

PGI<sup>®</sup> Server 7.0

PGI<sup>®</sup> Workstation 7.0

# Installation & Release Notes

The Portland Group™  
STMicroelectronics, Inc  
Two Centerpointe Drive  
Lake Oswego, OR 97035  
[www.pgroup.com](http://www.pgroup.com)

While every precaution has been taken in the preparation of this document, The Portland Group™ (PGI®), a wholly-owned subsidiary of STMicroelectronics, Inc., 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.

*PGF95*, *PGF90* and *PGC++* are trademarks and *PGI*, *PGHPF*, *PGF77*, *PGCC*, *PGPROF*, and *PGDBG* are registered trademarks of STMicroelectronics, Inc. \*Other brands and names are the property of their respective owners.

*PGI Server 7.0 / PGI Workstation 7.0*  
*Installation & Release Notes*  
Copyright © 2007

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

First Printing:	Release 7.0-2, February, 2007
Second Printing:	Release 7.0-3a, March, 2007
Third Printing:	Release 7.0-3, April, 2007
Fourth Printing:	Release 7.0-4, May, 2007
Fifth Printing:	Release 7.0-5, June, 2007
Sixth Printing:	Release 7.0-6, July, 2007
Seventh Printing:	Release 7.0-7, August, 2007

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

# Table of Contents

---

<b>1</b>	<b>PGI RELEASE 7.0 INTRODUCTION .....</b>	<b>1</b>
1.1	PRODUCT OVERVIEW .....	1
1.2	TERMS AND DEFINITIONS .....	2
<b>2</b>	<b>PGI RELEASE 7.0 INSTALLATION NOTES .....</b>	<b>7</b>
2.1	INTRODUCTION .....	7
2.2	LICENSING.....	8
2.2.1	<i>PGI Workstation Licensing</i> .....	9
2.2.2	<i>PGI Server Licensing</i> .....	9
2.2.3	<i>Trial Licensing Key Constraints</i> .....	10
2.2.4	<i>Licensing Keys and System Configurations</i> .....	10
2.3	INSTALLING ON LINUX .....	11
2.3.1	<i>Preparing to Install on Linux</i> .....	11
2.3.2	<i>Installation Steps for Linux</i> .....	13
2.3.3	<i>End-user Environment Settings on Linux</i> .....	20
2.4	INSTALLING FLEXLM ON LINUX .....	21
2.5	INSTALLING ON WINDOWS .....	25
2.5.1	<i>Preparing to Install on Windows</i> .....	25
2.5.2	<i>Installation Steps for Windows</i> .....	27
2.5.3	<i>Customizing the Command Window</i> .....	30
2.5.4	<i>PGI Workstation Directory Structure</i> .....	30
2.5.5	<i>Using LM_LICENSE_FILE</i> .....	32
2.5.6	<i>Common Windows Installation Problems</i> .....	33
2.6	INSTALLING ON APPLE MAC OS X .....	34
2.6.1	<i>Preparing to Install on Apple Mac OS X</i> .....	35
2.6.2	<i>Installation Steps for Mac OS</i> .....	35
2.6.3	<i>End-user Environment Settings on Mac OS X</i> .....	39
2.7	INSTALLING FLEXLM ON MAC OS X.....	40

<b>3</b>	<b>PGI RELEASE 7.0 RELEASE NOTES .....</b>	<b>43</b>
3.1	PGI RELEASE 7.0 CONTENTS .....	44
3.2	SUPPORTED SYSTEMS .....	44
3.2.1	<i>Supported Processors</i> .....	44
3.2.2	<i>Supported Operating Systems</i> .....	46
3.2.3	<i>New System Calls</i> .....	48
3.3	NEW OR MODIFIED COMPILER FEATURES .....	48
3.4	COMPILER OPTIONS .....	52
3.4.1	<i>Getting Started</i> .....	52
3.4.1.1	<i>Using -fast, -fastsse, and Other Performance-Enhancing Options</i> .....	52
3.4.2	<i>New or Modified Compiler Options</i> .....	53
3.5	PGI WORKSTATION 7.0 FOR WINDOWS.....	55
3.5.1	<i>The Windows Command Environment</i> .....	55
3.5.2	<i>MKS Toolkit Compatibility</i> .....	56
3.5.3	<i>Using Shared object files in SFU and SUA</i> .....	56
3.6	PGI WORKSTATION 7.0 FOR MAC OS .....	58
3.6.1	<i>Mac OS Debugging Requirements</i> .....	58
3.7	GENERATING PGI UNIFIED BINARIES .....	59
3.7.1	<i>Unified Binary Command-line Switches</i> .....	59
3.7.2	<i>Unified Binary Directives and Pragmas</i> .....	60
3.8	USING ENVIRONMENT MODULES .....	60
3.9	PGDBG AND PGPROF .....	61
3.9.1	<i>PGDBG New Features</i> .....	62
3.10	THE REDIST DIRECTORIES .....	63
3.10.1	<i>PGI Redistributables</i> .....	63
3.10.2	<i>Microsoft Redistributables</i> .....	63
3.11	CUSTOMIZING WITH SITERC AND USER RC FILES .....	64
3.12	KNOWN LIMITATIONS .....	65
3.13	CORRECTIONS .....	69
3.13.1	<i>Corrections in 7.0-7</i> .....	70
3.13.2	<i>Corrections in 7.0-6</i> .....	70
3.13.3	<i>Corrections in 7.0-5</i> .....	71
3.13.4	<i>Corrections in 7.0-4</i> .....	71
3.13.5	<i>Corrections in 7.0-3</i> .....	72
3.13.6	<i>Corrections in 7.0-2</i> .....	74
<b>4</b>	<b>CONTACT INFORMATION AND DOCUMENTATION.....</b>	<b>79</b>

# 1 PGI Release 7.0 Introduction

---

Welcome to Release 7.0 of *PGI Workstation* and *PGI Server*, a set of Fortran, C, and C++ compilers and development tools for 32-bit and 64-bit x86-compatible processor-based workstations and servers running versions of the Linux, Windows, and Mac OS operating systems.

All workstation-class compilers and tools products from The Portland Group (*PGHPF Workstation*, for example) are subsets of the *PGI Workstation Complete* product. These workstation-class products provide a node-locked single-user license, meaning one user at a time can compile on the one system on which the *PGI Workstation* compilers and tools are installed.

*PGI Server* products are offered in configurations identical to the workstation-class products, but provide network-floating multi-user licenses. This means that two or more users can use the *PGI* compilers and tools concurrently 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 compiler products from The Portland Group.

## 1.1 Product Overview

Release 7.0 of *PGI Workstation* and *PGI Server* includes the following components:

- *PGF95* OpenMP\* and auto-parallelizing Fortran 90/95 compiler.
- *PGF77* OpenMP and auto-parallelizing FORTRAN 77 compiler.
- *PGHPF* data parallel High Performance Fortran compiler.

NOTE: *PGHPF* is not supported on Windows platforms.

- *PGCC* OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- *PGC++* OpenMP and auto-parallelizing ANSI C++ compiler.
- *PGPROF* graphical OpenMP/multi-thread performance profiler.
- *PGDBG* graphical OpenMP/multi-thread symbolic debugger.
- Online documentation in PDF, HTML and `man` page formats.
- A UNIX\*-like shell environment for *Win32* and *Win64* platforms.

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

## 1.2 Terms and Definitions

Following are definitions of terms used in the context of these release notes.

*AMD64* – a 64-bit processor from AMD designed to be binary compatible with 32-bit *x86* processors, and incorporating new features such as additional registers and 64-bit addressing support for improved performance and greatly increased memory range. This includes the AMD\* Athlon64\*, AMD Opteron\* and AMD Turion\* processors.

*Barcelona* – In this document the Quad-Core AMD Opteron(TM) Processor (i.e. Opteron Rev x10) is referred to as Barcelona.

*DLL* – a dynamic linked library on *Win32* or *Win64* platforms of the form *xxx.dll* containing objects that are dynamically linked into a program at the time of execution.

*driver* – the compiler *driver* controls the compiler, linker, and assembler, and adds objects and libraries to create an executable. The `-dryrun` option illustrates operation of the driver. `pgf77`, `pgf95`, `pghppf`, `pgcc`, `pgCC` (Linux), and `pgcpp` are drivers for the PGI compilers. A `pgf90` driver is retained for compatibility with existing makefiles, even though `pgf90` and `pgf95` drivers are identical.

*Dual-core* – some x64 CPUs incorporate two complete processor cores (functional units, registers, level 1 cache, level 2 cache, etc) on a single silicon die. These are referred to as Dual-core processors. For purposes of OpenMP, threads, or MPI parallelism, these cores function as two distinct processors. However, the two processing cores are on a single chip occupying a single socket on the system motherboard. For purposes of PGI software licensing, one dual-core processor is treated as a single CPU. In particular, a PGI Workstation license that typically limits OpenMP process creations to a maximum of 8, will, on a Dual-core system with 8 CPUs, enable process creations up to a maximum of 16 (4 CPUs \* 2 Cores per CPU or 8 total processes).

*EM64T* – a 64-bit IA32 processor with *Extended Memory 64-bit Technology* extensions designed to be binary compatible with AMD64 processors. This includes Intel Pentium 4, Intel Xeon, and Intel Core 2 processors.

*FLEXlm* – The flexible license management software from Macrovision.

*Hyperthreading (HT)* – some IA32 CPUs incorporate extra registers that allow 2 threads to run on a single CPU with improved performance for some tasks. This is called *hyperthreading* and abbreviated *HT*. Some *linux86* and *linux86-64* environments treat IA32 CPUs with HT as though there were a 2<sup>nd</sup> *pseudo* CPU, even though there is only one physical CPU. Unless the Linux kernel is *hyperthread-aware*, the second thread of an *OpenMP* program will be assigned to the *pseudo* CPU, rather than a real second physical processor (if one exists in the system). *OpenMP* Programs can run very slowly if the second thread is not properly assigned.

*IA32* – an Intel Architecture 32-bit processor designed to be binary compatible with x86 processors, but incorporating new features such as streaming SIMD extensions (SSE) for improved performance. This includes the Intel Pentium\* 4 and Intel Xeon\* processors. For simplicity, these release notes refer to x86 and IA32 processors collectively as *32-bit x86* processors.

