Commonly Asked Questions (andAnswers) about ANDF

Dr. R. R. Rowlingson
Dr. N. E. Peeling
DRA Malvern UK

British Crown Copyright © 1992

Index of Questions

Questions

1 Background to ANDF, TDF and DRA

1.1 What is ANDF?

1.2 What is TDF?

1.3 Who are DRA?

1.4 Which organizations support ANDF?

1.5 Why are DRA developing TDF?

1.6 What is the history of TDF?

1.7 How much is being invested in TDF?

1.8 Is TDF an open specification?

1.9 Who controls TDF?

1.10 Has ANDF been oversold?

2 Commercial Aspects

2.1 How can I get hold of TDF?

2.2 What software is available?

2.3 What real-life experience is there in using ANDF?

3 ANDF and ISVs

3.1 Does ANDF protect proprietary information as well as a binary does?

3.2 Does ANDF make it much easier to pirate software?

3.3 How will ANDF affect an ISV's ability to do differential pricing?

3.4 How does an ISV ensure quality if a customer can run a ANDF program on a machine on which the ISV has not tested it?

6.8 Will operating systems be compiled using ANDF?

6.9 Is TDF suitable for parallel computers?

Questions (and Answers)

1. Background to ANDF, TDF and DRA

1.1 What is ANDF?

ANDF stands for Architecture Neutral Distribution Format. The concept of an ANDF was pioneered by the Open Software Foundation (OSF). The OSF issued a Request For Technology (RFT) in May 1989 to solicit proposals for technologies that might form the basis of their ANDF; OSF received 23 such proposals.

The objective of ANDF is to encourage the development of a large body of ‘shrink-wrapped’ applications capable of running on any Open Systems platform which implements the necessary libraries (APIs).

1.2 What is TDF?

TDF is an ANDF technology, designed and implemented at DRA Malvern. In June 1991 DRA’s TDF technology was selected to be the basis of the OSF’s ANDF.

TDF represents an evolution of multi-language, multi-target compiler intermediate languages. If there are n programming languages and m machines then instead of needing the product of n and m compilers one only needs the sum ($n+m$) “half compilers”. OSF refer to the n front-ends which compile the programming languages into ANDF as *producers*, and the m back-ends, which translate ANDF into the machine code of the different machines, as *installers*.

The TDF technology provides software developers with an environment to create portable applications. It provides mechanisms for defining the portability interface which an application will use so that the application can run unchanged on any computer that implements that interface. This mechanism can also be used to define architecture neutral header files for common APIs such as POSIX, XPG3, SVID and the AES

The document “ANDF Features and Benefits” gives a more detailed answer to this question.

1.3 Who are DRA?

The UK Defence Research Agency (DRA) was established as an Executive Agency of the UK Ministry of Defence on 1st April 1991. It is wholly owned by the UK government but is a distinct organization headed by a Chief Executive who derives his authority from, and is directly accountable to, the Secretary of State for Defence.

The aim of the DRA is to provide the expert scientific and technical services required of it, primarily by the Ministry of Defence and Other Government Departments in a way which is cost-effective and impartial; in support of this, the DRA is also encouraged to provide services to non-government customers. The DRA is expected to continue to be the MoD’s principal source of scientific advice, technical support and research.

DRA’s “turnover” is approximately a billion dollars per annum.

DRA Malvern, formerly known as RSRE, is the Agency’s main centre for work on information technology and electronic systems and devices and on their defence applications. It is staffed by approximately 800 high calibre scientists and engineers from many disciplines working at the frontiers of scientific knowledge.

1.4 Which organizations support ANDF?

The OSF has an advanced technology program in their Research Institute covering ANDF. UNIX System Labs (USL) have signed a letter of understanding with DRA to license the TDF technology. The European Commission is supporting the Esprit GLUE project. Many other organizations from market leaders to standards bodies and industry groupings have shown interest and have offered support to the ANDF project.

1.5 Why are DRA developing TDF?

The Ministry of Defence is a large end-user of computer systems. Through the widespread uptake of ANDF DRA aims to reduce the costs for software development and software and hardware procurement for the MoD. Related goals include the potential for early availability of compilers (e.g. for Ada) on new architectures, separating software and hardware purchasing, easier mid-life hardware updates, and improving software longevity and scalability.

