The NIST EXPRESS Toolkit – Obtaining and Installing

Don Libes

Abstract

The NIST EXPRESS toolkit is a software library for building EXPRESS-related tools. The EXPRESS Toolkit is based on Draft International Standard (DIS) 10303-11 (N151). The NIST Part 21 Exchange File Toolkit is a software library for building Part 21-related tools. The Part 21 Toolkit is based on DIS 10303-21. This paper describes how to obtain the toolkits and install them. No knowledge of EXPRESS or the EXPRESS Toolkit is presumed other than a familiarity with file transfer and similar installation procedures. A rudimentary grasp of the theory of typical computer language implementations is useful but not necessary.

Keywords: compiler, EXPRESS; implementation; National PDES Testbed; PDES; STEP

Context

The PDES (Product Data Exchange using STEP) activity is the United States' effort in support of the Standard for the Exchange of Product Model Data (STEP), an emerging international standard for the interchange of product data between various vendors' CAD/CAM systems and other manufacturing-related software [1]. A National PDES Testbed has been established at the National Institute of Standards and Technology to provide testing and validation facilities for the emerging standard. The Testbed is funded by the Computer-aided Acquisition and Logistic Support (CALS) program of the Office of the Secretary of Defense.

As part of the testing effort, NIST is charged with providing a software toolkit for manipulating STEP data. The NIST EXPRESS Toolkit and NIST Part 21 Exchange File Toolkit are a part of this. The toolkits are evolving, research-oriented software tools. This document is one of a set of reports (an overview of each appears in [2]) which describe various aspects of the Toolkit.

Introduction

The NIST EXPRESS Toolkit (called the "toolkit" or the "EXPRESS toolkit" from here on) [2] is a software library for building EXPRESS-related tools [3]. The NIST Part 21 Exchange File Toolkit (called the "P21 toolkit" from here on) [4] is a software library for building Part 21-related tools. The EXPRESS toolkit may be installed by itself, but the P21 toolkit requires the EXPRESS toolkit be present.

Obtaining the Toolkits

The toolkits are distributed in two ways: latest public release, and all releases. The latest public release is just what it sounds like: the latest version of the software that has been approved for public release. This is generally what most people want.

Alternatively, it is possible to get the entire set of releases. This is useful, not so much because of the earlier releases, but this includes experimental releases well before they are released to the public. At the same time, such experimental releases are often unstable, while the public releases are more thoroughly tested. Access to the entire set of releases is only available by explicit approval of the National PDES Testbed. For more information, contact:

FASD - National PDES TEstbed
National Institute of Technology and Standards
Bldg 220, Rm A-127
Gaithersburg, MD 20899

npt-info@cme.nist.gov
1-301-975-3508

Occasional references will be made to SOLIS [5][6]. SOLIS stands for "the STEP OnLine Information Service" and provides access to files in a number of ways. These are described below.

Documentation

A number of documents are (or soon will be) available that describe the toolkit. These can be received similarly to the way that the toolkit is received. The documents are stored in a number of files. The list of file names will be referred to in other parts of these document. The following file names and references (see References on page 13) correspond to the EXPRESS toolkit:

File Name	Reference
exptk-intro.ps.Z	[2]
exptk-requirements.ps.Z	[7]
exptk-design-and-impl.ps.Z	[8]
exptk-lessons-learned.ps.Z	[9]
exptk-using-apps.ps.Z	[10]
exptk-obtaining-installing.ps.Z	this document
exptk-programmer-ref.ps.Z	[11]
exptk-creating-apps.ps.Z	[12]
exptk-updating-apps.ps.Z	[13]

The following file names and references correspond to the P21 toolkit:

p21tk-update.ps.Z	[4]
p21tk-ref.ps	[14]
p21tk-design.ps	[15]

Internet ftp

If you have **`ftp`** and are connected to the Internet, you can retrieve the latest public release of the toolkits as described here. The following explanation assumes you are familiar with anonymous **`ftp`**. For more information, read [6].

