A Conformance Comparison between ANDF and GCC for X/Open Verification of an OSF/1 MK SS Platform

Thomas J. Watt Jr.

March 1993

Open Software Foundation
Research Institute
1 Cambridge Center
Cambridge, MA 02142

Abstract

The VSX Verification Test Suite for XPG3 was used to conduct a contolled experiment to gauge the XPG3 conformance of the ANDF compiler. An OSF/1 NORMA[1] Micro-Kernel (MK) Single Server (SS) Release 4.1 platform on an i486 architecture was used to establish the baseline environment. The native gcc compiler was used as a conformance baseline against which to compare the maturity of the ANDF compiler.

The ANDF compiler achieved a 94.2% rating of total XPG3 conformance. We are currently analyzing failed, uninitiated, and unresolved tests to understand, report and capture actual problems that may exist with the ANDF technology. Several error reports have been filed from this effort. Some tests test against a reference implementation or depend on ambiguous parts of the XPG3 standard; others are coded incorrectly against their intended test objective — all cases of which are thus invalid tests against the XPG3 standard and cause for request for waiver. This experiment has served to underscore the adage that testing is insufficient to reveal the presence of errors in program code.

This work was sponsored by a cooperative research agreement between the Open Software Foundation (OSF) and Unix System Laboratories (USL).

1. NORMA stands for No Remote Memory Access.

1. Introduction

This paper reports on our experience in the OSF Research Institute with using the ANDF compiler technology to build and execute the X/Open Verification Suite, VSX.

Essentially, we are pursing the goal of increasing the robustness of the ANDF compiler technology toward an industrial strength level of quality such that it delivers product quality for applications across a wide variety of heterogenous platforms. Toward that end, we have installed a version of VSX using the ANDF compiler technology, and submitted it for execution on one of our project's platforms. VSX is one of a number of test suites in our robustness project plan.

In subsequent sections we explain what VSX tests; summarize and compare the ANDF test results in table form with gcc for an OSF/1 MK SS R4.1 platform; review the current status of the failure analysis which gives a preliminary explanation of the delta between the conformance of gcc versus ANDF, and render tentative conclusions.

2. Software Information.

Software Category

VSX is the X/Open Verification Suite used for testing against conformance with the X/Open standard.

VSX from X/Open Ltd. tests all of the interfaces and definitions documented in the XPG.3 volume 2, 1989. VSX includes tests for ANSI C, POSIX.1 and XPG.3 interfaces and definitions which covers all header file, macro support and function calls in each standard. [Note: XPG.4 is the version used for OSF/1.2].

Version and Release Level

VSX Version 3.205 was used. It verifies conformance against the XPG3 standard.

ANDF Technology Version Release

Release TDF-93-01-27 of the ANDF Technology was used. It is based on the TDF Specification Issue 2.0 Revision 1 dated December 1992.

GCC Compiler Version and Release

Gcc 1.37.1 OSF Release 1.2.4.2 (OSF internal release) was the native platform compiler.

OS Platform Environment

OSF/1 NORMA Micro-Kernel 13.16 Single Server Release 4.1 was used as the native platform for this experiment.

VSX Authors and source

The authors of VSX are Unisoft Ltd, Hayne St, London EC1.

VSX 3.205 tests represent 228,675 lines of C source code which is copyrighted (C) 1989 X/Open Company Limited.

3. Summary of Test Results and Native Compiler Comparison.

The following tables report summary results for the ANDF and gcc compilers. Each column of data reports a particular category of result. The meaning of these categories are given below in order to interpret the results.

Succeeded means that the test executed correctly and to completion without any kind of problem. The implementation conforms with the definitions in the XPG3 standard.

Failed tests imply that some condition necessary for test success was not satisfied for some reason. In order to understand the reason for failure, it is necessary to understand the test strategy and identify the conditions which led to the failure.

Warnings happen when the system behaves in a way which is different from the functionality described explicitly in the XPG. Whenever a warning is given, the functionality is acceptable, but later issues of VSX may change the requirements in this area.

FIP is information which cannot be easily checked by the system and is given for you to validate manually. For example, the result of the uname test is non-negative and needs visual verification.

Unresolved means that the test started but did not reach the point where the test was able to report a result. **Uninitiated** means that the particular test in question did not start to execute. When a test is reported as uninitiated or unresolved, the reason why the test was not performed may be because of incorrect test parameters, preceding failures or external events**.**

Unsupported means that an optional feature within XPG3 is not available or supported in the implementation under test.

The **Untested** category is reported because there is no test written to check a particular feature. For example, it is not possible to check that session IDs are inherited accross a fork() when job control is not available.

The **NotInUse** category pertains to a number of tests which are reserved for future use. These are not failures and require no further work.