By providing TDF as the de-facto standard ANDF technology DRA ensures that MoD has access to a civil standard which can also handle defence requirements such as support for Ada.

1.6 What is the history of TDF?

TDF is derived from the Ten15 project at DRA Malvern. The Ten15 project was a 50 staff year project which involved the definition and implementation of a high-level abstract machine. Like TDF it was an architecture neutral representation of programming languages but it was strongly typed and also had the capability to describe operating system features, such as the network and filestore, explicitly. TDF was conceived as a portable code-generator for Ten15 and was purpose-built to the requirements for creating and distributing shrink-wrapped software.

1.7 How much is being invested in TDF?

DRA has a team of 15 people plus 7 contractors currently working on TDF. OSF and USL also have technical staff working to develop the technology. DRA are one of seven participants in a 3 year, $15 million, ESPRIT project known as OMI/GLUE which also includes work on developing producers (compilers to ANDF) for Ada, Fortran, C++ and a functional language as well as work on validation, TDF tools and extending TDF for parallelism.

OSF and USL are working with ISVs to demonstrate the viability of the ANDF approach to shrink wrapping existing applications.

1.8 Is TDF an open specification?

The specification of TDF is freely available from DRA. All the Intellectual Property contained in the TDF specification is in the Public Domain. DRA takes no royalties for any use of the specification. The TDF encoding as a bitstream is also in the public domain as is software for encoding and decoding TDF. The other TDF tools (which are proprietary) work from a central database that specifies TDF - this database is also in the public domain.

1.9 Who controls TDF?

TDF is a technology owned and controlled by the DRA. It forms the basis of the OSF's ANDF technology. USL and OSF have publicly announced that they will support the same specification. The latest versions of TDF and ANDF are identical.

DRA intends to submit TDF's definition to an internationally recognized standards process once TDF has proven its viability in the market place. At such time DRA will cede control of the TDF specification to the appropriate standards body/bodies.

1.10 Has ANDF been oversold?

Many of the properties of ANDF (the concept) are now inextricably linked to the properties of TDF (an ANDF technology). Until OSF had selected TDF it was difficult to say exactly what were the expected properties of ANDF and this did lead to some confusion. It is now clear that ANDF as implemented by TDF is much more than just a distribution format - it is also an aid in recasting applications in shrink-wrapped form, and it is also an architecture-neutral medium for creating header files for APIs.

Some complex issues may have been oversimplified, such as the issue of reverse-engineering ANDF. The latest documentation from DRA and OSF avoids such oversimplifications.

2. Commercial Aspects

2.1 How can I get hold of TDF?

Software availability is currently through the OSF snapshot program, for further details contact Andy Johnson at the OSF Research Institute at 1 Cambridge Center, Cambridge MA, USA. email: andyj@osf.org

DRA are also looking into possible mechanisms for distributing a developers kit.

2.2 What TDF software is available?

DRA are maintaining installers for MIPS, 680X0, 80X86 and SPARC. OSF has had an RS 6000 installer developed under contract. A TDF producer from ANSI C and other dialects has also been written. A TDF linker program, TDF pretty-printer, TDF installation manager and a compiler interface to UNIX, known as tcc, are also implemented. There are architecture neutral C header files for POSIX, XPG3 and SVID3. This software has been sufficient to allow the shrink-wrapping of “real-life” applications such as Informix’s Wingz product.

2.3 What real-life experience is there in using ANDF?

DRA has successfully transferred the installer technology out of the lab - the Sun SPARC installer was written under contract to DRA and achieved better performance targets than the installers written in-house. OSF have ported and demonstrated Informix’s’ Wingz (TM) on ANDF. USL and OSF are collaborating on an ISV program to encourage ports of major commercial applications to ANDF on SVR4.2, starting with Oracle, and DRA supports and assists in this important area. In addition DRA and OSF are working to shrink wrap a number of public domain applications such as the Postgres database.

