
fxdr - Eine Bibliothek zur Datenkonvertierung

1 Motivation

Häufig werden mithilfe von Fortran-Programmen mehr oder weniger große Datensätze erstellt und unformatiert in Dateien abgelegt. Soll anschließend eine Weiterverarbeitung dieser Daten auf einem anderen Rechnersystem vorgenommen werden, so bereitet das unterschiedliche interne Format meist Schwierigkeiten.

Der **xDR**-Standard (e**X**ternal **D**ata **R**epresentation) erlaubt es C-Programmierern, die gängigsten Datentypen in einem maschinenunabhängigen Format abzulegen. Die hier vorgestellte Programm-bibliothek **fxdr** stellt die Schnittstelle zwischen **Fortran** und C bereit und ermöglicht damit die Konvertierung der Daten von und nach **xDR** direkt aus einem Fortran-Programm heraus. Die Portierung von Daten im **xDR**-Format kann problemlos über binäres ftp geschehen. Damit können die oben angesprochenen Probleme mit relativ geringem Aufwand gelöst werden.

2 Unterstützte Rechner

Zur Zeit ist die Bibliothek **fxdr** auf folgenden Systemen verfügbar:

- CRAY T90
- CRAY J90
- CRAY T3E
- IBM RS/6000 R50 (zentraler AIX-Server)
- SUN (zam127 und zam255)
- Intel Paragon
- AIX Workstation-Gruppen
- DEC Workstation-Gruppen

Dies bedeutet, daß auf jedem dieser Rechner Daten im **xDR**-Format geschrieben und auch Dateien im **xDR**-Format gelesen werden können.

Inhalt	
1 Motivation	1
2 Unterstützte Rechner	1
3 Nutzung der Programm-bibliothek fxdr	2
3.1 Verfügbare Unterprogramme	2
3.2 Syntax der Unterprogrammaufrufe	3
3.3 Besonderheiten und Hinweise	4
3.4 Anbindung der Bibliothek an ein Fortran-Programm	6
3.5 Datenaustausch zwischen Fortran- und C-Programmen	6
4 Anwendungsbeispiel	7

Zusätzlich dazu wird der Quelltext auf dem anonymous-ftp Server des ZAM angeboten. Dort finden sich auch makefiles für folgende Systeme:

- SUN OS
- SUN Solaris
- AIX
- DEC Ultrix
- DEC OSF
- HP UX
- SGI Irix
- UNICOS

(Verzeichnis `/pub/unix/fxdr`, Datei `fxdr.tar.z`).

3 Nutzung der Programmbibliothek fxdr

3.1 Verfügbare Unterprogramme

Unterprogramme zur Dateibehandlung:

- fxdrini** initialisiert die C-Arbeitsumgebung
- fxdropn** öffnet die zu bearbeitende XDR-Datei
- fxdrcls** schließt die bearbeitete XDR-Datei

Unterprogramme zur Datenkonvertierung:

- fxdrint** Konvertierung von INTEGER-Werten (nur Standardwortlänge (4 bzw. 8 Byte (Cray)) unterstützt, dabei muß der Wert auf 32 Bit darstellbar sein). Werden z.B. INTEGER*2 Werte verwendet, so sollten diese zur Verarbeitung mit fxdr-Routinen auf Variablen mit Standardlängen umgespeichert werden.
- fxdrrl** Konvertierung von REAL-Werten bis maximal 4 Byte; zusätzlich kann dieses Unterprogramm verwendet werden, um Daten des Typs COMPLEX (4 Byte pro Komponente) zu bearbeiten.
- fxdrdbl** Konvertierung von REAL-Werten bis maximal 8 Byte; dieses Unterprogramm ist bei der Verarbeitung von Cray Single Precision zu verwenden, falls die 64 Bit Genauigkeit erhalten bleiben soll. Eine 128 Bit Genauigkeit (Cray Double Precision oder IBM Extended Precision) kann bei der Konvertierung nicht erhalten bleiben. Die Verwendung dieser Genauigkeiten führt nicht zu Genauigkeitsverlust, sondern zu **falschen Ergebnissen**.
Außerdem kann dieses Unterprogramm zur Verarbeitung von Daten des Typs COMPLEX (8 Byte pro Komponente) verwendet werden.
- fxdrlog** Konvertierung von LOGICAL-Werten (nur Standardwortlänge (4 bzw. 8 Byte (Cray)) unterstützt); bei der Bearbeitung von Werten des Datentyps LOGICAL wird **nicht** das Bitmuster erhalten, sondern lediglich der resultierende logische Wert (.TRUE. oder .FALSE.).
- fxdrchr** Konvertierung von CHARACTER-Werten gemäß der Fortran Deklaration; die maximale String-Länge beträgt 32760 Zeichen. Nur Zeichen aus dem 7-Bit ASCII-Zeichensatz werden unterstützt, hierzu zählen u.a. **nicht** die Umlaute.

