COMP 3400 Mainframe Administration[1]

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

[1]These slides are based in part on materials provided by IBM's Academic Initiative.

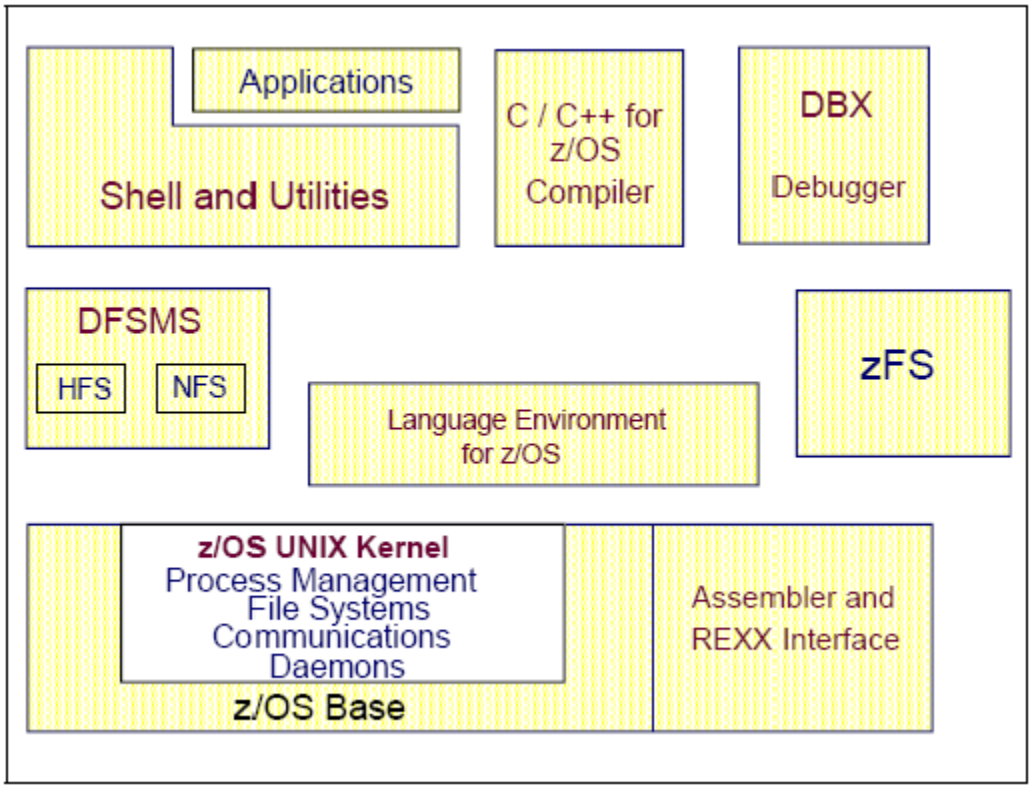
Today

- z/OS: Introduction to the most widely used mainframe OS
- Interacting with z/OS (TSO, ISPF, UNIX)

z/OS Fundamentals

- Multiprogramming (many programs) and Multiprocessing (many processors) capable
- OS 360 / MVS heritage
- UNIX support (POSIX 1003.2)
- 64-bit operating system

z/OS overview



z/OS vs. UNIX

Quite a few concepts are common to both, but have different names:

UNIX	z/OS
boot	IPL
Files	data sets
vi, ed, sed	ISPF/oedit
telnet login	TSO logon

z/OS vs. UNIX

Quite a few execution-related concepts have close relatives:

UNIX	z/OS
Process	Address Space
PID	ASID
Thread	Task
`pthread_t`	TCB/SRB

LOGON

- Use x3270 with `zos.kctr.marist.edu:1023`
- Specify "L TSO" (for now)
- Then use your username and password
- ISPF will automatically start (exit using F3)
- Use LOGOFF to log off (from TSO)

TSO Logon Procedures

On the LOGON screen, you can select a logon procedure.

IKJACCNT Minimal initialization; Marist default (only one for us)

ISPFPROC z/OS default providing ISPF panels for RACF, SDSF, SMP/E, HCD and ISMF

OMVSPROC Starts OMVS Shell directly

DBSPROC ISPFPROC plus DB2 and QMF panels

Interacting with z/OS

- Time Sharing Option/Extensions (TSO/E): allows users to logon and use a limited set of basic commands; this is sometimes called using TSO in its *native mode*
- Interactive System Productivity Facility (ISPF): menu-driven interface for TSO used by most users
- UNIX shell: omvs is the z/OS UNIX shell

TSO in native mode

- TSO prompts the user with the string “READY”
- You can start EXEC scripts using “EXEC”
- You can start ISPF using “ispf”
- You can start OMVS using “omvs”
- You can LOGOFF using “logoff”
- More on TSO and ISPF later!

Starting a UNIX shell

- From TSO: type "omvs"
- From ISPF: select option "6", then enter "omvs"
- Directly: `telnet zos.kctr.marist.edu` (use your usual username and password)

$\Rightarrow$ Welcome back to the world of COMP 2400...

