Tutorial 2

Erstellen, Kompilieren und Ausführen eines COBOL-Programms

ph v/2010/03

In dieser Aufgabe wiederholen wir das Anlegen von Datasets (Allocate) sowie das Füllen mit Daten unter Verwendung des ISPF-Editors, und Sie lernen kennen, wie man ein COBOL-Programm unter z/OS schreibt, kompiliert und ausführt.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAKT20" erstellt. In allen Dateinamen müssen Sie "PRAKT20" durch ihre eigene Benutzer-ID ersetzen.

Aufgabe: Arbeiten Sie nachfolgendes Tutorial durch.

1. Einrichten der Entwicklungsumgebung

Wir müssen als erstes noch 2 Datasets anlegen. Der eine Dataset soll den Quellkode des COBOL-Programms aufnehmen, der zweite die ausführbare Datei. Einen Dataset haben wir schon in der letzten Aufgabe erstellt: Den "PRAKT20.TEST.CNTL", welcher das JCL-Script enthält und zum Kompilieren benutzt wird.

Aufgabe: Legen Sie den Dataset PRAKT20.TEST.COB ("PRAKT20" durch Ihre Benutzer-ID ersetzen) an. Verwenden Sie die gleichen Parameter wie im Tutorial zur Aufgabe 1.
*Legen Sie den Dataset PRAKT20.TEST.LOAD ("PRAKT20" durch Ihre Benutzer-ID ersetzen) an, welcher die ausführbare Datei nach dem Kompilieren aufnehmen soll. Verwenden Sie wieder die Ihnen bekannten Parameter, mit einem Unterschied: Statt im Dateiformat (Record format) "**F**ixed **B**lock" soll dieser Dataset im Dateiformat "Undefined" erstellt werden. Dazu ist an der dafür vorgesehenen Stelle ein "U" als Parameter anzugeben.*

