UNIVERSITÄT LEIPZIG

Enterprise Computing
Einführung in das Betriebssystem z/OS

Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

WS 2012/13

z/OS Betriebssystem Teil 1

Job Control Language

System z und S/390 Betriebssysteme

Im Laufe der Jahre sind eine ganze Reihe von Betriebssystemen für die System z Plattform entstanden:

Name	Hersteller	
• z/OS	IBM	große Installationen (früher als OS/390, MVS bezeichnet)
• z/VSE	IBM	mittelgroße Installationen
• z/VM	IBM	Virtualisierung, Software Entwicklung
• z/TPF	IBM	spezialisierte Transaktionsverarbeitung
• zLinux	Public Domain	Normale Suse oder Red Hat Adaption für System z Hardware
• UTS 4	Amdahl	Unix Betriebssystem, based on System V, Release 4 (SVR4)
• Open Solaris	Sun	Open Solaris Adaption für System z Hardware
• BS2000	Siemens/Fujitsu	von Siemens entwickelte Alternative zu OS/390

Alle System z bzw. S/390 Betriebssysteme sind Server Betriebssysteme, optimiert für den Multi-User Betrieb. Eine sehr ungewöhnliche Rolle spielt das z/TPF Betriebssystem.

z/Transaction Processing Facility
z/TPF

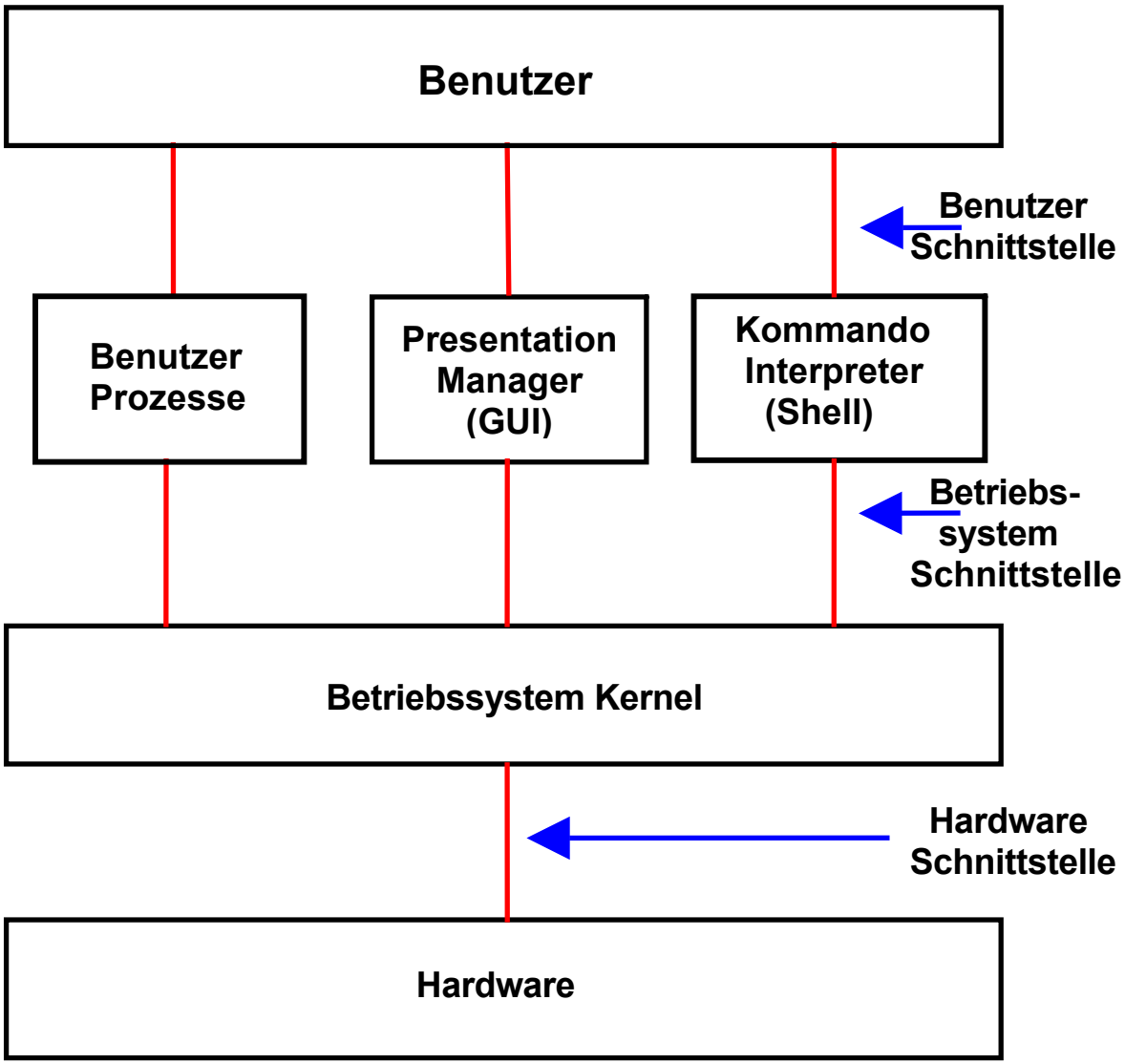
Transaction Processing Facility (TPF) wurde ursprünglich als Platzreservierungssystem für die Fluggesellschaft American Airlines entwickelt und wird heute sowohl für Reservierungen für Fluggesellschaften, Hotels, Reisebüros, Mietwagenfirmen und Eisenbahngesellschaften als auch (vorrangig) für die Steuerung von Geldausgabe-Automaten eingesetzt. TPF unterscheidet nicht zwischen Kernel- und User-Status; sämtliche Anwendungen laufen aus Performance-Gründen im Kernel-Status. Es existieren weltweit etwa 300 (sehr große) Installationen. Die Visa Kreditkartenverifizierung und das AMADEUS-Flugplatzreservierungssystem der Deutschen Lufthansa sind Beispiele für den Einsatz von TPF.

z/TPF kann nur für die Transaktionsverarbeitung eingesetzt werden, bietet nur sehr einfache Funktionen, ist aber mit dieser Einschränkung der weltweit leistungsfähigste Transaktionsserver. z/TPF Anwendungen werden auch heute noch häufig in Assembler programmiert.

Ein Beispiel ist die Firma Worldspan, ein weltweiter Anbieter von Reise-Reservierungs-Systemen. Worldspan hat sechs System z Rechner mit TPF installiert. Als Provider von elektronischen Datendiensten stellt Worldspan hiermit circa 700 Anbietern von Reiseangeboten und Millionen von Reisenden weltweit eine gemeinsame Plattform zur Verfügung.

Worldspan setzt die z/TPF Mainframe Server ein, um sowohl Reisebüros als auch Anbietern von Online-basierten Reisediensten die Möglichkeit zur Nutzung des weltweiten Global Distribution System (GDS) zu geben, über das zum Beispiel die Bestellung und Buchung von Reiseprodukten wie Flugzeugtickets, Hotels, Mietwagen und andere Reisedienstleistungen durchgeführt wird.

Durch die Nutzung von z/TPF ist Worldspan in der Lage, 17.000 Kundenanfragen pro Sekunde auf den Mainframes zu beantworten.

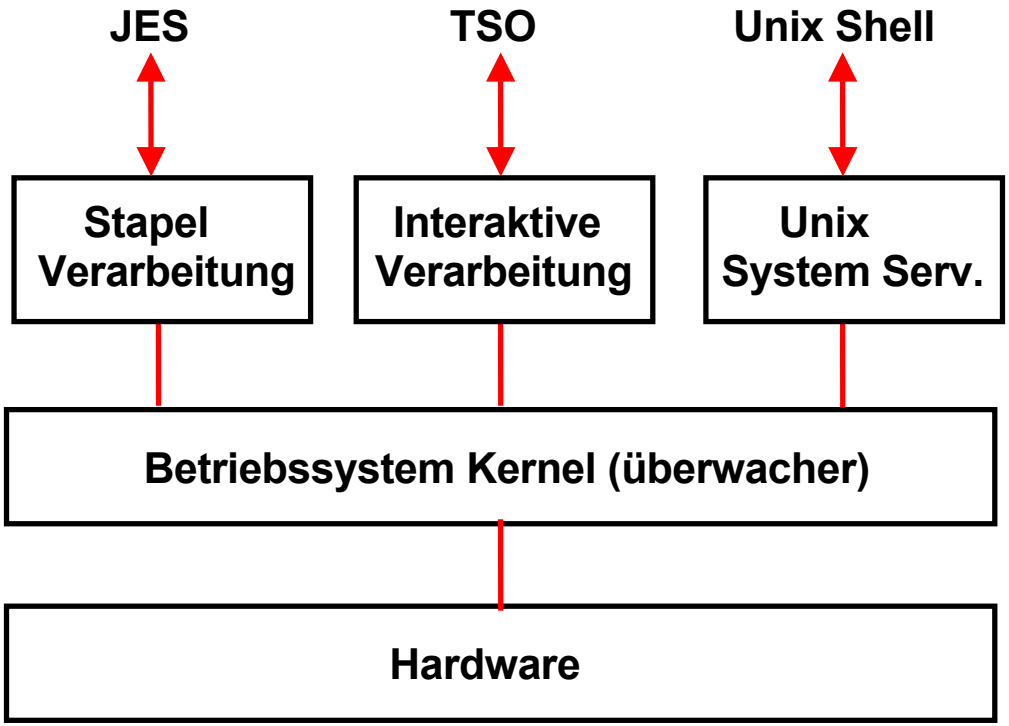


Schichtenmodell der Rechnerarchitektur Windows, Linux, Unix, z/OS

Bei allen Rechnern setzt der Betriebssystem Kernel (Überwacher, Supervisor, Basic Control Programm) direkt auf der Hardware auf. Der Kernel kann nur über wohl definierte Schnittstellen aufgerufen werden; in der Praxis sind das meistens Unterbrechungen.

