FILE SYSTEM CHANGES IN OpenVMS V8.4

Shyam Sankar G
OpenVMS Engineering

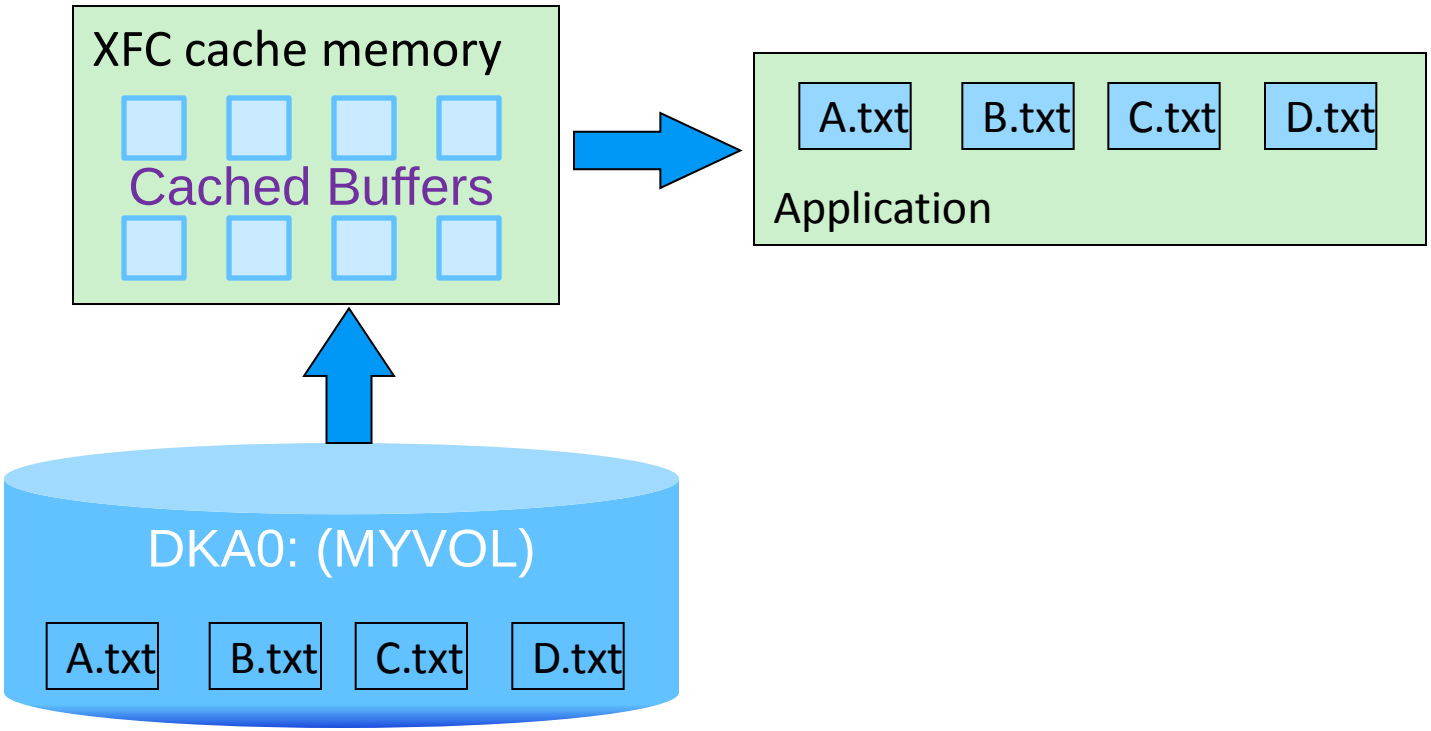
Agenda

- File System updates with OpenVMS V8.4
  - XFC caching enhancements
  - 2 TB support
  - Enhanced Symlinks