Table 1: VSX3.205 on OSF/1 MK R4.1 (i486) with ANDF 93-01-27:

	Total tests	Succeeded	Failed	Warnings	FIP	Unresolved	Uninitiated	Unsupported	Untested	Not in Use
ANSI.hdr	336	314	12		2			8		
ANSI.os F	1521	1479	16	1		2		3		20
ANSI.os M	56	56								
POSIX.hdr	332	312	13					7		
POSIX.os F	1388	1264	12	4	3	11	1	54	5	34
POSIX.os M	2	2								
XOPEN.cmd	16						16			
XOPEN.hdr	60	44	15		4			1		
XOPEN.os	219	162	10	1			46			
lang.C	1292	1285	4	3						
TOTAL	5222	4918	82	9	5	13	63	72	5	54

Table 2: VSX3.205 on OSF/1 MK R4.1 (i486) with gcc 1.37.1:

	Total tests	Succeeded	Failed	Warnings	FIP	Unresolved	Uninitiated	Unsupported	Untested	Not in Use
ANSI.hdr	336	185			6			145		0
ANSI.os F	1521	1472	9	1		16		3		20
ANSI.os M	56	56								
POSIX.hdr	332	224	2		5			101		
POSIX.os F	1388	1266	11	4	3	29		36	5	34
POSIX.os M	2	2								
XOPEN.cmd	16	16								
XOPEN.hdr	60	42			1			17		
XOPEN.os	219	216	2	1						
lang.C	1292	1279		3	10					
TOTAL	5222	4758	24	9	25	45	0	302	5	54

4. A Side by Side Comparison of ANDF vs gcc.

The raw data in the previous summary tables can best be viewed in a side by side presentation given below. The data for each cell is [ANDF gcc] with blank spaces substituted for “0” data :

Table 3: A Side by Side Comparison of ANDF *vs*. gcc

	Total	Succeeded		Failed		Uninitiated		Unsupported	
		ANDF	gcc	ANDF	gcc	ANDF	gcc	ANDF	gcc
Header Tests									
ANSI.hdr	336	314	185	12				8	145
POSIX.hdr	332	312	224	13	2			7	101
XOPEN.hdr	60	44	42	15				1	17
Subtotal	728	670	451	40	2			16	263
Function Tests									
ANSI.os	1521	1479	1472	16	9			3	3
POSIX.os	1388	1264	1266	12	11	1		54	36
XOPEN.os	219	162	216	10	2	46			
Subtotal	3128	2905	2954	38	22	47		57	39
Macro Tests									
ANSI.os M	56	56	56						
POSIX.os M	2	2	2						
Subtotal	58	58	58						
Other Tests									
XOPEN.cmd	16		16			16			
lang.C	1292	1285	1279	4					

	FIP		Unresolved	
Header Tests	ANDF	gcc	ANDF	gcc
ANSI.hdr	2	6		
POSIX.hdr		5		
XOPEN.hdr	4	1		
Function Tests				
ANSI.os			2	16
POSIX.os	3	3	11	29

5. Preliminary Failure Analysis.

In this section, we will address the reasons for failure and uninitiated results of the ANDF compiler in order to understand the delta between it and gcc. For this report, we are not interested in tests which fail for both gcc and the ANDF compiler, and point this out where relevant by use of an asterisk (*****).

5.1 ANSI header test failures occur for one of three reasons:

1. Seven tests contain source code which declares a local old-style pointer to a function returning an int, e.g. `int (*func)();`, and attempts to assign the address of a prototyped ANSI function of the same return type to the old-style pointer to a function. The ANDF compiler correctly rejects this code because it does not conform to ANSI C. If the local pointer to a function were properly prototyped for ANSI C, the ANDF compiler would succeed for these tests.

2. Four tests are affected by a deficiency in the current release of the ANDF compiler related to name space pollution between macros and ANDF tokens. This problem occurs when a #ifdef *ANDF-token* is used, e.g #ifdef bsearch. The cause of this problem is improper detection of an *ANDF-token* as a macro where the token is declared as a function abstraction and no macro definition exists for the function. This deficiency caused another problem when #undef errno was used in the VSX source code. In this case, an “Undeclared” message was generated against errno which is an lvalued *ANDF-token*.

3. One test fails due to a generic CISC installer deficiency related to testing the values of mathematical constants in XPG3 (a floating point accuracy problem).

5.2 POSIX header failures occur for one of three reasons:

1. Five are for the same address of function assignment as cited above in 5.1.1.

2. Five are due to token variety mismatches between ANDF header file constructs and the source code. The mismatch occurs only because the source code is not properly casting an address of an expression which matches the pointer type of the lvalue assignment target. If the right hand side of the expression were properly cast, the ANDF compiler would properly compile and pass these tests.

