/dsfs

z/OS Data Set File System

Setup und wie kann man es einsetzen

Daniel Cattin
Osys Software AG

Inhalt

- Installation
- Konfiguration
- Einschränkungen
- Anwendungsbeispiele



Installation

- z/OS Data Set File System ist Teil der z/OS Basis
- Es muss aber Konfiguriert werden, um es zu benutzen



Konfiguration

- Die Konfiguration ist in folgende Schritte aufgeteilt
  - DSFS STC einrichten
  - DSFS File System für USS aktivieren
  - Utility ZFS erstellen
  - Mount Point erstellen
  - BPXPRMxx Updaten
  - DSFS IDFPRMxx Member erstellen



DSFS STC

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       SYS1.OSYS.PROCLIB(DSFS) - 01.00                    Columns 00001 00080
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ********************************
000001 //*
000002 //* Licensed Materials - Property of IBM
000003 //* 5650-ZOS
000004 //* Copyright IBM Corp.  2022
000005 //*
000006 //DSFS     PROC REGSIZE=0M
000007 //*
000008 //DSFSGO   EXEC PGM=BPXVCLNY,REGION=&REGSIZE,TIME=1440
000009 //*
000010 //*STEPLIB DD DISP=SHR,DSN=hlq.SIEALNKE              <--DSFS LOADLIB
