoriert werden.

Papst Gregor XIII setzte 1582 Feb 24 mit seiner Bulle ‚Inter Gravissimus' seine Reform in Kraft. Sie umfasste zwei Korrekturen:

1. Im Oktober 1582 fielen 10 Tage aus. Auf Oktober 4 folgte Oktober 15. Die Wochentagsfolge wurde aber beibehalten. In 400 Jahren sollten drei Schaltjahre ausfallen. In allen vollen Jahrhunderten wird nur Geschaltet, wenn sie durch 400 teilbar sind. Die mittlere Länge des Kalenderjahres wird so auf 365,2425 Tage gebracht. Nach etwa 2500 Jahren führt dies zu einer Abweichung von einem Tag zum tropischen Jahr.
2. Die Mondphasen wurden durch einen verbesserten Algorithmus - den Lilianischen Epakten - den wahren Mondphasen angepasst. Nicht alle Länder übernahmen sofort diese Reform. Russland führte erst ab 1918 den gregorianischen Kalender ein. Der Name ‚Oktober Revolution' bezieht sich noch auf den Julianischen Kalender, obwohl sie bezogen auf den gregorianischen Kalender im November 1917 stattfand. Der gregorianische Kalender ist heute internationaler Standard weltweit. Obwohl andere Kalender auch heute noch benutzt werden, sind sie beschränkt auf einzelne Religionen und Kulturen.

III. Die julianische Tagesnummer (Jd: Julian Day Number)

Vorbemerkung: Alle Datumsangaben erfolgen immer in der Form: Jahr Monat Tag Stunde Minute Sekunde.

Das System der julianischen Tagesnummern ist eine kontinuierliche Zählung der Tage Ab -47 12 Jan 1. Oft wird Jd als ‚Julianisches Datum' bezeichnet. Das ist ein Missverständnis, denn ein julianisches Datum ist ein Datum im julianischen Kalender und keine Tagesnummer.

Tage als Zähleinheit der Chronologie zu wählen beruht auf einer Idee von Joseph Scaliger (1540-1609). Er nannte seine Zählung ‚Julianisch' zu Ehren seines Vaters Julius Scaliger. Die Idee wurde von Astronomen übernommen (John Herschel, 1849) und bietet eine bequeme Möglichkeit Datums-Abstände in Tagen zu finden und Kalenderumrechnungen durchzuführen. Außerdem lässt sich auf einfache Weise zu jedem Datum der entsprechende Wochentag finden: Aus dem Datum wird Jd gebildet (siehe unten), der Rest bei Division durch 7 ergibt dann die Tagesnummer innerhalb der Woche. Ist Jd ganzzahlig, so entspricht dies einem Datum 12 Uhr. Zum Beispiel: 2004 Mai 23. 12 Uhr = JD 2453149

Für Astronomen, die nachts beobachteten, hatte dies die Annehmlichkeit, dass für die Ereignisse in einer Nacht die julianische Tagesnummer nicht wechselt.

Die Umrechnung eines julianischen/gregorianischen Datums in eine jd erfolgt mit der Apl2-Funktion `'jdfromdate'`. Die Umkehrfunktion ist: `'datefromjd'`. Im linken Argument der Funktionen kann angegeben werden, ob im julianischen (j) oder gregorianischen (g) Kalender gerechnet werden soll. Werden die Funktionen ohne linkes Argument verwendet, dann gilt ab 1582 Okt 15 der gregorianische, bis 1582 Okt 14 der julianische Kalender.

IV. Der Arbeitsbereich ‚APLKAL'

Dieser Arbeitsbereich ist auf folgende Weise verfügbar:

Geben Sie in Ihrem Internet - Browser folgende Adresse ein: HTTP://APL2.PCS-KRETSCHMER.DE. Auf der folgenden Internetseite Können Sie den Workspace Aplkal und Beschreibungen herunterladen.

`'APLKAL'` enthält APL2-Programme zur Behandlung der Kalender:

Kennzeichen und Kalender		Kalender-Typ
B:	babylonisch	arithmetisch/solar
C:	chinesisch	astronomisch/lunisolar
F:	französisch	astronomisch/solar
G:	gregorianisch	arithmetisch/solar
H:	hebräisch/jüdisch	arithmetisch/lunisolar
I:	islamisch/moslemisch	arithmetisch/lunar
J:	julianisch	arithmetisch/solar
K:	koptisch	arithmetisch/solar
P:	persisch	astronomisch/solar

Das Programm `'PRINTHOW'` erstellt die Beschreibung fertiger Routinen zum Ausdruck von Kalendern sowie Hinweise zur Kalender-Konvertierung und Datumsprüfung.

V. Datumsprüfung und Kalender-Konvertierung

In Abschnitt III wurden die APL2-Programme `'JDFROMDATE'` und `'DATEFROMJD'` schon erwähnt. Das folgende Beispiel zeigt, wie sie zur Datumsprüfung und Kalenderkonvertierung eingesetzt werden können. Beispiel: ist das Datum 1900 2 29 im gregorianischen Kalender gültig?

Umwandlung in `JD`: `JD = 'G'JDFROMDATE 1900 2 29`, ergibt `JD=2415079.5` `JD` zurück in das Datum: `'G'DATEFROMJD 2415079.5` ergibt: `1900 3 1`. Das letzte Datum stimmt nicht mit dem Ausgangsdatum überein. Im gregorianischen Kalender ist `1900 2 29` kein gültiges Datum!

Allgemein kann das Programm ‚L Checkdate R' verwendet werden: im rechten Argument wird ein Datum (Jahr, Monat, Tag) als Vektor oder Matrix angegeben, im linken Argument das Kennzeichen des Kalenders. Das Ergebnis ist eine Matrix mit M[;1] = 1/0 (gültig/ungültig) M[;2 3 4] = Datum, Beispiel: ist 4702 2 12 ein gültiges Datum im chinesischen Kalender?

`R<-'C' Checkdate 4702 2 12` ergibt: `R = 1 4702 2 12`, das Datum ist gültig.

Kalender-Konvertierungen erfolgen immer nach demselben Schema: Das Datum in irgendeinem Kalender wird in die julianische Tagesnummer umgerechnet. Aus dieser wird das Datum eines anderen Kalenders ermittelt. Für jeden Kalender existieren zwei APL2-Programm-'Paare': `'xxxFROMJD'` und `'JDFROMxxx'`, wobei `'xxx'` eine Abkürzung für den Namen des Kalenders ist. Wie diese Programme eingesetzt werden können, ersieht man aus dem APL2-CODE der Programme `'CONVFNS'` und `'CONVDATE'`.

Beispiel für eine Kalenderkonvertierung: welchem gregorianischen Datum entspricht das julianische Datum 2004 1 20? Das wird vom Programm `'CONVDATE'` beantwortet: 2004 2 2 = `'JG'CONVDATE 2004 1 20 2004 JAN 20` julianisch entspricht `2004 FEB 2` gregorianisch.

Im rechten Argument steht ein Datum in der Form Jahr, Monat, Tag (Vektor oder Matrix), das 1. Element des linken Arguments bezeichnet den Kalender, in dem sich dieses Datum befindet, das 2. Element gibt an, in welchen Kalender das Datum konvertiert werden soll.

Achtung:

In den APL2-Programmen zum chinesischen Kalender erscheint die Monatsnummer in der Datumsangabe negativ, wenn es sich um einen Schaltmonat handelt. Weil der chinesische Kalender nur die Monatsnummern 1 bis 12 kennt, ist das negative Vorzeichen nur ein Kennzeichen für Schaltmonate. Es hat keine arithmetische Bedeutung.

Das gregorianische Datum 2004 3 22 soll in das entsprechende chinesische Datum umgewandelt werden:

```
          4702 ¯2 2 = 'GC' CONVDATE 2004 3 22
Kontrolle: 2004 3 22 = 'CG' CONVDATE 4702 ¯2 2
```

VI. Die WOCHE

Die erste Woche eines Jahres ist die, die den Donnerstag enthält. Der Zusammenhang zwischen einem Datum und der dazugehörenden Woche wird mit den APL2 Programmen ‚JDFROMWEEK' und ‚WEEKFROMJD' hergestellt.

▶ Beispiel 1: Gesucht Wochennummer und Wochentag für 2004 2 16. Umwandlung dieses Datums in die julianische Tagesnummer ergibt: 2453051.5 = `JDFROMDATE 2004 2 16`. `WEEKFROMJD 2453051.5` ergibt: `2004 8 1`. Das Datum fällt in die 8. Woche des Jahres 2004 und ist ein Montag. Umgekehrt ergibt `JDFROMWEEK 2004 8 1` wieder die julianische Tagesnummer: `2453051.5`, die mit der Funktion `DATEFROMJD` wieder in das Ausgangsdatum `2004 2 16` umgewandelt wird.

▶ Beispiel 2: `WEEKFROMJD JDFROMDATE 2005 1 1` ergibt: `2005 53 6`. Der Neujahrstag 2005 fällt in die 53. Woche des Vorjahres und ist ein Samstag.