3. One is due to the same macro name space pollution cited above in 5.1.2.

4. ***Two tests also fail with gcc[2].

5.3 XOPEN header failures occur for one of three reasons:

1. Seven are related to lack of support for the “compile” token construct in the ANDF header file xpg3/regexp.h by the ANDF compiler (tdfc).

2. Six are related to the macro name space pollution cited above in 5.1.2.

3. Two are related to the same address of function assignment problem cited above in 5.1.1.

5.4 ANSI Function failures occur for one reason:

1. Fourteen are related to the floating point CISC installer problem cited above in 5.1.3.

2. ***Two also fail with gcc.

5.5 *Eleven POSIX Function failures also fail with gcc.

One failure has been identified for retesting, and is undergoing further analysis.

5.6 All **XOPEN Function failures** (10) are related to the floating point CICS installer problem cited above in 5.1.3.

5.7 Four **lang.C failures** occured because they all violate some aspect of ANSI C**.**

5.8 Sixteen **XOPEN.cmd** tests resulted in an **uninitiated** status due to type incompatibilities between the source code and the ANDF header file declarations. The ANDF compiler correctly rejects them as invalid ANSI C.

5.9 Thirty-one **XOPEN Function** tests resulted in an **uninitiated** status due to the same reason cited in 5.3.1 above.

2. All tests also failed by gcc are the same tests in all instances cited in this report.

5.10 Fifteen tests resulted in an **uninitiated** status due to the presence of some amount of ambiguity with type definitions in the XPG standard regarding nl_catd. [Note: nl_item is similarly affected by the lack of a precise definition]

VSX does not claim to be ANSI conformant itself, i.e. implemented in ANSI C, but does claim to test the ANSI interfaces — there is a difference. It does this with source code that violates ANSI C — a condition that causes several of the XOPEN Function tests to result in an uninitiated status manifested during run-time test compilation.

All of the tests are locale related and contain source code similar to the following code fragment:

```
#include <nl_types.h>
nl_catd catval;
if((catval = catopen (...)) == (nl_catd)-1) {...}
```

The ANDF compiler correctly rejects this code fragment for two non-ANSI behaviorisms:

(1) the number -1 is being cast to something to which it may not necessarily convert, and

(2) the “==” operator is being used between two operands which are not necessarily comparable, e.g. where nl_catd may be defined as a structure on some platforms, ANSI C does not allow structure comparisons (due to padding holes, etc.).

The source of the problem appears to be a shortcoming in XPG where nl_catd is defined as an arbitrary general type, and requires `(nl_catd)-1` as the error return type for catopen. In principle, nl_catd could be defined on a platform as a struct or union, and in fact, several of our platforms define it as `int` while others define it as `typedef struct *.`

Under ANSI C conversion rules, integers can be converted to arbitrary pointer types, which would allow `(nl_catd)-1`.

Currently, XPG implies that nl_catd should be defined to be any type to which a -1 can be converted, including all arithmetic and pointer types.

We believe the correct solution is to replace `(nl_catd)-1` in the XPG specification and in the VSX source code by a constant, `nl_catd_error`.

5.11 The **Unsupported** ANSI and POSIX Function tests are under investigation and are not reported here.

6. Preliminary Conclusions.

Despite having 3.4 times as many failures as the gcc compiler, the ANDF compiler recorded more successful tests than gcc. The ANDF compiler provided 94.2% of complete XPG3 conformance for this version of VSX. As a comparison, this version of the gcc compiler provided 91.1 % under the same environmental conditions in a controlled experiment.

It is clear that the ANDF compiler provides more comprehensive API header file support for ANSI C, POSIX.1 andf XPG3 than this version of gcc: i.e. ANDF provides 92% of the total support required for complete conformance, while gcc provides only 62%.

The ANDF compiler provides the same macro support for ANSI C and POSIX as gcc, where both provide 100% conformance.

Since VSX is not implemented in ANSI C, we conjecture that an ANSI C implementation of the VSX suite would allow the ANDF compiler to achieve a higher conformance to XPG3 than the the data reported here indicates for this implementation of VSX..

We noticed that some tests were not accurately coded to test against their intended test objective. For example, there are a number of tests which attempt to test against a macro version of a function, and increment local counter arguments to determine if the arguments are evaluated only once. In particular, where the setjmp and longjmp tests are concerned, these tests should declare the local counters as volatile to determine their test objective accurately.

There remains a question about the validity of the VSX test suite given the above findings in the area of tests which are either incorrectly encoded or encoded against a reference implementation instead of the XPG3 standard. One can understand the reason for this in the following context: If any area of the

standard is ambiguous, this leaves little room but to test against a reference implementation. Such is not the intention of a verification suite, and needs to be addressed in the XPG4 standard as well — nl_catd and nl_item are not precisely defined!