Auf dem Kernel setzen Benutzerprozesse (von einem Benutzer geschriebenen Anwendungen) oder Systemprozesse auf. Systemprozesse werden auch als Subsysteme bezeichnet. Zwei der wichtigsten Systemprozesse sind der Presentation Manager (Graphical User Interface, GUI, z.B. Windows Desktop, Motiv, KDE) und der Kommando Interpreter (z.B. Windows DOS Shell, Unix Shell. TSO).

z/OS arbeitet fast ausschließlich mit verschiedenen Kommando Interpretern. GUIs werden vor allem für Anwendungen benutzt.



Die drei wichtigsten z/OSSubsysteme (Shells) für die Steuerung des Systems sind:

- JES (Job Entry Subsystem) für die Stapelverarbeitung
- TSO (Time Sharing Option) für die interaktive Verarbeitung (z.B. Programmentwicklung, Administration)
- Unix System Services, Posix kompatibles Unix Subsystem

z/OS Grundstruktur

Begriffe

z/OS, wie auch Linux und Windows arbeitet mit dem Konzept eines Prozesses. Ausführbare Programme sind normalerweise in Programmbibliotheken auf einem Plattenspeicher abgespeichert. Ein Prozess ist der Aufruf und die Ausführung eines Programms.

z/OS, wie auch Linux und Windows arbeiten mit einer virtuellen Adressumsetzung (Address Translation). Hierbei existieren viele virtuelle Adressenräume, die mit der hexadezimalen Adresse Hex 00 ... 00 beginnen und eine einstellbare maximale Größe haben. z/OS bezeichnet diese virtuellen Adressenräume als (virtual) Address Spaces, Access Spaces oder auch als Regions. Die Begriffe sind austauschbar.

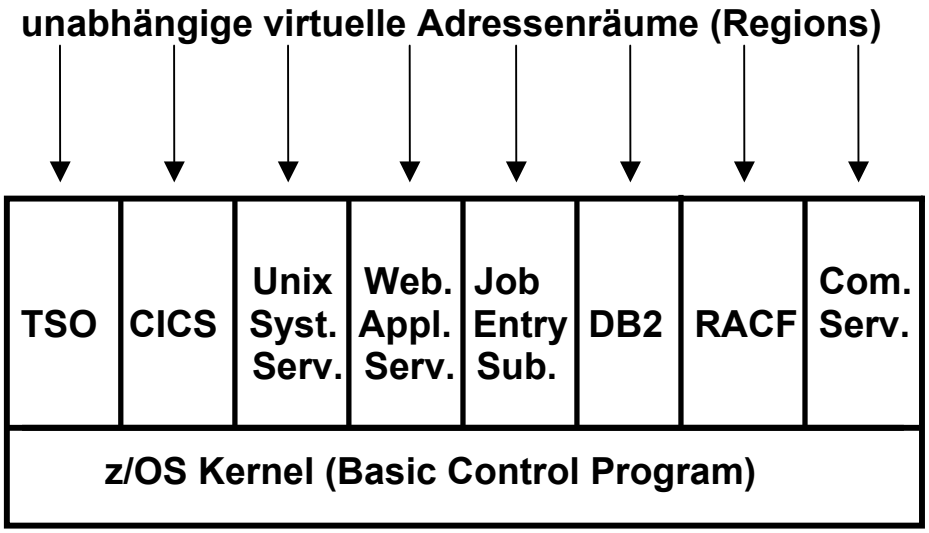
Moderne Rechner arbeiten multiprogrammiert. Auf einem Windows, Linux oder z/OS Rechner laufen in der Regel zahlreiche Prozesse parallel zueinander ab. Jeder der Prozesse läuft in einem eigenen virtuellen Adressenraum (Region, Address Space).

Manche der Prozesse sind Systemprozesse. Systemprozesse laufen als Teil des Betriebssystems. Wenn Sie unter Windows den Task Manager aufrufen (CTR+ALT+DEL), sehen Sie dort eine lange Liste von aktiven Systemprozessen in entsprechend vielen virtuellen Adressenräumen.

z/OS bezeichnet einige seiner Systemprozesse als Subsysteme. Subsysteme sind in sich abgeschlossene Softwareeinheiten. Manche Subsysteme wie JES, TSO, der Security Server oder der Communication Server sind ein Bestandtel des z/OS Betriebssystems. Andere Subsysteme wie die DB2 Datenbank, der WebSphere Application Server oder der CICS Transaktionsmanager sind optional, müssen extra installiert werden, und kosten zusätzliche Lizenzgebühren.

Im Gegensatz zu den Systemprozessen stehen die Benutzer- (Anwendungs-) Prozesse. Diese führen Programme (Anwendungen, Applications) aus, die vom Benutzer des z/OS Systems geschrieben wurden.

z/OS Grundstruktur



Einige der (zahlreichen) Subsysteme sind:

- **CICS** Transaktionsverarbeitung
- **TSO** Shell, Entwicklungsumgebung
- **USS** Unix kompatible Shell, Entwicklungsumgebung
- **WAS** WebSphere Web Application Server
- **JES** Job Entry Subsystem
- **DB2** relationale Datenbank
- **RACF** Sicherheitssystem
- Communications Server

Systemprozesse und Anwendungsprozesse laufen in getrennten virtuellen Adressenräumen. Der z/OS Kernel unterstützt eine Vielzahl von virtuellen Adressenräumen, die im z/OS Jargon als Regions bezeichnet werden.

Einige der Regions beherbergen Subsysteme, die Teil des Betriebssystems sind, aber im Benutzerstatus laufen. Gezeigt sind eine der wichtigsten Subsysteme, die wir uns im Einzelnen noch ansehen werden.

Programmarten

z/OS begann Anfang 1966 unter dem Namen OS/360 als reines Stapelverarbeitungssystem eingeführt. Es wurde von Fred Brooks entwickelt, und diente als Vorlage für sein berühmtes Buch „The mythical Manmonth“ (http://en.wikipedia.org/wiki/The_Mythical_Man-Month).

Mit wachsendem Funktionsumfang wurde der Name immer wieder geändert: OS/360 → MFT→ MVT → MVS . 1996 erfolgte eine Namensänderung von MVS nach OS/390. 2000 wurde der Name z/OS im Zusammenhang mit der neuen 64 Bit Adressierung eingeführt.

An Stelle von z/OS wird der Name MVS heute noch häufig gebraucht, was nicht ganz korrekt ist.

Subsysteme sind Programmprodukte wie Datenbanken und Transaktionsmonitore, die Laufzeitumgebungen für eigentliche Benutzerprogramme zur Verfügung stellen.

Benutzerprogramme können sein:

- **klassische z/OS (bzw. OS/390) Hintergrundprogramme (Batch Programs)**
- **Benutzeranwendungen, die unter der Kontrolle von Subsystemen wie TSO,CICS, IMS oder Websphere ablaufen, oder**
- **UNIX-Programme, die die UNIX System Services unter z/OS ausnutzen.**

Systems Management Funktionen werden für die Steuerung und Überwachung des Ablaufes benötigt. Es gibt sehr viele solcher Funktionen, sowohl von IBM als auch von Drittanbietern.Sie dienen der Überwachung des Betriebssystems und, da sich das Betriebssystem in weiten Bereichen selbst steuert, zur Überwachung der Subsysteme und der Benutzeranwendungen.

Job Control Language
JCL

Zwei Arten der Datenverarbeitung

Betriebswirtschaftliche Großrechner betreiben Datenverarbeitung auf zwei unterschiedliche Arten:

Bei der Interaktive Verarbeitung dient der Mainframe als Server in einer Client/Server Konfiguration. Zahlreiche Klienten (meistens PCs, aber auch Geldausgabeautomaten oder Registrierkassen im Supermarkt) nehmen Dienstleistungen des Servers in Anspruch. In Großunternehmen wie z.B. der Volkswagen AG können das mehrere 100 000 Klienten sein, die in der Fachsprache oft als Terminals bezeichnet werden.

Dies ist ein Beispiel für die interaktive Verabeitung: Sie sitzen an Ihrem PC und haben einen Excel Prozess gestartet. Sie arbeiten mit einer großen Excel Tabelle und stoßen eine Berechnung an, die viele Sekunden oder Minuten dauert.