Der Workspace `'APLKAL'` enthält Programme zum Berechnen weiterer Kalender: Longcount der Maya, äthiopischer, japanischer und römischer Kalender. Ferner sind zahlreiche Funktionen für das Erstellen eigener Druck-Programme vorhanden.

Literatur

1 H. Aslaksen, The mathematics of the chinese calendar, manuscript, department of mathematics, national university of Singapore,1999.
2 L. Basnitzki, Der jüdische Kalender, Jüdischer Verlag, Frankfurt 1998.
3 P. Bretagnon and J. L. Simon, planetary programs and tables from -4000 TO +2800, Willmann-Bell, INC., Richmond, VA, 1986.
4 D. A. von den Brincken, historische Chronologie des Abendlandes, Kohlhammer Verlag, Stuttgart 2000.
5 L. E. Dogget, 'Calendars', explantory supplement to the Astronomical Almanac, P. K. Seidelmann, ED., University science books, Mill Valley, Ca, 1992.
6 H. Grotefend, Taschenbuch der Zeitrechnung, Hahnsche Buchhandlung, Hannover 1991.
7 O. L. Harvey, calendar conversions by way of the Julian day number, American philosophical society, Philadelphia 1983.
8 J. Meeus, astronomical algorithms, 2nd ED.,Willmann-Bell,Inc., Richmond, VA, 1998.
9 E. M. Reingold and N.Dershowitz, Calendrical Calculations: The Millenium edition, Cambridge university press, Cambridge 2001
10 W. I. Seleschnikow, Wieviel Monde hat ein Jahr?, Aulis Verlag, Köln 1981.
11 H. ZEMANEK, Kalender und Chronologie, Oldenbourg Verlag, München 1984

Kontakt

Peter Antweiler, eMail: antweiler@freenet.de

Peter-Michael Hager

Dyalog Next Generation

Unnamed Window Refs

Dyalog hatte mit der Version 9 einen weiteren eigenständigen Datentyp neben Variablen, Funktionen und Operatoren eingeführt, die Objekt-Referenz, auch kurz „Ref“ benannt. Refs sind Pointer auf Namespaces und können unter anderem zugewiesen, verglichen und formatiert werden. Sie verhalten sich daher sehr ähnlich wie Variablen.

Das Resultat der Formatierung ist der Name des Namespaces, auf den die Ref zeigt. Ist dieses Namespace mit dem voll qualifizierten Namen des Ref identisch, ist die Ref das Objekt selbst, also eine Selbst- oder Eigen-Referenz.

Refs müssen nicht Referenzen benannter Objekte sein. Mit ⎕NS'' können unbenannte (unnamed) Refs erzeugt werden. Das formatierte Resultat eines solchen unbenannten Ref sieht wie folgt aus:

```
    ⍕oRef←⎕NS''
 #.[Namespace]
```

Statt des Objekt-Namens wird bei unbenannten Refs nur noch die Objekt-Klasse, hier Namespace, zur Unterscheidung in eckigen Klammern, angezeigt.

Objekt-Refs sind von der Struktur her Skalare, die genauso wie andere Skalare zu Vektoren, Matrizen, usw. mit verbaut werden können, und auch den Regeln der Parallelverarbeitung von Matrizen folgen:

```
    os←⎕NS¨4⍴⊂''
    os[1].F←+
    os[2].F←-
    os[3].F←×
    os[4].F←÷
      8 os.F ⍳4
9 6 24 2
      os[1 3].(F/⍳10)
55 3628800
```

Seit der Version 10 nun funktionieren diese unbenannten Refs über eine Hilfsfunktion, in Form einer Root-Methode, auch für die Window/GUI-Klassen:

```
#.CreateObject←{α←⍎↑⎕NSI ⋄ ⍵{⍵{α{α}α.
##.⎕EX'o'}'o'⍵.##.⎕WC α}α.o←α.⎕NS''}
```

Syntaktisch ist die rechte Seite von `CreateObject` der von ⎕WC identisch.

Das Resultat von `CreateObject` ist jedoch eine unbenannte Objekt-Referenz, also nach allgemeinem Sprachgebrauch eine Instanz der angegebenen Klasse. Wird diese nicht zugewiesen, oder gelöscht, verschwindet sie auch wieder.

Optional kann als linkes Argument die Eltern-Objekt-Referenz angegeben werden:

```
      )CS #.Samples
#.Samples
      oForm  ←      #.CreateObject'Form'
      oButton←oForm
#.CreateObject'Button'
      ⍕oForm
#.Samples.[Form]
      ⍕oButton
#.Samples.[Form].[Button]
      osObject←oForm oButton
      ⍕osObject
 #.Samples.[Form]
#.Samples.[Form].[Button]
```

Gegenüber dem bisherigen Konzept bieten die Window-Refs eine Reihe von Vorteilen:

- Die Performance ist erheblich besser, da der Interpreter nicht mehr textuelle Namensketten analysieren muss, um das Objekt

zu finden, sondern mit der Referenz direkt einen Zeiger auf das Objekt hat, und auch keinen Namensbaum mehr verwalten muss.

- Das Design der grafischen Oberfläche (Skin), die zugrunde liegende Funktionalität (Callbacks), und die Lokalisierung (Sprache) können programmtechnisch getrennt werden:

```
[0]   Localize Parms;osObject;Language
[1]   osObject Language←Parms
[2]   :Select Language
[3]   :Case 'ENG'
[4]       osObject.Caption←'Sample Form' 'Send'
[5]   :Case 'GER'
[6]       osObject.Caption←'Beispiel-Form' 'Senden'
[7]   :EndSelect

      Localize osObject 'GER'
```

- Eine beliebige Anzahl von Instanzen, wie etwa für MDIs oder Services erforderlich, kann mühelos gebildet werden:

(Programme #.Samples.OneForm und #.Samples.SixForms)

Der einzige Wehrmutstropfen derzeit ist, dass die Implementierung noch nicht durchgängig ist. So sollte die CreateObject-Methode, (oder wie sie einmal heißen wird), in jeder Klasse vorhanden sein, und referenzierende Eigenschaften von Objekten, wie `FontObj`, `IconObj`, `SplitObj`, sollten als Refs angebbar sein.

Kontakt

HAGER-ELECTRONICS GmbH, Peter-Michael Hager, Hamburger Str. 97, D-44135 Dortmund, eMail: Hager@t-online.de, hager@dortmund.net

Vector Front Page - Journal of the British APL Association http://old.vector.org.uk.novaredirects.com/

How to use this site | About APL | APL Fonts | BAA News | Subscribing to Vector | Products | Events | [More...]

Vector 20th birthday celebrations

Visit the Vector Wiki

VECTOR

Vol.20 No.2
October 2003

The Journal of the British APL Association

Dyalog 2003 Conference ...
- Conference Report 44
- Take Care in Sweden 63
- Daintree's CD Database 70
- "Now I Know" (Peelle) 92
- Clark on Formula Inversion 100
- Crossley's Variable Editor 132

- Contents
- Editorial
- Conference Reports: Dyalog 2003 Conference
- "Now I Know"
- Recent Vectors
- Resources
- Quick Reference Diary
- Product Guide
- Bookmarks
- Search the Index

Contributing to Vector | Order form for back issues | Resources | Advertising | Contacts | Jobs