XFC CACHING ENHANCEMENTS

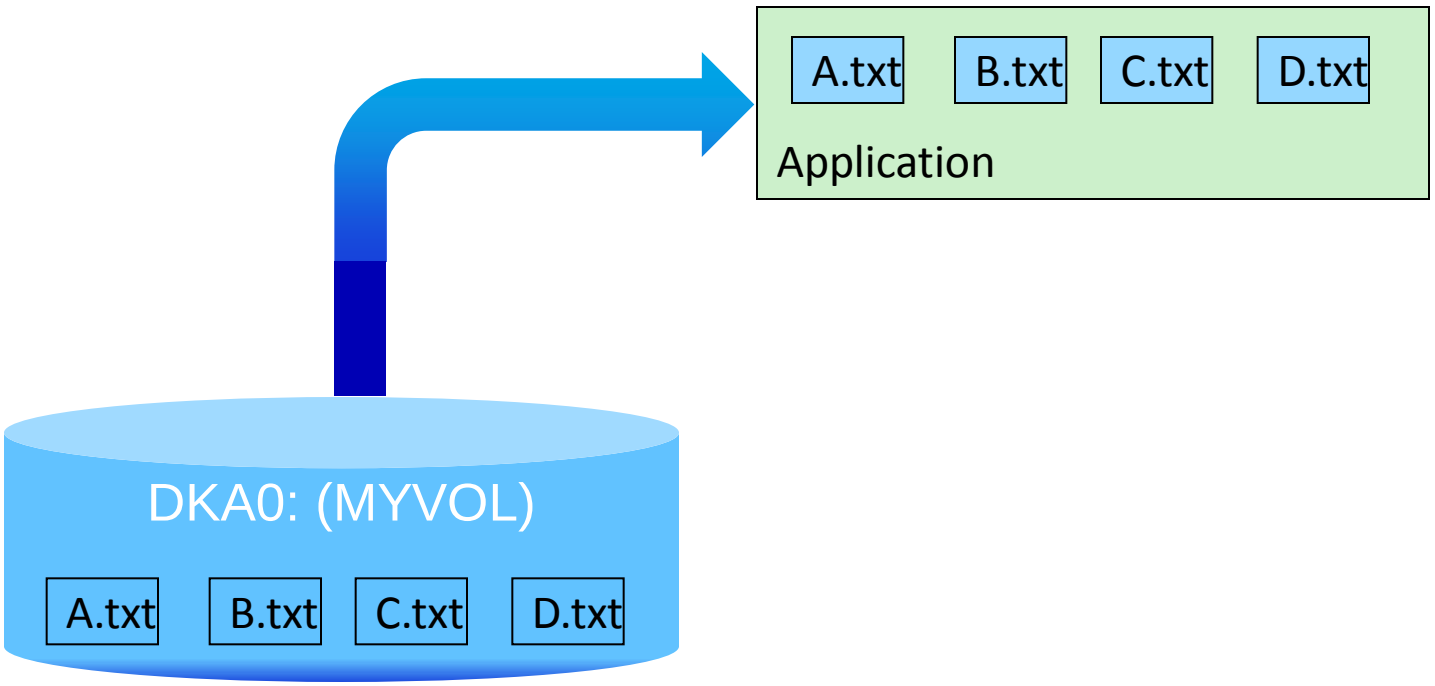
Enabling XFC caching

- $ MOUNT /CACHE DKA0: MYVOL
- Read I/O satisfied from cache, when caching enabled



Disabling XFC caching

- $ MOUNT /NOCACHE DKA0: MYVOL
- Read I/O performed from disk, when caching is disabled



V8.3 Limitations and V8.4 Solutions

- Limitation:
- $ MOUNT /[NO]CACHE affects both data and metadata cache

- Solution:
- Use new [NO]DATA keyword on /CACHE qualifier

V8.4 Solution: Use new [NO]DATA keyword

- Enhanced $ MOUNT qualifier /CACHE
  - Differentiates XFC and XQP cache
- New with V8.4:
  - /CACHE=NODATA: MOUNT volume, but no data cache
  - /CACHE=DATA: MOUNT volume, enable data cache (same as /CACHE)
- Does not change the settings of the metadata caches
  - EXTENT, FILE_ID, QUOTA
  - These are also controlled with the /CACHE qualifier

Special Consideration for Mixed Version Cluster

- $ MOUNT /CACHE=NODATA /CLUSTER
- Pre-V8.4 nodes cannot perform such a mount
- Will return %MOUNT-W-RMTMNTFAIL
  - V8.4 nodes will mount successfully
- $ MOUNT /CACHE /CLUSTER works as always

V8.3 Limitations and V8.4 Solutions

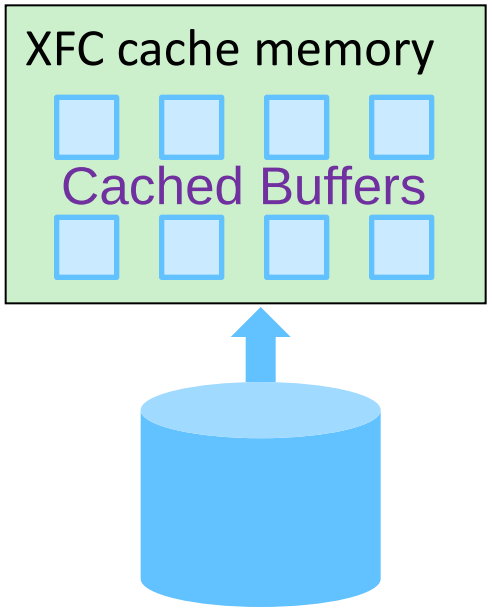
- Limitation:
- Caching enabled/disabled only with $ MOUNT
  - To change caching attribute, must dismount /cluster and mount again
- Solution:
- Use new $ SET VOLUME /CACHE qualifier to change caching on the fly

