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 3

TSO und Data Sets

© copyright W. G. Spruth,

TSO

Server Zugriff

Unterschied zwischen Einzelplatzrechner und Client/Server Betriebssystemen.

Wir unterscheiden zwischen Arbeitsplatzrechnern (Klienten) und Servern. Arbeitsplatzrechner arbeiten im *Single-User-Modus* und können mit einem *Single-User-Betriebssystem* betrieben werden. Windows und CMS sind typische Single-User-Betriebssysteme. Windows fehlt die Benutzerverwaltung für den simultanen Multi-User-Betrieb. Unix und z/OS sind traditionelle Multi-User-Betriebssysteme.

Linux und Unix werden sowohl als Arbeitsplatzrechner- als auch als Server-Betriebssysteme eingesetzt. Windows Server 2008 R2 ist ein Server Betriebssystem, wlches teilweise die gleichen Komponenten wie die Windows 7 Produktfamilie benutzt. z/OS ist ein reinrassiges Server-Betriebssystem. Andere Beispiele für Server-Betriebssysteme sind i/5 (OS/400), Tandem Non-Stop und DEC OpenMVS. Unix Betriebssysteme wie Solaris, HP-UX (unter Itanium) sowie AIX werden bevorzugt als Server Betriebssysteme eingesetzt

Bei einem Server-Betriebssystem unterscheiden wir zwischen einem interaktiven zeitscheibengesteuerten und einem *Run-to-Completion-Betrieb*. Wenn mehrere Benutzer sich gleichzeitig in einen Unix-Server einloggen, geschieht dies typischerweise im interaktiven, Zeitscheiben-gesteuerten Modus. SQL-Aufrufe und Transaktionen arbeiten in vielen Fällen im *Run-to-Completion-Modus*. Run-to-Completion bedeutet den Wegfall der Zeitscheibensteuerung, was unter Performance Gesichtspunkten vorteilhaft sein kann. Hierbei ist es die Aufgabe des Entwicklers, seine Programme so zu schreiben, dass nicht ein bestinntes Programm alle Resourcen eines Rechners usurpieren kann.

Client Software

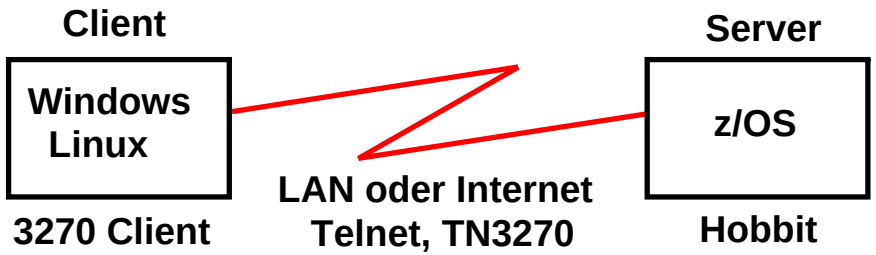
Ein Server Zugriff erfordert spezielle Client Software. Hierfür existieren drei Alternativen:

1. **Selbstgeschriebene Anwendungen:**

   Sockets, RPC, Corba, DCOM, RMI

2. **Zeilenorientierte Klienten:**

Unix Server	**Telnet, Putty Client, FTP**
z/OS Server	**3270 Client (3270 Emulator)**
VMS Server	**VT 100 Client**



3. **Klienten mit graphischer Oberfläche:**

Windows Server	**Citrix Client**
WWW Server	**Browser Client**
SAP R/3 Server	**SAPGUI Client**
Unix Server	**XWindows, Motif**
z/OS und OS/390 Server	**Servlet, JSP Client**

Der 3270 Client, meistens als 3270 Emulator bezeichnet, ist der einfachste Klient für den Zugriff auf einen Mainframe Sever. Unter http://jedi.informatik.uni-leipzig.de/de/access.html können Sie einen kostenlosen 3270 Emulator herunterladen.

TSO („Time Sharing Option“)

TSO ist eine z/OS zeilenorientierte Shell, vergleichbar mit der Windows „DOS Eingabeaufforderung“ oder den Unix/Linux Bourne, Korn, Bash oder C-Shells.

TSO wird hauptsächlich für angewendet für:

- **Software Entwicklung und Test,**
- **System Administration.**

Systemadministratoren (Systemprogrammierer) verwenden TSO um Files zu editieren, Steuerungen vorzunehmen, Systemparameter zu setzen und einen Job Status zu überprüfen.

