Silicon Graphics IRIX™ 3.2 Operating System Verification

Terance L. Lam and Pamela P. Walatka
Computer Sciences Corporation
Numerical Aerodynamic Simulation Division
NASA Ames Research Center

Abstract

This document discusses the experience of compiling a Silicon Graphics IRIX™ 3.2 Operating System from source code on a 4D workstation and its verification using the AT&T System V ™ Verification Suite.

I. Introduction

NASA's NAS division has a history of building operating systems from source code for the engineering workstations it has acquired. This practice is to ensure that the operating system can be customized to fit NAS needs. An operating system has to be verified. Previously, NAS would build an operating system and then install it on a workstation. If the operating system did not fail under use within a certain time interval, it was considered functional. User tests helped to bring out some operating system flaws, but it is unlikely that this level of testing exercised all commands and functions of an operating system. There is a need for a systematic approach to verifying an operating system.

The AT&T System V Verification Suite (SVVS™) is a set of tests that check whether an operating system conforms to the specifications in the System V Interface Definitions (SVID™). The IRIX 3.2 operating system complies with SVID Issue 3 and with BSD 4.3 inter-networking interfaces. It should conform to SVID and pass all applicable tests of SVVS.

This project is divided into two phases. The first phase builds an IRIX 3.2

operating system from source code while the second phase verifies it with SVVS.

II.Building an IRIX System V Release 4D1-3.2D Operating System

Mk is a shell script SGI™ uses to control software builds on the 4D workstations. This program builds an IRIX operating system. The following list summarizes the procedure to build an IRIX operating system from source code (a detailed section of a build is listed in Appendix A of this document).

- Set up environment variables for an operating system build.
- Use tlink to create header files.
- Install header files.
- Compile source code and create object files common to all 4D family workstations.
- Link all object modules to build an operating system.

This process builds a System V operating system and NFS™ from source code. It does not build any of the compilers because they are proprietary to MIPS™. SGI's System V source tape does not include source code for their libraries (e.g. gl™). We can only link those libraries existing on the workstation to the operating system built, unless the source code for the libraries is made available.

It takes approximately four hours to compile an operating system. A successfully-built operating system is stored in $ROOT/uts/mips/$PRODUCTbootarea, where ROOT is the working root and PRODUCT is the workstation model number. Once the operating system is installed on a workstation, this operating system is ready for verification.

Many discrepancies were encountered in obtaining the SGI source tape. First, SGI 's source tape only had the basic AT&T System V source code; it did not include NFS. Without NFS, Mk failed to build the operating system. It was then found that NAS did not hold an appropriate System V Re-

lease 3 source license for the SGI workstation.

III. SVVS Installation

The AT&T System V Verification Suite(SVVS) is a set of tests that check whether an operating system conforms to the specifications in the System V Interface Definitions (SVID) Issue 2 and 3. The operating systems to which conformance can be verified include System V Release 2.0, 2.1 and 3.0. SVVS also provides utilities to build a test suite, install it, verify the operating system and report its results. IRIX 3.2 complies with SVID Issue 3 with BSD 4.3 inter-networking interfaces; therefore, SVVS can be used to tests its conformance to the Base Kernel and the Kernel Extension of SVID. Appendix B is a SVVS Installation Summary which indicates tests installed.

- The Base (BA) tests check all system calls and libraries routines listed in the SVID Base and Base Addendum sections (see Volumes I and III of SVVS).
- The Kernel Extension (KE) tests all system calls listed in the SVID Kernel Extension section (see Volume I of SVVS).

The following list summarizes the procedure that must be completed to install SVVS:

- Check main memory of target system. A minimum of two megabytes is required.
- Set up development and target systems before installing the SVVS software.
- Load the SVVS software from tape.
- Edit config and sv_const.h. These system-dependent parameters can be found in /usr/include/limits.h.
- Build install and instmen.
- Run instmen or create a script file for install.
- Run install and check the results.
- Transfer the executable files to the target machine and run setaccess as

root.