- Return to ISPF/TSO using "exit"

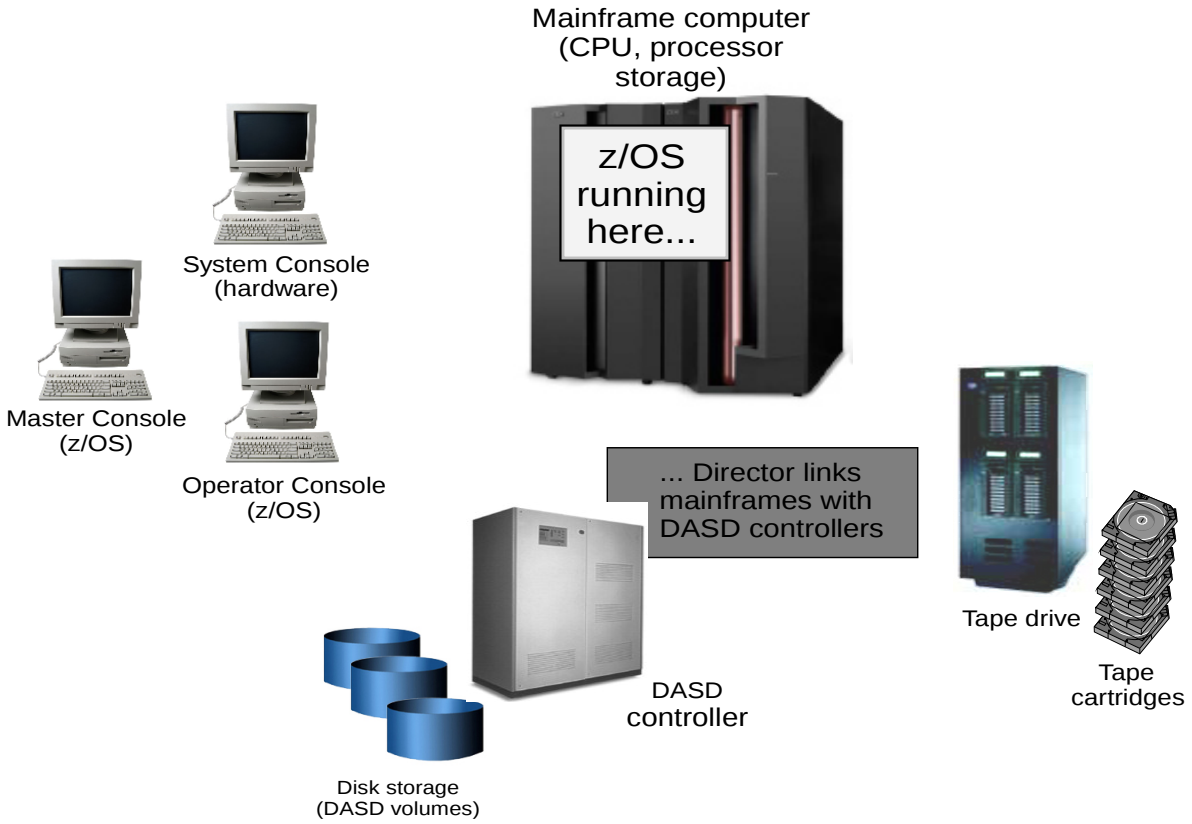
z/OS Design Goals

- Serving 1000s of users concurrently
- I/O intensive computing
- Processing very large workloads
- Running mission critical applications securely

Key z/OS features

- Can process a large number of concurrent batch jobs, with automatic workload balancing
- Can manage mixed workloads.
- Provides extensive recovery facilities, making unplanned system restarts very rare.
- Can manage large I/O configurations of 1000s of disk drives, automated tape libraries, large printers, networks of terminals, etc.

Hardware resources managed by z/OS



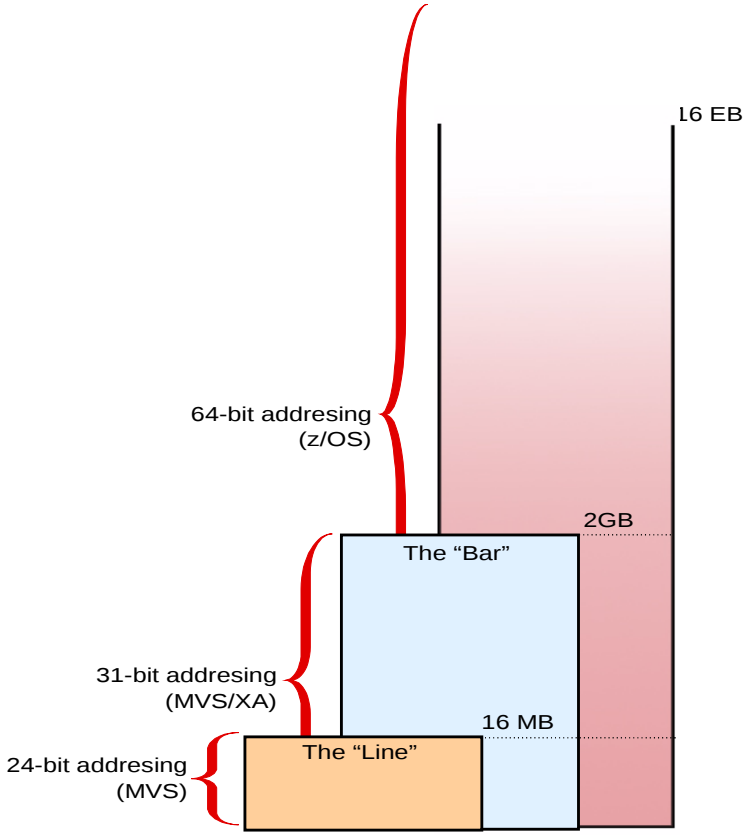
Virtual storage

- z/OS knows three kinds of storage (memory!): real storage, auxiliary storage and virtual storage
- Virtual storage is created through z/OS management of real storage and auxiliary storage through tables
- The running portions of a program are kept in *real storage*; the rest is kept in *auxiliary storage*