V8.4 Solution: Modify caching status dynamically

- New qualifier "/CACHE" with "$ SET VOLUME"
  - /CACHE=NODATA: Disable further caching of volume
  - /CACHE=DATA: Enable caching of volume
- No need to dismount the volume

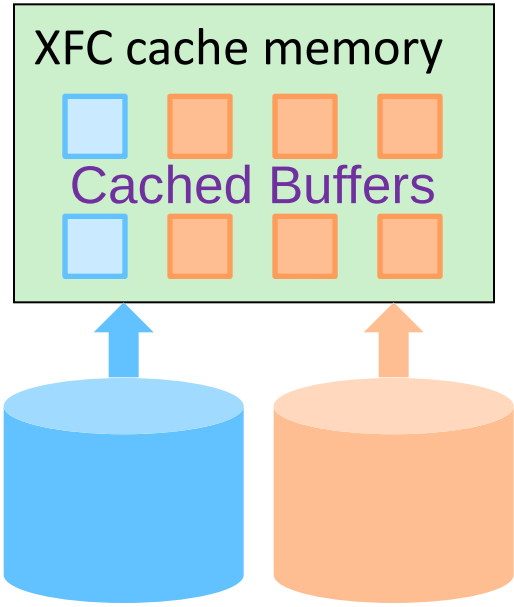
V8.3 limitations and V8.4 Solutions

- All volumes compete for entire cache memory
  - Good and bad
  - Buffers provided on demand, but lower-priority volume can dominate



Left: Blue volume occupies all cache

Right: Orange volume displaces blue buffers



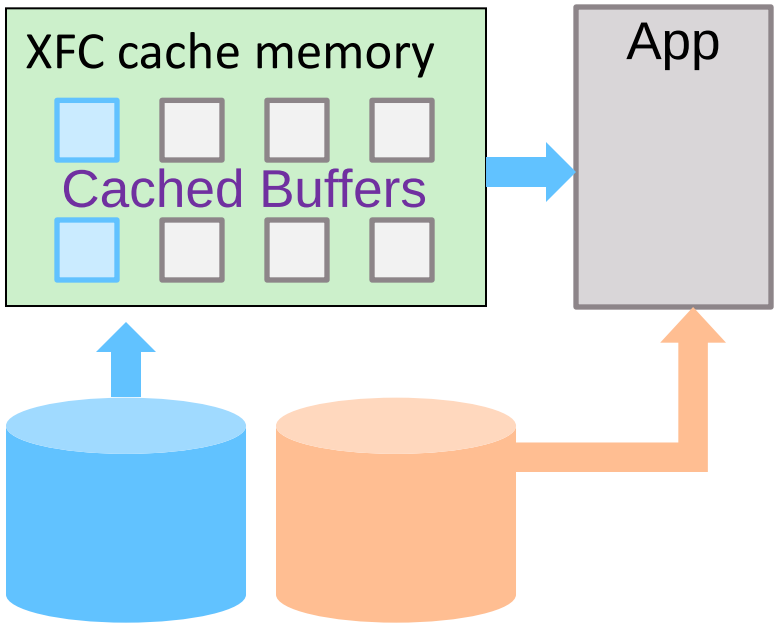
V8.4 Solution: Modify caching status dynamically

- New qualifier "/CACHE" with "$ SET VOLUME"
- … and the CLEAR_DATA keyword
  - /CACHE=CLEAR_DATA: Free up cached buffers of volume
- Example: Disable XFC cache and free up buffers
  - $ SET VOLUME ORANGE_VOL /CACHE=(NODATA,CLEAR_DATA)
- Example: Re-enable XFC cache
  - $ SET VOLUME ORANGE_VOL /CACHE=DATA

