

PGI[®] Workstation 5.1 Installation & Release Notes 5.1-2 Version

The Portland Group™ Compiler Technology
STMicroelectronics, Inc
9150 SW Pioneer Court, Suite H
Wilsonville, OR 97070
www.pgroup.com

While every precaution has been taken in the preparation of this document, The Portland Group™ Compiler Technology, STMicroelectronics, Inc. (PGI®) makes no warranty for the use of its products and assumes no responsibility for any errors that may appear, or for damages resulting from the use of the information contained herein. STMicroelectronics, Inc. retains the right to make changes to this information at any time, without notice. The software described in this document is distributed under license from STMicroelectronics, Inc. and may be used or copied only in accordance with the terms of the license agreement. No part of this document may be reproduced or transmitted in any form or by any means, for any purpose other than the purchaser's personal use without the express written permission of STMicroelectronics, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this manual, STMicroelectronics was aware of a trademark claim. The designations have been printed in caps or initial caps.

PGF90 is a trademark and *PGI*, *PGHPF*, *PGF77*, *PGCC*, *PGPROF*, and *PGDBG* are registered trademarks of The Portland Group Compiler Technology, STMicroelectronics, Inc. Other brands and names are the property of their respective owners.

PGI Workstation 5.1 Installation & Release Notes
Copyright © 2003

The Portland Group™ Compiler Technology
STMicroelectronics, Inc. - All rights reserved.
Printed in the United States of America

First Printing: Release 5.1-2, November 2003

Technical support: trs@pgroup.com
<http://www.pgroup.com>

Table of Contents

1	PGI WORKSTATION 5.1 INTRODUCTION	5
2	PGI WORKSTATION 5.1 INSTALLATION NOTES	5
2.1	INTRODUCTION	5
2.2	INSTALLING ON LINUX86 OR LINUX86-64	7
2.3	USING FLEXLM ON LINUX	12
2.3.1	<i>Linux Installation Summary</i>	15
2.4	INSTALLING PGI WORKSTATION ON WIN32	16
2.4.1	<i>Installing the EMACS Editor for Win32</i>	17
2.4.2	<i>Customizing the PGI Workstation Command Window</i>	17
3	PGI WORKSTATION 5.1 RELEASE NOTES	19
3.1	PGI WORKSTATION RELEASE 5.1 CONTENTS	19
3.2	SUPPORTED SYSTEMS AND LICENSING	20
3.3	NEW FEATURES FOR 32-BIT X86 AND AMD64	21
3.4	NEW FEATURES EXCLUSIVE TO AMD64	23
3.5	NEW COMPILER OPTIONS	25
3.5.1	<i>Getting Started</i>	25
3.5.2	<i>Important Linux Compiler Options</i>	26
3.5.3	<i>New Win32 Compiler Options</i>	28
3.6	PGDBG AND PGPROF SUPPORT	28
3.7	PROBLEMS CORRECTED IN RELEASE 5.1	28
3.8	AMD64 LARGE ARRAY SUPPORT	34
3.8.1	<i>Practical Limitations of -mmodel=medium</i>	35
3.8.2	<i>Compiler Limitations of -mmodel=medium</i>	36
3.8.3	<i>Large Array Example in C</i>	37
3.8.4	<i>Large Array Example in Fortran</i>	39
3.9	PGI WORKSTATION 5.1 AND LIBPTHREAD	41
3.10	LIMITATIONS	42
3.11	PGI WORKSTATION 5.1 FOR WIN32	43
3.11.1	<i>Shell Environment</i>	43
3.11.2	<i>PGI Workstation Compilers and MinGW32</i>	43
3.11.3	<i>Creating DLLs with PGI Workstation Compilers</i>	44

4 CONTACT INFORMATION & DOCUMENTATION 49

1 PGI Workstation 5.1

Introduction

Welcome to *PGI Workstation Release 5.1* for 32-bit Intel Pentium II/III/4/Xeon and AMD Athlon/AthlonXP (32-bit x86), and the new 64-bit AMD Opteron/Athlon64 (AMD64 technology) processor-based systems.

With the exception of the *PGI CDK Cluster Development Kit*, other workstation-class compilers & tools products from The Portland Group Compiler Technology (*PGHPF Workstation*, for example) are subsets of the *PGI Workstation* product. These workstation-class products are node-locked single-user, meaning one user at a time can compile on the system on which the *PGI Workstation* compilers and tools are installed. *PGI Server* products are network-floating multi-user, meaning two or more users at a time can compile on any compatible system networked to the system on which the *PGI Server* compilers are installed. These release notes apply to all workstation-class and server-class products from The Portland Group Compiler Technology.

Release 5.1 is the second production release of *PGI Workstation* that supports AMD64 technology processor-based systems. These systems can run a native 64-bit operating system, such as SuSE Linux Enterprise Server 8 SP2, or SuSE 9.0, and utilize a 64-bit address space while retaining the ability to run legacy 32-bit x86 executables at full speed. Executables generated by 32-bit x86 compilers and tools, such as previous releases of the *PGCC*, *PGC++*, *PGF77*, or *PGF90* compilers from The Portland

Group Compiler Technology, or the open source *gcc* compiler for 32-bit Linux operating systems, can execute unchanged on AMD64 technology processor-based systems. Applications that are re-compiled to take advantage of the new features of AMD64 technology processor-based systems can realize significant performance improvements. In addition, AMD64 technology processor-based systems can boot and run existing 32-bit operating systems, such as Microsoft Windows XP Professional, in which case they operate purely in 32-bit mode.

The *PGI Workstation Release 5.1* compilers have been enhanced to improve performance of generated executables through improved global optimization, vectorization and inter-procedural analysis (IPA). In addition, scheduling and code generation optimizations have been improved on both 32-bit x86 and AMD64 targets.

The *PGI Workstation Release 5.1* products can be installed in three types of code development environments;

1. **linux86** – 32-bit x86 or AMD64 processor-based systems running a 32-bit Linux operating system, with 32-bit GNU tools, utilities and libraries used by the *PGI Workstation* compilers to assemble and link for execution.
2. **win32** – 32-bit x86 processor-based systems running any of the Microsoft Windows Operating Systems (98/NT/Me/XP/2000), or AMD64 processor-based systems running Microsoft Windows XP Professional Service Pack 1. On these targets, the PGI compilers & tools products include additional tools and libraries needed to build executables for 32-bit Windows systems.
3. **linux86-64** – 64-bit AMD64 technology processor-based systems running 64-bit SuSE Linux Enterprise Server 8 SP2 or SuSE 9.0, including both 64-bit and 32-bit GNU tools, utilities and libraries used by the *PGI Workstation* compilers to assemble and link for execution. In this environment, the *PGI Workstation Release 5.1* compilers can produce either 64-bit AMD64 technology or legacy 32-bit x86 Linux executables. The 32-bit development tools and execution environment under linux86-64 are considered a cross development environment for x86 processor-based applications.

For purposes of this document, we will refer to applications compiled to use the AMD64 technology 64-bit addressing abilities as *linux86-64* executables, and programs compiled to run on systems running 32-bit Linux as *linux86* executables. It is important to remember that in a *linux86-64* environment, both *linux86-64* executables and *linux86* executables will run successfully. However, it is not the case that *linux86-64* executables will run in a *linux86* environment.

The 5.1 release of *PGI Workstation* includes support for the *linux86-64 medium memory model*, which extends the aggregate size of *.bss* sections beyond 2GB. Data objects of significantly greater size can be handled within the constraints of a *linux86-64* system's configuration (for example the amount physical memory resident on your machine can limit the size of programs you can run).

Win32 executables are created solely for execution on 32-bit Microsoft Windows x86 or AMD64 technology processor-based systems, and have no compatibility with *linux86* or *linux86-64* executables.

It is anticipated that many users will want to build and run programs from identical source code as either *linux86* executables or as *linux86-64* executables, except for the potential size of data structures. It is key that these programs be ported to run as *linux86* executables prior to building them as *linux86-64* executables, to ensure no 64-bit dependencies exist. As a methodology for developing such applications, users can build programs with *PGI Workstation Release 5.1* targeting 32-bit *linux86* environments on an AMD64 technology processor-based system running 64-bit Linux (cross-compilation). These same applications can then be recompiled on an AMD64 technology processor-based system running 64-bit Linux without source code changes to target *linux86-64* environments (native compilation).

In general, developers should find migration to the *linux86-64* environment much easier when they can compile, profile, and debug a *linux86* executable in one window while also compiling, profiling, and debugging the same program, built as a *linux86-64* executable, in a different window. In fact, it is possible to incrementally migrate portions of an application to the *linux86-64* environment if the application comprises multiple executables. For example, it is possible for a *linux86* executable and a

linux86-64 executable running in a *linux86-64* environment to communicate using sockets.

2 PGI Workstation 5.1 Installation Notes

2.1 Introduction

Section 2.2 below describes how to install *PGI Workstation* in a generic manner on Linux. Section 2.4 describes how to install *PGI Workstation* on Win32 systems. Installations using these instructions do not need to run a license daemon, except as noted below.