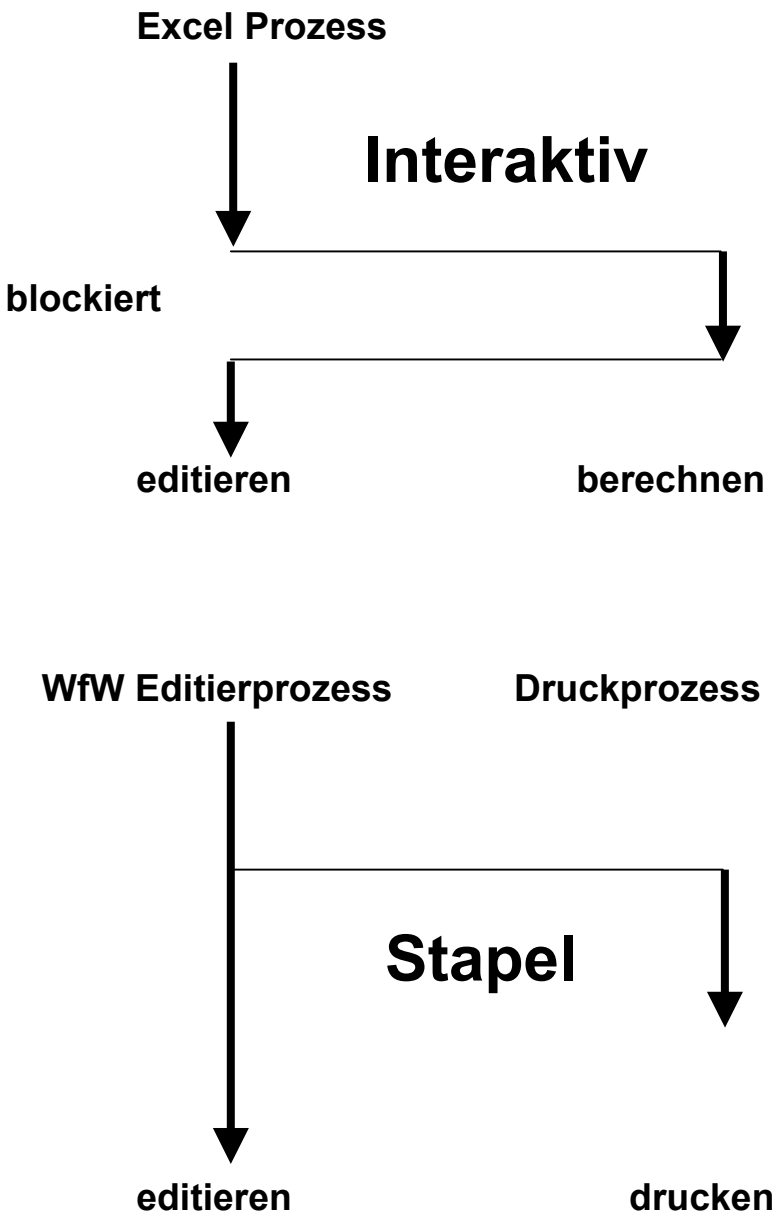
Während der Berechnung blockiert der EXEL Prozess, reagiert z.B. nicht auf Tastatur-Eingaben. Der Grund ist, die Bearbeitung wird als Teil des Excel Prozesses durchgeführt; während dieser Zeit kann der Prozess nichts anderes machen.

Der Gegensatz zur interaktiven Verarbeitung ist die Stapelverarbeitung. Hier stößt ein Prozess, z.B. für länger laufende Verarbeitungsaufgaben, einen zweiten Prozess an.

Dies ist ein Beispiel für die Stapelverabeitung: Sie sitzen an Ihrem PC und editieren ihre Masterarbeit mit Word for Windows. Dies ist ein interaktiver Prozess. Sie beschließen, die ganze Arbeit auf Ihrem Tintenstrahldrucker probeweise auszudrucken. Dies dauert viele Sekunden oder Minuten.

Während des Druckens blockiert der Word Prozess nicht. Sie können die Diplomarbeit weiter editieren.

Hierzu setzt Word for Windows neben dem Editierprozess einen getrennten Druckprozess auf, der als Stapelberareitungsprozess (batch job oder background Prozess) bezeichnet wird. Der Scheduler/Dispatcher des Betriebssystems stellt zeitscheibengesteuert beiden Prozessen CPU Zeit zur Verfügung.



Interaktive und Stapelverarbeitung

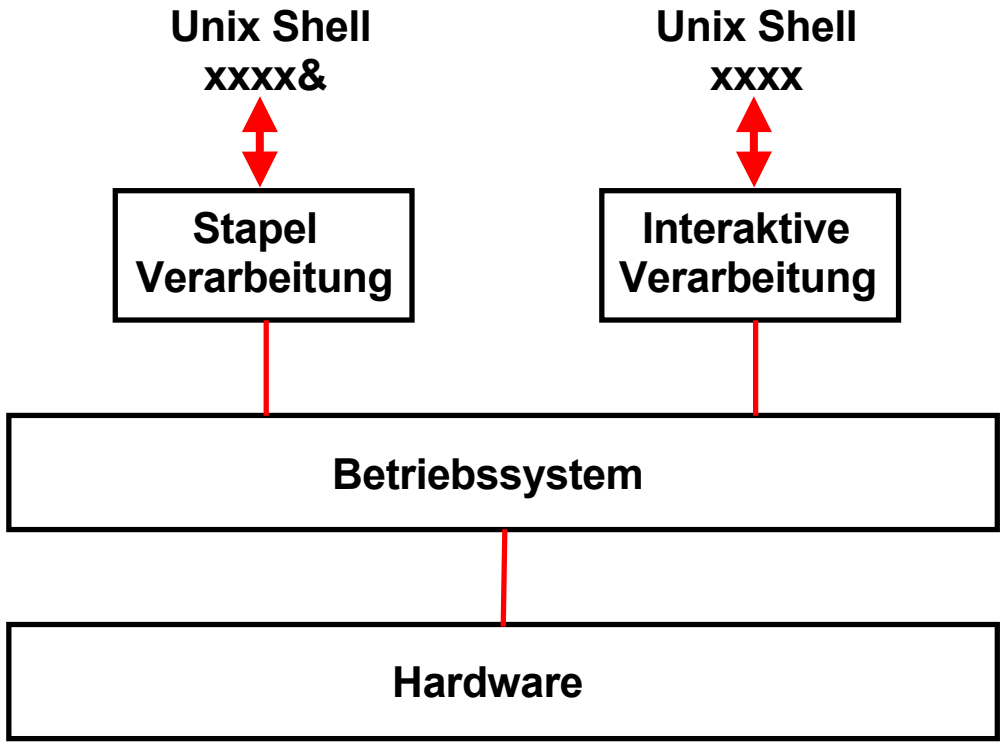
Die nebenstehende Grafik zeigt diesen Zusammenhang im zeitlichen Ablauf.

Ein Mainframe kann ohne weiteres 100 000 Benutzer mit 100 000 angeschlossene Terminals bedienen. Hierzu wird für jeden Benutzer ein eigener interaktiver Prozess in einem eigenen virtuellen Adressenraum gestartet.

Parallel dazu laufen auf dem Mainframe zahlreiche Stapelverarbeitungsprozesse, die als Jobs bezeichnet werden.

Im Rechenzentrum der Credit Suisse, einer Schweizer Großbank in Zürich, waren es im Frühjahr 2010 durchschnittlich 86 000 Jobs pro Tag. Die Ausführungszeit der einzelnen Jobs konnte Sekunden, Minuten, Stunden, und in Extremfällen auch Tage betragen.

Starten, Überwachung des Ablaufs und Terminierung der einzelnen Jobs ist die Aufgabe eines z/OS Subsystems, des Job Schedulers.



Unix Verarbeitung - Grundstruktur
Zugriff über Telnet und eine Shell auf einen Unix Rechner

Unter Unix kann die Stapelverarbeitung (Batch Processing) als Sonderfall der interaktiven Verarbeitung betrieben werden. In der Shellsprache werden Batch-Aufträge durch ein nachgestelltes „&“ gekennzeichnet.

Stapelverarbeitung unter Linux

Der cron-Daemon ist eine Jobsteuerung von Unix bzw. Unix-artigen Betriebssystemen wie Linux, BSD oder Mac OS X, die wiederkehrende Aufgaben (cronjobs) automatisch zu einer bestimmten Zeit ausführen kann. cron ist die Software, die dem z/OS Batch Processing Subsystem (Job Entry Subsystem, JES) am nächsten kommt. Allerdings geht der JES Funktionsumfang weit über den von cron hinaus.

Cron startet Skripte und Programme zu vorgegebenen Zeiten. Der auszuführende Befehl wird in einer Tabelle, der sog. crontab, gespeichert. Jeder Benutzer des Systems darf eine solche crontab anlegen.