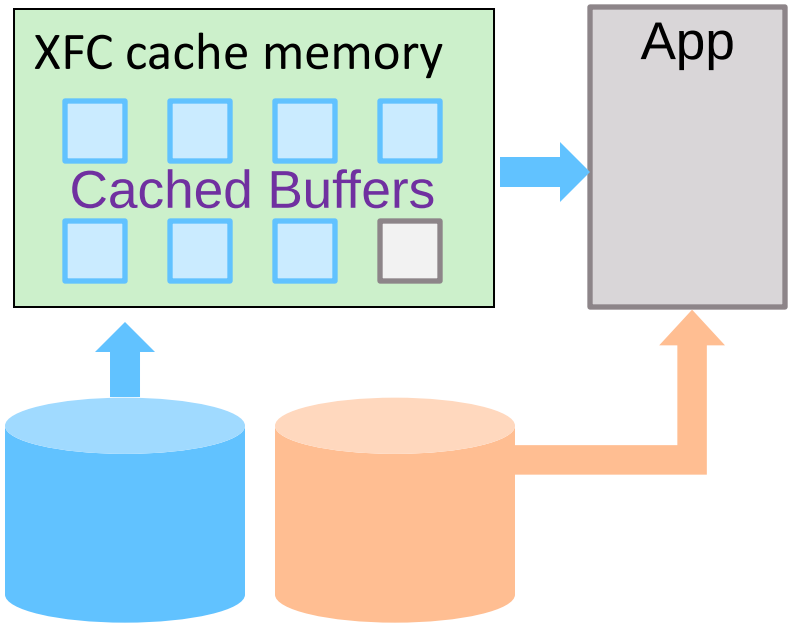
V8.4: Now we can…

–Free-up ‘orange’ buffers from cache

–Stop ‘orange’ buffers from being cached



–Allow more ‘blue’ buffers to be cached



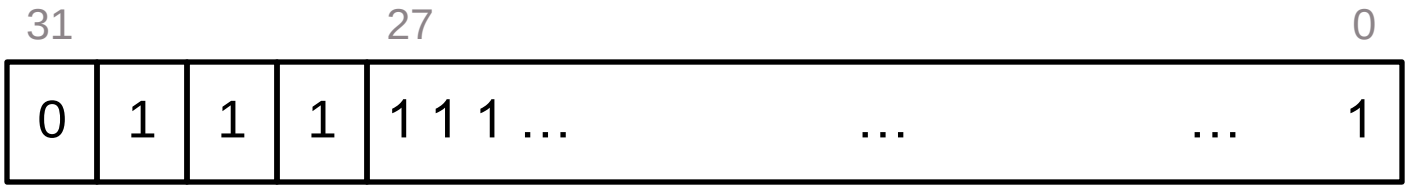
Summary: Advantages to V8.4 Changes

- No downtime to change volume caching
  - Can change volume caching on demand
- Can prevent cache from filling up with useless data
  - Good for performance
- Can enable/disable caching on system disk
- Can disable data caching without affecting metadata performance

2 TB support

Existing Limitation

- V8.3 file system supports volumes up to 1TB
  - Single disk volume is 1TB or less
  - Bound volume set can have 255 such volumes
- Logical Block Number (LBN) is signed 32-bit integer



- 1TB limitation comes from the following calculation:
  - LBN 0 through 0x7FFFFFFF -> 0x80000000 blocks * 0d512 bytes = 1 TB

V8.4: 2TB volumes

- Device drivers and I/O subsystem: 64-bit LBN
  - SCSI disks only
- These support 2 TB
  - RMS, XQP, INIT, MOUNT, BACKUP, SHADOWING, MSCP
  - Use sign bit to double capacity
- These continue with current limits
  - DFO: No change, still 1 TB
  - C RTL: No change
  - NFS: No change
- Retrieval Pointer format: NO CHANGE
  - 30-bit Count stays, each ‘extent’ ≤ 512 GB

V8.4: 2TB volumes

```
$ show dev/full $1$DGA100:

Disk $1$DGA100: (COEREF), device type COMPAQ MSA1000 VOLUME, is online,
    allocated, deallocate on dismount, mounted, file-oriented device, shareable,
    served to cluster via MSCP Server, error logging is enabled.

    Error count                    0    Operations completed            1657976
    Owner process           "_TNA2:"    Owner UIC                       [SYSTEM]
    Owner process ID        21400434    Dev Prot            S:RWPL,O:RWPL,G:R,W
    Reference count                4    Default buffer size                 512
    Current preferred CPU Id       0    Fastpath                              1
    WWID   01000010:6001-4380-0008-E010-3250-94C7-EF07-000C
    Total blocks          3145728000    Sectors per track                   255
    Total cylinders            48378    Tracks per cylinder                 255
    Logical Volume Size   3145728000    Expansion Size Limit         4261348350
    Allocation class               1
    :
    :
```