The *PGI Workstation* compilers and tools are license-managed. As noted in the sections that follow, generation of permanent license keys is performed using your personalized account on the <http://www.pgroup.com> web page. When you purchase a permanent license, the e-mail order acknowledgement you receive includes complete instructions for logging on to the *pgroup.com* web page and generating permanent license keys.

For *PGI Workstation* products using PGI-style licensing (the default), a single user can run as many simultaneous copies of the compiler as desired, on a single system, and no license daemon or ethernet card is required. However, usage of the *PGI Workstation* compilers and tools is restricted to a pre-specified *username*. If you would like the compilers and tools to be usable under any *username*, or if you are installing a multi-user *PGI Server* product, you must request FLEXlm-style license keys when generating your keys and use FLEXlm-style licensing as outlined below.

Installation of FLEXlm-style licensing is more complicated than PGI-style licensing. If you require FLEXlm-style licensing, you must follow the installation instructions as specified in section 2.2 and then use section 2.3

to complete your installation. Section 2.3 describes how to configure license daemons for Linux, including installation of the license daemon and proper initialization of the `LD_LIBRARY_PATH` environment variable. FLEXlm-style licensing is not currently available with *PGI Workstation* products for Win32.

Regardless of the licensing mechanism you choose, when the *PGI Workstation* compilers and tools are first installed they are usable for 15 days without a permanent license key.

NOTE

At the conclusion of the trial period, the PGI compilers and tools and any executable files generated prior to the installation of permanent license keys will cease to function. Any executables, object files, or libraries created using the PGI compilers in demo mode must be recompiled with permanent license keys in place.

Executable files generated with permanent license keys in place are unconstrained, and will run on any compatible system regardless of whether the *PGI Workstation* compilers are installed. However, if you change the configuration of your system by adding or removing hardware, your license key may become invalid. Please contact The Portland Group Compiler Technology if you expect to reconfigure your system to ensure that you do not temporarily lose the use of the PGI compilers and tools.

For the first 60 days after your purchase, you may send technical questions about these products to the e-mail address trs@pgroup.com. If you have purchased a subscription, you will have access to e-mail support for an additional 12 months and will be notified by e-mail when maintenance releases occur and are available for electronic download and installation. Phone support is not currently available. Contact us at sales@pgroup.com if you would like information regarding the subscription service for the PGI products you have purchased.

2.2 Installing on Linux86 or Linux86-64

Those familiar with releases of PGI Workstation for Linux prior to release 5.0 should note that the installation directory structure has changed. The path to the PGI Workstation 5.1 Release compilers must be modified accordingly.

For installation on 32-bit x86 processor-based systems, the PGI Workstation installation script will install only the linux86 versions of the PGI compilers and tools. For installation on an AMD64 technology processor-based system running a linux86-64 execution and development environment, the PGI Workstation installation script will attempt to install both the linux86 version and linux86-64 version of the compiler products requested. If the user specifies /usr/pgi as the base directory, for example:

Name of directory	Contents
/usr/pgi/linux86/5.1/bin	linux86 versions of the compilers and tools
/usr/pgi/linux86/5.1/lib	linux86 versions of the libraries, created on all platforms
/usr/pgi/linux86/5.1/liblf	linux86-only large-file-support (-Mlfs) versions of the libraries.
/usr/pgi/linux86/5.1/include	linux86 versions of header files
/usr/pgi/linux86-64/5.1/bin	linux86-64 versions of the compilers and tools
/usr/pgi/linux86-64/5.1/lib	linux86-64 versions of the libraries, created on all platforms. Not to be used for -mcmmodel=medium development.

/usr/pgi/linux86-64/5.1/libso	<i>linux86-64</i> <i>-fPIC</i> libraries for <i>-mmodel=medium</i> support
/usr/pgi/linux86-64/5.1/include	<i>linux86-64</i> versions of header files

When the install script installs the *linux86-64* versions on a supported AMD64 technology processor-based system running a *linux86-64* environment, the *linux86* versions will be installed as well in a separate area. The compilers and supporting components have the same names, and the environment you target by default (*linux86-64* or *linux86*) will depend on the version of the compiler that comes first in your path.

Bring up a shell command window on your system. The instructions below assume you are using *cs*, *sh*, *ksh*, or some compatible shell. Appropriate modifications will be necessary when setting environment variables if you are using a shell that is not compatible with one of these three.

Step 1 – If you received this software on a CD-ROM, please skip to step 2. If you downloaded the PGI Workstation compilers and tools software from <http://www.pgroup.com/Downloads> or another electronic distribution site, then in the instructions that follow, <tarfile> needs to be replaced with the name of the file that was downloaded.

The compressed tar file needs to be uncompressed and untar'd before installation.

```
% gunzip <tarfile>.tar.gz
% tar xpf <tarfile>.tar
```

Note that the products cannot be installed into the same directory where the tar file is unpacked, so it is recommended you execute the above commands in /tmp or another location that is *not* the installation directory.

All software should fit into less than 100 MB of disk space. Approximately 250 MB are required during installation. Half of that can be recovered by deleting the tar file after installation is complete. For AMD64 technology *linux86-64* installations, assuming double the disk

space requirements (200/500) is sufficient.

Step 2 – The install script *must* be run to properly install the software. If you are installing from a CD-ROM, issue the following command:

```
% /mnt/cdrom/install
```

NOTE: If you have difficulty running this script, check the permissions on /dev/null. Permission should be set to “crw-rw-rw-“. Reset permissions to this value if necessary – super-user permissions are required.

Also note that some systems use a CD-ROM volume manager that may insert an additional directory in the above pathname. For example, the pathname might be

```
% /cdrom/pgisoft/install
```

If you are not sure how to access the CD-ROM drive, check with your system administrator.

If you downloaded the software from the Internet, change to the directory where you uncompressed and untar’d the tar file, and run:

```
% ./install
```

The install script will list the products that are available on the CD-ROM or in the *download*. You will be asked which products should be installed and to select an installation directory. After the software is installed, the script will do some system-specific customization and then initialize the licensing, which is covered in the steps below.

Step 3 – All of the *PGI Workstation* products are license-managed. *PGI Workstation* products that are node-locked and limited to a single user do not require the running of a license daemon. If you want the *PGI Workstation* compilers to be usable by any one user rather than locked to a specific username, or if you are installing a multi-user *PGI Server* product, you must use FLEXlm and must specifically request FLEXlm-style keys

when generating license keys on the <http://www.pgroup.com> web page. If you have purchased the compiler or tools that you are installing, you should have received an order acknowledgement e-mail with instructions on how to generate your license keys through the *pgroup.com* web page. *Note:* FLEXlm-style licensing of the *PGI Workstation* products is not available on Win32 systems.

The install script asks for your real name, your username, and your email address. It then creates a fifteen-day license and prints a message like this:

```
NOTE: your evaluation license will expire in
14 days, 23.6 hours. For a permanent license,
please read the order acknowledgement that you
received. Connect to https://www.pgroup.com/License
with the username and password in the order
acknowledgement.
```

```
Name: <your name>
User: <your username>
Email: <your e-mail address>
Hostid: PGI=9BF378E0131FF0C3CD37F6
FLEXlm hostid: 00a024a3dfe7
Hostname: yourhost.yourdomain.com
Installation: /usr/pgi
PGI Release: 5.1-2
```

The message above is also saved to the file `/usr/pgi/license.info` for retrieval at a later time.

Once you have obtained your permanent license keys using your personalized account on the *pgroup.com* web page, place them in the file `/usr/pgi/license.dat` (substitute the appropriate installation directory path if you have not installed in the default `/usr/pgi` directory). If you want the *PGI Workstation* compilers to be usable by any one user, rather than locked to a specific username, you must use FLEXlm and must specifically request FLEXlm-style license keys using your account on the *pgroup.com* web page.

Step 4 – You can view the online HTML and PDF documentation using any web browser. Assuming you use *Netscape*, issue the following

command:

```
% netscape /usr/pgi/index.htm
```

You may want to place a bookmark on this location for easy future reference to the online manuals.

Step 5 – With either the temporary or permanent license file in place, execute the following commands to make the products you have purchased accessible. Note that the path settings below assume that a Linux product has been installed.

Assuming `cs` and installation in the default `/usr/pgi` directory:

```
% set path = (/usr/pgi/linux86/5.1/bin $path)
% setenv MANPATH "$MANPATH":/usr/pgi/linux86/man
```

Or, assuming `bash`, `sh` or `ksh`:

```
% PATH=/usr/pgi/linux86/5.1/bin:$PATH
% export PATH
% MANPATH=$MANPATH:/usr/pgi/linux86/man
% export MANPATH
```

If you are also installing the *linux86-64* versions of the compilers, and wish to target the *linux86-64* environment as the default, perform the same setup with an alternate path setting:

```
% set path = (/usr/pgi/linux86-64/5.1/bin $path)
% setenv MANPATH "$MANPATH":/usr/pgi/linux86-64/man
```

Or, assuming `bash`, `sh` or `ksh`:

```
% PATH=/usr/pgi/linux86-64/5.1/bin:$PATH
% export PATH
% MANPATH=$MANPATH:/usr/pgi/linux86-64/man
% export MANPATH
```

You should add the above commands to your startup files to ensure you

have access to the *PGI Workstation* products upon future logins.