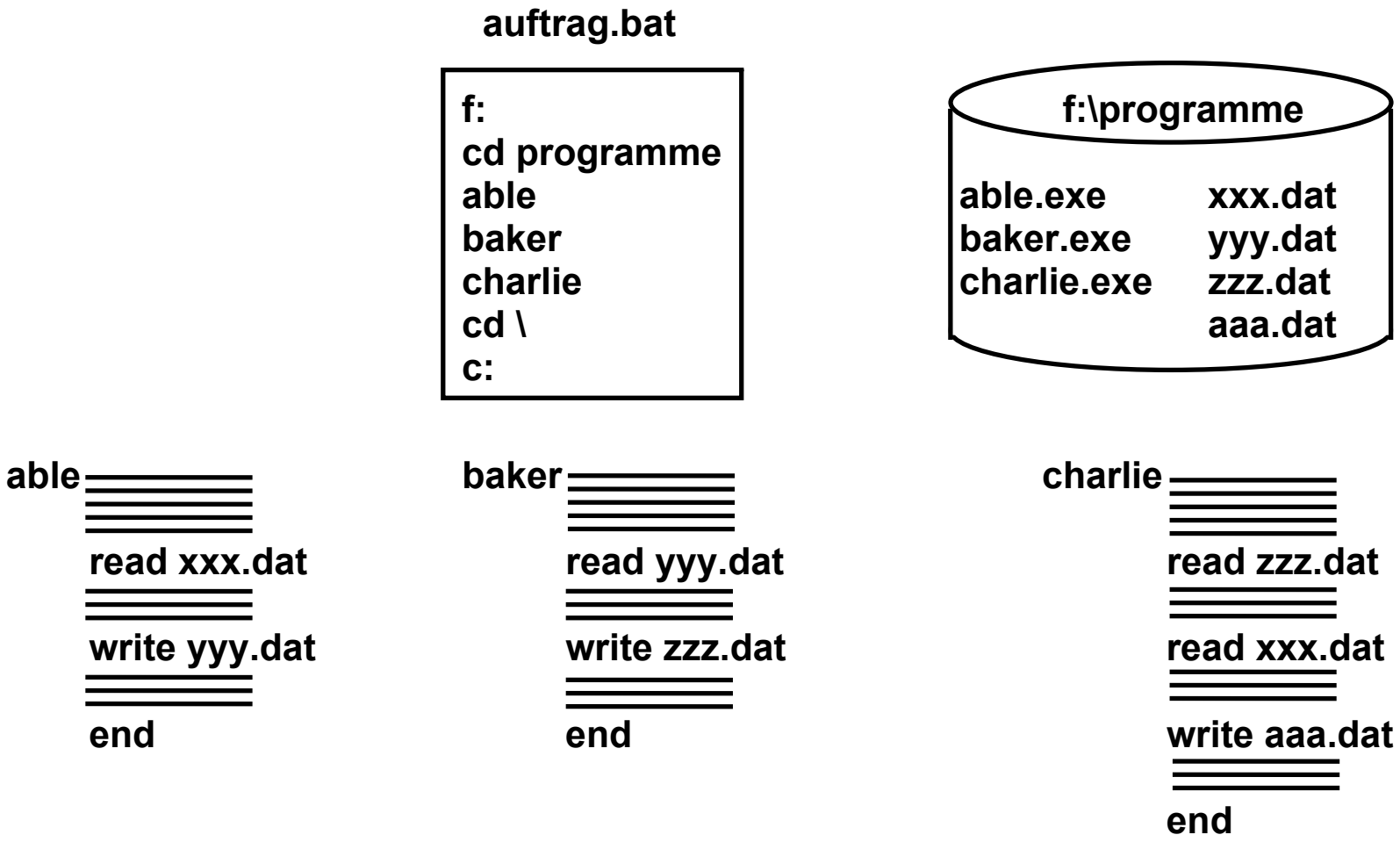
Diese Tabelle besteht aus sechs Spalten; Die ersten fünf dienen der Zeitangabe (Minute, Stunde, Tag, Monat, Wochentage), die letzte enthält den Befehl. Die einzelnen Spalten werden durch Leerzeichen oder Tabulatoren getrennt.

Häufig führt der Cron-Daemon wichtige Programme für die Instandhaltung des Systems aus, wie zum Beispiel Dienste für das regelmäßige Archivieren und Löschen von Logdateien.

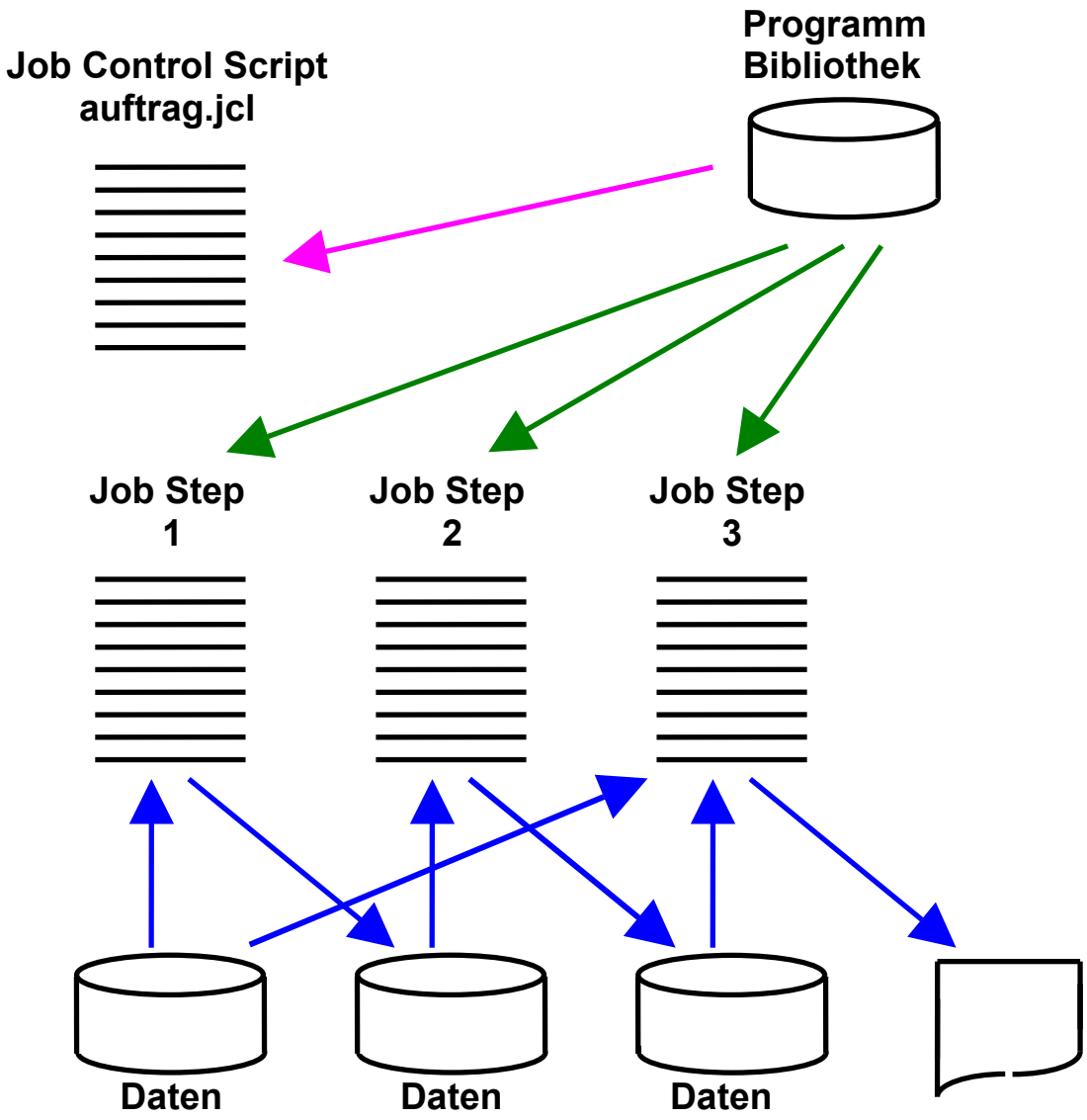
Beim Hochfahren eines Linux Systems wird als Erstes cron gestartet. cron wiederum startet einen Shell Prozess. Ähnlich wird beim Hochfahren eines z/OS Systems JES als erstes gestartet. Das Hochfahren selbst bewirkt ein weiterer Prozess, der Master Scheduler.

Griechisch chronos (χρόνος) bedeutet Zeit.

Beispiel eines DOS Jobs



Als Beispiel sei angenommen, dass Sie unter Benutzung Ihrer Windows DOS Eingabeaufforderung drei Programme able, baker und charlie der Reihe nach ausführen wollen. Um dies zu automatisieren können Sie ein .bat Programm mit dem Namen auftrag.bat schreiben. auftrag.bat ist ein Job Script, welches die sequentielle Ausführung der drei Programme automatisch steuert. Es ist in der .bat Language geschrieben.