Address spaces

- An *address space* is the range of addressable virtual storage to a user or program or operating system
- Each user (or separately running program) is represented by an address space; each user gets a limited amount of "private" storage
- There is at least one address space per job.
- There can be multiple tasks (TCBs & SRBs) per AS

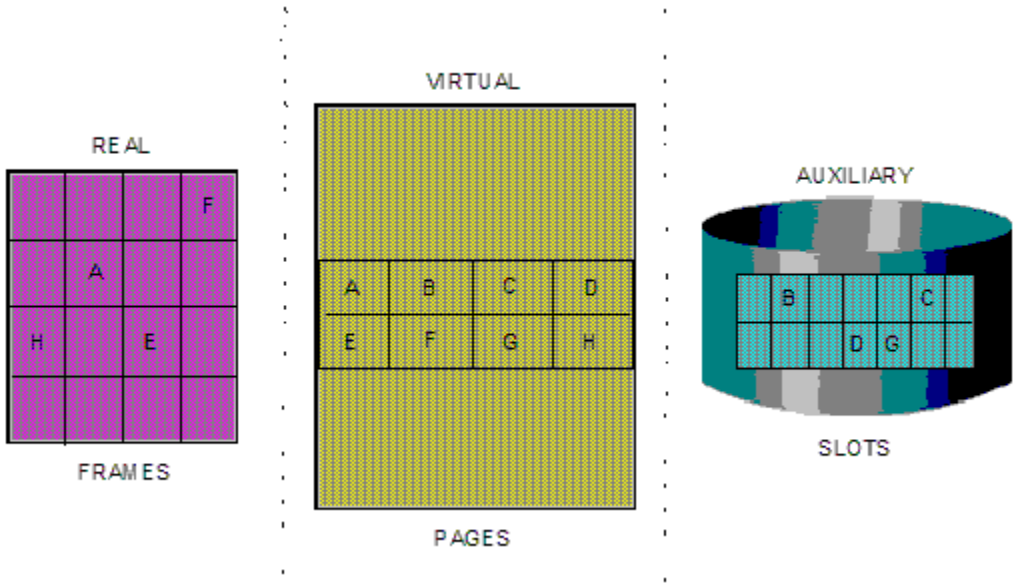
Historical Artefacts



More Terms

- Virtual storage is divided into 1 MB *segments*, composed of 4 KB *pages*; z/OS uses segment and page tables to track these
- Transfer of pages between auxiliary storage and real storage is called *paging* (and can be triggered by *page fault* interrupts)
- A *frame* is a 4K piece of real storage
- A *slot* is a 4K record in a page data set
- Translation of virtual addresses to actual physical addresses is called *dynamic address translation* (DAT)

Pages, Frames and Slots illustrated



Page Stealing

z/OS tries to always have some real storage frames on hand; pages that have not been accessed for a relatively long time are good candidates for *page stealing*.

z/OS uses various storage managers to track storage:

- Auxiliary Storage Manager (ASM)
- Real Storage Manager (RSM)
- Virtual Storage Manager (VSM)

Swapping

- *page stealing* is moving a single page into auxiliary storage
- The RSM can be used for *page fixing* which makes frames **unavailable** for stealing.
- *swapping* is moving an entire address space into auxiliary storage (preventing its processes from executing!)
- A *logically-swapped out* address space is inactive; its pages are ready to be swapped out but have not yet been swapped out

Translation Lookaside Buffer (TLB)

- When resolving a virtual address, the memory management hardware must consult the page tables to translate it to a physical address

$\Rightarrow$ This is a costly and frequent operation

- TLBs are used to cache the results of recent address translations

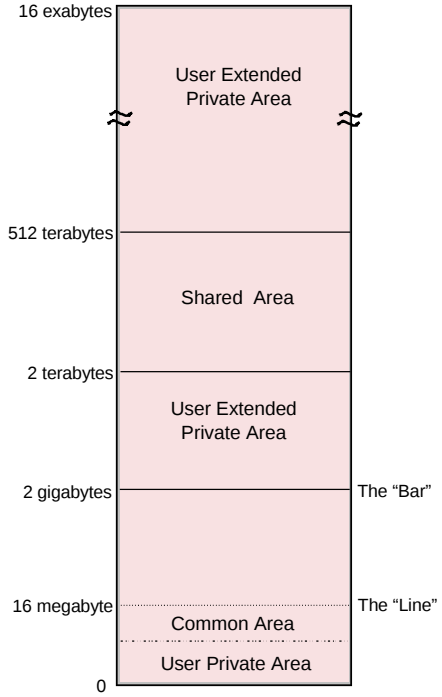
$\Rightarrow$ Faster translation!

