Diplomarbeit

Portierung von Anwendungen auf den Mainframe-Rechner der Universität Tübingen

erstellt von
Carolin Lacher

Betreut durch: Prof. Wilhelm G. Spruth
Abgabetermin: 04. Mai 2009

,

Eberhard Karls Universität Tübingen
Fakultät für Informations und Kognitionswissenschaften

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit eigenständig und nur unter Verwendung der genannten Literatur und den aufgeführten Hilfsmitteln verfasst habe.

Tübingen, 04. Mai 2009

Ort, Datum Unterschrift

Zusammenfassung

Ziel dieser Diplomarbeit war es, Tutorials, die für das Praktikum Client Serversysteme der Universitäten Tübingen und Leipzig verwendet werden, aufzubereiten und für das neue System z der Firma IBM mit z/OS 1.8 lauffähig zu machen. Grundlage für diese Diplomarbeit waren die bereits existierenden Tutorials für den Großrechner der Universität Leipzig mit dem Betriebssystem OS/390. Die Tutorials werden für den Praktikumsbetrieb der beiden Universitäten eingesetzt und sollen den Studenten den grundlegenden Umgang mit dem System vermitteln. Dabei geht es um die Erstellung von Datasets, einfacher Programme, sowie Transaktionen und auch Datenbanken.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

An den Universitäten Tübingen und Leipzig wird seit vielen Jahren regelmäßig, im Turnus von einem Jahr, ein Praktikum zur Vorlesung Client/Server-Systeme angeboten und durchgeführt. Zu diesem Zweck existieren einige Übungsaufgaben, die den Studenten einen Einblick in die verschiedenen Bereiche der Client/Server-Systeme ermöglichen sollen.

Die Aufgaben, die in diesem Praktikum zu bearbeiten sind, behandeln Themen wie CORBA, RMI, Anwendungsentwicklung unter z/OS und TSO, CICS sowie Java Servlets. Ein wichtiger Schwerpunkt wird dabei vor allem auf die Einführung in die Grundlagen von Großrechnersystemen mit dem Betriebssystem OS/390 bzw. z/OS gelegt.

Bisher wurden die Übungen beider Universitäten auf dem Mainframe der Universität Leipzig mit dem Betriebssystem OS/390 2.7 durchgeführt. Durch die Umstellung auf das Betriebssystem z/OS 1.8 sind die bisher existierenden Tutorials zur Einführung in den Umgang mit Mainframes nicht mehr lauffähig. Um den weiteren Praktikumsbetrieb zu gewährleisten war es deshalb nötig, die Tutorials und verschiedene Konfigurationen des Mainframes so zu ändern, dass diese auf dem neuen Mainframe bearbeitet werden können.

Die Tutorials, die aus dieser Diplomarbeit hervorgehen, befinden sich alle auf der CD, die diesem Dokument beiliegt und konnten teilweise schon während der Erstellung dieser Diplomarbeit für den Praktikumsbetrieb verwendet werden. Es handelt sich hierbei um eine überarbeitete und aktualisierte Version der bereits existierenden Tutorials, die für den Mainframe der Universität Leipzig mit de Betriebssystem OS/390 erstellt wurden.

1.2 System z

Die aktuellen Großrechnersysteme der Firma IBM werden als System z bezeichnet. Die ursprüngliche Bezeichnung war zSeries. Die neuen Mainframes basieren auf einer 64 Bit-Architektur, was zur Folge hat, dass die Registergröße von 4 Bytes auf 8 Bytes erhöht wurde. Ältere Programme sind jedoch selbst ohne eine erneute Compilation weiterhin lauffähig.

Die Hardware dieser Systeme ist redundant ausgelegt, was einen Ausfall sehr unwahrscheinlich macht. Das z in System z steht für Zero Downtime. Für das Betriebssystem OS/390 wird eine Verfügbarkeitsrate von 99,999% angegeben. Dies entspricht einer Downtime von ca. 5 Minuten pro Jahr. Das Betreibssystem z/OS übersteigt diese Verfügbarkeit seines Vorgängers.

Desweiteren werden für diese Mainframes LPARs[1] verwendet. Diese Verwendung ermöglicht es, auf den Mainframes verschiedene Betriebssysteme parallel auszuführen. So läuft auf dem Mainframe der Universität Tübingen nicht nur das Betriebssystem z/OS sondern auch z/Linux.

Abbildung 1.1: z/Series Rechner

1.3 Hardware

Der an der Universität Tübingen zur Verfügung stehende Großrechner für das Praktikum ist ein IBM z9 Business Server Model S07 BC. Er ist mit 16 GByte Hauptspeicher, acht Processor Units (PUs), vier Central Processors (CPs) mit

[1] LPAR ist die gebräuchliche Abkürzung für **L**ogical **Par**tition. LPAR beschreibt die Aufteilung eines Mainframes in verschiedene virtuelle Systeme.

1786 MIPS, drei Integrated Facility for Linux (IFLs)[2] und einem SAP (System Assist Processor) ausgestattet. Außerdem besitzt er einen Processor Cache mit 256 KByte. Solche Subsysteme stellen die Laufzeitumgebung für Benutzerprogramme zur Verfügung.

Der Mainframe ermöglicht eine Aufteilung in bis zu 30 virtuelle Systeme (LPAR).

Desweiteren besitzt er ein DS 6800 Enterprise Storage Subsystem, 12 Platten mit je 146 GByte, sowie 3 RAID5 mit je 4 Platten.

Die Systembedienung findet über Consolen-Rechner statt. Hierfür wird ein 3270 - Emulator benötigt.

1.4 z/OS

Z/OS ist der Name des Betriebssystem für die System z Großrechner der Firma IBM. Z/OS ist der Nachfolger von OS/390, welches das erste Betriebssystem für die zSeries Rechner der Firma IBM war.

Das Betriebssystem z/OS beinhalten verschiedene Subsysteme. Dazu zählen das Transaktionsverarbeitung-Subsystem CICS,die TSO Shell als Entwicklungsumgebung, sowie USS (Unix System Services) als Unix kompatible Shell, die ursprünglich als OMVS bezeichnet wurde. Außerdem werden die Subsysteme WAS (WebSphere Web Application Server), JES (Job Entry Subsystem), DB2 für die Verwendung relationaler Datenbanken, das RACF-Sicherheitssystem, sowie Communications Server von z/OS bereit gestellt.

1.4.1 TSO - Time Sharing Option

TSO [TSO] ist die Dialogkomponente des z/OS. Es ist ein Kommandozeileninterpreter, der es mehreren Benutzern ermöglicht, den Rechner gleichzeitig zu verwenden.

Der ursprüngliche Einsatz von TSO fand auf Schreibmaschinenterminals statt. Später kamen Bildschirmterminals zum Einsatz. Diese IBM 3270-Terminals stellten dem Entwickler die Möglichkeit zur Verfügung, Fullscreenanwendungen zu entwickeln. TSO ist die wichtigste Komponente für Systemadministratoren und Programmentwickler auf Großrechnern unter z/OS.

TSO kommt zum Einsatz, um Daten auf den Terminals einzugeben, zu speichern und zu ändern. Es wird zur Entwicklung und zum Testen von Programmen verwendet.

[2] Spezialprozessoren für IBM Mainframes, zur Ausführung der Betriebssysteme z/VS und Linux.

1.5 JCL - Job Control Language

Durch JCL [JCL] lassen sich Jobs erstellen und zur Ausführung bringen. Diese Jobs laufen im Background ab.

Jobs beinhalten den Aufruf von einem oder mehreren Programmen, sowie die Bereitstellung der notwenigen Daten. Sie definieren administrative Dinge wie die Abrechnungsnummer und Informationen, wohin eventuell vom Programm erzeugte Listen gebracht werden sollen.

JCL ist für die Definition solcher Jobs verantwortlich.

Jobs werden durch eine Reihe von Records (Datensätzen) in dieser Sprache definiert, wobei Records immer die feste Länge 80 haben. Diese Form beruht auf der ursprünglichen Form die Daten über Lochkarte in die Maschine einzulesen, die diese Länge hatten.

1.5.1 Records

Im Folgenden werden die für die Bearbeitung der Tutorials wichtigsten Records aufgeführt und erläutert. Diese Beschreibungen finden hier anhand eines Beispiels (Abbildung 1.2 bis 1.4) aus den Tutorials statt. Die Beschreibungen zur Erklärung des JCL-Scripts orientieren sich an den Zeilennummern dieses Scipts.

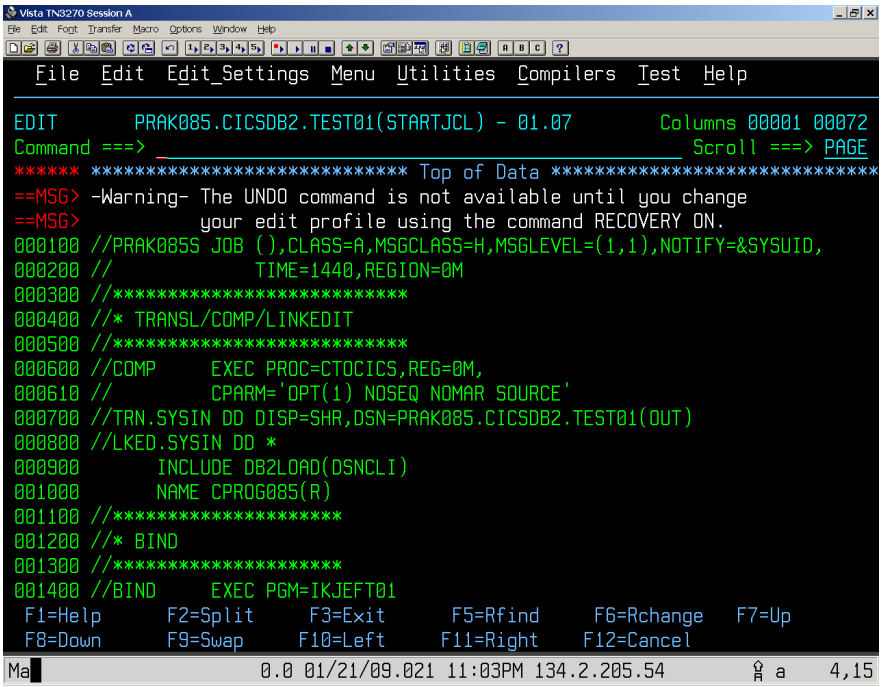

Abbildung 1.2: Beispiel für ein JCL-Script

Der JOB-Record in Zeile 000100:

Enthält unterschiedliche Informationen darüber, wie dieser JOB heißt und wie er zu behandeln ist. Dieser Job wurde mit PRAK085 bezeichnet und kann über diese Bezeichnung im Spool Search and Display Facility (SDSF) Menü gefunden werden, um dort die Meldungen anzuschauen, die bei der Übersetzung aufgetreten sind.