2. Erstellen des Quelltextes des COBOL-Programms

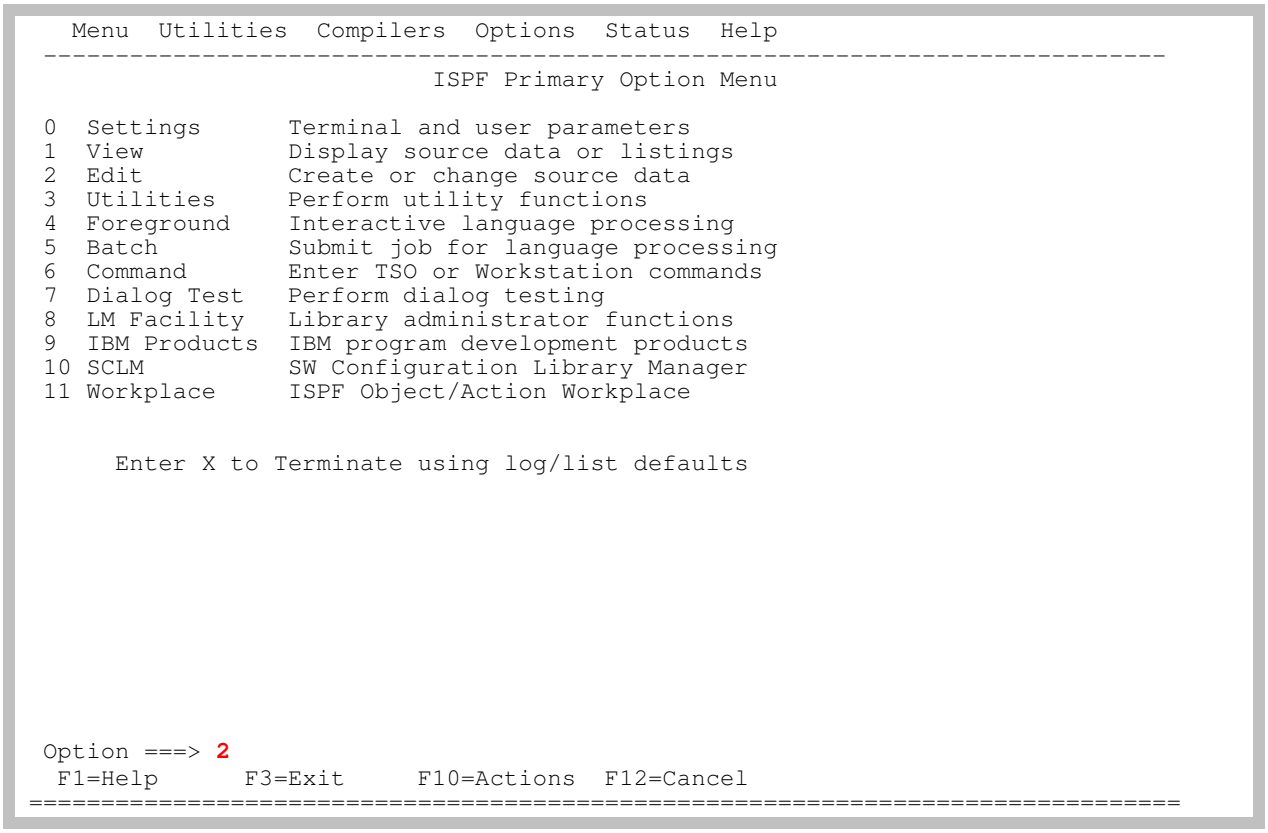


Abbildung 1: "ISPF Primary Option Bildschirm"

Wir haben bisher die Utilities-Funktion benutzt, um unsere Entwicklungsumgebung anzulegen. Hierzu haben wir drei Partitioned Datasets angelegt. Jetzt wollen wir diesen Speicherplatz benutzen, um ein Programm zu schreiben, zu übersetzen und auszuführen.

Dies geschieht mit Hilfe der "Edit"-Funktion. Wie in Abbildung 1 demonstriert, geben wir eine "2" in die Kommandozeile des "ISPF Primary Option Menu" ein und betätigen die Eingabetaste.