Zunächst muß aus dem Fortran- **Hauptprogramm** eine C-Umgebung initialisiert werden (**fxdrini**). Anschließend werden die zu bearbeitenden Dateien geöffnet (**fxdropn**). Dabei bestimmt der Parameter **iomode** jeweils, ob es sich um eine bereits bestehende Datei ("Lesen"), oder um eine neu anzulegende ("Schreiben") handelt. Anschließend können die Daten mittels der aufgeführten Routinen zur Datenkonvertierung bearbeitet werden. Nach Beendigung müssen alle geöffneten XDR-Dateien **explizit** geschlossen werden (**fxdrcls**).

3.2 Syntax der Unterprogrammaufrufe

```
call fxdrini
```

```
call fxdropn (fnxdr,iomode,ihandle)
```

CHARACTER **fnxdr** - (*input*)

Name der zu bearbeitenden XDR-Datei; auf UNIX-Systemen können die speziellen Dateien *stdin* und *stdout* angegeben werden.

CHARACTER **iomode** - (*input*)

Schlüsselwort, das angibt, ob es sich um einen Lese- oder Schreibvorgang handelt

iomode = 'ENCODE' - Konvertierung ins XDR-Format (Schreiben)

iomode = 'DECODE' - Konvertierung vom XDR-Format (Lesen)

('ENCODE' bzw. 'DECODE' müssen in Großbuchstaben spezifiziert werden)

INTEGER **ihandle** - (*output*)

Unitnummer der XDR-Datei, **ihandle** identifiziert die aktuelle Datei, solange diese geöffnet ist. **ihandle** muß im Aufruf einer Variablen entsprechen, die Angabe einer Konstanten ist nicht erlaubt.

```
call fxdrcls (ihandle)
```

INTEGER **ihandle** - (*input*)

Unitnummer der XDR-Datei (Rückgabewert von **fxdropn**)

```
call fxdrint (ihandle,int_var,num)
```

INTEGER **ihandle** - (*input*)

Unitnummer der XDR-Datei (s. **fxdrcls**)

INTEGER **int_var** - (*input* bzw. *output*)

ein- oder mehrdimensionale Variable, deren Wert(e) auf die XDR-Datei geschrieben bzw. auf die von der XDR-Datei gelesene Daten zugewiesen werden

INTEGER **num** - (*input*)

Anzahlparameter, gibt an, wieviele Werte bei diesem Aufruf bearbeitet werden

```
call fxdrreal (ihandle,real_var,num)
```

INTEGER **ihandle** - (*input*)

Unitnummer der XDR-Datei (s. **fxdrcls**)

REAL **real_var** - (*input* bzw. *output*)

ein- oder mehrdimensionale Variable, deren Wert(e) auf die XDR-Datei geschrieben bzw. auf die von der XDR-Datei gelesene Daten zugewiesen werden

INTEGER **num** - (*input*)

Anzahlparameter, gibt an, wieviele Werte bei diesem Aufruf bearbeitet werden

```
call fxdrdbl (ihandle,dbl_var,num)
```

INTEGER **ihandle** - (*input*)

Unitnummer der XDR-Datei (s. **fxdrcls**)

DOUBLE PRECISION **dbl_var** - (*input* bzw. *output*)

ein- oder mehrdimensionale Variable, deren Wert(e) auf die XDR-Datei geschrieben bzw. auf die von der XDR-Datei gelesene Daten zugewiesen werden

INTEGER **num** - (*input*)