JOB-Parameter

CLASS in Zeile 000100: Gibt die Jobklasse „A“ an, die in der Installation definiert wurde.

MSGCLASS in Zeile 000100: Die Informationen und Meldungen, die während der Ausführung vom System erstellt werden, werden in der Output-Klasse MSGCLASS=H bereitgestellt.

MSGLEVEL in Zeile 000100: Gibt an, welche Informationen im für MSGCLASS angegebenen Output enthalten sein soll. MSGLEVEL=(1,1) das in unserem Beispiel verwendet wurde gibt dem System an, dass alle Meldungen ausgegeben werden sollen.

NOTIFY in Zeile 000100: Dieser Parameter bewirkt, dass das System eine Meldung an den hier angegebenen Benutzer bereit hält, sobald der Job beendet wurde. &SYSUID sagt dabei aus, dass dieser NOTIFY an den aktuellen TSO-Benutzer ausgegeben werden soll.

TIME in Zeile 000200: Dieser Parameter gehört noch immer zum JOB-Record. durch das Komma am Ende der ersten Zeile wird dies dem System gemeldet. Der Parameter TIME gibt die maximal erlaubte Rechenzeit für diesen Job an. Mit TIME=1440 gibt man dem System die Information, dass kein Zeitlimit existiert.

REGION in Zeile 000200: Dieser Parameter gibt an, wie groß der benötigten Speicherplatzes für das durch den Job aufgerufene Programm ist. Die Angabe von 0M hier in unserem Beispiel gibt an, dass keine Speicherbegrenzung definiert wird.

Der EXEC-Record in den Zeilen 000600, 001400 und 002700:

Jede JCL beginnt mit solch einem EXEC-Record. Dieser Record definiert, welches Programm oder welche Prozedur in diesem Job-Step ausgeführt werden soll. Jede Programmausführung bekommt hierbei einen extra Job-Step zugeordnet.Wird nur ein EXEC-Record in einem JCL-Script verwendet, so wird diesem häufig kein Name zugeordnet. Werden jedoch mehrere Job-Steps verwendet, so werden diese benannt um eine bessere Übersicht zu erhalten.

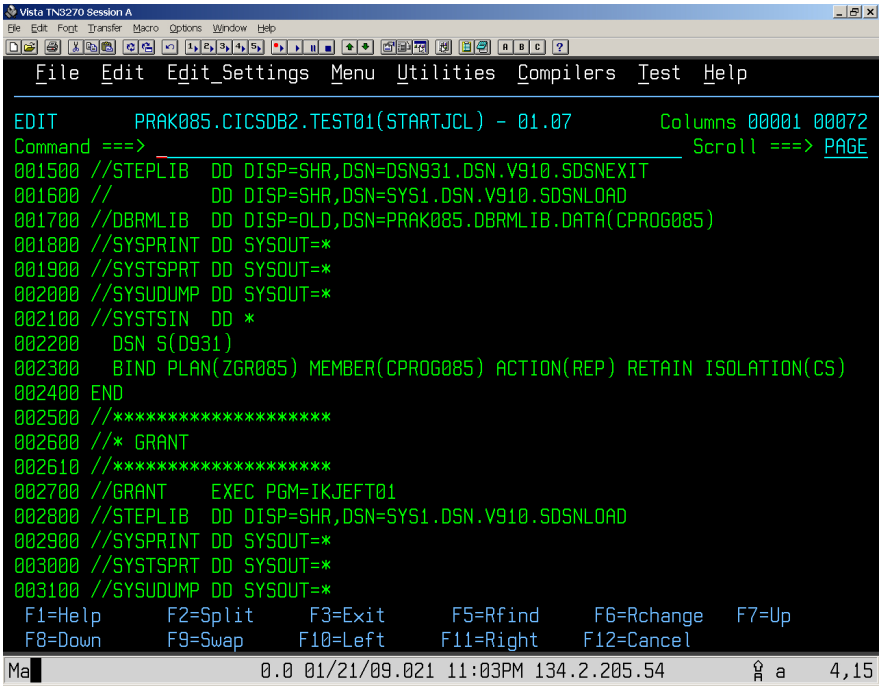

Abbildung 1.3: Beispiel für ein JCL-Script

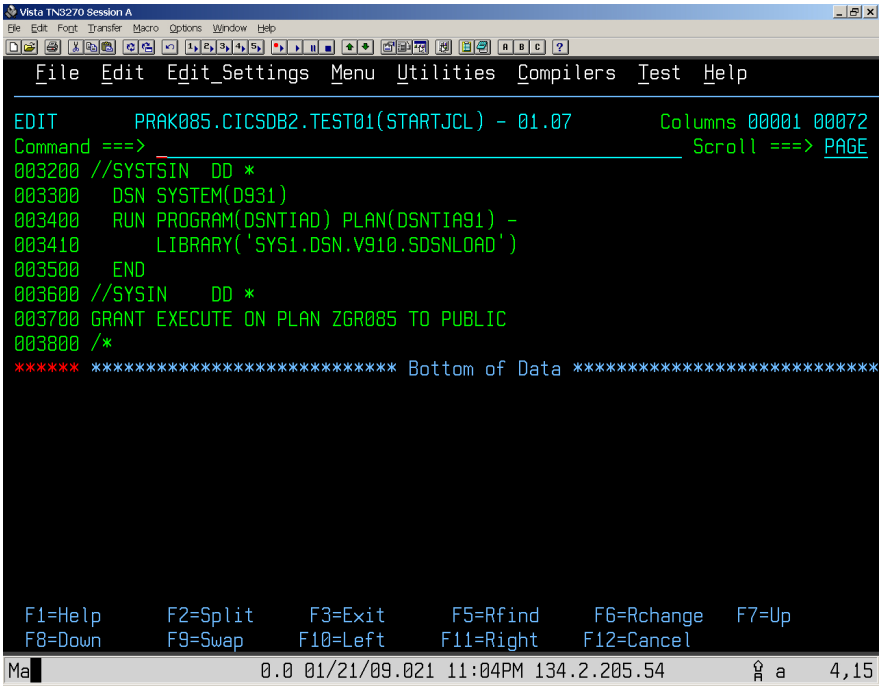

Abbildung 1.4: Beispiel für ein JCL-Script

EXEC-Parameter:

PROC: Der Parameter PROC gibt an, dass in unserem Beispiel kein Programm sondern eine JCL-Prozedur aufgerufen werden soll, nämlich der C Compiler für CICS.

PGM: Soll ein Programm ausgeführt werden, so wird dies mit dem Parameter PGM angegeben.

REG: Gibt die maximale Größe der Region an, die vom Compiler verwendet werden darf.

Der DD-Record:

DD ist ein Record, der sehr häufig vorkommt. Deshalb wird in diesem Fall auf die Angabe von Zeilennummern verzichtet. DD steht für Data Definition und legt fest, welche Daten verarbeitet werden sollen.

Der DD-Record hat das Format //ddname DD parameter

DD-Parameter:

***:** Legt fest, dass die Records in der nachfolgenden Zeile dem entsprechenden DD-Record zugeordnet werden.

DSN: Gibt den Namen des Datasets an, mit dem gearbeitet werden soll. Dieser Parameter kann auch als DSNAME vorkommen. DSN ist hierbei eine Abkürzung des eigentlichen Parameters.

DISP: Definition des Status und der Verarbeitungsweise des Datasets. Dabei muss man beachten, ob es sich um einen bereits existierenden Dataset handelt(OLD) oder ob dieser neu angelegt werden muss. SHR erlaubt dabei, dass mehrere Jobs auf diese Daten zugreifen dürfen.

SYSOUT: Definition derselben OUTPUT-Klasse wie die im JOB-Record mit MSGCLASS angegebene.

DD-Namen:

Die folgenden DD-Namen stellen eine ´Übersicht häufig verwendeter DD-Namen dar.

SYSIN: Sind die Eingabedaten für ein Programm

STEPLIB: Ist der Dataset, der das im Step aufgerufene Programm beinhaltet.

SYSTSPRT: Ist ein Terminal-Output

SYSUDUMP: Stellt Daten bereit, falls ein Fehler aufgetreten ist.

SYSPRINT: Ist die Ausgabe.

SYSTSIN: Ist die Eingabe für ein Programm.

SYSUT1: Stellt eine temporäre Datei bereit.

Kommentare:

Die Zeilen 000300 bis 000500, 001100 bis 001300, sowie die Zeilen 002500 bis 002610 sind Kommmentare.

Kommentare werden in JCL-Scipts durch //* eingeleitet und nehmen ab hier die restliche Zeile ein. Normalerweise werden jedoch Kommentare in einem JCL-Script immer in eine extra Zeile geschrieben in der kein Code steht.

1.6 Überblick über die folgenden Kapitel 2-8

Im folgenden wird die Arbeit und die Änderungen an den bereits existierenden Tutorials beschrieben, um diese für den Praktikumsbetrieb der Universität Tübingen zum Einsatz bringen zu können.

Dabei wird genau beschrieben an welchen Stellen und in welcher Art und Weise die Tutorials verändert werden mussten um einen Einsatz für das Praktikum zu sichern. Hierbei wurden nicht nur die Tutorials in der Programmiersprache C einsatzfähig gemacht - die am häufigsten für das Praktikum eingesetzt werden - sondern auch die Tutorials, die für die Programmiersprachen Cobol, PL/I und Assembler existieren.

Kapitel 2

TSO und ISPF

Das Tutorial 1 konnte nahezu ohne Änderungen aus dem Tutorial, das für das Betriebssystem OS/390 des Leipziger Mainframes geschrieben wurde, übernommen werden. Eine kleine Änderung ergab sich lediglich aus der Tatsache, dass die neuen Mainframes der beiden Universitäten neue IP-Adressen besitzen und somit der Verbindungsaufbau über diese neuen IP-Adressen statt findet.

Die IP-Adresse für den Mainframe in Tübingen ist *134.2.205.54 (hobbit.cs.uni-tuebingen.de)*.