Connect to **ftp.cme.nist.gov** [1] as **anonymous**:

```
ftp ftp.cme.nist.gov
Connected to ftp.cme.nist.gov
220 tribble FTP server (Version 4.167 Wed Aug 14 22:26:49 EDT
     1991) ready.
Name: anonymous
331 Guest login ok, send ident as password.
Password:              (type your email address)
230 Guest login ok, access restrictions apply.
```

Retrieve the EXPRESS toolkit:

```
ftp> cd /pub/step/npttools
200 PORT command successful.
ftp> get exptk.tar.Z
200 PORT command successful.
150 Opening data connection for exptk.tar.Z (binary mode) (122273
     bytes).
226 Transfer complete.
122273 bytes received in 1.26 seconds (8.7 Kbytes/s)
```

Retrieve the P21 toolkit:

```
ftp> get p21tk.tar.Z
200 PORT command successful.
150 Opening data connection for p21tk.tar.Z (binary mode) (73932
     bytes).
226 Transfer complete.
73932 bytes received in 0.83 seconds (8.7 Kbytes/s)
```

Retrieve the corresponding documentation:

```
ftp> cd /pub/step/nptdocs
ftp> get exptk-intro.ps.Z
ftp> get exptk-requirements.ps.Z
ftp> get...
```

For the complete list of documentation files, see Documentation on page 2. Once you have retrieved all the files, read Unpacking the Documentation on page 12.

Email

If you can email to the Internet, you can receive the latest public release of the toolkits in response to sending mail. The following explanation assumes you are familiar with electronic mail. For more information, read [6].

1. ftp.cme.nist.gov is a CNAME record which can points to a number of different hosts depending on load averages and other factors. If your system can not dereference ftp.cme.nist.gov (i.e. does not understand CNAME records), use the Internet address 129.6.32.54. This address is not guaranteed to always work, but is the best that can be advertised short of fixing your machine to understand CNAMEs.

To receive files send a mail message to **nptserver@cme.nist.gov**[2]. This will be read by software which understands certain requests. Each request should be on a line by itself. Do not include a signature or header.

To get the EXPRESS and P21 toolkits include the lines:

```
send step/npttools/exptk.tar.Z
send step/npttools/p21tk.tar.Z
```

To get each document, include a line naming the appropriate document. For all the document file names, see Documentation on page 2. For example:

```
send step/nptdocs/exptk-intro.tar.Z
send step/nptdocs/exptk-requirements.tar.Z
send step/nptdocs/exptk-design-and-impl.tar.Z
```

In response to your mail, you will receive a number of mail messages. Since the files are longer than many mailers permit, they are broken into a number of mail messages. For each group of mail messages that was originally a single file, strip off the headers added by the mailer and join the messages together. Call the result ***X*.uu** where ***X*** is the original file name according to the subject of each message. Run each resulting file through **uudecode**. **uudecode** is a popular program available from numerous public-access source-code repositories such as **uunet.uu.net**. On a UNIX system, **uudecode** would be run on the toolkit as follows:

```
uudecode exptk.tar.Z.uu
```

This will create the file **exptk.tar.Z** in the current directory. You can now delete **exptk.tar.Z.uu**. Repeat this procedure with any other files you receive. After all files have been run through **uudecode**, read Unpacking on page 11.

Kermit

If you have a modem and Kermit, you can receive the latest public release of the toolkit. Kermit is a popular communications and file transfer program. It is available from numerous public-access source-code repositories such as uunet.uu.net. The following explanation assumes you are familiar with modems and Kermit. For more information, read [6].