Die Ausführung von auftrag.bat besteht aus drei Schritten, den Job Steps, die jeweils die Ausführung der drei Programme able, baker und charlie bewirken.

Unter z/OSverwendet man für die Job Steuerung eine eigene Scriptsprachen , die Job Control Language, JCL.

Da das JCL Programm die verwendeten Dateien angibt, ist ein „late Binding“ der verwendeten Dateinen an die auszuführenden Programme möglich.

Eine „Cataloged Procedure“ ist ein JCL Programm, welches vom Benutzer für eine spätere Verwendung zwischengespeichert wird (.z.B. in einer vom Benutzer erstellten Library JCLLIB) und bei Bedarf mittels eines JCL Befehls aufgerufen wird.

Konzept eines z/OS Jobs

Die Eigenschaften eines Stapelverarbeitungsprozesses sind häufig:

- **Lange Laufzeit**
- **wird häufig periodisch zu festgelegten Zeiten ausgeführt**
- **Hohes Datenvolumen. Es kann aus Tausenden oder Millionen von Datenbankelementen (z.B. Reihen in einer SQL Tabelle) bestehen**
- **Für die Daten möglicherweise komplexe Verarbeitungsanforderungen**
- **Der Verarbeitungsprozess benötigt möglicherweise große Datenmengen von einem anderen Rechner. Diese werden als ein große Datei zu einer bestimmten Zeit angeliefert.**
- **Der Stapelverarbeitungsprozess läuft asynchron zu irgendwelchen Benutzer-Aktionen. Er ist nicht Teil einer Benutzer Session in einem Online (Client/Server) System.**
- **Wird typischerweise nicht von einem Benutzer eines Online Systems gestartet, Normalerweise startet ein Online Benutzer keinen Stapelverarbeitungsprozesses, und wartet auch nicht auf eine Antwort.**

  (Eine Ausnahme sind die ersten Übungsaufgaben unseres Enterprise Computing Praktikums).

Job Scheduler

Ein reguläres Windows oder Linux Betriebssystem enthält nur primitive Job Scheduling funktionen wie z.B. cron. Zum Füllen dieser Marktlücke stellen eine ganze Reihe von Software Unternehmen Job Scheduler für Apple, Windows, Unix und Linux Plattformen her. Beispiele sind:

- **cosbatch der Firma OSM** http://www.cosbatch.com/
- **Dollar Universe der Firma Orsyp,** http://www.orsyp.com/en.html
- **OpenPBS (Portable Batch System),** http://www.openpbs.org,
  open source package ursprünglich von der NASA entwickelt
- **PBS Pro der Firma PBS Grid Works,** http://www.pbsgridworks.com,
  enhanced commercial version von OpenPBS.
- **Global Event Control Server der Firma Vinzant Software,** http://www.vinzantsoftware.com/
- **ActiveBatch der Firma Advanced–System Concepts,** http://www.advsyscon.com/

All diese Produkte erreichen bei weitem nicht den Funktionsumfang von z/OS JES

Script Sprachen

Sie kennen vermutlich bereits eine der gängigen Scriptsprachen wie Pearl, PHP, Java Script, Tcl/Tk, Phyton, Ruby usw.

Sie haben vielleicht auch schon einen der erbitterten Religionskriege miterlebt, welche Scriptsprache die beste ist, und welche grottenschlecht ist. Glauben Sie mir, es gibt keine „beste“ Scriptsprache.

Eine weitere Scriptsprache ist REXX (*Restructured Extended Executor*). REXX wird vor allem auf Mainframes eingesetzt. REXX Interpreter sind aber auch für Windows, Apple, Linux, alle Unix Dialekte usw. verfügbar, und weiter verbreitet als allgemein angenommen. REXX wird vor allem von Leuten benutzt, die schon auf dem Mainframe damit gearbeitet haben.

Hier ist ein kurzes Programm in REXX

```
/* A short program to greet you                                  */
/* First display a prompt                                        */

say ‘Please type your name and then press ENTER:‘
parse pull answer              /* Get the reply into answer */

/* If nothing was typed, then use a fixed greeting              */
/* otherwise echo the name politely                              */

if answer=‘‘   then say ‘ Hello Stranger! ‘
               else say ‘ Hello ‘ answer ‘!‘
```

Auffallend ist, dass an Stelle des Schlüsselwortes “write” das Schlüsselwort “say” benutzt wird. Daran können sie immer erkennen, ob ein Script Programm in REXX geschrieben ist.

Shell Scripte

Einige Script Sprachen sind sog. shell Scripte. Dazu gehören .bat Files unter Windows, Unix und Linux Shell Scripts und eben JCL uner z/OS. Shell Scrips sind aus Sequenzen von shell Kommandos entstanden. Auch die make file, die Sie als C++ Programmierer für das Übersetzen Ihres C++ Programms benutzen , benutzt eine eigenes Shell Script.

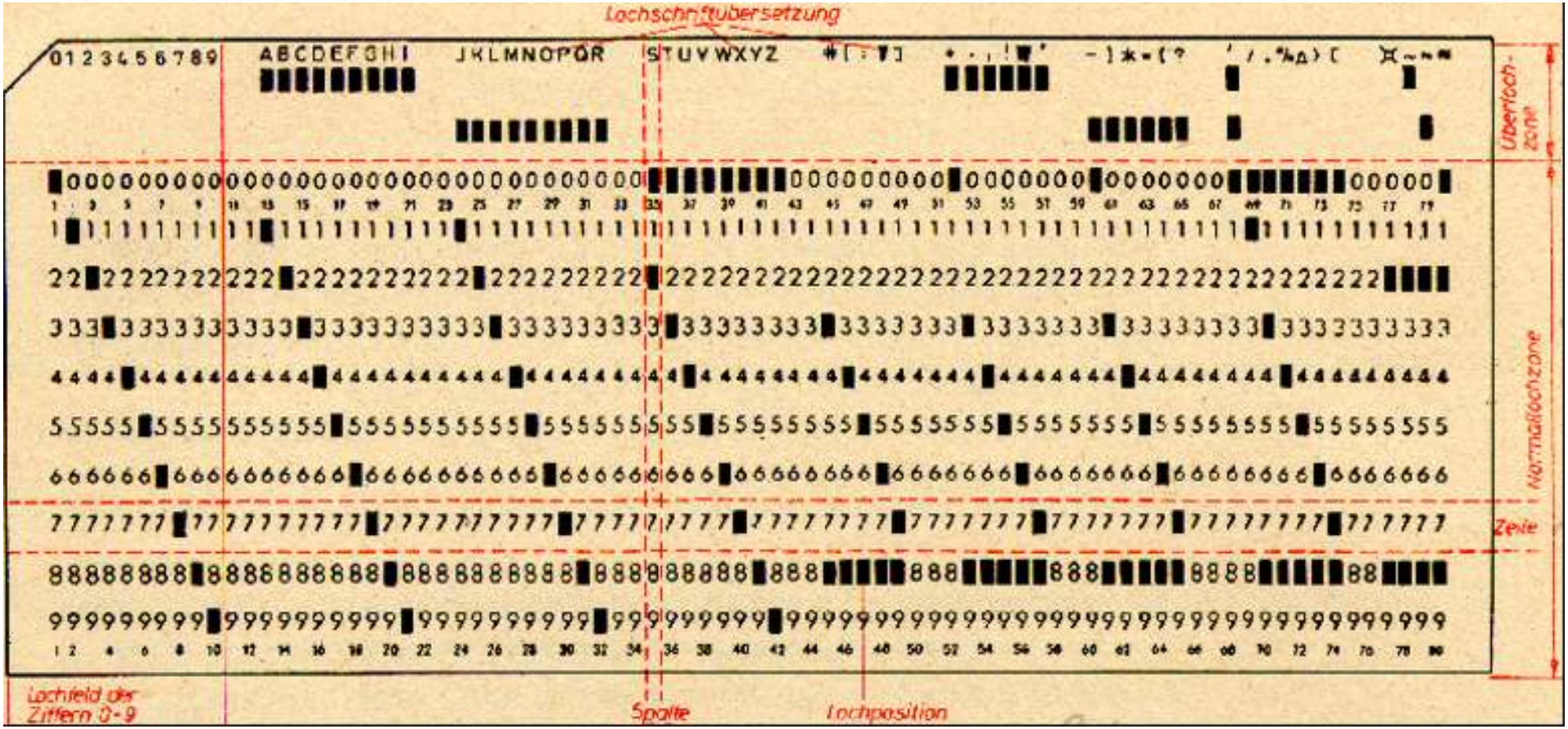
Shell Scripte sind in der Regel für den Anfänger unübersichtlicher und schwieriger zu verstehen als normale Script Sprachen wie Tcl/Tk, REXX oder PHP. Das gilt besonders auch für JCL.

Die JCL Syntax macht einen absolut archaischen Eindruck, und ist für den erstmaligen Benutzer ein Kulturschock. Das wird auch für Sie gelten, wenn sie Ihr erstes JCL Script schreiben,also „be prepared“. JCL hat sicher viel dazu beigetragen, Mainframes zu dem Ruf „obsolete Technology“ zu verhelfen.