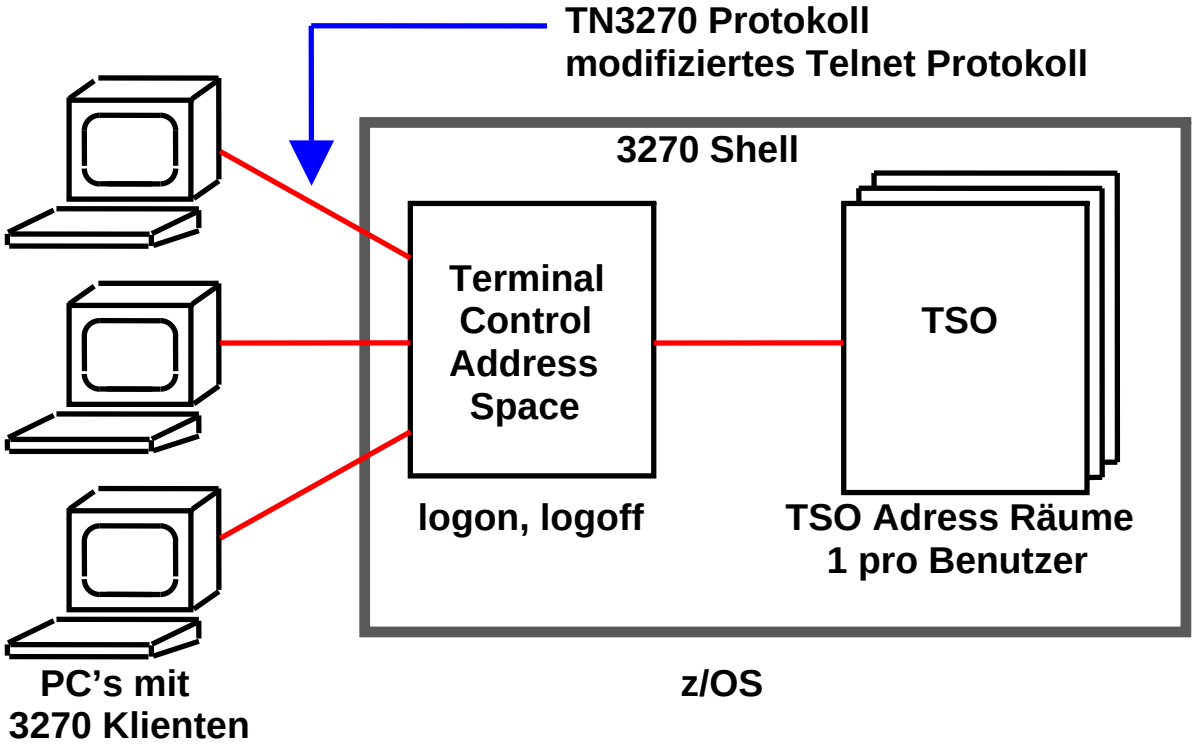
ISPF ist eine wichtige Erweiterung / Ergänzung für die einfache TSO Shell. Hierbei wird auf dem Klienten ein Bildschirm wiedergegeben, der neben der Kommadozeile eine Auflistung der möglichen Kommandos enthält. IBM nennt dies „Full Screen Mode“ und bezeichnet derartige Bildschirminhalte als Panels.

Daneben enthält ISPF einen Bildschirm Editor, den ISPF Editor. Der Editor lässt sich mit dem Unix/Linux vi editor vergleichen. Beide Editoren sind hoffnungslos inkompatibel, kryptisch, schwer erlernbar, sehr mächtig, und werden von ihren Benutzern heiß und innig geliebt.

TSO („Time Sharing Option“)

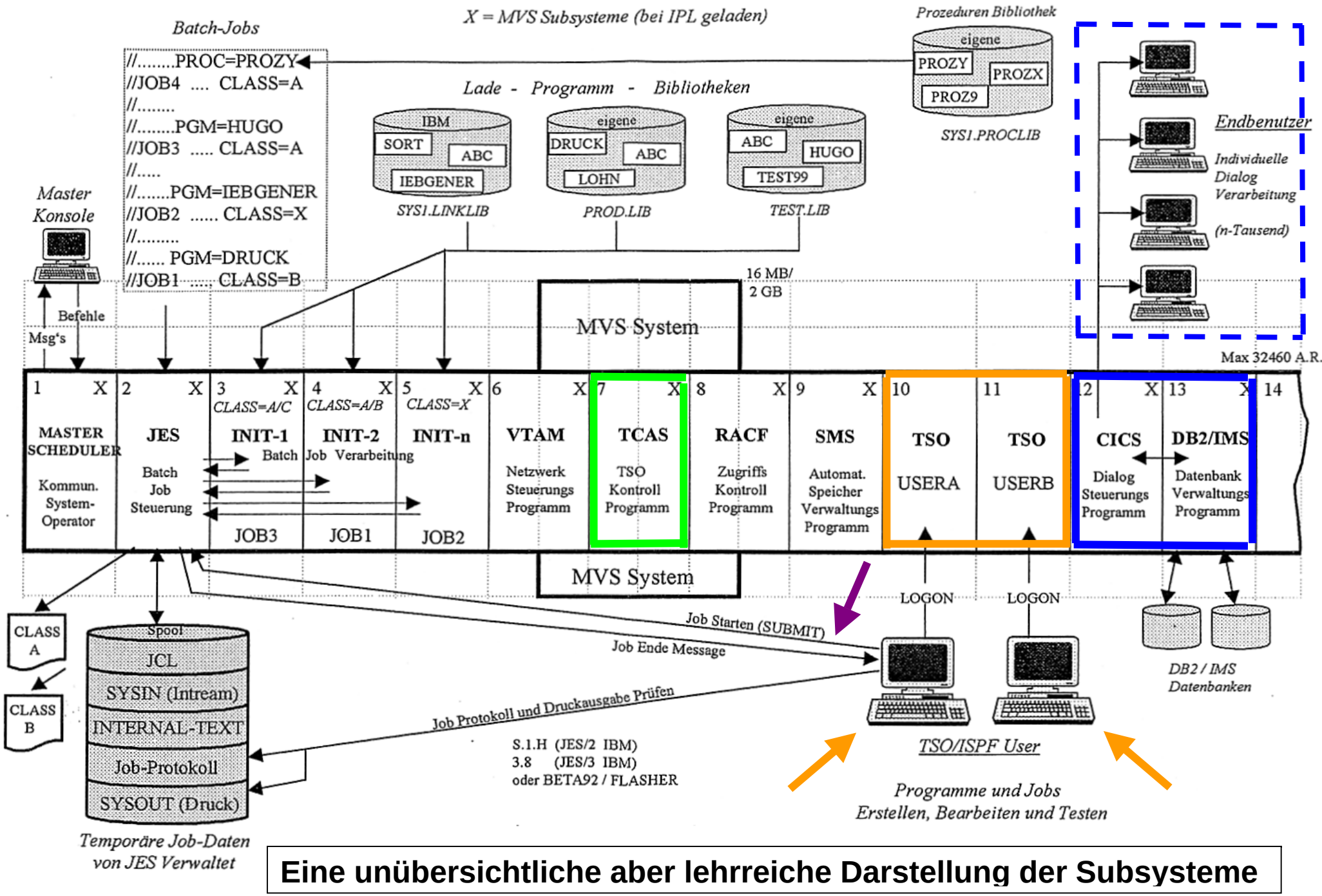
Die TSO Shell ist vergleichbar mit den UNIX Time Sharing Shells (korn, bash, ...)