000011 //*IDFZPRM DD DISP=SHR,DSN=SYS4.PVT.PARMLIB(IDFFSPRM) <--DSFS PARM FILE
000012 //*
****** ***************************** Bottom of Data *****************************
```

- Die Loadmodule liegen in der SYS1.SIEALNKE, welche APF authorisiert sein muss.
- DSFS Beispiel Procedure findet man in der SIOEPROC.
- Die Unix Programme sind unter /usr/lpp/dfs/global/bin/ abgelegt
- Im RACF muss
  - ein User für DSFS angelegt werden
  - ein STARTED Profile für DSFS erstellt werden

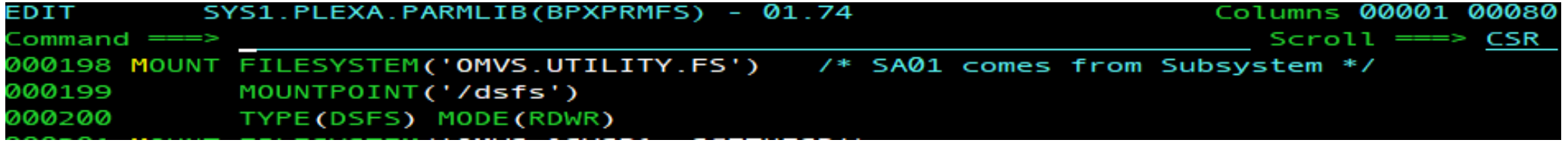
DSFS File System im USS aktivieren

- DSFS wird als ein weiterer Filesystem Type im BPXPRMxx eingetragen

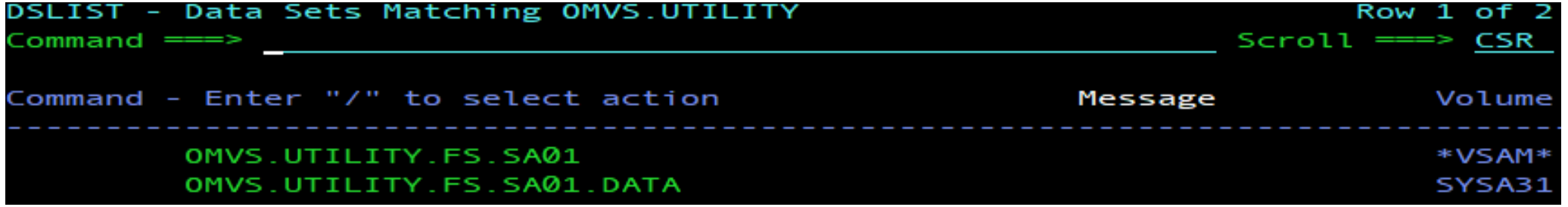
```
EDIT       SYS1.PLEXA.PARMLIB(BPXPRM00) - 01.09                 Columns 00001 00080
Command ===> _____________________________________________________ Scroll ===> CSR
000366
000367     FILESYSTYPE TYPE(DSFS)            /* Type of file system to start  */
000368                 ENTRYPOINT(IDFFSCM)
000369                 ASNAME(DSFS,'SUB=MSTR')
000370                 PARM('PRM=(00)')       /* Use IDFPRMxx parmlib member   */
000371
```

ZFS erstellen und mounten

- DSFS braucht ein Utility ZFS