Für den Verbindungsaufbau mit dem Mainframe in Leipzig wird die IP-Adresse *139.18.4.34 (binks.informatik.uni-leipzig.de)* benötigt.

Die weiteren Schritte des Tutorials können aus der alten Version übernommen werden.

Im nächsten Abschnitt wird die Aufgabe kurz erläutert. Die gesamte Aufgabe des Tutorial 1 ist in zwei Teile aufgeteilt, die unter

http://hobbit.cs.uni-tuebingen.de/tutor/tut01a.pdf
und
http://hobbit.cs.uni-tuebingen.de/tutor/tut01-ispf.pdf

zum downlaod bereit stehen und zusätzlich auf der beigefügten CD in den Tutorials unter den Namen tut01.doc und tut01-ispf.doc abgespeichert sind.

2.1 Aufgabe

In diesem Tutorial werden dem Praktiumsteilnehmer die Grundlagen des Betriebssystems z/OS nahe gebracht.

2.1.1 Tutorial 1a – Dateiverwaltung und Editieren von Dateien unter TSO und ISPF

Um am Praktikum teilzunehmen zu können, muss zuerst ein 3270-Emulator auf einem Arbeitsplatzrechner installiert werden, um damit auf das TSO-System und die weiteren Subsysteme zugreifen zu können. Eine kurze Einführung soll dem Benutzer den Umgang mit dem Emulator erleichtern.

2.1.2 Benutzung von ISPF

Im zweiten Teil des Tutorial 1 werden dem Teilnehmer die Grundlagen zur Nutzung von TSO bzw. ISPF erläutert und durch diese Übung nahe gebracht. Hierzu werden Datasets erstellt, die sich vom Dateiformat unter Windows oder Unix unterscheiden und in den weiteren Tutorials befüllt werden. Um alle Grundlagen für die weiteren Tutorials zu erlangen wird hier noch eine kurze Einleitung über den Umgang mit dem ISPF-Editor gegeben.

Dem User wird es durch dieses Tutorial ermöglicht, ein einfaches JCL-Script anzulegen, das in den weiteren Tutorials gebraucht wird.

Wurden all diese einleitenden Aufgaben erfüllt kann sich der User wieder ausloggen. Auch dieser Teil wird im Tutorial 1 für den neuen TSO-User genau beschrieben.

2.2 Änderungen am Tutorial

Dieses Tutorial wurde schon im Rahmen der Studienarbeit von Martin Lachmayer [Lachmayer] verändert. Hierbei wurde angegeben, dass die Tutorials unter Verwendung von hobbit.informatik.uni-tuebingen.de bzw. unter frodo.informatik.uni-tuebingen.de erstellt werden sollen. Diese Angaben stimmten jedoch nicht. Statt dieser Angaben muss die IP-Adresse 134.2.205.54 oder der Symbolische Namen hobbit.cs.uni-tuebingen.de verwendet werden, um sich auf dem Mainframe der Universität Tübingen einzuloggen.

Kapitel 3

Programme Erstellen, Kompilieren und Ausführen

3.1 Aufgabe

Das Tutorial 2 befasst sich mit dem Erstellen und Ausführen von Programmen in verschiedenen Programmiersprachen. Die verwendeten Programmiersprachen sind C, Cobol, PL/I und Assembler. Da die Anpassungen für C schon gemacht wurden lag meine Aufgabe darin, die Anpassungen für die Programmiersprache C zu überprüfen, sowie die Anpassungen für die Programmiersprachen Cobol, PL/I und Assembler vorzunehmen.

Für die Tutorials, die bisher in Leipzig unter OS/390 liefen, werden jeweils zwei Member in den Datasets angelegt. Das erste Member enthält den ausführbaren Code, das zweite beinhaltet den Job, der in der Job Control Language (Abschnitt 1.5) erstellt wird.

3.2 Für die Programmiersprache C

Das Tutorial 2 wurde, wie schon das Tutorial 1, in der Studienarbeit von Martin Lachmayer [Lachmayer] bearbeitet. Die Bearbeitung des Tutorials zeigte, dass keine weitere Veränderung mehr notwendig war und das Tutorial 2 in der Programmiersprache C wie auf der beigefügten CD im Tutorial unter dem Namen tut02cob.doc gezeigt für den Praktikumsbetrieb verwendet werden konnte.

3.3 Für die Programmiersprache Cobol

Der Versuch, das Tutorial wie gehabt zu übernehmen, führte zu folgender Fehlermeldung:

```
01.47.46 JOB03218 \$HASP165 PRAK085B ENDED AT N1 - JCL ERROR CN(INTERNAL)
```

***

Um herauszufinden, woran das vorliegende Problem liegt und um dieses zu beheben, ist es deshalb notwendig in den „Status of Jobs“ nachzusehen, welche Fehler für die Übersetzung angegeben werden. Im Falle des Cobol-Programms bekommt man hier den folgenden Hinweis:

```
IEF212I PRAK085B COBOL STEP1 STEPLIB - DATA SET NOT FOUND
IEF272I PRAK085B COBOL STEP1 - STEP WAS NOT EXECUTED.
```

Diese Meldungen weisen darauf hin, dass die hier benötigte Library nicht gefunden werden kann. Die Fehlermeldung besagt hier außerdem, dass der vorliegende Fehler im STEP1 zu suchen ist. Schaut man in das JCL-Script, das für die Übersetzung des Cobol-Programms verwendet wird, so entdeckt man, dass hier im EXEC-Statement (in Zeile `000300 //STEP1 EXEC IGYWCL`) das Member IGYWCL verwendet wird. Um dieses Problem zu lösen ist eine Anpassung der Libraries erforderlich. Diese Anpassungen werden im folgenden Abschnitt beschrieben:

3.3.1 Änderungen der Libraries

Die Library `ADCD.Z18.PROCLIB(IGYWC)`: hatte bisher in Zeile 000001 die Parameter `//IGYWC  PROC LNGPRFX='IGY.V3R3M0',SYSLBLK=3200` stehen. Dem Language Prefix Parameter (LNGPRFX) musste hier eine andere Library zugewiesen werden. Somit wurde aus der oben angegebenen Zeile

```
//IGYWC  PROC LNGPRFX='IGY340',SYSLBLK=3200.
```

Diese Änderung, die an dem Member IGYWC vorgenommen wurden müssen für alle mit Cobol in Verbindung stehenden Members dieser Library vorgenommen werden. Nachfolgend sind alle Libraries aufgeführte, die wie das Member IGYWC geändert werden müssen. Wieder werden die Parameter von LNGPRFX nach IGY340 abgeändert. Durch dieser Änderung kann die Prozedur für den Cobol-Compiler aufgerufen werden, der im Dateset IGY340.SIGYCOMP gefunden werden kann.

```
ADCD.Z18.PROCLIB(IGYWCL)
ADCD.Z18.PROCLIB(IGYWCLG)
ADCD.Z18.PROCLIB(IGYWCPG)
ADCD.Z18.PROCLIB(IGYWCPL)
ADCD.Z18.PROCLIB(IGYWCPLG)
```

Im eigentlichen Programm mussten keine Veränderungen vorgenommen werden. Die Zeile 000700 NAME COB02(R)des ursprünglichen JCLs wurde gelöscht, was jedoch nicht von Bedeutung ist.

Das durch die Veränderungen entstandene Tutorial ist auf der beigefügten CD unter dem Namen tut02cob.doc zu finden.

3.4 Für die Programmiersprache PL/I

Das Submitten des JCL-Scripts für das PL/I-Programm ergab wie schon das Cobol-Programm zuvor einen Fehler. Auch in PL/I ergab das Betrachten des „Status of Jobs“-Menüs, dass Library-Parameter falsch gesetzt waren. Für die Programmiersprache PL/I erhielt man beim Betrachten des „Status of Jobs“ die folgende Fehlermeldung:

```
4 IEF632I FORMAT ERROR IN THE REGION FIELD
```

Auch hier waren wieder Anpassungen zu machen um das Programm lauffähig zu machen. Das erste Problem lag an der Tatsache, dass einige Parameter der Libraries nicht richtig gesetzt waren und verändert werden mussten. Die Änderungen der Parameter in den Libraries wurden wie folgt durchgeführt:

3.4.1 Änderung der Parameter

Wie oben in der Fehlermeldung angegeben, war hier beim Parameter Region der Fehler zu suchen. Allerdings lag dieses Problem wie schon zuvor nicht am verwendeten JCL-Script, sondern an der verwendeten Library, die im ersten und einzigen EXEC-Statement dieses Scipts verwendet wird. Diese Library mit dem Namen IBMZCB hatte in der Zeile 000028 den Parameter REGION=512 stehen. Zeile 000041 die zum Bind dazu gehört hatte ebenfalls eine Region, nämlich REGION=2048K, Werden diese beiden Werte für die Region weg gelassen, so dass die Standardwerte verwendet werden, kann dieses Problem behoben werden. Die Änderungen die vorgenommen werden, betreffen das Member

```
ADCD.Z18.PROCLIB(IBMZCB)
```

Die Änderungen sehen dabei wie folgt aus:

```
000028 //PLI EXEC PGM=IBMZPLI,PARM='OBJECT,OPTIONS',REGION=512
```

wurde abgeändert nach

```
000028//PLI      EXEC PGM=IBMZPLI,PARM='OBJECT,OPTIONS'
000048// PARM='XREF,COMPAT=PM3',REGION=2048K
```

wurde abgeändert nach

```
000048 // PARM='XREF,COMPAT=PM3'
```

Für die korrekte Übersetzung des PL/I-Programms waren jedoch die Änderungen der Parameter noch nicht ausreichend. Weitere Fehlermeldungen besagten das Folgende:

```
IEF212I PRAK085A PLI STEP1 STEPLIB
- DATA SET NOT FOUND
IEF272I PRAK085A PLI STEP1 - STEP WAS NOT EXECUTED.
```

Diese Fehlermeldung machte wie schon beim Cobol-Tutorial die Anpassung von Libraies notwendig um das Programm lauffähig zu machen, welche im folgenden Abschnitt beschrieben werden:

3.4.2 Änderung der Libraries

Durch einen Blick in das JCL-Script sieht man, dass im STEP1 vom EXEC-Record IBMZCB aufgerufen wird. Dieses Member ist im Dataset
`ADC.Z18.PROCLIB(IBMZCB)` abgelegt und enthält in Zeile 000001
`//IBMZCB  PROC LNGPRFX='IBMZ.V3R5M0',LIBPRFX='CEE',.`

Hier musste wie schon in Abschnitt 3.3.1 der Parameter LNGPRFX angepasst werden. In diesem Fall musste der Eintrag jedoch wie folgt geändert werden:
`//IBMZCB  PROC LNGPRFX='IEL350',LIBPRFX='CEE',`

Diese Änderung musste auch für die Library `ADCD.Z18.PROCLIB(IBMZCPL)` durchgeführt werden. Für dieses Member war es außerdem nötig, die Regions zu ändern, die in den Zeilen 000029 und 000041 standen. Zeile 000029 enthielt dabei
`//PLI EXEC PGM=IBMZPLI,PARM='OBJECT,OPTIONS',REGION=512K` und Zeile 000041 enthielt `//PLKED EXEC PGM=EDCPRLK,COND=(8,LT,PLI),REGION=2048K`.