2 TB on older VMS version

- Older VMS versions (pre V8.4)
- $ MOUNT will incorrectly mount 2 TB volume
  - ***Unpredictable results!***
- $ MOUNT made to fail in patch, as a safety measure
  - %MOUNT-F-UNSUPPORTED, unsupported operation or function
- Unpatched MOUNT does not have safety measure!
- Install latest patches
  - V73-2R: VMS732_MOUNT96-V0200
  - V82R: VMS82A_MOUNT96-V0100
  - VMS821I_MOUNT96-V0100
  - V83R: VMS83A_MOUNT96-V0100
  - VMS83I_MOUNT96-V0100
  - V83-1H1R: Not Released Yet

Impact, Challenges

- DCL symbols are signed 32-bit
  - Lexicals can return ***un***signed 32-bit
  - F$GETDVI VOLSIZE, FREEBLOCKS, etc
  - Techniques/workarounds to handle unsigned symbols
  - See V8.4 DCL Dictionary, Appendix
- Applications handling LBNs: Change signed to unsigned
  - Source code changes
  - Arithmetic operations on LBN must be converted to **64-bit** operations
  - Comparisons must be unsigned

SYMLINK ENHANCEMENTS

Outline

- What is a symlink
- Symlinks on OpenVMS
- V8.3 Limitations and V8.4 Solutions
  - Programming Interface
  - Metadata
- Compatibility Between V8.3 and V8.4

About symlinks

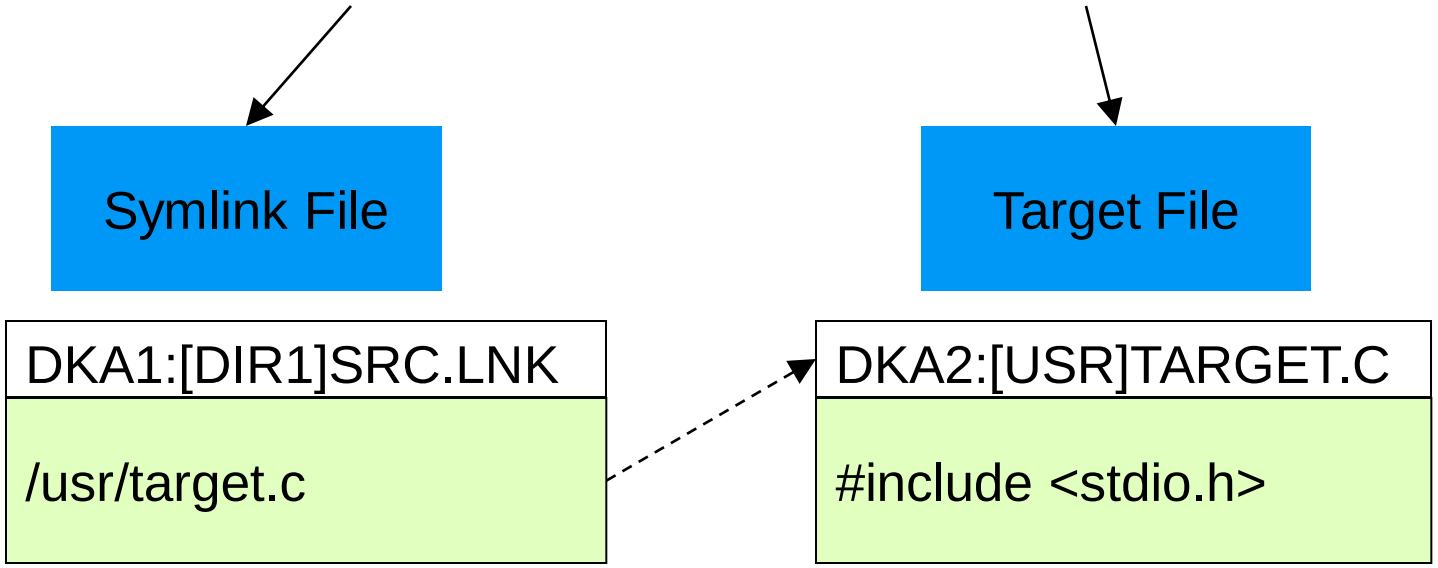
- Symbolic link, symlink, soft link
- Pointer to a target object
- Target object: file name, directory name, arbitrary string
- On OpenVMS, a symlink is implemented as a separate file
  - The file contains the target string
- On UNIX, a symlink is ***not a file*** at all, but only a ***directory entry***
  - The entry contains the symlink name and target string
- Supported by most Unix-like operating systems, Windows, … and of course, OpenVMS

Advantages of Symlinks

- Unix-like development / runtime environment
- Eases porting of Unix applications to VMS

Symlinks on OpenVMS

- Available starting with OpenVMS V8.3
- Symlink is a ***special file***
- Target string stored as data in this file
- File System interprets target string as POSIX path name
- `$ CREATE DKA1:[DIR1]SRC.LNK /SYMLINK="/usr/target.c"`