Step 6 – You can verify the release number of the products you have installed using the `-V` option on any of the compiler commands. If you use `-v` instead of `-V`, you will also see the sequence of steps the compiler will use to compile and link programs for execution on your system.

- For Fortran 77, use "pgf77 -v x.f"
- For Fortran 90, use "pgf90 -v x.f"
- For HPF, use "pghpf -v x.f"
- For C++, use "pgCC -v x.c"
- For ANSI C, use "pgcc -v x.c"

Note that the files `x.f` or `x.c` need not exist in order for you to successfully execute these commands.

Installation is now complete. For the first 60 days after your purchase, you may send technical questions about these products to the e-mail address trs@pgroup.com. If you have purchased a subscription, you will have access to e-mail support and automatic minor upgrade releases for an additional 12 months and will be notified by e-mail whenever a new release is available for electronic download and installation. Phone support is not currently available. Contact us at sales@pgroup.com if you would like information regarding the subscription service for the products you have purchased.

2.3 Using FLEXlm on Linux

If you want the *PGI Workstation* compilers to be usable by any one user, rather than locked to a specific *username*, or if you are installing a multi-user *PGI Server* product, you must use the FLEXlm software license management system from MacroVision Software as outlined below.

Step 1 – Install the software as described in section 2.2 above.

Step 2 – Once you have obtained permanent FLEXlm-style license keys (see section 2.2 above, *Step 3*, for how to obtain these), place them in a file named `license.dat` in the `/usr/pgi` directory. For example, if you have purchased *PGF77 Workstation* for Linux, the `license.dat` file should look similar to the following:

```
SERVER <hostname> <hostid> 7496
DAEMON pgroupd <install_dir>/linux86/bin/pgroupd

FEATURE pgf77-linux86 pgroupd 5.100 31-dec-0 1 \
2B9CF0F163159E4ABE32 VENDOR_STRING=107209:16 \
HOSTID=<hostid> ck=49

FEATURE pgprof pgroupd 5.100 31-dec-0 1 \
6BDCE0B12EC19D0909F0 VENDOR_STRING=107209:16 \
HOSTID=<hostid> ck=60
```

`<hostname>` and `<hostid>` should match those you submitted to us and `<install_dir>` must be changed to match the directory in which the compilers are installed. In particular, `<install_dir>` should match the value of `/usr/pgi` as defined above.

NOTE: In the feature line component `VENDOR_STRING=107209`, `107209` is the Product ID Number (PIN) for this installation. You will have a similar unique PIN number for your installation. Please include your PIN number when sending mail to us regarding technical support for the products you have purchased.

Step 3 – When the license file is in place, execute the following commands to make the products you have purchased accessible. If you are not using other products managed by FLEXlm, and have not previously set the environment variable `LM_LICENSE_FILE`, issue the following command to do so (assuming `cs`):

```
% setenv PGI /usr/pgi
% setenv LM_LICENSE_FILE $PGI/license.dat
```

Or, assuming `bash`, `sh` or `ksh`:

```
% LM_LICENSE_FILE=/usr/pgi/license.dat
% export LM_LICENSE_FILE
% export PGI=/usr/pgi
```

If you are using other products managed by FLEXlm, and have previously set the environment variable `LM_LICENSE_FILE`, either incorporate our license keys into your existing license file or issue the following command to append our license file to the definition of `LM_LICENSE_FILE` (assuming `cs`):

```
% setenv PGI /usr/pgi
% setenv LM_LICENSE_FILE \
"$LM_LICENSE_FILE":/usr/pgi/license.dat
```

Or, assuming `sh` or `ksh`:

```
% LM_LICENSE_FILE= \
$LM_LICENSE_FILE:/usr/pgi/license.dat
% export LM_LICENSE_FILE
% export PGI=/usr/pgi
```

You should add the above commands to your startup files to ensure you have access to our products upon future logins.

If `LM_LICENSE_FILE` is not set or exported, and the node-locked 15-day temporary license file `/usr/pgi/PGIinstall` still exists, then `/usr/pgi/PGIinstall` will be used for resolving compiler licenses.

Step 4 – You must now start the license manager daemon. Edit the shell script template `/usr/pgi/linux86/5.1/bin/lmgrd.rc`. If you have installed the compiler(s) in a directory other than `/usr/pgi`, substitute the correct installation directory into ‘`/usr/pgi`’ part on line 3 of the script. Now exit the editor and issue the following command to start the license server and pgroup license daemon running on your system:

```
% lmgrd.rc start
```

If you wish to stop the license server and license daemon at a later time, you can do so with the command:

```
% lmgrd.rc stop
```

To make sure that the license server and pgroupd daemon are started each time your system is booted, log in as root, set the PGI environment variable as above, and then execute the following two commands:

```
% cp /usr/pgi/linux86/5.1/bin/lmgrd.rc \  
  /etc/rc.d/init.d/lmgrd  
% ln -s /etc/rc.d/init.d/lmgrd \  
  /etc/rc.d/rc3.d/S90lmgrd
```

Note that your system's default runlevel may be something other than '3', and if it is, that number should be used above in setting the correct subdirectory. Run `/sbin/runlevel` to check the system's runlevel. Note also that if you're using a Linux distribution other than Red Hat, your `rc` files may be in a directory other than `/etc/rc.d`. Specifically, SuSE startup files are in `/etc/init.d`.

Some Linux distributions, such as SuSE and Red Hat, include the *chkconfig(8)* utility that manages the runlevel scripts. If your system has this tool and you wish to use it, then run the following commands:

```
% cp /usr/pgi/linux86/5.1/bin/lmgrd.rc \  
  /etc/rc.d/init.d  
% chkconfig -- add lmgrd.rc
```

The appropriate links will be created in the `/etc/rc.d` directory hierarchy. For more information on *chkconfig*, please see the manual page.

Installation of your FLEXlm-style licensing of our products for Linux is now complete. If you have difficulties with the installation, send e-mail to trs@pgroup.com for assistance.

2.3.1 Linux Installation Summary

Now that you have installed the compilers in, for example, `/usr/pgi`, it is important that you set up your compiler environment in order to access the compilers successfully. Assume the license file is in `/usr/pgi/license.dat`, and that the **lmgrd** license manager is running. Each user should do the

following before running the compilers.

In csh

```
% setenv PGI /usr/pgi
% setenv LM_LICENSE_FILE $PGI/license.dat
% set path = ($PGI/linux86/5.1/bin $path)
% setenv MANPATH "$MANPATH":$PGI/linux86/man
```

Or, assuming bash, sh or ksh:

```
% export PGI=/usr/pgi
% export PATH=$PGI/linux86/5.1/bin:$PATH
% export MANPATH=$MANPATH:$PGI/linux86/man
% export LM_LICENSE_FILE= $PGI/license.dat
```

2.4 Installing PGI Workstation on Win32

If you are installing *PGI Workstation* from a CD-ROM, insert the CD-ROM into the CD-ROM drive on the system on which the install is to take place. An installation script will automatically be invoked and the installation process will begin. Follow the directions printed to your screen.

If you are installing *PGI Workstation* from the self-extracting file downloaded electronically via ftp, double-click on the `pgiws.exe` file with the left mouse button. The installation process will begin. Follow the instructions printed to your screen.

As with Linux, the *PGI Workstation* compilers and tools on Win32 are license-managed. However, FLEXlm-style licensing is not available on Win32. All licenses are node-locked. The Win32 serial number is used as the *hostid*. This number will be printed to your screen during the installation process, or can be located by left-clicking on *Start->Settings->Control Panel* and then double-left-clicking on the "System" icon and left-clicking on the "General" tab. The Win32 serial number will be in the middle of the System Properties window and look something like the following:

```
Registered to:  
  <your name>  
  <your organization>  
  22296-oem-0014072-07487
```

The last number above is the Win32 serial number. Obtain your permanent license keys using your personalized account on our web page as outlined in your order acknowledgement, and place them in the file `C:/PGI/license.dat` (or specify the appropriate directory path if you have installed in a directory other than the default `C:/PGI`). You should now be able to use our compilers and tools from any *PGI Workstation* command window.

2.4.1 Installing the EMACS Editor for Win32

The *emacs* editor consumes nearly 20 MB of installation space within the Win32 version of the *PGI Workstation*. For this reason, it is de-coupled from the main distribution file, `pgiws.exe`. If you are an *emacs* user and would like it installed, retrieve the EMACS file at

```
http://www.pgroup.com/Downloads
```

It is a self-installing file. As with `pgiws.exe`, simply double-left-click on `emacs.exe` after downloading and follow the instructions for installation.