Nachdem diese Veränderungen durchgeführt waren, erreichte man, dass kein JCL-Error mehr auftrat. Allerdings konnte das Programm noch immer nicht fehlerfrei übersetzt werden. Der nun noch vorhandene Fehler lag jetzt allerdings im Programm.

Ein Problem hierbei waren die Parameter für die Region. Im JCL für das PL/I Programm musste diese auf 100M angepasst werden, da die Angabe der Region im ursprünglichen Tutorial nicht ausreichend war.

Eine weitere Fehlerquelle bei PL/I-Programmen ist, dass man stets darauf zu achten hat, dass man mit dem Programm nicht in der ersten Spalte anfängt zu programmieren. Dies ist auf den ursprünglichen Einsatz von PL/I zurück zu führen. PL/I wurde zum Einsatz mit Lochkarten entwickelt. Hier war es von Nöten, dass man in der ersten Spalte des Programms Raum für Printsteueranweisungen (z. B. eine neue Seite, Leerzeile etc.) hatte. Um dies zu erreichen, wurde die ersten Spalte für diese Befehle reserviert und man beginnt jedes PL/I-Progamm erst in der zweiten Spalte.

Das durch die Veränderungen entstandene Tutorial ist auf der beigefügten CD unter dem Namen tut02pli.doc zu finden.

3.5 In Assembler

Auch das Tutorial 2 in Assembler konnte nicht ohne Änderungen auskommen. Wieder traten beim Submit der JCL des Hello World-Programms Fehler auf. Das betrachten des „Status of Jobs “- Menüs wie schon in den anderen Programmiersprachen ergab hier die folgenden Fehler:

```
** ASMA035S Invalid delimiter - 180'Hallo
** ASMA435I Record 39 in PRAK085.TEST.ASSEM(ASS1) on volume: Z8SYS1
```

Dieses Problem wurde durch Ersetzen der folgenden Zeilen im Programm erreicht:

```
PRINTREC DC    C180'Hallo Welt, unser erstes TSP-Programm in Assembler'
```

geändert nach

```
PRINTREC DC    CL180'Hallo Welt, unser erstes TSP-Programm in Assembler*
```

Worauf bei der Erstellung eines JCLs immer geachtet werden muss, ist dass Befehle in der richtigen Spalte des Codes beginnen. Ein versehentliches Verschieben des Codes `/*)` führte zur folgenden Meldung im Status of Jobs:

```
IEW2332E 1223 CONTROL STATEMENT SYNTAX ERROR NEAR '/*'.
IEW2008I 0F03 PROCESSING COMPLETED.  RETURN CODE =  8.
```

Die Zeichenkette `/*` begann in diesem Fall in der zweiten Spalte des Editors und nicht, wie gefordert, in der ersten. Wird die Zeichenkette richtig gesetzt, so funktioniert auch dieses Programm.

Das durch die Veränderungen entstandene Tutorial ist auf der beigefügten CD unter dem Namen tut02ass.doc zu finden.

Kapitel 4

CICS

4.1 Aufgabe

Das Tutorial 3 beschäftigt sich damit, ein BMS (Basic Mapping Support) anzulegen, dass über ein in den Programmiersprachen C, Cobol, PL/I oder Assembler geschriebenes Hello World Programm in CICS ausgibt. Dafür wird das BMS mit dem SUB-Befehl übersetzt. Die Übersetzung erstellt ein Programm in der Programmiersprache, die in dem BMS angegeben wurde. Das aus der Übersetzung der BMS resultierende Programm kann anschließend in das eigentliche Programm, dessen Ausgabe unter CICS statt finden soll, eingebunden werden. Dieses Programm wird wiederum mit einem JCL-Script übersetzt. Durch das Programm, das durch die Übersetzung der angelegte BMS entstand, war es anschließend möglich, das ausgeführte Programm über das CICS-Subsystem aufzurufen.

Für das Tutorial 3 wurden auf dem alten Mainframe der Universität Leipzig die Libraries SYS1.PROCLIB(DFHMAPS) sowie CICSTS13.CICS.PRAKLAOD benötigt. Auf dem neuen Mainframe der Universität Leipzig existieren diese beiden Libraries nicht mehr. So ist das Member DFHMAPS nun unter ADCD.Z18.PROCLIB zu finden. Die zweite Library, nimmt die übersetzte Version des BMS auf und ist auf dem neuen Mainframe unter DFH310.CICS.PRAKLOAD zu finden. Um dieses Member verwenden zu können waren keine Änderungen mehr vorzunehmen. Dieses Member sowie der Dataset in dem die Ausgabe des BMS abgelegt werden soll, waren für das System bereits zugänglich.

4.1.1 Beispiel für eine BMS

Die Abbildung 4.1 zeigt die BMS für das Tutorial 3 in der Programmiersprache C. Die drei wichtigsten Befehle, die in einer BMS zu finden sind, sind DFHMSD, DFHMDI und DFHMDF.

Beschreibung der Befehle:

Die drei aufgeführten Befehle sind Assembler-Makros.

DFHMSD in den Zeilen 000500 und 001300: Der DFHMSD-Befehl in Zeile 000500 definiert die Mapset mit dem Namen MSET085, der bei der Übersetzung im Dataset PRAK085.LIB angelegt wird. Dieses Member enthält die Ausgabe, die später auf dem Bildschirm angezeigt werden soll und kann in das C-Programm eingebunden werden. LANG=C gibt dabei dem System die Programmiersprache an in der das übersetzte Member ausgegeben werden soll.
In der Zeile 001300 bedeutet DFHMSD TYPE=FINAL, das keine weiteren Maps angelegt werden sollen.

DFHMDI in Zeile 000800: Dieser Befehl definiert die einzelnen Maps, also die Screens, die benötigt werden. MAP085 ist dabei ein Label. Der DFHMDI-Befehl wird immer von der Zeile `*  MENU MAP.` eingeleitet. Die hier angegebene SIZE=(24,80) ist die Größe der Bildschirmausgabe. Ein 3270-Bildschirm besteht immer aus 24 Zeilen mit je 80 Zeichen.

DFHMDF in den Zeilen 000900 und 001100: Der letzte BMS-Befehl dient zur Definition der Felder, die unter CICS ausgegeben werden sollen. In unserem Fall werden zwei Felder definiert. Diese Definitionen geben die Position, die Länge sowie den Inhalt der Felder an.

Für Tutorial 3 waren jedoch noch einige Code-Änderungen nötig, die im Folgenden beschrieben werden.

4.2 Änderungen am C-Tutorial

Das Tutorial 3 wurde ebenfalls schon in der Studienarbeit von Martin Lachmayer [Lachmayer] bearbeitet. Beim Versuch dieses Tutorial nach der neuen Anleitung durchzuarbeiten war es mir nicht möglich, die in CICS angelegte Transaktion auszuführen. Ein Blick in das RACF-Profil für die Gruppe der PrakuserIDs gibt Aufschluss, warum dies nicht möglich war.

Das C-Tutorial wurde im Rahmen einer Studienarbeit bereits von Martin Lachmayer [Lachmayer] geändert. Die Änderung, die jedoch noch vorgenommen werden musste, war die Bezeichnung der zu verwendenden Transaktion. Bisher wurde die Transaktion mit Txxx (xxx steht hierbei für die Nummer der Praktikums-ID) bezeichnet. Hierfür haben die User jedoch keine Rechte. Die Transaktionen müssen deshalb mit Xxxx bezeichnet werden.

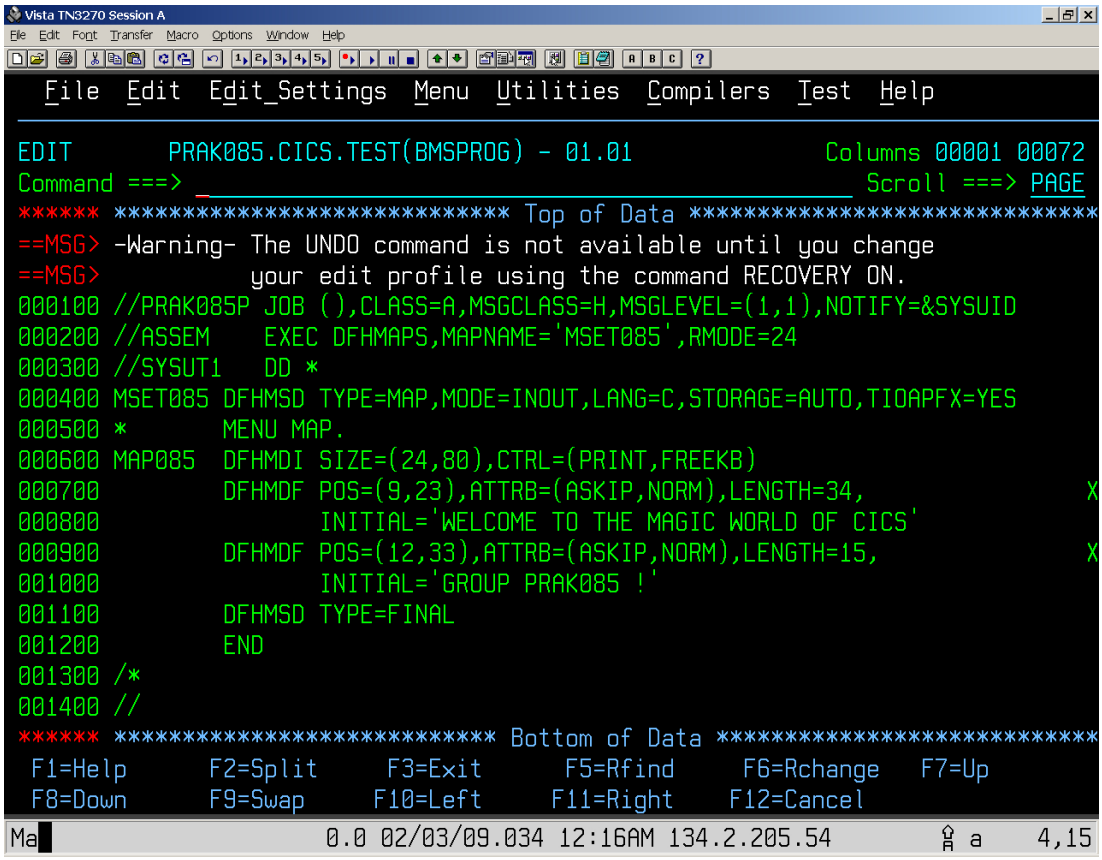

Abbildung 4.1: Der BMS-Code für das C-Programm

4.3 Änderungen am Cobol-Tutorial

4.3.1 Änderung des BMS

Für dieses Tutorial wurde in der BMS die Region weg gelassen. Dieser Parameter kann dann weg gelassen werden, wenn der systemseitig festgelegte Wert groß genug gesetzt wurde. Da dies hier der Fall war, ist eine Angabe nicht notwendig, kann jedoch trotzdem gesetzt werden.