Anzahlparameter, gibt an, wieviele Werte bei diesem Aufruf bearbeitet werden

```
call fxdrlog (ihandle,log_var,num)
```

INTEGER **ihandle** - (*input*)

Unitnummer der XDR-Datei (s. **fxdrcls**)

LOGICAL **log_var** - (*input* bzw. *output*)

ein- oder mehrdimensionale Variable, deren Wert(e) auf die XDR-Datei geschrieben bzw. auf die von der XDR-Datei gelesene Daten zugewiesen werden

INTEGER **num** - (*input*)

Anzahlparameter, gibt an, wieviele Werte bei diesem Aufruf bearbeitet werden

```
call fxdrchr (ihandle,string)
```

INTEGER **ihandle** - (*input*)

Unitnummer der XDR-Datei (s. **fxdrcls**)

CHARACTER **string** - (*input* bzw. *output*)

eindimensionale Variable, deren Wert auf die XDR-Datei geschrieben bzw. auf die ein von der XDR-Datei gelesener Wert zugewiesen wird. Felder von Zeichenketten können von **fxdrchr** nicht bearbeitet werden; falls dies erforderlich ist, so muß das Unterprogramm elementweise aufgerufen werden.

3.3 Besonderheiten und Hinweise

3.3.1 Dateinamen

Dateinamen können absolut oder relativ zum aktuellen Verzeichnis angegeben werden. Es dürfen jedoch keine Shell-Variablen (wie z.B. **\$HOME**, **\$PERM** etc.) zur Spezifikation der Datei verwendet werden.

3.3.2 Öffnen und Schließen von XDR-Dateien

Das Öffnen und Schließen von XDR-Dateien muß immer **explizit** im Fortran-Programm geschehen. Da die Dateien von C-Programmen angelegt und geöffnet werden, beinhaltet das END-Statment im rufenden Fortran-Programm **kein** implizites Schließen. Wird eine XDR-Datei vor Verlassen des Fortran-Programms nicht geschlossen, so kann dies zu Systemfehlern führen.

Die Anzahl der von einem Programm gleichzeitig geöffneten XDR-Dateien ist theoretisch auf 140 begrenzt, wird jedoch praktisch meist durch das jeweilige Betriebssystem limitiert.

3.3.3 Verarbeitung von Werten des Datentyps COMPLEX

Obwohl für die Konvertierung von Werten des Datentyps COMPLEX keine eigenen Routinen zur Verfügung stehen, ist die Bearbeitung solcher Werte aufgrund ihrer Struktur recht einfach. Ein komplexer Wert wird aus Real- und Imaginärteil gebildet. Diese beiden Komponenten werden in Fortran wie zwei eigenständige Gleitkommazahlen behandelt und im Speicher hintereinander abgelegt. Daher können die Unterprogramme **fxdr1** und **fxdrdbl** (je nach Genauigkeit) zur Konvertierung komplexer Werte verwendet werden:

```
COMPLEX*8 C1(N)           ! jede Komponente 4 Byte
COMPLEX*16 C2(N)          ! jede Komponente 8 Byte
[ ... ]
CALL FXDRRL(IHANDLE,C1,2*N) ! konvertiert 2N Komponenten der Laenge 4
CALL FXDRDBL(IHANDLE,C2,2*N)! konvertiert 2N Komponenten der Laenge 8
[ ... ]
```

3.3.4 Anzahlparameter **num**

Bei der Verarbeitung logischer oder numerischer Werte muß darauf geachtet werden, daß der zu übergebende Anzahlparameter **num** einen nicht-negativen Wert hat. Ein negativer Wert kann, je nach System, zu unsinnigen Operationen oder gar zum Abbruch des Programms ("Segmentation violation" o. ä.) führen.

3.3.5 Genauigkeit numerischer Werte

Die Genauigkeit numerischer Werte wird bis zu einer Darstellung von 64 Bit beibehalten. Cray Double Precision, ebenso wie IBM Extended Precision, kann nicht erhalten werden und die Verwendung dieser Genauigkeiten führt nicht zu einem Genauigkeitsverlust, sondern zu **falschen** Ergebnissen. Auf den CRAY-Systemen sollte daher auf jeden Fall die Compiler-Option **-dp** angegeben werden, damit die Verwendung von doppelt genauer Arithmetik ausgeschaltet wird.