- Dieses Utility ZFS wird zwar mounted, kann aber nicht als normales ZFS benutzt werden.

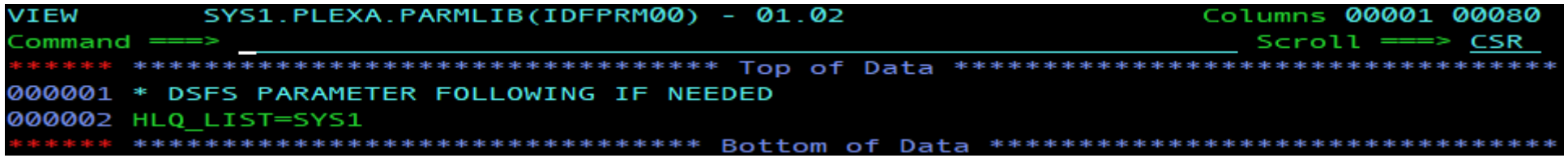


- Der Name des ZFS muss das Subsystem im letzten Qualifier haben, ausser UTFS_NAME wird im IDFPRMxx Member definiert.



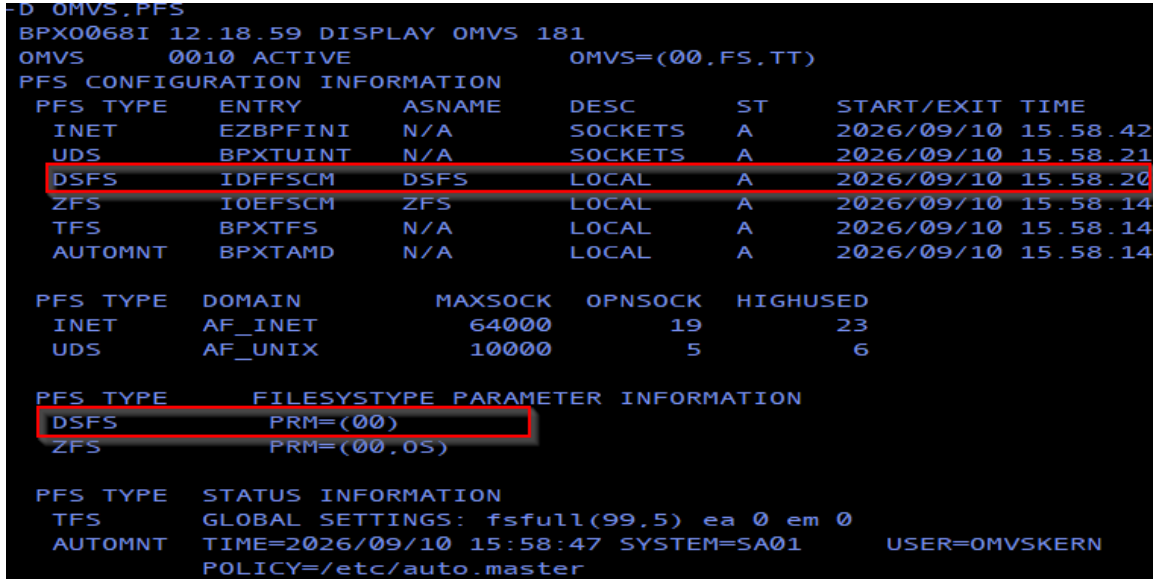
IDFPRMxx Member

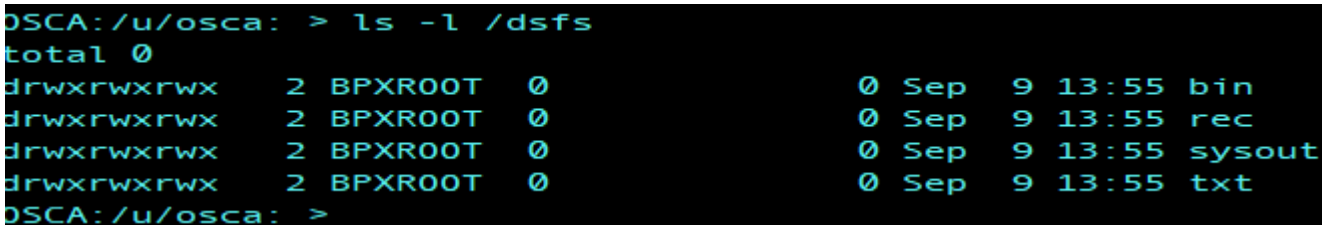
- Hier kann man das verhalten von DSFS beeinflussen. Nachfolgend ein paar Beispiele
  - Komprimierung (compress=on) wird nur beim neu formatieren des Utility ZFS verwendet. Anschliessend werden die Inhalte im Utiltiy ZFS zEDC komprimiert.
  - HLQ_LIST ist eine Liste von HLQs die der DSFS User nicht verwenden darf. Der Parameter kann mehrfach vorkommen.
  - UTFS_NAME damit kann der Defaultname des Utility ZFS beeinflusst werden.



Kontrolle

- Wenn als korrekt gemacht ist, dann
  - zeigt D OMVS,PFS das DSFS als aktive an
  - ls –l /dsfs seine Unterverzeichnisse bin, rec, sysout und txt





Kontrolle

- Mit dem Unix Kommando dsadm können verschieden Informationen abgerufen werden
  - zB dsadm fsinfo können Informationen über das ZFS abgefragt werden