Similarly, *emacs* does not auto-install from the *PGI Workstation 5.1* CD-ROM. You must explore the CD-ROM drive and double-left-click on `emacs.exe` to install *emacs* from CD-ROM.

2.4.2 Customizing the PGI Workstation Command Window

By default, when you double-left-click on the *PGI Workstation* desktop icon, a standard black-background command window appears on your screen pre-initialized with environment and path settings for use of the *PGI Workstation* compilers and tools. If you prefer different background or text colors, font style, window size, or scrolling capability, you can customize

the “shortcut” that creates the *PGI Workstation* command window. Right-click on the *PGI Workstation* desktop icon, and left-click “Properties” from the pop-up menu. Modify the features mentioned above by selecting the appropriate tabs in the pop-up window and making modifications as desired.

3 PGI Workstation 5.1 Release Notes

This document describes changes between *PGI Workstation Release 5.1* and previous releases, as well as late-breaking information not included in the current printing of the *PGI User's Guide*.

3.1 PGI Workstation Release 5.1 Contents

PGI Workstation 5.1 includes the following components:

- *PGF90* native OpenMP and auto-parallelizing Fortran 90 compiler, in versions that will run and produce code for execution in *linux86*, *linux86-64*, and *win32* development environments.
- *PGF77* native OpenMP and auto-parallelizing F77 compiler, in versions that will run and produce code for execution in *linux86*, *linux86-64*, and *win32* development environments.
- *PGHPF* data parallel High Performance Fortran compiler, in versions that will run and produce code for execution in *linux86*, *linux86-64*, and *win32* development environments.
- *PGCC* native OpenMP and auto-parallelizing ANSI and K&R C compiler in versions that will run and produce code for execution in *linux86*, *linux86-64*, and *win32* development environments.

- *PGC++* native OpenMP and auto-parallelizing ANSI C++ compiler, in versions that will run and produce code for execution in *linux86* and *linux86-64* development environments.
- *PGPROF* graphical profiler in versions that will run on *linux86* and *linux86-64*, and a command-level version for *linux86*, *linux86-64*, and *win32* environments.
- *PGDBG* multi-thread graphical debugger, in versions that will run on *linux86* and *linux86-64* development environments.
- Complete online documentation in a mixture of PDF and HTML.
- A UNIX-like shell environment for Win32, and improved support for DLLs under Win32.

Depending on the product you purchased, you may not have received all of the above components.

3.2 Supported Systems and Licensing

PGI Workstation 5.1 is supported on 32-bit Intel Pentium II/III/4/Xeon and AMD Athlon/AthlonXP (32-bit x86) processor-based systems, and 64-bit AMD Opteron (AMD64 technology) and Athlon64 processor-based systems running:

- A 32-bit *linux86* environment with a kernel version of 2.2.10 or above, including versions of Linux that use *glibc2.2.x*, such as Red Hat 7.0 to 9.0, and SuSE 7.1 to 8.2.
- A 64-bit *linux86-64* environment with a kernel version of 2.4.19 or above, including versions of Linux that use *glibc2.2.5*, such as SuSE Linux Enterprise Server 8 SP2 (supported only on AMD64 technology processor-based systems).
- A 32-bit Win32 operating system (98/NT/Me/XP/2000)

For more information about release levels and operating systems supported, go to <http://www.pgroup.com/faq/install.htm> .

The *PGI Workstation* compilers and tools are license-managed. For *PGI Workstation* products using PGI-style licensing (the default), a single user can run as many simultaneous copies of the compiler as desired, on a single system, and no license daemon or ethernet card is required. However, usage of the compilers and tools is restricted to a pre-specified *username*. If you would like the *PGI Workstation* compilers and tools to be usable under any *username*, you must request FLEXlm-style license keys and use FLEXlm-style licensing. See section 2, *PGI Workstation 5.1 Installation Notes*, for a more detailed description of licensing. The web page <http://www.pgroup.com/faq/install.htm#install1m> covers many common online license key generation questions.

3.3 New Features for 32-bit x86 and AMD64

Following are the new features included in *PGI Workstation 5.1*:

- A number of optimizations have been made to improve 32-bit x86 register allocation, vectorization, scalar arithmetic, loop unrolling, function inlining, scheduling, interprocedural analysis and optimization (IPA), and various other optimization phases within the compilers. SPEC FP2K performance on 32-bit x86 processor targets is improved by 42% over the *PGI Workstation Release 4.1*, and SPEC FP2K performance on AMD64 technology processor-based systems improves an additional 10% when re-compiled for *linux86-64* execution. Performance improvements in user-developed benchmarks or applications will of course vary.
- Support for vectorization of loops that operate on integer data
- Fortran 90 language and runtime optimizations
- Tuned Fortran 90 intrinsic MATMUL support libraries for all data types

- Additional interprocedural analysis and optimization (IPA) enhancements for Fortran 90, specifically including propagation of array shape information
- Support for the `-pg` flag, which enables *gprof*-style sample-based profiling
- Improved interoperability with the Etnus TotalView debugger on *linux86* platforms.
- There are several enhancements to the *PGDBG* debugger:
 1. When the `-g` compiler option is specified, the compilers emit debug information in DWARF2 format, improving interoperability with 3rd party debuggers, and better compatibility with *gcc* and *g77*.
 2. *PGDBG* has made advances in internal information management, so that large programs on large systems will run more efficiently under *PGDBG* with faster load times and event handling.
 3. Thread handling has improved, and *PGDBG* navigates threads with greater stability.
 4. Source level debugging from shared libraries is now supported. A shared object must be loaded before it can be debugged using *PGDBG*.
- New installation directory structure. Users simply add `/usr/pgi/linux86/5.1/bin` to their path, instead of `/usr/pgi/linux86/bin`, when the install directory is `/usr/pgi`. Users of the *linux86-64* versions of the *PGI Workstation* compilers add `/usr/pgi/linux86-64/5.1/bin` to their path to use the AMD64 technology version of the compilers and tools by default. This change in directory structure enables installation and use of the *PGI Workstation* compilers for both *linux86* and *linux86-64* targets on AMD64 technology processor-based systems. It will also allow installation and use of multiple releases of the *PGI Workstation* compilers and tools for all releases subsequent to

Release 5.0 (using the `-V` switch will tell you which release you are using for a given compilation). Installing the 5.1 release in `/usr/pgi` will not disturb the 5.0 release installation that was also installed in `/usr/pgi`.

- ACML high-performance LAPACK and BLAS math libraries bundled.
- Fully ISO-compliant C++, with the exception of exported templates.
- Improved support for creation of DLLs on Win32. *PGF90* runtime libraries are now also available as DLLs.
- Updated UNIX-like command-level environment on Win32
- Completely updated hardcopy and online documentation.
- A new environment variable `MP_WARN`, to enable/disable runtime OpenMP warning messages.

3.4 New Features Exclusive to AMD64

Following are the features included in *PGI Workstation 5.1* that are specific to AMD64 technology processor-based systems.

- Further tuning of the AMD64 code generator, resulting in SPEC_{FP2K} Fortran benchmarks geometric mean improved by over 8%, Polyhedron geometric mean improved by over 25%, and significantly improved overall performance on the AMD Core Math Library (ACML) relative to *PGI Workstation 5.0-2*.
- Support for the 64-bit *linux86-64* execution environment and the x86-64 *medium memory model*.
- Support for 64-bit addressing and aggregate data sets in excess of the 2GB limit on 32-bit x86 processor-based system. Single static data objects larger than 2GB are now supported in *PGF77* and *PGCC*.

- Support for 64-bit integer arithmetic in hardware.
- Use of the extended general-purpose register set available on AMD64 technology processors (16 general-purpose registers instead of the 8 available on 32-bit x86 processor-based systems).
- Use of the extended Streaming SIMD Extensions (SSE) register set available on AMD64 technology processors (16 SSE registers instead of the 8 available on 32-bit x86 processor-based systems).
- Most 32-bit and 64-bit floating-point arithmetic is performed using SSE instructions. *Note:* SSE instructions use standard IEEE 32-bit and 64-bit arithmetic for floating point calculations and rounding behavior. The x87 floating-point stack used on x86 processor-based systems uses IEEE 80-bit arithmetic for register-to-register operations by default. As a result, arithmetic results on AMD64 technology processor-based systems running a *linux86-64* environment should closely match results obtained on most 32-bit and 64-bit RISC processor-based workstations. These results can sometimes differ from those obtained on a 32-bit x86 or AMD64 processor-based system running a *linux86* environment, where scalar arithmetic is by default performed using x87 instructions.
- Subroutine and function calling sequences are modified to comply with the *x86-64 Application Binary Interface*, which can be found online at <http://www.x86-64.org/documentation>.
- The *linux86-64* compilers themselves are 64-bit applications, and can only run on AMD64 technology processor-based machines.
- *PGDBG* supports debugging of *linux86-64* executables.
- *PGPROF* supports graphical display of *linux86-64* profiled output.
- In addition to SuSE Linux Enterprise Server 8 SP2, SuSE 9.0 is

now also a supported *linux86-64* environment. We have done extensive testing internally using a Beta Release of 64-bit Red Hat Enterprise Linux 3.0, but *PGI Workstation 5.1* has not been formally validated for RHEL 3.0 64-bit.