From your local Kermit client software, dial 1-301-948-9720. This will connect you to NISTnet, an in-house NIST network. Connections up to 9600 baud (V.32) are supported. Once connected, press return.

```
VCP-1000 V3.302
NISTnet Modem Pool

Please type HELP if you need assistance

Enter username>
```

Enter a name. Any name is acceptable. You must then connect to the computer running the NPT Kermit server. Log in to this computer as "**kermit**".

```
Enter username> libes
```

2. "NPT" stands for "National PDES Testbed".

```
nistnet_6>connect solis.cme.nist.gov
Local -010- Session 1 to SOLIS.CME.NIST.GOV established

login: kermit
Last login: Tue Jan 19 16:39:07 from modems.nist.gov
SunOS Release 4.1.2 (TRIBBLE) #2: Wed Sep 2 12:17:13 EDT 1992

                              WELCOME to the SOLIS
                                 Kermit Server!
```

The system may ask you some questions such as your name and organization. After answering them you will be rewarded with the prompt "**Solis-Kermit**". To retrieve the EXPRESS toolkit enter the following commands:

```
Solis-Kermit>cd step/npttools
/pub/solis/npttools
Solis-Kermit>set file type binary
Solis-Kermit>send exptk.tar.Z
```

Now escape back to your local Kermit and give the **receive** command. A similar sequence is followed in order to receive each document, except that the directory is different and the "**set file type binary**" command is not necessary:

```
Solis-Kermit>cd step/nptdocs
/pub/step/nptdocs
Solis-Kermit>send exptk-intro.ps.Z
```

The P21 toolkit and other documents can be received by issuing analogous **send** commands. For all the documentation file names, see Documentation on page 2.

When you have finished, type "**quit**" to the kermit server. This will return you to the "**nistnet**" prompt to which you should type "**logout**". This will terminate your phone connection.

```
Solis-Kermit>quit

Local -011- Session 1 disconnected
nistnet_6>logout
```

Once you have retrieved all the files, read Unpacking on page 11.

NFS

By agreement with the NPT adminstrator, NFS access is permitted to selected users inside and outside NIST. If you have NFS access to the Testbed, it is possible to retrieve the latest release or all releases. It is also possible to obtain precompiled code for selected platforms in the testbed. All file references in this section are assumed to be from **cme.nist.gov**.

~pdes and **~pdevel** are two file systems that contain PDES code. **~pdes** contains files that are available to the public. This is exactly what is available through SOLIS. Through the RCS archives, it is also possible to obtain earlier releases that were once available to the public but are no longer. **~pdevel** contains files that are used by the developers at NIST. ("pdevel" stands for

"PDES Development".) Files in **~pdevel** typically reflect the latest work available however they can be less stable and are often experimental, alpha, or beta releases. **~pdevel** also contains archives of earlier releases. These earlier releases do not change although releases that are no longer considered useful are occasionally destroyed for space reclamation.

These instructions assume that you will create your own directory in which to store copies of the original code. This will prevent collisions between you and others caused by changing files while other people are working.

Out of convention, we assume your personal directory in which you are working with the toolkits is **~/pdes**. This is created as:

```
mkdir ~/pdes
```

Obtaining Precompiled Code

The toolkit is composed of a number of files. Not all files are necessary for some uses. In order to run fedex, the schema analysis software tool, all that is necessary is **arch/bin/fedex** in **~pdes** or **~pdevel**. In order to run **p21**, the STEP Part 21 Exchange File analysis tool, only **arch/bin/p21** is necessary.

In order to write EXPRESS toolkit applications, the following file is necessary:

```
arch/lib/libexpress.a
```

In order to write P21 toolkit applications, the following file is also necessary:

```
arch/lib/libp21.a
```

If you have your own lib directory, say in **~/pdes**, you can link the individual files to it:

```
ln -s ~pdes/arch/lib/* ~/pdes/arch/lib
```

If you do not have your own lib directory, but still do not wish to use **~pdes** directory, you can link to the directory itself:

```
ln -s ~pdes/arch/lib ~/pdes/arch/lib
```

The rationale for the extra directory ("**arch**") in each path is described in Installing the EXPRESS Toolkit Library and Executables on page 11.