The TDF software has been successfully tested on a large body of software, which includes:

- bootstrapping the TDF software
- the Plum-Hall test suite
- the Perennial test suite
- SPEC C Benchmarks
- Motif Window Manager
- Gnu Emacs
- Wingz
- Perl
- gcc 2.2.2
- TeX and Metafont
- Xfig
- X11R5
- Modula 3 to C compiler
- Postgres
- GhostScript
- Bison

3. ANDF and ISVs

3.1 Does ANDF protect proprietary information as well as binary does?

There are two separate issues that should be distinguished when discussing the reverse engineering of ANDF:

(1) how easy is the process of reverse engineering?

(2) how understandable an output can be produced from a reverse engineering process or tool?

ANDF is such an attractive target to reverse engineer that we would expect reverse engineering tools to become available, regardless of how difficult it is to produce such tools. This makes point (1) above irrelevant (in fact TDF is somewhat easier to reverse engineer than a binary). The important fact is that the quality of information recoverable from TDF is very similar to that recoverable from binary - rearranged source text without identifiers (what others have sometimes referred to as “uglified” or “shrouded” source). It would certainly be possible to scramble the ANDF to make it even uglier. If an ISV does not regard this as adequate protection we would suggest that the shrink-wrapped ANDF be converted to binaries for final distribution.

3.2 Does ANDF make it much easier to pirate software?

A shrink-wrapped application in ANDF can obviously be pirated to many platforms. We would expect that many ISVs will use licensing control software to protect against such theft. This is an issue that the industry is already grappling with - ANDF just makes it more urgent that workable solutions be widely implemented.

3.3 How will ANDF affect an ISV’s ability to do differential pricing?

This is already an issue that the industry faces. In machine ranges which are binary compatible it is already common for licenses fees to vary for the same binary distribution (license fees being tied to processor power, numbers of concurrent users etc.).

3.4 How does an ISV ensure quality if a customer can run an ANDF program on a machine on which the ISV has not tested it?

It is likely that as applications start to be shipped in ANDF that they will have been tested by the ISV on many targets. The machines that the application has been tested on can be listed on the packaging and the product need not be warranted for use on any others. As new architectures come onto the market, the system vendor will wish to provide an installer, so that ANDF applications are available to customers. The onus will then be on the installer writer to ensure that it is a robust and validated product. The situation is analogous to the PC world where software developers cannot test on every PC clone that exists but are confident, through testing on the most important in the market, that the product is likely to work on most clones.

3.5 Will ANDF reduce testing costs?

Even if specific platform testing by ISVs continues the ISVs' testing costs could still be significantly reduced. Testing costs are related to the number of problems encountered during testing and are often due to compiler bugs or different interpretations of the high-level programming language being compiled. ANDF can reduce these costs through the use of a single compiler front-end - the producer - for a particular language. The installer, which translates ANDF to code, has much simpler syntax and semantics to contend with than for C, so the validation of installer semantics should be easier and more reliable than relying on the consistency and robustness of different C compilers. Also a large body of shrink-wrapped applications will make it easy to test new installers.

3.6 If a shrink-wrapped application fails who does the customer blame?

This question was addressed by question 3.4. The issue with ANDF is exactly analogous to that faced by suppliers of PC software, and that has not stopped ISVs from marketing PC software - indeed the possibility of shipping shrink-wrapped applications is often quoted as the reason for the strength of the PC market.

3.7 How long does installation take?

Installation time, for an application, is generally somewhat less than the compile time for a native compiler. On gcc, from the SPEC tests, installation time varies between 32% and 83% of native compile time on

DRA's installers for 386, HP-PA, MIPS, SPARC and VAX architectures. An example from the Wingz application is that a complete build from source on a DecStation 3100 using the native compiler took about one hour and thirty minutes. An ANDF install on the same platform took approximately twenty one minutes.

For many users / applications the install time will not be a problem. If it is there are a number of possible solutions:

- installation may be run as a background task (on an operating system that permits multi-tasking).
- installation may be done overnight.
- binary distribution may be available (possibly at extra cost from a software distributor). CD-ROMs may provide a convenient mechanism for marketing binary versions of applications, as might ISDNs.
- if a user is part of a network, there may be a node that has responsibility for turning ANDF applications into binaries for other users on the network.