3.3.6 Die Unitnummer

Der Parameter **ihandle** (Integer), der beim Öffnen einer XDR-Datei von dem entsprechenden Unterprogramm (**fxdropn**) gesetzt wird, ordnet jeder geöffneten Datei für die Dauer deren Bearbeitung eine eindeutige ganzzahlige Kennung zu.

3.3.7 Behandlung von konstanten Werten

Beim Schreiben (**fxdropn** mit der Angabe 'ENCODE') können anstelle von Variablen auch Konstanten angegeben werden. Beim Lesen kann dies jedoch zu ungewolltem Überschreiben von Konstanten führen:

```
call fxdrini
* ---- OEFFNEN DER DATEI XDR.TEST ZUM SCHREIBEN
call fxdropn('XDR.test','ENCODE',ihandle)
* ---- SCHREIBEN DES INTEGER-KONSTANTEN 2
call fxdrint(ihandle,2,1)
* ---- SCHLIESSEN DER DATEI XDR.TEST
call fxdrcls(ihandle)
* ---- OEFFNEN DER DATEI XDR.TEST ZUM LESEN
call fxdropn('XDR.test','DECODE',ihandle)
* ---- ZUWEISEN DES GELESENEN INTEGER-WERTS AUF 3
call fxdrint(ihandle,3,1)
* ---- SCHLIESSEN DER DATEI XDR.TEST
call fxdrcls(ihandle)
* ---- AUSGABE DER SUMME 3+3
write(*,*) '3+3=',3+3
```

Inhalt von stdout:

```
3+3=          4
```

3.3.8 Transfer der XDR-Dateien

1. Auf allen Systemen möglich: Binäres ftp
2. Für Transfer zu/von einem CRAY-System möglich:
 - **dispose** - zum Dateitransfer von einem CRAY-Rechner
Es muß die Option **-fTR** (transparent, send as is) angegeben werden.
 - **fetch** - zum Dateitransfer zu einem CRAY-Rechner
Es muß die Option **-fTR** (transparent, send as is) angegeben werden.
 - **ftua** (file transfer user agent) - hat im Wesentlichen die gleiche Funktionalität wie ftp.
Zum Transfer von XDR-Dateien muß der binäre Übertragungsmodus gewählt werden.
Detaillierte Information hierzu in KFA-ZAM-BHB-0138 sowie im Cray Manual SR-2011 **UNICOS User Commands Reference Manual**).

3.4 Anbindung der Bibliothek an ein Fortran-Programm

Paragon	Bei der Compilation bzw. beim Laden muß der Pfad /usr/local/lib im Suchpfad eingetragen sein (bei Cross-Compilation auf der Intel-Sun muß /usr/local/paragon/lib-kfa eingetragen sein; beides ist voreingestellt). Über die -l Option zum f77-Befehl müssen außerdem die Bibliotheken fxdr und rpc spezifiziert werden.
SUN Solaris	Bei der Compilation bzw. beim Laden müssen über die -l Option zum f77-Befehl die Bibliotheken fxdr und ns1 spezifiziert werden.
Alle anderen zentralen Systeme	Für die übrigen unterstützten zentralen Systeme ist lediglich zu beachten, daß bei der Compilation bzw. beim Laden der Pfad /usr/local/lib als Suchpfad eingetragen ist (Default auf den meisten Systemen) und der Name der Bibliothek mittels der Option -lfxdr angegeben wird.
Nichtzentrale Systeme	Bei Installationen auf nichtzentralen Systemen muß bei der Compilation bzw. beim Laden der Pfad angegeben werden, unter dem die Bibliothek (libfxdr.a) zu finden ist (i.a. über die Option -L [Pfad]), sowie der Name der Bibliothek (i.a. über die Option -lfxdr). Fragen Sie hierzu Ihren Systemadministrator.

3.5 Datenaustausch zwischen Fortran- und C-Programmen

Die sich durch die fxdr-Unterprogramme bietende Schnittstelle zwischen Fortran- und C-Programmen kann natürlich - nach Modifikationen - auch für den Datenaustausch zwischen diesen beiden Programmiersprachen verwendet werden. Nähere Informationen finden sich auf UNIX-Systemen in den man-pages (**man xdr**). Bei Fragen oder Problemen wenden Sie sich bitte an die Programmberatung.