Obtaining Documentation

The documentation can be found in **docs/exptk** and **docs/p21tk** in either **~pdes** or **~pdevel**. The names are subject to frequent change and are therefore not documented here. It is highly recommended to retrieve the documents as described in Documentation on page 2

Obtaining Source

If you want to retrieve the source, you should instantiate your own version of it from the **RCS** archives. You may choose to do this either from **~pdes** or **~pdevel** depending on your needs. The following examples will demonstrate how to do this from **~pdes** although this could also be done from **~pdevel**.

To retrieve the source, create the appropriate subdirectories. Each subdirectory will contain a link to the corresponding **RCS** directory. Check out the **CheckOut** file, and then run **CheckOut** itself.

This checks out all the other files from the **RCS** archives. The following commands do this from **~pdes**. Similar commands would suffice to extract files from **~pdevel**.

First check out the source to the EXPRESS library:

```
% mkdir -p ~/pdes/src/express
% ln -s ~pdes/src/express/RCS ~/pdes/src/express
% co CheckOut
% CheckOut
```

After running each **CheckOut**, expect a page or so of output as each file composing the toolkit is checked out.

If you want to use the P21 toolkit, you can repeat this sequence of commands, with "**express**" replaced by "**p21**".

Check out a copy of **mkrules**.

```
% cd ~/pdes
% mkdir include
% cd include
% co ~pdes/include/mkrules
```

The following utilities must be obtained:

```
bin/bison.errors
bin/move-if-change
bin/uniquify_flex
bin/uniquify_lex
bin/uniquify_yacc
bin/yaccpar.sun
bin/yacctokens.sh
bin/bisontokens.sh
/depot/gnu/arch/bin/bison
/depot/gnu/arch/bin/flex
```

Most of these files live in **~pdes/bin** and it is normally sufficient to create a symbolic link between this and your own **bin** directory as:

```
ln -s ~pdes/bin ~/pdes/bin
```

If you already have a directory by that name, you may link the individual files:

```
ln -s ~/pdes/bin/* ~/pdes/bin
```

The GNU utilities in the list above are only necessary if you are using Bison and/or Flex. If is almost never necessary to copy the GNU utilities. Instead, the directory **/depot/gnu/arch/bin** should be in your path.

Building and Installing the Toolkits

This section describes how to convert the sources to executables. If you do not yet have the toolkit source, read Obtaining the Toolkits on page 2.

Customizing the Toolkits

It is possible and in some cases necessary to customize the toolkit to each new site. This is usually done by modifying various files before compilation.

Mkrules

mkrules contains definitions common to both toolkits as well as some of the applications. There are actually two different versions of mkrules, one for the EXPRESS toolkit, and one for the applications. The one for the toolkit is in the **src** directory itself, while the application version is in the include directory. If you are only compiling the toolkit and other programs in the toolkit directory, the following discussion applies only to that one **mkrules**, otherwise it applies to both.

If you examine **mkrules**, you will find ways to customize your installation. One customization is almost always necessary. Namely, you must tell **mkrules** the directory in which you are keeping all your the toolkit code.

In order to make this change, start by making **mkrules** writeable:

```
chmod +w mkrules
```

Change the definition of **PDES** to reflect the root of the directories where you have the toolkit sources, binaries and other utilities stored. Make does not understand the **~** notation, so you must provide the hardcoded path. For example, a person named "Fred" whose home directory is **~fred** would have to expand this to an absolute path name. For example:

```
PDES=/usr/fred/pdes/
```

The **mkrules** file contains many other basic definitions such as which C compiler to use, flags to pass to it, and the locations of most of the libraries and include directories. Such customizations can also be moved to an application-specific **makefile**, if the customization is not appropriate for all applications. From here on, if we say a change can be made by modifying a **makefile**, you can assume that this also means that it can be modified for all applications in **mkrules**.