```
BPXROOT:/: > dsadm fsinfo
   File System Name:          OMVS.UTILITY.FS.SA01

   System:                 SA01            Devno:                  42
   Size:                   226800K         Free 8K Blocks:         28053
   Free 1K Fragments:      7               Log File Size:          2272K
   Bitmap Size:            32K             Anode Table Size:       8K
   File System Objects:    7               Version:                1
   Overflow Pages:         0               Overflow HighWater:     0
   Space Monitoring:       0,0
   ENOSPC Errors:          0               Disk IO Errors:         0
   Status:                 NE,NC

   File System Creation Time: Sep  9 13:55:58 2026
   Mount Time:                Sep 10 15:58:28 2026

   Last Grow Time:            n/a


Legend: NE=Not encrypted, NC=Not compressed
```

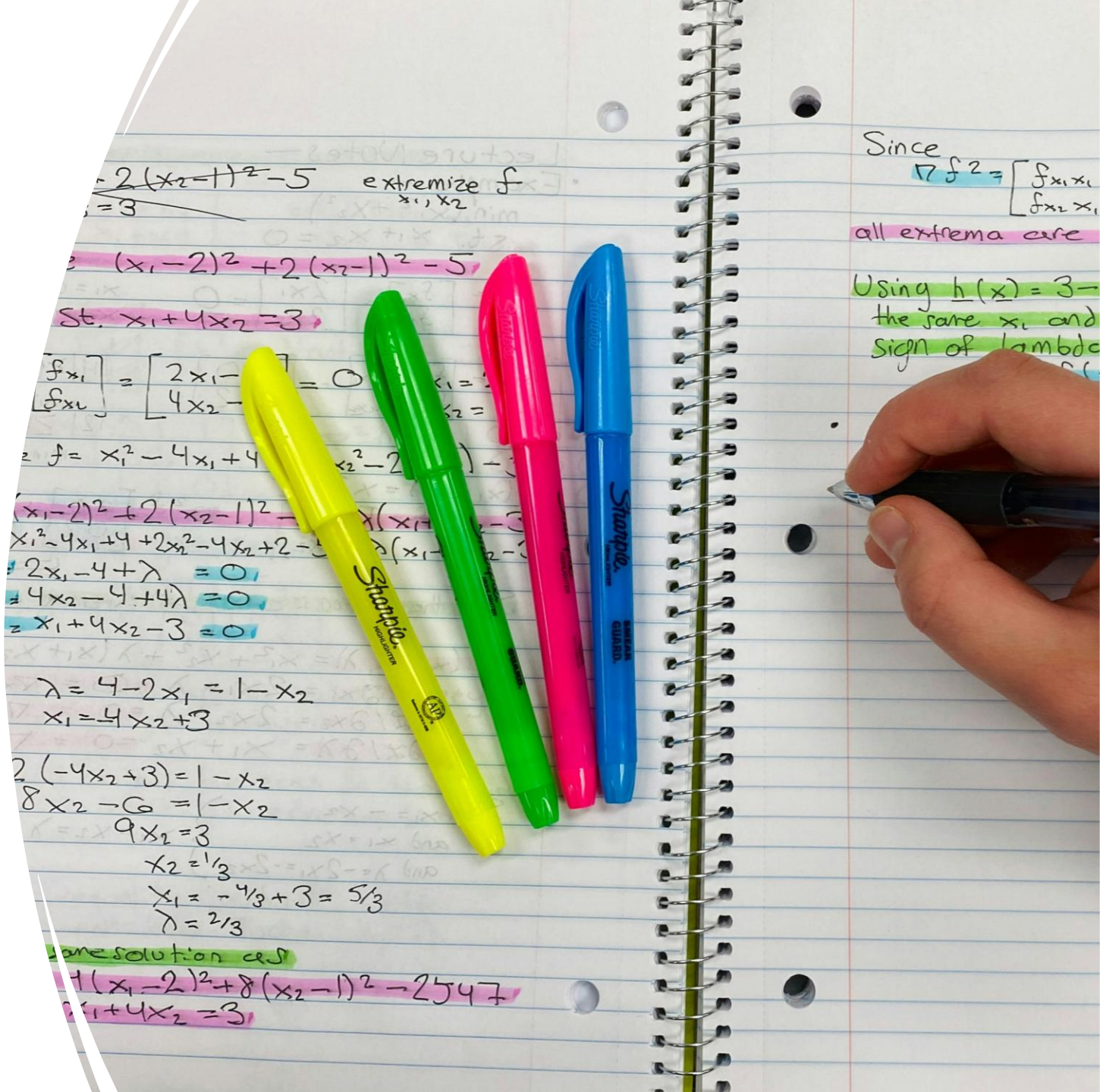
Einschränkungen



Welche File Typen sind unterstützt

- Es können nur PS, PDS und PDSE Datasets verarbeitet werden
- Es können nur folgende RECFM verarbeitet werden. F, FB, FBA, FS, FBS, V, VB, VBA und U. Wobei U nur bei PDS oder PDSE und FS und FBS nur bei PS Dataset unterstützt ist

Anwendungsbeispiele



Verzeichnis Struktur

/dsfs

- /bin — Dateien werden binary verarbeitet
- /rec — Dateien werden im Rekordformat verarbeitet.
  Jeder Rekord hat 4-Byte Länge der Daten am Anfang.
- /txt — Dateien werden Rekordweise verarbeitet
  Das Ende des Rekords wird mit x'15' (LF) angezeigt
- /sysout — Zugriff auf Inhalte im JES Spool

  - /hlq — HLQ des Datasets, welches verarbeitet werden soll
    - /nql — Nächste Qualifier des Datasets mit Punkten getrennt