*Large Arrays* – arrays with aggregate size larger than 2GB, which require 64-bit index arithmetic for accesses to elements of arrays. Program units that use *Large Arrays* must be compiled using *-mmodel=medium*. If *-Mlarge\_arrays* is specified and *-mmodel=medium* is not specified, the default small memory model is used, and all index arithmetic is performed in 64-bits. This can be a useful mode of execution for certain existing 64-bit applications that use the small memory model but allocate and manage a single contiguous data space larger than 2GB.

*linux86* – 32-bit Linux operating system running on an *x86*, *AMD64* or *EM64T* processor-based system, with 32-bit GNU tools, utilities and libraries used by the PGI compilers to assemble and link for 32-bit execution.

*linux86-64* – 64-bit Linux operating system running on an *AMD64* or *EM64T* processor-based system, with 64-bit and 32-bit GNU tools, utilities and libraries used by the PGI compilers to assemble and link for execution in either *linux86* or *linux86-64* environments. The 32-bit development tools and execution environment under *linux86-64* are considered a cross-development environment for *x86* processor-based applications.

*Mac OS* – collectively, all *osx86* and *osx86-64* platforms supported by the PGI compilers.

*-mcmmodel=small* – compiler/linker switch to produce *small memory model* format objects/executables in which both code (*.text*) and data (*.bss*) sections are limited to less than 2GB. This switch is the default and only possible format for *linux86* 32-bit executables. This switch is the default format for *linux86-64* executables. Maximum address offset range is 32-bits, and total memory used for OS+Code+Data must be less than 2GB.

*-mcmmodel=medium* – compiler/linker switch to produce *medium memory model* format objects/executables in which code sections are limited to less than 2GB, but data sections can be greater than 2GB. This option is supported *only* in *linux86-64* environments. It must be used to *compile* any program unit that will be linked in to a 64-bit executable that will use aggregate data sets larger than 2GB and will access data requiring address offsets greater than 2GB. This option must be used to *link* any 64-bit executable that will use aggregate data sets greater than 2GB in size. Executables linked using *-mcmmodel=medium* can incorporate objects compiled using *-mcmmodel=small* as long as the *small* objects are from a shared library.

*Network Installation* – A term that applies to installing software on a shared file system available to many machines. Typically this type of installation allows multiple users to access and run the software – up to the number of licenses associated with the software.

*NUMA – Non-Uniform Memory Access.* A type of multi-processor system architecture in which the memory latency from a given processor to a given portion of memory can vary, resulting in the possibility for compiler or programming optimizations to ensure frequently accessed data is “close” to a given processor as determined by memory latency.

*osx86* – 32-bit Apple Mac OS Operating Systems running on an *x86* Core 2 or Core 2 Duo processor-based system with the 32-bit Apple and GNU tools, utilities, and libraries used by the PGI compilers to assemble and link for 32-bit execution. The *PGI Workstation* preview supports *Mac OS* 10.4.9 only.

*osx86-64* – 64-bit Apple Mac OS Operating Systems running on an *x64* Core 2 Duo processor-based system with the 64-bit and 32-bit Apple and GNU tools, utilities, and libraries used by the PGI compilers to assemble and link for either 64- or 32-bit execution. The *PGI Workstation* preview supports *Mac OS* 10.4.9 only.

*SFU* – *Windows Services for Unix* is the precursor to *SUA*. *SFU* supports 32-bit applications on *Windows 2000*, *Windows Server 2003*, and *XP*.

*Shared library* – a Linux library of the form *libxxx.so* containing objects that are dynamically linked into a program at the time of execution.

*SSE* – collectively, all SSE extensions supported by the PGI compilers.

*SSE1* – 32-bit IEEE 754 FPU and associated *streaming SIMD extensions* (SSE) instructions on Pentium III, AthlonXP\* and later 32-bit *x86*, *AMD64* and *EM64T* compatible CPUs, enabling scalar and packed vector arithmetic on single-precision floating-point data.

*SSE2* – 64-bit IEEE 754 FPU and associated SSE instructions on P4/Xeon and later 32-bit *x86*, *AMD64* and *EM64T* compatible CPUs. SSE2 enables scalar and packed vector arithmetic on double-precision floating-point data.

*SSE3* – additional 32-bit and 64-bit SSE instructions to enable more efficient support of arithmetic on complex floating-point data on 32-bit *x86*, *AMD64* and *EM64T* compatible CPUs with so-called *Prescott New Instructions* (PNI), such as Intel IA32 processors with EM64T extensions and newer generation (Revision E and beyond) AMD64 processors.

*SSE4A and ABM* – AMD Instruction Set enhancements for the Quad-Core AMD Opteron Processor. Support for these instructions is enabled by the `-tp barcelona` or `-tp barcelona-64` switch.

*SSSE3* – an extension of the SSE3 instruction set found on the Intel Core 2.

*Static linking* – a method of linking:

- On Linux, use `-Bstatic` to ensure all objects are included in a generated executable at link time. Static linking causes objects from static library archives of the form `libxxx.a` to be linked in to your executable, rather than dynamically linking the corresponding `libxxx.so` shared library. Static linking of executables linked using the `-mcmodel=medium` option is supported.
- On Windows, the Windows linker links statically or dynamically depending on whether the libraries on the link-line are DLL import libraries or static libraries. By default, the static PGI libraries are included on the link line. To link with DLL versions of the PGI libraries instead of static libraries, use the `-Mdll` option.

*SUA* – the Subsystem for Unix-based Applications is a source-compatible subsystem for compiling and running 32- and 64-bit UNIX-based applications on a computer running Windows. *SUA* is supported on *Windows 2003 R2* and *Vista*. *SUA* is bundled with Windows; however, the full *SUA Utilities SDK* must be installed to link programs.

*Win32* – any of the 32-bit Microsoft\* Windows\* Operating Systems (XP/2000/Server 2003) running on an *x86*, *AMD64* or *EM64T* processor-based system. On these targets, the PGI compiler products include additional Microsoft tools and libraries needed to build executables for 32-bit Windows systems.

*Win64* – any of the 64-bit Microsoft Windows Operating Systems (XP Professional / Windows Server 2003 x64 Editions) running on an *x64* processor-based system. On these targets, the PGI compiler products include additional Microsoft tools and libraries needed to build executables for either Win32 or Win64 environments.

*Windows* – collectively, all *Win32* and *Win64* platforms supported by the PGI compilers.

*x64* – collectively, all AMD64 and EM64T processors supported by the PGI compilers.

*x86* – a processor designed to be binary compatible with i386/i486 and previous generation processors from Intel\* Corporation. Used to refer collectively to such processors up to and including 32-bit variants.

*x87* – 80-bit IEEE stack-based floating-point unit (FPU) and associated instructions on *x86*-compatible CPUs.

# 2 PGI Release 7.0 Installation Notes

---

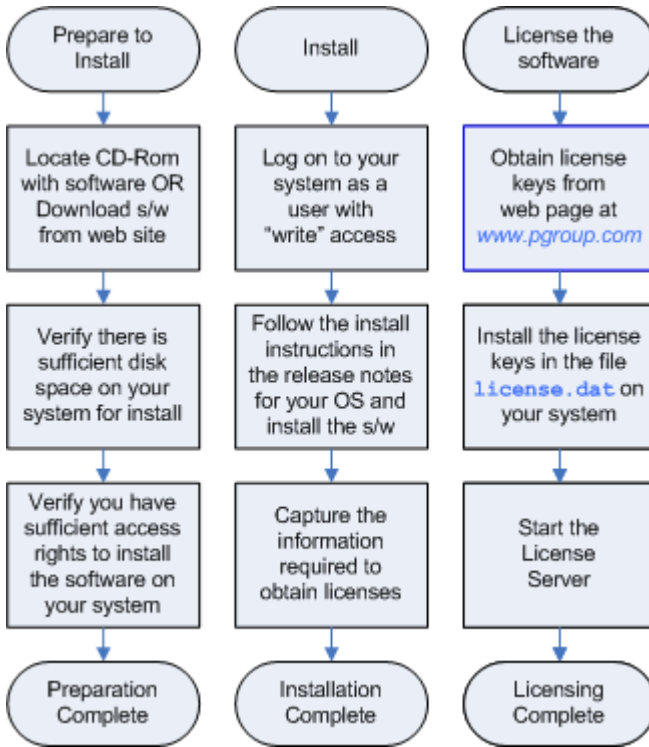
## 2.1 Introduction

This section provides an overview of the steps required to successfully install PGI Workstation or PGI Server. The remainder of this chapter provides the details of each of the steps. Specifically, section 2.2 describes licensing and how to successfully generate either trial or permanent license keys. Section 2.3 describes how to install *PGI Workstation* or *PGI Server* in a generic manner on Linux. Section 2.4 describes how to install and run a FLEXlm license daemon on Linux. Section 2.5 describes how to install *PGI Workstation* on Windows systems and how to install and run a FLEXlm license daemon on Windows. Section 2.6 describes how to install *PGI Workstation* on an Apple system.

Before you begin the installation, it is essential that you understand the flow of the installation process. There are basically three stages of the process:

- **Prepare to install** – verifying that you have all the required information and software.
- **Install the software** – installing the software appropriate for your operating system.
- **License the software** – installing the licenses and starting the license server.

The following illustration provides a high-level overview of the installation process.



For more complete information on these steps and the specific actions to take for your operating system, refer to the following sections.

## 2.2 Licensing

The PGI compilers and tools are license-managed. There are two types of license keys: trial and permanent. This section describes the generation of these license keys.

**Note.** You must install the software before you obtain your license because the licensing request prompts you for information that is generated during the software installation.

To generate these keys, you use your personalized account on the <http://www.pgroup.com> web page.

- **Permanent keys:** When you purchase a permanent license, the e-mail order acknowledgement you receive includes complete instructions for logging on to the [pgroup.com](http://www.pgroup.com) web page and generating permanent license keys.
- **Trial keys:** When you register for a trial, you should generate trial keys using the web page: <http://www.pgroup.com/license/trial>.

*Note.* The preview release of *PGI Workstation for Mac OS* has a time-limited test license. No license keys are required for Mac OS during the preview period.

Now let's look at specific licensing issues.

## 2.2.1 PGI Workstation Licensing

PGI Workstation is licensed to a single system.

On Linux, there are two licensing options for PGI Workstation: PGI-style licensing and FLEXlm-style licensing.

