19

Kapitel 19
Control Language (CL)

Control Language (CL) ist die wichtigste Programmiersprache auf der AS/400, weil jede AS/400 sie haben muss. CL ist ein integraler Teil des Betriebssystems und es ist die Art, wie Sie die AS/400 steuern. CL wurde mit dem Betriebssystem der S/38, CPF, ins Leben gerufen.

Seit der CPF-Veröffentlichung 1.0 hat CL eine Reihe von Verbesserungen durchlaufen, die bis zum heutigen Tag anhalten.

Befehle und Parameter

CL besteht aus Befehlen und Befehle bestehen aus einem Befehlsnamen und einer Liste von Parametern. Deshalb hat CL ein bestimmtes Aussehen, das die alltäglichen Programmiersprachen nicht teilen. Die Addition der numerischen Variablen A, B und C (Speichern des Ergebnisses in R) zum Beispiel würde in einer Sprache wie C wie folgt ausgedrückt werden:

```
r = a + b + c;
```

CL drückt dieselbe Anweisung in einer Form aus, die auf den ersten Blick völlig unterschiedlich aussieht:

```
CHGVAR VAR(&R) VALUE(&A + &B + &C)
```

Der Name des Befehls ist Change Variable (CHGVAR) und er hat zwei Parameter, die die Schlüsselwörter (Namen) VAR und VALUE haben. Der Wert des VAR-Parameters ist &R, was eine Variable ist. Der Wert des VALUE-Parameters ist der Ausdruck &A + &B + &C.

CL-Befehle können in ein Quelldateimitglied (Quellentyp CLP) geschrieben werden und das Mitglied kann in ein Programm mit dem Create-CL-Program-(CRTCLPGM)-Befehl kompiliert werden.

Wann man CL verwendet

Sie können CL verwenden, immer, wenn Ihr Programm eine Systemaktivität durch einen Befehl starten soll. Sie könnten zum Beispiel ein CL-Programm erschaffen, das dem Benutzer ein Menü präsentiert. Wenn der Benutzer die mit "Starte Drucker" gekennzeichnete Option auswählt, würde das CL-Programm die Start Printer Writer-(STRPRTWTR)- oder Release-Writer-(RLSWTR)-Befehle starten.

TIPP: Jeder der in OS/400 verfügbaren Befehle kann von innerhalb eines RPG-Programms ausgeführt werden, indem man die Anwendungsprogrammschnittstelle, Command Execute (QCMDEXC) verwendet. Befehle, die Daten wiedergeben, werden Daten jedoch nicht durch die API wiedergeben.

Sie können CL-Programme schreiben, die fast jeden in OS/400 verfügbaren Befehl ausführen. Zusammen mit ein paar elementaren Programmfluss-Steuerbefehlen kann ein CL-Programm die Operation Ihrer AS/400 automatisieren, indem Sie die Notwendigkeit eliminiert, dass Sie die Befehle manuell ausführen.

TIPP: CL sollte nicht für normale Datenverarbeitungsaktivitäten verwendet werden, weil es für diese Aufgabe nicht geeignet ist. CL ist etwas langsamer als andere Sprachen wie RPG oder COBOL und liefert nur die elementarste Unterstützung zum Verarbeiten von Datenbankdateien. CL kann nur von Datenbankprogrammen lesen; eine Datei pro Programm.

Teile eines CL-Programms

Alle CL-Programme haben die selbe allgemeine Gliederung:

1. Der Program-(PGM)-Befehl an der obersten Spitze des Programms. Der PGM-Befehl markiert den Beginn des Programms. Wenn irgendwelche Parameter von diesem Programm empfangen werden sollen, sind sie in dem PARM-Parameter aufgelistet.
2. Eine Reihe von Declare-(DCL)-Befehlen zum Definieren aller Variablen, die in dem Programm verwendet werden. Alle Variablen müssen ausgewiesen werden, bevor das Programm irgendetwas anderes macht. Folglich müssen alle DCL-Befehle direkt nach dem PGM-Befehl zusammen gruppiert werden. Wenn das CL-Programm eine Datei benutzt, muss sie hier auch mit dem Declare-File-(DCFL)-Befehl ausgewiesen werden.
3. Eine optionale, globale Fehlerfalle mit dem Monitor-Message-(MONMSG)-Befehl. Dieser wird später detaillierter beschrieben.
4. Der Körper des Programms, der alle Befehle, die notwendig zum Ausführen der Aufgabe sind, an die Sie denken, ausführt. Sie können Programmfluss-Steuerbefehle einfügen zum Ändern der Reihenfolge, in der die Befehle ausgeführt werden. Programmfluss-Steuerbefehle werden an späterer Stelle beschrieben.
5. Der End-Program-(ENDPGM)-Befehl zum Kennzeichnen des Programmendes.

CL-Variablen

CL unterstützt drei Arten von Variablen: Zeichen, Dezimal und logische.

- Zeichenvariablen werden mit TYPE(*CHAR) ausgewiesen. Sie können jede beliebige Länge zwischen 1 und 9.999 Zeichen haben. Ihnen kann jeder Wert gegeben werden, einschließlich Hexadezimalwerte wie zum Beispiel X‘01‘.
- Dezimalvariablen werden mit TYPE(*DEC) ausgewiesen. Sie können jede Länge zwischen 1 und 15 Ziffern haben und haben zwischen 0 und 9 Dezimalstellen. Sie können jeden numerischen Wert, der innerhalb diese Begrenzungen fällt, haben, sei er positiv, Null oder negativ.
- Logische Variablen werden mit TYPE(*LGL) ausgewiesen. Sie sind immer ein Byte lang und können nur zwei Werte haben: wahr (‚1‘) oder falsch (‚0‘).

TIPP: Alle Variablennamen müssen mit einem Undzeichen (&) beginnen. Erst das Undzeichen macht den Rest des Namens zu einer Variablen.

Der Rest des Namens muss den folgenden Regeln konform sein:

- Das erste Zeichen muss ein Buchstabe (A bis Z) oder eines der Zeichen @, # oder $ sein.
- Die folgenden Zeichen in dem Namen können Buchstaben, die Zeichen @, #, $ oder _ (Unterzeilenzeichen) oder Ziffern (0 bis 9) sein.
- Dem &-Zeichen muss mindestens ein Zeichen folgen.
- Nicht mehr als 10 Zeichen dürfen dem &-Zeichen folgen.

Den Variablen Werte geben

CL-Variablen können einen Wert auf drei verschiedene Arten erhalten:

- Wenn die Variable ein Parameter, der Ihrem CL-Programm zugeschoben wurde, ist, erhält die Variable einen Wert von dem Anrufer.
- Indem Sie mit einem Anfangswert ausgewiesen wird. Der DCL-Befehl kann einen optionalen Parameter, VALUE, haben, in dem Sie jeden beliebigen Anfangswert, den Sie der Variablen geben möchten, festlegen können. Wenn Sie VALUE nicht festlegen, dann ist der Default von Zeichenvariablen die Leerstelle, von Dezimalvariablen Null und von logischen Variablen falsch.
- Von dem Change-Variable-(CHGVAR)-Befehl, welcher CLs Zuordnungsbefehl ist. Wenn Sie der Variablen &X den Wert ‚ABC' zuordnen möchten, codieren Sie ihn in einem CHGVAR-Befehl:

```
CHGVAR VAR(&X) VALUE('ABC')
```

  Der VALUE-Parameter kann eine einzige Konstante enthalten, eine Variable oder einen Ausdruck, der aus vielen Variablen und/

```
CHGVAR VAR(&NUMBER) VALUE(3 * (&COUNTER + 2))
```

Neben den üblichen vier arithmetischen Operationen (die Sie nur an Dezimalvariablen durchführen können) können Sie die folgenden Operationen auf Zeichenfolgen verwenden:

- Verkettung. Es gibt drei Arten von Verkettungen: *CAT (das Zusammenfügen der zwei Zeichenfolgen wie sie sind, unter Behalten von folgenden und führenden Leerstellen), *BCAT

(entfernt die nachfolgenden Leerstellen am Ende der ersten Zeichenfolge und fügt dann eine einzige Leerstelle zwischen den beiden ein) und *TCAT (entfernt nachfolgende Leerstellen am Ende der ersten Zeichenfolge und verbindet sie dann).