Tatsache ist, nachdem Neulinge ihren ersten Schock überwunden haben (das ist nach wenigen hndert Zeilen JCL Code der Fall), beginnen sie JCL heiß und innig zu lieben. Es hat viele Versuche gegeben, JCL durch eine „moderne“ Scriptsprache zu ersetzen – bisher mit wenig Erfolg. Tatsache ist, JCL ist relativ leicht erlernbar, sehr mächtig und sehr produktiv.

JCL wurde zusammen mit OS/360 (der ersten Version des heutigen z/OS) entwickelt und hat Ähnlichkeiten mit Unix Shell Scrips; ein Beispiel ist die Ähnlichkeit der Unix dd und des JCL DD Kommandos.

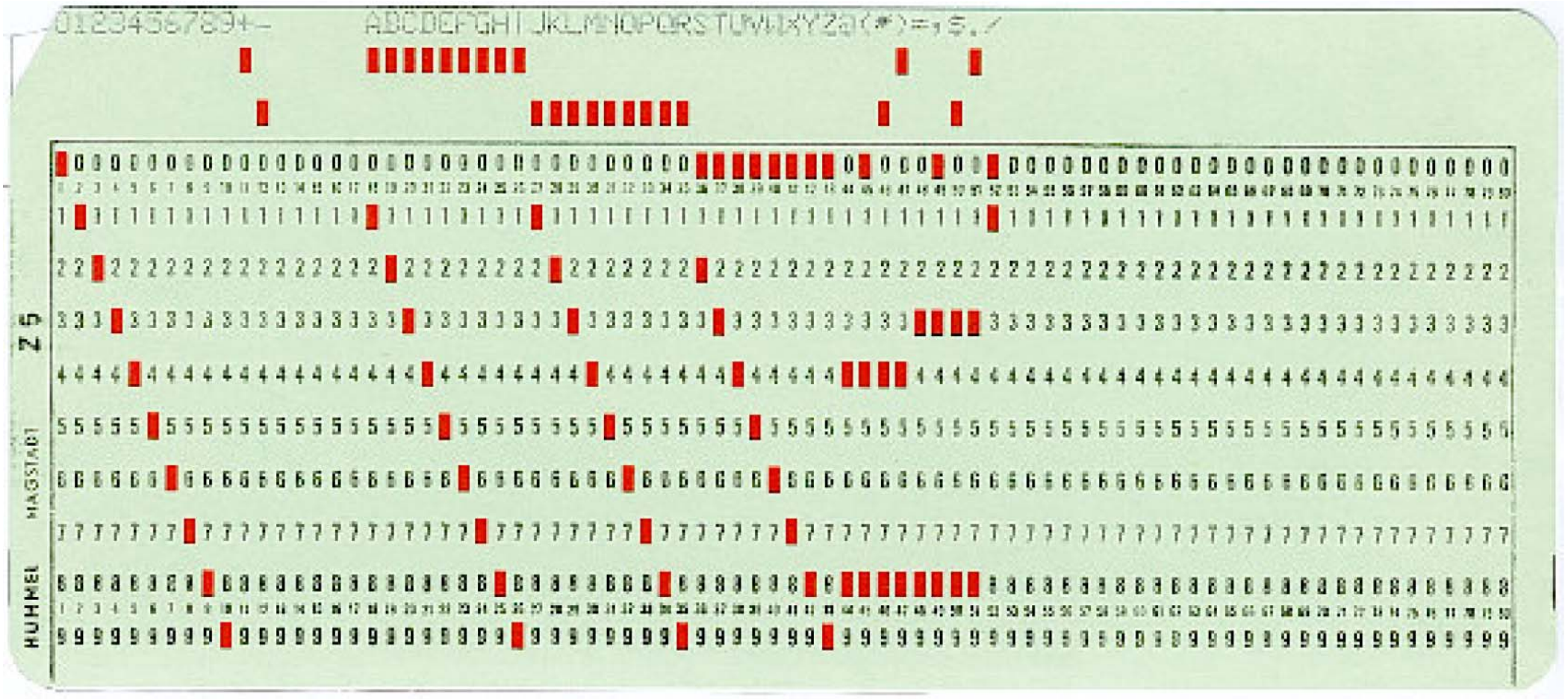
Es ist unmöglich, JCL zu verstehen, ohne sich mit der historischen Entwicklung auseinanderzusetzen. Diese geht auf die frühere Benutzung von Lochkarten zurück. Lochkarten waren bis in die 70er Jahre beliebte und kostengünstige Elemente für die Speicherung von Daten.



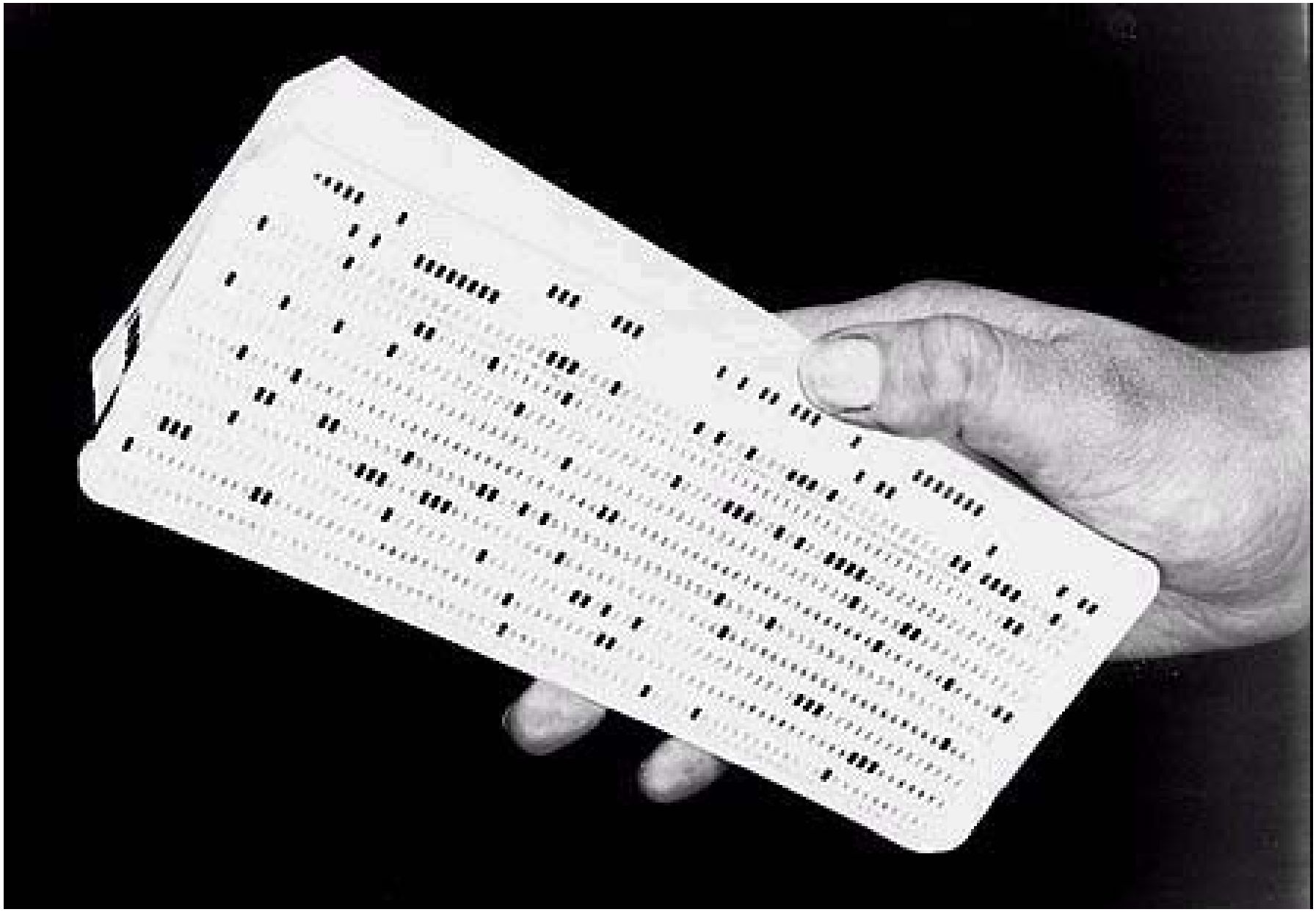
IBM Lochkarte

Die IBM Lochkarte, etwa 19 x 8 cm groß, wurde 1928 eingeführt. Vorläufer mit einem etwas anderen Format wurden von Herrmann Hollerith erfunden und 1890 bei der Volkszählung in den USA und wenig später auch in Deutschland eingesetzt.

Die IBM Lochkarte besteht aus 80 Spalten. In jede Spalte konnte an einer von 12 Stellen jeweils ein Loch gestanzt werden. Die Position konnte in einem Lochkartenleser abgefühlt werden. Da,mit war es möglich, mittels der sog. BCD Kodierung (binary coded decimal) in jeder Lochkarte 80 alphanumerische Zeichen zu speichern.



