IBM Education Assistance (IEA) for z/OS V2R3

Line Item Name: Year 2038/2042 support, StackProtect support
Element/Component: Language Environment

Agenda

- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- Session Summary
- Appendix

Trademarks

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

Session Objectives

- Explain the purpose of working on Language Environment V2R3 line items: Year 2038/2042 support, StackProtect support

Overview

- Line Item: Year 2038/2042 support
- Problem Statement / Need Addressed
  - The Year 2038 issue is also known as the UNIX Y2K problem.
  - In UNIX-like systems, time values are stored or calculated as a signed 32-bit integer (time_t), and this number is interpreted as the number of seconds since 00:00:00 UTC on January 1, 1970 ("the epoch"). Such implementations cannot encode times after 03:14:07 UTC on January 19, 2038.
    - On z/OS, time_t is type long - the C/C++ compiler generates a signed 32-bit integer (in AMODE 31).
  - Trying to represent a value beyond the latest time in time_t will result in an integer overflow. The time value will "wrap around" and be stored internally as a negative number, which will be interpreted as a time value of December 13, 1901 rather than January 19, 2038.
  - This is not related to the Year 2042 TOD Clock issue on z Systems, though we are addressing both issues at the same time in z/OS over multiple releases.

Overview

- Solution
  - In z/OS V1.9, Language Environment (the C run-time library) provided support to allow referencing a future date beyond 2038 - constructed time
  - In z/OS V2.3, Language Environment will provide replacements for all the other structures and functions that reference a time value.

- Benefit / Value
  - High level language application is enabled to run with system dates and times beyond year 2038 and 2042.

Usage & Invocation

- Structures

Existing	New
struct stat	struct stat64
struct utimbuf	struct utimbuf64
struct utmpx	struct utmpx64
union ipcqproc	union ipcqproc64
struct msgid_ds	struct msgid_ds64
struct semid_ds	struct semid_ds64

Existing	New
struct shmid_ds	struct shmid_ds64
struct w_psproc	struct w_psproc64
struct timeb	struct timeb64
struct f_attributes	struct f_attributes64
struct msgxbuf	struct msgxbuf64
struct timespec	struct timespec64

Usage & Invocation

- Functions

Existing	New
fstat()	fstat64()
lstat()	lstat64()
stat()	stat64()
__readdir2()	__readdir2_64()
__open_stat()	__open_stat64()
ftw()	ftw64()
nftw()	nftw64()
utime()	utime64()
utimes()	utimes64()
pututxline()	pututxline64()
getutxent()	getutxent64()
getutxid()	getutxid64()

Existing	New
getutxline()	getutxline64()
__getipc()	__getipc64()
msgctl()	msgctl64()
semctl()	semctl64()
shmctl()	shmctl64()
w_getpsent()	w_getpsent64()
ftime()	ftime64()
__chattr()	__chattr64()
__fchattr()	__fchattr64()
__lchattr()	__lchattr64()
msgxrcv()	msgxrcv64()
pthread_cond_timedwait()	pthread_cond_timedwait64()

Usage & Invocation

- What Should I do?
- Investigate to see if your AMODE 31 C/C++ application uses time_t and calculate dates into the future or if the application will live beyond Year 2038.
  - Remember, AMODE 64 C/C++ programs are not affected.
- If changes are required in your application then assess the changes necessary to use the new structures and functions we have introduced.
  - By the way, it may be easier to convert the program to 64-bit (AMODE 64).
- If your application uses C functions beyond those provided by the C run-time library of Language Environment, then you will need to investigate the availability of those interfaces.
- Use _LARGE_TIME_API feature test macro in your program as appropriate to expose new structures and functions.

Overview

- Line Item: Language Environment StackProtect support
- Problem Statement / Need Addressed
  - Stack overflow issues in applications need to be detected early to prevent them from causing program misbehavior or from becoming serious security vulnerabilities.

Overview

- Solution
  - With STACKPROTECT enabled, code will be added to check the function return address upon completion of a function and fail fast if the return address was overwritten.

- Benefit / Value
  - StackProtect support will aid in the detection of buffer overflows and enhance the security of Language Environment programs.

Usage & Invocation

- Enabling StackProtect support
  - New XL C/C++ compiler option STACKPROTECT
  - A new keyword STCKPROT is added to macros that generate the prolog
  - A new keyword STKPROT is added to macros that preinitialize subroutine environment
  - Main routine has to be enabled if you want to turn ON stack protect support
- If the StackProtect is enabled:
  - A certain field in the DSA will be altered to protect the DSA
  - If buffer overflow is detected when program runs, Abend 4088-96 will be raised.

Usage & Invocation

- SYSDUMP support
  - Warning message will be issued in LEDATA 'STACK' section when buffer overflow is identified and StackProtect is enabled.
- PPA1 Updates
  - A certain bit indicates if StackProtect is enabled or not.
- PPA1 Updates – CEEPPA macro
  - A keyword added to Indicate whether this procedure has StackProtect enabled.

Session Summary

- The following z/OS V2R3 Language Environment line items have been explained:
- Year 2038/2042 support
- StackProtect support

Appendix

- Publications
  - z/OS XL C/C++ Runtime Library Reference (SC14-7314)
  - z/OS XL C/C++ Programming Guide (SC14-7315)
  - z/OS Language Environment Programming Guide (SA38-0682)
  - z/OS Language Environment Debugging Guide (GA32-0908)
  - z/OS Language Environment Programming Guide for 64-bit Virtual Addressing Mode (SA38-0689)
  - z/OS Language Environment Vendor Interfaces (SA38-0688)
  - z/OS Language Environment Runtime Messages (SA38-0686)