Wenn zum Beispiel die Variable &FIRST 10 Zeichen lang ist und ‚John' enthält und &LAST 10 Zeichen lang ist und ‚Smith' enthält, dann ergibt &FIRST *CAT &LAST ‚John Smith' (sechs Leerstellen dazwischen), weil *CAT die nachfolgenden Leerstellen in &FIRST behält. &FIRST *BCAT &LAST würde ‚John Smith' (eine Leerstelle) ergeben und &FIRST *TCAT &LAST ergäbe ‚JohnSmith'.

- Unterzeichenfolge. Die Unterzeichenfolgefunktion lässt Sie einen Teil einer Zeichenfolge herausziehen und das Ergebnis in eine andere Zeichenfolgenvariable platzieren. Die Unterzeichenfolgefunktion wird als %SST oder %SUBSTRING bezeichnet. Wenn Sie die ersten drei Zeichen von &FIRST herausziehen möchten, codieren Sie %SST(&FIRST 1 3). Die ‚1' zeigt die beginnende Stelle für das Herausziehen an und die ‚3' wie viele Zeichen herausgezogen werden sollen.

  Beispiel:

```
CHGVAR VAR(&NAME) VALUE(&FIRST *BCAT %SST(&LAST 1 4))
```

  Dieser CHGVAR-Befehl verkettet (mit einer einzigen Leerstelle dazwischen) die Variable &FIRST und die ersten vier Zeichen von &LAST. Wenn &FIRST ‚John' enthält und &LAST ‚Smith' enthält, erhält &NAME den Wert ‚John Smith'.

Steuerbefehle

Alle Programmiersprachen müssen Anweisungen verwenden, die das Ausführen des Programms steuern, weil es in den meisten Fällen nicht wünschenswert ist, die Anweisungen in einem Programm sequentiell auszuführen. Programme haben immer Schleifen, Entscheidungen und Sprünge.

CL ist mit diesen Eigenschaften eher schlecht bestückt. Tatsächlich hinkt sie hinter den anderen Sprachen hinterher und liefert überhaupt keine Schleifeneigenschaften oder Unterroutinen. Wenn Sie ein hochentwickeltes Programm schreiben werden, sollten Sie eine andere Sprache als CL verwenden.

IF- und ELSE-Befehle

Alle Entscheidungsprozesse werden von dem IF-Befehl ausgeführt. Die allgemeine Form ist:

```
IF COND(...) THEN(...)
ELSE CMD(...)
```

Die Bedingung, die in dem COND-Parameter enthalten ist, wird zuerst ausgewertet. Wenn sie wahr ist, führt das Programm die Anweisung, die indem THEN-Parameter enthalten ist, aus. Wenn sie falsch ist, dann führt es die Anweisung in dem CMD-Parameter von ELSE aus. Der ELSE-Befehl ist optional.

Wenn Sie mehr als eine Anweisung in einem oder beiden Fällen der Entscheidung auszuführen haben, können Sie das DO/ENDDO-Befehlspaar wie folgt verwenden:

```
IF COND(...) THEN(DO)
:
:
ENDDO
ELSE CMD(DO)
:
:
ENDDO
```

In diesem Fall führt das Programm die in dem ersten DO/ENDDO-Paar eingebetteten Anweisungen durch, wenn die Bedingung wahr ist oder die zweite DO/ENDDO, wenn sie falsch ist.