Lex vs. Flex and Yacc vs. Bison

The toolkit can be built using either **lex** or **flex** and **yacc** or **bison** [16][17][18]. Either pairing works. For example, **lex** and **yacc** can be used and so can **lex** and **bison**. Similarly with **flex**.

To switch the scanner or parser tools, **mkrules** must be edited. A large block of commands compose the rules for **.y.c** and **.y.o** (i.e., how to transform **yacc** into C source and object code). One definition exists for **yacc** and one for **bison**. Comment out one block and uncomment out the other block. Similarly for **lex** and **flex**. These rules are called **.l.c** and **.l.o**. An additional rule in each set is provided for profiling.

If the parser definitions are changed, **expparse.y** (in the EXPRESS toolkit) and **p21yacc.y** (in the P21 toolkit) should be touched in order to force recompilation. If the scanner definitions are changed, things that depend on them should be recompiled. **expscan.l** and **lex_actions.c** (in the EXPRESS toolkit) and **p21lex.o** (in the P21 toolkit) should be recompiled.

It is possible to augment the normal **yacc/bison** parser so that it generates better error messages [19]. With this modification, syntax error messages will look like "*syntax error, found TYPE but expected ENTITY or RULE or FUNCTION*" rather than just "*syntax error near TYPE*". This requires modifications to the parser as well as the parser skeleton provided with the parser generator. By default, this is enabled by the **CFLAGS_I** definition via the flag **-DAXEL**.

In order to modify the SunOS 4.1.2 yacc (others may vary), the file **~pdes/bin/yacctokens.-sh** should be copied to **/usr/lib/yaccpar**. No explicit modification is required for Bison (1.03); others may vary. In this case, the parser-generated C code is corrected by bin/bison.errors.

Profiling

It is possible to compile the code for profiling – i.e., timing and counting subroutine calls. To enable profiling, add "**-pg**" to the **CFLAGS** definition in the **makefile**. Then touch all sources. (It is not necessary to touch **.l** and **.y** files) and recompile.

Diagnostics and Debugging

There are numerous ways to enable the toolkits to generate additional diagnostic information that is not ordinarily useful to end-users. Most require compile-time flags in order so that the non-debugging mode runs as quickly as possible.

Flex Debugging

The flag **-DFLEX_DEBUG** can be added to the **CFLAGS** line in the **makefile** to have **flex** print the characters seen and tokens created. **-d** will generate debugging information, and **-v** will generate summary statistics.

Malloc Debugging

The EXPRESS toolkit includes a special memory allocation library that performs better than the Standard C library (**malloc**). The toolkit memory allocator reduces the execution time of applications by about 10-15% so it is very desirable. However, it can make debugging more difficult since it uses less structural overhead. (Less trace of damages are left behind if an application crashes.)

The P21 toolkit uses the special memory allocator as well, however control of it is enabled only through the EXPRESS toolkit **makefile**. To disable the special memory allocator and use the Standard C library allocator, uncomment the definition of **MALLOC** in the **makefile**. To enable more extensive (and expensive) debugging with the Standard C allocator (on a Sun), uncomment **DEBUG_MALLOC** and **DEBUG_MALLOC_LIB**. Non-Sun systems may provide alternative techniques for debugging the Standard C allocator. See the man page for more information.

Hasher Debugging

The EXPRESS toolkit includes a hash library which performs better than the Standard C library (**hsearch**). Normally, the hasher collects no information other than what it needs to run. It is possible for the hasher to collect extra information by adding two definitions to the **CFLAGS** definition in the **makefile**. **-DHASH_DEBUG** causes debugging information to be printed out during hash table creation and deletion. **-DHASH_STATISTICS** causes the hasher to count the number of hash table accesses and collisions.

The P21 toolkit uses the EXPRESS toolkit hasher. There is no way to change the performance of the hasher through the P21 toolkit **makefile**.

