International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Operating Systems Platform Migration of Multi – Threaded C++ Application from HPUX to RHEL

Siva Sathyanarayana Movva

Email: *sivasathya[at]gmail.com*

Abstract: ***Global Payments or Transactions happen in the form of messages for cash transfers, securities, metals, and various other needs generating millions of data daily and totaling up to a billion in a year. Along with the central applications handling the msg transfer end to end numerous other applications are needed to serve the purpose of reconciliation, business intelligence, tracking of payments and other customer specific requirements. All these application have been developed on HPUX platform more than a decade ago and have been migrated to RHEL driven by business needs/challenges and technology changes. The intent of this paper is to provide the high - level approach, technical solution and constraints and provide standard LINUX Operating Systems that meet established security baselines that are deployed by automation.***

Keywords: payments, transactions, messages, HPUX, RHEL, application, multi - threading, platform

1. Introduction

Red Hat Enterprise Linux 7 was released in June 2014, and has a full support life - cycle of 5 years and Maintenance Support until 2024. Red Hat Enterprise Linux 8 was released in May 2019. The first minor release (8.1) was available in Nov 2019. End of full support of RHEL 8 is scheduled for May 2024 and Maintenance Support 2029. Red Hat Linux standard release cycle for a minor release is 6 months. During this cycle the release is in full support and patches are released on a continuous basis.
Once a new release is made available patches are no longer applied to the previous releases.
Extended update support is available for an additional subscription cost to back port critical patches to previous minor releases for a maximum of 24 months from the initial release date.

Application Changes done for compatibility on RHEL:

- Handling socklen_t which is different across platforms.
- Handling of issue with different approach permissions on named unix sockets.
- Handling of semi standard MAXHOSTNAMELEN.
- Use of correct standard header files instead of overly inclusive ones.
- Handling misuse of pre - processor line continuation in C++ code.
- Eliminate references to X11 approach unused for years.
- Correct misuse of ## operator from C pre - processor to just use constant string concatenation at compile time.
- RHEL support in Imakefiles.
- Reorder of code to remove warnings of no return when throwing exceptions instead.
- Removal of use of NULL as a scalar 0. It is not an integer type conceptually and should not be used as one.
- Fixed warnings related to 32/64 integers for formatting printing.
- Added pre - processor symbols needed to get portability functions exposed form glib/clib
- Handled code added by CM for HPUX what that is now unsupported by CM leaving misinformation in objects.
- Provision of function mkdirp () that doesn't exist on Linux.
- Taking care of misuse of "char *" to point at constant strings.
- Smarter handling of system dependant "snprintf" calls. Use proper c++ constructors.
- Fixed badly defined function macros to be correctly defined using do{}while (0) to allow safe use mixed with conditionals and other C/C++ statements.
- RHEL does not have some signals we expected - >which signals.
- Moved from #ifdef to #if defined () to allow multiple definitions to trigger conditional compilation.
- Changed to the Posix preferred method of turning off terminal echo.
- Fstat on HPUX is non standard. Had to use and external call to stat –f.
- We need to perform an expilict, case insensitive check for references to HP - UX and HPUX
- ps –w was ps –xx in Linux
- netstat option was different in HP as in Linux
- Another issue was in JNI. Library locations needed to be fully qualified.

Some Issues encountered on RHEL:

- Found issues with hard coded paths in many scripts that needed to be sourced in
- 1. Red - hat Linux doesn't support the following system library which is specific to HP - UX. sys/pstat. h - The HP - UX pstat facility is an Application Programming Interface (API) that returns detailed information about many aspects of a running kernel.
- The SWITCH source files which would be using the pstat APIs have to be updated with the Linux specific API "/proc/<PID>/status".
- The new API would return the process state code of a running process. The following are the different process state codes and the corresponding state description.

D - Uninterruptible sleep (usually IO)
R - Running or runnable (on run queue)
S - Interruptible sleep (waiting for an event to complete)
T - Stopped, either by a job control signal or because it is being traced.
W - Paging (not valid since the 2.6. xx kernel)
X - Dead (should never be seen)

Z - Defunct ("zombie") process terminated but not reaped by its parent.