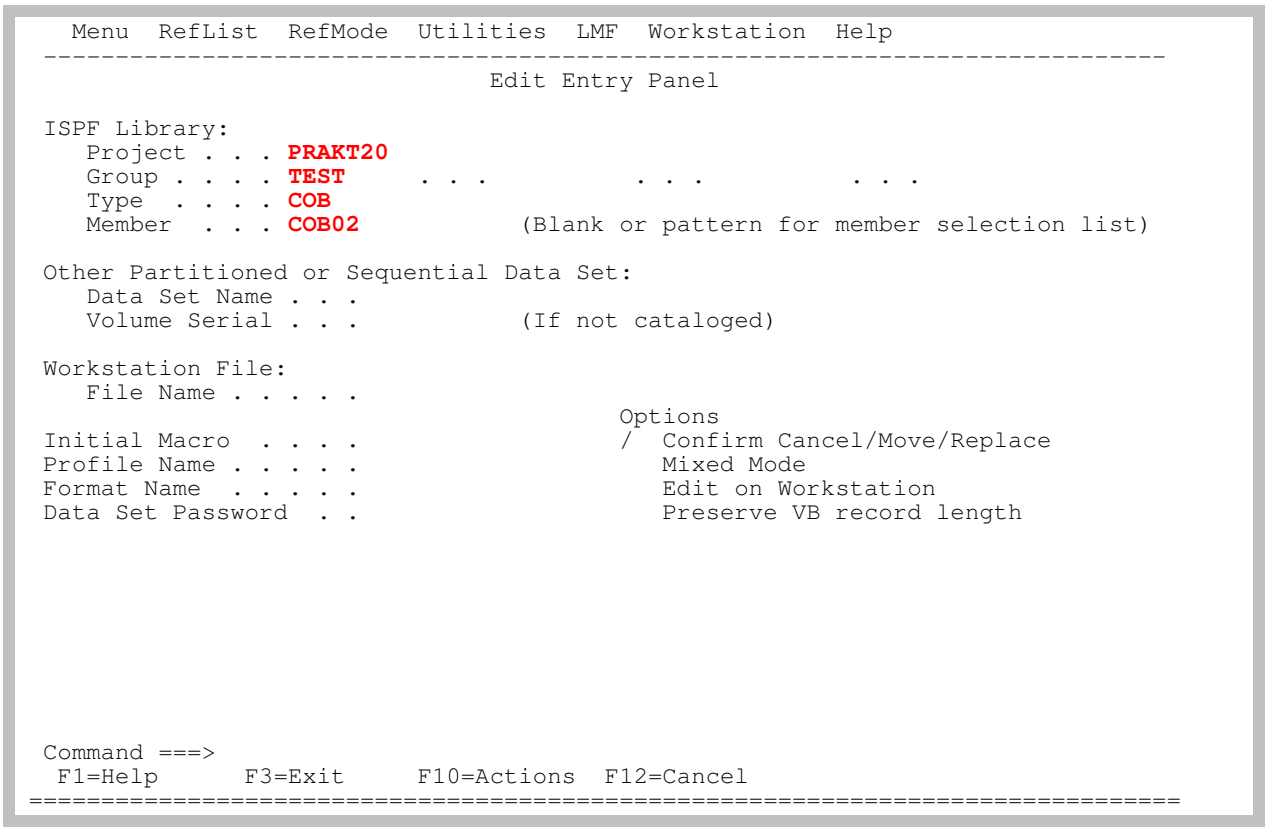


Abbildung 2: "Edit Entry"-Bildschirm

Wir wollen zuerst das Quellprogramm mit Hilfe des ISPF-Editors erstellen. Der "Edit Entry"-Bildschirm fordert uns auf, den Namen des zu editierenden Programms einzugeben (s. Abbildung 2).

Unser Quellprogramm soll als eine (von potentiell mehreren) Files in dem für Quellprogramme von uns vorgesehenen Partitioned Dataset PRAKT20.TEST.COB gespeichert werden. Files innerhalb eines Partitioned Datasets werden als Members bezeichnet. Zur Unterscheidung brauchen die einzelnen Members einen Namen.

Wir bezeichnen unseren Member als COB02. Der volle Name dieses Members ist PRAKT20.TEST.COB.COB02. Wir geben diese Werte in die dafür vorgesehenen Felder des "Edit Entry"-Bildschirmes ein. Es ist also, wie in Abbildung 2 gezeigt, ihre Benutzer-ID ins Feld "***Project***", "TEST" ins Feld "***Group***", "COB" ins Feld "***Type***" sowie "COB02" ins Feld "***Member***" einzutragen. Anschließend betätigen Sie die Eingabetaste.