Zahlreiche Darstellungen von Lochkarten sind zu finden unter
http://www.google.de/search?q=punched+card&hl=de&lr=&as_qdr=all&prmd=imvns&tbm=isch&tbo=u&source=univ&sa=X&ei=wiqlUMu4FobV4QSk0oHICA&ved=0CCoQsAQ&biw=1516&bih=963



Ein Stapel Lochkarten
http://www-03.ibm.com/systems/z/os/zvse/about/history1960s.html

Eine lustige Geschichte

Als Student hatte ich die Gelegenheit, an einer Besichtigung des Rechenzentrums der Bergbaugesellschaft Hibernia AG im Ruhrgebiet teilzunehmen. Dort erfolgte die Lohn- und Gehaltsabrechnung ausschließlich auf Lochkartenmaschinen; es gab noch keinen Computer. Beim Herausgehen erhielt jeder Besucher eine leere Lochkarte als Andenken.

Einige Zeit später musste ich zur mündlichen Prüfung Betriebswirtschaftslehre als Nebenfach im Grundstudiums. Beim Weggang aus meinem Zimmer lag die Lochkarte auf dem Tisch; ich sah sie mir an und steckte sie in die innere Brusttasche.

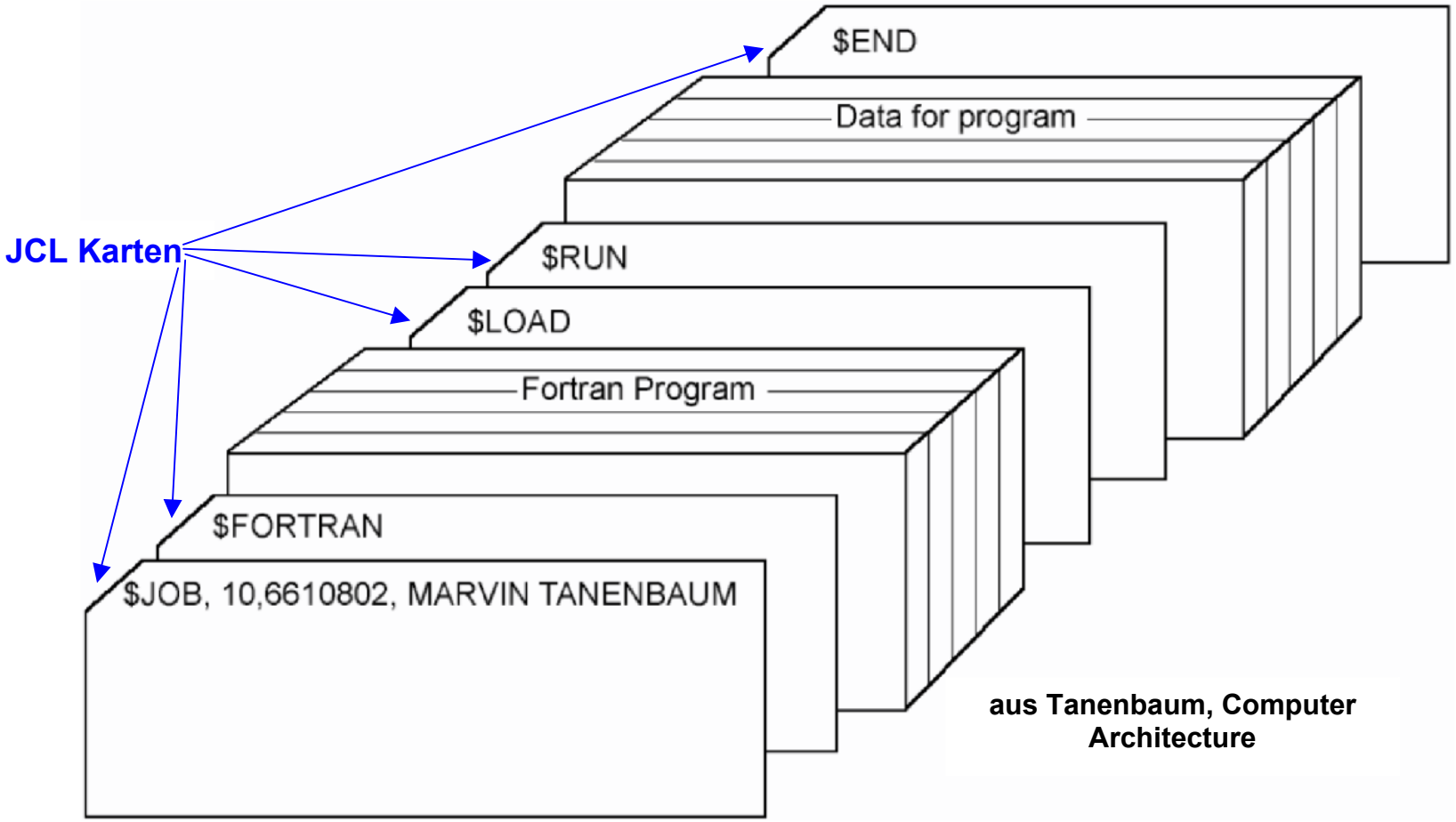
Prompt begann die Prüfung mit Fragen zur Lochkartentechnik. Wieviel Spalten hat denn eine Lochkarte fragte der Professor. 82 Spalten war meine Antwort. Fast richtig, sage mein Professor, in Wirklichkeit sind es 80 Spalten. Nein, Herr Professor, sagte ich, es sind 82 Spalten, zog die Lochkarte der Bergwerksgesellschaft Hibernia aus der Tasche, und diese hatte tatsächlich 82 Spalten. Und die Prüfung war gerettet.

Manchmal – leider viel zu selten – hat man halt auch mal unverdientes Glück.

Was ich damals nicht wusste, die Hibernia AG hatte Lochkartenmaschinen von einem anderen Hersteller als IBM installiert, und Lochkartenformate unterschiedlicher Hersteller waren noch nicht standardisiert. In den 60er Jahren setzte sich das 80 Spalten IBM Format langsam als Industrie-Standard durch.

Und damit zurück zur JCL.

v



Gezeigt ist die Implementierung eines FORTRAN Programms etwa Anno 1960. Es besteht aus einer Reihe von JCL Karten, ein JCL Statement pro Karte. Zwischen den JCL Karten befindet sich das FORTRAN Programm Karten Deck, jeweils eine Karte pro Fortran Statement, sowie ein weiteres Karten Deck mit den von dem vom Fortran Programm zu verarbeitenden Daten. Das gesamte Kartendeck wurde von dem Lochkartenleser des Rechners in den Hauptspeicher eingelesen und verarbeitet. Die Ausgabe erfolgte typischerweise mittels eines angeschlossenen Druckers.

Später wurde es üblich, das Fortran Programm und die Daten auf Magnetband (und noch später auf einem Plattenspeicher) zu speichern, und das Programmdeck und das Datendeck durch zwei Library Call Lochkarten zu ersetzen. Noch später speicherte man dann das JCL Script ebenfalls in einer Library auf dem Plattenspeicher.

JCL Format

Ein JCL Script bestht aus einzelnen JCL Statements. Ein JCL Statement passte früher auf eine Lochkarte mit 80 Spalten, und daran hat sich bis heute nichts geändert.

Wie auch bei anderen Programmiersprachen besteht ein JCL Statement aus mehreren (genau fünf) Feldern.

/ /	Label	Operation	Parameter	Kommentar

Um die Logik zum Parsen eines Statements zu vereinfachen, führte man einige Konventionen ein:

- **Jedes Feld beginnt immer an einer bestimmten Spalte der Lchkarte. Das vereinfachte damals das Parsen des Statements. Diese Spaltenabhängigkeit existiert auch heute noch !!! Wenn Sie dagegen verstoßen, gibt es eine Fehlermeldung. Warnung: Dies ist mit großem Abstand der häufigste Programmierfehler beim Bearbeiten unserer ersten Mainframe Tutorials.**
- **Jedes JCL Statement beginnt mit den beiden Zeichen / / (forward slashes) . Damit war es möglich, die JCL Statement leicht von den Lochkarten des Fortran Decks und des Datendecks zu unterscheiden. Die beiden Slashes befinden sich in Spalte 1 und 2 .**
- **Das Label Feld beginnt in Spalte 3 , die maximale Länge ist 8 .**
- **Das Operation folgt dem Label Feld, getrennt durch ein Leerzeichen, in Spalte 12 .**
- **Das Parameter Feld (Operand Feld) folgt dem Operation Feld, getrennt durch ein (oder mehr) Leerzeichen**
- **Alles was hinter dem Operand Feld folgt, ist ein Kommentar. Zwischen Operand und Kommentar muss sich (mindestens) ein Leerzeichen befinden.**

z/OSMVS JCL Reference SA22-7597-10 Eleventh Edition, April 2006 , chapter 3, http://ulita.ms.mff.cuni.cz/pub/predn/swi119/iea2b661.pdf
see also http://www.informatik.uni-leipzig.de/cs/Literature/Features/CodeJCL.pdf

Statement Bezeichnung beginnt in Spalte 12

```
//SPRUTHC  JOB (123456),'SPRUTH',CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),
//             NOTIFY=&SYSUID,TIME=1440
//PROCLIB  JCLLIB ORDER=CBC.SCBCPRC
//CCL      EXEC PROC=EDCCB,
//           INFILE='SPRUTH.TEST.C(HELLO1)'
//           OUTFILE='SPRUTH.TEST.LOAD(HELLO1),DISP=SHR'
```

Label, Spalte 3 - 10

Ein einfaches JCL Script

JOB Statement markiert den Anfang eines Jobs

EXEC Statement bezeichnet Prozedur, die ausgeführt werden soll

PROC Statement gibt die Adresse einer Procedur an

DD Statement bezeichnet die zu benutzenden Dateien (hier nicht verwendet)

Achten Sie darauf, dass Ihr JCL Script die richtigen Spalten benutzt !!!