2. itoa () and ltoa () functions don't work with GCC compiler on Linux platform, so these have to be coded. The following source code would use the following work - around,

```
inline static const char* < itoa | ltoa > (< int | long> input)
{
std:: stringstream out;
std:: string str;
out << input;
str = out. str ();
return str. c_str ();
};
```

const string& - For a const string, begin () and end () methods return std:: string:: const_iterator. So it's not possible to type case it to a std:: string:: iterator.

The work - around would be to change the type definition of "typedef string:: iterator iter; " as "typedef string:: const_iterator iter; "

1) endl is of unknown type - Explicitly add std:: before anything in the std namespace if the source code doesn't include the line using namespace std;
2) const char* - The GCC compiler doesn't let the type conversion from 'const char*' to 'char*, but the other way around is possible. So when ever this kind of build error occurs, it's required to change the declaration as 'const char*'.
3) strlen () - The strlen () function on a NULL pointer will crash the application, so the code has to be explicitly check for NULL pointer before calling strlen ().

Patching and upgrades wrt RHEL

New content view versions will be generated quarterly to enable patching. A minor release content view will subscribe to extended update service to prolong the ability to patch without upgrading to the next minor release. Access to additional repos is not supported in EUS mode. Additional repos include JBOSS, RHDS, EPEL, and extras

Platform owners will have an option to upgrade to the next minor release at any point in the life - cycle.

Red Hat does not recommend upgrading between major releases and recommended consulting services if this is required. Linux Platform Squad will not support upgrading between Major releases.

Linux Platform squad does not plan to release new images for non - EUS minor releases in RHEL 8 (8.3, 8.5, 8.7). Upgrades to even numbered minor versions is possible.

2. Conclusion

1) New satellite infrastructure will enable periodic content updates to provide patches to current releases.
2) Content views are defined in satellite and specification of the version of RPMs to be installed done.
3) Creation of VM playbook installation and configuration of the operating systems done.
4) Reusable playbook installation and configures the extended platform.
5) Extend Platform components are installed with an ansible playbook and the associated content view.
6) Platform releases have specific extend component versions as configured in the Software content view.
7) If new extended platform components are required a content view update will have to be performed.

References

[1] Berg, Formal Methods of Program Verification and Specification, Prentice Hall, 1982.
[2] B. W. Boehm, "A Spiral Model of Software Development and Enhancement", IEEE Computer, pp.61 - 72, May 1988.
[3] G. Booch, Object - Oriented Design with Applications, Benjamin Cummings, 1991.
[4] S. Adve and K. Gharachorloo. Shared memory consistency models: a tutorial. IEEE Computer, 29 (12): 66–76, Dec.1996.
[5] B. L. O. Andersen. Program Analysis and Specialization for the C Programming Language. PhD thesis, DIKU, University of Copenhagen, May 1994.
[6] S. R. Ladd, Turbo C++ Techniques and Applications, M T Books, 1990.
[7] S. B. Lippman, C+ + Primer, Addison Wesley, 1991.
[8] P. J. Lukas, The C+ + Programmers Handbook, Prentice Hall, 1992.
[9] B. Meyer, Object - Oriented Software Construction, Prentice Hall, 1988.
[10] A. Aiken and D. Gay. Barrier inference. In Proceedings of the 25th Annual ACM Symposium on the Principles of Programming Languages, Paris, France, Jan.1998. ACM
[11] R. R. Seban, A Temporal Logic for Proofs of Correctness of Distributed Protocols, March 1993.
[12] R. R. Seban, An Introduction to Object - Oriented Design with C++, December 1992.
[13] I. Sommerville, Software Engineering, Addison Wesley, 1992.
[14] B. Stroustrup, The C++ Programming Language, Addison Wesley, 1989.
[15] R. H. Thayer, "System and Software Requirements Engineering", IEEE Tutorial, 1990.
[16] Callahan, K. Kennedy, and J. Subhlok. Analysis of event synchronization in a parallel programming tool. In Proceedings of the 2nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Seattle, WA, Mar.1990.
[17] D. Chase, M. Wegman, and F. Zadek. Analysis of pointers and structures. In Proceedings of the SIGPLAN '90 Conference on Program Language Design and Implementation, pages 296–310, White Plains, NY, June 1990. ACM, New York.
[18] Curtis Anderson: xFS Attribute Manager Design. Technical Report, Silicon Graphics, October 1993. https: //oss. sgi. com/projects/xfs/design_docs/xfsdocs93_pdf/attributes. pdf