3.5 New Compiler Options

3.5.1 Getting Started

By default, the *PGI Workstation* compilers generate code optimized for the type of processor on which compilation is performed (the compilation host). Typically, for best Fortran performance, you will want to use the *PGF90* compiler (even for FORTRAN 77 code) and the *-fastsse* option. This option is similar to *-fast*, but incorporates additional optimization options to enable use of streaming SIMD (SSE/SSE2) instructions where appropriate. The contents of the *-fastsse* switch are host-dependent, but typically include the options *-O2 -Munroll -Mnoframe -Mlre -Mvect=sse -Mcache_align*. On some systems, *-fastsse* also includes *-Mscalarsse* and *-Mflushz*.

In addition to *-fastsse*, the *-Mipa=fast* option for inter-procedural analysis and optimization can improve performance. See the *PGI User's Guide* for details on how to use this option. In particular, note that *-Mipa* requires two passes and that you must compile and link with *-Mipa* for it to be effective.

You may be able to obtain further performance improvements by experimenting with the individual *-Mpgflag* options detailed in the *PGI User's Guide* (*-Mvect*, *-Munroll*, *-Minline*, *-Mconcur*, etc). However, speed-ups using these options are typically application and system-dependent, so it is important to time your application carefully when using these options to ensure no performance degradations occur.

For OpenMP programmers, runtime warning messages can now be disabled via the **MP_WARN** environment variable. Setting it to *no* will prevent messages like "Warning OMP_NUM_THREADS greater than the

number of cpus” from being posted.

3.5.2 Important Linux Compiler Options

The following compiler options have been added or modified in *PGI Workstation Release 5.1*:

- *-Mlarge_arrays* – applies only to the PGF77 compiler. This option must be used, in combination with *-mcmode=medium*, when compiling F77 applications that use single data objects larger than 2GB in size.
- *-Mdwarf1* | *-Mdwarf2* – generate debug information in either the DWARF1 or DWARF2 format. The Release 5.1 compilers produce DWARF2 by default. Must be used with *-g*.
- *-fast* – has been modified to be equivalent to: *-O2 -Munroll=c:1 -Mnoframe -Mlr*.
- *-fastsse* – has been modified to be equivalent to: *-fast -Mvect=sse -Mscalarsse -Mcache_align -Mflushz*.
- *-Minline[={... | except:<func>}]* – A new sub-option *except:<func>* directs the compiler to not inline an entry *func* or list of entries.
- *-pg* – Generate code and an executable to enable gprof-style sample-based profiling. A *gmon.out* trace file is produced when the program is executed on the target processor.
- *-tp { k7 | k8-32 | k8-64 | piii | p5 | p6 | p7 | px }* – Set the target architecture. By default, the *PGI Workstation* compilers produce code specifically targeted to the type of processor on which the compilation is performed. In particular, the default is to use all supported instructions wherever possible when compiling on a given system. As a result, executables created on a given system may not be useable on previous generation systems (for example,

executables created on a Pentium 4 may fail to execute on a Pentium III or Pentium II). Processor-specific optimizations can be specified or limited explicitly by using the `-tp` option. In this way, it is possible to create executables that are useable on previous generation systems. With the exception of `k8-64`, any of these sub-options are valid on any x86 or AMD64 technology processor-based system. The `k8-64` sub-option is valid only on AMD64 technology processor-based systems running a 64-bit operating system. Following is a list of possible sub-options to `-tp`, and the processors they are intended to target:

<code>k7</code>	generate 32-bit code for AMD AthlonXP and compatible processors.
<code>k8-32</code>	generate 32-bit code for AMD64 technology and compatible processors.
<code>k8-64</code>	generate 64-bit code for AMD64 technology and compatible processors.
<code>piii</code>	generate 32-bit code for a Pentium III processor-based system.
<code>px</code>	generate 32-bit code that is useable on any x86 processor-based system.
<code>p5</code>	generate 32-bit code for Pentium and compatible processors.
<code>p6</code>	generate 32-bit code for Pentium Pro/II and compatible processors.
<code>p7</code>	generate 32-bit code for Pentium 4 and compatible processors.

- The environment flag **MP_WARN** controls the runtime posting of OpenMP warnings. `'setenv MP_WARN no'` will prevent OpenMP runtime warnings.

3.5.3 New Win32 Compiler Options

New Win32-specific compiler options are documented below in section 3.9.3, *Creating DLLs with PGI Workstation Compilers*.

3.6 PGDBG and PGPROF Support

PGDBG is supported as a graphical debugger in both the *linux86* and *linux86-64* execution and development environments. Like the compilers, *PGDBG* for *linux86-64* must run in a *linux86-64* execution environment. *PGDBG* for *linux86* execution is a separate version, and it will also run in the *linux86-64* execution environment, but only with *linux86* executables. The *linux86-64* version of *PGDBG* will only debug executables built to run as *linux86-64* executables. *PGDBG* for *linux86-64* has been enhanced to disassemble the AMD64 technology new instructions, and is more compatible with *gcc*, *g77*, and *g++* debug information.

PGPROF is a graphical display tool of profile information created through execution of programs compiled and linked using *-Mprof*. Only a *linux86* executable version of *PGPROF* is provided, but it is able to read either types of profile output, and resides in both *linux86* and *linux86-64 bin* areas.

See the *PGI Tools Guide* for a complete description of the usage and capabilities of *PGDBG* and *PGPROF*.

3.7 Problems Corrected in Release 5.1

The following problems were corrected in the current release. Most were reported in *PGI Workstation 5.0-2* or previous releases. Problems found in *PGI Workstation 5.0-2* may not have occurred in the previous releases. A description of the problem is given, but some problems can only be described in general terms because of complexity or confidentiality. An *Internal Compiler Error* (ICE) is usually the result of checks the compiler components make on internal data structures, discovering inconsistencies that could lead to faulty code generation.

Technical Problem Reports (TPRs) Corrected in PGI Release 5.1-2				
TPR	Rel	Lang	Description	Symptom
2840	4.0	pgCC	pgCC bug.c causes severe errors	PGCC-ICE error. getsname:
2862	4.0	pgf90	<pre> module mymodule integer, public, parameter :: n=10 integer, private :: i type, public :: mytype integer :: array(n) end type mytype type, public :: yourtype integer :: array(n) end type yourtype type, public :: histype integer :: array(n) end type histype type(mytype), public, parameter :: myex = mytype ((/(0,i=1,n)/)) type(yourtype), public, parameter :: & yourex = yourtype (myex %array) type(histype), public, parameter :: hisex = histype(0) end module mymodule program main use mymodule write(*,*) myex write(*,*) write(*,*) yourex write(*,*) write(*,*) hisex write(*,*) stop end program main </pre>	PGF90-S-0069-Illegal implied DO expression (bug1.f90: 22) PGF90-S-0000-Internal compiler error. _dinit_acl,array 29
2997	5.0	pgcc	<pre> #include <search.h> char * key1 = "entry1"; char * key2 = "entry2"; int main () { int size; ENTRY entry1; ENTRY entry2; ENTRY * e; size = 100; if (!hcreate(size)) { printf("error creating hash table\n"); return 1; } entry1.key = key1; entry2.key = key2; e = hsearch(entry1, FIND); </pre>	gcc passes small structs differently on Opteron than larger structs. We have fixed this to be compatible with gcc. Program seg faults on Optérons.