3.8 How large is the distributed TDF compared to binary?

This depends on the nature of the machine. On CISC machines the distributed TDF is around twice the size of the binary. On RISC machines the distributed TDF (same size as on CISC machines!) is around 1.4 times the size of the corresponding binary produced by the native compiler. This figure assumes that architecture neutral headers are being used - if existing header files are used the size of distributed ANDF can increase by up to 50% for some applications.

3.9 What are the performance costs of using TDF?

DRA has demonstrated that the run-time performance of applications compiled from C through TDF can be made comparable and in some cases faster than for applications compiled through native compilers. The information in the source language required to do code optimization is preserved in TDF allowing the fullest range of code optimization algorithms to be used. The performance of existing implementations is given in the document ‘TDF Facts and Figures’. All installer implementations are currently within 5% of the performance of native compilers as measured on the SPEC tests.

ANDF has significant economic advantages which may lead to performance improvements - the cost of tuning an installer potentially benefits all the programming languages that compile to ANDF; also the effort in tuning an installer for one machine may benefit other installers (many optimizations can be expressed portably as TDF-to-TDF transformations).

3.10 How long does it take to shrink-wrap an existing application?

There is as yet limited experience as to how much effort a software developer should expect to have to expend in such a recasting. OSF’s experience shrink wrapping Informix’s Wingz (TM) product and DRA’s experience shrink wrapping public domain applications suggests that the creation of a shrink wrapped version of an application takes roughly the same amount of time as doing a port to a new platform. Once an application has been shrink-wrapped, its ability to run on a given platform is constrained only by the availability on that platform of implementations of all the portability interfaces which the application requires.

4. ANDF and System Vendors

4.1 How much effort does it take to implement a new installer?

The TDF technology is designed to allow new high-performance installers to be implemented economically. To achieve this as many optimizations as possible - both universally applicable optimizations and machine specific ones - are carried out as TDF-to-TDF transformations. As a measure of the effort to produce a new installer: the existing SPARC installer was implemented by a DRA contractor to a level where it equals the performance of the best C compilers on a SUN SparcStation in one staff-year. The implementation was performed by an experienced compiler writer who was familiar with SPARC but who had not worked with TDF before.

The implementation was produced at a site remote from DRA with less than a week's consultancy from DRA.

4.2 How has DRA matched the performance of compilers which have taken many tens of staff-years to develop (for example the MIPS compiler on the DecStation and the native compiler on the 80x86 running SVR4.2) with installers that have only taken one or two staff-years to develop?

We cannot know for sure but the following factors may be relevant:

- the approach adopted by DRA is a conventional engineering approach (it is not an automated code generator technology) which reuses large amounts of code between installers. This means that:
    - as much of 70% of a new installer is re-used code
    - the installer being hand-crafted is easy to tune
    - the installer implementation is small enough be done by one person (a very efficient method of working).
- the staff used to implement the installers were very experienced and shared experiences within a close-knit community.
- DRA implementors had access to existing high-performance compilers which showed a large number of the optimizations which had to be implemented.

4.3 Do you have to use DRA technology to create a new TDF installer?

No. As far as existing code-generator technologies are concerned TDF is just another language - hence existing techniques can be used to implement a TDF installer. The only reason to use DRA's technology is economic - is it cheaper to use our technology than some other?

5. ANDF and End Users

5.1 How much will it cost?

The pricing of TDF is a matter for the commercial organizations which license the technology from DRA.

5.2 Does ANDF help me with the problem of legacy systems?

ANDF is not a "magic bullet". It does not mandate portability. This means that ANDF is not a magical solution to the problems of legacy software. You cannot just run a non-portable application through the ANDF technology and expect to produce a portable ANDF version. The advantages of using ANDF are obtained by re-casting the software into shrink-wrapped form.

Much legacy software is written in legacy programming languages (languages that are no longer popular and hence difficult to obtain compilers for). ANDF provides a means of implementing a portable compiler for a legacy language which will generate code to run on all the computers that have ANDF installers (within the limits of the portability of the application being compiled). For some legacy languages it may be an economic proposition to produce an ANDF producer. This may lead to increased longevity for some legacy languages.