- **PGI-style licensing** for *PGI Workstation* allows a named user to run as many simultaneous copies of the compiler and tools as desired, but usage is restricted to a pre-specified username.
- **FLEXlm-style licensing** allows any user of the system to use PGI Workstation; however, only a single copy of a compiler or tool is allowed to run at any given time.

On Windows, including SUA and SFU, only FLEXlm-style licensing is supported.

## 2.2.2 PGI Server Licensing

*PGI Server* supports multi-user, network-floating licenses. FLEXlm-style licensing is required for *PGI Server*. Multiple users can use the PGI compilers simultaneously from multiple systems on a network when those systems have a properly configured version of PGI Server installed.

On Linux, *PGI Server* may be installed locally on each machine on a network or may be installed once on a shared file system available to each machine. If you select the second method, a *network install*, adding another machine to the group running the compilers is a much simpler process in this release; you adjust to the unique characteristics of the newly added system with a customization script that must be executed on each machine in the group.

Note. On Windows, *PGI Server* must be installed locally on each machine.

If you require FLEXlm-style licensing, you must request FLEXlm-style license keys when generating permanent keys. First install the PGI compilers and tools according to the instructions in the following sections. You must then install and configure the FLEXlm license management software according to the instructions in sections 2.3 or 2.4. These sections describe how to configure license daemons for Linux and Windows, respectively, including installation of the license daemon and proper initialization of the `LM_LICENSE_FILE` environment variable.

## 2.2.3 Trial Licensing Key Constraints

### *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 with a trial key must be recompiled with permanent license keys in place.*

## 2.2.4 Licensing Keys and System Configurations

Executable files generated with permanent license keys in place are unconstrained, and will run on any compatible system regardless of whether the PGI compilers are installed.

If you change the configuration of your system by adding or removing hardware, your license key may become invalid. Please contact The Portland Group 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 PGI Software Subscription, you have access to e-mail support for an additional 12 months and you are 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.

The following sections describe how to install *PGI Workstation* or *PGI Server*.

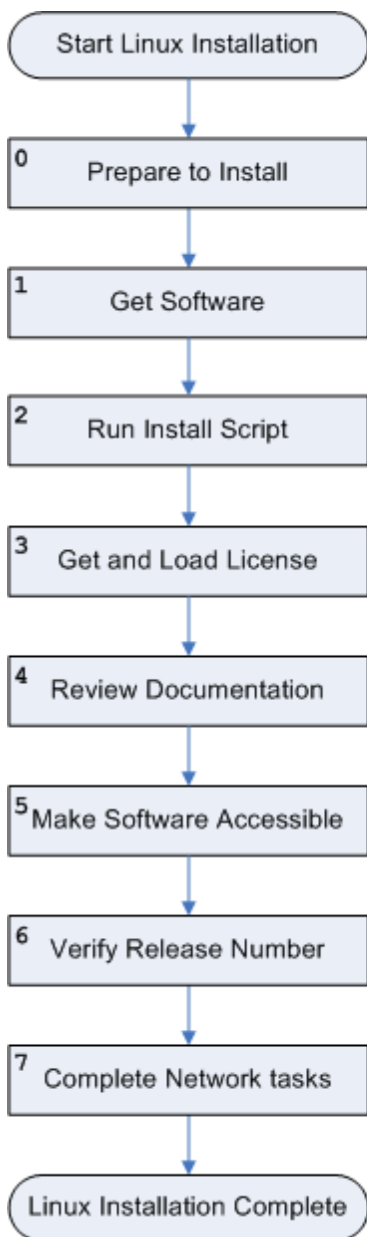
## 2.3 Installing on Linux

This section describes how to install *PGI Workstation* or *PGI Server* on a Linux system. It covers local and network installs and is applicable to permanent or trial installations.

- For installations on 32-bit x86 systems, the PGI installation script installs only the linux86 versions of the PGI compilers and tools.
- For installations on 64-bit x64 systems running a linux86-64 execution and development environment, the PGI installation script installs the linux86-64 version of the PGI compilers and tools. Further, if the 32-bit gcc development package is already installed on the system, the 32-bit linux86 tools are also installed on a 64-bit x64 system.

### 2.3.1 Preparing to Install on Linux

In preparation for installing *PGI Workstation* or *PGI Server* on Linux, first review the following overview of the Linux installation process.



- Select which products?
- Select 32- or 64- bit?
- Local or network install?
- Check disk space availability
- Start Shell Command Window

- Locate CD-ROM OR
- Download from web site

- Do I have write permissions?
- Can I access the software?

- Use PGI-style or FLEXlm- style?
- Obtain licenses from web site
- Place licenses in `license.dat` file
- Start license server

- HTML and PDF versions are available
- Did I bookmark the location?

- Execute appropriate commands to make products available

- Use the `-V` option on any compiler command to verify the release number

**For Network Installations Only:**

- Run local installation script on each system on network where compilers and tools are to be available

*Note.* For Linux installations, **each user** must also set their environment variables to properly access the software, as described in 2.3.3 End-user Environment Settings on Linux on page 20.

The 32-bit and 64-bit compilers, tools, and supporting components have the same command names. Further, the environment you target by default (*linux86-64* or *linux86*) depends on the version of the compiler that comes first in your path settings.

In a traditional local installation, the default installation base directory is `/opt/pgi`. If you choose to perform a network install, you should specify a shared file system for the installation base directory. You must also specify a second directory name that is local to each of the systems where the PGI compilers and tools are used. This local directory will contain the libraries to use when compiling and running on that machine. This approach allows a network installation to support a network of machines that run different versions of Linux.

To prepare for the installation:

- Locate the email that contains your Order Acknowledgement. This email contains your username and password for the web page as well as the PIN number associated with your licensing file.
- Bring up a shell command window on your system. The installation instructions assume you are using `csh`, `sh`, `ksh`, `bash`, or some compatible shell. If you are using a shell that is not compatible with one of these shells, appropriate modifications are necessary when setting environment variables.
- Verify you have enough free disk space. On the `linux86` platform, PGI installation requires 250 MB of free disk space. On the `linux86-64` platform, PGI installation requires 500 MB of free disk space.

## 2.3.2 Installation Steps for Linux

Follow these instructions to install the software:

**Step 1.** If you received this software on a CD-ROM, please skip to step 2. If you downloaded the software from <http://www.pgroup.com> or another electronic distribution site, then in the instructions that follow, replace `<tarfile>` with the name of the file that was downloaded.

**Note.** The PGI products cannot be installed into the same directory where the tar file is unpacked. Use the following command sequence to unpack the tar file in a temporary directory before installation:

```
% mkdir /tmp/pgi
% mv <tarfile>.tar.gz /tmp/pgi
% cd /tmp/pgi
% tar xpfz <tarfile>.tar.gz
```

## **Step 2.** Run Install Script.

The install script *must* be run to properly install the software.

The install script lists the products that are available on the CD-ROM or in the download package. You will be asked which products should be installed, whether to perform a traditional local installation or a network installation, and where to place the installation directory.

After the software is installed, the install script performs some system-specific customization and then initializes the licensing, as described in step 3.

- If you downloaded the software from the Internet, execute the following script in the directory where you unpacked the tar file:

```
% ./install
```

- If you are installing from a CD-ROM, issue the following command:

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

**Note.** For a network installation, you are asked for a common local directory. This local directory will be created once on each system utilizing the network installation; further, it must be created on each system *before* adding that system to the network using the compilers.

**Note.** If you have difficulty running this script, especially on a Slackware Linux system, check the permissions on `/dev/null`. Permissions should be set to “`crw-rw-rw-`”. If necessary, reset permissions to this value; to do this, super-user permissions are required.

**Note.** Some systems use a CD-ROM volume manager that may insert an additional directory in the above pathname, and require the longer pathname. For example, you may need to execute a script like this:

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

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

**Step 3.** Get and Load License.

All of the PGI compilers and tools are license-managed. *PGI Workstation* products that are PGI-style licensed and limited to a single user have no need to run a FLEXlm license daemon.

To obtain licensing information, you need the following information:

- The username and password required to connect to the website.
  - For permanent license keys, this information is in your order acknowledgement email.
  - For trial license keys, this is the username and password you used to download the installation software from the web site.
- The FLEXlm hostid and hostname of the computer on which the software is installed, provided by the installation script. You can also obtain your FLEXlm hostid by using the following command after you have installed the products and initialized the environment variables:

```
% lmutil lmhostid
```

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. Further, you must specifically request FLEXlm-style keys when generating license keys on the PGI web page at <http://www.pgroup.com/support/keylogin.htm>.

If you have purchased the compilers and tools that you are installing, you received an order acknowledgement e-mail with instructions on how to generate your license keys through the *pgroup.com* web page. The installation script asks for your real name, your username, and your email address and prints a message similar to this:

To obtain an evaluation license, go to:

<https://www.pgroup.com/license/evaluation.php>

and use your web-user access codes (email address and password) and the information below to generate a trial license.

For a permanent license, please read the order acknowledgement that you received. Connect to

<https://www.pgroup.com/support/keylogin.htm>

with the username and password provided in the order acknowledgement.

```
FLEXlm hostid: <your host id>
Hostname: <your host name>
Installation: /opt/pgi
PGI Release: 7.0-7
```

For retrieval at a later time, the preceding message is also saved to the file `/opt/pgi/license.info`, where `/opt/pgi` is the installation directory.

Once you have obtained your trial or permanent license keys using your personalized account on the *pgroup.com* web page, place them in the file `/opt/pgi/license.dat`, or substitute the appropriate installation directory path if you have not installed in the default `/opt/pgi` directory. For more information, refer to Step 2 of Installing FLEXlm on Linux on page 21.

#### **Step 4.** Review Documentation.

You can view the online HTML and PDF documentation using any web browser by opening the file:

```
/opt/pgi/linux86/7.0/doc/index.htm
```

or

```
/opt/pgi/linux86-64/7.0/doc/index.htm
```

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

#### **Step 5.** Make Software Accessible

With either the trial or permanent license file in place, execute the following commands to make the products you have purchased accessible.

For x64 *linux86-64*:

To use the x64 *linux86-64* version of the compilers and tools, execute the following commands.

**Note.** In these commands, the installation directory is the default: `/opt/pgi`.

For `csch`:

```
% set path = (/opt/pgi/linux86-64/7.0/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86-64/7.0/man
```

For `bash`, `sh` or `ksh`:

```
% PATH=/opt/pgi/linux86-64/7.0/bin:$PATH
% export PATH
% MANPATH=$MANPATH:/opt/pgi/linux86-64/7.0/man
% export MANPATH
```

### For *linux86*:

To use only the *linux86* version of the compilers and tools, or to target *linux86* as the default, use a setup similar to the previous one, changing the path settings as illustrated in the following commands.