The /[no]symlink qualifier

- Use it to distinguish between the symlink file and the target file.
  - Defaults to /symlink on DIRECTORY command
  - Defaults to /nosymlink on other commands

Symlinks on OpenVMS

```
$ dump /symlink src.lnk



Dump of file MDA1:[000000.TEST]SRC.LNK;1 on  9-SEP-2009 10:53:45.24
File ID (12,2,0)   End of file block 1 / Allocated 1

Virtual block number 1 (00000001), 512 (0200) bytes

 00000000 00000000 00000000 00000000 00000063 2E746567 7261742F 7273752F /usr/target.c...
 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 ................
```

POSIX pathnames on OpenVMS

- Simulate Unix-like directory tree on VMS
- $ SET ROOT
  - An OpenVMS directory acts as the UNIX “/”
  - Pathnames parsed relative to this root directory

```
$ SET ROOT DKA100:[000000]

$ SHOW ROOT
DKA100:[000000]

$ DIR “^UP^/dir1/dir2/a.txt”

$! The above becomes $ DKA100:[DIR1.DIR2]A.TXT
```

Symlink file attributes

```
$ dir /full src.lnk    !  /symlink qualifier is assumed by default

Directory MDB0:[000000.DIR1]

SRC.LNK;1                       File ID:  (14,1,0)
Size:            1/1            Owner:    [SYSTEM]
Created:    22-AUG-2010 07:51:29.56
Revised:    22-AUG-2010 07:51:29.56 (1)
Expires:    <None specified>
Backup:     <No backup recorded>
Effective:  <None specified>
Recording:  <None specified>
Accessed:   <None specified>
Attributes: <None specified>
Modified:   <None specified>
Linkcount:  1
File organization:  Special: symbolic link
  Link Contents: /dir1/dir2/a.txt
Shelved state:      Online
Caching attribute:  Writethrough

    :
    :
```

Symlink file header

```
$ dump /symlink /header /block=c=0 src.lnk   ! /symlink to be stated explicitly

File ID (14,1,0)   End of file block 1 / Allocated 1

                              File Header

Header area
    Identification area offset:           40
    Map area offset:                      100
    Access control area offset:           255
    Reserved area offset:                 255
    Extension segment number:             0
    Structure level and version:          5, 1
    File identification:                  (14,1,0)
    Extension file identification:        (0,0,0)
    VAX-11 RMS attributes
        Record type:                      none
        File organization:                Special
        Record attributes:                Symbolic link
        Record size:                      32767
        Highest block:                    1
        End of file block:                1
        End of file byte:                 21
```

V8.3 symlink limitations

- Implementation not consistent
  - RMS/QIO programmer: Confusing attribute/field names
    - RMS uses ‘special’, XQP uses ‘symlink’
  - CRTL supports logical names but RMS does not
  - XQP provides some flags, RMS doesn’t use them
- To know if dir entry = symlink
  - XQP provides DIR$V_TYPE, but not used
  - Need to read File Header (need read access to file)
- RMS performance problems, minor functional deficiencies
- All of these limitations addressed in V8.4

Symlinks: what's new in V8.4

- Interface and metadata changes
- RMS enhancements
  - Fuller support for POSIX pathnames
  - Fuller support for Logical Names in POSIX paths
  - Search List support
  - Wildcards
- Symlink compatibility between V8.3 and V8.4
  - Converting VMS 8.3 Symlinks to VMS 8.4 Symlinks

Programming interface changes

- No changes needed in your program, if using:
  - DCL commands
  - Lexicals
  - C RTL
  - RMS
- Changes are in ACP QIO interface
  - You are impacted if you use ACP QIO and deal with symlinks
  - RMS and CRTL interface to XQP suitably updated

ACP QIO changes

- New flags defined to request/identify symlink
- Naming convention consistency – ‘special’ instead of ‘symlink’
- Need to re-compile ACP QIO program!

On V8.3

```
! INPUT: Operate on symlink, not target
MY_FIB [FIB$V_SYMLINK] = 1;

! Make QIO call
status = SYS$QIOW (..., IO$_ACCESS, ...,
         FIB_DESC, file_desc, ...);

! OUTPUT: Did XQP operate on symlink?
IF (.MY_FIB [FIB$V_SYMLINKENTRY]) THEN
   :
   :
```

On V8.4

```
! INPUT: Operate on symlink, not target
MY_FIB [FIB$V_CHECK_SPECIAL] = 1;

! Make QIO call
status = SYS$QIOW (..., IO$_ACCESS, ...,
         FIB_DESC, file_desc, ...);

! OUTPUT: Did XQP operate on symlink?
IF (.MY_FIB [FIB$V_IS_SPECIAL]) THEN
   :
   :
```