Der Name für den DD-Record COPY.SYSUT1 wurde für das neue System anders gewählt und wird jetzt nur noch mit SYSUT1 bezeichnet. Dies sind Angaben, die man vom Systemadministrator erhält.

An die Zeile `*  MENU MAP` wurde ein Punkt angehängt. Diese Änderung ist nicht unbedingt nötig, wird aber normalerweise so geschrieben.

4.3.2 Änderung des Programms

Das eigentliche Cobol-Programm musste in dieser Aufgabe nicht verändert werden.

4.3.3 Änderung des JCL-Scripts

In diesem JCL-Script war es möglich wie schon in Abschnitt 4.3.1 erklärt wurde, die Region weg zu lassen. Diese Angabe war hier durch den installationsseitigen Standardwert ausreichend definiert.

Allerdings wurde dieses JCL-Script um eine Angabe von TIME=1440 ergänzt. Diese TIME gibt an, dass für diesen Prozess kein CPU Zeitlimit existieren soll.

Die Zeile `//STEP1  EXEC COBCICS,PARM.TRN='COBOL3'` wurde abgeändert zu `//  EXEC COBCICS,PARM.TRN='COBOL3'`. Das STEP1 konnte deshalb entfernt werden, weil die stepname-Parameter hauptsächlich der Übersichtlichkeit in Jobs dienen, falls diese aus mehreren Steps bestehen. Durch die Bezeichnung der Steps kann somit bei System-Meldungen direkt auf die Steps Bezug genommen werde. Da hier nur ein Step vorliegt, ist der Stepname nicht nötig.

Die Zeile `//COB.SYSLIB DD DSN=&SYSUID..LIB,DISP=SHR` wurde zu den zwei Zeilen

```
//COB.SYSLIB DD
//           DD DSN=&SYSUID..LIB,DISP=SHR
```

abgeändert. Der Grund für das Aufteilen der Zeile liegt darin, dass dieser DD-Record aus mehreren DD-Statements besteht. Da man für diesen DD-Record für das erste Statement die Standardparameter übernehmen möchte füllt man diese Parameter einfach nicht aus und wechselt durch Hinzufügen einer weitere Zeile in das nächste DD-Statement.

Auch für das Cobol-Tutorial musste zusätzlich die Transaktions-ID verändert werden. Hier wurde angegeben, dass der User die Transaktions-ID Y<xxx> verwenden soll. Auch dies ist jedoch durch die Einschränkung der Rechte nicht möglich und man muss die Transaktions-ID, wie schon im C-Tutorial, auf X<xxx> setzten.

Das aktualisierte Tutorial ist auf der beigefügten CD unter dem Namen Tut03cob.doc zu finden.

4.4 Änderungen am PL/I-Tutorial

Wie schon für das Tutorial 2 war auch in diesem Tutorial die Region so wie sie für den Mainframe der Universität Leipzig verwendet wurde nicht mehr groß genug. Um das Programm lauffähig zu machen musste daher die Region, im JCL-Script auf den Wert 100 M erhöht werden.

Außerdem wurde die Zeile 000300 ´mit OUTC='*',REG=100M ergänzt. Durch diese Ergänzung wird der komplette Spooloutput an MSGCLASS abgegeben. Hier also an die Klasse H. Diese Änderung bewirkt, dass man den gesamten Spooloutput an einem Stück vorliegen hat. Somit wird es möglich, im SDSF mit N für next und P für previous im Spooloutput zu navigieren.

Die letzte Änderung für dieses Tutorial besteht darin, wie schon in den anderen Tutorials aufgrund der Rechte, die Transaktions-ID von P<xxx> auf X<xxx> abzuändern.

Das aktualisierte Tutorial ist auf der beigefügten CD unter dem Namen Tut03pli.doc zu finden.

4.5 Änderungen am Assembler-Tutorial

Dieses Tutorial bedurfte nur der Änderungen der Transaktions-ID von V<xxx> nach X<xxx>.

Das aktualisierte Tutorial ist auf der beigefügten CD unter dem Namen Tut03cob.doc zu finden.

Für eine eindeutige Unterscheidung der Tutorials wurde jeweils der Ausgabetext in CICS auf die verwendete Sprache angepasst, so dass sofort ersichtlich ist, um welches Tutorial in welcher Sprache es sich handelt.

Kapitel 5

DB2

5.1 Aufgabe

Das Tutorial 4 befasst sich mit der Erstellung einer Datenbank unter z/OS. Dieses Tutorial wurde ebenfalls von Martin Lachmayer in seiner Studienarbeit [Lachmayer] bearbeitet.

Um auf dem Mainframe eine Datenbank anzulegen wird zuerst eine Storage Group benötigt, die die eigentliche Datenbank aufnimmt. Diese in der Storage Group eingerichtete Datenbank kann nun den Tablespace für die Tabellen sowie die Tabellen selbst aufnehmen.

5.2 Änderungen des Tutorials

Da auf dem neuen Mainframe nicht mehr die gleichen Rechte für die Praktikums-Teilnehmer gelten wie auf dem alten Mainframe der Universität Leipzig können für das Tutorial 4 nicht mehr alle Schritte vom Teilnehmer selbst durchgeführt werden. Durch die Einschränkung der Rechte ist es dem Praktikums-Teilnehmer weder erlaubt, eine Storage Group anzulegen, noch die Datenbank selbst. Aus diesem Grund wird beim Erstellen des Prak-Accounts beim Ausführen des dafür verwendeten JCL-Scipts direkt eine Datenbank für den entsprechenden Account angelegt, die denselben Namen erhält, wie der Account selbst.

Beim Durcharbeiten dieses Tutorials trat nun jedoch das Problem auf, dass die Storage Group, die schon vorhanden sein sollte, nicht existierte. Dies lag daran, dass im Tutorial die Angabe gemacht wurde, DB2 V8 zu verwenden. Da jedoch auf dem Mainframe die aktuellere Version V9 installiert ist, wurden vom Systemadministrator die Anpassungen für diese Version vorgenommen. Dies hat zur Folge, dass im DB2I DEFAULTS-Menü (erreichbar über die Eingabe von *m.11.d* auf der Kommandozeile) für den DB2-Namen *D931* eingestellt werden muss. Die restlichen Werte konnten wie schon im alten Tutorial gesetzt werden.

Da Herr Lachmayers Account über mehr Rechte verfügte als die Prak-User-Accounts und er somit seine eigene Datenbank anlegen konnte, ergab dies noch eine weitere Änderung. Diese liegt im Namen der zu verwendenden Storage Group sowie der zu verwendenden Datenbank. Im Tutorial wird STOGR060 (060 für die Prak-Account-Nummer) für die Storage Group, sowie für die Datenbank DB1 angegeben. Allerdings existiert weder diese Storage Group noch die Datenbank und der User hat nicht die Rechte diese selbst anzulegen. Hier hilft nun ein Blick in das unten stehende JCL-File zur Erstellung der Prak-Accounts weiter um Klarheit zu verschaffen. Das JCL-Script zeigt ab der Zeile 000056, dass beim Erstellen des Accounts für jeden Benutzer vom Systemadministrator automatisch eine Datenbank angelegt wird, die vom User verwendet und gefüllt werden kann.

```
000001 //ADDUSRP    JOB (ACCT),ADCD,COND=(0,NE),
000002 //            CLASS=A,MSGCLASS=H,NOTIFY=\&SYSUID
000003 //* add non-authorisierten User\\
000004 //* 1. change zur gewünschten nummer des Praktikanten
000005 //* Command ===> c #n         1      all
000006 //* 2. submit den job
000007 //* Command ===> sub
000008 //* 3. cancel um keine Veränderungen vorzunehmen
000009 //* Command ===> cancel
000010 //*
000011 //TMP    EXEC PGM=IKJEFT01
000012 //SYSTSPRT DD SYSOUT=*
000013 //SYSLBC   DD DISP=SHR,DSN=SYS1.BRODCAST
000014 //SYSTSIN  DD *
000015  ADDUSER                          +
000016   PRAK#n                          +
000017   NAME('Praktikum Userid#n')      +
000018   DFLTGRP(PRAKT)                  +
000019   PASSWORD(SCHNEE)                +
000020   TSO(                            +
000021     ACCTNUM(ACCT#n)               +
000022     PROC(DBSPROC)                 +
000023     MAXSIZE(6072)                 +
000024     SIZE(5000)                    +
000025     MSGCLASS(H)                   +
000026     UNIT(SYSALLDA))               +
000027   OMVS(UID(100#n)                 +
000028     HOME('/u/prak#n')             +
000029     PROGRAM('/bin/sh'))
000030  ADDSD 'PRAK#n.*' UACC(NONE) GENERIC OWNER(ADMIN)
000031  PERMIT 'PRAK#n.*' ACCESS(ALTER) ID(ADMIN)
```

```
000032 /*
000033 //*
000034 //DEFCAT   EXEC PGM=IDCAMS
000035 //*
000036 //SYSPRINT DD SYSOUT=*
000037 //SYSIN    DD *
000038   DEF ALIAS (NAME('PRAK#n') RELATE('USERCAT.Z18.USER') )
000039 /*
000040 //MKDIR   EXEC PGM=IKJEFT01
000041 //*
000042 //STDOUT   DD PATH='/tmp/stdout',
000043 //            PATHOPTS=(OCREAT,OTRUNC,OWRONLY),
000044 //            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
000045 //JES      DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
000046 //SYSTSPRT  DD SYSOUT=*
000047 //SYSTSIN   DD *
000048  BPXBATCH SH mkdir /u/prak#n +
000049    >/tmp/stdout 2>&1;chown prak#n /u/prak#n +
000050    >>/tmp/stdout 2>&1
000051  BPXBATCH SH chmod 700 /u/prak#n +
000052    >>/tmp/stdout 2>&1
000053  OCOPY INDD(STDOUT) OUTDD(JES)   TEXT PATHOPTS(OVERRIDE)
000054  BPXBATCH SH rm /tmp/stdout
000055 /*
000056 //DSNTEP2  EXEC PGM=IKJEFT01,DYNAMNBR=20
000057 //STEPLIB  DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
000058 //DBRMLIB  DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNSAMP
000059 //SYSTSPRT DD SYSOUT=*
000060 //SYSPRINT DD SYSOUT=*
000061 //SYSUDUMP DD SYSOUT=*
000062 //SYSTSIN  DD *
000063    DSN SYSTEM(D931)
000064    RUN PROGRAM(DSNTIAD)  PLAN(DSNTIA91) PARM('RC0')
000065    END
000066 /*
000067 //*
000068 //SYSIN    DD *
000069   DROP DATABASE PRAK#n;
000070 /*
000071 //*
000072 //*
000073 //DSNTEP2  EXEC PGM=IKJEFT01,DYNAMNBR=20
000074 //STEPLIB  DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
```

```
000075 //DBRMLIB  DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNSAMP
000076 //SYSTSPRT DD SYSOUT=*
000077 //SYSPRINT DD SYSOUT=*
000078 //SYSUDUMP DD SYSOUT=*
000079 //SYSTSIN  DD *
000080    DSN SYSTEM(D931)
000081    RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP91)
000082    END
000083 //*\\
000084 //SYSIN    DD *
000085   CREATE DATABASE PRAK#n;
000086   GRANT DBADM ON DATABASE PRAK#n TO PRAK#n
000087 //*
```