4 Anwendungsbeispiel

Auf der CRAY T90 wird mittels des Programms `test.f` ein Datensatz erstellt, der anschließend auf dem zentralen AIX-System weiterverarbeitet werden soll:

```
1      PROGRAM  TEST
2*-----
3      REAL RARR(10000)
4      DOUBLE PRECISION DARR(500,20)
5      COMPLEX CARR(10000)
6      INTEGER IARR(10000)
7      CHARACTER HEADER*40
8      LOGICAL NEG(10000)
9
10     INTEGER I,J,IND
11*-----
12* ---- BERECHNUNG DER DATEN:
13         DO 10, I=1,20
14             DO 10, J=1,500
15                 IND = (I-1)*500+J
16                 DARR(J,I) = DBLE(J+I-SIN(J*2.0))
17                 IARR(IND) = I+J
18                 RARR(10000-IND+1) = COS(I*2.0)
19                 CARR(IND) = CMPLX(2.*I,0.5*J)
20                 IF (RARR(10000-IND+1) .LT. 0) THEN
21                     NEG(I) = .TRUE.
22                 ELSE
23                     NEG(I) = .FALSE.
24                 ENDIF
25     10    CONTINUE
26
27         HEADER = 'Erste Berechnung der Testdaten:'
28* ---- OEFFNEN DER AUSGABE-DATEI
29         OPEN(7,FILE='test.daten',FORM='UNFORMATTED')
30
31* ---- AUSGABE AUF UNFORMATIERTE FORTRAN-DATEI:
32         WRITE(7) HEADER,RARR,DARR,CARR,IARR,NEG
33
34
35         END
```

Das Weiterverarbeiten der Datei `test.daten` auf dem AIX-System ist ohne weiteres nicht möglich.

Die folgenden Änderungen im Quelltext des Programms `test.f` (gekennzeichnet durch "-->") bewirken eine Konvertierung der Daten in das **XDR**-Format sowie ein Erstellen einer XDR-Datei anstelle der oben erhaltenen Datei `test.daten`:

```

1      PROGRAM TEST
2*-----
3      REAL RARR(10000)
4      DOUBLE PRECISION DARR(500,20)
5      COMPLEX CARR(10000)
6      INTEGER IARR(10000)
7      CHARACTER HEADER*40
8      LOGICAL NEG(10000)
9
10     INTEGER I,J,IND
11*-----
-->* ---- INITIALISIEREN DER C-UMGEBUNG
-->      CALL FXDRINI
-->*
12* ---- BERECHNUNG DER DATEN:
13     DO 10, I=1,20
14         DO 10, J=1,500
15             IND = (I-1)*500+J
16             DARR(J,I) = DBLE(J+I-SIN(J*2.0))
17             IARR(IND) = I+J
18             RARR(10000-IND+1) = COS(I*2.0)
19             CARR(IND) = CMPLX(2.*I,0.5*J)
20             IF (RARR(10000-IND+1) .LT. 0) THEN
21                 NEG(I) = .TRUE.
22             ELSE
23                 NEG(I) = .FALSE.
24             ENDIF
25 10    CONTINUE
26
27     HEADER = 'Erste Berechnung der Testdaten:'
28* ---- OEFFNEN DER AUSGABE-DATEI
-->      CALL FXDROPN('xdr.daten', 'ENCODE', IHANDLE)
29
30
-->* ---- KONVERTIEREN DER DATEN UND AUSGABE AUF XDR.DATEN
-->      CALL FXDRCHR(IHANDLE,HEADER)
-->      CALL FXDRDBL(IHANDLE,RARR,10000)
-->      CALL FXDRDBL(IHANDLE,DARR,10000)
-->      CALL FXDRDBL(IHANDLE,CARR,20000)
-->      CALL FXDRINT(IHANDLE,IARR,10000)
-->      CALL FXDRLOG(IHANDLE,NEG,10000)
31
-->* ---- SCHLIESSEN VON XDR.DATEN
-->      CALL FXDRCLS(IHANDLE)
32
33
34
35     END

```

Zu beachten ist hier, daß bei der Verwendung von Cray einfach genauen Real-Werten (Feld RARR) aufgrund der 64-Bit Genauigkeit die Routine **fxdrdbl** anstelle von **fxdr1** verwendet werden muß.