After these procedures are completed, SVVS can be executed to verify an operating system.

IV. SGI IRIX 3.2 Operating System Verification

The Base Operating System and the Kernel Extension sections of SVVS are installed to verify the target operating system on ew07, a 4D/60 workstation. The Base Operation System consists of Base Environment, Base Operating System Services and Base Operating System Library. Results of this verification can be found in Appendix C, SVVS Verification Report.

A. Base Operating System

The Base Operating System (BA) tests check all system calls and library routines listed in the SVID Base and Base Addendum sections (see Volumes I and III of SVVS).

1)Base Operating System Environment

Base Operating System Environment section checks for the set up of eight different environments listed in Appendix B. Termio is the only test that failed (see Summary report for details). The failure is a result of SGI's enhancement to the header file /usr/include/sys/termio.h. Unsigned char c_cc[NCC] has been changed from 8 to 31.

2)Base Operating System Services

This section checks for all system calls under BA_OS in Appendix C. Three out of fifty eight tests failed. These failures, their error messages and causes are explained below.

getcwd - Get current working directory is tested with buf=array and size=-1. It expects a return value of NULL, and an errno of

ERANGE. This test failed because MIPS has modified this function call.

pipe - This test creates a pipe and tries to overflow it with PIPE_MAX+1 bytes of data hoping that it will be blocked. The test failed because the write did not block on PIPE_MAX+1 bytes written into the pipe. It blocks on a byte size of PIPE_SIZE (3*4096). This is a modification SGI made to the file system.

ulimit - Get and set user limit test writes a file of what the maximum process limit should be and expects a return value of zero. The test failed. SGI declared maximum possible file size of 2 gigabytes but my hard disk is not big enough to store a 2 gigabyte file. It failed the "no space on device" test. The kernel has been modified to write maximum size files of 20 megabytes and the test passed.

The majority of the IRIX 3.2 kernel failures are the result of SGI's enhancements to the file system. SGI changes the maximum block size from 512 to 1024 and allows a maximum file size of 2 gigabytes. These modifications allow faster block I/O transfer and larger files can be stored. Although these enhancements violate SVID, they do not affect the operating system's performance.

3) Base Operating System Library

The Base Operating System Library checks for libraries listed under Appendix B. It is found that a large number of the library routines failed. Perhaps this is because SVVS was originally developed to verify System V on a 3B2 computer .

SVVS uses the data structure in /usr/include/varargs.h as a base to manipulate a value expected from a library call. Since varargs.h is machine dependent, SGI has modified this data structure. SVVS calculates the expected value, assuming SGI's varargs.h is the same as the 3B2's. This

value was compared to the actual value returned. It found that these two values were different and assumed the library call returned an erroneous value. Values returned by library calls were verified correct. SVVS interpreted varargs.h differently; therefore, the BA_LIB test is not applicable for testing the libraries of the IRIX 3.2 operating system.

B Kernel Extension

The Kernel Extension (KE) tests all system calls listed in the SVID Kernel Extension section (see Volume I of SVVS). This section is made up of two subsections:

- Kernel Extension Environment
- Kernel Extension Operating System services.

The SGI IRIX 3.2 passed this verification.

V. Conclusion

The IRIX 3.2 operating system does conform to Issue 3 of SVID with enhancements to the file system. The majority of the failures in the verification are a result of SGI's Extent File System. Although this enhancement violates SVID, the failures are illegitimate. These enhancements do not affect the functionality of the operating system. The failures found in the libraries are the results of SVVS' incompatibilities with IRIX. The libraries are functioning properly.

An operating system has been installed on a 4D/70G workstation in addition to the one existing on ew07. Both operating systems have been running for two months without discrepancies. Although SVVS provides a means to verify an operating system's conformance to SVID, it does not guarantee to detect all flaws in the operating system. More tests should be performed to detect any other possible flaws. The next stage of testing will be to install the operating system on a Personal IRIS and run more applica-

tions to determine its reliability.