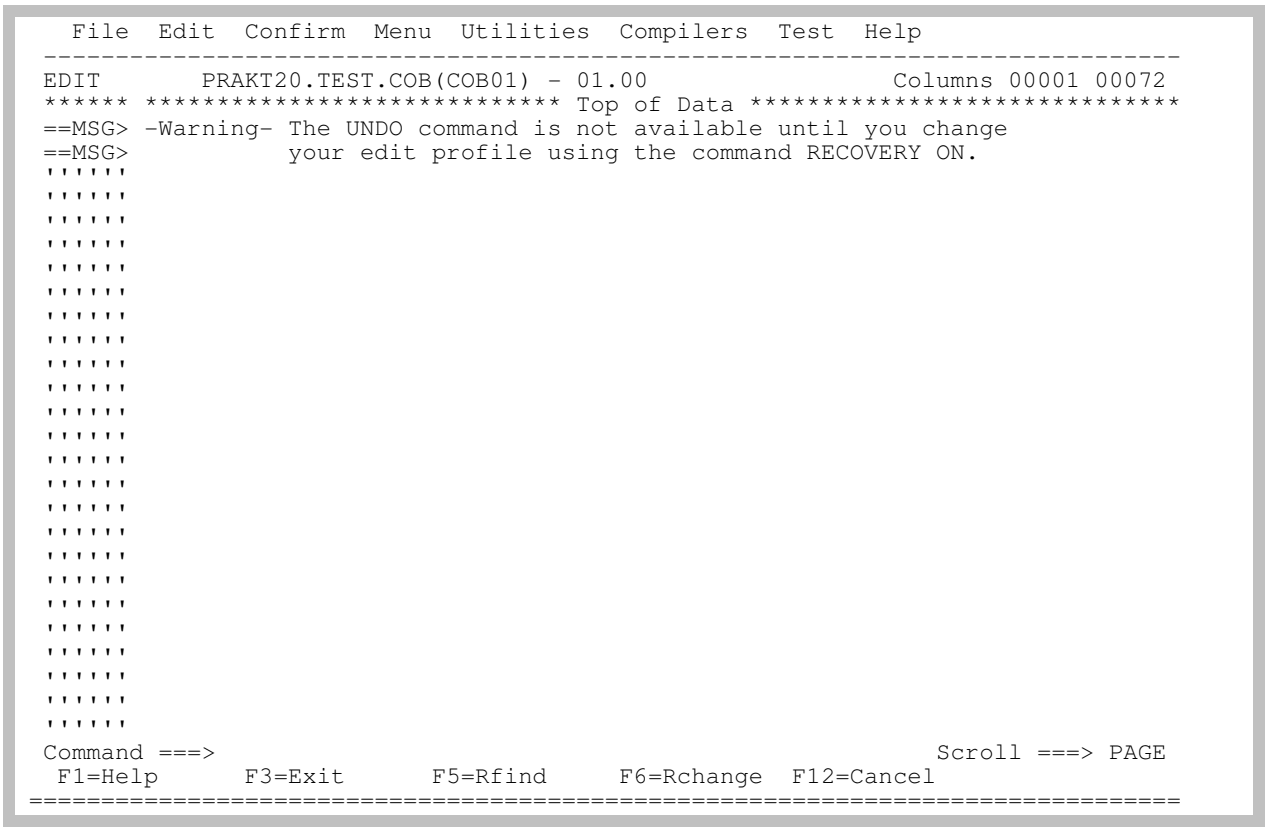


Abbildung 3: ISPF-Editor

Abbildung 3 zeigt die Oberfläche des ISPF-Editors (s. Tutorial 1c). Für die Eingabe des Programmtextes benutzen wir nur die erforderlichen alphanumerischen Tasten. Wir verwenden keine Tasten zur Steuerung außer der DEL-Taste und bewegen den Cursor mit Hilfe der Pfeiltasten.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       PRAKT20.TEST.COB(COB02) - 01.04               Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000100        IDENTIFICATION DIVISION.
 000200        PROGRAM-ID. COB02.
 000300        ENVIRONMENT DIVISION.
 000400        input-output section.
 000410
 000500        FILE-CONTROL.
 000600            SELECT PRINTOUT
 000700                   ASSIGN TO SYSPRINT.
 000800        DATA DIVISION.
 000900        FILE SECTION.
 001000        FD      PRINTOUT
 001100                RECORD CONTAINS 80 CHARACTERS
 001200                RECORDING MODE F
 001300                BLOCK CONTAINS 0 RECORDS
 001400                LABEL RECORDS ARE OMITTED.
 001500        01  PRINTREC         PIC X(80).
 001600        WORKING-STORAGE SECTION.
 001610
 001700        LINKAGE SECTION.
 001800        procedure DIVISION.
 001900        anfang.
 002000            OPEN OUTPUT PRINTOUT.
 002100            move 'Hallo Welt, unser erstes TSO-PROGRAMM in COBOL' to prin
 002200       -    trec.
 Command ===>                                                 Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
================================================================================
```

Abbildung 4: ISPF-Editor mit COBOL-Programm (1/2)

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       PRAKT20.TEST.COB(COB02) - 01.04               Columns 00001 00072
 002300            WRITE PRINTREC.
 002400            CLOSE PRINTOUT.
 002500            GOBACK.
 002600
 ****** **************************** Bottom of Data ****************************
```

```
 Command ===>                                                 Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
================================================================================
```

Abbildung 5: ISPF-Editor mit COBOL-Programm (2/2)

Wir schreiben das in Abbildung 4 und Abbildung 5 gezeigte COBOL-Programm.

Durch Betätigen der F3-Taste kehren wir zum vorherigen Bildschirm zurück. Unser Programm wird automatisch abgespeichert (saved).

```
  Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel            Member COB02 saved

ISPF Library:
   Project . . . PRAKT20
   Group . . . . TEST     . . .          . . .          . . .
   Type  . . . . COB
   Member  . . .                 (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                      Options
Initial Macro  . . . .                /  Confirm Cancel/Move/Replace
Profile Name . . . . .                   Mixed Mode
Format Name  . . . . .                   Edit on Workstation
Data Set Password  . .                   Preserve VB record length




Command ===>
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
================================================================================
```

Abbildung 6: "Edit Entry Panel"-Bildschirm

Rechts oben erscheint die Meldung, dass unser Member abgespeichert wurde (Abbildung 6).

Durch erneutes Eingeben des Member-Namens sowie durch Bestätigen mit der Eingabetaste (Abbildung 2) lässt sich unser Member nochmals aufrufen und bei Bedarf modifizieren.