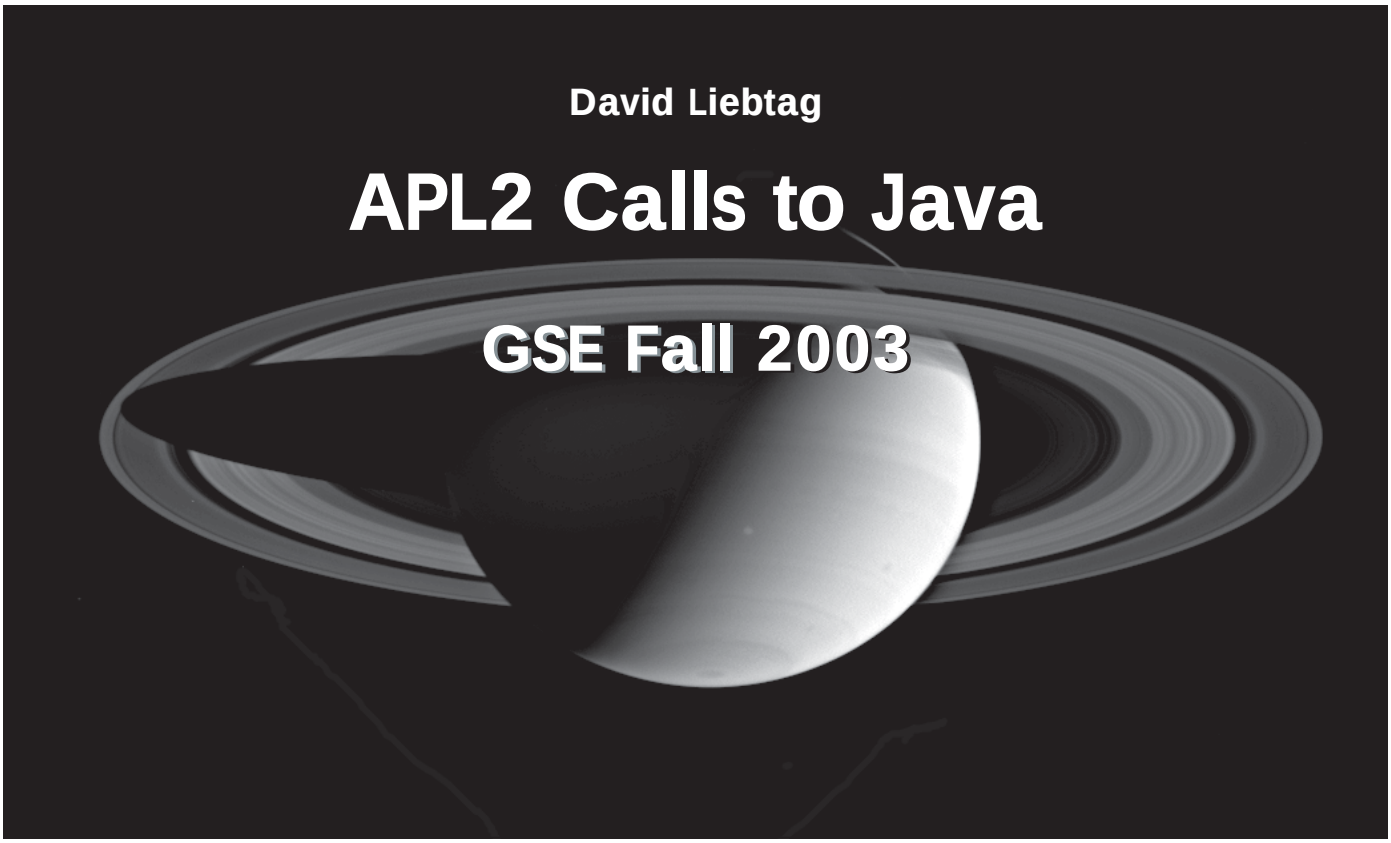
To receive an automatic e-mail when this page is updated, please enter your e-mail address: Submit

Webmaster: Bob Hoekstra
Vector Administration: Gill Smith at apl385@compuserve.com

David Liebtag

APL2 Calls to Java

GSE Fall 2003



The IBM APL group is working on an exciting new facility: Calls to Java. Through this facility, you will be able to call Java programs and have those programs call back to APL2.

The heart of the calls to Java facility is a new associated processor 14. A processor 14 association looks like this:

```
('Class' 'Signature') 14 ⎕NA 'Surrogate Member'
```

Where:

- ‚Class' is the name of a Java class or an instance returned by a constructor
- ‚Signature' is a description of a Java field's datatype or a Java method's arguments and results.
- Member is the name of the Java field or method
- Surrogate is the name to associate with the member. (A surrogate is required if the member name is not a valid APL2 object name. For example, constructors are always named <init> and so always require a surrogate.)

Note:

Note: For those readers who do not know Java, please see the Java Language Overview at the end of this article.

Here is an example of using static members of a Java class.

```
   ('Sample' 'I') 14 ⎕NA 'StaticField'
1
   StaticField ← 24
   ('Sample' '(I)I') 14 ⎕NA 'StaticMethod'
1
   Added ← StaticMethod 47
```

Here is an example of using an instance of a Java class

```
   ('Sample' '(I)V') 14 ⎕NA 'Constructor <init>'
1
   Object ← Constructor 999
   (Object '(I)I') 14 ⎕NA 'InstanceMethod'
1
   Result ← InstanceMethod 456
```

The APL2 Calls to Java interface includes several Java classes. The first, named Apl2java, contains a method that is used to delete objects created by APL2:

```
'com/ibm/apl2/Apl2java' '(I)V') 14 ⎕NA 'Delete'
Delete Object
```

Other classes are supplied to enable Java programs to call back to APL2. For example, the Apl2interp class includes a static field named Apl2 that is used to call back to the APL2 interpreter. And the Apl2object class is used to create and manipulate APL2 workspace objects. For example:

```
Apl2interp Apl2 = Apl2interp.Apl2 ;
Apl2object Vector = new Apl2object(Apl2,"IBM APL2") ;
```

The Apl2interp class also contains methods for executing APL2 expressions. If the APL2 workspace contained a function named SUM which performed a plus reduction, you could do this:

```
Apl2object Sum = Apl2.Execute("SUM",Vector) ;
```

And finally, the APl2interp class includes constructors which can be used to start new APL2 interpreters. For example, if you had packaged the GRAPHPAK workspace, you could do this:

```
int[] Array = {2,34,6,28,19} ;
Apl2interp Slave = new Apl2interp() ;
Apl2object PlotData = new
Apl2object(Slave,Array) ;
Slave.Execute("PLOT",PlotData);
Slave.Execute("VIEW") ;
```

Summary

The callback portion of Calls to Java facility is built on top of a new general Calls to APL2 API. This API enables APL2 to be called from other languages. The API is written to support calling APL2 directly from C.

The IBM APL Products and Services group intends to supply wrappers around the Calls to APL2 API to enable use of APL2 from other languages and technologies. Possible technologies include Java, ActiveX, COM, Corba, C#, and VisualBasic. We're starting with a Java wrapper because we believe Java will be of most use to our corporate customers. Using Java, we intend that our customers will be able to deploy their APL2 applications as Java beans and use mainstream tools such as WebSphere deliver their applications to end-users.

Java Language Overview

Java programs are composed of Java classes. Each class performs a service such as file IO, database access, or user interface services. Some classes are built-in to Java; some are built my application developers.

A Java class is similar to an APL workspace. Like a workspace, a class resides in a single file. Also like a workspace, a class can contain variables and functions. And also like a workspace, variables can be global or local. In addition, functions can also be local or global.

Here is a sample class:

```
public class Sample
{
   public static int StaticField = 0 ;
   public static int StaticMethod(int Argument) {
      return Argument + StaticField ;
   }
   public int InstanceField ;
   public Sample(int InitialValue) {
      this.InstanceField = InitialValue ;
     return ;
   }
   public int InstanceMethod(int Argument) {
      return Argument + InstanceField ;
   }
}
```

The Java designers decided to use different words for things than we use in the APL community. In Java,

- Public means global
- Private means local
- Field means variable
- Method means function
- Member means either a field or a method.

The first line of the class definition supplies the name of the class, in this case Sample. Everything within the curly braces constitutes the class's definition.

The keyword public indicates that a field or method is accessible from outside the class.

The first two members are declared static. This means that they exist as soon as the class is loaded and can always be used. They correspond to global variables and functions in an APL workspace.

The next member, InstanceField, is not declared static; this means that it is an „instance" member and can only be used in the contact of an object. To create an object, you must call a special method called a constructor. Constructors have the same name as the class; the fourth item is a constructor. When you call the Sample constructor, an instance of the Sample class is created. Once you have an instance of the class, you can use the instance, also called an object, to access instance fields and methods.

To illustrate, here is some Java code that uses the Sample class:

```
//* Set the static field
Sample.StaticField = 24 ;
//* Use the static method
int Added = Sample.StaticMethod(47) ;
//* Create an instance of the Sample class, a Sample object
Sample Object = new Sample(999) ;
//* Call the instance method
int Sum = Object.InstanceMethod(456) ;
```


Contact:

David Liebtag, APL Products and Services, IBM Silicon Valley Laboratory, San Jose, California, Email: liebtag@us.ibm.com, Home page: http://www.ibm.com/software/awdtools/ aplAddress: APL Products and Services, RENA/F415 555 Bailey Avenue, San Jose, California, USA 95141.
Work tel: 408-463-APL2 Work fax: 408-463-4488

Information:

Notice for static members, you use the name of the class followed by a period and the member name. For instance members, you use the name of a field containing a reference to the instance followed by a period and the member name.

Mainframe APL2
Workstation APL2
Marketing and Support
What's Next?

David Liebtag

APL2: What's New?