Die Storage Group, die für die Praktikumsteilnehmer angelegt wurde ist die SYSDEFLT. Diese steht allen Praktikumsteilnehmern zur Verfügungen und beinhaltet die jeweiligen Datenbanken, die beim Einrichten der Accounts angelegt werden.

Werden diese Änderungen berücksichtigt kann der User seine eigene Datenbank füllen und hat für diese Datenbank sogar das Recht sie zu administrieren. Durch `GRANT DBADM ON DATABASE PRAK#n TO PRAK#n` werden alle Rechte, außer solche die sicherheitsrelevant sind, an den Prak-User übergeben.

Das durch die oben genannten Veränderungen entstandene Tutorial ist auf der beigefügten CD unter dem Namen tut04.doc zu finden.

Kapitel 6

Datenbankzugriff mit CICS

6.1 Aufgabe

In diesem Tutorial sollen die Praktikumsteilnehmer die in Tutorial 4 angelegte Datenbank über CICS aufrufen und ausgeben. Dafür wird ein sog. BMS-Programm angelegt, das für die Presentation-Logic zuständig ist. Weiter wird ein Programm jeweils in den Sprachen C, Cobol, Assembler und PL/I geschrieben, das für die Business-Logic verantwortlich ist. Der letzte Member, der für dieses Tutorial angelegt werden muss ist ein JCL-Script, das für Bind, Link und Grand verantwortlich ist.

6.2 Für die Programmiersprache C

Dieses Tutorial wurde wieder in der Programmiersprache C bereits von Martin Lachmayer [Lachmayer] berarbeiten. Leider traten auch hier erneut Probleme bei der Bearbeitung auf.

Beim Versuch die angelegten Codes BMS, das SQL-Precompiler-Scipt und die JCL zu übersetzten führte dies zur folgenden Meldung beim Übersetzten:

```
READY
  DSN S(D931)
DSN
  BIND PLAN(ZGR085) MEMBER(CPROG085) ACTION(REP) RETAIN ISOLATION(CS)
DSNT210I  -D931 BIND AUTHORIZATION ERROR
           USING PRAK085 AUTHORITY
           PLAN = ZGR085
           PRIVILEGE = BIND
DSNT201I  -D931  BIND FOR PLAN ZGR085   NOT SUCCESSFUL
DSN
END
READY
END
```

Dieses Problem lag daran, dass bereits ein PLAN mit diesem Namen existierte. Durch Löschen dieses PLAN war es anschließend möglich, den Code fehlerfrei zu übersetzen.

Sobald die Übersetzung erfolgreich vorgenommen wurde kann man die Transaktion unter dem Subsystem CICS, wie auf der beigefügten CD im Tutorial unter dem Namen tut05c.doc beschrieben, installieren und ausführen.

6.3 Probleme unter CICS

Leider trat dabei erneut ein Problem auf. Beim Versuch, die installierte Transaktion auszuführen, wurde die folgende Fehlermeldung ausgegeben:

```
DFHAC2206 15:31:49 CICS Transaction X085 failed with abend AEY9. Updates to
 local recoverable resources backed out.
```

Der Fehler lag daran, dass auf hobbit.cs.uni-tuebingen.de bisher das DB2CONN noch nicht installiert war, da die bisherige Anpassung der Tutorials von Herrn Lachmayer nicht auf hobbit.cs.uni-tuebingen.de vorgenommen wurde, sondern auf frodo.informatik.uni-tuebingen.de. DB2CONN ist ein Makro, das eine ständige Verbindung zwischen DB2 und der CICS-DB2-Attachmetn-Fasility herstellt.

Um das Tutorial lauffähig zu machen musste daher zuerst das DB2CONN installiert werden. Dazu war es erforderlich im Member `DFH310.SYSIN(DFH$SIP1)` folgende Zeilen hinzuzufügen:

```
DB2CONN=YES,
XUSER=NO,
```

Außer diesen Konfigurationsänderungen musste noch folgenden Job submittet werden:

```
//CICSDB2  JOB (FB3TR,60000,4365),'SVTSC.CBIPO.89C',
//             NOTIFY=&SYSUID,
//             CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1)
//*
//INITCSD  EXEC PGM=DFHCSDUP,REGION=1M
//STEPLIB  DD DSN=DFH310.CICS.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=DFH310.CICS.DFHCSD,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(100,100))
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
ADD GROUP(DB2GROUP) LIST(XYZLIST)
 DEFINE DB2CONN(RCT00) GROUP(DB2GROUP)
 DESCRIPTION(DB2CONN MIGRATED FROM LOAD MODULE DFHRCT00)
        CONNECTERROR(SQLCODE) DB2ID(D731) MSGQUEUE1(CSMT) NONTERMREL(NO)
        PURGECYCLE(0,30) RESYNCMEMBER(YES) STANDBYMODE(RECONNECT)
        STATSQUEUE(CSSL) TCBLIMIT(12) THREADERROR(ABEND)
        ACCOUNTREC(TXID) AUTHTYPE(SIGN) DROLLBACK(YES) PLAN(DSN8CC0)
```

```
        PRIORITY(HIGH) THREADLIMIT(3) THREADWAIT(YES)
        COMAUTHTYPE(USERID) COMTHREADLIM(1)
 DEFINE DB2ENTRY(D8CS) GROUP(DB2GROUP)
 DESCRIPTION(DB2ENTRY MIGRATED FROM LOAD MODULE DFHRCT00)
        ACCOUNTREC(TXID) AUTHTYPE(SIGN) DROLLBACK(YES) PLAN(DSN8CC0)
        PRIORITY(HIGH) PROTECTNUM(0) THREADLIMIT(1) THREADWAIT(YES)
 DEFINE DB2ENTRY(D8PP) GROUP(DB2GROUP)
 DESCRIPTION(DB2ENTRY MIGRATED FROM LOAD MODULE DFHRCT00)
        ACCOUNTREC(TXID) AUTHTYPE(SIGN) DROLLBACK(YES) PLAN(DSN8CQ0)
        PRIORITY(HIGH) PROTECTNUM(0) THREADLIMIT(1) THREADWAIT(YES)
DEFINE DB2ENTRY(D8PS) GROUP(DB2GROUP)
 DESCRIPTION(DB2ENTRY MIGRATED FROM LOAD MODULE DFHRCT00)
        ACCOUNTREC(TXID) AUTHTYPE(SIGN) DROLLBACK(YES) PLAN(DSN8CP0)
        PRIORITY(HIGH) PROTECTNUM(0) THREADLIMIT(1) THREADWAIT(YES)
 DEFINE DB2ENTRY(D8PT) GROUP(DB2GROUP)
 DESCRIPTION(DB2ENTRY MIGRATED FROM LOAD MODULE DFHRCT00)
        ACCOUNTREC(TXID) AUTHTYPE(SIGN) DROLLBACK(YES) PLAN(DSN8CH0)
        PRIORITY(HIGH) PROTECTNUM(0) THREADLIMIT(1) THREADWAIT(YES)
 DEFINE DB2ENTRY(D8PU) GROUP(DB2GROUP)
 DESCRIPTION(DB2ENTRY MIGRATED FROM LOAD MODULE DFHRCT00)
        ACCOUNTREC(TXID) AUTHTYPE(SIGN) DROLLBACK(YES) PLAN(DSN8CH0)
        PRIORITY(HIGH) PROTECTNUM(0) THREADLIMIT(1) THREADWAIT(YES)
 DEFINE DB2TRAN(D8CS) GROUP(DB2GROUP)
 DESCRIPTION(DB2TRAN MIGRATED FROM LOAD MODULE DFHRCT00)
        ENTRY(D8CS) TRANSID(D8CS)
 DEFINE DB2TRAN(D8PP) GROUP(DB2GROUP)
 DESCRIPTION(DB2TRAN MIGRATED FROM LOAD MODULE DFHRCT00)
        ENTRY(D8PP) TRANSID(D8PP)
 DEFINE DB2TRAN(D8PS) GROUP(DB2GROUP)
 DESCRIPTION(DB2TRAN MIGRATED FROM LOAD MODULE DFHRCT00)
        ENTRY(D8PS) TRANSID(D8PS)
 DEFINE DB2TRAN(D8PT) GROUP(DB2GROUP)
 DESCRIPTION(DB2TRAN MIGRATED FROM LOAD MODULE DFHRCT00)
        ENTRY(D8PT) TRANSID(D8PT)
 DEFINE DB2TRAN(D8PU) GROUP(DB2GROUP)
 DESCRIPTION(DB2TRAN MIGRATED FROM LOAD MODULE DFHRCT00)
        ENTRY(D8PU) TRANSID(D8PU)
/*
```

Dieser Job definiert DB2CONN, DB2ENTRY, sowie DB2TRAN. DB2ENTRY ist dabei für die Reservierung von Threads zur individuellen Verbindung mit DB2 für die Transaktionen zuständig. DB2TRAN erlaubt es, die Transaktionen mit einem bestimmten DB2ENTRY zu verbinden.