For `csh`:

```
% set path = (/opt/pgi/linux86/7.0/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86/7.0/man
```

For `bash`, `sh` or `ksh`:

```
% PATH=/opt/pgi/linux86/7.0/bin:$PATH
% export PATH
% MANPATH=$MANPATH:/opt/pgi/linux86/7.0/man
% export MANPATH
```

To ensure access to the PGI compilers and tools for future logins, we recommend all users of the PGI products add these commands to their startup files.

### **Step 6.** Verify the Release Number.

To verify the release number of the products you have installed, use the `-V` option on any of the compiler commands, as illustrated in the following examples. If you use `-v` instead, you 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 95, use: `pgf95 -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.** To successfully execute these commands, the files `x.f` or `x.c` need not exist.

### **Step 7.** Complete Network Installation Tasks.

Skip this step if you are not using a network installation.

For a network installation, you must run the local installation script on each system on the network where the compilers and tools will be available for use.

If your installation base directory is `/opt/pgi` and the common local directory is `/usr/pgi/shared/7.0`, run the following commands on each system on the network.

```
/opt/pgi/linux86/7.0-4/bin/makelocalrc \  
-x /opt/pgi/linux86/7.0-7 \  
-net /usr/pgi/shared/7.0
```

```
/opt/pgi/linux86-64/7.0-7/bin/makelocalrc \  
-x /opt/pgi/linux86-64/7.0-7 \  
-net /usr/pgi/shared/7.0
```

These commands create a system-dependent file `localrc.machinename` in both the `/opt/pgi/linux86/7.0-7/bin` directory and in `/opt/pgi/linux86-64/7.0-7/bin`. The commands also create the following three directories containing libraries and shared objects specific to the operating system and system libraries on that machine:

`/usr/pgi/shared/7.0/lib`, `/usr/pgi/shared/7.0/liblf`, and `/usr/pgi/shared/7.0/lib64`.

**Note.** The `makelocalrc` command does allow the flexibility to have local directories that have different names on different machines. However, using the same directory on different machines allows users to easily move executables that use PGI-supplied shared libraries between systems.

If you specify `/opt/pgi` as the base directory for installation, the following directory structure is created by the PGI installation script:

Name of directory	Contents
<code>/opt/pgi/linux86/7.0/bin</code>	<i>linux86</i> 32-bit compilers & tools
<code>/opt/pgi/linux86/7.0/lib</code>	<i>linux86</i> 32-bit runtime libraries
<code>/opt/pgi/linux86/7.0/liblf</code>	<i>linux86</i> 32-bit large-file support libs (used by <code>-Mlfs</code> )
<code>/opt/pgi/linux86/7.0/include</code>	<i>linux86</i> 32-bit header files
<code>/opt/pgi/linux86-64/7.0/bin</code>	<i>linux86-64</i> compilers & tools
<code>/opt/pgi/linux86-64/7.0/lib</code>	<i>linux86-64</i> <code>-mcmmodel=small</code> libs

<b>Name of directory</b>	<b>Contents</b>
/opt/pgi/linux86-64/7.0/libso	<i>linux86-64 -fpic</i> shared libraries for <i>-mcmmodel=medium</i> development
/opt/pgi/linux86-64/7.0/include	<i>linux86-64</i> header files
/opt/pgi/linux86/7.0/REDIST /opt/pgi/linux86-64/7.0/REDIST	Re-distributable runtime libraries
/opt/pgi/linux86/7.0/EXAMPLES /opt/pgi/linux86-64/7.0/EXAMPLES	Compiler examples
/opt/pgi/linux86/7.0/doc /opt/pgi/linux86-64/7.0/doc	Documentation
/opt/pgi/linux86/7.0/man /opt/pgi/linux86-64/7.0/man	UNIX-style man pages
/opt/pgi/linux86/7.0/jre /opt/pgi/linux86-64/7.0/jre	JAVA environment for <i>PGDBG</i> and <i>PGPROF</i> graphical user interfaces
/opt/pgi/linux86/7.0/src /opt/pgi/linux86-64/7.0/src	Fortran 90 source files for included modules.

Additionally, a network install creates the following directories:

<b>Name of directory</b>	<b>Contents</b>
/opt/pgi/linux86/7.0/lib-linux86-g	<i>linux86</i> 32-bit <i>libpgc</i> library dependent on the version of <i>glibc</i> installed on each machine
/opt/pgi/linux86/7.0/include-g	<i>linux86</i> 32-bit header files dependent on the version of <i>glibc</i> or <i>gcc</i> installed on each machine
/opt/pgi/linux86-64/7.0/include-g	<i>linux86-64</i> 64-bit header files dependent on the version of <i>glibc</i> or <i>gcc</i> installed on each machine

## 2.3.3 End-user Environment Settings on Linux

Now that you have installed the compilers in, for example, `/opt/pgi`, you must initialize your environment to use the compilers successfully. Assume the license file is in `/opt/pgi/license.dat`, and the `lmgrd` license manager is running.

**Note.** Each user must issue the following sequence of commands to initialize the shell environment before using the PGI compilers and tools.

The following commands make the 32-bit compilers the default.

In `ssh`, use these commands:

```
% setenv PGI /opt/pgi
% setenv MANPATH "$MANPATH":$PGI/linux86/7.0/man
% setenv LM_LICENSE_FILE $PGI/license.dat
% set path = ($PGI/linux86/7.0/bin $path)
```

In `bash`, `sh` or `ksh`, use these commands:

```
% PGI=/opt/pgi; export PGI
% MANPATH=$MANPATH:$PGI/linux86/7.0/man
% export MANPATH
% LM_LICENSE_FILE=$PGI/license.dat
% export LM_LICENSE_FILE
% PATH=$PGI/linux86/7.0/bin:$PATH
% export PATH
```

The following commands make the 64-bit compilers the default.

In `ssh`, use these commands:

```
% setenv PGI /opt/pgi
% setenv MANPATH "$MANPATH":$PGI/linux86-64/7.0/man
% setenv LM_LICENSE_FILE $PGI/license.dat
% set path = ($PGI/linux86-64/7.0/bin $path)
```

In `bash`, `sh` or `ksh`, use these commands:

```
% PGI=/opt/pgi; export PGI
% MANPATH=$MANPATH:$PGI/linux86-64/7.0/man
% export MANPATH
% LM_LICENSE_FILE=$PGI/license.dat
% export LM_LICENSE_FILE
% PATH=$PGI/linux86-64/7.0/bin:$PATH
% export PATH
```

## 2.4 Installing FLEXlm on Linux

If you want the *PGI Workstation* 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 in this section.

*PGI Workstation* trial keys use FLEXlm software license management. Please follow these directions to enable the trial.

**IMPORTANT NOTE:** Release 7.0 includes a newer version of the Macrovision FLEXlm software. The updated *lmgrd* and *pgroupd* daemons must be used in preference to versions shipped with previous releases of the PGI products. You can co-install Release 7.0 with Release 6.X and/or 5.2; and you can use any of these versions of the compilers and tools with the new versions of *lmgrd* and *pgroupd* and a single Release 7.0 license file. Further, if you use this file to start *lmgrd* automatically after a reboot of your system, you must modify your *lmgrd.rc* file in the */etc/rc.d* or */etc/init.d* directory to use the new *lmgrd*.

For example, your *lmgrd.rc* file may look like this one, where *<target>* is replaced appropriately with *linux86* or *linux86-64*.

```
## Path to master daemon lmgrd
# Commented out previous path to 5.2:
#LMGRD=$PGI/<target>/5.2/bin/lmgrd
LMGRD=$PGI/<target>/7.0/bin/lmgrd

## Command to stop lmgrd
#Commented out previous path to 5.2:
#LMUTIL=$PGI/<target>/5.2/bin/lmutil
LMUTIL=$PGI/<target>/7.0/bin/lmutil
```

For complete details on setup and usage of these files, see **Step 4** on page 23.

**Step 1.** Install the PGI software as described in section 2.2 on page 11.

**Step 2.** Obtain and Install License Keys.

Once you have obtained trial or permanent FLEXlm-style license keys, as described in Step 3 of section 2.2, place them in a file named *license.dat* in the installation directory, typically, the */opt/pgi* directory. For example, if you have purchased *PGF77 Workstation* for Linux, the *license.dat* file looks similar to the following:

```

SERVER <hostname> <hostid> 7496
DAEMON pgroupd pgroupd

FEATURE pgf77-linux86 pgroupd 7.000 31-dec-0 1 \
2B9CF0F163159E4ABE32 VENDOR_STRING=107209:16 \
HOSTID=<hostid> ck=49

FEATURE pgprof pgroupd 7.000 31-dec-0 1 \
6BDCE0B12EC19D0909F0 VENDOR_STRING=107209:16 \
HOSTID=<hostid> ck=60

```

In your license file:

- <hostname> and <hostid> must match those you submitted when obtaining your licenses.
- In the feature line component, VENDOR\_STRING=107209:16 includes the Product ID Number (PIN) for this installation, in this example 107209. You have a similar unique PIN number for your installation.

**Note.** Please include your PIN number when sending mail to us regarding technical support for the products you have purchased. This PIN number is also on your order acknowledgement email.

**Step 3.** Access products and initialize your environment.

When the license file is in place, execute the following commands to make the products you have purchased accessible and to initialize your environment for use of FLEXlm. These commands assume that you use the default installation directory: /opt/pgi

In `ssh`, use these commands:

```

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

```

In `bash`, `sh` or `ksh`, use these commands:

```

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

```

You should add these commands to your startup files to ensure that you have access to the PGI products upon future logins.

**Step 4.** Start the license manager daemon.

To complete your installation, you must now start the license manager daemon.

**Important:** If you are evaluating PGI Workstation with trial keys, you are done. You do not need to perform this step.

**Note.** The following paragraph refers to the shell script template for *linux86*. If you have installed only *linux86-64*, please substitute *linux86-64* for *linux86*.

Edit the shell script template `/opt/pgi/linux86/7.0/bin/lmgrd.rc`, where `/opt/pgi` is the default installation directory. If you installed the compilers in a directory other than `/opt/pgi`, substitute the correct installation directory for `'/opt/pgi'` on line 3 of the script. Then save the file and exit the editor.