V8.3 Metadata limitation

- No symlink indicator in directory entry
  - Was defined but not actually used
  - Indicator was only in File Header
- Caused a File Header read
  - Two issues:
    - Extra overhead
    - False Audit Alarm
- Corrected with new DIR$V_SPECIAL flag in directory entry

Metadata: Directory Entry Flags

- Directory entry -> Flags Byte -> New flag bit to **indicate symlink**
  - DIR$V_SPECIAL in DIR$B_FLAGS
  - Same as DIR$V_NEXTREC
- XQP enforces relation to File Header symlink attributes
  - Creating symlink sets entry flag AND header attributes

```
$ dump/dir v83_dir.dir

0000  Directory Entry:
0000    Size:              20
0002    Version limit:     32767
0004    Type:              0  (FID)
0004    Name type:         0  (ODS-2)
0005    Name count:        7
0006    Name:              SRC.LNK
000E    Version:     1     FID:  (14,1,0)

0016  End of records (-1)
```

```
$ dump/dir v84_dir.dir

0000  Directory Entry:
0000    Size:              20
0002    Version limit:     32767
0004    Type:              0  (FID)
        DIR$V_NEXTREC bit set
0004    Name type:         0  (ODS-2)
0005    Name count:        7
0006    Name:              SRC.LNK
000E    Version:     1     FID:  (14,1,0)

0016  End of records (-1)
```

V8.3 False AUDIT ALARM

– Only File Header can tell if file is symlink or not

– For a dir lookup (RMS $SEARCH), XQP has to read File Header

– Read-Attributes triggers access failure audit if enabled

```
%%%%%%%%%%  OPCOM   2-NOV-2007 09:00:39.97  %%%%%%%%%%
…
Auditable event:          Object access
Event information:        read file attributes request (IO$_ACCESS or IO$_CREATE)
…
Process owner:            [100,101]
Terminal name:            TNA4:
Object class name:        FILE
Object owner:             [100,100]
Object protection:        SYSTEM:RWE, OWNER:RWE, GROUP:E, WORLD:E
File name:                _$80$DKB0:[TEST]TEST.TMP;1
File ID:                  (42504,40,0)
Access requested:         READ
…
Status:                   %SYSTEM-F-NOPRIV, insufficient privilege or object
protection violation
```

Metadata: Volume Characteristics

– Home block -> Volume characteristics field -> New flag bit
  - HM2$V_NO_SPECIAL_FILES in HM2$W_VOLCHAR
– $ SET VOLUME, new parameter NOSPECIAL_FILES
– Eliminates False Audit Alarm