Es ist eine „Full Screen“ und eine Command Level Schnittstelle verfügbar. Die Full Screen Komponente wird als ISPF (Interactive System Productivity Facility) bezeichnet.



Der TSO Terminal Control Prozess (Terminal Control Access Space, TCAS) arbeitet in einem eigenen Adressenraum. Er nimmt Nachrichten von den einzelnen TSO Klienten entgegen und leitet sie an den für den Benutzer eingerichteten separaten TSO Adressenraum weiter.

Für jeden neuen TSO Benutzer wird beim login von TCAS ein eigener (virtueller) Adressenraum eingerichtet.



Eine unübersichtliche aber lehrreiche Darstellung der Subsysteme

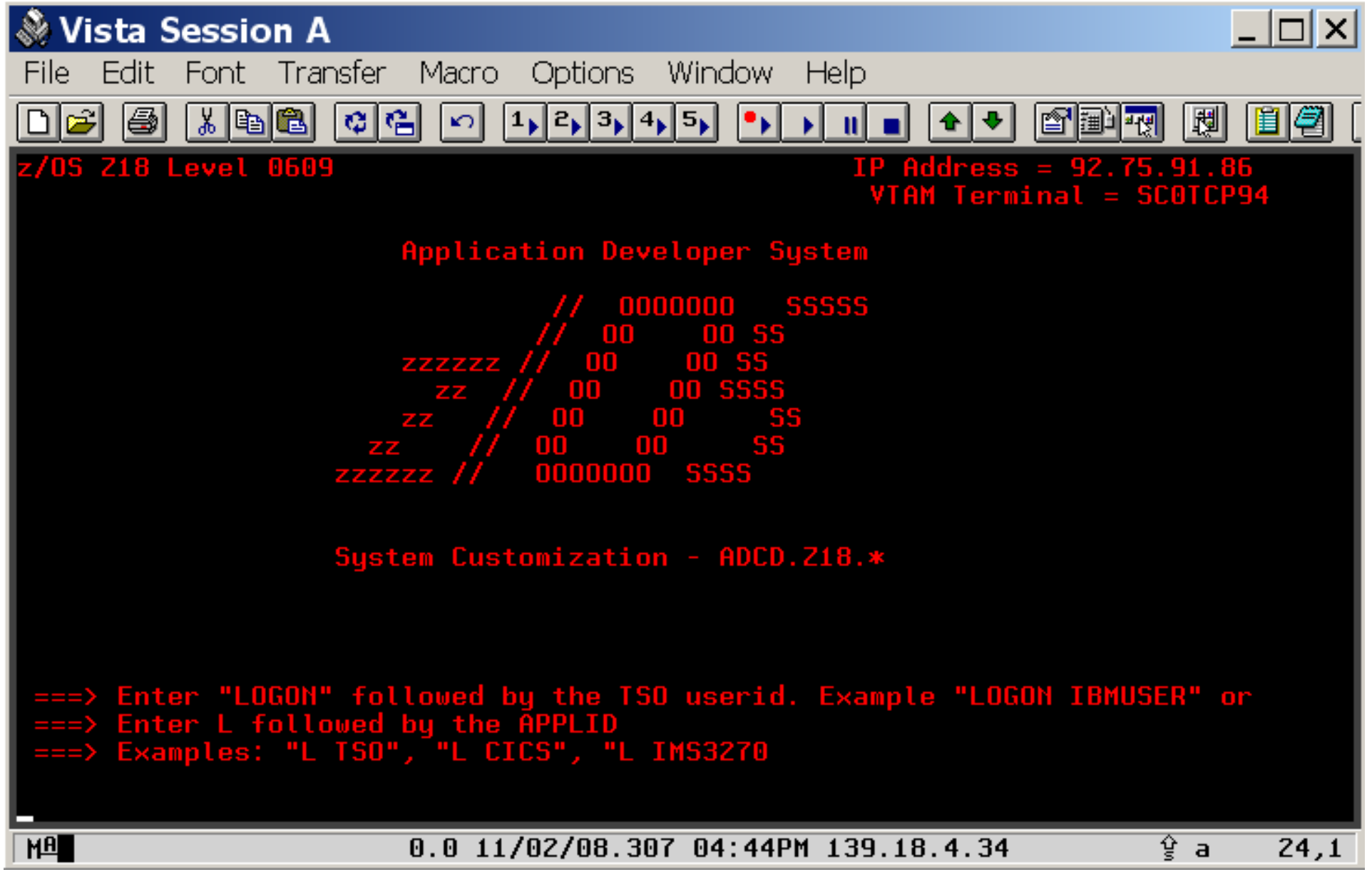
Eine unübersichtliche aber lehrreiche Darstellung der Subsysteme

Die obige Abblidung zeigt ein z/OS System mit einer größeren Anzahl von Subsystemen, die jeweils in eigenen virtuellen Adressräumen (Regions) laufen.