What is in an address space?

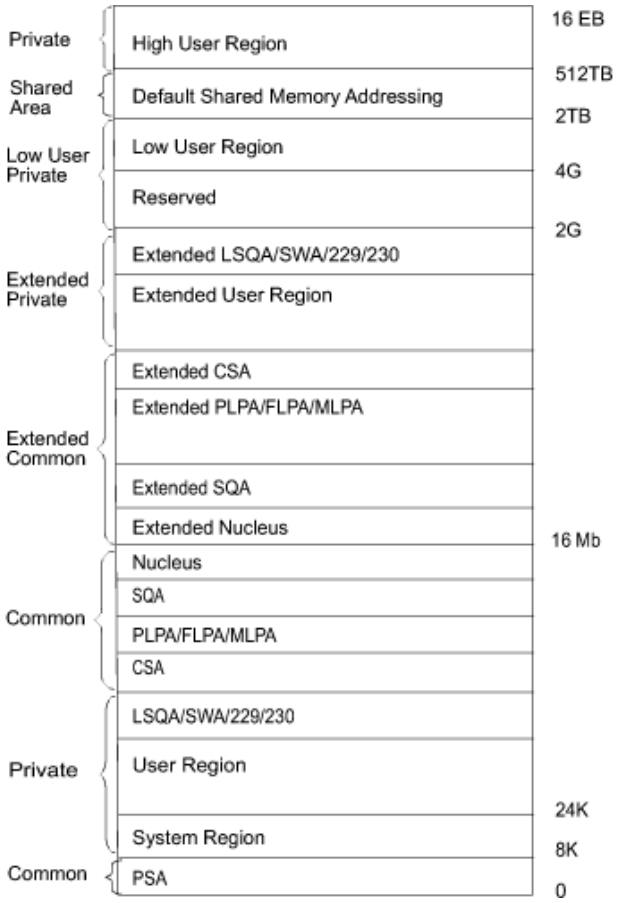
- Each user has a unique address space and z/OS maintains the distinction between the programs and data belonging to each address space.
- Because it maps all of the available addresses, however, an address space includes $system$ code and data as well as $user$ code and data.

$\Rightarrow$ Not all of the mapped addresses are available for user code and data!

64-bit address space map



Detailed 64-bit address space map



Address Space Sizes

24 bit 16 MegaByte

31 bit 2 GigaByte

32 bit 4 GigaByte

64 bit 16 ExaByte

z/OS address spaces

z/OS and its related subsystems require address spaces of their own to provide a functioning operating system:

- System address spaces are started after initialization of the master scheduler. These address spaces perform functions for all the other types of address spaces that start in z/OS.
- Subsystem address spaces for major system functions and middleware products such as DB2, CICS, and IMS.
- TSO/E address spaces are created for every user who logs on to z/OS
- Address spaces for every batch job that runs on z/OS.

Storage Management

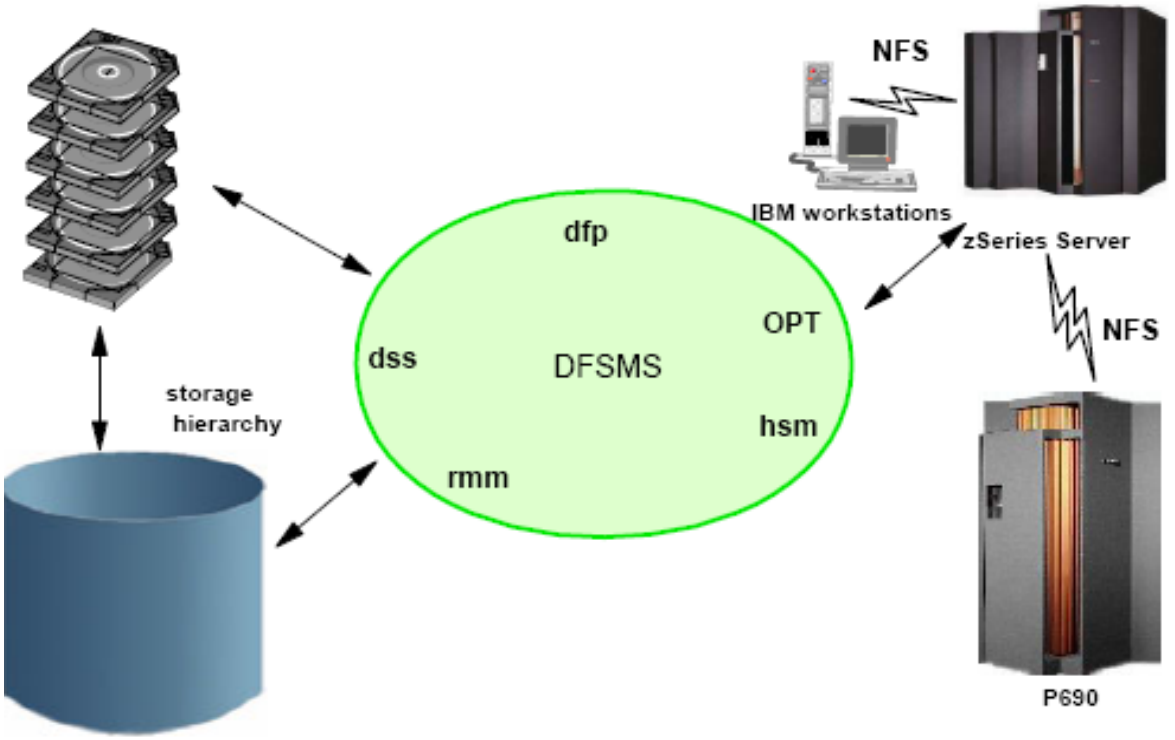
Storage management functions include:

- Allocation
- Placement
- Monitoring
- Migration (& Recall)
- Backup & Recovery
- Deletion

Storage Management in z/OS

- Typical production systems use combination of manual and automated processes for managing storage
- Users/programs can directly control many aspects of z/OS storage use
- The primary means of managing storage in z/OS is through DFSMS

Data Facility Subsystem Managed Storage (DFSMS)



Disk Space Allocation

Various units are used for disk space allocation:

- Cylinders (1 CYL $\approx$ 0.85 MB) contain tracks
- Tracks (usually 15 per CYL $\approx$ 56 KB) contain blocks
- Blocks (size depends on data set) contain records
- Records (size depends on data set) contain bytes
- Bytes

Data Sets

z/OS files are called data sets; you need to allocate space for them before you can write to them. Allocation functions require:

- Volume serial – six character name of disk volume
- Device type – type of disk (usually 3390)
- Organization – method of processing the data set (sequential)
- Record format – data is stored in *records* of fixed or variable length
- Block size – records are combined into *blocks* of a certain length
- Extent – allocation of space; when the primary extent is filled, z/OS will automatically allocate more extents, called secondaries

Example: Data Set Allocation from TSO

```
READY
 alloc dataset(USERID.test.text) volume(MARF0D) unit(3390)
 recfm(f) lrecl(80) dsorg(ps)
READY
 listds
 ENTER DATA SET NAME -
USERID.test.text
 USERID.TEST.TEXT
  --RECFM-LRECL-BLKSIZE-DSORG
    F      80     80       PS
  --VOLUMES
    MARF0D
```

Example: Sorting Data with TSO

```
 ALLOCATE DATASET(AREA.CODES) FILE(SORTIN) SHR
 ALLOCATE DATASET(*) FILE(SORTOUT) SHR
 ALLOCATE DATASET(*) FILE(SYSOUT) SHR
 ALLOCATE DATASET(*) FILE(SYSPRINT) SHR
 ALLOCATE DATASET(SORT.CNTL) FILE(SYSIN) SHR
 CALL 'SYS1.SLICELINK(SORT)'
ICE143I 0 BLOCKSET SORT TECHNIQUE SELECTED
ICE000I 1 - CONTROL STATEMENTS FOR Z/OS DFSORT V1R5
 SORT FIELDS=(1,3,CH,A)
201 NJ
202 DC
***
```

COBOL

The structure of the Common Business Oriented Language explains some of the TSO call:

- A COBOL program consists of four *Divisions*, the identification division, environment division, data division and procedure division
- The *environment division* describes all of the external dependencies of a COBOL program, in particular files (or data sets) accessed