3. Erstellen und Ausführung des JCL-Scriptes

```
  Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel

ISPF Library:
   Project . . . PRAKT20
   Group . . . . TEST     . . .          . . .          . . .
   Type  . . . . CNTL
   Member  . . . COBSTA02        (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                        Options
Initial Macro  . . . .                  /  Confirm Cancel/Move/Replace
Profile Name . . . . .                     Mixed Mode
Format Name  . . . . .                     Edit on Workstation
Data Set Password  . .                     Preserve VB record length




Command ===>
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
================================================================================
```

Abbildung 7: "Edit Entry Panel"-Bildschirm

Unter Unix brauchen wir in der Regel ein "Make File", um ein COBOL-Programm zu übersetzen. Unter TSO wird dafür ein "Compile Script File" benötigt. Wir legen alle "Compile Scripts" als Members in dem von uns dafür vorgesehenen Partitioned Dataset PRAKT20.TEST.CNTL ab. In der letzten Aufgabe haben wir schon vorgearbeitet und schon ein solches Script PRAKT20.TEST.CNTL(COBSTA02*)* erstellt, das wir nun verwenden wollen, um unser Quell-Programm zu übersetzen, zu linken und das ausführbare Maschinenprogramm (Binary) abzuspeichern.

Die hierfür verwendete Scriptsprache ist die "**J**ob **C**ontrol **L**anguage" (**JCL**).

JCL ist sehr leistungsfähig und bestens geeignet, sich wiederholende komplexe Vorgänge im Großrechnerbereich zu automatisieren. JCL ist der Standard für die Stapelverarbeitung.

Neben JCL existieren weitere Scriptsprachen unter z/OS. Weit verbreitet ist REXX. Letzteres ist etwa vergleichbar mit Perl oder Tcl/TK und wie diese auf unterschiedlichen Plattformen verfügbar.

Wir geben als Typ "CNTL" sowie als Member "COBSTA02" ein und betätigen die Eingabetaste (s. Abbildung 7).

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.TEST.CNTL(COBSTA02) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAKT20C JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M
000300 //STEP1 EXEC IGYWCL
000400 //COBOL.SYSIN DD DSN=&SYSUID..TEST.COB(COB02),DISP=SHR
000500 //LKED.SYSLMOD DD DSN=&SYSUID..TEST.LOAD,DISP=SHR
000600 //LKED.SYSIN DD *
000700  NAME COB02(R)
000800 /*
****** **************************** Bottom of Data ****************************
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
==============================================================================
```

Abbildung 8: JCL-Script

Wenn Sie an der letzten Übung teilgenommen haben, müsste jetzt, wie in Abbildung 9 dargestellt, Ihr damals erstelltes JCL-Script angezeigt werden. Wenn nicht, müssten Sie, wie in Abbildung 8 gezeigt, das JCL-Script nachträglich erstellen.

JCL-Scripte werden dadurch gekennzeichnet, dass alle Zeilen mit "//" beginnen.

Das Script besteht aus 3 Statements, die jeweils in den Zeilen 1, 3 und 4 anfangen. Wenn ein Statement nicht in eine Zeile paßt, besagt ein Komma am Ende der Zeile, dass die Fortsetzung in der nächsten Zeile erfolgt. Zeilen 2, 5 und 6 sind solche Fortsetzungszeilen.

Ein JCL Statement (Record) besteht aus 4 Teilen:

- // in Spalte 1 und 2
- Label Feld, bis zu 8 Zeichen lang, beginnt in Spalte 3
- Statement Type, beginnt in Spalte 12
- Parameter

Das erste Statement in einem JCL-Script ist immer ein "JOB" Statement. Es enthält eine Reihe von Dispositionsparametern, die von dem "z/OS Job Entry Subsystem" ausgewertet werden. Es ist üblich, als Label für das Job-Statement die TSO-Benutzer-ID (hier "PRAKT20") plus einen angehängten Buchstaben zu verwenden. Aus diesem Grund haben TSO-Benutzer-ID's eine maximale Länge von 7 Zeichen.

Das dritte Statement unseres Scripts ist ein EXEC Statement. Es enthält die Anweisung, die Prozedur "IGYWCL" abzuarbeiten. "IGYWCL" ist ein von TSO zur Verfügung gestelltes Script, welches

- den COBOL-Compiler aufruft,

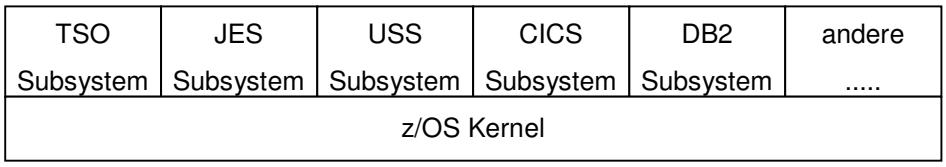
- anschließend den Linkage-Editor aufruft,
- den zu übersetzenden Quelltext als Member eines Partitioned Datasets mit dem Namen INFILE='...' erwartet
- das erstellte Maschinenprogramm unter OUTFILE='...' abspeichert.

Es existiert eine große Anzahl derartiger vorgefertigter Scripte, die zusammen mit z/OS ausgeliefert werden. Der Systemadministrator stellt sie in "JCL Libraries" (JCLLIB) zusammen. Es existieren häufig mehrere "JCL Libraries". Was, wie und wo ist von einer Installation zur nächsten oft verschieden und wird vom Systemadministrator verwaltet.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.TEST.CNTL(COBSTA02) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAKT20C JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //          REGION=4M
000300 //STEP1 EXEC IGYWCL
000400 //COBOL.SYSIN DD DSN=&SYSUID..TEST.COB(COB02),DISP=SHR
000500 //LKED.SYSLMOD DD DSN=&SYSUID..TEST.LOAD,DISP=SHR
000600 //LKED.SYSIN DD *
000700  NAME COB02(R)
000800 /*
****** **************************** Bottom of Data ****************************
 Command ===> SUB                                           Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
==============================================================================
```