Issue the following command to start the license server and *pgroupd* 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 described in Step 3 on page 22, and then execute the following two commands:

```
% cp /opt/pgi/linux86/7.0/bin/lmgrd.rc \  
  /etc/init.d/lmgrd  
% ln -s /etc/init.d/lmgrd \  
  /etc/rc.d/rc3.d/S90lmgrd
```

**Note.** There are two values in this example that may be different on your system:

- Your system's default `runlevel` may be something other than `'3'`, the level used in this example. You can run `/sbin/runlevel` to check the system's `runlevel`. If the `runlevel` on your systems is different, then you must set the correct subdirectory; use your system's `runlevel` in place of the `"3"` in the preceding example.
- Your `rc` files may be in a directory other than the one in the example: `/etc/init.d`. If your `rc` files are in a directory such as `/etc/rc.d/init.d`, then substitute that location in the example.

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 /opt/pgi/linux86/7.0/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. For assistance with difficulties related to the installation, send e-mail to [trs@pgroup.com](mailto:trs@pgroup.com).

## 2.5 Installing on Windows

This section describes how to install *PGI Workstation* on a Windows system, including Win64, Win32, SFU, and SUA. It is applicable to both permanent and trial installations.

- For installations on 32-bit x86 systems, the PGI installer installs only the 32-bit versions of the PGI compilers and tools.
- For installations on 64-bit x64 systems running a 64-bit operating system, the PGI installer installs the 64-bit and 32-bit versions of the PGI compilers and tools.

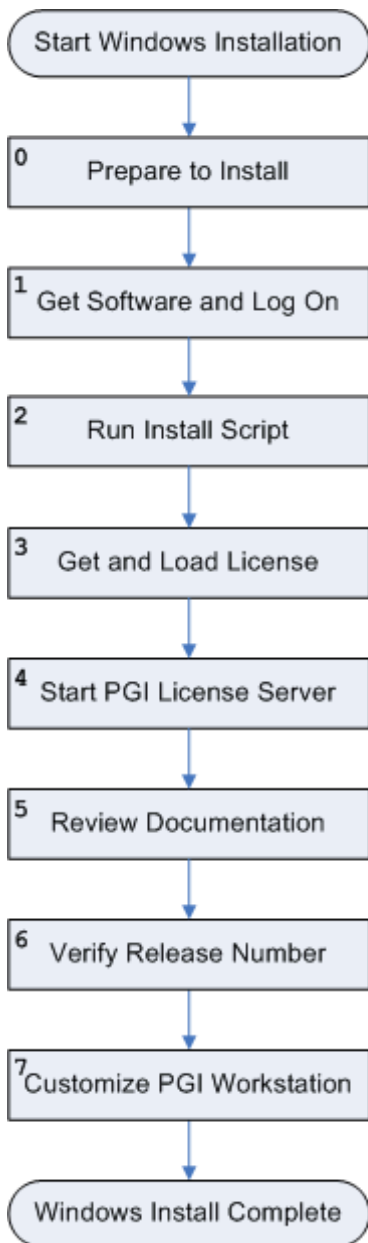
### 2.5.1 Preparing to Install on Windows

*PGI Workstation* for Windows and SFU/SUA includes the *Microsoft Open Tools*, the essential tools and libraries required to compile, link, and execute programs on Windows. No additional Microsoft tools or libraries are needed. The *Microsoft Open Tools* includes a subset of the full Microsoft Platform SDK. *PGI Workstation 7.0* can also compile and link against the Microsoft Platform SDK. For information about how to download the Platform SDK, visit <http://msdn.microsoft.com/platformsdk>.

For SFU and SUA installations, the GNU SDK and GNU utilities must be installed. For SFU, at installation time you must select both the "Interix SDK" and "Interix GNU SDK" components from the Custom Installation. For SUA, you must download the *Utilities and Software Development Kit (SDK) for UNIX-based Applications* from <http://www.microsoft.com> and install the Base SDK and GNU SDK.

In preparation for installing *PGI Workstation* or *PGI Server* on Windows, first review the following overview of the Windows installation process.

**Note.** You must install the software prior to getting the licenses.



- Install 32- or 64- bit?
- Check disk space availability
- If SUA or SFU, verify GNU SDK and GNU utilities are installed
- Know Administrator logon
  
- Locate CD-ROM OR Download from web site
- Log on as administrator.
  
- Access the software
- Follow directions in automatically invoked installation script
  
- Obtain licenses from web site
- Place licenses in `license.dat` file
  
- Open Services dialog and start the PGI License Server
  
- PDF versions are available
- Did I bookmark the online location?
  
- Open PGI Workstation from your desktop and verify the release number
  
- Optional Setup Step:**
- Change screen appearance
- Use `LM_LICENSE_FILE` to override default license location search

## 2.5.2 Installation Steps for Windows

Once you have prepared for the installation, follow these instructions to install the software:

**Step 1.** Have the software available and log on as Administrator.

Administrator privileges are required to install *PGI Workstation*.

If you received this software on a CD-ROM, please skip to step 2. If not, download the software from <http://www.pgroup.com> or another electronic distribution site.

**Step 2.** Start the installation.

If you are installing 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 is automatically invoked and the installation process begins. Choose the software you wish to install, and follow the directions printed to your screen.

If you have configured your system so that CD-ROM autorun is disabled, run the installation executable from the CD. If you obtained *PGI Workstation* from PGI electronically, execute this file on the target machine.

The installation executables are:

<code>pgiws-707.exe</code>	<i>32-bit Windows</i>
<code>pgiwsx64-707.exe</code>	<i>64-bit/32-bit Windows</i>
<code>pgiws-sfu-707.exe</code>	<i>32-bit SFU</i>
<code>pgiws-sua-707.exe</code>	<i>32-bit SUA</i>
<code>pgiwsx64-sua-707.exe</code>	<i>64-bit/32-bit SUA</i>

**Step 3.** Install the license keys.

The *PGI Workstation* compilers and tools on Windows are license-managed and must use the FLEXlm software license management system from Macrovision Software. This system requires that you possess valid license keys for the licensed product.

Further, you must specifically request FLEXlm-style keys when generating license keys at <http://www.pgroup.com/support/keylogin.htm>.

To obtain licensing information, you need the following information:

- The username and password required to connect to the website.
  - For permanent license keys, this information is in your order acknowledgement email.
  - For trial license keys, this is the username (email address) and password you used to download the installation software from the web site.
- The FLEXlm hostid and hostname of the computer on which the software is installed, provided by the installation script. After you have installed the products, you can also obtain your FLEXlm hostid by opening the PGI workstation window and typing these commands:

```
PGI$ cd $PGI
PGI$ cat license.info
```

You see information similar to the following:

For a permanent license, please read the order acknowledgment that you received. Connect to <https://www.pgroup.com/support/keylogin.php> with the username and password in your order acknowledgment.

```
FLEXlm Host ID: 001234A98765
Installation: C:\Program Files\PGI\
PGI Release: 7.0
```

If you want to obtain an evaluation license, go to:

<https://www.pgroup.com/license/evaluation.php>

and use your web-user access codes (email address and password) and the FLEXlm Host ID to generate a trial license.

For retrieval at a later time, the preceding message is also saved to the file `license.info` in the installation directory. For example, if you use the default installation directory of `C:\Program Files\PGI`, then the file location is this:

```
C:\Program Files\PGI\license.info
```

Once you have obtained your trial or permanent license keys using your personalized account on the *pgroup.com* web page, place them in the license file: `license.dat`. In a typical configuration, where `C:\` is the system drive and you installed the software using the default location, this file would be found in:

```
C:\Program Files\PGI\license.dat
```

- If you have never received license keys from PGI before, replace the `license.dat` file created during installation with the PGI Workstation keys.
- If your `license.dat` file already contains keys you have received from PGI, append the PGI Workstation keys to the keys already in this file.

If you are evaluating PGI Workstation with trial keys, skip to Step 5. You do not need to start the license server.

#### **Step 4.** Start the PGI License Server.

The FLEXlm license system requires that a license server be running. The installation process creates a Windows Service called PGI License Server. As soon as a valid `license.dat` file is in place, as described in Step 3, this service can be started.

**Important:** You do not need to start the license server with trial keys.

The PGI License Server is a Windows Service. Therefore, to start it, do this:

- 1.) Open the Services dialog from the Start menu:  
(Start | Control Panel | Administrative Tools | Services).
- 2.) Select “PGI License Server”.
- 3.) Select “Start.”

**Note.** The PGI License Server service starts automatically on system reboot, provided that the `license.dat` file contains valid keys.

#### **Step 5.** Review Documentation.

You can view the online HTML and PDF documentation using any web browser by opening the file:

```
http://www.pgroup.com/resources/docs.htm
```

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

**Step 6.** Verify Release Number.

Verify the release number of the products you have installed.

To do so, open PGI Workstation from your desktop and read the first line displayed in the file.

**Step 7.** Customize PGI workstation.

Optionally, customize the setup, as described in 2.5.3 Customizing the Command Window and in 2.5.5 Using LM\_LICENSE\_FILE.

## 2.5.3 Customizing the 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. This window is 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.

To customize your window, right-click the *PGI Workstation* desktop icon, and select “Properties” from the pop-up menu. In the PGI Workstation Properties dialog box, select the tabs for the features you want to customize, and make the desired modifications.

## 2.5.4 PGI Workstation Directory Structure

The PGI Workstation default installation directory depends on your platform.

On Win32, the default installation directory is

```
%SYSTEMDRIVE%\Program Files\PGI\win32\7.0-7\
```

On Win64 platforms, the default installation directories are

```
%SYSTEMDRIVE%\Program Files\PGI\win64\7.0-7\
```

```
%SYSTEMDRIVE%\Program Files (x86)\PGI\win32\7.0-7\
```

In addition to these two product directories, the Microsoft Open Tools and, optionally, Cygwin, are installed here:

```
%SYSTEMDRIVE%\Program Files\PGI\Microsoft Open Tools 8
```

```
%SYSTEMDRIVE%\cygwin
```

The *Cygwin* directory is not installed with *PGI Workstation* for *SFU* and *SUA*. Instead, any shell can be used. The *PGI Workstation* installation directory structure for *SFU* and *SUA* is similar to the Linux directory layout described in Chapter 2. You should also follow instructions in Section 2.3.3 End-user Environment Settings on Linux.