SGI workstations employ BSD networking features and SUN Microsystems' Network File System (NFS); its reliability is not verified by SVVS. SVVS only verifies Remote File System Services. There is a need to formally test the network functionality of the SGI workstations. Unfortunately, there is not a BSD network verification suite that can be used for this purpose.

IRIX , SGI and gl are trademarks of Silicon Graphics Computer Systems.
MIPS is trademark of MIPS computers.
NFS is a trademark of Sun Microsystems' Network File System.
SVVS, SVID and System V are trademarks of AT&T.

Appendix A.
Example of building an Operating System

Operating system for a particular machine is specified by PRODUCT (e.g. 4D60).

```
setenv ROOT /usr/u/wk/lam/3.2/4D        #workingroot
setenv REFSRC /usr/u/wk/lam/3.2/src     # reference
source tree
setenvVTABLE/usr/u/wk/lam/3.2/src/cmd/swtools/vtable
setenv PRODUCT 4D60                     # 4D/60
cd /usr/u/wk/lam/3.2/4D
mk tlink:cmd
mk tlink:head
mk tlink:lib
mk tlink:stand
mk tlink:uts
cd /usr/lam/wk/lam/3.2/4D/head
mk :install                             # install
headers
cd /usr/u/wk/lam/3.2/4D/uts/mips
mk PRODUCT=4D60 :headers                # make headers
mk PRODUCT=4D60 DBOPTS= :default_dirs   # make default
directories
mk PRODUCT=4D60 :unix.nfs               # make operat-
ing system with NFS
```

Appendix B.
System V Verification Suite Installation Report

Release 3.0
Date: Mar 19, 1990
Time: 16:43:50

Test Run 1

Section 1

BA/ENV/src/devcon/svcomp . . . . . .passed
BA/ENV/src/devnul/svcomp . . . . . .passed
BA/ENV/src/devtty/svcomp . . . . . .passed
BA/ENV/src/envvar/svcomp . . . . . .passed
BA/ENV/src/filsys/svcomp . . . . . .passed
BA/ENV/src/header/svcomp . . . . . .passed
BA/ENV/src/passwd/svcomp . . . . . .passed
BA/ENV/src/termio/svcomp . . . . . .passed
BA/LIB/src/abs/svcomp . . . . . .passed
BA/LIB/src/bessel/svcomp . . . . . .passed
BA/LIB/src/bsearch/svcomp . . . . . .passed
BA/LIB/src/clock/svcomp . . . . . .passed
BA/LIB/src/conv/svcomp . . . . . .passed
BA/LIB/src/crypt/svcomp . . . . . .passed
BA/LIB/src/ctermid/svcomp . . . . . .passed
BA/LIB/src/ctime/svcomp . . . . . .passed
BA/LIB/src/ctype/svcomp . . . . . .passed
BA/LIB/src/drand48/svcomp . . . . . .passed
BA/LIB/src/erf/svcomp . . . . . .passed
BA/LIB/src/exp/svcomp . . . . . .passed
BA/LIB/src/floor/svcomp . . . . . .passed
BA/LIB/src/frexp/svcomp . . . . . .passed
BA/LIB/src/ftw/svcomp . . . . . .passed
BA/LIB/src/gamma/svcomp . . . . . .passed
BA/LIB/src/getc/svcomp . . . . . .passed
BA/LIB/src/getenv/svcomp . . . . . .passed
BA/LIB/src/getopt/svcomp . . . . . .passed
BA/LIB/src/gets/svcomp . . . . . .passed
BA/LIB/src/hsearch/svcomp . . . . . .passed
BA/LIB/src/hypot/svcomp . . . . . .passed
BA/LIB/src/lsearch/svcomp . . . . . .passed
BA/LIB/src/matherr/svcomp . . . . . .passed
BA/LIB/src/memory/svcomp . . . . . .passed
BA/LIB/src/mktemp/svcomp . . . . . .passed