Parser Debugging

The parser can generate debugging information. If enabled, the parser is capable of printing out the token stream and parser rules used during the parse. To allow debugging code to be included by the parser add the flag **`-DYYDEBUG`** to the **`CFLAGS`** definition. This code must in turn be enabled at run-time [10].

C Debugging/Optimization

Debugging can be enabled with the addition of **`-g`** to the **`CFLAGS`** definition in the **`makefile`**. In the makefile, the definition of **`OPTIMIZE`** contains optimization flags that are passed to the C compiler. By default this is commented out. Uncomment to enable optimization. As distributed, this includes several optimization flags that are specific to the GNU C compiler. These can be deleted with impunity.

Building the EXPRESS Toolkit Library and Executables

Typically, the EXPRESS library and executables are built and then installed in a public area. It is possible to build them without installing them. This is useful during testing. If you do not want to test the libraries, go directly to Installing the EXPRESS Toolkit Library and Executables on page 11.

The following instructions assume the sources exist in the directory hierarchy **`~/pdes`**.

libexpress.a

The EXPRESS library is called **`libexpress.a`** and is created using the following commands.

```
cd ~/pdes/src/express
make libexpress.a
```

You can now build applications with the EXPRESS toolkit.

fedex

The EXPRESS toolkit includes two applications. One is **`fedex`** [10]. **`fedex`** reads schemas and reports any syntactic and semantic errors found. To build **`fedex`**, execute the following commands:

```
cd ~/pdes/src/express
make fedex
```

symlink

Another application that is provided in the **`express`** directory is **`symlink`**. **`symlink`** creates symbolic links to a schema using all the schema names that appear inside the schema. This provides a means of avoiding cutting and pasting schema files together. This is described further in [10]. Create **`symlink`** as follows:

```
cd ~/pdes/src/express
make symlink
```

Installing the EXPRESS Toolkit Library and Executables

To allow other people to use the software, it should be copied to a public place. The simplest way to do this is as follows:

```
cd ~/pdes/src/express
make install
```

The libraries and executable software are specific to a particular platform (i.e., computer hardware and operating system version). Thus, this install command copies files to an architecture-specific library called "**arch/lib**". With architecture-specific symbolic links [20], it is possible to create the illusion that machines of any architecture share **arch/lib**.

Building the P21 Toolkit Library and Executables

The P21 Toolkit library and executables are built similarly to the way that the EXPRESS Toolkit and executables are built. The EXPRESS Toolkit library must be installed before proceeding further.

libp21.a

The P21 library is called **libp21**.a and is created as:

```
cd ~pdes/src/p21
make libp21.a
```

p21

p21 reports errors in Part 21 files and any associated schemas. It is analogous in function to **fedex**.To build it:

```
cd ~/pdes/src/p21
make p21
```

Installing the P21Toolkit Library and Executables

The P21 Toolkit software is installed similarly to how the EXPRESS Toolkit software is installed.

```
cd ~/pdes/src/p21
make install
```

Unpacking

Any files that you have received that end with "**.Z**" are compressed and must be uncompressed. uncompress is a popular compression utility available on numerous public-access source-code repositories such as uunet.uu.net.

To uncompress files, run "uncompress" with the file names as arguments. For example:

```
uncompress exptk.tar.Z exptk-intro.ps.Z
```

After running uncompress, the **.Z**-terminated files will be replaced with files without the **.Z**.

```
% ls
exptk.tar        exptk-intro.ps
```

Unpacking the Documentation

Files that end with "**.ps**" are PostScript files. These files can be directly sent to a PostScript printer for printing. For example, on most UNIX systems this is done as:

```
% lpr exptk-intro.ps
```

Unpacking the Source Code

Files that end with "**.tar**" are archives of other files. These files are unpacked using **tar**, a popular archive program available on numerous public-access source-code repositories such as **uunet.uu.net**. For example, on most UNIX systems **tar** extracts all the files from an archives as:

```
% tar -xvf exptk.tar
extracting README (1 block)
extracting makefile (2 blocks)
```