      - /mem — Member Name, wenn PDS

Beispiel:

osca.ispf.jcl(iefbr14) ➔ /dsfs/txt/osca/ispf.jcl/iefbr14

Live Demo

- dsadm configquery -all
- ls /dsfs/txt/osca, ls /dsfs/rec/osca, ls /dsfs/bin/osca
- ls /dsfs/txt/osca/ispf.jcl
- cat /dsfs/[txt|rec|bin]/osca/cert.letsenc.intye1.crt
- vim /dsfs/txt/osca/ispf.jcl/iefbr14
  - Autofill mit TAB im zsh zeigen
- USS SRCHFOR: grep -i iefbr14 /dsfs/txt/osca/ispf.jcl/*
- ls /dsfs/sysout/osca/

Beispiel: Let’s Encrypt CA Zertifikat

- Im USS
  - dsadm createparm -path /dsfs/txt/osca -psmodel "cyl dsorg(ps) lrecl(255) recfm(vb) space(20,10) blksize(0)«
  - curl --output /dsfs/txt/osca/cert.letsenc.intye1.crt https://letsencrypt.org/certs/gen-y/int-ye1.pem

- Im TSO/ISPF
  - RACDCERT CERTAUTH +
    ADD('OSCA.CERT.LETSENC.INTYE1.CRT') +
    WITHLABEL('Lets Encrypt INT YE1') +
    TRUST