Mainframe APL2

- Calling COBOL with Environment Routines
  UQ72996 (TSO), UQ73001 (CMS)
  Technote 1083379
- AP 119 Support for Datagram Sockets
  UQ73321 (TSO), UQ73322 (CMS)
- Monadic ATR and RTA
  UQ74032 (TSO), UQ74033 (CMS)
- NETTOOLS Updates for Workstation Service Level 2
  UQ76096 (TSO), UQ76097 (CMS

Workstation APL2

Version 2 Service Level 1
October 2002

All Platforms

- ⎕PW session variable
- Maximum rank increase to 64
- External function versions of DISPLAY and TIME
- AP 119 support for datagram sockets
- AP 127 and AP 227 -isol invocation option

Windows

- Object Editor
  - Support for numeric and mixed matrices
  - Check for localized labels
  - Color coded localized names
  - Marked Lines Only checkbox in Find dialog
  - Sort Header command on Edit menu
- AP 145
  - Creating dialogs dynamically
  - Resizable dialogs
  - Setting print paper size, orientation, margins, and position
  - Grid control support for matrices containing numbers
- AP 207
  - MENU command supports setting print orientation

Version 1 End of Service
March 2003

Version 2 Service Level 2
April 2003

All Platforms

- 4-byte characters in function and operator definitions
  - Comments and character constants
  - Supported by ⎕CR, ⎕FX, 2 ⎕TF,)IN,)OUT
- Monadic ATR and RTA
  - CDR format used
  - ATR uses format of local machine
  - RTA supports mainframe and workstation CDR format
- New AP 227 commands
  - ODBC retrieves information about available datasources or drivers
  - ODBCOPEN sets up an SQL table containing information about tables or supported datatypes of the current database

Windows

- Session Manager/Object Editor
  - Unicode Clipboard system option
- AP 145
  - APL2 Programming: GUI Application Development
  - BORDER style for Frames and Rectangles
  - PATH, PATH SELECTION, and PATH STYLE properties
  - Dialog scroll bar events
  - STATE LIST VISIBLE property for combo boxes
  - PRINTERS, PRINTER and FILENAME properties for print objects
  - CLIENT ORIGIN property for group boxes
  - Event handlers can contain Unicode characters
  - GUITOOLS additions
    - o ALIGN, RESIZE, SPACE
    - o FILEDLG, FONTDLG
    - o MSGBOX
    - o POPUPMENU
    - o UNIEDIT
  - DEMO145 additions
    - o DEMO_EXPLORE
    - o DEMO_GDI
    - o DEMO_MODE
    - o DEMO_PATH
    - o DEMO_PROPERTIES
- AP 207
  - MENU command supports query of available printers, query or set of current printer

Marketing and Support

Newsletter 8

- April 2003

IBM Web site reorganization

- www.ibm.com/software/awdtools/apl
  - Downloads for Demo Version
  - Library for on-line APL2 manuals
  - News for latest announcements
  - Support for FAQ, Technotes, Service Updates

IBM APL2 Newsgroup

- news.software.ibm.com/ibm. software. apl

What's Next?

- Mainframe Linux and/or Posix ?
- Call APL2 from Websphere?
- ActiveX ?
- Object Editor / AP 145 for Unix?
- Mainframe Deviation Removal?

Contact:

David Liebtag, APL Products and Services, IBM Silicon Valley Laboratory, San Jose, California.
Email: liebtag@us.ibm.com

Why?

- To Save Space
- To Avoid Name Pollution
- To Facilitate Sharing
  - o ...with non-APL programs
  - o ...with other vendor's APL programs
  - o ... with other APL2 programs

What?

- Data Only
- APL Arrays
- APL Programs

Where?

- Database
- File
- Plain Text
- Formatted Records
- APL2 File
- Namespace

How?

Data Only

- Share with non-APL programs
- Data Types are limited
- Database:
- Store directly with AP 127, AP 227
- File:
- Use dyadic ATR to create formatted numbers, records

APL Arrays

- Share with APL programs
- Data Types are not limited
- File:
- Save objects in namespace
- Store directly in APL2 file
  - o ...by index (AP 121, AP 210, P12)
  - o ...by name (AP 211)
- Use dyadic RTA to create APL2 arrays from formatted records
- Data is not self-describing (*see example*)

Example:

```
      NUM←5.6 3.14 99 75
      CHAR←'SUPERCALIFRAGILISTICEXPIALIDOCIOUS'
      3 11 ⎕NA⊃'ATR' 'RTA' 'PFA'
1 1 1
      PFA CHAR
(34 C1 1 34)
      PFA NUM
(4 E8 1 4)
      CHAR ≡'C1 1 *' ATR CHAR
1
      NUM≡'E8 1 *' ATR NUM
0
      NUMREC←'E8 1 *' ATR NUM
      NUM≡'E8 1 *' RTA NUMREC
1
      NUM≡'I4 1 *' RTA NUMREC
0
```

- Convert to a self-describing format, store as text
  - o Transfer form (⎕TF)
  - o SCAR (ATS, STA)
  - o CDR (monadic ATR, RTA)
- Database:
- Convert to a self-describing format, store as text (*see example next page*)

http://www.apl-germany.de/index.htm

APL Germany e.V.

- Homepage
- Wir über uns
- Was ist APL?
- Wer bietet APL an?
- Veranstaltungen
- Literaturhinweise
- APL-Journal
- APL-Nachrichten
- APL-Links

NEW Frühjahrs-Tagung

Der *APL Germany e.V.* ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache *APL*, sowie darüberhinausgehend die Verbreitung des Verständnisses der Mensch - Maschine - Kommunikation zu fördern.

Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet *APL Germany e.V.* den adäquaten organisatorischen Rahmen.

Zu diesem Zweck veranstaltet der *APL Germany e.V.* beispielsweise Vorträge, Diskussionen und APL-Kurse, stellt Kontakte zu Computerfirmen her, die APL anbieten, und gibt das Mitteilungsblatt "APL-Journal" heraus.

Um die Kommunikation der Mitglieder untereinander zu erleichtern, hat APL Germany einen Nachrichten-Server eingerichtet. Bei diesem Server kann sich jeder, der an APL interessiert ist einschreiben. Eine Mitgliedschaft bei *APL Germany e.V.* ist nicht erforderlich (aber natürlich gerne gesehen).

Kontaktadresse:
APL Germany e.V.
Rheinstr. 23
D-69190 Walldorf
info@apl-germany.de

Sie haben eine Ausgabe des
APL-JOURNALs
verpaßt?

Das APL-Journal finden Sie zum Download als PDF-Datei auch auf der Webseite des RHOMBOS-VERLAGES unter:

http://www.rhombos.de/shop/a/show/article/?99

www.rhombos.de

Examples:

```
⍝ Namespace
      NEST←CHAR NUM
      )WSID OBJDEMO
WAS CLEAR WS
      )SAVE
2003-05-11 07.49.02 (GMT-4) OBJDEMO
      )CLEAR
CLEAR WS
      3 11 'NA 'CNS'
1
      CNS 'OBJDEMO'
E:\wintest\OBJDEMO.ans
  'OBJDEMO' 11 ⎕NA⊃'CHAR' 'NUM' 'NEST'
1 1 1
      CHAR
SUPERCALIFRAGILISTICEXPIALIDOCIOUS
      NUM
5.6 3.14 99 75
      NEST
 SUPERCALIFRAGILISTICEXPIALIDOCIOUS  5.6 3.14 99 75

⍝ APL file, by index
      ('ACW' 'TESTP12.GSE' '') 12 ⎕NA 'GSE'
1
      ⍴GSE
0
      GSE←GSE,CHAR NUM NEST
      ⍴GSE
3
      ↑GSE
SUPERCALIFRAGILISTICEXPIALIDOCIOUS
      GSE[2]
 5.6 3.14 99 75
      3⊃GSE
 SUPERCALIFRAGILISTICEXPIALIDOCIOUS  5.6 3.14 99 75

⍝ APL file, by name
      )COPY 1 UTILITY SVOFFER
SAVED 2003-05-02 14.46.52 (GMT-4)
      211 SVOFFER 'A'
2
      A←'CREATE' 'TEST211.GSE'
      A
0
      A←'USE' 'TEST211.GSE'
      A
0 1024
      A←'SET' 'CHAR' CHAR
      A
0
      A←'SET' 'NUM' NUM
      A
0
      A←'SET' 'NEST' NEST
      A
0
      A←'GET' 'NEST'
      A
 0   SUPERCALIFRAGILISTICEXPIALIDOCIOUS  5.6 3.14 99 75
⍝ Transfer form
      )LOAD OBJDEMO
SAVED 2003-05-11 07.49.02 (GMT-4)
       ⊃2 ⎕TF¨'CHAR' 'NUM' 'NEST'
CHAR←'SUPERCALIFRAGILISTICEXPIALIDOCIOUS'
NUM←5.6 3.14 99 75
NEST←('SUPERCALIFRAGILISTICEXPIALIDOCIOUS')(5.6 3.14 99 75)
⍝ SCAR
      3 11 ⎕NA ⊃'ATS' 'STA'
1 1
      ⍴¨ATS¨CHAR NUM NEST
 56  52  120
      CHAR NUM NEST≡¨STA¨ATS¨CHAR NUM NEST
1 1 1
⍝ CDR
      ⍴¨ATR¨CHAR NUM NEST
 64  64  160
      CHAR NUM NEST≡¨RTA¨ATR¨CHAR NUM NEST
1 1 1
```

APL Programs

- Convert to array to store (⎕TF, ⎕CR)
- Need to establish convention for distinguishing arrays and programs

Example:

```
      ∇Z←FUN A
[1]   Z←A+42
[2]   ∇
      2 ⎕TF 'FUN'
⎕FX 'Z←FUN A' 'Z←A+42'
      ⎕CR 'FUN'
Z←FUN A
Z←A+42
```

When?