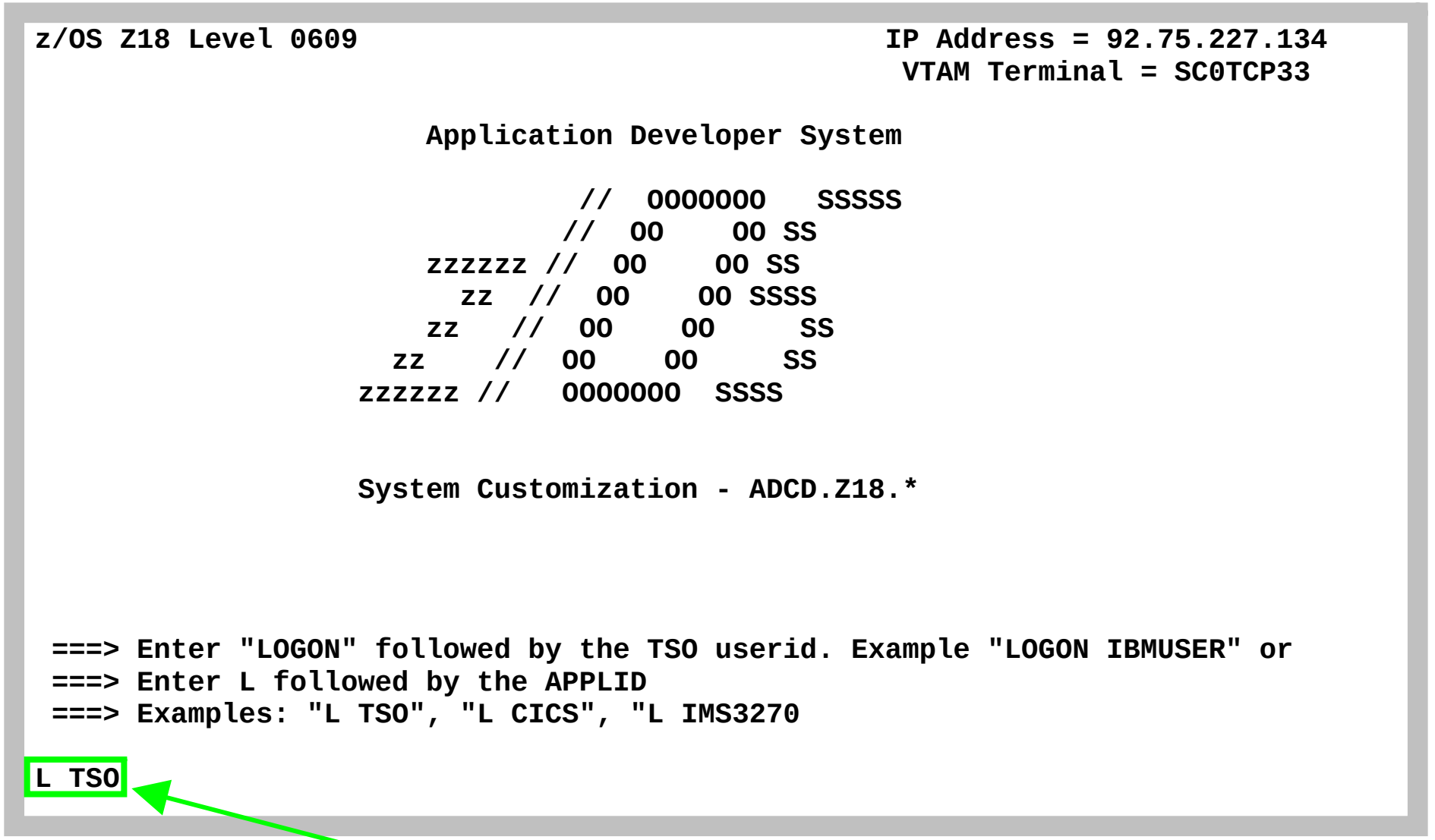
- **Der TSO Terminal Control Access Space(TCAS) wird beim Hochfahren von z/OS gestartet.**
- **Wenn immer ein Benutzer sich in TSO einlogged, richtet TCAS für ihn einen eigenen TSO Address Space eingerichtet.**
- **In dem hier gezeigten Beispiel haben sich 2 Benutzer (User A und User B) jeweils mit ihrem eigenen Bildschirm Terminal eingelogged. Jeder der beiden TSO Benutzer hat nur Zugriff zu seinem eigenen Address Space.**
- **Der linke der beiden TSO Benutzer übergibt gerade an JES einen Stapelverarbeitungsauftrag.**
- **Parallel dazu existieren mehrere Address Spaces für (in diesem Beispiel) eine Kombination der CICS und der DB2 Subsysteme. CICS Programme kommunizieren intern direkt mit DB2 Programmen.**
- **Parallel zu den TSO Benutzern haben sich 4 Benutzer mit Ihren Bildschirm Terminals in das CICS Subsystem eingeloggend. Während bei TSO für jeden Benutzer ein eigener Address Space eingerichtet wird, existiert für alle Benutzer (plus dem CICS Subsystem selbst) nur ein einziger Address Space.**

Dargestellt sind weiterhin virtuelle Adressenräume für den Master Scheduler, JES mit 3 Initiators, sowie für VTAM, RACF und SMS Subsysteme. Der Master Scheduler fährt z/OS hoch, steuert den laufenden Betrieb, und ermöglicht es einem Administrator mittels einer Master Konsole den laufenden Betrieb zu beobachten und in ihn einzugreifen. VTAM ist Teil des Communication Subsystems, RACF ist das Security Subsystem, und SMS ist weiter unten erläutert.