. . . and so on . . .

tar automatically creates a directory named "**src**" if it does not already exist. By convention, a directory such as "**~/pdes**" is a directory which contains subdirectories such as **src**, **bin**, etc. **src**, in turn, contains source for different applications or components. To follow this convention, **exptk.tar** is unpacked from the directory **~/pdes** as:

```
% cd ~/pdes
% tar -xvf exptk.tar
```

The p21 toolkit is unpacked from the new src directory in ~/pdes as:

```
% cd ~/pdes/src
% tar -xvf p21.tar
```

Once **tar** has completed, you can remove the original tar file, **exptk.tar**, and go on to compile the toolkit (Building and Installing the Toolkits on page 7).

Installing on DOS (IBM PC and clones)

The bulk of this paper assumes a UNIX environment. However, it is possible to use the NIST EXPRESS toolkit in the DOS environment.

Unfortunately, precise instructions vary from one compiler to another. For simplicity, pre-compiled DOS executables are available from the NIST server as **pub/step/npttools/fedex.exe.Z** and **p21.exe.Z**. No guarantees are made as to their performance. For maximal portability and safety, we strongly recommend rebuilding the executables.

Instructions are presented here which describe installation using "djgpp", a free publicly-available port of GCC, the GNU C compiler, to DOS. The instructions also depend on a Make-like tool, such as **qmk**. These tools have the convenient properties that they are compatible with the UNIX conventions used in the **makefile**. For example, object files end with **.o**, rather than the more conventional **.obj**.

As of this writing, djgpp can be ftp'd from **omnigate.clarkson.edu** from the directory **ftp/pub/msdos/djgpp**. Other miscellaneous utilities can be found in nearby directories. Required files include **`gas###.zip`**, **`gcc###.zip`**, **`djdev.zip`**, **`bnubn###.zip`**, and **`qddvx###.zip`**. (The ### indicates a number which may vary depending on the release.)

Mail access is available by emailing to **`archive-server@omnigate.clarkson.edu`**, using a message body of "**`help`**" and "**`index msdos/djgpp`**" on separate lines.

Installation instructions for djgpp come with djgpp itself. The remaining instructions in this section assume djgpp and the other utilities mentioned already.

lex and yacc

The **`lex`** and **`yacc`** files must be converted to C code. For simplicity, we recommend **`yacc`** and **`lex`** processing be done on a UNIX box and the output files be used on the PC. Building **`fedex`** and **`p21`** on a UNIX box suffices to force creation of the C files.

fedex

To build **`fedex`**, execute the command:

```
make dosfedex
```

This creates a file called **`fedex.ex`**e which is a DOS binary that can be run from the command line.

p21

`To build p21`, execute the command:

```
make dosp21
```

This creates a file called **`p21.exe`** which is a DOS binary that can be run from the command line.

Questions, Problems, and Support

While we are willing to listen to problems, requests for extension, etc., we cannot guarantee any kind of response. Since the system is distributed in source form, you are encouraged to experiment with the system, especially if you have problems with it. While it is often quicker for you to have us diagnose your problems, it is quicker for us to have you diagnose your own problems. This software is a research prototype, intended to spur development of commercial products.

Nonetheless, if you do have questions and/or problems, you may send e-mail to hotline@cme.nist.gov. Please include schemas, version numbers, platform descriptions, and any other information that could be relevant.

Disclaimer

Trade names and company products are mentioned in the text in order to adequately specify experimental procedures and equipment used. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

Acknowledgments

This work was funded by the NIST Scientific and Technical Research Services as part of the ARPA Persistent Object Base project, and the Computer-aided Acquisition and Logistic Support (CALS) program of the Office of the Secretary of Defense as part of the Application Protocol Development Environment project.