- Data
- Choice limited by parameters of database, non-APL program
- APL Arrays
- Namespaces
  - o No conversions required (data or program-to-array)
  - o Not portable across APL2 platforms
  - o Not portable to other vendors
- APL2 Files
  - o No data conversions required
  - o Indexed files not portable mainframe-to-workstation
  - o Not portable to other vendors
- Self-Describing Formats
  - o Can be stored in database, text file (easy to share)
  - o Portable across APL2 platforms
  - o SCAR, Transfer form portable to other vendors
  - o Varying amounts of conversion required
  - o SCAR has data conversion penalty when used on mainframe (EBCDIC/ASCII and X370/IEEE)
  - o Transfer form has numeric-to-text conversion penalty (performance and precision loss)
- APL Programs
- Same issues as APL Arrays

Who?

- **Everybody!**

Contact:

Nancy Wheeler, APL Products and Services, IBM Silicon Valley Laboratory, eMail: APL2@vnet.ibm.com

Simon Garland

Matrikx reloaded

All work and no k makes jack a dull boy

Although the array language K is immediately accessible to programmers familiar with APL or J comparatively few have spent any time looking into it. For anyone who wants to add a new language to their toolkit this short article describes how to download and install a trial version - and points to further information for those who would like to learn more.

Installation

Download and run a copy of kdbsetup.exe from the download page [2] of the Kx website [1], this 200K file will install the array language K and the database Kdb. It will install on any dialect of Windows, although NT or XP are more comfortable to use as the 95/98/ME command prompt is so clunky.

The trial version has restrictions on the amount of data it will work with, and the number of messages that can be sent between K sessions - but otherwise is a full version.

Try K:

Open a command window, and type k and you'll be in immediate execution mode in a K session, try a few familiar things:

```
C:\k>k

K 3.0 2003-09-05 Copyright (C) 1993-2003 Kx Systems
WIN32 1CPU 1023MB f 0 EVAL

   2+2
4
   2+2 3 4
4 5 6
   +/2 3 4
9
```

To get out just type \\ in immediate execution mode.

K includes a built in reference, access it by typing a single \ followed by a return in immediate execution, that will give you a list of pages available. Enter the particular one you want to see (\+ to list the verbs for example).

If you type something complicated and get suspended you can clear the execution stack with a single \ at the suspension prompt. Here a nonsense function foo is defined to add its argument to a string and cause an error:

```
     foo:{"string"+x}
     foo 17
type error
{"string"+x}
         ^
>  \
```

If you just want to disable suspensions while looking around enter the command \e 0 on a new line - the error will still be displayed but the stack will be cut back:

```
 \e 0
 foo 17
type error
```

Try Kdb:

Get out of your previous K session if you still have it open by entering \\ and then start a Kdb session by entering:

```
k db
```

This time you'll be in a kdb session as shown by the ksql> prompt:

```
C:\k>k db
K 3.0 2003-09-05 Copyright (C) 1993-2003 Kx Systems
WIN32 1CPU 1023MB f 0 EVAL
KDB 3.0 2003-09-22
ksql>
```

As Kdb is a database we'll load in a sample database to make Things more interesting, at the ksql> prompt type in:

```
ksql>load 'trade.t'
ksql>
```

Note the single quotes, strings are delimited by double quotes in K, single quotes in Kdb - that makes it easier to embed KSQL strings in K code.

Now try some simple KSQL:

```
ksql>tables
`stock `trade
ksql>vars stock
`stock `industry
ksql>count stock
6
ksql>select from stock where industry='auto'
  .((`stock
     `xx `yyyy
     .,(`K;1;))
    (`industry
     `auto `auto
     ))
  ksql>count trade
  100000
  ksql>vars trade
  `stock `price `amount `date
ksql>select avg price by stock from trade
.((`stock
   `aaa `bbb `ccc `dd `xx `yyyy
   .((`T
      ,`stock
      )
     (`K;1;)))
  (`price
   74.93937 74.93437 74.89528 74.95792 74.73562 74.71044
   ))
ksql>
```

Notice that the table columns are just vectors: At last - a database that you can work with not against/despite!

You can open a simple viewer on the database by typing in:

```
ksql>show ''
```

or you can view the results of a query by prefixing a show to the constraint:

```
ksql>show select avg price by stock from trade
```

You can escape from Kdb to K by entering a single \ at the ksql> prompt

```
ksql>\
  2+2 3 4
4 5 6
ring .d.l`
ksql>\
  count stock
value error
count
^
parse error
  .d.l`
ksql>count stock
6
ksql>
```

Exit from a Kdb session with \\ as you would from a K session.

Summaries of KSQL, and the Kdb SQL92 implementation can be found in the two files s.txt (SQL92) and t.txt (KSQL) in the directory c:\k\db created when you installed kdb earlier.

Documentation:

If the 5-10 minutes it took to try this out has caught your interest the best way to learn the basics of K and Kdb is to read through the Kdb manual 'Getting started with Kdb' available from the Documentation page [3] on the Kx website. Although its focus is on Kdb it takes you through the basics of K as well.

Apart from the other manuals on the Documentation page [3] there is a K reference card available from Mike Powell's directory on the User Contributions page [4].

When learning a new language its often helpful to have other peoples code to read, there is a large library of K and Kdb code on the Kx site in Arthur Whitney's directories [7].

More general articles about K and Kdb are indexed on Eberhard Lutz's Kaylang page [8]

More Information and Help:

If you have questions about K or Kdb the Kxforums [5] are open to anyone, or the K and Kdb listboxes can be searched from the archive page [6].

Links:

1) http://kx.com - home of k and kdb
2) http://www.kx.com/download/download.htm - download k and kdb
3) http://www.kx.com/download/documentation.htm - manuals
4) http://www.kx.com/download/contributors.htm - user contributions
5) http://kxforums.com/kxf/ - user forums
6) http://www.kx.com/listbox/index.html - k and kdb listbox archives
7) http://kx.com/a/ - sample code
8) http://www.e7l3.com/kaylang.html - extensive list of k articles

Contact:

Simon Garland

eMail: simon_garland@bluewin.ch

Herbert Voß

Advanced Encryption Standard

1. Einführung

Seit mehreren Jahren versuchte man offiziell einen geeigneten Nachfolger für den bisherigen DES[1], einen sogenannten AES (*A*dvanced *E*ncryption *S*tandard), zu entwickeln, denn die Sicherheit dieses Verschlüsselungsstandards ließ zu wünschen übrig. Auch das sogenannte Triple-DES stellte, sicherheitstechnisch betrachtet, letztlich keine befriedigende Lösung mehr dar. Unter der Federführung des amerikanischen NIST[2] wurde 1997 ein entsprechendes formales Verfahren eingeleitet:

> On January 2, 1997, NIST announced the initiation of the AES development effort and made a formal call for algorithms on September 12, 1997. The call stipulated that the AES would specify an unclassified, publicly disclosed encryption algorithm(s), available royalty-free, worldwide. In addition, the algorithm(s) must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits.[3]

Zu den wesentlichen Aspekten zählen die weltweite freie Verfügbarkeit des Algorithmus und die Verwendung eines symmetrischen Verfahrens. Neue Forderungen zielten auf die Flexibilität hinsichtlich der möglichen Unterstützung verschiedener Blocklängen und verschiedener Schlüssellängen, womit das bekannte IDEA kein Kandidat sein konnte, denn es hat seine Vorteile auf anderem Gebiet.

- Schlüssellänge *mindestens* mit den Standardwerten 128 Bit, 192 Bit und 256 Bit
- Blocklänge *mindestens* 128 Bit

„Mindestens“ ist hier nicht im mathematischen Sinne als untere Grenze zu verstehen, sondern als Mindestanforderung im Aufzählungssinne; wenn es mehr als die hier geforderten Möglichkeiten gibt, desto besser.

Grundsätzlich führte die NIST das gesamte Verfahren öffentlich durch, so dass der den jeweiligen Verfahren zugrunde liegende Algorithmus sehr umfassend begutachtet werden konnte. Die öffentliche Analyse der einzelnen Verfahren war im Mai 2000 abgeschlossen und es begann die letzte Runde der Kandidatenauswahl, die Ende 2000 mit der Ernennung von Rijndael zum DES-Nachfolger abgeschlossen wurde. Dies ist erstaunlich, da bei den Amerikanern gerade im Bereich der nationalen Sicherheit sehr restriktive Vorgehensweisen die Regel waren und die Erwartung, dass ein europäisches Verfahren den Vorzug bekommen würde, war sehr gering.

Mars	IBM, Armonk, New York
RC6	RSA Laboratories, Bedford, Massachusetts
Rijndael	Joan Daemen, Vincent Rijmen (Belgien)
Serpent	Ross Anderson (Großbritannien), Eli Biham (Israel), Lars Knudsen (Norwegen)
Twofish	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson (USA)

Tabelle 1: Die „Bewerber“ um den Nachfolger von DES

Von den zahlreichen Bewerbern der ersten Runde, darunter auch die Deutsche Telekom mit Magenta, blieben fünf übrig, aus denen der Sieger gekührt werden sollte: Mars[4], RC6[5], Rijndael[6], Serpent[7] und Twofish[8]. Wer jeweils an den einzelnen Verfahren beteiligt war, zeigt Tabelle 1. Daß die einzelnen Bewerber sich bemühten, das jeweils andere Verfahren als unsicher zu belegen, versteht sich von selbst. Einen Vergleich hinsichtlich der Geschwindigkeit unter Berücksichtigung der Programmiersprache Java findet man bei http://jcewww.iaik.at/aes/ und bei Voß [3].

