z/OS V2.5 IBM Education Assistant

Solution Name: mmap() 64-Bit Support

Solution Elements: z/OS UNIX BCP RSM/SMF

August 2021

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - none

Objectives

- Learn about 64-bit mmap() features.
- Learn how to use the changed syscalls.

- Background:
  - The mmap functions establish a mapping between an address space of a process and a file. The mmap syscall allocates storage such that when the storage is accessed by the process, data in the file is read into the storage and when storage is modified, the corresponding file can be subsequently modified.

- The main idea:
  - Remove address and length restrictions for the mmap syscalls to allow use of above the bar storage.

Overview

- Who (Audience)
  - Application developers using z/OS UNIX mmap() support.
- What (Solution)
  - Enhance the z/OS UNIX mmap() support to enable applications to utilize their calling process memory regions that are above the bar for mmap use.
  - By using the new support applications can map significantly larger amounts of data at one time, thereby potentially improving performance.
  - A new map type is provided to cause maps to use above the bar storage.
- Wow (Benefit / Value, Need Addressed)
  - Improve application access to large amounts of data
  - Alleviate below the bar memory constraints when mapping large files.

Usage & Invocation – General

- General characteristics:
  - A new Map_type is supported – MAP_64
  - The Map_address parameter will now allow values greater than 64G
  - The Map_length parameter will now allow values greater than 2G
  - Maps can use above the bar storage via 3 mechanisms:
    - Specifying MAP_64 for the Map_type
    - Specifying a value greater than 2 GB for the Map_Length
    - When providing a suggested Map_address, specifying an address of 64G or greater
  - All protect options are supported
  - Storage can be key 2 or key 8.
  - Allowance has been added for 31 bit assembler callers to establish above the bar maps.
  - Syscalls in assembler, C and Rexx are supported.
  - Fork will propagate above the bar maps to the child just like below the bar maps.
  - MAP_SHARED and MAP_FIXED supported.
  - Above the bar maps will start on a segment or megabyte boundary.
  - SMF option MAXSHARE will govern usage of above the bar maps
- Restrictions
  - Map_address must not be in the range 2G to 64G
  - When a file is mapped, all future maps to that file must be mapped using the same storage parameter until all maps to that file are unmapped.
  - MAP_MEGA and MAP_PRIVATE are not supported for above the bar maps.

Usage & Invocation – Assembler Syscalls

Assembler syscall changes

- BPX4MMP, BPX4MMP, BPX4MSY, BPX4MUN
  - Map_type - A new option, MAP_64, will cause the map to use above the bar storage.
  - Map_address - Value can now point to above the bar storage. mmap() will return an above the bar address if requested.
  - Map_length – Value can be up to 64 bit value
- BPX1MMP, BPX1MMP, BPX1MSY, BPX1MUN
  - Map_type - A new option, MAP_64, will cause the map to use above the bar storage.
  - Map_address, Map_length:
    - Allowance is made to pass a 64 bit value in the 32 bit parameter
    - If the high order bit is on, The lower 31 bits is interpreted as an address which points to a 64 bit value.
    - If MAP_64 is specified, or Map_length is >2G then Map_addr MUST be an address pointing to a 64 bit value.

Usage & Invocation – Rexx Syscalls

Rexx syscall changes

- Constants are now provided for the mmap syscalls:
  - Map_type: MAP_SHARED, MAP_PRIVATE, MAP_MEGA, MAP_FIXED, MAP_64
  - Protect_options: PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
- mmap syscall modified
  - Add map_type and protect_options parameters
  - Add support for MAP_64 map_type
  - Address_variable and length can be >2G
    - Remember: if you want to access above the bar storage, numeric digits must be increased:
      - Ex. numeric digits 20

```
mmap --- file_descriptor --- file_offset --- address_variable --- length ----------------------------------------

                                                                |--- map_type ----------------------

                                                                          |--- protect_options
```

Usage & Invocation – Rexx Syscalls

- mprotect syscall added
  - This syscall was missing so is added.
  - Address_variable and length can be >2G
- protect_option is the same as on mmap syscall.

```
mprotect --- address --- length --- protect_option
```

- msync and munmap changed to allow above the bar maps
  - address parameter can be >64G
  - length parameter can be >2G

Sample code:

```
mapopts=MAP_SHARED+MAP_64
address syscall 'mmap (fd) 0 addr (maplen) (mapopts)’
address syscall 'msync  (addr) (maplen)'
mapprot=PROT_READ
address syscall 'mprotect (addr) (maplen) (mapprot)'
address syscall 'munmap (addr) (maplen)'
```

Usage & Invocation – C Syscalls

- Mmap(), mprotect(), msync(), munmap() changed to allow above the bar maps
  - addr argument can be >64G
  - len argument can be >2G
  - Flags argument allows _MAP_64
  - Programs must be compiled 64 bit to use above the bar maps
  - Environment variable _EDC_AUTO_MAP64 can be set to ON and all maps will be above the bar maps.

Sample code:

```
flags = MAP_SHARED + _MAP_64;
av = mmap(addr,len,prot,flags,fd,off);
```

Usage & Invocation – SMF MAXSHARE

- The maximum number of shared views that can be created through IARVSERV SHARE services by unauthorized callers can be set in one of two ways:
  - The IEFUSI installation exit, word 7 subword 4.
  - The SMFLIMxx parmlib member MAXSHARE attribute, new with this support.
- The IBM-supplied default is 32. The maximum value is 2,147,483,647 (or 2^{31}-1).
  - Note that each shared view is represented by 64 bytes of real storage.
- The SMFLIMxx MAXSHARE attribute can be combined with any of the existing filters or attributes.
  - Systems without MAXSHARE support (systems prior to z/OS 2.4, or z/OS 2.4 without OA60316 installed) will invalidate any rule that includes MAXSHARE.
  - You should add a new rule to set MAXSHARE rather than adding it to an existing rule if you share a SMFLIMxx parmlib member with systems that do not support MAXSHARE.
- For example, to set a MAXSHARE value of 200,000 shared pages for JOB1, the following SMFLIMxx rule can be coded:

```
REGION JOBNAME(JOB1) MAXSHARE(200000)
```

Interactions & Dependencies

- Software Dependencies
  - none
- Hardware Dependencies
  - none
- Exploiters
  - none

Upgrade & Coexistence Considerations

- No sysplex considerations
- No toleration/coexistence requirements
- No upgrade considerations
- No coexistence considerations

Installation & Configuration

- z/OS V2.4 APARs:
  - OA60306
  - OA60310
  - OA60316
  - PH32235
- See the MAXSHARE slide for details on SMF configuration

Summary

- This solution allows applications to more easily access large amounts of data and alleviate constraints on above the bar storage.

Appendix

Publications

- z/OS UNIX System Services Programming: Assembler Callable Services Reference (SA23-2281)
- z/OS XL C/C++ Runtime Library Reference (SC14-7314)
- z/OS Using REXX and z/OS UNIX System Services (SA23-2283)
- SA23-1380 z/OS MVS Initialization and Tuning Reference
  - Describes the SMFLIMxx syntax
- SA23-1381 z/OS MVS Installation Exits
  - Describes the IEFUSI installation exit
- SA38-0675 z/OS MVS System Messages, Vol 8 (IEF-IGD)
  - Describes messages IEF043I, IFA900I