COBOL Input-Output Section Example

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
  SELECT MyFileName ASSIGN TO JCLDDLAB
     ORGANIZATION IS LINE SEQUENTIAL.
FILE SECTION.
  FD MyFileName.
  01 FileRecord.
     02 FirstName PIC X(10).
     02 MiddleInitial PIC X.
     02 LastName PIC X(15).
```

Programming TSO

TSO supports two scripting languages:

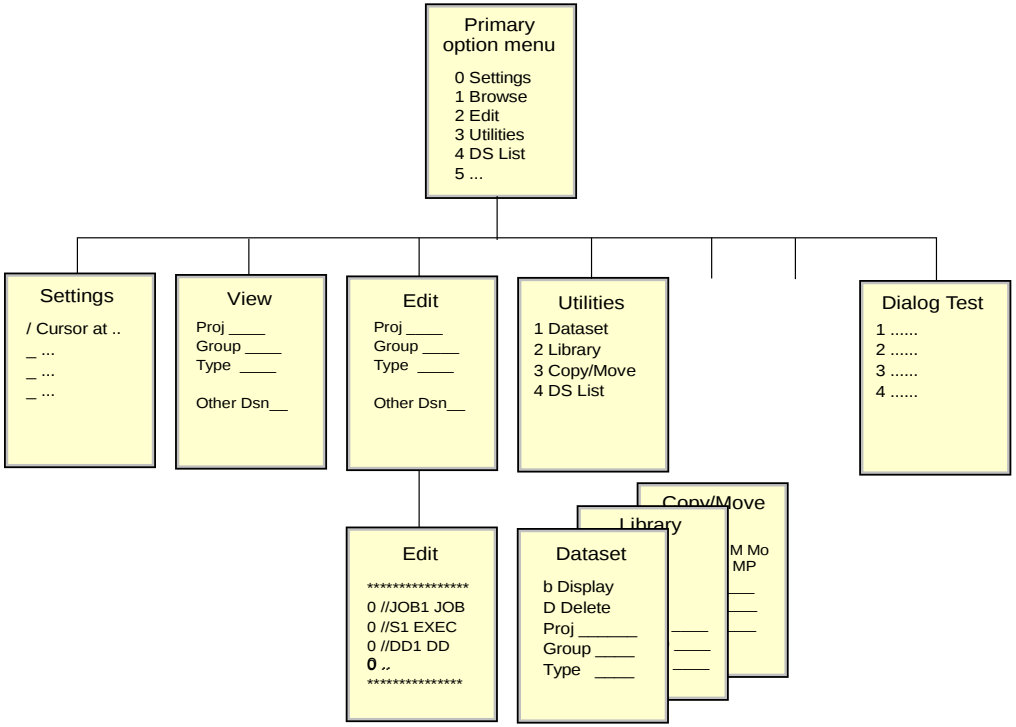
- CLIST (Command List) – like a UNIX shell script, interpreted; includes arithmetics and logical operations, string handling and of course routine tasks like dataset allocation
- REXX – more general, more portable, can be compiled

ISPF Features

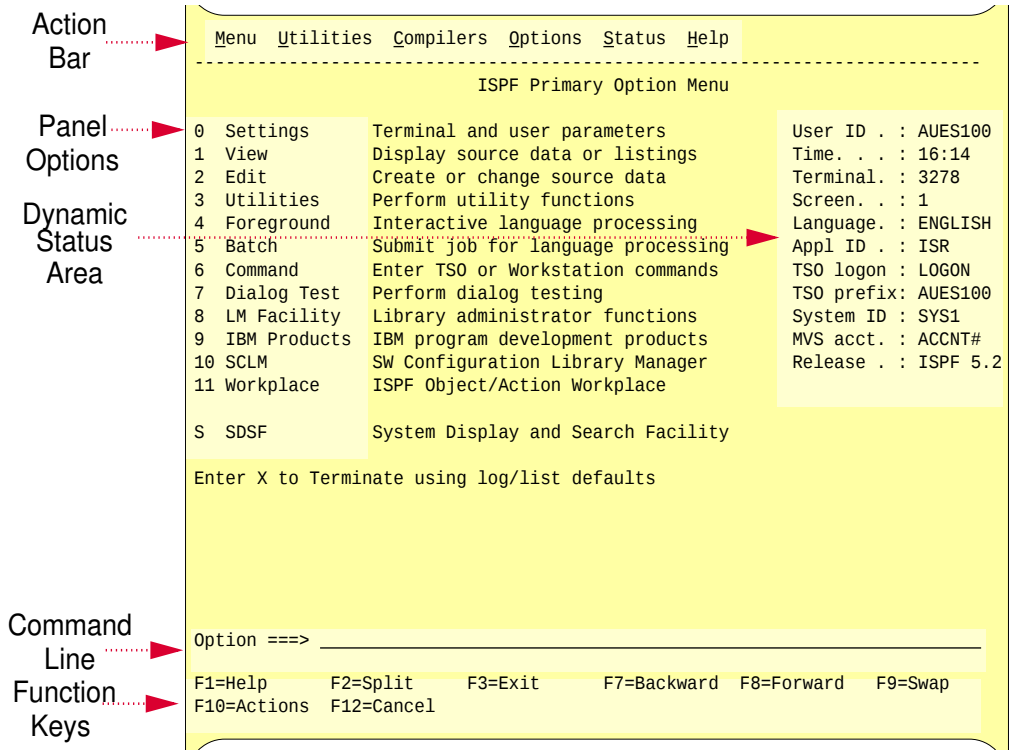
ISPF includes:

- An online help system (press the PF1 key)
- A text editor / text viewer
- Functions for allocating and locating files
- Functions for calling various utilities

ISPF Default Menu Structure



ISPF Panel Structure



Navigating ISPF

- Action Bar
- Point-and-Shoot
- Option Number
- Function Keys

Keyboard mapping

Function	Key
Enter	CTRL (right side)
Exit, end, return	PF3
Help	PF1
PA1/Attention	Alt-Ins or ESCape
PA2	Alt-Home
Cursor movement	Tab or Enter
Clear	Pause

Keyboard mapping

Function	Key
Page up	PF7
Page down	PF8
Scroll left	PF10
Scroll right	PF11
Reset locked keyboard	CTRL (left side)

Example: Data Set Allocation from ISPF