1 *D*ata *E*ncryption *S*tandard
2 *N*ational *I*nstitut of *S*tandards and *T*echnology -http://csrc.nist.gov/encryption/aes/
3 http://www.nist.gov/aes/
4 http://www.research.ibm.com/security/mars.html
5 http://www.rsasecurity.com/rsalabs/aes/
6 http://www.esat.kuleuven.ac.be/~rijmen/rijndael/
7 http://www.cl.cam.ac.uk/~rja14/serpent.html
8 http://www.counterpane.com/twofish.html

Einen allgemeinen Vergleich der in der ersten Runde beteiligten 15 Verfahren findet man bei http://www.counterpane.com/aes-performance.html.

ⓔ	Exclusiv-Oder-Verknüpfung
⊕	Addition
⊖	Subtraktion
⊙	Multiplikation
$K_0, K_1, ..., K_n$	Teilschlüssel

Tabelle 2: Erklärung der verwendeten Symbole. (Wenn nicht anders angegeben, erfolgen die mathematischen Operationen in der Regel modulo 32)

2. Funktionsweise

Entwickelt von Joan Daemen[9] und Vincent Rijmen[10] soll Rijndael folgende Kriterien erfüllen:

> The three criteria taken into account in the design of Rijndael are the following:
> - Resistance against all known attacks;
> - Speed and code compactness on a wide range of platforms;
> - Design simplicity.[2]

Die erste und letzte Bedingung sind letztlich evident, wenn Rijndael auch nur die geringste Chance haben will, als echter Kandidat für den DES-Nachfolger zu gelten. Der letzte Punkt unterliegt keiner objektiven Betrachtung, denn es gibt keine Kriterien, die darüber entscheiden, ob ein Algorithmus einfach ist oder nicht.

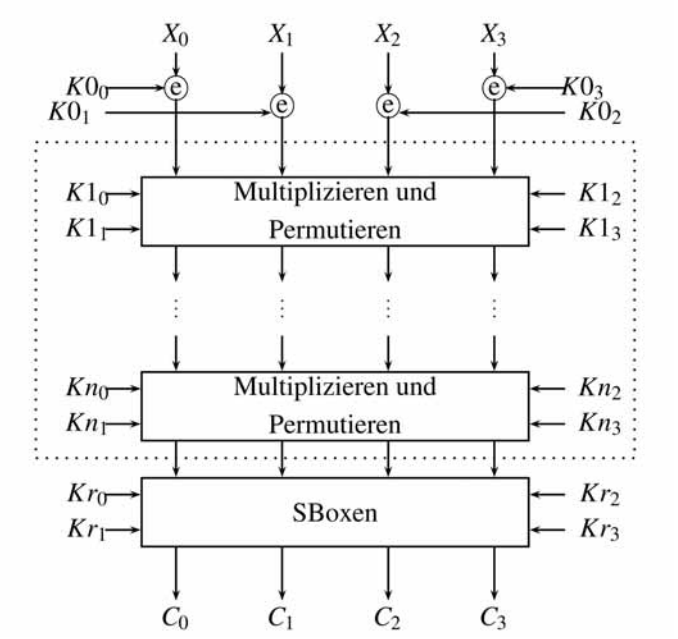


Abbildung 1: Das Prinzip von Rijndael der gestrichelte Bereich wird n = (r-1)-mal durchlaufen; r entspricht der Rundenzahl

Abgesehen davon muss ein einfacher Algorithmus nicht notwendigerweise auch der schnellste sein.

Abb. 1 zeigt das globale Prinzip von Rijndael, welches in dieser Form einfacher erscheint als die anderen Verfahren. Die Komplexität von Rijndael steckt jedoch in der Multiplikation, die im folgenden ausführlich behandelt wird. Die verwendeten Symbole sind in der Tabelle 2 zusammengefasst.

3. Multiplikationen

Im Gegensatz zu den anderen Verfahren verwendet Rijndael verstärkt die Multiplikation, die hier als Multiplikation von Polynomen in einem Galois-Feld[11] durchgeführt wird. Bezüglich der Addition braucht dies nicht weiter erläutert zu werden, da diese auch bei Polynomen einer Exclusiv-Oder-Operation entspricht. Im folgenden wird kurz der Vorgang der Multiplikation in einem Galois-Feld GF(2^8) erläutert.

Geht man von einem Byte aus, so entsprechen die einzelnen Bits den Koeffizienten eines Polynoms vom Grad 7:

$$9b_{16} = 10011011_2 \hat{=} x^7 + x^4 + x^3 + x + 1 \quad (1)$$

Multipliziert man jetzt z.B. die beiden hexadezimalen Zahlen 57_{16} und 83_{16}, dann entspricht dies einer Polynommultiplikation

$$57_{16} = 01010111_2 \hat{=} x^6 + x^4 + x^2 + x + 1 \quad (2)$$

$$83_{16} = 10000011_2 \hat{=} x^7 + x + 1 \quad (3)$$

$$57_{16} \cdot 83_{16} = (x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) \quad (4)$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \quad (5)$$

Da es sich um ein Galois-Feld GF 2^8 handelt, kann das Ergebnis nicht einfach mod256 bzw. mod100_{16} genommen werden, sondern muss aufgrund der Gruppeneigenschaften des Galois-Feldes modulo einem Polynom genommen werden, welches nur sich selbst und die Zahl 1 als Teiler enthält sowie vom Grad acht ist. x^8 wäre nicht ein solches Polynom, denn es enthält z.B. x, x^2,... als Teiler. Grundsätzlich bezeichnet man ein derartiges Polynom als Permutationspolynom[12], welches für Rijndael von der Ordnung 8 ist:

$$m(x) = x^8 + x^4 + x^3 + x + 1 \hat{=} 11b_{16} \quad (6)$$

Dieses Permutationspolynom enthält als Teiler also nur sich selbst und die Eins. Für das Ergebnis der Multiplikation aus der Gl. 5 folgt daher:

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \mod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1 = c1_{16} \quad (7)$$

9 Proton World International, Brüssel, Belgien
10 Katholieke Universiteit Leuven, Heverlee, Belgien
11 Galois, Evariste (25.10.1811-30.5.1832), französischer Mathematiker. Ein Galois-Feld *GF(p)* ist der Körper $(Z_p,+,\cdot)$ wobei Z_p die Restklassen mod p sind (0,1,2, ..., p-1) und folglich auch die Addition (+) und Multiplikation (·) mod p. Hierbei reicht für p die Bedingung, dass p eine Potenz einer Primzahl ist, z.B. 2^8.
12 Irreducible binary polynomial (nicht reduzierbares binäres Polynom).

Das Ergebnis ergibt sich durch Polynomdivision:

$$\left(x^{13}+x^{11}+x^{9}+x^{8}+x^{6}+x^{5}+x^{4}+x^{3}+1\right) : \left(x^{8}+x^{4}+x^{3}+x+1\right) = x^{5}+x^{3}+\text{Rest} \quad (8)$$

Rest ist vom Grad < 8

Multiplikation des ganzzahligen Teils der Polynomdivision mit dem Permutationspolynom und anschließender Subtraktion liefert dann den Rest, der in hexadezimaler Schreibweise $c1_{16}$ entspricht. Eine Multiplikation eines Polynoms mit x stellt einen Sonderfall dar. Allgemein erhält man als Ergebnis der Multiplikation $x \bullet a(x)$

$$a_7x^8+b_6x^7+b_5x^6+\ldots+b_1x^2+b_0x \quad (9)$$

- $a_7 = 1$: dieses Ergebnis muss, wie oben erwähnt, mod $m(x)$ genommen werden. Da aber der Grad dieses Polynoms gleich 8 ist, kann das Permutationspolynom nur einmal enthalten sein, so dass gilt:

$$x \cdot a(x) = m(x) + \text{Rest} \Rightarrow (x \cdot a(x)) \mod m(x) = \text{Rest} \quad (10)$$