6. Technical questions

6.1 Is ANDF tied to UNIX?

ANDF itself is independent of operating system and will help create and distribute shrink-wrapped software for any well defined API. OSF's motivation in creating the ANDF concept was to turn the UNIX software market which is currently heterogeneous with respect to processor architecture into a homogeneous ANDF market. USL's interest is to create a homogeneous software market for UNIX SVR4.2. As a result the majority of DRA's implementations of TDF software are for UNIX based machines. The TDF software has also been ported to MSDOS (with DOS extender) which supports the ANSI C library, and a port to OpenVMS has also been started.

6.2 How does ANDF relate to APIs?

APIs are extremely relevant to the ANDF scenario for application development, porting and distribution. To distribute an application to many machines using ANDF requires the use of suitable portability guidelines. ANDF has features designed to allow the representation of the interfaces required for portability. These might be established APIs such as Posix, XPG3, SVID3, the AES etc. or they might be interfaces created by an ISV.

TDF contains a notion called "tokenisation". This is the ability of TDF to represent program as a TDF tree any part of which can be represented by a symbol called a "token". The token can represent any part of the program detail (e.g. a type, a part of a type, a procedure, a macro, an expression or part of an expression, etc.) which is architecture specific. After distribution the token can have an appropriate architecture specific piece of TDF tree supplied as its definition. This provides a natural mechanism whereby the architecture neutral information inherent in the API is abstracted and forms the portability interface to the application.

Once an application has been shrink wrapped in ANDF it can run unchanged on any computer that correctly implements the portability interface that the application is designed to use. This means that API conformance is of critical importance. Shrink wrapped applications will provide practical tests of API conformance. The applications that have been shrink-wrapped have already shown up a number of subtle errors in some existing implementations of APIs.

6.3 How does ANDF relate to ABIs?

ABIs define a machine level target for compilers to ensure software compatibility for a single processor family. ANDF defines a machine-independent target for compilers to ensure software portability across all systems that implement ANDF and have the interfaces (e.g. GUI) assumed by the program. An ABI is a binary compatibility interface, whereas ANDF provides compatibility at the functional API level - this means that an ABI specification must remain unchanged if existing ABI applications are still to work, whereas in ANDF if the API functionality is maintained then ANDF applications will still install properly.

ANDF installers can use ABIs so that only one installer is needed for all processors that implement that ABI - hence efforts to develop ABIs will continue even after the use of ANDF becomes widespread. In the future ANDF could remove the need for ISVs to be aware of the details different ABIs. This would allow much easier introduction of new processor features, such as superscaling, because the need to preserve upwards compatibility in the ABI would no longer exist.

6.4 How does ANDF relate to distribution of multiple binaries on CD-ROM?

ANDF is an advanced porting tool as well as a distribution format. The use of multiple binaries on a CD-ROM may reduce the packaging and distribution costs for ISVs but has no impact on portability or testing problems. For each new architecture supported, a new CD-ROM needs to be issued - an application distributed in ANDF can be installed on any machine with an installer.

We expect that ANDF will co-exist with binary distribution and that CD-ROMs may be used to distribute binaries of the most popular shrink-wrapped applications to avoid the overhead of longer installation times.

6.5 How does ANDF relate to using shrouded C for distribution?

Shrouded C is just a distribution mechanism - it has none of the other properties of ANDF to aid in the creation of shrink-wrapped software, or the description of portability interfaces.

Comparing Shrouded C and ANDF purely on the basis of distribution formats:

- Shrouded C gives only limited support to users of other programming languages (compilers to C from other languages are often far from optimal). TDF was designed as a multi-lingual ANDF so should be much better.
- Shrouded C requires a user site to have a full Software Developers Kit, whilst the ANDF installation environment is likely to be bundled at no extra charge.
- installer availability for Shrouded C is no problem
- protection of proprietary information is similar
- size of distributed applications is similar
- installation times favor ANDF
- run-time performance is similar
- tools to support the installation process tend to be primitive in conventional C compilers
- ANDF users get the benefit of using the same producer for all targets (reduced exposure to compiler incompatibilities and bugs)