Der COND-Parameter kann irgendeinen Ausdruck enthalten, der zu einem wahren oder falschen Ergebnis führt, wie zum Beispiel:

```
IF COND(&A *EQ &B)
```

In diesem Beispiel sind &A und AB Variablen desselben Typs (Zeichen, Dezimal oder logisch). Der *EQ-Operator vergleicht sie auf ihre Gleichheit. Tabelle 19.1 zeigt alle der verfügbaren Vergleichsoperatoren:

Tabelle 19.1: Vergleichsoperatoren

Operator	Description
*EQ	Equal
*NE	Not equal
*LT	Less than
*LE	Less than or equal to
*NL	Not less than (same as *GE)
*GT	Greater than
*GE	Greater than or equal to
*NG	Not greater than (same as *LE)

Sie können diese Vergleiche auch mit *AND, *OR und *NOT kombinieren, wenn notwendig, unter Verwendung von Klammern. CL erlaubt sehr komplizierte Bedingungen in dem COND-Parameter.

GOTO-Befehl

Die GOTO-Anweisung ist praktisch tabu in den meisten Programmiersprachen, wird in CL jedoch noch verwendet. In CL gibt es keine andere Möglichkeit, um von einem Ort des Programms zu einem anderen zu springen, deshalb muss sie sich ausschließlich auf OTO verlassen.

Die GOTO-Anweisung überträgt immer die Steuerung zu einem Etikett (oder “Schild”) in dem CL-Programms. Ein Etikett ist der Name gefolgt von einem Doppelpunkt (:), der vor dem Befehlsnamen geschrieben wird. Zum Beispiel:

```
GOTO CMDLBL(HERE)
:
:
HERE: CHGVAR ...
```

Die GOTO-Anweisung überträgt die Steuerung an das Etikett HERE, welches auf die CHGVAR-Anweisung hindeutet.

Der CALL-Befehl

Ein anderer Steuerbefehl ist der CALL-Befehl. Mit CALL können Sie ein anderes Programm (das in jeder beliebigen Sprache geschrieben sein kann) aufrufen und Parameter verschieben. Wenn das andere Programm beendet, gibt das System automatisch die Steuerung an Ihr CL-Programm zurück. Das CL-Programm nimmt die Ausführung bei der Anweisung wieder auf, die unmittelbar dem CALL folgt.

Sie können entweder Variablen oder Konstanten als Parameter verschieben. Im Allgemeinen ist das Verschieben von Parametern als Konstanten schwierig. Es gibt Möglichkeiten, es zu schaffen, dass es funktioniert, aber diese Diskussion geht über die Reichweite dieses Buches hinaus. Ein guter Ansatzpunkt wäre für Sie, die Parameter immer als Variablen zu verschieben. Wenn Sie an Erfahrung gewinnen, können Sie anfangen, mit Konstanten zu experimentieren.

Der Anrufer und das angerufene Programm muss in der Anzahl der Parameter übereinstimmen. Wenn Programm A Programm B anruft, muss der CALL-Befehl in Programm A dieselbe Anzahl von Parametern auflisten, wie Programm B bei seinem Eintrittspunkt. Wenn das gerufene Programm ein RPG-Programm ist, muss die Anzahl der Parameter nicht unbedingt übereinstimmen. Der Kompiler überprüft nicht, ob die Anzahl der Parameter

gleich ist, so dass das RPG-Programm nicht "explodieren" wird, bis es tatsächlich einen Parameter zu verwenden versucht, der ihm nicht zugeschoben wurde.

Parameter müssen auch in Typ und Länge übereinstimmen, wenn auch manche Sprachen hierin weniger streng sind als andere. Wenn Sie Zweifel haben, dann achten Sie darauf, dass Sie übereinstimmen. Wenn Programm A B ruft unter Auflistung einer 10-Zeichen-Zeichenfolgenvariablen und einer 7-ziffrigen numerischen Variablen, gehen Sie sicher, dass das Programm B das selbe erwartet.