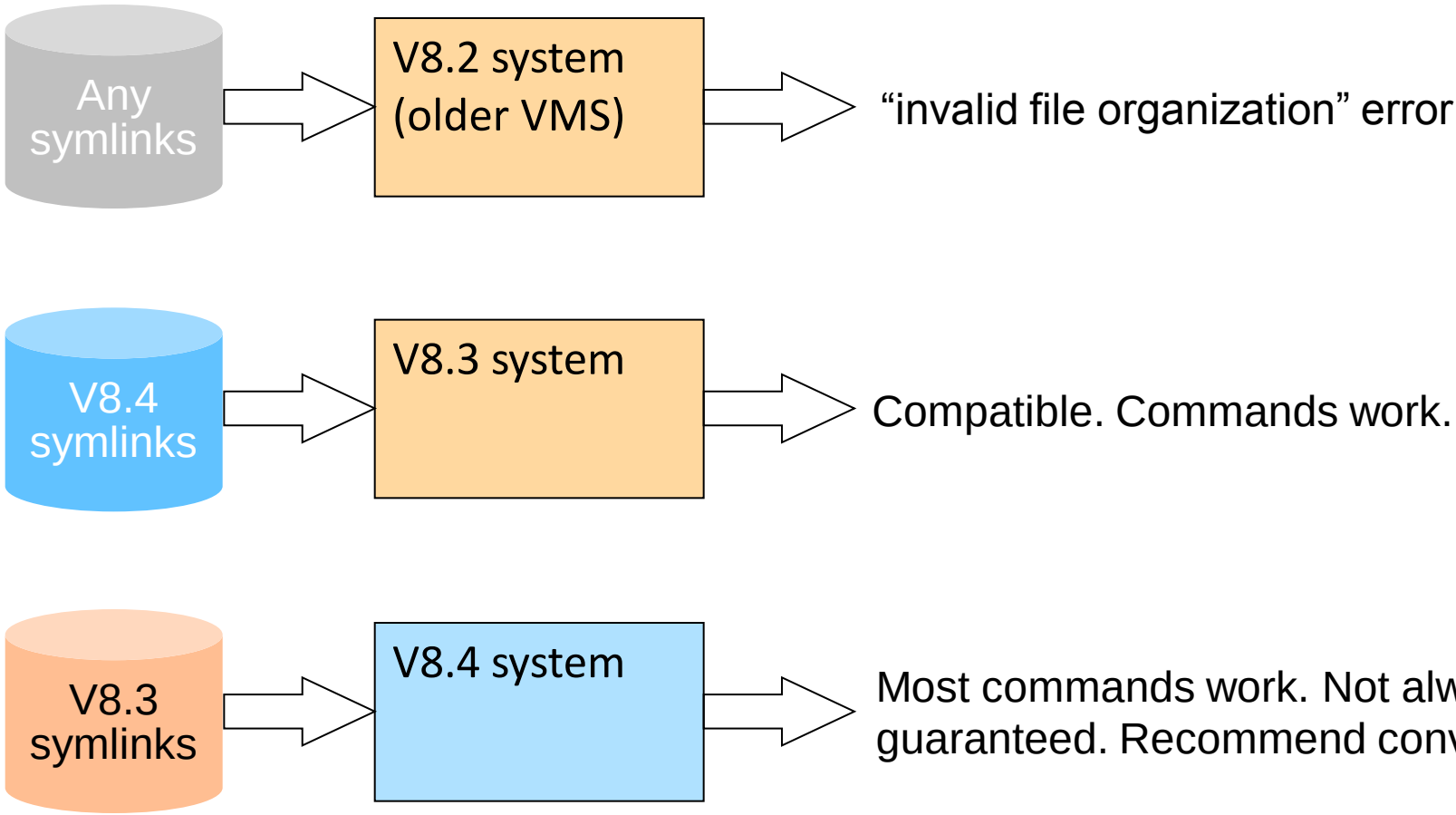
```
$ show dev /full mda0
  :
  :
  Volume Status:  ODS-5, subject to mount verification, file high-water marking,
      XFC caching is disabled, write-back XQP caching enabled, special files
      enabled.

$ set volume mda0 /volume_characteristics = NOSPECIAL_FILES

$ show dev /full mda0
  :
  :
  Volume Status:  ODS-5, subject to mount verification, file high-water marking,
      XFC caching is disabled, write-back XQP caching enabled.

$
```

COMPATIBILITY: V8.3, V8.4 SYMLINKS



ANALYZE /DISK /REPAIR

– $ analyze/disk checks consistency of symlink attributes/flags
  - Checks if dir entry flag DIR$V_SPECIAL corresponds to File Header file org FAT$C_SPECIAL
  - Checks version limit = 1 for symlink file

– /REPAIR ‘upgrades’ V8.3 symlink entry to V8.4 entry
  - This is recommended.

```
$analyze/disk/repair $10$DKB200:
…
%ANALDISK-W-BADSYMENTRY, directory entry for SYM_ON_83. in directory
(61531,39558,0) example.link  does not match symlink attribute in file
header
…
```

RMS: SYMLINKS AS LOGICAL NAMES

- Translates the first element of an absolute pathname
- Equivalence string in the path is substituted in the path if the translation succeeds.
- Logical Name can be a search list.

RMS: SYMLINKS IN SEARCH LISTS

```
$ define example SYS$SYSDEVICE:[example1.],SYS$SYSDEVICE:[example2.]
$ create/symlink="/example" search_link
$ dir [.search_link]

Directory SYS$SYSDEVICE:[000000.SEARCH_LINK]

EXAM1_FILE1.DAT;1   EXAM1_FILE2.DAT;1   EXAM1_FILE3.DAT;1   EXAM2_FILE1.DAT;1
EXAM2_FILE2.DAT;1

Total of 5 files.
$ ! Note we see files from both directories in the search list
```

RMS: SYMLINKS WILDCARD CONTROL

– In a $SEARCH operation, whether symlinks are to be followed or not depends on how the namespace is structured and the intent of the user.

– A user can control this using set process/symlink command

```
SET PROCESS/SYMLINK=keyword

        NOWILDCARD          do not follow symlinks in directory wildcarding

        WILDCARD            follow symlinks in all wildcarded directory
                            specifiers

        NOELLIPSIS          follow symlinks matched by any wildcard fields in
                            the directory string except ellipsis

        ELLIPSIS            equivalent to WILDCARD (included for command
                            symmetry)
```

Symlinks Summary: What You Need to Do

- Make coding change if using the ACP $QIO interface.
- Recommend converting existing symlinks to V8.4 using ANALYZE DISK /REPAIR.
- Use new RMS features