  Hierbei ist der Rest ein Polynom von kleinerem Grad als 8! Grundsätzlich kann damit die Modulo-Operation auf eine einfache Subtraktion, bzw. informationstechnisch gesehen auf eine Exclusiv-Oder-Operation zurückgeführt werden. Insgesamt entspricht die Multiplikation eines Polynoms mit x in einem Galois-Feld GF(2^8) einer Links-Schiebeoperation (entspricht der Multiplikation mit $02_{16} \hat{=}$ x) und anschließendem bitweisem Exclusiv-Oder- Vergleich mit $1b_{16}$.[13]
- $a_7 = 0$: damit ist der höchste auftretende Exponent kleiner als das Permutationspolynom, so dass nach dem Linksshift um eine Stelle (Multiplikation) die Modulo-Operation zu keiner Veränderung führt:

$$a_7 = 0 \Rightarrow (x \cdot a(x)) \mod m(x) = x \cdot a(x) \quad (11)$$

Grundsätzlich werden derartige Multiplikationen bei Rijndael als $a = xtime(b)$ bezeichnet, vgl. Fußnote 14:

$$57_{16} \cdot 02_{16} = xtime(57) = ae_{16} \quad (12)$$

Multiplikation mit höheren Potenzen im Galoisfeld GF(2^8) kann somit auf eine wiederholte Anwendung dieser „Grundmultiplikation" zurückgeführt werden, was an einem Beispiel gezeigt wird:

$$\left(x^6+x^4+x^2+x+1\right) \cdot \left(x^4+x+1\right) \hat{=} 57_{16} \cdot 13_{16} \quad (13)$$
$$57_{16} \cdot 01_{16} = 57_{16} \quad (14)$$
$$57_{16} \cdot 02_{16} = xtime(57_{16}) = ae_{16} \quad (15)$$
$$57_{16} \cdot 04_{16} = xtime(xtime(57)) = xtime(ae_{16}) = 47_{16} \quad (16)$$
$$57_{16} \cdot 08_{16} = xtime(47_{16}) = 8e_{16} \quad (17)$$
$$57_{16} \cdot 10_{16} = xtime(8e_{16}) = 07_{16} \quad (18)$$
$$\Rightarrow$$

$$57_{16} \cdot 13_{16} = 57_{16} \cdot (01_{16} \textcircled{e} 02_{16} \textcircled{e} 10_{16}) \quad (19)$$
$$= 57_{16} \textcircled{e} ae_{16} \textcircled{e} 07_{16} \quad (20)$$
$$= fe_{16} \quad (21)$$
$$\hat{=} x^7+x^6+x^5+x^4+x^3+x^2+x \quad (22)$$

Wichtig ist hierbei die Erkenntnis, dass damit auch die komplexe Multiplikation auf reine Basisoperationen von Mikroprozessoren zurückgeführt werden kann, was einen großen Einfluss auf die Geschwindigkeit des Algorithmus hat.

3.1. Polynome mit Koeffizienten im Galoisfeld GF(2^8)

Die Festlegung der Koeffizienten auf die Menge {0, 1} stellt eine starke Einschränkung dar, weshalb auch Polynome mit Koeffizienten aus der Menge {0, 1, 2, ..., 255} definiert werden. Der sogenannte n-dimensionale Koeffizientenvektor korrespondiert dann mit einem Polynom vom Grad $n - 1$ bzw. kleiner. Die Addition zweier derartiger Vektoren entspricht der einfachen Exclusiv-Oder- Verknüpfung entsprechender Koeffizienten.

Schwieriger gestaltet sich die Multiplikation. Gegeben seien zwei Polynome dritten Grades über dem Galois-Feld GF(2^8):

$$\begin{aligned} a(x) &= a_3x^3+a_2x^2+a_1x+a_0 \\ b(x) &= b_3x^3+b_2x^2+b_1x+b_0 \end{aligned} \quad (23)$$

Das Produkt $c(x) = a(x) \bullet b(x)$ ergibt sich nach einfachen Regeln zu

$$\begin{aligned} c(x) &= c_6x^6+c_5x^5+\ldots+c_1x+c_0 \\ c_0 &= a_0 \cdot b_0 \\ c_1 &= a_1 \cdot b_0 \textcircled{e} a_0 \cdot b_1 \\ c_2 &= a_2 \cdot b_0 \textcircled{e} a_1 \cdot b_1 \textcircled{e} a_0 \cdot b_2 \\ c_3 &= a_3 \cdot b_0 \textcircled{e} a_2 \cdot b_1 \textcircled{e} a_1 \cdot b_2 \textcircled{e} a_0 b_3 \\ c_4 &= a_3 \cdot b_1 \textcircled{e} a_2 \cdot b_2 \textcircled{e} a_1 \cdot b_3 \\ c_5 &= a_3 \cdot b_2 \textcircled{e} a_2 \cdot b_3 \\ c_6 &= a_3 \cdot b_3 \end{aligned} \quad (24)$$

Es erscheint einsichtig, dass der Koeffizientenvektor $\vec{c}$ nicht mehr von der Länge vier ist, so dass hier eine Reduktion notwendig ist, damit der Koeffizientenvektor wieder einem Polynom vom Grad drei entspricht. Rijndael verwendet dazu das Polynom M(x) = $x^4 + 1$. Es gilt daher

$$x^k \mod (x^4+1) = x^{k \mod 4} \quad (25)$$

Angewendet auf alle Potenzen von x größer als 3 in Gl. 24 ergeben sich die folgenden Änderungen:

$$\begin{aligned} c_6x^6 \mod \left(x^4-1\right) &= c_6x^{6 \mod 4} = c_6x^2 \\ c_5x^5 \mod \left(x^4-1\right) &= c_5x^{5 \mod 4} = c_5x^1 \\ c_4x^4 \mod \left(x^4-1\right) &= c_4x^{4 \mod 4} = c_4x^0 \end{aligned} \quad (26)$$

13 $1b_{16}$, da vom Permutationspolynom bei der Exclusiv-Oder-Operation nur die unteren 8 Bits interessieren.

14 Da bei $57_{16} = 01010111_2$ das führende Bit $a_7 = 0$ ist, führt ein Linksshift um eine Bit-Stelle bereits zum Ergebnis der Multiplikation; denn Shift-Links(01010111) ergibt $10101110_2 = ae_{16}$.

Damit ergibt sich dann das Produkt $d(x) = a(x) \otimes b(x)$ zu

$$\begin{aligned} d(x) &= d_3x^3 + d_2x^2 + d_1x + d_0 \\ d_3 &= c_3 = a_3 \cdot b_0 \,ⓔ\, a_2 \cdot b_1 \,ⓔ\, a_1 \cdot b_2 \,ⓔ\, a_0 b_3 \\ d_2 &= c_2 \,ⓔ\, c_6 = a_2 \cdot b_0 \,ⓔ\, a_1 \cdot b_1 \,ⓔ\, a_0 \cdot b_2 \,ⓔ\, a_3 \cdot b_3 \\ d_1 &= c_1 \,ⓔ\, c_5 = a_1 \cdot b_0 \,ⓔ\, a_0 \cdot b_1 \,ⓔ\, a_3 \cdot b_2 \,ⓔ\, a_2 \cdot b_3 \\ d_0 &= c_0 \,ⓔ\, c_4 = a_0 \cdot b_0 \,ⓔ\, a_3 \cdot b_1 \,ⓔ\, a_2 \cdot b_2 \,ⓔ\, a_1 \cdot b_3 \end{aligned} \tag{27}$$

Es ist leicht zu erkennen, dass jeweils zuerst eine Multiplikation mit b_o, dann b_1, b_2 und abschließend mit b_3 erfolgt. Daher lässt sich sinnvollerweise die Matrizenschreibweise anwenden:

$$\begin{aligned} \underline{d} &= \underline{A} \cdot \underline{b} \\ \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} &= \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \end{aligned} \tag{28}$$

Wie schon im vorhergehenden Abschnitt, wird auch hier die Multiplikation mit x gezeigt. Ausgehend von einem Polynom des Grades drei ergibt die Multiplikation mit x:

$$b(x) \otimes x = b_3x^4 + b_2x^3 + b_1x^2 + b_0x \tag{29}$$

Dieses Ergebnis ist Modulo $(x^4 + l)$ zu nehmen, was den folgenden Ausdruck ergibt:

$$d(x) = (b(x) \otimes x) \quad \text{mod } (x^4 + 1) = b_2x^3 + b_1x^2 + b_0x + b_3 \tag{30}$$

Derselbe Ausdruck muss sich natürlich auch durch Anwendung der Matrizengleichung 28 ergeben. Durch einfache Überlegung ergibt sich

$$\underline{d} = \begin{pmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{pmatrix} \cdot \underline{b} \tag{31}$$

d.h. bis auf a_1 =01 sind alle anderen Koeffizienten logischerweise gleich Null, denn $a(x) = x$ war ja gegeben. Zusammenfassend lässt sich feststellen, dass eine Multiplikation mit x oder Potenzen davon einem zyklischen Shift[15] des Koeffizientenvektors entspricht.

15 Rotierender Linksshift, wenn man die horizontale Schreibweise nach Gl. 3.1 betrachtet oder rotierender Shift von oben nach unten, wenn die vertikale Vektordarstellung nach Gl. 3.1 berücksichtigt wird.