```
                          Allocate New Data Set
Command ===>
Data Set Name  . . . : USERID.TEST2.TEXT
Management class . . .                (Blank for default management class)
Storage class  . . . .                (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 . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 2              (In above units)
 Secondary quantity    1              (In above units)
 Directory blocks  . . 0              (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . .
 Data set name type  :                (LIBRARY, HFS, PDS, or blank)  *
```

ISPF Editor

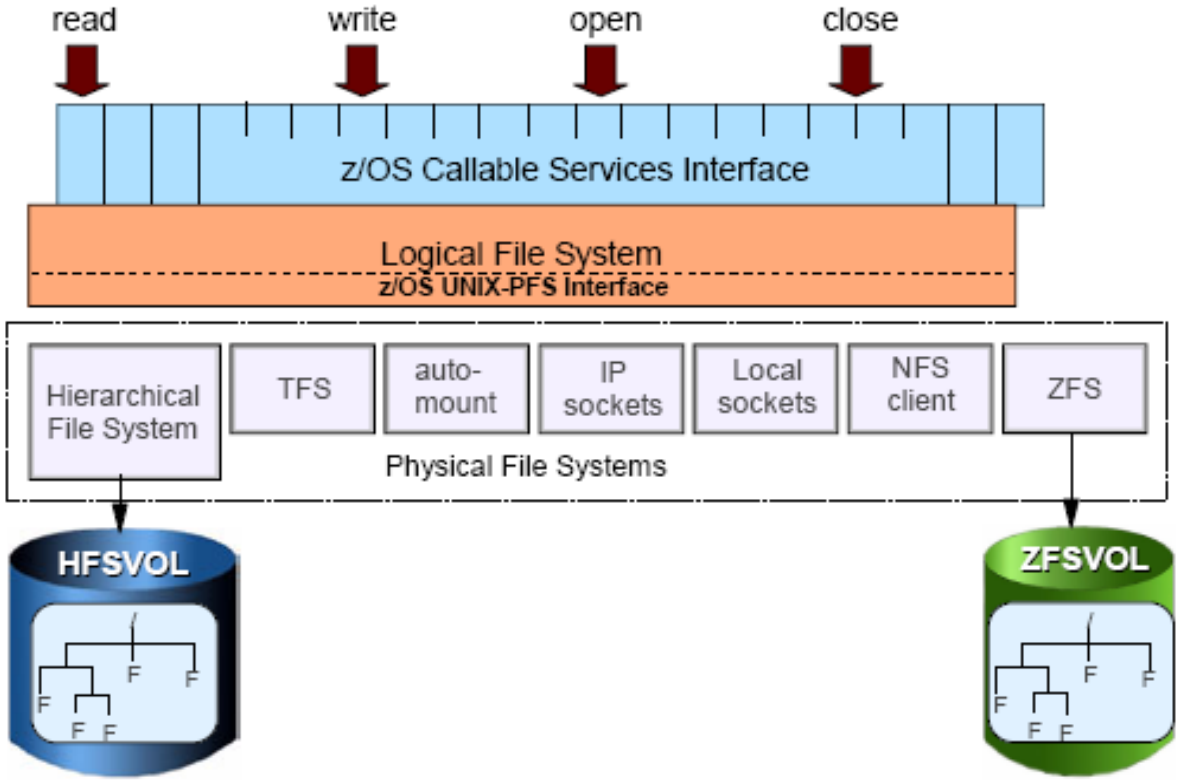
- Select "Utilities–Dslist–TEST2.TEXT" and use the prefix command "e"; confirm the next dialog by pressing "ENTER"