Thanks to Stefan Schwarz (University of the Federal Armed Forces, Munich, Germany) for figuring out how to use djgpp and qmk to compile the NIST Toolkits. Thanks to Craig Pawlak (NIST) and Joshua Lubell (NIST) for additional assistance.

Thanks to Newton Breese (NIST) for significant improvements to the content and style of this paper.

References

[1] Mason, H., ed., "Industrial Automation Systems – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles", Version 9, ISO TC184/SC4/WG PMAG Document N50, December 1991.

[2] Libes, Don, "The NIST EXPRESS Toolkit – Introduction and Overview", NISTIR 5242, National Institute of Standards and Technology, Gaithersburg, MD, October 25, 1993

[3] Spiby, P., ed., "ISO 10303 Industrial Automation Systems – Product Data Representation and Exchange – Part 11: Description Methods: The EXPRESS Language Reference Manual", ISO DIS 10303-11:1992(E), July 15, 1992.

[4] The NIST STEP Part 21 Exchange File Toolkit:An Update, NISTIR 5187, National Institute of Standards and Technology, Gaithersburg MD, September 8, 1993.

[5] Ressler, Sandy, "The National PDES Testbed Mail Server User's Guide", NISTIR 4508, National Institute of Standards and Technology, Gaithersburg, MD, Jan., 1991.

[6] Katz, Susan, "STEP On-Line Information Service User's Guide", NISTIR 4491, National Institute of Standards and Technology, Gaithersburg, MD, Jan., 1991.

[7] Libes, Don, and Fowler, Jim, "The NIST EXPRESS Toolkit – Requirements", NISTIR 5212, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

[8] Libes, Don, "The NIST EXPRESS Toolkit – Design and Implementation", *Proceedings of the Seventh Annual ASME Engineering Database Symposium*, San Diego, CA, August 9-11, 1993.

[9] Libes, Don, and Clark, Steve, "The NIST EXPRESS Toolkit – Lessons Learned", *Proceedings of the 1992 EXPRESS Users' Group (EUG '92) Conference*, Dallas, Texas, October 17-18, 1992.

[10] Libes, Don, "The NIST EXPRESS Toolkit – Using Applications", NISTIR 5206, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

[11] Libes, Don, "The NIST EXPRESS Toolkit – Programmer's Reference", National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[12] Libes, Don, "The NIST EXPRESS Toolkit – Creating Applications", National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[13] Libes, Don, "The NIST EXPRESS Toolkit – Updating Existing Applications", NISTIR 5205, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

[14] Clark, S.N., "The NIST Working Form for STEP", NISTIR 4351, National Institute of Standards and Technology, Gaithersburg, MD, November 1990.

[15] Clark, S.N., "NIST STEP Working Form Programmer's Reference", NISTIR 4353, National Institute of Standards and Technology, Gaithersburg, MD, November, 1990.

[16] Johnson, S.C., "Yacc: Yet Another Compiler compiler", *UNIX Programmer's Manual*, Seventh Edition, Bell Laboratories, Murray Hill, NJ, 1978.

[17] Stallman, Richard M., et al, *GNU's Bulletin*, Free Software Foundation, Inc., Cambridge, MA, June 1992.

[18] Lesk, M.E. and Schmidt, E., Lex: A Lexical Analyzer Generator, *UNIX Programmer's Manual*, Seventh Edition, Bell Laboratories, Murray Hill, NJ, 1978.

[19] Schreiner, Axel T. and Friedman, Jr., H. George, *Introduction to Compiler Construction with UNIX*, New York, NY, Prentice Hall, 1985.

[20] Manheimer, Kenneth L., Warsaw, Barry A., Clark, Stephen N., and Rowe, Walter, "The Depot: A Framework for Sharing Software Installation Across Organizational and UNIX Platform Boundaries", *Proceedings of the Fourth USENIX Large Installation Systems Administration (LISA) Conference*, pp. 37-46, Colorado Springs, CO, October 17-19, 1990.