Abbildung 9: Ausführung des JCL-Scriptes

Unser Compile- und Link-Script kann nun ausgeführt werden. Wir geben, wie in Abbildung 9 gezeigt, auf der Kommandozeile "SUB" (für Submit) ein und betätigen die Eingabetaste.



Das "**J**ob **E**ntry **S**ubsystem" (**JES**) des Z/OS-Betriebssystems dient dazu, Stapelverarbeitungsaufträge (Jobs) auf die einzelnen CPU's zu verteilen und der Reihe nach abzuarbeiten. Jobs werden dem "JES"-Subsystem in der Form von JCL-Scripten zugeführt, wobei deren erstes JCL-Statement ein JOB-Statement sein muß. PRAKT20.TEST.CNTL(COBSTA02*)* ist ein derartiges Script. Das Kommando "SUB" (Submit) bewirkt, dass PRAKT20.TEST.CNTL(COBSTA02*)* in die Warteschlange der von JES abzuarbeitenden Aufträge eingereiht wird.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAKT20.TEST.CNTL(COBSTA02) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAKT20C JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M
000300 //STEP1 EXEC IGYWCL
000400 //COBOL.SYSIN DD DSN=&SYSUID..TEST.COB(COB02),DISP=SHR
000500 //LKED.SYSLMOD DD DSN=&SYSUID..TEST.LOAD,DISP=SHR
000600 //LKED.SYSIN DD *
000700  NAME COB02(R)
000800 /*
****** **************************** Bottom of Data ****************************









IKJ56250I JOB PRAKT20C(JOB03979) SUBMITTED
***


================================================================================
```

Abbildung 10: Meldung "JOB PRAKT20C(JOB03979) SUBMITTED"

Der JCL-Kommando-Interpreter überprüft die Syntax des Scripts. Falls er keinen Fehler findet, übergibt (submitted) er den Job zur Abarbeitung an das JES-Subsystem. Die Meldung oberhalb der Kommandozeile besagt, dass dies hier der Fall ist (s. Abbildung 10). Der Job erhält die Nummer 03979. Diese Nummer kann z.B. vom Systemadministrator benutzt werden, um den Status der Verarbeitung dieses Jobs abzufragen.

Wir warten einige Sekunden und betätigen anschließend die Eingabetaste. Erscheint keine Meldung, hat JES das JCL-Script noch nicht endgültig abgearbeitet. Wir warten erneut einige Sekunden und Betätigen die Eingabetaste; wir wiederholen dies notfalls mehrfach, bis eine Statusmeldung, so ähnlich wie in Abbildung 11 dargestellt ist, ausgegeben wird.