[19] Austin Common Standards Revision Group. https: //www.opengroup. org/austin/
[20] Steve Best, Dave Kleikamp: How the Journaled File System handles the on - disk layout. IBM developerWorks, May 2000. https: //www - 124. ibm. com/developerworks/oss/jfs/
[21] B. Callaghan, B. Pawlowski, and P. Staubach: NFS Version 3 Protocol Specification. Technical Report RFC 1813, Network Working Group, June 1995.
[22] Andreas Dilger: [RFC] new design for EA on - disk format. Mailing list communication, July 10, 2002. https: //acl. bestbits. at/pipermail/acl - devel/2002 - July/001077. html
[23] Marius Aamodt Eriksen: Mapping Between NFSv4 and Posix Draft ACLs. Internet Draft, October 2002. https: //www.citi. umich. edu/u/marius/draft - eriksen - nfsv4 - acl - 01. txt
[24] Andreas Grünbacher: Known Problems and Bugs in the Linux EA and ACL implementations. March 20, 2003. https: //acl. bestbits. at/problems. html
[25] Andreas Grünbacher: Preserving ACLs and EAs in editors and file managers. February 18, 2003. https: //www.suse. de/~agruen/ea - acl - copy/ for a description.
[26] Hewlett - Packard: acl (2): Set a file's Access Control List (ACL) information. HP - UX Reference. https: //docs. hp. com/
[27] Hewlett - Packard: acl (4): Access control list. Compaq Tru64 Reference Pages. https: //www.hp. com/
[28] IEEE Std 1003.1 - 2001 (Open Group Technical Standard, Issue 6), Standard for Information Technology - - Portable Operating System Interface (POSIX) 2001. ISBN 0 - 7381 - 3010 - 9. https: //www.ieee. org/
[29] IEEE 1003.1e and 1003.2c: Draft Standard for Information Technology - - Portable Operating System Interface (POSIX) - - Part 1: System Application Program Interface (API) and Part 2: Shell and Utilities, draft 17 (withdrawn). October 1997. https: //wt. xpilot. org/publications/posix.1e/
[30] Jim Mauro: Controlling permissions with ACLs. Describes internals of UFS's shadow inode concept. SunWorld Online, June 1998.
[31] Microsoft Platform SDK: Access Control Lists. https: //msdn. microsoft. com/
[32] Mark Lowes: Proftpd: A User's Guide March 31, 2003. https: //proftpd. linux. co. uk/
[33] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck: NFS version 4 Protocol. Technical Report RFC 3010, Network Working Group, December 2000.
[34] Silicon Graphics: acl (4): Access Control Lists. Irix manual pages. https: //techpubs. sgi. com/
[35] J. Spadavecchia, E. Zadok: Enhancing NFS Cross - Administrative Domain Access. Proceedings of the Annual USENIX Technical Conference, FreeNIX Track, Pages 181 - 194. Monterey, CA, June 2002.
[36] W. Richard Stevens: Advanced Programming in the UNIX (R) Environment. Addison - Wesley, June 1991 (ISBN 0 - 2015 - 6317 - 7).
[37] Storage Networking Industry Association: Common Internet File System Technical Reference. Technical Proposal, March 2002. https: //www.snia. org/tech_activities/CIFS/
[38] Sun Microsystems: acl (2): Get or set a file's Access Control List. Solaris 8 Reference Manual Collection. https: //docs. sun. com/
[39] Sun Microsystems: NFS: Network file system protocol specification. Technical Report RFC 1094, Network Working Group, March 1989.
[40] Robert N. M. Watson: acl (3): Introduction to the POSIX.1e ACL security API. FreeBSD Library Functions Manual. https: //www.FreeBSD. org/
[41] Robert N. M. Watson: TrustedBSD: Adding Trusted Operating System Features to FreeBSD. USENIX Technical Conference, Boston, MA, June 28, 2001. https: //www.trustedbsd. org/docs. html
[42] Robert N. M. Watson: Introducing Supporting Infrastructure for Trusted Operating System Support in FreeBSD. BSDCon 2000, Monterey, CA, September 8, 2000. https: //www.trustedbsd. org/docs. html
[43] Robert N. M. Watson: Personal communication. March 28, 2003.