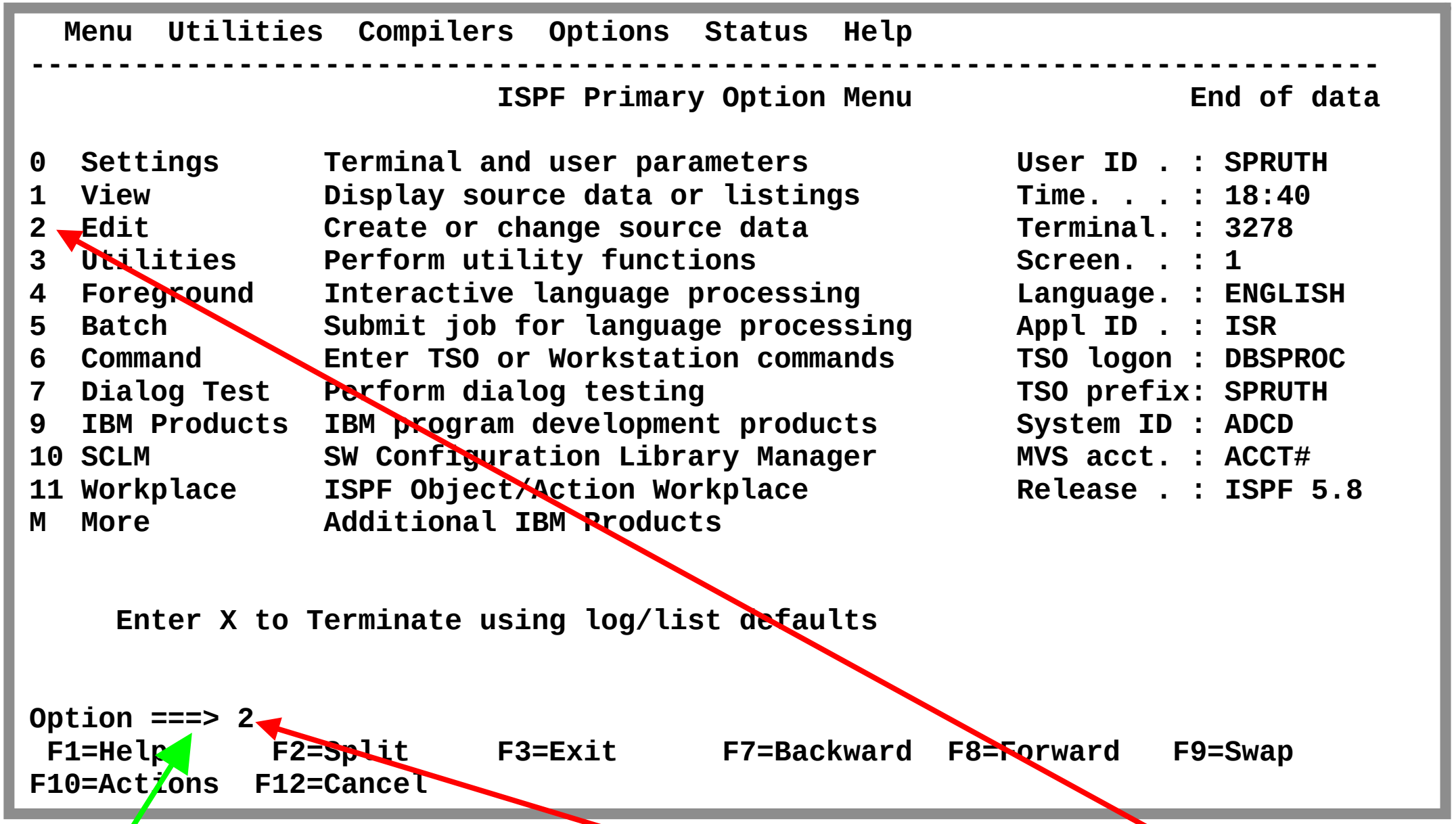
JES holt sich ein Script aus seiner Procedure Bibliothek. Die Initiators laden Programme aus Programmbibliotheken.



Diesen Welcome Screen sehen sie, wenn Sie sich in unseren Mainframe Rechner jedi.informatik.uni-leipzig.de oder 139.18.4.30 einloggen. Von hier aus können Sie sich in eins von mehreren Subsysten, wie z.B. TSO oder CICS einloggen.



Hier wird gerade das Logon (abgekürzt L) für das TSO Subsystem eingegeben.



===> ist das Cursor Symbol. Eingabe einer „2“ hinter dem Cursor ruft den „ISPF“ Editor auf.

Detaillierte Screen by Screen Tutorials sind zu finden unter http://jedi.informatik.uni-leipzig.de

Datasets

Dateisystem

Betriebssysteme wie Unix, Linux oder Windows speichern Daten in Files (Dateien). Die Identifizierung von Dateien erfolgt über Dateiverzeichnisse, die selbst wie Dateien aussehen und wie Dateien behandelt werden können.

Ein Dateisystem (File System) verbirgt die Eigenschaften der physischen Datenträger (Plattenspeicher, CD, evtl. Bandlaufwerke) weitestgehend vor dem Anwendungsprogrammierer. Für ein Betriebssystem existieren typischerweise mehrere inkompatible File Systeme, z. B. FAT32 oder NTFS unter Windows, oder extf2, extf3 und Reiser unter Linux.

Unter Unix, Linux und Windows sind Dateien strukturlose Zeichenketten (unstrukturierte Menge von Bytes), welche über Namen identifiziert werden. Hierfür dienen Dateiverzeichnisse, die selbst wie Dateien aussehen und behandelt werden können.

Die meisten z/OS Dateien haben im Gegensatz zu Linux oder Windows Dateien zusätzliche Struktureigenschaften, und werden dann als Data Sets bezeichnet. Viele Data Sets bestehen aus Records. Hier ein Record Beispiel: Angenommen eine Datei, die das Telefonverzeichnis einer Stadt speichert. Die Datei besteht aus vielen Records (Einträgen). Jeder Record besteht aus Feldern, z.B. Nachnahme, Vornahme, Straße, Haus Nr, Tel. Nr. usw.

Im Gegensatz zu einer Unix File speichert eine Unix SQL Datenbank die Daten strukturiert in der Form von Tabellen (Tables), Relationen, Rows, Columns usw. Die Rows innerhalb einer Unix SQL Datenbank Tabelle enthalten Records, im Gegensatz zu Unix Files.

Datenspeicherung unter z/OS

z/OS verwendet gleichzeitig mehrere Filesysteme. Eine z/OS „Access Method“ definiert das benutzte Filesystem und stellt gleichzeitig Routinen für den Zugriff auf die Records des Filesystems zur Verfügung. Das Filesystem wird durch die Formattierung eines physischen Datenträgers definiert. Bei der Formattierung der Festplatte wird die Struktur des Datasets festgelegt. Es gibt unter z/OS unterschiedliche Formattierungen und Zugriffsmethoden für Datasets mit direktem Zugriff , sequentiellen Zugriff oder index-sequentiellen Zugriff.

Genauso wie bei Unix existieren eine ganze Reihe von File Systemen (Access Methods) mit Namen wie BSAM, BDAM, QSAM, ISAM usw. Die wichtigste Access Method (File System) hat den Namen VSAM (Virtual Storage Access Method).