			<pre> if (e) printf("found %s\n", e->key); else printf("not found\n"); e = hsearch(entry1, ENTER); e = hsearch(entry2, ENTER); e = hsearch(entry1, FIND); if(e) printf("found %s\n", e->key); e = hsearch(entry2, FIND); if (e) printf("found %s\n", e->key); hdestroy();return 0; } </pre>	
3005	5.0	pgf77	users program worked with g77, not with pgf77.	-mcmmodel=medium failing
3007	5.0	all	openmp programs properly warn when cpu count is less than thread count. MP_WARN flag prevents the message	Annoying warning message
3009	5.0	pgcc	Problem with programs built by inlining alloca.	GCC test fails with -Minline
3011	5.0	pgf90	program test character*4 C50 character c*4(50) end	The c*4(50) format was not accepted.
3022	5.0	fortran	problems with -mcmmodel=medium	seg faults,
3028	5.0	pgcc	<pre> #include <stdlib.h> > #include <time.h> #include <sys/types.h> #include <sys/times.h> void main() { clock_t elp; ldiv_t res; struct tms itim; elp += times(&itim); res = ldiv(elp, CLK_TCK); } </pre>	seg faults due to small structs
3034	5.0	pgCC	<pre> #include <stdio.h> struct { void Add_Unique(int gn) {printf ("gn = %d\n", gn);} } node_connectivity[1]; struct { int Global_Number () {return 6067;} int Index () {return 0;} } inode; static int inverse_map[1] = {0}; main () { int iv=inverse_map[inode.Index()]; if(iv>=0) node_connectivity[iv].Add_Uniqu e(inode.Global_Number()); </pre>	seg faults when compiled pgCC -O2 x.C

			}	
3035	5.0	pgf90	<pre>integer in,iam logical doit doit=.true. in=1 iam=1 IN1=1 if(in.eq.IN1 .or. iam.ne.0 .or. .not.doit)then Print *,'in loop 1' endif if(.not.doit .or. in.eq.IN1 .or. iam.ne.0)then Print *,'in loop 2' endif jj=0 if(in.eq.IN1) jj=1 if(iam.ne.0) jj=2 if(.not.doit) jj=3 if(jj.ne.0)Print*,'in loop 3' stop end</pre>	Program fails if compiled with -i8
3037	5.0	pgf90	<pre>Added new f95 syntax to external stmt external xx,yy !f90 syntax external :: xx,yy ! f95 syntax</pre>	reported error on second format.
3038	4.1	pgcc	<pre>#include <stdio.h> void main() { int guard; unsigned char next,val; guard = 0; val = 0x80; next = val << 1; if (next) {guard = 1;} printf("%d\n", guard); }</pre>	pgcc -O1 foo.c prints '1' pgcc -O0 foo.c prints '0'
3039	5.0	pgf77	program fails with pgf77 -fast	ICE when compiled pgf77 -fast bad.f
3041	5.0	fortran	<pre>Function IRTyp(W) Implicit Real*8(A-H,O-Z) Dimension W(3,4), IOrd(5) Save IOrd, One Data IOrd/2,3,4,6,1/, One/1.d0/ Call TrDet(TrI,DetI,W) ITr = ANInt(TrI) Det = ANInt(DetI) IS = Sign(One,Det) IRTyp = IS*IOrd(IS*ITr+2) End</pre>	ICE with -fastsse -tp p7

3042	5.0	pgf90	PROGRAM testing implicit none structure /mystruct/ integer*4 i1,i2,i3,i4,too much end structure record /mystruct/ s1,t1,t2 common /t_cmn/ t1,t2 s1.i1 = 101 s1.i2 = 102 t1 = s1 t2 = t1 print *, 'out:', s1.i1, t1.i1, t2.i1 END	Output differs when compiled with or without -fpic
3043	4.1	pgf90	program caused ICE	%pgf90 -c -O BHS.f Lowering Error: symbol sinv\$sd is a member reference
3048	5.0	pgf90	64-bit pgcc sizeof(sizeof()) should return 8	sizeof(sizeof()) returns 4
3049	5.0-b	pgf90	program foo integer(8) :: i real(8), dimension(1000) :: x x(:) = 1.0 print *, (x(i),i=1,400) end	implicit do loop failed with integer*8 index
3051	5.0	pgf90	CASE stmt with DEFAULT not last case fails.	MCNP5 fails
3053	5.0	pgf90	solved by 3041	ICE (stack87pos)
3059	5.0	pgf90	MODULE pure_sub_mod IMPLICIT NONE CONTAINS PURE SUBROUTINE pure_sub(a) IMPLICIT NONE INTEGER, INTENT(INOUT) :: a a=a*2 END SUBROUTINE pure_sub END MODULE pure_sub_mod MODULE module_mod IMPLICIT NONE CONTAINS PURE SUBROUTINE another_pure_sub(b,pure_sub) IMPLICIT NONE INTEGER, INTENT(INOUT) :: b INTERFACE PURE SUBROUTINE pure_sub(a) IMPLICIT NONE INTEGER, INTENT(INOUT) :: a END SUBROUTINE pure_sub END INTERFACE CALL pure_sub(b) b=b*2 END SUBROUTINE another_pure_sub END MODULE module_mod PROGRAM test_PURE_error_prog USE module_mod,ONLY: another_pure_sub USE pure_sub_mod,ONLY: pure_sub IMPLICIT NONE INTEGER :: a a=2 CALL another_pure_sub(a,pure_sub) PRINT *,a	ICE - alignment

			END PROGRAM test_PURE_error_prog	
3060	5.0	pgf90	Correct DWARF information.	bad debug information
3062	5.0	pgf90	Correct the DWARF information	bad debug information
3063	5.0	pgf90	Correct the DWARF information	Bad debug information
3064	5.0	pgf90	Correct the DWARF information	Bad debug information
3066	5.0	pgf77	NPB LU fails with -fast . -Munroll is the problem.	program fails to compile -fast
3067	5.0	64 MPI	NPB in CDK fails with CLASS=B	mpich 64-bit fails.
3068	5.0	pgCC	program fails to compile	compile failure
3069	5.0	pgf90	POP from lanl program fails.to compile tagv.f90	compile failure
3075	4.1	pgf90	<pre> module tpg_mod implicit none public integer ,parameter,private::X1=5 contains subroutine outer() character(LEN=X1),dimension(:), pointer :: plist integer :: ierr allocate(plist(8),STAT=ierr) if (ierr.ne.0) then write(6,*)'did not alloc plist' stop endif plist(1) = "ONE " plist(2) = "TWO " -and so on plist(8) = "EIGHT" call inner(plist(3:5)) end subroutine outer subroutine inner(list) character(LEN=X1), dimension(:),intent(in)::list integer :: i do i=1,size(list) write(6,*) trim(list(i)) enddo end subroutine inner end module tpg_mod program test use tpg_mod implicit none call outer end program test </pre>	bad answers from trim(..)
3076	5.0	pgf90	<pre> module foo_module integer dummy interface subroutine assumed_shape6(xx) integer five, negfour common /bounds/ five, negfour integer*4 xx(: ,negfour:) end subroutine assumed_shape6 end interface end module foo_module program tx_f90_arrays use foo_module integer*4,dim(20,10) ::array5 integer i,j,k,l </pre>	ICE pgf90-g

			<pre> integer five, negfour common /bounds/ five, negfour do i = 0,19 do j = 0,9 array5(i+1,j+1) = j*20 + i + 1 end do end do call assumed_shape6(array5) end program tx_f90_arrays subroutine assumed_shape6(xxx) integer five, negfour common /bounds/ five, negfour integer*4 xxx(:,negfour:) print*,lbound(xxx,1),ubound(xxx,1) print*,lbound(xxx,2),ubound(xxx,2) end subroutine assumed_shape6 </pre>	
3082	5.0	pgf90	fails with -Mvect-sse	ICE

3.8 AMD64 Large Array Support

Support for the *medium memory model* in the *linux86-64* environment is provided, with some limitations due to the *PGI Workstation 5.1* compilers and some limitations that are inherent to the medium memory model itself.

The *small memory model* of the *linux86-64* environment limits the combined area for a user's object or executable to 1GB, with the Linux kernel managing usage of the second 1GB of address for system routines, shared libraries, stacks, etc. Programs are started at a fixed address, and the program can use a single instruction to make most memory references.

The *medium memory model* allows for larger than 2GB data areas, or *.bss* sections. Program code for the *medium memory model* must be compiled *-fPIC*, or position-independent, and will require additional instructions to reference memory. The effect on performance is a function of the data-use of the application.

The *linux86-64* environment provides static *libxxx.a* archive libraries that are built without *-fPIC*, and dynamic *libxxx.so* shared object libraries that are compiled *-fPIC*. The *-mmodel=medium* linker switch implies the *-fPIC* switch and will utilize the shared libraries by default. Similarly, the *\$PGI/linux86-64/5.1/lib* directory contains the libraries for building *small memory model* codes, and the *\$PGI/linux86-64/5.1/libso* directory contains

shared libraries for building `-mmodel=medium` and `-fPIC` executables. *Note:* It appears from the GNU tools and documentation that creation of *medium memory model* shared libraries is not supported. However, you can create static archive libraries (.a) that are `-fPIC`.

3.8.1 Practical Limitations of `-mmodel=medium`

The 64-bit address capability of the AMD64 technology can cause unexpected problems when data sizes are enlarged significantly. For example:

- Initializing a large array with a data statement may result in a very large assembly file and object file, where a line of assembler source is required for each element in the array initialized. Code compilation and linking will be very time consuming as well. To avoid this time and space-consuming problem, consider initializing large arrays in the program area in a loop rather than in the declaration.
- Stack space can be problem for data that is stack based. *limit stacksize unlimited* can be used to enable as much stack space as possible, but it will be limited nonetheless and is dependent on the amount of physical memory. Determine if *limit stacksize 512M* gives as large a stack area as unlimited.
- If your executable is much larger than the physical size of memory, page swapping can cause it to run dramatically slower and it may even fail. This is not a compiler problem. Try smaller data sets to determine if a problem is due to page thrashing, or not.
- Be sure your linux86-64 system is configured with swap space sufficiently large to support the data sets used in your application(s). If your memory+swap space is not sufficiently large, your application will likely encounter a segmentation fault at runtime.