- Essentially like `xedit` (minor differences)
- Can be invoked from `omvs` using `oedit`

ISHELL

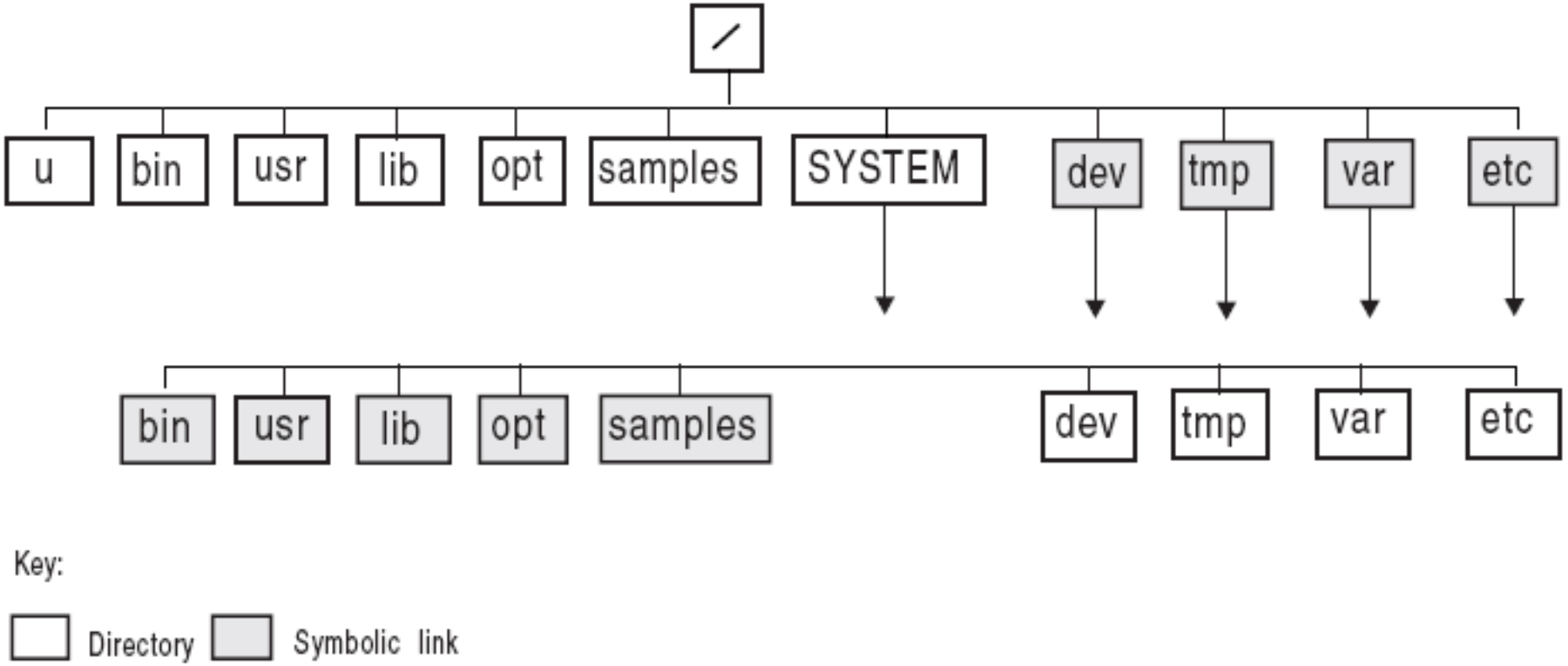
- ISHELL is the ISPF SHELL
- Provides panels for working with UNIX files, file systems and UNIX administration
- Essentially, ISPF-like menu-driven replacement for `omvs`

$\Rightarrow$ For users more familiar with TSO/ISPF than UNIX, not for you!

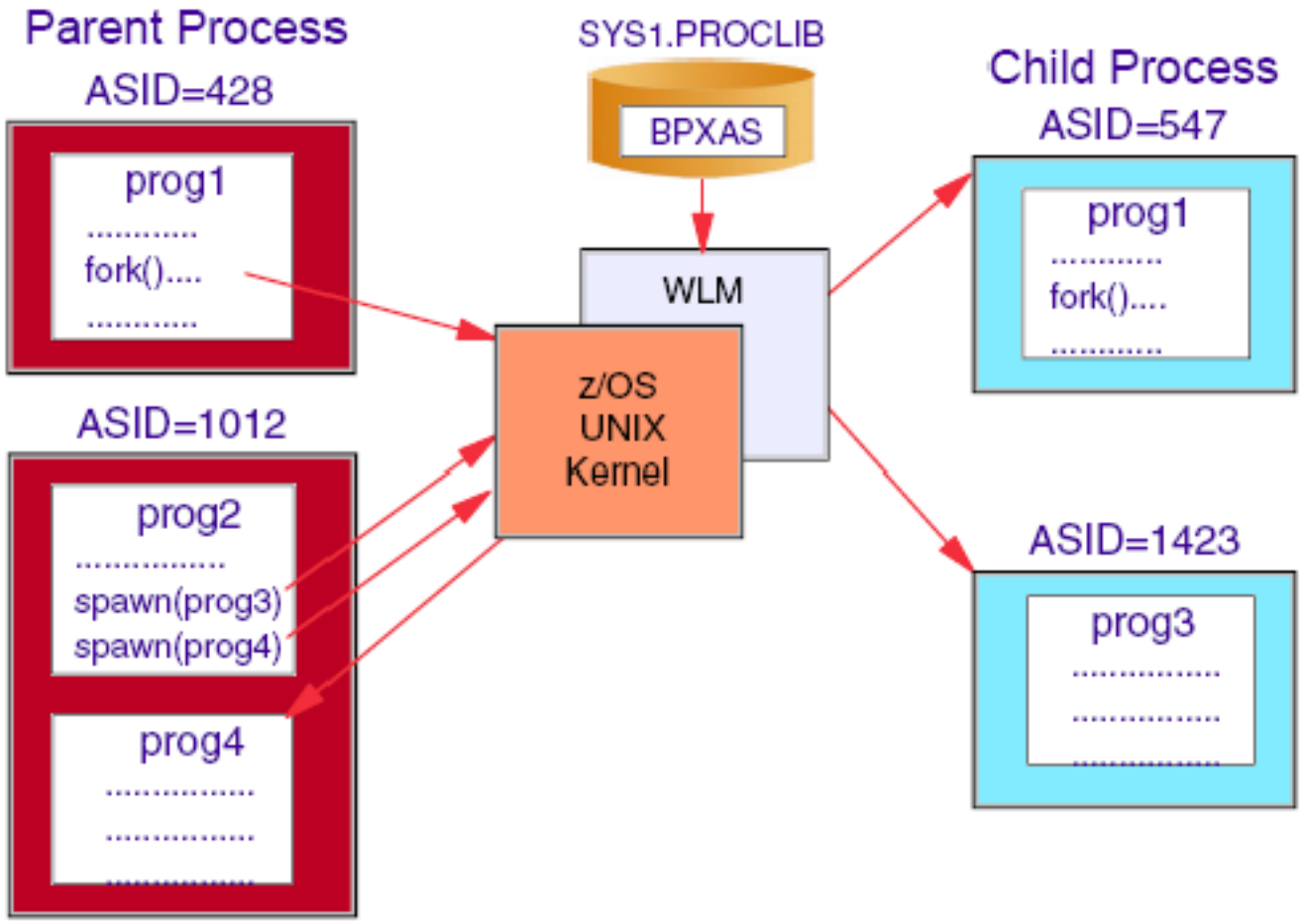
UNIX File Systems on z/OS



UNIX File System Organization



UNIX Processes on z/OS



"Priced" features

A z/OS system usually requires additional licensed programs:

- Security manager
- Database manager
- Compilers
- Utility programs
- ...

We will discuss some of these in the 2nd half of the course!

Questions

?