# API für die Metatypen

# 1 Allgemein

## 1.1 bisheriger Stand

Das bisherige Auftragskonzept konnte nur ein einem Netz mit gleichen Betriebssystemen zum Einsatz gebracht werden. Auf der einen Seite gibt es Unterschiede in der Byte-Reihenfolge von verschiedenen Prozessoren. Hier seihen nur die Stichwörter „big-endian“ und „little-endian“ genannt. Außerdem legen Compiler ihre Variablen auf unterschiedliche Adressgrenzen ab, das sogenannte Alignment.
Deshalb konnte z. B. das Auftragskonzept zwischen einer Sparc- und einer Linux-Maschine nicht angewendet werden.

## 1.2 Ziel

Das Auftragskonzept sollte dahingehend erweitert werden, daß die Software in heterogenen Netzen lauffähig ist und die übertragenen Daten zwischen verschiedenen Rechnern konsistent interpretierbar sind. D.h. die zu übertragenden Daten, die auf jedem Rechner in einer „Rechnerstruktur“ vorliegen, müssen in eine entsprechende „Netzwerkstruktur“ übertragen werden, die beide Partner verstehen.

# 2 Techniken

Um die „Rechnerstruktur“ in eine „Netzwerkstruktur“ abzubilden, müssen die einzelnen Komponenten dieser Struktur beschrieben werden.

## 2.1 Systemdefinierte Metatypen

Das System stellt folgene Basistypen zur Verfügung:

- MT_SHORT 2 Byte natürliche Zahl
- MT_LONG 4 Byte natürliche Zahl
- MT_FLOAT 4 Byte Fließkommazahl
- MT_DOUBLE 8 Byte Fließkommazahl mit doppelter Genauigkeit
- MT_CHAR 1 Byte Text
- MT_BYTE 1 Byte (keine Interpretation)
- MT_FILE Datei (Zeiger auf eine Filestruktur: FILE)
- MT_ENUM_ITEM Komponente eines Aufzählltyps
- MT_BITVECTOR_ITEM Komponente eines Bitvektors

## 2.2 interne Verarbeitung

Aufbauend auf diesen Basistypen muß es möglich sein, andere Typen zu definieren. Die Metatypsoftware stellt einen Satz von Funktionen zur Verfügung, mit denen neue Metatypen zusammengestellt werden können. Dabei ist zu beachten, daß die Verwaltung der Metatypen komplett der Systemsoftware überlassen wird. Der Anwendungsprogrammierer arbeitet nur mit eindeutigen Bezeichner für die Metatypen.

Ein Beispiel aus dem Auftragskonzept*:*

*Auftrag A;*
*short s = 200;*

*mt_Setze_Inhalt(A,s,MT_SHORT);*

## 2.3 benuterdefinierte Metatypen

Aufbauend auf diesen Basistypen können weitere Metatypen definiert werden. Der Anwendungsprogrammierer teilt dem System mit, daß er einen neuen Metatyp definieren möchte und erhält einen eindeutigen Bezeichner für diesen Metatyp zurück. Anschließend wird dem System mitgeteilt, aus welchen Komponenten dieser Metatyp besteht. Das System erhält mit diesen Informationen genaue Erkenntnisse darüber, wie die „Rechnerstruktur“ dieses

Metatyps aussieht und kann aufgrund der Typbeschreibungen die „Rechnerstruktur“ in die „Netzwerkstruktur“ übertragen.
Wird ein Metatyp nicht mehr benötigt, so kann er wieder aus System gelöscht werden.

Ein Beispiel:

```
typedef struct _trans {
        short smallint;
        char character[20];
        long longint;
} TRANS;

Auftrag A;
int mt_id;
TRANS t;

mt_id = mt_OpenMetaRecord(sizeof(TRANS),“TRANS“);
mt_PushMetaVar(mt_id,&t,&t.smallint,MT_SHORT,1,“smallint“);
mt_PushMetaVar(mt_id,&t,&t.character[0],MT_CHAR,20,“character“);
mt_PushMetaVar(mt_id,&t,&t.longint,MT_LONG,1,“longint“);

mt_Setze_Inhalt(A,t,mt_id);
...
mt_CloseMetaTyp(mt_id);
```

# 3 Funktionsbeschreibung

Die Funktionsbeschreibung erfolgt auf den nächsten Seiten.