Overall, it is important to understand the practical limitations of the *linux86-64* environment, and users should take reasonable care to determine if a program failure is due a compiler limitation or an operating

system limitation.

3.8.2 Compiler Limitations of `-mmodel=medium`

There are a number of limitations on large arrays that are not due to the medium memory model, but are due to compiler limitations. Some of these limitations are common between the GNU compilers (*gcc* and *g77*) and the *PGI Workstation 5.1* compilers, and some are specific to the *PGI Workstation 5.1* compilers.

1. Individual (static) data objects are still limited to less than 2GB in size in *PGF90*, *PGHPF* and *PGC++*. *PGF77* and *PGCC* now support static data objects larger than 2GB. F77 applications that use single data objects larger than 2GB must be compiled with *PGF77* using both the `-mmodel=medium` and `-Mlarge_arrays` options. C applications with large data objects need only be compiled with *PGCC* using `-mmodel=medium`. Support for single data objects larger than 2GB will be added to the *PGF90*, *PGHPF* and *PGC++* compilers in a future release.
2. *PGF90* treats all local static data in a program as one collective data object (it is aggregated into a single ELF section). In *PGI Workstation 5.1*, section sizes in *PGF90*-compiled programs are still limited to be less than 2GB in size. To utilize multiple very large static arrays (aggregate greater than 2GB) using *PGF90*, you will need to assign the large arrays to distinct COMMONs. This limitation will be removed in a future release of the *PGI* compilers and tools. As of Release 5.1, this limitation no longer applies to *PGF77*.
3. Dynamically allocated arrays in C can be larger than 2GB, using the `malloc()` operation to return a pointer in either *PGCC* or *gcc*. *PGF90*, however, cannot `ALLOCATE` individual arrays of size greater than 2GB. NOTE: no compile-time or pre-specified run-time error will occur if you attempt to `ALLOCATE` an array larger than 2GB using *PGF90*. 32-bit truncation will usually mask the problems you encounter. If you pass a pointer to an array larger

than 2GB (for example by returning it to a Fortran program unit from a called C program unit), you must access it very carefully. Usually, the array should be referenced as if it is a one-dimensional array. All referencing of multiple dimension arrays within Fortran is restricted to a 32-bit index derived from the individual indices of the array. Indexing into a dynamically allocated 1-dimensional array, where the index is declared `INTEGER*8`, will be successfully evaluated as a 64-bit address.

3.8.3 Large Array Example in C

Consider the following example, where the aggregate size of the arrays exceeds 2GB.

```
% cat bigadd.c
#include <stdio.h>
#define SIZE 600000000 /* > 2GB/4 */
static float a[SIZE],b[SIZE];
main() {
long long i,n,m;
float c[SIZE]; /* goes on stack */
n=SIZE;m=0;

    for(i=0;i<n;i+=10000){
        a[i]=i+1;
        b[i]=2.0*(i+1);
        c[i]=a[i]+b[i];
        m=i;
    }
    printf("a[0]=%g b[0]=%g c[0]=%g\n", a[0], b[0],
c[0]);
    printf("n=%d a[%d]=%g b[%d]=%g c[%d]= %g\n", n, m, m,
m, a[m], b[m], c[m]);
}
```

Compiled using `gcc`, without using `-mmodel=medium`:

```
% gcc -o bigadd bigadd.c
/tmp/ccWt7q8Q.o: In function `main':
```

```
/tmp/ccWt7q8Q.o(.text+0x6e): relocation truncated to
fit: R_X86_64_32S .bss
/tmp/ccWt7q8Q.o(.text+0x8c): relocation truncated to
fit: R_X86_64_32S .bss
```

This is a link-time error, and is due to the linker attempting to create a *small memory model* executable when the static arrays exceed the less than 1GB aggregate limit inherent in that model. Re-compiling using `-mmodel=medium`:

```
% gcc -mmodel=medium -o bigadd bigadd.c
/tmp/ccVQpbPj.s: Assembler messages:
/tmp/ccVQpbPj.s:97: Error: .COMMON length (-2147483648.)
<0! Ignored.
```

The `gcc` compiler incorrectly converts a greater than 2G value to a *negative* 32-bit number in an assembler statement. This error does not occur using `pgcc 5.1`:

```
% pgcc -mmodel=medium -o bigadd bigadd.c
```

Why? When `SIZE` is greater than `2G/4`, and the arrays are of type `float` with 4 bytes per element, the size of each array is *greater* than 2GB. With 5.1 `pgcc`, using the `-mmodel=medium` switch, a static data object *can now be* > 2GB in size. Note that if you execute with the above settings in your environment, you may see the following:

```
% bigadd
Segmentation fault
```

Execution fails because the stack size is not large enough. Try resetting the stack size in your environment:

```
% limit stacksize 3000M
```

Note that `'limit stacksize unlimited'` will probably not provide as large a stack as we are using above.

```
% bigadd
a[0]=1 b[0]=2 c[0]=3
n=600000000 a[599990000]=5.9999e+08
```

```
b[599990000]=1.19998e+09 c[599990000]=1.79997e+09
```

The size of the *bss* section of the *bigadd* executable is now larger than 2GB:

```
% size --format=sysv bigadd | grep bss
.bss                4800000008    5245696
% size --format=sysv bigadd | grep Total
Total                4800005080
```

3.8.4 Large Array Example in Fortran

The following example needs preprocessing, and illustrates the major difference between *pgf90* and *pgf77* in the 5.1 release. *pgf90* and *pgf77* both use 64-bit addresses when compiled *-mmodel=medium*, but only *pgf77* allows for 64-bit integer index support.

Consider the following example:

```
% cat matadd.f
      program matadd
      integer i, j, k, size, l, m, n
#ifdef define (USE_PGF90)
      parameter (size=13000) ! 1GB<size<2GB
#else
      parameter (size=16000) ! >2GB
#endif
      parameter (m=size,n=size)
      real*8 a(m,n),b(m,n),c(m,n),d
#ifdef define (USE_PGF90)
      common/aa/a
      common/bb/b
      common/cc/c
#endif
      do i = 1, m
        do j = 1, n
          a(i,j)=10000.0D0*dbble(i)+dbble(j)
          b(i,j)=20000.0D0*dbble(i)+dbble(j)
        enddo
      enddo
```

```

!$omp parallel
!$omp do
  do i = 1, m
    do j = 1, n
      c(i,j) = a(i,j) + b(i,j)
    enddo
  enddo
!$omp do
  do i=1,m
    do j = 1, n
      d = 30000.0D0*dbble(i)+dbble(j)+dbble(j)
      if(d .ne. c(i,j)) then
        print *, "err i=", i, "j=", j
        print *, "c(i,j)=", c(i,j)
        print *, "d=", d
        stop
      endif
    enddo
  enddo
!$omp end parallel
print *, "M =", M, ", N =", N
print *, "c(M,N) = ", c(m,n)
end

```

When compiled with *PGF90* using `-mmodel=medium`:

```
% pgf90 -mp -o matadd matadd.f -mmodel=medium
```

The *PGF90* compiler places all local static data for a program in a single section of the generated executable, and therefore the combined sizes exceed the 2GB limit on any single section (in this case *.bss*) generated by *PGF90*. This limitation is handled by

```

common/aa/a
common/bb/b
common/cc/c

```

after *a*, *b*, and *c* are first declared. You can compile and execute as

follows:

```
% pgf90 -mp -o matadd -DUSE_PGF90 -Mpreprocess matadd.f
-mcmodel=medium
% setenv OMP_NUM_THREADS 2
% matadd
M =          13000 , N =          13000
c(M,N) =      390026000.0000000
```

For compiling with pgf77, we enlarge the array dimensions to create (>2GB) a very large array, and compile with

```
% pgf77 -mp -o matadd matadd.f -mcmodel=medium
-Mlarge_arrays -fast
% setenv OMP_NUM_THREADS 2
% matadd
M =          16000 , N =          16000
c(M,N) =      480032000.0000000
```

On a 1.8 GHz Dual processor Opteron box with 4GB of memory, the above example ran about 33% faster with OMP_NUM_THREADS set to 2, instead of 1.

3.9 PGI Workstation 5.1 and *libpthread*

Previous releases of the *PGI Workstation* Linux compiler products have included a customized version of *libpthread.so* called *libpgthread.so*. The purpose of this library is to give the user more thread stack space to run OpenMP and *-Mconcur* compiled programs. With Release 8.0 Red Hat and equivalent releases, *libpthread.so* and *libpthread.a* have ‘re-sizeable’ thread stack areas. In these cases

1. The filename `$PGI/linux86/5.1/lib/libpgthread.so` is a soft link to `/usr/lib/libpthread.so`.
2. Instead of ‘setenv MPSTKZ 256M’, for example to increase the *libpgthread.so* thread stack area, the Linux system call ‘limit stacksize 256M’ now applies to thread stacks.