```
00.27.07 JOB03979 $HASP165 PRAKT20C ENDED AT N1  MAXCC=0 CN(INTERNAL)
 ***
```

Abbildung 11: Statusmeldung nach Abarbeitung des JCL-Scriptes

"MAXCC-0" ist eine Erfolgsmeldung: Die Übersetzung ist erfolgreich durchgeführt worden. "MAXCC-4" ist ebenfalls OK, alles andere besagt, dass ein Fehler aufgetreten ist. In diesem Fall greifen Sie besser zum z/OS-COBOL-Compiler-Handbuch.

Das übersetzte Programm ist nun ausführungsfertig in dem File

PRAKT20.TEST.LOAD(COB02) abgespeichert.

z/OS gestattet es grundsätzlich, Programme entweder interaktiv im Vordergrund oder als Stapelverarbeitungsprozesse durch JES im Hintergrund abzuarbeiten. Ersteres garantiert bessere Antwortzeiten, letzteres führt zu einem besseren Durchsatz. Warum wurde unser Programm im Hintergrund (Stapel) übersetzt ?

Ein z/OS-Server ist normalerweise ein Produktionssystem. Die Programmentwicklung ist dabei ein störender Faktor. Die Entwicklung von z/OS-Anwendungen erfolgt deshalb meistens auf einem separaten Entwicklungssystem. Dieses arbeitet vielfach mit einem für Entwicklungsaufgaben besser geeignetem Betriebssystem. Handelt es sich dabei um einen separaten z/OS-Rechner (oder eine LPAR auf dem gleichen Rechner), so wird dafür häufig das VM/390-Betriebsystem eingesetzt.

Das resultierende Quellprogramm ist dann fehlerfrei und im Normalfall sehr umfangreich. Nur in diesem Zustand wird es auf den z/OS-Produktionsrechner portiert. Das endgültige Übersetzen ist ein längerdauernder Prozess, dessen Ausführung besser im Stapel erfolgt. Diese Übersetzung ist in der Regel Teil eines komplexeren Produktionseinführungsprozesses, den ein Unternehmen benutzt, um unternehmenskritische Anwendungen einzuführen.

4. Ausführung des COBOL-Programms

```
  Menu  Utilities  Compilers  Options  Status  Help
--------------------------------------------------------------------------------
                          ISPF Primary Option Menu

 0  Settings      Terminal and user parameters
 1  View          Display source data or listings
 2  Edit          Create or change source data
 3  Utilities     Perform utility functions
 4  Foreground    Interactive language processing
 5  Batch         Submit job for language processing
 6  Command       Enter TSO or Workstation commands
 7  Dialog Test   Perform dialog testing
 8  LM Facility   Library administrator functions
 9  IBM Products  IBM program development products
 10 SCLM          SW Configuration Library Manager
 11 Workplace     ISPF Object/Action Workplace