Dataset Zugriffe benutzen an Stelle eines Dateiverzeichnisse „Kontrollblöcke“, welche die Datenbasis für unterschiedliche Betriebssystemfunktionen bilden. Die Kontrollblöcke haben exotische Namen wie VTOC, DSCB, DCB, UCB, deren Inhalt vom Systemprogrammierer oder Anwendungsprogrammierer manipuliert werden können.

Unter z/OS werden Datenbanken ebenfalls in Data Sets abgespeichert. Eine Datenbank ist also eine höhere Abstraktionsebene als ein Data Set. Während DB2 unter z/OS hierfür normale z/OS Data Sets verwendet, benutzen Unix Datenbanken hierfür häufig Files mit proprietären Eigenschaften sowie direkte (raw) Zugriffe auf Dateien.

Blocks, physische und logische Records

In der Anfangszeit von OS/360 wurde bei einem Plattenspeicherzugriff ein einzelner Record zwischen Plattenspeicher und Hauptspeicher transportiert. Dies ist nicht sehr effektiv, weil Plattenspeicherzugriffe relativ lange dauern (häufig mehr als 10 ms). Records sind meistens nicht sehr lang; Sie würden überrascht sein, wie populär heute noch eine Recordlänge von 80 Bytes ist (in Anlehnung an die 80 Spalten der IBM Lochkarte).

Man ging dazu über, bei jedem Plattenspeicherzugriff eine Gruppe von nebeneinanderliegenden Records auszulesen, in der Hoffnung, dass das Anwendungsprogramm demnächst einen benachbarten Record brauchen würde, der sich dann schon im Hauptspeicher befand. Besonders bei der seqentiellen Verarbeitung von Datenrecords ist das sehr effektiv.

Man bezeichnete diese Gruppe von gleichzeitig ausgelesenen Records als Block oder physischen Record, im Gegensatz zum eigentlichen Record, der dann als logischer Record bezeichnet wird.

Ein Block (physischer Record) besteht also aus mehreren logischen Records, welche die eigentlichen Verarbeitungseinheiten darstellen. Die Blockungsgröße definiert, wie viele logischen Records in einem physischen Record (Block) untergebracht werden können.

Wenn man heute von Records redet, meint mann damit meistens logische Records.

Benutzung eines z/OS Systems

Wenn sie sich das erste Mal unter TSO einloggen ist Ihre allererste Aufgabe die Zuordung (allocate) von Data Sets.

Ihre Frage sollte sein: Was soll das ? „Das habe ich unter Windows noch nie gemacht“.

Dies ist falsch !!! Wenn Sie unter Windows eine neue Datei erstmalig anlegen, müssen Sie z. B. entscheiten, ob sie unter C: oder D: gespeichert wird. Wenn Ihr Rechner über 24 Plattenspeicher verfügt, z.B.

C:, D:, E:, ...Z: ,

wird die Verwaltung schon etwas schwieriger. Bei der Auswahl kann auch sein, dass einige der Platten (welche ?) mit FAT32 und andere mit NTFS formattiert sind, dass sie unterschiedliche Kapazität und/oder Performance Eigenschaften haben , und dass dies für Ihre Entscheidung relevant sein mag. Stellen Sie sich vor, Ihr Rechner hat eine normale Festplatte und zusätzlich eine besonders schnelle, aber kleine, Solid State Disk.

Was machen Sie wenn Ihr z/OS Rechner über 60 000 Plattenspeicher verfügt, dazu 1500 Solid State Disks ?

Sie verwenden für die Verwaltung eine z/OS Komponente Storage Management System (SMS). . Wenn Sie Platz für eine neue Datei brauchen, melden Sie dies bei SMS an. Dieser Vorgang wird als Allocation bezeichnet.

Genau genommen besteht das z/OS Storage Management System aus mehreren Komponenten. Die wichtigste dieser Komponenten heißt DFSMS (Data Facility Storage Management System).

Die Menge aller von SMS verwalteten Data Sts wird als „System Managed Storage“ bezeichnet. In der Vergangenheit war es möglich, Data Setz auch ohne Benutzung von SMS zu verwalten. Das ist heute unüblich.

Erstellen einer Datei: Entscheidungen

Benutzer, die Datenverarbeitung betreiben, haben beim Umgang mit den Daten täglich viele Entscheidungen zu treffen. Zusätzlich zum Umgang mit Themen, die nur die Daten oder die Anwendung betreffen, müssen sie die Speicherverwaltungsmaßnahmen der Installationen kennen. Sie müssen sich außerdem mit den Themen auseinandersetzen, die Format, Bearbeitung und Positionierung der Daten betreffen:

- **Welchen Wert soll die Blockgröße haben? Wieviel Speicherplatz ist erforderlich?**
- **Welcher Einheitentyp soll verwendet werden? Sollen die Daten in den Cache geschrieben werden? Soll eine Fehlerbehebung durchgeführt werden?**
- **Wie oft soll eine Sicherung oder Migration durchgeführt werden? Soll die Sicherung/Migration erhalten bleiben oder (wann ?) gelöscht werden?**
- **Welche Datenträger stehen für die Dateipositionierung zur Verfügung?**

Wenn die Verwendung von Datenverarbeitungsservices vereinfacht werden soll, müssen dem System einfachere Schnittstellen zur Verfügung gestellt werden. Insbesondere JCL ist einer der Bereiche, in denen Vereinfachungen vorgenommen werden.

DFSMS Klassen

Data Facility Storage Management System

Angesichts der unübersichtlich großen Anzahl von Plattenspeichern, File Systemen und Data Sets werden letztere in Klassen eingeteilt. Beim Neuanlegen eines Data Sets müssen angegeben werden:

Data Class

Eine Data Class fasst Data Sets mit identischen File System Attributen wie Record Format, Record Länge, Schlüssellänge bei VSAM Dateien, usw zusammen.