Nach diesen Anpassungen war es nun noch nötig einige Einstellungen unter CICS vorzunehmen. Hierzu wurde unter CICS der Befehl `CEDA EXP GR(DB2GRUP)` eingegeben. Diese Eingabe führte zum Menü, das in Abbildung 6.1 dargestellt ist. Der Eintrag alter hinter der ersten Zeile dieser Auflistung führte danach zu dem Bildschirm, der in Abbildung 6.2 gezeigt ist. In dieser Abbildung musste der Wert für DB2Id, wie abgebildet, angepasst werden. Auf der zweiten Seite dieses

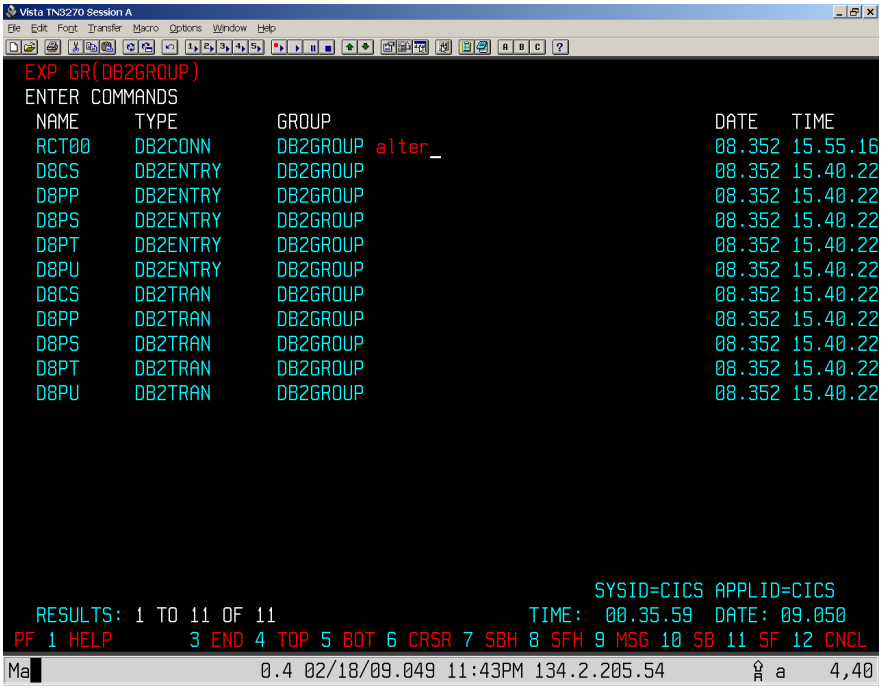

Abbildung 6.1: Aufruf von Alter

Menüs musste nun noch der Eintrag für PLAN, wie in Abbildung 6.3 gezeigt, abgeändert werden. Anschließend kann man das Menü mit der Taste F3 wieder verlassen.

Versuchte man nach all diesen Änderungen nun die Transaktion zu starten, bekam man jedoch noch eine Fehlermeldung:

```
DFHAC2206 20:35:43 CICS Transaction X085 failed with abend AD2S. Updates to
 local recoverable resources backed out.
```

Dieser Fehler lag nun noch daran, dass die UserID CICS im RACF noch nicht definiert wurde. Um diese Definition vorzunehmen muss das im folgenen dargestellte JCL-Script submittet werden:

```
//STEP1   EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 ADDUSER                          +
  CICS                            +
  NAME('cics       ')       +
  DFLTGRP(SYS1)                   +
  PASSWORD(UNI4YOU)              +
  OMVS(UID(20090121)              +
    HOME('/u/cics')              +
    PROGRAM('/bin/sh'))
 ADDSD 'WGS.*' GENERIC UACC(NONE)
 CO WGS GROUP(SYSCTLG) OWNER(SYS1)
```

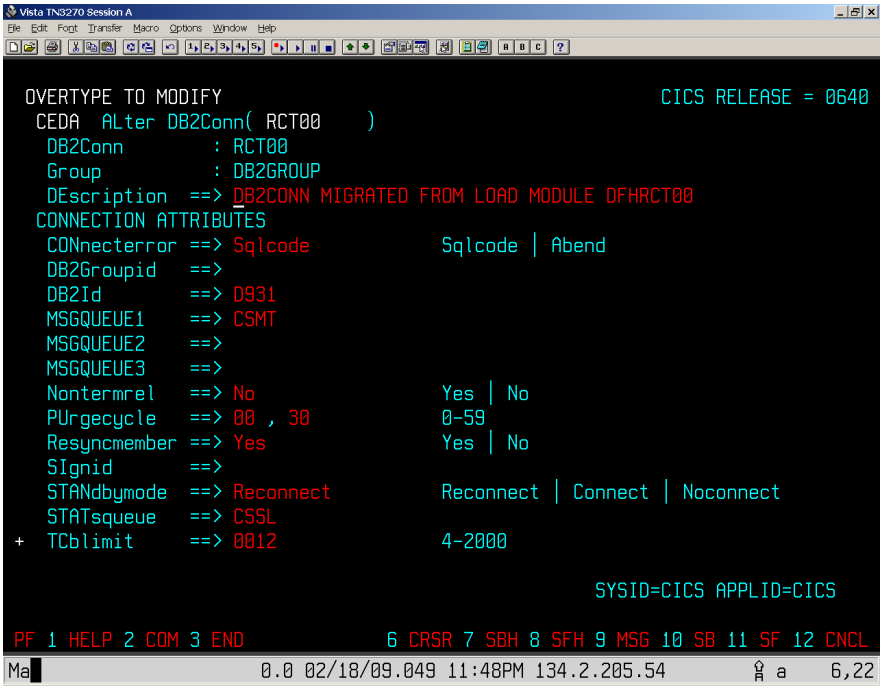

Abbildung 6.2: Änderung von DB2Id

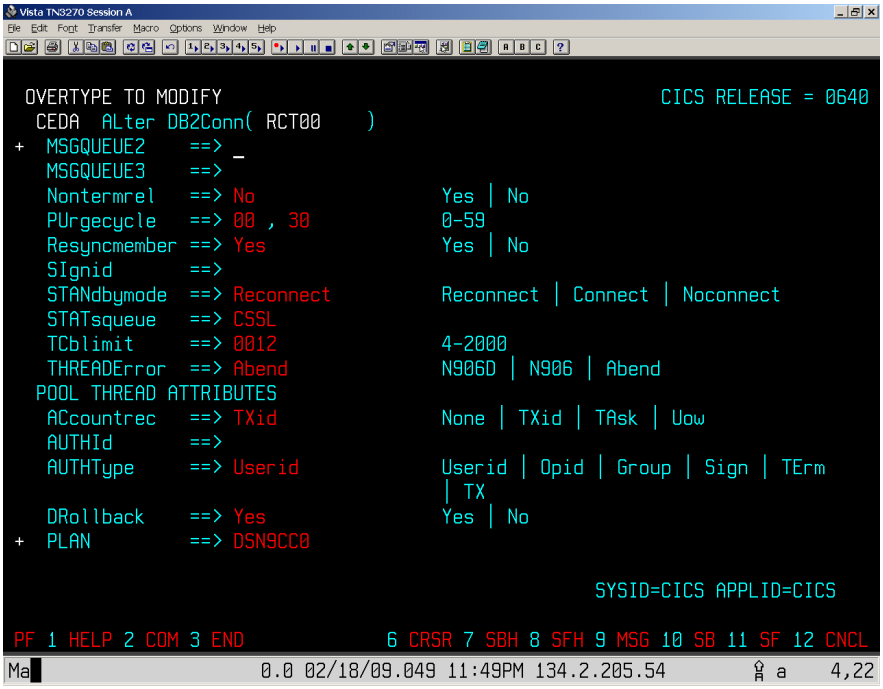

Abbildung 6.3: Änderung von PLAN

```
 CO WGS GROUP(VSAMDSET) OWNER(SYS1)
 CO WGS GROUP(ADMIN) OWNER(SYS1)
 PERMIT 'WGS.*' ACCESS(ALTER) ID(WGS)
//*
//MKDIR   EXEC PGM=IKJEFT01
//*
//STDOUT   DD PATH='/tmp/stdout',
//            PATHOPTS=(OCREAT,OTRUNC,OWRONLY),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//JES      DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
 BPXBATCH SH mkdir /u/cics +
   >/tmp/stdout 2>&1;chown cics /u/cics +
   >>/tmp/stdout 2>&1
 OCOPY INDD(STDOUT) OUTDD(JES)   TEXT PATHOPTS(OVERRIDE)
 BPXBATCH SH rm /tmp/stdout
/*
```

Nach all diesen Änderungen war es nun möglich, die Transaktion zu installieren und auszuführen.

Das aktualisierte Tutorial ist unter dem Namen tut05c.doc auf der beigefügten CD zu finden.