3.10 Limitations

- We are seeing problems in the latest 32-bit Linux *libpthread.a* libraries. They appear to no longer have floating stacks, and actually reset the stack size to 2MB if linked into an application. We consider this a bug. This bug appears not to exist in current 64-bit *libpthread.a* implementations, but does exist in the 32-bit *libpthread.a* libraries distributed with 64-bit Linux operating systems.
- Support for full 64-bit memory access from a *linux86-64* system currently has limitations.

The *linux86-64* *PGF90*, *PGHPF* and *PGC++* compilers currently only support 32-bit array index values, and an aggregate (product of dimensions) memory limit of a 32-bit integer index over a 2GB memory. In theory, a large number of ‘less than 2GB’ arrays can be defined and should compile without problem. Note that an *AMD64* 64-bit address is being generated by the compiler and used at runtime, with the *AMD64* 64-bit memory instructions.

- Using *-Mipa=arg* and *-Minline* together with *PGF77* can result in segmentation faults or incorrect execution by the generated executable. You can work around this problem by listing specific sub-options to *-Mipa*, not including *arg* or *fast* (*arg* is included in *-Mipa=fast*)
- *PGDBG* does not support the *call* command in *linux86-64* environments.
- *PGDBG* cannot print the values of *PRIVATE* variables while debugging Fortran threads in an *OpenMP* parallel region.
- *PGDBG* now supports the source-level debugging of shared objects, but a shared object must be loaded before it can be debugged using *PGDBG*.
- *PGF90* does not properly inline functions with optional

arguments. It will report an *'argument mismatch'* and not inline the function, but it should not cause errors.

3.11 PGI Workstation 5.1 for Win32

3.11.1 Shell Environment

On Win32, a UNIX-like shell environment is bundled with *PGI Workstation*. After installation, a double-left-click on the *PGI Workstation* icon on your desktop will launch a *bash* shell command window with pre-initialized environment settings. Most familiar UNIX commands are available (*vi*, *emacs*, *sed*, *grep*, *awk*, *make*, etc). If you are unfamiliar with the *bash* shell, reference the user's guide included with the online HTML documentation.

Alternatively, you can launch a standard Win32 command window pre-initialized for usage of the compilers by selecting the appropriate option from the *PGI Workstation* program group accessed in the usual way through the "Start" button.

Except where noted in the *PGI User's Guide*, the command-level compilers and tools on Win32 function identically to their UNIX counterparts. You can customize your command window (white background with black text, add a scroll bar, etc.) by right-clicking on the top border of the *PGI Workstation* command window, selecting "Properties", and making the appropriate modifications. When the changes are complete, Win32 will allow you to apply the modifications globally to any command window launched using the *PGI Workstation* desktop icon.

3.11.2 PGI Workstation Compilers and MinGW32

With *PGI Workstation 5.1*, certain MinGW32 components have been updated to enable creation of DLLs within the *PGI Workstation* UNIX-like development environment under Win32 without the need to install the full *cygwin* environment.

3.11.3 Creating DLLs with PGI Workstation Compilers

Release 5.1 contains the *PGI Workstation* runtime libraries in both dynamic-link library (DLL) form and static library form. The static libraries are still the default. To use the *PGI Workstation* compilers to create an executable that links to the runtime DLLs, use the compiler flag *-Mdll* at the link step.

The process of creating a dynamic-link library (DLL) has changed significantly with the Release 5.1 compilers. Changes to both the compilers and linker have simplified the process and the resulting DLLs are more robust. As a result, however, object files compiled with releases prior to Release 5.1 must be recompiled with the 5.1 compilers before being put into a DLL. Objects compiled with pre-5.0 compilers, including all

PGI Workstation 5.0-beta releases, cannot be used in building DLLs with *PGI Workstation 5.1*.

The process of creating DLLs using the *PGI Workstation* compilers is outlined in the *PGI User's Guide*. Here are some limitations and restrictions:

1. If an executable is linked with any *PGI Workstation*-compiled DLL, the *PGI Workstation* runtime library DLLs must be used (in particular the static libraries can't be used). To accomplish this, use the compiler option *-Mdll* when creating the executable.
2. If a DLL is built with the *PGI Workstation* compilers, the runtime DLLs must be used. The compiler option *-Mmakedll* ensures the correct runtime libraries are used.
3. A MAIN program cannot be contained in a DLL.
4. Do not use *-Mprof* with *PGI Workstation* runtime library DLLs. To build an executable for profiling, use the static libraries. The static libraries will be used by default in the absence of *-Mdll*.
5. When the *PGF90* and *PGHPF* compilers build a DLL, an

additional object file, *pgdllmain.o*, is linked into the DLL. This object contains a modified *DllMain()* routine that performs certain initialization steps. It is possible to create a DLL without linking in this object by using *-Mnopgdllmain*, however, an alternate *DllMain()* that contains the same *PGI Workstation DLL* initialization routines must be provided for the created DLL to function correctly. The contents of the *PGI Workstation pgdllmain.c* file follow. Add these changes to the alternate DLL main file.

```
/*
 * Copyright 1990-2000, The Portland
 * Group, Incorporated. Copyright
 * 2000-2003, STMicroelectronics,
 * Incorporated. All rights reserved.
 *
 * STMICROELECTRONICS, INCORPORATED
 * PROPRIETARY INFORMATION This software
 * is supplied under the terms of a license
 * agreement or nondisclosure agreement
 * with STMicroelectronics and may not be
 * copied or disclosed except in accordance
 * with the terms of that agreement.
 */

/* pgdllmain.c - DllMain contains
 * initialization of pghpf commons to
 * addresses of their counterparts in the
 * runtime. Link this module into user-created
 * (hpf/f90) dynamic-link libraries.
 */

#include <windows.h>

struct {
    void * pghpf_01p;
    void * pghpf_02p;
    void * pghpf_03p;
    void * pghpf_04p;
}
```

```

} pghpf_0;

struct {
    void * pghpf_0cp;
} pghpf_0c;

struct {
    void * pghpf_linenop;
} pghpf_lineno;

struct {
    void * pghpf_local_modep;
} pghpf_local_mode;

struct {
    void * pghpf_mep;
} pghpf_me;

struct {
    void * pghpf_npp;
} pghpf_np;

struct {
    void * pghpf_typep;
} pghpf_type;

extern void * __pgget_hpf_01_addr(void);
extern void * __pgget_hpf_02_addr(void);
extern void * __pgget_hpf_03_addr(void);
extern void * __pgget_hpf_04_addr(void);
extern void * __pgget_hpf_0c_addr(void);
extern void * __pgget_hpf_lineno_addr(void);
extern void * __pgget_hpf_local_mode_addr(void);
extern void * __pgget_hpf_me_addr(void);
extern void * __pgget_hpf_np_addr(void);
extern void * __pgget_hpf_type_addr(void);

BOOL WINAPI
DllMain( HINSTANCE dll_handle, DWORD reason,

```

```

LPVOID reserved )
{
    switch(reason) {
    case DLL_PROCESS_ATTACH:
        pghpf_0.pghpf_01p = __pgget_hpf_01_addr();
        pghpf_0.pghpf_02p = __pgget_hpf_02_addr();
        pghpf_0.pghpf_03p = __pgget_hpf_03_addr();
        pghpf_0.pghpf_04p = __pgget_hpf_04_addr();
        pghpf_0c.pghpf_0cp =
__pgget_hpf_0c_addr();
        pghpf_lineno.pghpf_linenop =
__pgget_hpf_lineno_addr();
        pghpf_local_mode.pghpf_local_modep =
__pgget_hpf_local_mode_addr();
        pghpf_me.pghpf_mep =
__pgget_hpf_me_addr();
        pghpf_np.pghpf_npp =
__pgget_hpf_np_addr();
        pghpf_type.pghpf_typep =
__pgget_hpf_type_addr();
        break;
    case DLL_PROCESS_DETACH:
        break;
    case DLL_THREAD_ATTACH:
        break;
    case DLL_THREAD_DETACH:
        break;
    }

    return TRUE;
}

```


4 Contact Information & Documentation

You can contact The Portland Group Compiler Technology at:

*The Portland Group Compiler Technology
STMicroelectronics, Inc.
9150 SW Pioneer Court, Suite H
Wilsonville, OR 97070*

Or contact us electronically using any of the following means:

*Fax: +1-503-682-2637
Sales: sales@pgroup.com
Support: trs@pgroup.com
WWW: http://www.pgroup.com*

All technical support is provided by e-mail or submissions using an online form at <http://www.pgroup.com/support>. Phone support is not currently available. Many common questions and problems can be resolved at our frequently asked questions (FAQ) site at <http://www.pgroup.com/faq>.

Online documentation is available by pointing your browser at either your local copy of the documentation:

`file:/usr/pgi/doc/index.htm`

or online at <http://www.pgroup.com/doc>.