The following directory structure will be created during the installation on a Win64 system:

Name of directory	Contents
C:\Program Files\PGI\win64\7.0\bin C:\Program Files (x86)\PGI\win32\7.0\bin	<i>PGI Workstation 7.0</i> compilers and tools binaries
C:\Program Files\PGI\win64\7.0\lib C:\Program Files (x86)\PGI\win32\7.0\lib	<i>PGI Workstation 7.0</i> runtime and support libraries
C:\Program Files\PGI\win64\7.0\include C:\Program Files (x86)\PGI\win32\7.0\include	<i>PGI Workstation 7.0</i> header files
C:\Program Files\PGI\win64\7.0\REDIST C:\Program Files (x86)\PGI\win32\7.0\REDIST	Re-distributable runtime libraries
C:\Program Files\PGI\win64\7.0\doc C:\Program Files (x86)\PGI\win32\7.0\doc	Documentation
C:\Program Files\PGI\win64\7.0\man C:\Program Files (x86)\PGI\win32\7.0\man	Man pages for commands
C:\Program Files\PGI\Microsoft Open Tools 8	Microsoft tools
C:\cygwin	Cygwin tools

The following directory structure will be created during the installation on a Win32 system:

Name of directory	Contents
C:\Program Files\PGI\win32\7.0\bin	<i>PGI Workstation 7.0</i> compilers and tools binaries
C:\Program Files\PGI\win32\7.0\lib	<i>PGI Workstation 7.0</i> runtime and support libraries
C:\Program Files\PGI\win32\7.0\include	<i>PGI Workstation 7.0</i> header files
C:\Program Files\PGI\win32\7.0\REDIST	Re-distributable runtime libraries
C:\Program Files\PGI\win32\7.0\doc	Documentation
C:\Program Files\PGI\win32\7.0\man	Command Man pages
C:\Program Files\PGI\Microsoft Open Tools 8	Microsoft tools
C:\cygwin	Cygwin tools

## 2.5.5 Using LM\_LICENSE\_FILE

The system environment variable LM\_LICENSE\_FILE is not required by *PGI Workstation* on Windows but you can use it to override the default location that is searched for the license.dat file.

To use the system environment variable LM\_LICENSE\_FILE, set it to the full path of the license key file. To do this, follow these steps:

1. Open the System Properties dialog (Start | Control Panel | System).
2. Select the 'Advanced' tab.

3. Click the 'Environment Variables' button.
  - If `LM_LICENSE_FILE` is not already an environment variable, create a new system variable for it. Set its value to the full path, including the name of the file, for the license key file.
  - If `LM_LICENSE_FILE` already exists as an environment variable, append the path to the license file to the variable's current value using a semi-colon to separate entries.

## 2.5.6 Common Windows Installation Problems

The most common installation problems on Windows are related to licensing.

To troubleshoot your installation, first check that the `license.dat` file you are using contains valid license keys. Second, check that the PGI License Server, a Windows Service, has been started.

Typical FLEXlm errors encountered may include the following:

- When using a PGI compiler or tool, a Flexible License Manager dialog appears that states 'LICENSE MANAGER PROBLEM: No such feature exists.'  
This message may appear because the `license.dat` file accessed by the FLEXlm License Manager does not contain valid license keys.
- When using a PGI compiler or tool, a Flexible License Manager dialog appears that states 'LICENSE MANAGER PROBLEM: Cannot connect to license server system.'  
This message may appear because the PGI License Server has not been started.
- When starting the PGI License Server, a system message appears that states 'The PGI License Server service on Local Computer started and then stopped. Some services stop automatically if they have no work to do, for example, the Performance Logs and Alerts service.'  
This message may appear because the `license.dat` file accessed by the FLEXlm License Manager does not contain valid license keys.
- A message stating 'LICENSE MANAGER PROBLEM: Failed to checkout license' appears.  
This message may appear because the PGI License Server has not been started.

- By default, on Windows, the license server creates interactive pop-up messages to issue warning and errors. You can use the environment variable `FLEXLM_BATCH` prevent interactive pop-up windows. To do this, set the environment variable `FLEXLM_BATCH` to 1.
- On SFU, if the compilers get segmentation faults or produce core dumps, the problem might be related to Windows Data Execution Prevention. The solution to these errors is to modify the system `boot.ini` file to set `/noexecute=AlwaysOff`; and then reboot. For more information, refer to this good online resource for SFU: <http://www.interopsystems.com>.

## 2.6 Installing on Apple Mac OS X

This section describes how to install *PGI Workstation* on an Apple system. It covers local installs, and is applicable to permanent or trial installations.

*Note:* PGI Workstation for Mac OS X is only supported on Intel Core and Core 2 Duo processors running Mac OS X version 10.4.10. Previous versions of Mac OS may be unstable for 64-bit programs. Using this release requires that Apple Xcode 2.4.1 be installed. Xcode is available from <http://developer.apple.com>.

For installations on 32-bit *x86* systems, the PGI installation process installs only the *osx86* versions of the PGI compilers and tools. For installations on 64-bit *x64* systems running an *osx86-64* execution and development environment, the PGI installation process installs the *osx86-64* version of the PGI compilers and tools. Additionally, if the 32-bit *gcc* development package is already installed on the system, the 32-bit *osx86* tools will be installed on a 64-bit *x64* system.

The 32-bit and 64-bit compilers, tools, and supporting components have the same command names, and the environment you target by default, either *osx86-64* or *osx86*, depends on the version of the compiler that comes first in your path settings.

The default installation base directory is `/opt/pgi`.

## 2.6.1 Preparing to Install on Apple Mac OS X

To prepare for the installation:

- Verify you have enough free disk space.
  - On the osx86 platform, PGI installation requires 250 MB of free disk space.
  - On the osx86-64 platform, PGI installation requires 500 MB of free disk space.
- Verify that Xcode 2.4.1 is installed.
  - If you know how to run Xcode, start Xcode and click About Xcode to verify the version is 2.4.1.
  - If you do not know how to run Xcode or are uncertain if it is installed on your system, do the following:
    - 1.) From the Apple Menu, select About This Mac.
    - 2.) Click More Info.
    - 3.) Select System profiler | Software | Applications.
    - 4.) Scroll through the alphabetical list and verify Xcode is in it.
    - 5.) Verify the version is 2.4.1.

**Note.** PGI Workstation for Mac OS requires the Xcode application, which provides several components of the tool chain, including the system assembler, linker, and run-time libraries. However, PGI Workstation runs in a Terminal, not in the Xcode IDE, and the PGDBG debugger and PGPROF profiler open Java windows.

## 2.6.2 Installation Steps for Mac OS X

Once you have prepared for the installation, follow these instructions to install the software:

**Step 1.** Access the installation package.

- If you received this software on a CD-ROM, place the CD-ROM in the CD drive and wait for the disk to mount of your system.

- If you do not have a CD-ROM, download the software from <http://www.pgroup.com> or another electronic distribution site. The file you download appears on your system as `pgiosx-707.dmg`. Open this file to mount it.

**Step 2.** Install the software.

Click on `PGI Workstation.pkg`, which is part of the mounted disk. Follow the installation instructions.

- When you see the initial system check dialog, click continue to allow the install script to check that your system has the required components for installing the software, such as Xcode 2.4.1 and gcc.
- While you must select the hard drive on your system for the installation, the install program does not allow you to select an installation directory other than the default one: `/opt/pgi`.
- After the software is installed, the install script performs some system-specific customization and then initializes the licensing, as described in step 3.

**Step 3.** All of the PGI compilers and tools are license-managed. PGI Workstation and PGI Server products use the FLEXlm license daemon.

To obtain licensing information, you need the following information:

- The username and password required to connect to the website.
  - For permanent license keys, this information is in your order acknowledgement email.
  - For trial license keys, this is the username and password you used to download the installation software from the web site.
- The FLEXlm `hostid` and `hostname` of the computer on which the software is installed, provided by the installation script. You can also obtain your FLEXlm `hostid` by using the following command after you have installed the products and initialized the environment variables:

```
% lmutil lmhostid
```

Once you have obtained your trial or permanent license keys using your personalized account on the [pgroup.com](http://www.pgroup.com) web page, place them in the file `/opt/pgi/license.dat`, or substitute the appropriate installation directory path if you have not installed in the default `/opt/pgi` directory.

**Step 4.** You can view the online HTML and PDF documentation using any

web browser by opening the file:

```
file:/opt/pgi/osx86/7.0/doc/index.htm
```

or

```
file:/opt/pgi/osx86-64/7.0/doc/index.htm
```

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

### **Step 5. Make Tools Accessible**

To use the compilers and tools, execute the following commands.

#### **For x64 osx86-64:**

To use the x64 *osx86-64* version of the compilers and tools, execute the following commands.

*Note.* In these commands, the installation directory is the default: `/opt/pgi`. You currently cannot change this directory.

For bash, zsh or ksh:

```
% PATH=/opt/pgi/osx86-64/7.0/bin:$PATH
% export PATH
% MANPATH=$MANPATH:/opt/pgi/osx86-64/7.0/man
% export MANPATH
```

For csh:

```
% set path = (/opt/pgi/osx86-64/7.0/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/osx86-64/7.0/man
```

#### **For osx86:**

To install only the *osx86* version of the compilers and tools, or to target *osx86* as the default, use the setup similar to the previous one, changing the path settings:

For bash, zsh or ksh:

```
% PATH=/opt/pgi/osx86/7.0/bin:$PATH
% export PATH
% MANPATH=$MANPATH:/opt/pgi/osx86/7.0/man
% export MANPATH
```

For csh:

```
% set path = (/opt/pgi/osx86/7.0/bin $path)
```

```
% setenv MANPATH "$MANPATH":/opt/pgi/osx86/7.0/man
```

To ensure they have access to the PGI compilers and tools upon future logins, we recommend that all users of the PGI products add these commands to their startup files.