Wenn das angerufene Programm beendet, werden die Parameter an den Anrufer zurückgegeben. Die Parameter können durch das Programm, das Sie angerufen haben, verändert worden sein.

ENDPGM und RETURN

Die ENDPGM und RETURN-Befehle können verwendet werden, um das Ende eines Programms zu kennzeichnen. ENDPGM muss die allerletzte Anweisung in einem CL-Programm sein, aber RETURN kann überall sein. Während es nur eine ENDPGM-Anweisung geben kann, kann eine beliebige Anzahl von RETURNs über das CL-Programm verstreut sein.

Die Befehle führen dieselbe Funktion aus; es gibt überhaupt keinen Unterschied. Sie können RETURN verwenden, um ein Programm zu beenden, egal, wo Sie sich innerhalb des Programms befinden. Dieser Befehl erspart Ihnen das Codieren eines GOTO-Befehls, um Sie zu dem ENDPGM zu führen. Wenn das CL-Programm, das Sie gestartet haben, von einem anderen gerufen wurde, kann entweder ENDPGM oder RETURN verwendet werden, um zu dem Anrufer zurückzukehren.

TIPP: Gute Codierstandards sollten nach einem einzigen Ausgangspunkt für ein Programm rufen. Wenn Sie RETURN-Anweisungen über das ganze Programm verstreuen, können Sie es schwierig für andere Programmierer machen, das Programm zu erhalten.

Die Behandlung von Fehlern

Eine der Stärken von CL ist die Fähigkeit, Fehlersituationen würdig zu behandeln. Was diese Sache angeht, ist sie unübertrefflich.

*ESCAPE-Nachrichten

Jedes Mal, wenn das System ein Problem mit einem Befehl, den Sie starten, findet, schickt es eine *ESCAPE-Nachricht. Eine vollständige Erklärung der AS/400-Nachrichten wird in diesem Grundlagenbuch nicht geliefert. Nachrichten sind von anderen Typen und einer dieser Typen ist *ESCAPE. Fluchtnachrichten werden nur ausgeschickt, wenn es einen Fehler gibt, der ernst genug ist, dass er das Rückgängigmachen eines Befehls, den Sie zu starten angefordert haben, verdient.

Tippen Sie zum Beispiel den folgenden Befehl auf der Befehlszeile:

```
DSPLIB LIB(XYZ) OUTPUT(*PRINT)
```

Wenn das System diese Anfrage erhält, prüft es, um zu sehen, ob die Bibliothek XYZ existiert. Wenn dies der Fall ist, startet der DSPLIB-Befehl ohne Probleme. Wenn dies nicht der Fall ist, schickt das Befehl eine *ESCAPE-Meldung los.

Weil Sie den DSPLIB-Befehl von der Tastatur aus starten, sehen Sie die Meldung und denken: “Oh, ich muss wohl einen Fehler gemacht haben!” Dann versuchen Sie es noch einmal. Vielleicht war der Name der Bibliothek nicht XYZ sondern XXX.

CL-Programme haben nicht so viel Glück. Wenn der DSPLIB-Befehl, der oben gezeigt ist, ein Teil eines CL-Programms wäre und Sie das Programm ausführen, würde der DSPLIB-Befehl genauso schnell versagen und die selbe *ESCAPE-Meldung ausschicken. Diese *ESCAPE-Meldung würde Ihr CL-Programm aber zum Abbruch zwingen.

Der MONMSG-Befehl

Sie können den MONMSG-Befehl verwenden, um *ESCAPE-Meldungen zu fangen und das Problem würdig zu erledigen. Um diesen Befehl zu verwenden, müssen Sie Folgendes machen:

1. Finden Sie heraus, welche Nachrichten-ID das System für die Fehlerbedingung ausgibt, für die Sie bereit sein möchten. Der Delete File-(DLTF)-Befehl gibt die Nachricht CPF2105 (die Nachrichten-ID) heraus, wenn die Datei nicht existiert.
2. Codieren Sie einen MONMSG-Befehl sofort unmittelbar nach der Anweisung, von der Sie denken, dass sie die *ESCAPE-Meldung los schicken wird. Sie würden zum Beispiel einen MONMSG-Befehl nach dem DLTF-Befehl codieren, wenn Sie denken, dass die Datei nicht existieren wird, wenn der DLTF-Befehl ausgeführt ist.