Beispiel

Beispiel für einen JCL Befehl:

```
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
```

RECFM, FB, LRECL und BLKSIZE sind Schlüsselwörter der JCL Sprache.

Der Befehl besagt, daß die hiermit angesprochene Datei (bzw. deren Data Control Block, DCB) ein Fixed Block (FB) Record Format (RECFM) hat (alle Datensätze haben die gleiche Länge), dessen Länge (Logical Record Length LRECL) 80 Bytes beträgt, und daß für die Übertragung vom/zum Hauptspeicher jeweils 5 Datensätze zu einem Block von (Blocksize BLKSIZE) 400 Bytes zusammengefaßt werden.

Literatur

M.Winkler: „MVS/ESA JCL“. Oldenbourg, 3. Auflage, 1999

http://www.mainframezone.com/blog/the-it-skills-shortage-where-will-we-find-mainframe-talent

A February 2011 study conducted by SHARE entitled, "CLOSING THE IT SKILLS GAP: 2011 SHARE Survey for Guiding University & College IT Agendas", found that half of the companies surveyed hire new IT employees straight out of school, with relatively little actual work experience. The study also indicates a strong demand for mainframe skills with the finding, "In terms of platform-specific skills, companies seek applicants skilled in running two types of environments—database administration and mainframe administration. Specific mainframe administration skill areas also are in demand by a majority, or close to a majority, of companies in the survey — 55% seek mainframe administrative skills, and half are in need of skills involving JCL, or Job Control Language."

So archaisch JCL auch sein mag, es ist nach wie vor sehr populär. Nach Absolvierung des Einarbeitungs-Kulturschocks finden die Benutzer JCL sehr produktiv und mächtig.

Was ist die Funktion von „Submit“ bzw. „Sub“

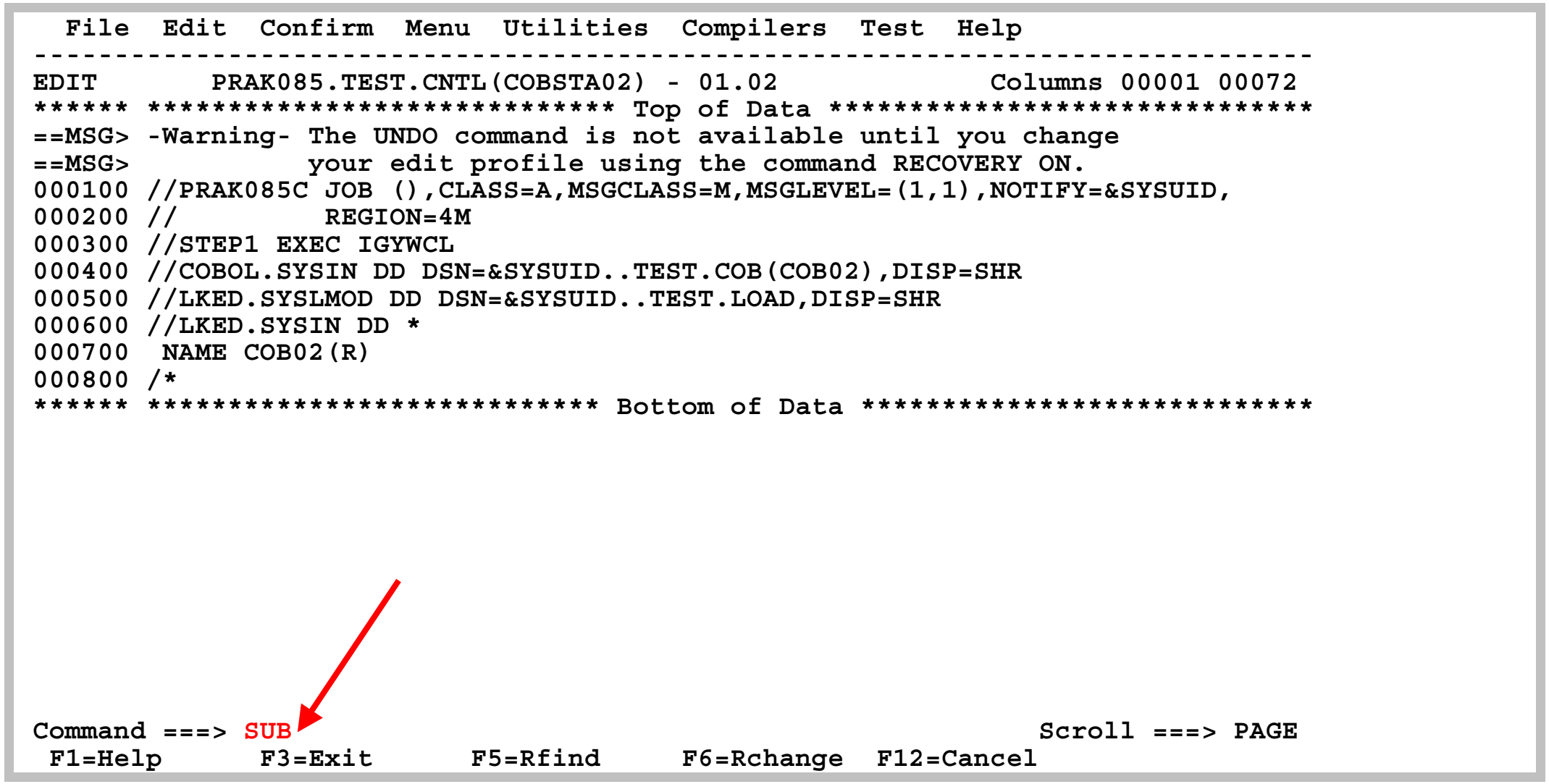
TSO bzw. ISPF sind Command Line Shells, vergleichbar mit den korn, bash Shells unter Unix/Linux bzw. der DOS Eingabeaufforderung unter Windows. Solche Shells beinhalten eine primitive Ausführungsumgebung, die es ermöglicht, von der Kommandzeile aus ein Programm (z.B. xyz.exe) .ddirekt ausführen. Dies ist auch unter der TSO Shell möglich.

Professioneller ist die Programmausführung unter einem Anwendungsserver. Viele Anwendungsserver sind Bestandteil einer Client/Server Umgebung (sog. interaktive Server). Ein Beispiele hierfür ist z.B. der Web Application Server Ihrer persönlichen Home Page, auf dem eine CGI oder Java Servlet Anwendung läuft.

Andere Anwendungsserver implementieren Stapelverarbeitung, und das JES Subsystem unter z/OS ist eine führende Implementierung. Ein JES Job wird typischerweise durch die Ausführung eines JCL Scripts gestartet, und genau dies tun Sie bei der Durchführung ihres ersten Cobol Tutoruials, bei dem Sie das unten stehende JCL Script erzeugen.

Das Kommando „SUB“ (Abkürzung für Submit) bewirkt, dass das JCL Script zwecks Ausführung an das JES Subsystem übergeben wird. JES generiert hierfür einen Job mit einer eindeutigen Job Nummer, und übergibt ihn zwecks Ausführung an einen JES „Initiator“ (siehe den folgenden Teil 2 dieses Themas) .

Der wichtigste Befehl in dem JCL Script auf der folgenden Seite ist „EXEC IGYWCL“. Er bewirkt, dass ein weiteres JCL Script mit dem Namen IGYWCL aus einer Programmbibliothek aufgerufen wird. IGYWCL enthält Anweisungen, die bewirken, dass der in der folgenden Zeile angegebene Data Set TEST.COB.(COB02) compiled und linked wird.


File Edit Confirm Menu Utilities Compilers Test Help
EDIT PRAK085.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 //PRAK085C 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

Literatur

Ramesh Krishna Reddy: JCL Tutorial:
http://www.mainframegurukul.com/srcsinc/drona/programming/languages/jcl/jcl.chapter1.html

Introduction to JCL
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/jcl/jclintro.pdf

IBM: z/OS JCL Users Guide. SA22-7598-04, July 2004
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/jcl/MvsJclUserGuide.pdf

IBM: z/OS JCL Reference. GC28-1757-09 September 2000
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/jcl/MvsJclReference.pdf