NAME

mt_OpenMetaRecord - Einrichten einer neuen Metatypbeschreibung für einen Verbund

SYNOPSIS

**int mt_OpenMetaRecord(int siz, char *name)**

name - optionaler Name für diesen Verbund

BESCHREIBUNG

Die Funktion mt_OpenMetaRecord richtet einen neuen Metatyp ein. Der Parameter siz gibt die Rechnergröße eines Verbundes an. Mit dem zweiten Parameter kann optional diesem Verbund einen Namen hinzugefügt werden.

RETURN VALUES

>= 10 - eindeutiger Bezeichner für diesen Verbund

< 0 - Fehler

BEISPIEL

```
typedef struct _trans {
    short smallint;
    char character[20];
    long longint;
} TRANS;

int mt_id;
TRANS t;

mt_id = mt_OpenMetaRecord(sizeof(TRANS),“TRANS“);
```

NAME

mt_OpenMetaSequence- Einrichten einer neuen Metatypbeschreibung für eine Folge

SYNOPSIS

**int mt_OpenMetaSequence(char *name)**

name - optionaler Name für diese Folge

BESCHREIBUNG

Die Funktion mt_OpenMetaSequence richtet einen neuen Metatyp für eine Folge ein. Mit dem Parameter kann optional dieser Folge einen Namen gegeben werden. Mit Hilfe dieser Funktion können Folgen von Verbunden beschrieben werden, wobei die Anzahl der Elemente einer Folge nicht bekannt sein muß.

RETURN VALUES

>= 10 - eindeutiger Bezeichner für diese Folge

< 0 - Fehler

NAME

mt_CloseMetaTyp- Freigeben einer Metatypbeschreibung für Verbunde und Folgen

SYNOPSIS

**int mt_CloseMetaTyp(int mt_id)**

mt_id - Bezeichner für einen vorher eingerichteten Metatyp

BESCHREIBUNG

Die Funktion mt_CloseMetatyp entfernt eine mit mt_OpenMetaRecord oder mt_OpenMetaSequence eingerichtete Metatypbeschreibung.

RETURN VALUES

= 0 - erfolgreiches Entfernen

< 0 - Fehler

NAME

mt_PushMetaVar- Hinzufügen einer Beschreibung von Elementen eines Verbundes oder einer Folge

SYNOPSIS

**int mt_PushMetaVar(int id, const void *h,const void *v,int typ,int len, char *name**)

| | |
|---|---|
| id | - Bezeichner für diese Metatypbeschreibung |
| h | - Anfangsadresse eines Verbundes |
| v | - Adresse der hinzukommenden Variable eines Verbundes |
| typ | - Typ dieser Variable (Basistyp oder benutzerdefinierter Typ) |
| len | - Größe des Arrays der Variable |
| name | - optionaler Name für dieser Variable |

BESCHREIBUNG

Diese Funktion fügt eine neue Variable eines Verbundes zum Metatyp hinzu. Die Variable h enthält die Anfangsadresse des Verbundes und v enthält die Adresse der neu zu definierenden Variablen.

RETURN VALUES

| | |
|---|---|
| = 0 | - erfolgreiches Einrichten |
| < 0 | - Fehler |

BEISPIEL:

```
int mt_id;
TRANS t;

mt_id = mt_OpenMetaRecord(sizeof(TRANS),“TRANS“);
mt_PushMetaVar(mt_id,&t,&t.smallint,MT_SHORT,1,“smallint“);
mt_PushMetaVar(mt_id,&t,&t.character[0],MT_CHAR,20,“character“);
mt_PushMetaVar(mt_id,&t,&t.longint,MT_LONG,1,“longint“);
```

NAME

mt_Setze_Inhalt, mt_Lese_Inhalt - Beschreiben und Auslesen der Inhalte eines Auftragsformulars unter Berücksichtigung der Metatypbeschreibung (Makros!)