Hier ist ein Beispiel, wie die Reihenfolge aussieht:

```
DLTF FILE(MYLIB/MYFILE)
MONMSG MSGID(CPF2105) EXEC(...)
```

Der MONMSG-Befehl, der oben gezeigt wurde, hat zwei Parameter: MSGID, der verschiedene Nachrichten-IDs auflistet, auf die Sie achten wollen (sie können tatsächlich mehrere einschließen) und den EXEC-Parameter, der einen Befehl enthalten sollten, den Sie starten möchten, wenn CPF2105 ausgeschickt wird.

Sie können den EXEC-Parameter weglassen, wenn Sie nichts tun wollen, wie zum Beispiel, wenn Sie die *ESCAPE-Meldung ignorieren möchten. Alternativ können Sie auch einen DO-Befehl in den EXEC-Parameter stecken, wenn Sie verschiedene Befehle ausführen möchten. In diesem Fall wird die Reihenfolge der Befehle mit einem ENDDO-Befehl beendet werden müssen:

```
DLTF FILE(MYLIB/MYFILE)
MONMSG MSGID(CPF2105) EXEC(DO)
:
:
ENDDO
```

Der globale MONMSG-Befehl

Der MONMSG-Befehl kann auch an die Spitze des CL-Programms platziert werden, unmittelbar nach den DCL/DCLF-Befehlen. Wenn es hier platziert wird, agiert MONMSG wie eine Decke, die das gesamte Programm zudeckt. Jede Anweisung in dem CL-Programm ist durch den MONMSG-Befehl "geschützt". Dieses Arrangement wird üblicherweise ein "globaler" MONMSG oder ein MONMSG auf Programmebene genannt.

Der globale MONMSG kann nur einen GOTO-Befehl in seinem EXEC-Parameter enthalten oder der EXEC-Parameter kann fehlen. Normalerweise wird der globale MONMSG für Catch-All-Situationen verwendet. In diesem Fall gehört es zur allgemeinen Praxis, nach CPF0000 Ausschau zu halten.

CPF0000 ist ein Joker. Es ist soviel, wie wenn man sagt, "Halte Ausschau nach jeder CPFXXXX-Fehlermeldung." Das CL-Programm wird vor jeder CPFXXXX-Fehlermeldung geschützt sein. Eine andere Nullstellen-Meldungs-ID ist jede Nachrichten-ID, die mit zwei Nullen endet, so wie CPF1400, die jede Nachricht zwischen CPF1401 und CPF14FF abdeckt.

Das Abrufen von Daten

Eine der Spezialitäten von CL ist ihre Fähigkeit, Daten von Systemobjekten wie zum Beispiel Benutzerprofilen, Datengebieten und Systemwerten abrufen zu können.

Datengebiete

Datengebiete sind Objekte, die in Bibliotheken des Typs *DTAARA enthalten sind, die Sie verwenden können, um kleine Mengen von Informationen zu speichern, wie zum Beispiel Berichtüberschriften oder den Namen Ihrer Firma.

Manuell können Sie Datengebiete erschaffen, ändern, löschen und anzeigen. Sie verwenden die offensichtlichen Befehle CRTDTAARA, CHGDTAARA, DLTDTAARA und DSPDTAARA. Diese Befehle können leicht in CL-Programme platziert werden.

Immer, wenn Sie ein Datengebiet manipulieren, müssen Sie sich auf die beginnende Position und die Länge der Daten beziehen. Sie werden zum Beispiel vielleicht die Inhalte des Datengebietes ändern wollen. Sie verwenden den CHGDTAARA-Befehl und legen fest, mit welcher Position Sie mit der Änderung beginnen möchten und wie viele Zeichen es sind:

```
CHGDTAARA DTAARA(ABC (25 3)) VALUE('***')
```

Dieser Befehl ändert die Positionen 25, 26 und 27 des Datengebietes ABC zu Sternen.