Storage Class

Eine Storage Class fasst mehrere Plattenspeicher mit identischen Performance Eigenschaften wie Antwortzeit (ms), Nutzung von Read/Write Caching, Verwendung der Dual Copy Funktion usw. zusammen. Beispielsweise könnte man z.B alle Solid State Disks zu einer Storage Class zusammenfassen

Management Class

Eine Management Class fasst mehrere Plattenspeicher zu einer Gruppe zusammen, wenn diese identisch verwaltet werden. Verwaltungseigenschaften sind z.B. Migration nach x Tagen, Anzahl Backup Versionen, schrittweiser Abbau der Backup Versionen, Löschen der Daten, ...

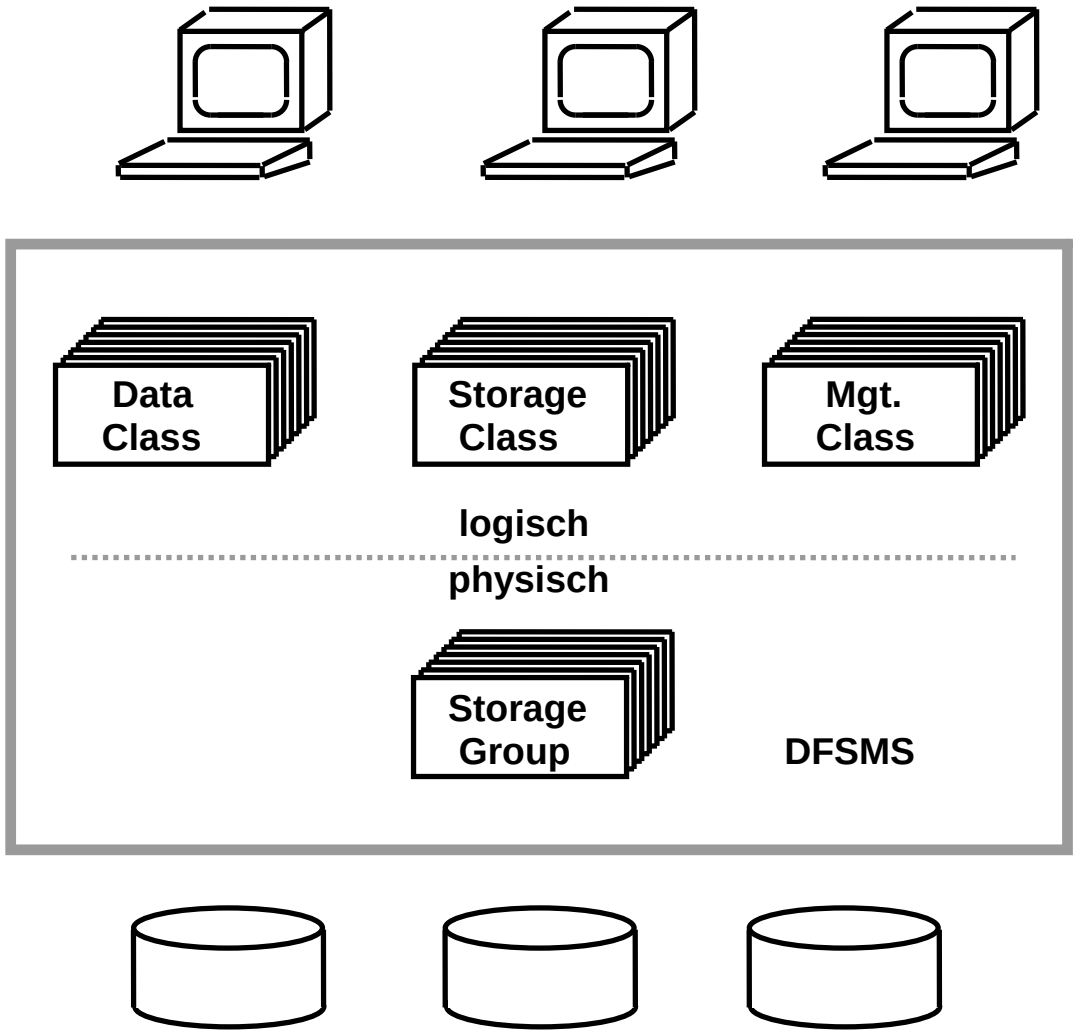
DFSMS bewirkt die Zuordnung eines Data Sets zu einer Storage Group, einer Gruppe von Plattenspeichern (Volumes) in der gleichen Data Class, Storage Class und Management Class.

Lebenszyklus eines Data Sets

Alle Dateien und Data Sets in einem Unternehmen haben einen Lebenszyklus: Sie entstehen irgendwann, werden verarbeitet und geändert und werden irgendwann wieder gelöscht. Das Was, wie und Wo ist in der Management Class festgelegt.

Es werden beispielsweise Eigenschadften wie die Folgenden festgelegt:

- **Das Anlegen (allocate) der Datei erfolt durch den Benutzer**
- **Benutzung (Schreiben und/oder Lesen der Daten)**
- **Es werden Sicherungskopien angelegt**
- **Angaben zur Platzverwaltung (wann wird der Data Set freigegeben / erweitert / komprimiert)**
- **Auslagern von inaktiven Dateien, sowie Wiederbenutzung**
- **Ausmustern von Sicherungskopien**
- **Wiederherstellung von Dateien**
- **Löschen der Datei**