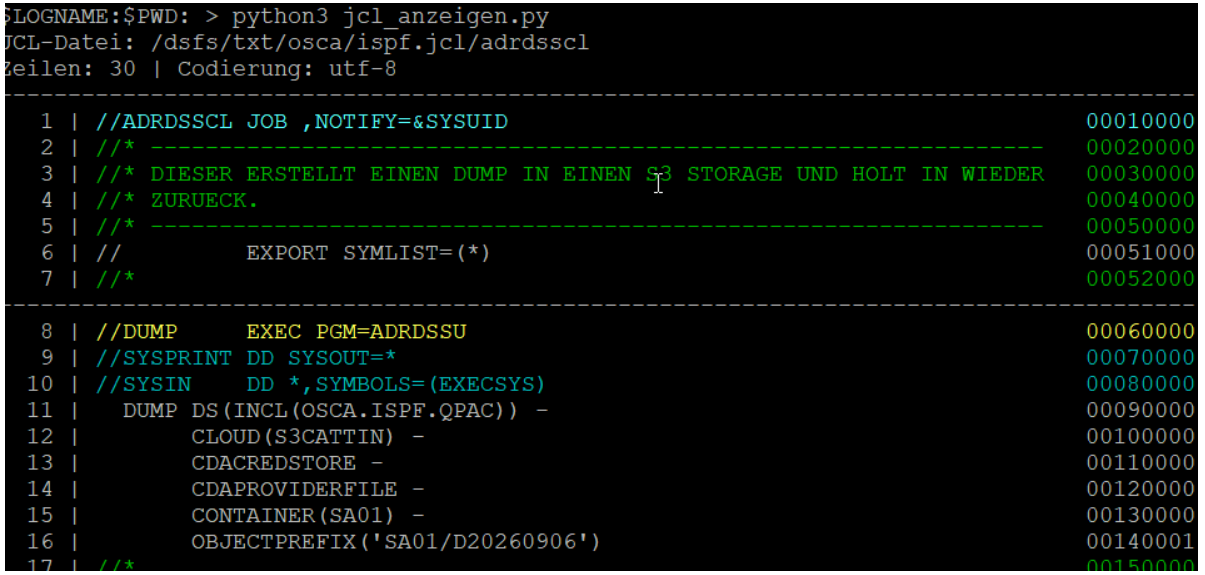
Weitere Anwendungsbeispiele

- Überall dort wo heute in einem zwischen Step via OGET oder OPUT Files von oder ins USS kopiert werden.

- Diese Steps können ersetzt werden und die Dateien direkt im USS angesprochen werden.

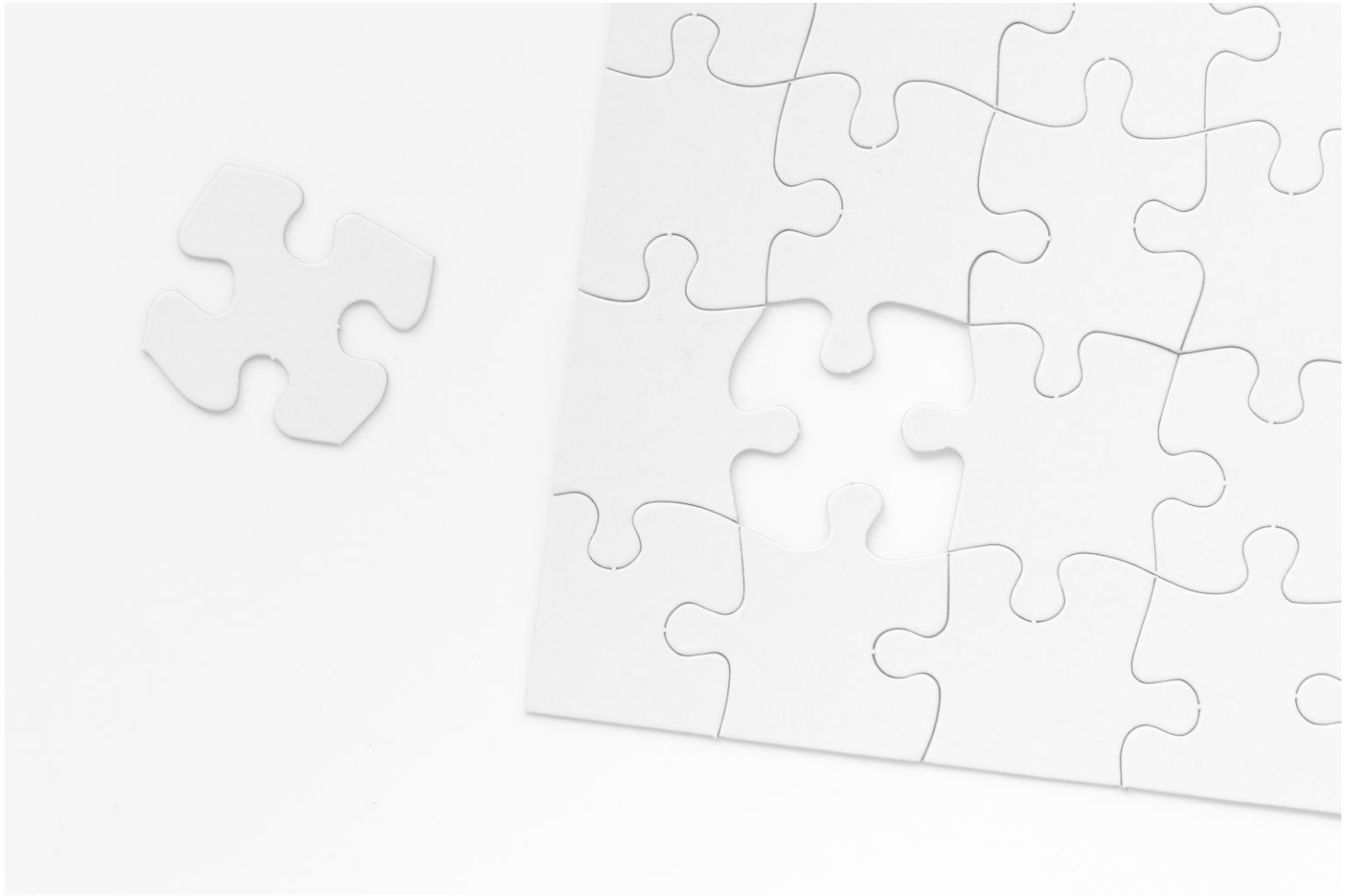
Beispiel: Zugriff auf eine Datei aus einem python Programm

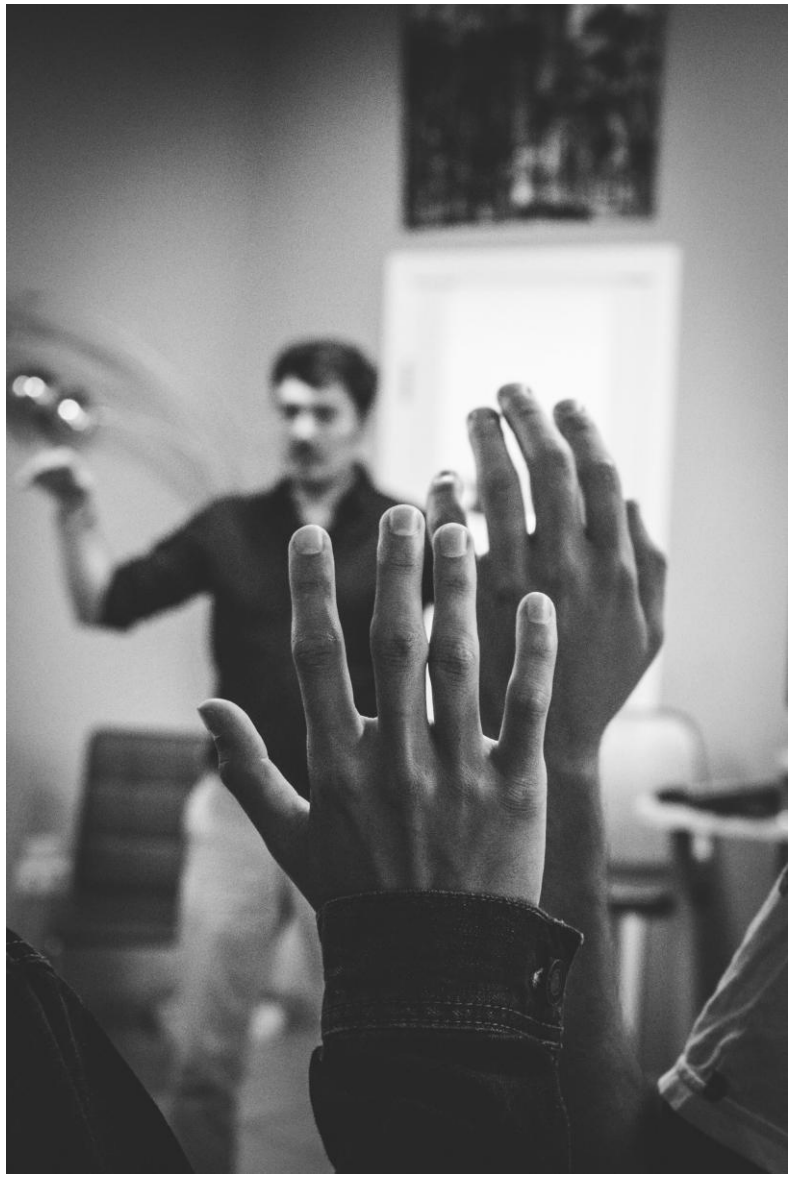
- Ich habe einer KI den Auftrag gegeben, ein Python Script zu schreiben, welches eine JCL schön formatiert auf der Konsole ausgibt.
- Demo: python3 jcl_anzeigen.py



Fazit

- DSFS erlaubt es z/OS Dateien direkt aus dem USS zu lesen und/oder zu schreiben.
- Dies spart Steps, bei welchen Dateien von oder ins USS kopiert werden müssen. Stichwort oget/oput





DANKE!
THANK YOU!
MERCI!
GRAZIE!
GRACIAS!
DANK JE WEL!