### **Step 6** – Verify Release Number.

To verify the release number of the products you have installed, use the `-V` option on any of the compiler commands, as illustrated in the following examples. If you use `-v` instead, you 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 95, use: `pgf95 -V x.f`
- For ANSI C, use: `pgcc -V x.c`

**Note.** To successfully execute these commands, the files `x.f` or `x.c` need not exist.

If you specify `/opt/pgi` as the base directory for installation, the following directory structure is created during the PGI installation process:

<b>Name of directory</b>	<b>Contents</b>
<code>/opt/pgi/osx86/7.0/bin</code>	<i>osx86</i> 32-bit compilers & tools
<code>/opt/pgi/osx86/7.0/lib</code>	<i>osx86</i> 32-bit runtime libraries
<code>/opt/pgi/osx86/7.0/include</code>	<i>osx86</i> 32-bit header files
<code>/opt/pgi/osx86/7.0/doc</code>	Documentation
<code>/opt/pgi/osx86/7.0/man</code>	UNIX-style man pages
<code>/opt/pgi/osx86/7.0/src</code>	Fortran 90 source files for included modules.
<code>/opt/pgi/osx86-64/7.0/bin</code>	<i>osx86-64</i> compilers & tools
<code>/opt/pgi/osx86-64/7.0/lib</code>	<i>osx86-64 -mcmodel=small</i> libs
<code>/opt/pgi/osx86-64/7.0/include</code>	<i>osx86-64</i> header files
<code>/opt/pgi/osx86-64/7.0/doc</code>	Documentation
<code>/opt/pgi/osx86-64/7.0/bin</code>	<i>osx86-64</i> compilers & tools
<code>/opt/pgi/osx86-64/7.0/lib</code>	<i>osx86-64 -mcmodel=small</i> libs
<code>/opt/pgi/osx86-64/7.0/include</code>	<i>osx86-64</i> header files
<code>/opt/pgi/osx86-64/7.0/man</code>	UNIX-style man pages

Name of directory	Contents
/opt/pgi/osx86-64/7.0/src	Fortran 90 source files for included modules.

### 2.6.3 End-user Environment Settings on Mac OS X

Now that you have installed the compilers in, for example, /opt/pgi, you must initialize your environment to use the compilers successfully. Assume the license file is in /opt/pgi/license.dat, and the lmgrd license manager is running. Each user must issue the following sequence of commands to initialize the shell environment before using the PGI compilers and tools.

The following commands make the 32-bit compilers the default.

In bash, zsh or ksh, use these commands:

```
% PGI=/opt/pgi; export PGI
% MANPATH=$MANPATH:$PGI/osx86/7.0/man
% export MANPATH
% LM_LICENSE_FILE=$PGI/license.dat
% export LM_LICENSE_FILE
% PATH=$PGI/osx86/7.0/bin:$PATH
% export PATH
```

In csh, use these commands:

```
% setenv PGI /opt/pgi
% setenv MANPATH "$MANPATH":$PGI/osx86/7.0/man
% setenv LM_LICENSE_FILE $PGI/license.dat
% set path = ($PGI/osx86/7.0/bin $path)
```

To make the 64-bit compilers the default, use these commands:

In bash, zsh or ksh, use these commands:

```
% PGI=/opt/pgi; export PGI
% MANPATH=$MANPATH:$PGI/osx86-64/7.0/man
% export MANPATH
% LM_LICENSE_FILE=$PGI/license.dat
% export LM_LICENSE_FILE
% PATH=$PGI/osx86-64/7.0/bin:$PATH
% export PATH
```

In csh, use these commands:

```
% setenv PGI /opt/pgi
% setenv MANPATH "$MANPATH":$PGI/osx86-64/7.0/man
% setenv LM_LICENSE_FILE $PGI/license.dat
% set path = ($PGI/osx86-64/7.0/bin $path)
```

## 2.7 Installing FLEXlm on Mac OS X

If you are using PGI Workstation compilers or you are installing the multi-user PGI Server product, you must use the provided FLEXlm software license management system from Macrovision Software as outlined here.

To install FLEXlm on Mac OS X, follow these steps.

Step 1 - Install the PGI software as described in section 2.6.

Step 2 - Create or update license.dat file.

Once you have obtained trial or permanent FLEXlm-style license keys, as described in Step 3 of section 2.6, place them in a file named license.dat in the /opt/pgi directory.

For example, the license.dat file looks similar to the following:

```
SERVER <hostname> <hostid> 7496
DAEMON pgroupd pgroupd
FEATURE pgf90-osx86 pgroupd 7.000 31-dec-0 1 \
2B9CF0F163159E4ABE32 VENDOR_STRING=507209:16 \
HOSTID=<hostid> ck=49
FEATURE pgprof pgroupd 7.000 31-dec-0 1 \
6BDCE0B12EC19D0909F0 VENDOR_STRING=507209:16 \
HOSTID=<hostid> ck=60
```

...

In your license file, <hostname> and <hostid> must match those you submitted when obtaining your licenses.

In the feature line component, VENDOR\_STRING=507209:16 contains the Product ID Number (PIN) for this installation. You have a similar unique PIN for your installation.

NOTE. Please include your PIN when sending mail to us regarding technical support for the products you have purchased. This PIN is also on your order acknowledgement email.

Step 3 - Initialize environment so you can access software.

When the license file is in place, execute the following commands to make the products you have purchased accessible, and to initialize your environment for use of FLEXlm. These commands assume that you use the default installation directory: /opt/pgi

In bash, zsh or ksh, use these commands:

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

In csh, use these commands:

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

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

If you are evaluating PGI Workstation with trial keys, you are done. You do not need to start the license manager daemon.

Step 4 - Start license manager daemon.

You must now configure and start the license manager daemon.

Login in and become root:

```
% su root
<password>
```

Create directory /Library/StartupItems/PGI:

```
% mkdir /Library/StartupItems/PGI
```

Copy the PGI license configuration files:

```
% cp /opt/pgi/PGI /Library/StartupItems/PGI/PGI
% cp /opt/pgi/StartupParameters.plist \
/Library/StartupItems/PGI/StartupParameters.plist
```

Start the license server:

```
% cd /Library/StartupItems/PGI
% ./PGI start
```

The license server should now be running. It will restart automatically when you reboot.

Installation of your FLEXlm-style licensing of our products is now complete. For assistance with difficulties related to the installation, send e-mail to [trs@pgroup.com](mailto:trs@pgroup.com).

# 3

## PGI Release 7.0 Release Notes

---

This document describes changes between previous releases and Release 7.0 of the PGI compilers, as well as late-breaking information not included in the current printing of the *PGI User's Guide*. There are nine platforms supported by the *PGI Workstation* and *PGI Server* compilers and tools:

- *32-bit Linux* – supported on *32-bit Linux operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit Linux* – includes all features and capabilities of the 32-bit Linux version, and is also supported on *64-bit Linux operating systems* running an *x64* compatible processor.
- *32-bit Windows* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit Windows* – includes all features and capabilities of the 32-bit Windows version, and is also supported on *64-bit Windows operating systems* running an *x64* compatible processor.
- *32-bit SFU* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *32-bit SUA* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit SUA* – includes all features and capabilities of the 32-bit *SUA* version, and is also supported on *64-bit Windows operating systems* running an *x64* compatible processor.
- *32-bit Apple Mac OS X* – supported on 32-bit Apple Mac operating systems running on either a 32-bit or 64-bit Intel-based Mac system.

- *64-bit Apple Mac OS X* – supported on 64-bit Apple Max operating systems running on a 64-bit Intel-based Mac system.

These release notes distinguish these versions where necessary.

## 3.1 PGI Release 7.0 Contents

Release 7.0 of PGI Workstation and PGI Server includes the following components:

- *PGF95* native OpenMP and auto-parallelizing Fortran 95 compiler.
- *PGF77* native OpenMP and auto-parallelizing FORTRAN 77 compiler.
- *PGHPPF* data parallel High Performance Fortran compiler.  
Note: *PGHPPF is not supported in Windows environments.*
- *PGCC* native OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- *PGC++* native OpenMP and auto-parallelizing ANSI C++ compiler.
- *PGPROF* multi-thread and OpenMP graphical profiler.
- *PGDBG* multi-thread and OpenMP graphical debugger.
- Complete online documentation in PDF, HTML and UNIX `man` page formats.
- A UNIX-like shell environment for *Win32* and *Win64* environments.

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

## 3.2 Supported Systems

### 3.2.1 Supported Processors

The following table contains the processors on which Release 7.0 of the PGI compilers and tools is supported. The `-tp <target>` command-line option generates executables that utilize features and optimizations specific to a given CPU and operating system environment. Compilers included in a 64-bit/32-bit PGI installation can produce executables targeted to any 64-bit or 32-bit target, including cross-targeting for AMD and Intel 64-bit AMD64 compatible CPUs.

In addition to the capability to generate binaries optimized for specific AMD or Intel processors, the PGI 7.0 compilers can produce PGI Unified Binary object or executable files containing code streams fully optimized and supported for both AMD and Intel x64 CPUs. The `-tp <target>` option must be used to produce unified binary files.

The table also includes the CPUs available and supported in multi-core versions.

Processors Supported by PGI 7.0								
Brand	CPU	Cores	<target>	Memory Address	Floating Point HW			
					x87	SSE 1	SSE 2	SSE 3
AMD	Opteron/Quadcore	4	barcelona-64	64-bit	Yes	Yes	Yes	Yes
AMD	Opteron/Quadcore	4	barcelona	32-bit	Yes	Yes	Yes	Yes
AMD	Opteron/Athlon64	2	k8-32	32-bit	Yes	Yes	Yes	No
AMD	Opteron/Athlon64	2	k8-32	32-bit	Yes	Yes	Yes	No
AMD	Opteron Rev E/F Turion /Athlon64	2	k8-64e	64-bit	Yes	Yes	Yes	Yes
AMD	Opteron Rev E/F	2	k8-32	32-bit	Yes	Yes	Yes	No
AMD	Turion64 Turion /Athlon64	1	k8-64e	64-bit	Yes	Yes	Yes	Yes
AMD	Turion64	1	k8-32	32-bit	Yes	Yes	Yes	No
Intel	Core 2	2	core2	32-bit	Yes	Yes	Yes	Yes
Intel	Core 2	2	core2-64	64-bit	Yes	Yes	Yes	Yes
Intel	P4/Xeon EM64T	2	p7-64	64-bit	Yes	Yes	Yes	Yes
Intel	P4/Xeon EM64T	2	p7	32-bit	Yes	Yes	Yes	Yes
Intel	Xeon/Pentium4	1	p7	32-bit	Yes	Yes	Yes	No
AMD	Athlon XP/MP	1	athlonxp	32-bit	Yes	Yes	No	No
Intel	Pentium III	1	piii	32-bit	Yes	Yes	No	No
AMD	Athlon	1	athlon	32-bit	Yes	No	No	No
AMD	K6	1	k6	32-bit	Yes	No	No	No
Intel	Pentium II	1	p6	32-bit	Yes	No	No	No
Other	Other x86	No	p5 or px	32-bit	Yes	No	No	No

## 3.2.2 Supported Operating Systems

The table lists the operating systems, and their equivalents, that Release 7.0 of the PGI compilers and tools supports.