Außerdem muß, aufgrund der Verwendung von Cray Double Precision (Feld DARR), die Compiler-Option **-dp** unbedingt angegeben werden, um Fehler bei der Konvertierung zu vermeiden.

Die Compilation des Programms `test.f` könnte dann wie folgt aussehen:

```
cf77 -o toxdr test.f -dp -lfxdr
```

Man erhält das ausführbare Programm **toxdr**, das die XDR-Datei anlegt.

Anschließend kann die XDR-Datei `xdr.daten` via ftp zum AIX portiert werden:

```
> ftp aix.sp.kfa-juelich.de
> [Userid eingeben]
> [Password eingeben]
> [in das entsprechende Unterverzeichnis wechseln]
> binary (schaltet auf binaere Uebertragung um)
> put xdr.daten
> quit
```

Zum Lesen der Daten von der XDR-Datei können die gleichen Programmteile verwendet werden, die zum Schreiben der Daten dienten, es muß lediglich das Schlüsselwort 'DECODE' beim Öffnen der Datei angegeben werden (gekennzeichnet durch "!!>"). Anschließend können dann die Daten auf eine unformatierte Fortran-Datei ausgegeben werden (gekennzeichnet durch "OO>") und stehen damit im AIX in dieser Form zur Verfügung (Datei 'test.daten'):

```
1          PROGRAM TEST
2*-----
DD>        DOUBLE PRECISION RARR(10000)
4          DOUBLE PRECISION DARR(500,20)
DD>        COMPLEX*16 CARR(10000)
5          INTEGER IARR(10000)
6          CHARACTER HEADER*40
7          LOGICAL NEG(10000)
11*-----
28* ---- OEFFNEN DER AUSGABE-DATEI
!!>       CALL FXDROPN('xdr.daten', 'DECODE', IHANDLE)
30
-->* ---- KONVERTIEREN DER DATEN UND AUSGABE AUF XDR.DATEN
-->       CALL FXDRCHR(IHANDLE,HEADER)
-->       CALL FXDRDBL(IHANDLE,RARR,10000)
-->       CALL FXDRDBL(IHANDLE,DARR,10000)
-->       CALL FXDRDBL(IHANDLE,CARR,20000)
-->       CALL FXDRINT(IHANDLE,IARR,10000)
-->       CALL FXDRLOG(IHANDLE,NEG,10000)
33
-->* ---- SCHLIESSEN VON XDR.DATEN
-->       CALL FXDRCLS(IHANDLE)
OO>*-----
OO>* ---- OEFFNEN DER AUSGABEDATEI TEST.DATEN
OO>       OPEN(7,FILE='test.daten',FORM='UNFORMATTED')
OO>* ---- AUSGABE DER RUECKKONVERTIERTEN DATEN
OO>       WRITE(7) HEADER,RARR,DARR,CARR,IARR,NEG
OO>
35          END
```

Hier ist darauf zu achten, daß, da ja der XDR-Datensatz unter Verwendung von Cray Single Precision (8 Byte) erstellt wurde, die Deklaration der entsprechenden Variablen im AIX als DOUBLE PRECISION erfolgen muß (gekennzeichnet mit "DD>"). Dies gilt sowohl für die Real-Variablen RARR und DARR, als auch für das COMPLEX-Feld CARR. Die Genauigkeitsangabe bei CARR ("*16") gibt die Gesamtlänge der Darstellung (Real- plus Imaginärteil) an.

Soll auf dem Zielsystem (hier: AIX) mit 32-Bit Genauigkeit weitergerechnet werden, so muß eine Umspeicherung der rückkonvertierten Werte erfolgen.

Das Fortran-Programm (`test.f`) wird wieder mit den entsprechenden Optionen compiliert

```
f77 -o fromxdr test.f -lfxdr
```

und das entstandene ausführbare Programm **fromxdr** kann anschließend zur Rückkonvertierung des XDR-Datensatzes verwendet werden.