3.2 Die Teilschlüssel

Bei einer Rundenzahl von 10 und einer Blocklänge von 128 Bits werden 1408 Bits für die Rundenschlüssel benötigt.

$$\begin{aligned} n_{kBits} &= n_{Block} \cdot (n_{Runden} + 1) \\ &= 128 \cdot 11 = 1408 \end{aligned} \tag{32}$$

Der gewählte Grundschlüssel KG wird zuerst in ein Feld von 4 Byte-Worten expandiert (Integer für Java), wobei sich die Feldgröße aus der Rundenzahl und der Blockgröße ergibt. Den Zusammenhang zwischen Schlüsselgröße, Blockgröße und Rundenzahl zeigt Tab. 3, wobei für einen Vergleich mit anderen AES-Kandidaten im Prinzip nur die erste Spalte interessant ist, denn nur diese Blocklänge wird von AES gefordert. Der Größe des expandierten Schlüsselfeldes ist allgemein gegeben durch

$$W\left[\frac{n_B}{32} \cdot (n_R + 1)\right] \tag{33}$$

Bei 10 Runden und einer Blocklänge von 128 Bit ergibt sich somit ein Feld von 44 Elementen. Die ersten $n_K/32$ ergeben sich durch den Grundschlüssel, die anderen werden rekursiv aus diesem bestimmt.[2] Die Größe eines Rundenschlüssel hängt sinnvollerweise von der gewählten Blockgröße ab und beträgt bei 128-Bitblöcken 4 • 128 Bit.

	nk= 128	192	256
nk= 128	10	12	14
192	12	12	14
256	14	14	14

Tabelle 3: Rundenzahl n_R in Abhängigkeit von Blockgröße n_B (in Bits) und Schlüssellänge n_K (in Bits)

3.3 Die Verschlüsselung

Global besteht der Verschlüsselungsvorgang aus den folgenden drei Sequenzen

- initiale Rundenschlüsseladdition
- *(n_R — 1)* Runden
- eine Schlussrunde

3.4 Die Entschlüsselung

Dieser Vorgang ist formal völlig identisch zum Verschlüsseln. Es werden nur unterschiedliche Rundenschlüssel und unterschiedliche Matrizen benutzt.

3.5 Beispiel

Der im folgenden gezeigte Testlauf von Rijndael liefert das geforderte Ergebnis für den 16 Byte langen „leeren“ Text mit dem

gewählten Schlüssel $K = 80....._{16}$.

Das entsprechende Java-Programm *RijndaelDemo.java* steht unter http://www.perce.de/krypto/Rijndael/RijndaelDemo.java zum allgemeinen Download bereit.

```
voss@maria:~/Kryptologie/Rijndael > java
    RijndaelDemo
Klartext (16 Bytes):
00000000000000000000000000000000
Starte Verschlüsselung ...
Schlüssellänge (128,192,256): 128
Schlüssel: 80000000000000000000000000000000
1 128-Bit-Blöcke = 16 Bytes
0EDD33D3C621E546455BD8BA1418BEC8
Starte Entschlüsselung ...
00000000000000000000000000000000
```

Literatur

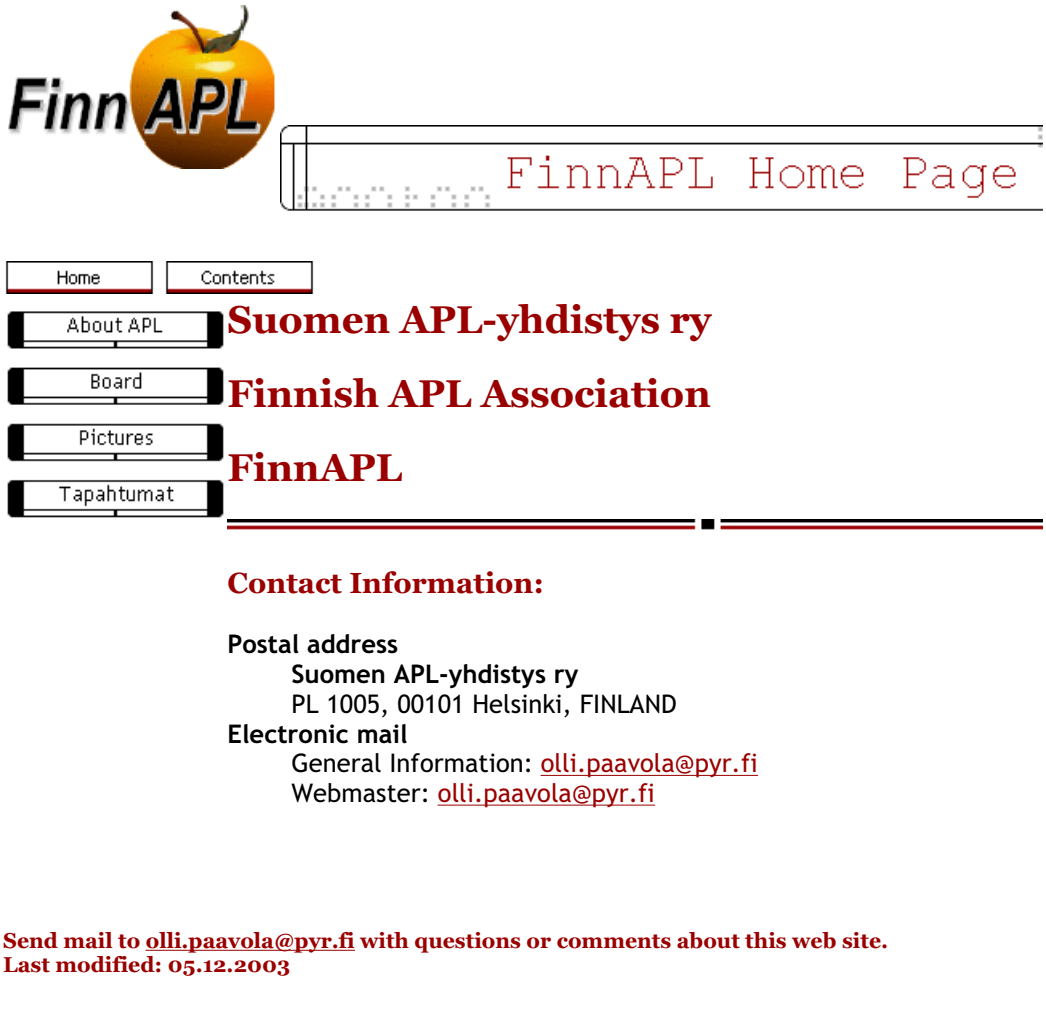
[1] C. Adams: The cast-128 encryption algorithm, may 1997.
[2] Joan Daemen and Vincent Rijmen: The rijndael block cypher, 1999.
[3] Herbert Voß: Praktische Kryptologie mit Java. BoD, jan 2001.

Kontakt

Dr. Herbert Voß, Berlin

eMail: Herbert.Voss@perce.de

FinnAPL Home Page http://www.pyr.fi/apl/



Allgemeine Informationen

(Stand Frühjahr 2003)

APL-Germany e.V. ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie die Verbreitung des Verständnisses der Mensch-Maschine Kommunikation zu fördern. Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen, sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet APL-Germany den adäquaten organisatorischen Rahmen.

Auf Antrag, über den der Vorstand entscheidet, kann jede natürliche oder juristische Person Mitglied werden. Organe des Vereins sind die mindestens einmal jährlich stattfindende Mitgliederversammlung sowie der jeweils auf zwei Jahre gewählte Vorstand.

Vorstand

Vorsitzender:

Dieter Lattermann
Rheinstraße 23
69190 Walldorf
Tel. (06227) 63469
eMail: 100332.1461@compuserve.com

Schriftführer

Conrad Hoesle-Kienzlen
CHK IT-Consulting
Augustenstr. 57
70178 Stuttgart
Tel. 0172 722 04 48
Fax 0711 674 38 53
eMail: chk@hoesle-kienzlen.de

2. Vorsitzender:

Martin Barghoorn
(Technische Universität Berlin)
Lückhoffstr. 8
14129 Berlin
Tel. (030) 314 24392
Fax: (030) 314 25901
eMail: martin@barghoorn.de
http://stat.cs.tu-berlin.de/~barg/

Schatzmeister

Jürgen Beckmann
Im Freudenheimer Grün 10
68259 Mannheim
eMail: JBecki@compuserve.com

Beitragssätze

Ordentliche Mitglieder:

Natürliche Personen	31,00 Euro	p.a.
Studenten / Schüler	10,00 Euro	p.a.

Außerordentliche Mitglieder:

Juristische / natürliche Personen	511,00 Euro	p.a.

Bankverbindung:

BVB Volksbank eG Bad Vilbel
BLZ 518 613 25, Konto-Nr. 523 2694

Hinweis:

Wir bitten alle Mitglieder, uns *Adressenänderungen* und *neue Bankverbindungen* immer sofort mitzuteilen.

Geben Sie bei *Überweisungen* den *Namen* und/oder die *Mitgliedsnummer* an.

Einzugsermächtigung

Mitglieds-Nr.: _____

Ich erkläre mich hiermit widerruflich damit einverstanden, daß APL Germany e.V.

den jeweils gültigen Jahres-Mitgliedsbeitrag von meinem unten angegebenen Konto abbucht.

Einen eventuell bestehenden Dauerauftrag habe ich bei meiner Bank gelöscht.

Bankbezeichnung: _____

BLZ: _____ Konto-Nr.: _____

Datum: _____ Unterschrift: _____