      Enter X to Terminate using log/list defaults









 Option ===> tso call 'prakt20.test.load(cob02)'
  F1=Help      F3=Exit     F10=Actions  F12=Cancel
================================================================================
```

Abbildung 12: "ISPF Primary Option Menu"-Bildschirm

Wir sind nun soweit, dass unser Programm ausgeführt werden kann. Durch mehrfaches Betätigen der F3-Taste kehren wir in das "ISPF Primary Option Menu" zurück (s. Abbildung 12). Auf der Kommandozeile geben wir den Befehl

tso call 'prakt20.test.load(cob02)'

ein und betätigen die Eingabetaste. "prakt20.test.load(cob02)" enthält das vom Compiler erzeugte Maschinenprogramm. "call" ist ein TSO-Kommando und ruft ein Programm auf. Wir sind aber im ISPF-Subsystem und nicht im TSO-Subsystem. "tso call" an Stelle von "call" bewirkt, dass der "call"-Befehl auch innerhalb des ISPF-Subsystems aufgerufen werden kann.

Wichtiger Hinweis:

Achten Sie darauf, daß Sie bei dem Befehl "tso call 'prakt20.test.load(cob02)'" die richtigen Hochkommas verwenden. Das Hochkomma, das auf den meisten Tastaturen über dem Zeichen "#" steht, ist das korrekte.

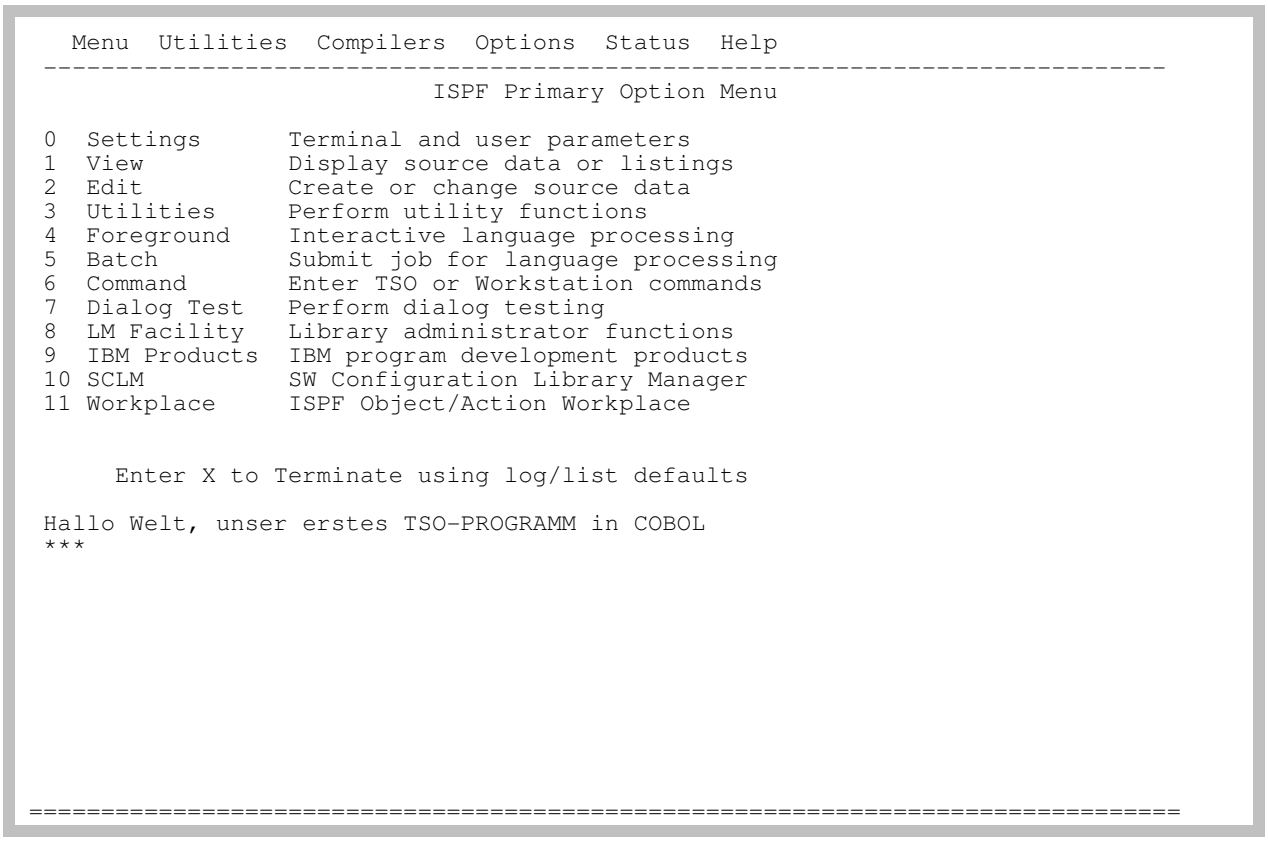


Abbildung 13: Ausgabe unseres COBOL-Programms

Abbildung 13 zeigt: Oberhalb der Kommandozeile erscheint die Ausgabe unseres COBOL-Programms.

Sie können ein neues Quellprogramm PRAKT20.TEST.COB(COBn) schreiben und hierfür ein neues JCL-Script PRAKT20.TEST.CNTL(COBSTAn) erzeugen, was sich von PRAKT20.TEST.CNTL(COB02) durch andere INFILE- und OUTFILE-Parameter unterscheidet. Letzteres resultiert in zusätzlichen Members in unseren drei Partitioned Datasets.

Aufgabe: *Verfassen Sie ein eigenes funktionsfähiges COBOL-Programm (keine Modifikation des vorgegebenen Hallo-Welt-Programms) und legen Sie den Quellkode in PRAKT20.TEST.COB(COBn) ab. Das angepaßte JCL-Script legen Sie bitte in PRAKT20.TEST.CNTL(COBSTAn) ab ("PRAKT20" ist bei beiden Datasets durch Ihre Benutzer-ID zu ersetzen). Erstellen Sie je einen Print-Screen von Ihrem ISPF-Fenster mit dem Quellkode Ihres Programms sowie von Ihrem ISPF-Fenster mit der Ausgabe Ihres COBOL-Programms. Erzeugen Sie ebenfalls einen Print-Screen von dem ISPF-Fenster, das das von Ihnen modifizierte JCL-Script enthält. Schicken Sie die drei Print-Screens im Bitmap- oder JPEG-Format (pro Bild maximal 250 KByte) an die untenstehende Mailadresse.*