6.6 Is ANDF a C oriented technology?

At the present, the only compiler front-end for TDF is for C (ANSI and other dialects). A great deal of effort has gone into ensuring that ANDF can be used as an intermediate language in “conventional” compilers for languages other than C. Implementing the C front-end for TDF exposed a number of issues that related to the production of architecture-neutral ANDF. There are parts of the semantic definition of C, such as the precise definition of some integer promotions, which depend on the target. In addition, ANSI C was extended with a new pragma (called “#pragma token”) which allows software developers access to ANDF’s features for describing portability interfaces. These issues have been satisfactorily resolved for C, but a similar exercise would have to undertaken for other languages if they are intended for use in producing shrink-wrapped applications. Work is underway to write producers from several important high-level languages.

See also the answer to question 6.7.

6.7 Will ANDF collapse as more languages and targets are considered (remember UNCOL)?

The problem with earlier attempts at a ‘universal’ compiler intermediate languages (IL) were they were defined at the level of the target machine architectures. If a new architecture needed to be supported that was radically different from the baseline for the IL, large penalties in performance were paid in effectively doing the binary to binary conversion.

TDF is not a pseudo-code which represents machine instructions for some abstract stack or register computer. Its definition is an abstraction of programming languages not computer architectures. It is a tree-structured intermediate language containing abstractions for common programming language concepts such a data structures, procedures, numbers, conditionals, loops, labels, jumps etc. The intention in TDF is to retain all the information in the programming language being compiled to TDF which can then be used by code optimizers in the installers. In this way it is equally applicable to any new architecture as to any existing ones. If you can efficiently compile C to an architecture then you can implement a TDF installer with equal efficiency.

TDF constructs have been carefully designed so as to be able to accommodate the particular variants found in different programming languages. However, TDF cannot guarantee coverage of new programming languages as it can for new architectures. New languages might contain novel features that are not efficiently implementable using existing features of TDF.

DRA have undertaken extensive research to try to ensure that TDF is an efficient target for the conventional compilation of many of today’s most common languages - C++, Cobol, Fortran, Ada, Lisp, Pascal etc. and several new producers are now being written. It is likely that the implementation of producers for programming languages other than C will expose new features that would enhance the efficiency of the run-time code for these new languages - it is likely that such extensions can be added in an upwards compatible manner, but this cannot be guaranteed.

At the current time the only compiler front-end for TDF is for C (ANSI and other dialects). Implementing the C front-end for TDF exposed a number of issues that related to the production of architecture-neutral ANDF. There are parts of the semantic definition of C, such as the precise definition of some integer promotions, which depend on the target. In addition, ANSI C was extended with the “#pragma token” facility to allow software developers access to TDF’s features for describing portability interfaces. These issues have been satisfactorily resolved for C, but a similar exercise would have to undertaken for other languages if they are intended for use in producing shrink-wrapped applications.

6.8 Will operating systems be compiled using ANDF?

TDF is a highly portable, highly optimized compiler technology and is therefore very well suited to compiling or bootstrapping operating systems. As TDF need not represent purely architecture neutral information, it is possible for the operating system vendor to use TDF as a ‘normal’ compiler and therefore avoid the need to shrink-wrap the operating system. It may take a while before operating system suppliers have enough confidence in the TDF technology to throw out their existing bootstrap compilers - until then TDF will exist alongside existing native compilers.

6.9 Is TDF suitable for parallel computers?

It depends what programming language is being used to program the parallel computer. If a conventional programming language is being used (such a C or Fortran) then TDF preserves enough information to allow a paralysing installer to generate as efficient code as that produced by a compiler direct from the language to the parallel computer. TDF does not contain any explicit parallel constructs so a parallel language (such as high-performance Fortran) would need to compile to TDF that has been extended with tokens that represent the parallel constructs in the parallel language. If a de-facto standard emerges in parallel programming languages then it should be possible to define a standard extension to TDF to represent a standard model of parallel processing.

There is an active research program at DRA, OSF and within Esprit that is studying this topic.

For further information please contact:

Dr. Nic Peeling
internet: peeling%hermes.mod.uk@relay.mod.uk
janet: peeling@uk.mod.hermes
fax: +44 684 894303