CL erlaubt es Ihnen, ihre Inhalte abzurufen und in eine CL-Variable zu platzieren. Es ist ähnlich dem Lesen einer Datei. Zum Abrufen der Inhalte eines Datengebietes in eine CL-Variable, verwenden Sie den Retrieve-Data-Area-(REVDTAARA)-Befehl. Der Name der CL-Variablen geht in den RTNVAR-Parameter, wie folgt:

```
RTVDTAARA DTAARA(ABC (16 5)) RTNVAR(&VALUE)
```

In diesem Beispiel erhält die Variable &VALUE den Wert, der in den Positionen 16 bis 20 des Datengebietes ABC enthalten ist.

Systemwerte

Systemwerte sind sehr wichtige Objekte, die zum Konfigurieren des Systems verwendet werden. Sie können weitere Informationen zu Systemwerten in Kapitel 10 finden. CL-Programme können die Display-, Change- und Work with System Values-(DSPSYSVAL, CHGSYSVAL und WRKSYSVAL)-Befehle enthalten, die verwendet werden, um sie zu manipulieren. Dies bedeutet, dass ein CL-Programm die Konfiguration des Systems ändern kann, wenn die Person, die das Programm startet, ausreichende Autorisierung besitzt.

CL-Programme können auch den derzeitigen Wert von Systemwerten mit dem Retrieve-System-Value-(RTVSYSVAL)-Befehl abrufen. Je nachdem, welchen Systemwert Sie abrufen, muss der Wert einer Zeichen- oder Dezimalvariablen gegeben werden. Der Wert muss auch eine bestimmte Länge haben, die auch von dem Systemwert abhängt.

Ihr CL-Programm kann den zum Beispiel den Wert QTIME, den Systemwert, der die Systemzeit hält, wie folgt abrufen:

```
RTVSYSVAL SYSVAL(QTIME) RTNVAR(&SYSTIM)
```

Diese Anweisung nimmt an, dass &SYSTIM eine Zeichenfolgevariable ist, die aus sechs Zeichen besteht. Wenn RTVSYSVAL ausgeführt wird, wird &SYSTIM die Tageszeit enthalten des Momentes, zu dem RTVSYSVAL ausgeführt wurde, in dem Format HHMMSS.

Weitere Informationen

Sie können noch viele weitere Informationen in CL-Variablen abrufen, mit anderen Retrieve-(RTVXXX)-Befehlen. Unter ihnen sind die zwei wichtigsten:

- RTVJOBA. Der Retrieve-Job-Attributes-Befehl lässt sie die Einstellungen Ihres eigenen Jobs herausfinden, dessen, in dem das CL-Programm gerade läuft. Sie können zum Beispiel den Namen der derzeitigen Ausgabeschlange mit den OUTQ- und OUTQLIB-Parametern abrufen oder den Namen der Anzeigestation, von der Sie den Befehl starten (wenn Sie ihn interaktiv starten), mit dem Job-Parameter. Der TYPE-Parameter gibt ‚0‘ aus, wenn der Job im Batch läuft oder ansonsten ‚1‘.
- RTVUSRPRF. Der Retrieve-User-Profile-Befehl erlaubt es Ihnen, wichtige Informationen über jedes beliebige Benutzerprofil abzurufen. Wenn Sie nicht festlegen, welches Benutzerprofil, wird Ihr eigenes angenommen.

Wenn Sie möchten, können Sie andere RTVXXX-Befehle, die in OS/400 verfügbar sind, durch Ausführen des folgendes Befehls finden:

```
SLTCMD CMD(QSYS/RTV*)
```

Die Verwendung von Dateien

CL-Programm sind schwach in der Dateiverarbeitung. Tatsächlich können CL-Programme nur drei Arten von Dateien verarbeiten: Datenbank-, Quellen- und Anzeigedateien.