BA/LIB/src/perror/svcomp . . . . . .passed
BA/LIB/src/printf/svcomp . . . . . .passed
BA/LIB/src/putc/svcomp . . . . . .passed
BA/LIB/src/putenv/svcomp . . . . . .passed
BA/LIB/src/puts/svcomp . . . . . .passed
BA/LIB/src/qsort/svcomp . . . . . .passed
BA/LIB/src/rand/svcomp . . . . . .passed
BA/LIB/src/regexp/svcomp . . . . . .passed
BA/LIB/src/scanf/svcomp . . . . . .passed
BA/LIB/src/setbuf/svcomp . . . . . .passed
BA/LIB/src/setjmp/svcomp . . . . . .passed
BA/LIB/src/sinh/svcomp . . . . . .passed
BA/LIB/src/ssignal/svcomp . . . . . .passed
BA/LIB/src/string/svcomp . . . . . .passed
BA/LIB/src/strtod/svcomp . . . . . .passed
BA/LIB/src/strtol/svcomp . . . . . .passed
BA/LIB/src/swab/svcomp . . . . . .passed
BA/LIB/src/tmpfile/svcomp . . . . . .passed
BA/LIB/src/tmpnam/svcomp . . . . . .passed
BA/LIB/src/trig/svcomp . . . . . .passed
BA/LIB/src/tsearch/svcomp . . . . . .passed
BA/LIB/src/ttyname/svcomp . . . . . .passed
BA/LIB/src/ungetc/svcomp . . . . . .passed
BA/LIB/src/vprintf/svcomp . . . . . .passed

Section 2

BA/OS/src/abort/svcomp . . . . . .passed
BA/OS/src/access/svcomp. . . . . .passed
BA/OS/src/alarm/svcomp . . . . . .passed
BA/OS/src/chdir/svcomp . . . . . .passed
BA/OS/src/chmod/svcomp . . . . . passed
BA/OS/src/chown/svcomp . . . . . passed
BA/OS/src/close/svcomp . . . . . .passed
BA/OS/src/creat/svcomp . . . . . .passed
BA/OS/src/directory/svcomp. . . . . .passed
BA/OS/src/dup/svcomp. . . . . .passed
BA/OS/src/dup2/svcomp . . . . . .passed
BA/OS/src/exec/svcomp . . . . . .passed
BA/OS/src/exit/svcomp . . . . . passed
BA/OS/src/fclose/svcomp. . . . . .passed
BA/OS/src/fcntl/svcomp . . . . . .passed
BA/OS/src/ferror/svcomp. . . . . .passed
BA/OS/src/fopen/svcomp . . . . . .passed
BA/OS/src/fork/svcomp . . . . . .passed
BA/OS/src/fread/svcomp . . . . . .passed
BA/OS/src/fseek/svcomp . . . . . .passed
BA/OS/src/getcwd/svcomp. . . . . passed