6.4 Für die Programmiersprachen PL/I

Versucht man, die PL/I-Anwendung so auszuführen wie dies das alte Tutorial beschreibt, so kann man das BMS sowie den SQL-Precompiler-Aufruf ohne Fehler ausführen. Bei der Übersetzung des JCL-Scripts tritt jedoch ein Fehler auf und das Programm wird nicht korrekt übersetzt. Ein Blick in den Output bringt folgenden Fehler:

```
26 XXSYSLIB   DD DSN=&INDEX..SDFHLOAD,DISP=SHR
   IEFC653I SUBSTITUTION JCL - DSN=DFH310.CICS.SDFHLOAD,DISP=SHR
27 XX         DD DSN=&LE370HLQ..SCEELKED,DISP=SHR
   IEFC653I SUBSTITUTION JCL - DSN=CEE.SCEELKED,DISP=SHR
28 XX         DD DSN=&DB2V..SDSNLOAD,DISP=SHR          (###)
   IEFC653I SUBSTITUTION JCL - DSN=DSN810.SDSNLOAD,DISP=SHR
```

Hier fällt auf, dass im Output von PLISTA85 im Step „SYSLIB“ im dritten DD-Statement des ersten Steps auf die DB2 Version V8 zugreift. Dies kann an der angegebenen Library für den DSN erkannt werden. Im zweiten Step wurde bei der Eingabe des Steps schon die richtige Library für die Verwendung von DB2 Version V9 gesetzt, da hier bei der Eingabe der Daten schon zu erkennen war, dass eine falsche DB2 Version verwendet wurde. Auf dem System laufen die beiden Versionen V8 und V9. Hier fiel die Entscheidung auf V9, da dieses schon für das Tutorial 4 zur Erstellung der Datenbank eingesetzt wurde. Durch

die Verwendung der DB2 Version V8 im ersten Step des JCL-Scipts kommt es dadruch zu einer Vermischung der Versionen, was nicht zulässig ist.

Das Problem wird behoben, indem man in den COMP-Stept für TRANSLATE, COMPILE und LINKEDIT vor den DD-Record `000900 //LKED.SYSIN DD *` die folgenden Zeilen einfügt:

```
000810 //LKED.SYSLIB DD
000820 //           DD
000830 //           DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
```

Schaut man sich nun noch den Output von PLIDB205 (dem SQL-Precompile-File) an, so erkennt man, dass auch im DD-Record DBRMLIB „DSN810.SDSNLOAD“ verwendet wird. PLISTA05 benutzt inzwischen jedoch „SYS1.DSN.V910.SDSNLOAD“ sowie das DB2 System D931. Die Veränderung des Members PLIDB205 von

```
000100 //PRAK085P JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //             REGION=4M
000300 //PCOMP EXEC DB2PLI
000400 //DBRMLIB  DD  DSN=&SYSUID..DBRMLIB.DATA(PLI05),DISP=SHR
000500 //SYSCIN   DD  DSN=&SYSUID..CICSDB2.PLI(PLI05OUT),DISP=SHR
000600 //SYSIN    DD  DSN=&SYSUID..CICSDB2.PLI(PLI05),DISP=SHR
000700 //SYSLIB   DD  DSN=&SYSUID..CICSDB2.PLI,DISP=SHR
```

nach

```
000100 //PRAK085P JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //             REGION=4M
000300 //PCOMP EXEC DB2PLI
000310 //PC.STEPLIB  DD  DISP=SHR,DSN=SYS1.DSN.V910.SDSNEXIT
000320 //         DD  DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
000400 //DBRMLIB  DD  DSN=&SYSUID..DBRMLIB.DATA(PLI05),DISP=SHR
000500 //SYSCIN   DD  DSN=&SYSUID..CICSDB2.PLI(PLI05OUT),DISP=SHR
000600 //SYSIN    DD  DSN=&SYSUID..CICSDB2.PLI(PLI05),DISP=SHR
000700 //SYSLIB   DD  DSN=&SYSUID..CICSDB2.PLI,DISP=SHR
```

durch Ergänzen der Zeilen 000310 und 000320 bewirkt, dass auch dieses Member die DB2 Version V9 verwendet und das Tutorial in PL/I ist lauffähig.

Für dieses Tutorial musste die zusätzlich die Transaktions-ID von Z<xxx> nach X<xxx> angepasst werden.

Das aktualisierte Tutorial ist unter dem Namen tut05pli.doc auf der beigefügten CD zu finden.

Kapitel 7

REXX

7.1 Aufgabe

Das Tutorial 6 gibt dem Teilnehmer des Praktikums den Einblick in eine Programmiersprache, die im bisherigen Praktikumsbetrieb noch nicht verwendet wurde. Es handelt sich hierbei um ein Hello World-Programm in der Programmiersprache *REXX*. Das Ausführen des *REXX*-Programms hängt nicht von einem speziell dafür geschriebenen JCL-Script zur Ausführung im Batch-Betrieb ab. Das Tutorial 6 soll dem User eine andere Art der Programmausführung auf dem *System z* näher bringen. Das hier erstellte REXX-Programm wird direkt über die Kommandozeile des ISPF-Subsystems aufgerufen.

7.2 Ergebnis

Bei diesem Tutorial hat sich bei der Bearbeitung ergeben, dass *REXX* auf dem neuen Mainframe wie schon auf OS/390 lauffähig ist. Für die Ausführung von *REXX*-Programmen wird in diesem Tutorial, anders als bei der Erstellung von C-, Cobol-, Assembler- und PL/I-Programmen, auf die Verwendung eines JCL-Scripts verzichtet. Dieses Programm wird also nicht im Background über einen Job gestartet, sonder direkt im Foreground.

Um das Tutorial dennoch auf den neuesten Stand zu bringen wurden einige Erklärungen ergänzt und erweitert, um den Studenten eine genauere Einführung zu ermöglichen.

Das REXX-Programm, das in diesem Tutorial lauffähig gemacht werden soll, ist zum einen ein einfaches Hello World Programm und zum zweiten ein Programm, das eine Eingabe vom Benutzer erwartet. Hier ist darauf zu achten, dass bei der Ausgabe von *** durch das System von diesem die Betätigung der Eingabetaste erwartet wird, bevor eine erneute Eingabe von Daten durch den Benutzer erfolgen kann.

Das aktualiesierte Tutorial 6 befindet sich auf der beigefügten CD unter dem Namen tut06.doc.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

Nach der Einleitung in der die Grundlagen der IBM Mainframes, die für diese Arbeit verwendet wurden, vorgestellt wurden beschäftigte sich die restliche Arbeit damit, die Programme und Tutorials auf dem Mainframe der Universität Tübingen lauffähig zu machen. Dabei wurde es durch diese Diplomarbeit ermöglicht, das Praktikum Client/Server Systeme schon im letzten Semester nahezu vollständig auf dem neuen Mainframe durchzuführen. Nur das Tutorial 5 war zum Zeitpunkt des Praktikums noch nicht für den Praktikumsbetrieb verwendbar und so mussten Tutorial 4 und Tutorial 5, das auf dem vierten Tutorial aufbaut wie gewohnt in Leipzig gemacht werden.

Ab diesem Semester hat sich nun der Turnus für das Client/Server Praktikum an der Universität Tübingen geändert und findet im Sommersemester statt. Für dieses Semester wird es nun möglich sein, alle Tutorials auf dem neuen Mainframe in Tübingen durchzuführen.

Bisher wurden für das Praktikum hauptsächlich die Tutorials in der Programmiersprache C verwendet. Durch diese Diplomarbeit wird es den Studenten jedoch ermöglicht das Praktikum auch in den Programmiersprachen Cobol, PL/I oder Assembler zu machen.

8.2 Ausblick

Im Rahmen dieser Arbeit wäre es möglich, weitere Tutorials, die bisher noch nicht auf dem neuen Mainframe lauffähig sind weiterzuentwickeln und lauffähig zu machen. Außerdem wären auch neue Tutorials für den Praktikumsbetrieb Client/Server-Systeme denkbar.

Aufgrund der heutzutage noch häufig zum Einsatz kommenden Programme in den Programmiersprachen Cobol, PL/I und Assembler wäre es in diesem Zusammenhang auch Denkbar einige neue Tutorials in diesen Programmiersprachen

zu entwickeln, um den Studenten so einen kleinen Einblick in diese alten, aber dennoch gebräuchlichen Sprachen zu ermöglichen.

Anhang A

Inhalt der beigefügten CD

A.1 Tutorials

tut01.doc: Das Tutorial 1 beschreibt die Installation und die Verwendung des 3270-Emulators.

tut01-ispf: Der zweite Teil des Tutorial 1 beschreibt den Umgang mit ISPF und TSO. Sowie das Anlegen von Datasets und Members.

tut02ass, tut02c, tut02cob und tut02pli: Das Tutorial 2 befasst sich in den Programmiersprachen Assembler, C, Cobol und PL/I mit dem Erstellen und Ausführen eines Programms durch ein JCL-Scipt.

tut03ass, tut03c, tut03cob und tut03pli: Das Tutorial 3 befasst sich mit der Erstellung einer CICS-Anwendung.

tut04: Im Tutorial 4 wird die Erstellung und Verwendung einer Datenbank mittels DB2 behandelt.

tut05ass, tut05c, tut05cob und tut05pli: Das Tutorial 5 behandelt die Ausgabe der in Tutorial 4 erstellten Datenbank mittels CICS.

tut06: Im Tutorial 6 wird ein einfaches REXX-Programm erstellt. Diese wird, im Gegensatz zu den bisher erstellten Programmen, ohne ein JCL-Script ausgeführt

A.2 Elektronisch verfügbare Referenzen

Bln01kq.pdf, Bln02kq.pdf, rechner.html und paul6.pdf: Diese Dateien beinhalten die elektronischen Referenzen, die für diese Arbeit verwendet wurden.

Literaturverzeichnis

[Lachmayer] Martin Lachmayer Datum Titel der Studienarbeit

[z/Series01] Prof. Dr. Ing. Wilhelm G. Spruth (2. Juli 2007) Einführung in das System z Mainframe, *http://www.informatik.uni-leipzig.de/cs/andrVorl/Berlin/Bln01kq.pdf*

[z/Series02] Prof. Dr.-Ing. Wilhelm G. Spruth (2. Juli 2007) Einführung in das System z Mainframe, *http://www.informatik.uni-leipzig.de/cs/andrVorl/Berlin/Bln02kq.pdf*

[hobbit.cs] *http://hobbit.cs.uni-tuebingen.de/rechner/rechner.html*

[TSO] M.Teuffel (2002) TSO Time Sharing Option im Betriebsystem z/OS MVS *Das ausführliche Lehr- und Handbuch für den erfoglreichen TSO-Benutzer*, Oldenburgverlag, München Wien

[JCL] G. D. Brown (2004) z/OS JCL *Job Control Language im Betriebssystem z/OS*, Oldenburgverlag, München Wien

[BMS] *http://www.ti-leipzig.de/os390/paul/docs/paul6.pdf*