Datenbank- und Quelldateien können nur gelesen (nie geschrieben) werden von einem CL-Programm. CL fehlt es an der Fähigkeit, neue Datensätze zu schreiben, existierende zu aktualisieren oder Datensätze zu löschen.

Anzeigedateien können entweder gelesen oder geschrieben werden. CL unterstützt nicht alle Anzeigedateieigenschaften. Es gibt zum Beispiel keine Unterstützung in CL für Unterdateien, außer für Nachrichtenunterdateien.

> **TIPP:** Als ob diese Einschränkungen noch nicht genug wären, kann CL nur eine Datei verarbeiten. Sie kann nicht eine Datei öffnen, schließen und eine andere verarbeiten. Sie kann nur eine Datei verarbeiten.

Der DCLF-Befehl

Bevor Sie eine Datei in CL verarbeiten, müssen Sie sie mit dem DCLF-Befehl ausweisen. Der DCLF-Befehl muss an den Beginn des Programms platziert werden, entweder bevor oder nach (oder mitten drin) die DCL-Befehle.

Der DCLF-Befehl hat zwei Parameter: FILE, der die Datei identifiziert, die Sie in dem CL-Programm verarbeiten möchten und RCDFMT, der die Datensatzformate benennt, die Sie in dem Programm verwenden werden. Der Parameter hat den Default *ALL und Sie sollten ihn in Ruhe lassen, bis Sie mehr von Datensatzformaten verstehen.

Der DCLF-Befehl zieht tatsächlich die ewige Definition aus der Datei und erstellt CL-Variablen unter Verwendung des DDS-Namens, mit dem Und-Zeichen (&) als Vorsilbe. Wenn die Datei zum Beispiel eine Variable namens INPUT hat, erstellt der DCLF-Befehl eine CL-Variable namens &INPUT mit demselben Typen und derselben Länge wie sie in der DDS der Datei definiert ist.

TIPP: Wenn die Datei nicht mit DDS erstellt wurde (d. h. wenn sie eine “flache” Datei ist), erstellt der DCLF-Befehl eine Variable, die aus einem einzigen Zeichen besteht, die die gleiche Länge hat, wie der Datensatz in der Datei, genauso genannt wie die Datei. Wenn die Datei zum Beispiel MASTER heißt, erstellt DCLF eine CL-Variable namens &MASTER.

Anzeigedateindikatoren werden auch CL-Variablen. Ihre Namen sind &INXX, wobei XX die Indikatornummer ist. Wenn eine Anzeigedatei zum Beispiel den Indikator 03 verwendet, erstellt der DCLF-Befehl eine CL-Variable namens &IN03, die als TYPE(*LGL) (logische) ausgewiesen wird.

Andere Befehle

CL liefert drei Befehle um Dateien Datensatz für Datensatz zu verarbeiten:

- RCVF. Der Receive-File-Befehl liest einen Datensatz aus der Datei. Wenn es keine weiteren Datensätze in der Datei gibt, endet der RCVF-Befehl mit *ESCAPE-Meldung CPF0864. Wenn Sie eine Anzeigedatei verarbeiten, müssen Sie den Namen des Datensatzformates, von dem Sie lesen, angeben, wenn es mehr als eines gibt.
- SNDF. Der Send-File-Befehl schreibt einen Datensatz auf die Datei. Sie können SNDF nur auf Anzeigedateien verwenden und Sie müssen den Namen des Datensatzformates, das Sie schreiben, angeben, wenn es mehr als eines gibt.
- SNDRCVF. Der Send/Receive-File-Befehl schreibt einen Datensatz und liest ihn dann zurück. Wie SNDF kann er nur auf Anzeigedateien verwendet werden und Sie müssen den Namen des Datensatzformates zum Verarbeiten festlegen, wenn es mehr als eines gibt.