BA/OS/src/getpid/svcomp. . . . . .passed
BA/OS/src/getuid/svcomp. . . . . .passed
BA/OS/src/ioctl/svcomp . . . . . .passed
BA/OS/src/kill/svcomp . . . . . passed
BA/OS/src/link/svcomp . . . . . .passed
BA/OS/src/lockf/svcomp . . . . . .passed
BA/OS/src/lseek/svcomp . . . . . .passed
BA/OS/src/malloc/svcomp. . . . . passed
BA/OS/src/mkdir/svcomp . . . . . passed
BA/OS/src/mknod/svcomp . . . . . .passed
BA/OS/src/mount/svcomp . . . . . passed
BA/OS/src/open/svcomp . . . . . .passed
BA/OS/src/pause/svcomp . . . . . .passed
BA/OS/src/pipe/svcomp . . . . . passed
BA/OS/src/popen/svcomp . . . . . passed
BA/OS/src/read/svcomp . . . . . .passed
BA/OS/src/rmdir/svcomp . . . . . .passed
BA/OS/src/setpgrp/svcomp . . . . . passed
BA/OS/src/setuid/svcomp. . . . . .passed
BA/OS/src/signal/svcomp. . . . . .passed
BA/OS/src/sigset/svcomp. . . . . .passed
BA/OS/src/sleep/svcomp . . . . . .passed
BA/OS/src/stat/svcomp . . . . . passed
BA/OS/src/stime/svcomp . . . . . .passed
BA/OS/src/sync/svcomp . . . . . .passed
BA/OS/src/system/svcomp. . . . . passed
BA/OS/src/time/svcomp . . . . . .passed
BA/OS/src/times/svcomp . . . . . .passed
BA/OS/src/ulimit/svcomp. . . . . .passed
BA/OS/src/umask/svcomp . . . . . passed
BA/OS/src/umount/svcomp. . . . . .passed
BA/OS/src/uname/svcomp . . . . . .passed
BA/OS/src/unlink/svcomp. . . . . passed
BA/OS/src/ustat/svcomp . . . . . .passed
BA/OS/src/utime/svcomp . . . . . .passed
BA/OS/src/wait/svcomp . . . . . .passed
BA/OS/src/write/svcomp . . . . . .passed
KE/ENV/src/header/svcomp . . . . . .passed
KE/OS/src/acct/svcomp . . . . . passed

Section 3

KE/OS/src/chroot/svcomp. . . . . .passed
KE/OS/src/msgctl/svcomp. . . . . passed
KE/OS/src/msgget/svcomp. . . . . passed
KE/OS/src/msgop/svcomp . . . . . passed
KE/OS/src/nice/svcomp . . . . . passed
KE/OS/src/plock/svcomp . . . . . .passed

KE/OS/src/profil/svcomp. . . . . .passed
KE/OS/src/ptrace/svcomp. . . . . .passed
KE/OS/src/semctl/svcomp. . . . . .passed
KE/OS/src/semget/svcomp. . . . . passed
KE/OS/src/semop/svcomp . . . . . passed
KE/OS/src/shmctl/svcomp. . . . . passed
KE/OS/src/shmget/svcomp. . . . . passed
KE/OS/src/shmop/svcomp . . . . . .passed
src/clraccess/svcomp. . . . . passed
src/driver/svcomp. . . . . passed
src/report/svcomp. . . . . passed
src/setaccess/svcomp. . . . . passed
src/svvs/svcomp . . . . . passed
src/tools/sv_abort/svcomp . . . . . passed
src/tools/sv_cmd/svar . . . . . passed
src/tools/sv_cmd/svcomp. . . . . . passed
src/tools/sv_sys/svar . . . . . passed
src/tools/sv_sys/svcomp. . . . . .passed
src/tools/sv_tools/svar. . . . . .passed
src/tools/sv_tools/svcomp . . . . . passed
src/tools/z_base/svar . . . . . .passed
src/tools/z_base/svcomp. . . . . .passed
src/tools/z_kext/svar . . . . . .passed
src/tools/z_kext/svcomp. . . . . .passed
src/tools/z_nse/./svar . . . . . passed
src/tools/z_nse/./svcomp . . . . . passed
src/tools/z_tie/../../../TI/LIB/src/tielib/svar. . . . . passed
src/tools/z_tie/../../../TI/LIB/src/tielib/svcomp . . . . . passed
src/tools/z_tie/./svar . . . . . .passed
src/tools/z_tie/./svcomp . . . . . passed

Section 4—Summary of Run # 1

Passed:	154	Not Supported:	0		
Failed: .	0	Not Executed:	0		
Executed:	154	Omitted:	0	Total Tests:	154

Itemization of diagnostics

Errors: . 0
Warnings: 0
Informational Messages: 172Total diagnostics: 172

Error responses:

Aborts: 0
Exits: 0
Breaks: 0