SYNOPSIS

**void mt_Setze_Inhalt(Auftrag A, void I, int mt_id)**
**int mt_Lese_Inhalt(Auftrag A, void I, int mt_id)**

| | |
|---|---|
| A | - Auftragsformular aus dem Auftragskonzept |
| I | - Verbund oder Folge |
| mt_id | - Bezeichner für einen vorher eingerichteten Metatyp |

BESCHREIBUNG

Das Makro mt_Setze_Inhalt setzt die Inhaltstruktur eines Auftragsformulars. Hier wird nur ein Verweis eingerichtet. Erst beim Auftragerteilen wird eine Kopie dieses Inhalts erstellt und an den Auftragnehmer übertragen.

Das Makro mt_Lese_Inhalt beschreibt mit Hilfe der Metatypbeschreibung die Variable I. Anschließend wird die Inhaltsstruktur in dem Auftragsformular freigegeben.

RETURN VALUES

| | |
|---|---|
| = 0 | - erfolgreiches Kopieren |
| < 0 | - Fehler |

BEISPIEL

Auftraggeber

```
Auftrag A;
int mt_id;
TRANS t;

mt_id = mt_OpenMetaRecord(sizeof(TRANS),“TRANS“);
```

```
mt_PushMetaVar(mt_id,&t,&t.smallint,MT_SHORT,1,“smallint“);
mt_PushMetaVar(mt_id,&t,&t.character[0],MT_CHAR,20,“character“);
mt_PushMetaVar(mt_id,&t,&t.longint,MT_LONG,1,“longint“);
mt_Setze_Inhalt(A,t,mt_id);
Erteile_Auftrag(A);
```

## Auftragnehmer

```
Auftrag A;
int mt_id;
TRANS t;

mt_id = mt_OpenMetaRecord(sizeof(TRANS),“TRANS“);
mt_PushMetaVar(mt_id,&t,&t.smallint,MT_SHORT,1,“smallint“);
mt_PushMetaVar(mt_id,&t,&t.character[0],MT_CHAR,20,“character“);
mt_PushMetaVar(mt_id,&t,&t.longint,MT_LONG,1,“longint“);
Erwarte_Auftrag(A);
mt_Lese_Inhalt(A,t,mt_id);
```

NAME

mt_GetMetaTypName, mt_GetMetaVarName, mt_GetFirstMetaVar, mt_GetNextMetaVar, mt_IsUserMetaTyp - Navigieren innerhalb einer Metatypbschreibung und Ausgabemöglichkeiten der optionalen Namen

SYNOPSIS

**char *mt_GetMetaTypName(int mt_id)**
**char *mt_GetMetaVarName(const METAVAR *mv)**
**METAVAR *mt_GetFirstMetaVar(int mt_id)**
**METAVAR *mt_GetNextMetaVar(const METAVAR *mv)**
**int mt_IsUserMetaTyp(int mt_id)**

mt_id - Bezeichner für eine Metatypbeschreibung
mv - Element einer Metatypbeschreibung (Variable)
I - Verbund oder Folge
mt_id - Bezeichner für einen vorher eingerichteten Metatyp

BESCHREIBUNG

Die Funktion mt_GetMetaTypName gibt den optionale Namen der Metatypbeschreibung mt_id zurück.

Mit Hilfe der Funktionen mt_GetFirstMetaVar und mt_GetNextMetaVar können durch alle Variablenbeschreibungen navigiert werden.

Die Funktion mt_IsUserMetaTyp entscheidet, ob es bei einer Metatypbeschreibung um einen benutzerdefinierten Metatyp oder um einen Basismetatyp handelt.

BEISPIEL

Beispiel für die Namensausgabe aller Variablen eines Metatyps:

```
int PrintNames(int id)
{
  METAVAR *mv;
  for (mv = mt_GetFirstMetaVar(id); mv; mv = mt_GetNextMetaVar(id))
```

```
    {
        if (mt_IsUserMetaTyp(mv->id))
            PrintNames(mv->id);
        else
            printf(„%s\n“,mt_GetMetaVarName(mv));
    }
}
```