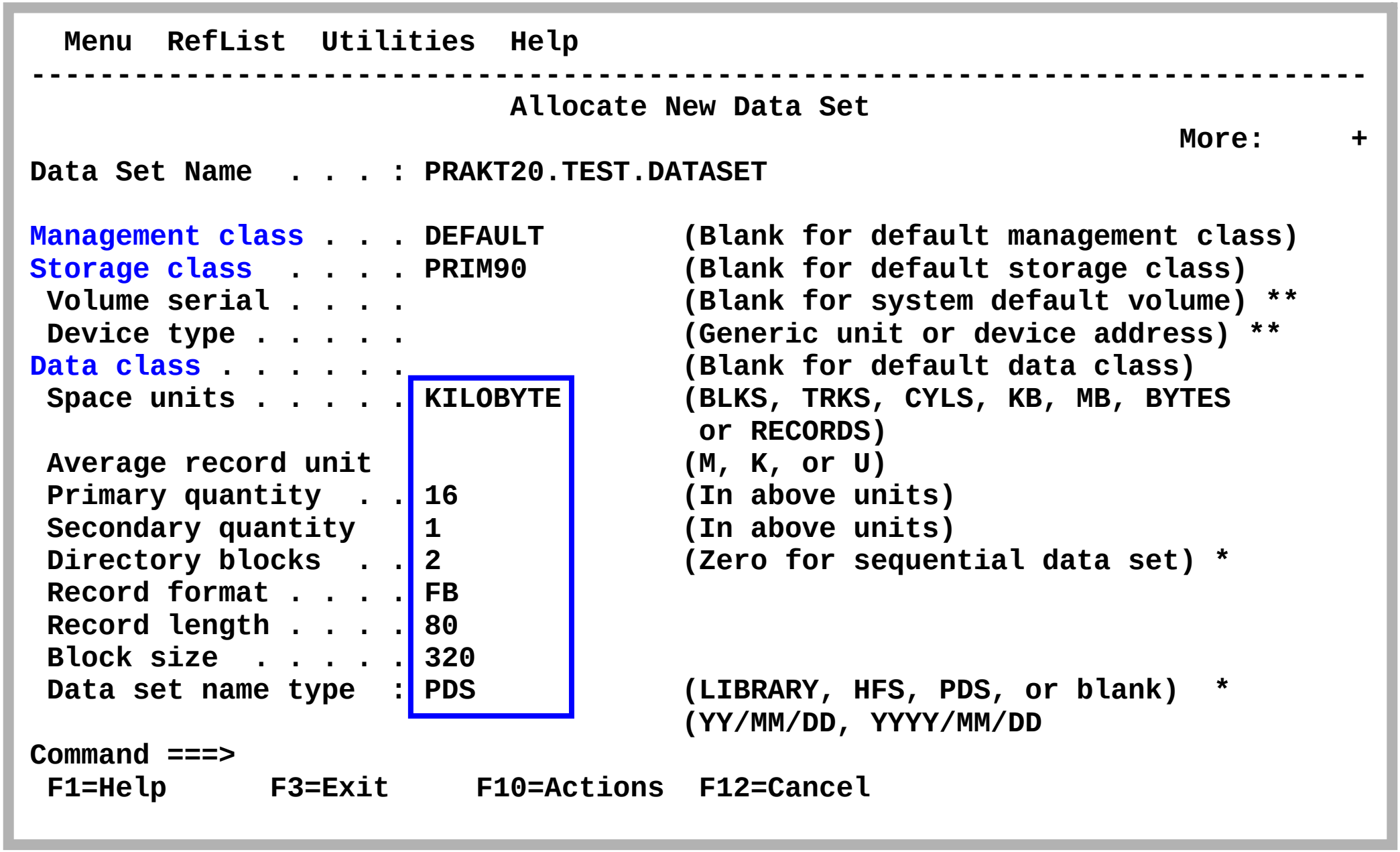


Der Benutzer sieht nur die logische Sicht der Daten:

- **vorbereitete Datenmodelle in den Datenklassen**
- **in den Storageklassen festgelegte Serviceanforderungen**
- **Management Kriterien für die Auslagerung der Daten**

Plattenspeicher mit der gleichen Data Class, Storage Class und Management Class werden zu einer Storage Gropup zusammengefasst.

Data Facility Storage Management System


Menu RefList Utilities Help
Allocate New Data Set
More: +
Data Set Name . . . : PRAKT20.TEST.DATASET
Management class . . . DEFAULT (Blank for default management class)
Storage class . . . . PRIM90 (Blank for default storage class)
Volume serial . . . . (Blank for system default volume) **
Device type . . . . . (Generic unit or device address) **
Data class . . . . . (Blank for default data class)
Space units . . . . KILOBYTE (BLKS, TRKS, CYLS, KB, MB, BYTES or RECORDS)
Average record unit (M, K, or U)
Primary quantity . . 16 (In above units)
Secondary quantity 1 (In above units)
Directory blocks . . 2 (Zero for sequential data set) *
Record format . . . . FB
Record length 80
Block size 320
Data set name type : PDS (LIBRARY, HFS, PDS, or blank) *
(YY/MM/DD, YYYY/MM/DD
Command ===>
F1=Help F3=Exit F10=Actions F12=Cancel

Allocating a new Data Set.

Die oben gezeigte Abbildung gibt den Bildschirminhalt (Screen) wieder, mit dem Sie das Allocating eines Data Sets vornehmen.

In dem gezeigten Beispiel ist die Management Class mit „Default“ vorgegeben, desgleichen die Storage Class mit PRIM90.

Bei der Data Class haben Sie die Auswahl, die gwünschte Größe in Bytes, Kilobytes, Megabytes, Records, Blöcken, Plattenspeicher-Spuren (Tracks) oder Zylindern anzugeben.

Die angegebene Größe ist 16 KByte (Primary quantity) mit einer Reserve von 1 KByte (Secondary Quantity) . Elemente innerhalb des Data Sets können mit 2 Directory Blocks adressiert werden.

Alle Records haben die gleiche Länge (Fixed Block, FB). Die Länge der logischen Records (Record length) beträgt 80 Bytes. Vier logische Records werden zu einem physischen Record zusammengefasst (Block size = 4 x 80 Bytes, = 320 Bytes).

Es existieren eine ganze Reihe von unterschiedlichen File Systemen für z/OS Data Sets. (z/OS bezeichnet die File Systeme als „Access Methods“). Das hier gewählte File System ist PDS (was immer das auch sein mag, wir schauen es uns später an).