To determine if Release 7.0 will install and run under a Linux equivalent version, such as Mandrake\*, Debian\*, Gentoo\*, and so on, check the table for a supported system with the same glibc and gcc versions. Version differences in other operating system components can cause difficulties, but often these can be overcome with minor adjustments to the PGI software installation or operating system environment.

- Newer distributions of the Linux and Windows operating systems include support for x64 compatible processors and are designated 64-bit in the table. These are the only distributions on which the 64-bit versions of the PGI compilers and tools will fully install.
- If you attempt to install the 64-bit/32-bit Linux version on a system running a 32-bit Linux distribution, only the 32-bit PGI compilers and tools are installed.
- If you attempt to install the 64-bit Windows version on a system running 32-bit Windows, the installation fails.

Most newer Linux distributions support the *Native POSIX Threads Library* (NPTL), a new threads library that can be utilized in place of the *libpthread* library available in earlier versions of Linux. Distributions that include NPTL are designated in the table. Parallel executables generated using the *OpenMP* and auto-parallelization features of the PGI compilers automatically make use of NPTL on distributions when it is available. In addition, the *PGDBG* debugger is capable of debugging executables built using either NPTL or earlier thread library implementations.

Multi-processor AMD Opteron processor-based servers use a *NUMA* (Non-Uniform Memory Access) architecture in which the memory latency from a given processor to a given portion of memory can vary. Newer Linux distributions, including SuSE 9.2/9.3/10.0 and SLES 9/10, include NUMA libraries that can be leveraged by a compiler and associated runtime libraries to optimize placement of data in memory.

In the table headings, HT = hyper-threading, NPTL = Native POSIX Threads Library, and NUMA = Non-Uniform Memory Access. For more information on these terms, see Terms and Definitions on page 2.

Operating Systems and Features Supported in PGI 7.0									
Distribution	Type	64-bit	HT	pgC++	pgdbg	NPTL	NUMA	glibc	GCC
<b>RHEL 4.0</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.4	3.4.3
<b>Fedora C-6</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.5	4.1.1
<b>Fedora C-5</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
<b>Fedora C-4</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.5	4.0.0
<b>Fedora C-3</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.4.2
<b>Fedora C-2</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.3.3
<b>SuSE 10.2</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.5	4.1.0
<b>SuSE 10.1</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
<b>SuSE 10.0</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.3.5	4.0.2
<b>SuSE 9.3</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.3.4	3.3.5
<b>SuSE 9.2</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.3.3	3.3.4
<b>SLES 10</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
<b>SLES 9</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	No	Yes	2.3.3	3.3.3
<b>SuSE 9.1</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.3.3
<b>RHEL 3.0</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.2	3.2.3
<b>SuSE 9.0</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	No	No	2.3.2	3.3.1
<b>SuSE 8.2</b>	<i>Linux</i>	Yes	Yes	Yes	Yes	No	No	2.3.2	3.3
<b>RedHat 9.0</b>	<i>Linux</i>	No	No	Yes	Yes	Yes	No	2.3.2	3.2.2
<b>Red Hat 8.0</b>	<i>Linux</i>	No	No	Yes	Yes	No	No	2.2.93	3.2
<b>SLES8 SP2</b>	<i>Linux</i>	Poor	Yes	Yes	Yes	No	No	2.2.5	3.2.2
<b>SuSE 8.1</b>	<i>Linux</i>	Poor	Yes	Yes	Yes	No	No	2.2.5	3.2.2
<b>SuSE 8.0</b>	<i>Linux</i>	No	No	Yes	Yes	No	No	2.2.5	2.96
<b>Red Hat 7.3</b>	<i>Linux</i>	No	No	Yes	Yes	No	No	2.2.5	2.96
<b>Microsoft Windows</b>	<i>XP</i>	No	Yes	Yes	Yes	NA	Yes	NA	NA
	<i>2003</i>	No	No	Yes	Yes	NA	Yes	NA	NA
	<i>2000</i>	No	No	Yes	Yes	NA	Yes	NA	NA
	<i>XP x64</i>	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	<i>2003 x64</i>	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	<i>SFU</i>	No	Yes	Yes	Yes	NA	Yes	SFU	3.3
	<i>SUA x86</i>	No	Yes	Yes	Yes	NA	Yes	SUA	3.3
	<i>SUA x64</i>	Yes	Yes	Yes	Yes	NA	Yes	SUA	3.3
<b>Apple Mac OS X</b>	<i>Vista</i>	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	<i>Core 2</i>	No	No	No	Yes	NA	NA	NA	4.0.1
<b>Apple Mac OS X</b>	<i>Core 2 Duo</i>	No	No	No	Yes	NA	NA	NA	4.0.1

*Note.* <http://www.pgroup.com/support/install.htm> lists any new Linux, Apple or Windows distributions that may be explicitly supported by the PGI compilers. If your operating system is newer than any of those listed in the preceding table, the installation may still be successful.

### 3.2.3 New System Calls

Release 7.0 of the PGI run-time libraries makes use of Linux system libraries to implement features, such as OpenMP and Fortran I/O. The PGI run-time libraries make use of several additional system library routines in this release.

On 64-bit Linux systems, the additional system library routines are:

aio_error	pthread_attr_init
aio_read	pthread_mutex_init
aio_return	pthread_mutex_lock
aio_suspend	pthread_mutex_unlock
aio_write	setrlimit
calloc	sleep
getrlimit	

On 32-bit Linux systems, the additional system library routines are:

aio_error	calloc
aio_read	getrlimit
aio_return	pthread_attr_init
aio_suspend	setrlimit
aio_write	sleep

## 3.3 New or Modified Compiler Features

Following are the new features of Release 7.0 of the PGI compilers and tools as compared to prior releases.

- *PGI Unified Binaries* – When using the `-tp x64` target option, PGI compilers produce PGI Unified Binary programs containing code streams fully optimized and supported for both AMD and Intel x64 CPUs.
- *Multiple PGI Unified Binary Targets* – The `-tp` switch now supports a comma-separated list of targets allowing programs to be optimized for more than two 64-bit targets. Unified Binary directives and pragmas may be applied to functions, subroutines, or whole files, directing the compiler to generate unified binary code optimized for one or more targets.

- *PGI Unified Binary Culling* – Only those functions and subroutines where the target affects the generated code will have unique binary images. This change results in a code-size savings of 10-90% compared to PGI 6.2, without any adverse affect on performance.
- *Fortran 2003 ISO\_C\_BINDING* – The ISO\_C\_BINDING module is partially implemented. The BIND attribute is supported for derived types and constant kind definitions are provided that map to C types. For procedures, the VALUE and BIND attributes are supported, as well as the BIND attribute for global data. The procedure C\_LOC is supported, which returns the C address of an object. Other procedures in ISO\_C\_BINDING, such as C\_ASSOCIATED, are not yet implemented.
- *Fortran 2003 Allocatable Regularization* – Fortran 2003 allocatable regularization is implemented in PGF95 and is always enabled. These changes allow allocatable arrays to be passed as dummy arguments, to be returned from functions, and to be components of derived types.
- *Fortran 2003 Allocatable Array Assignment* – Fortran 2003 allocatable array assignment is available in PGF95. The default is to use the Fortran 95 assignment semantics; however, the option `-Mallocatable=03` enables the Fortran 2003 assignment semantics.
- *Fortran 2003 Asynchronous Input/Output* – Fortran 2003 asynchronous I/O is partially implemented in PGF77 and PGF95 compilers. For external files opened with `ASYNCHRONOUS='YES` in the OPEN statement, asynchronous I/O is allowed for external files opened with `ASYNCHRONOUS='YES'` in the OPEN statement. Asynchronous I/O operations are indicated by `ASYNCHRONOUS='YES'` in READ and WRITE statements. The compilers do not implement the ASYNCHRONOUS attribute or ASYNCHRONOUS statement.
- *Fortran 2003 Stream Input/Output* – Fortran 2003 Stream access I/O is implemented.
- *Fortran lib3f* – `sleep3f` and `sleepqq3f` for Win32 are implemented, matching other platforms.
- *ANSI C99* – The default value of `__STDC_VERSION__` is now defined as 199901L. Designated initializers and compound literals are implemented.
- *C Preprocessor* – Files with an upper-case `.S` suffix are treated as assembler source that must be preprocessed before passing the file to the assembler. A macro is not expanded by the preprocessor if it is preceded by '\$', because a 'C' identifier can contain a '\$'. When the source being preprocessed is an assembly file and the input file suffix is `.s` or `.S`, then '\$' is excluded from an identifier in the preprocessor.

- *C++ `__restrict` type qualifier* – The `__restrict` type qualifier indicates that all data accessed through the pointer will only be accessed through that pointer for the scope of the restricted pointer. The `__restrict` type qualifier may appear in the same context as a `const` type qualifier. This qualifier allows the compiler to perform additional optimizations.
- *Expanded gcc compatibility* – PGC++ now supports extended asm, the inline intrinsic libraries, GNU statement expressions and these redefinable predefined macros: `__PGIC__`, `__PGIC_MINOR__`, and `__PGIC_PATCHLEVEL__`. On Linux, now use the `.ctor` and `.dtor` sections for constructors and destructors instead of the `.init` and `.fini` sections.
- *Expanded `-pg` compatibility* – Programs that are compiled and linked with the `-pg` option now use the environment variable `GMON_OUT_PREFIX` to specify the name of the output file. The default name is `gmon.out`.
- *Expanded cl compatibility* – On Windows, `__STDC__` is defined to be compatible with Microsoft CL, that is, `-D__STDC__=0`. Support now exists for the integral constant suffixes `i64`, `ui64`, and `__declspec(align(n))`, where `n` can be 2, 4, 8, 16, or 32. Also, the GNU `__extension__` keyword is added on Windows.
- *OpenMP 3.0 Stack Size* – The OpenMP 3.0 `OMP_STACK_SIZE` environment variable and the `stack-size` API are supported to control the size of the stack for newly created threads. API functions `omp_set_stack_size` and `omp_get_stack_size` are implemented.
- *OpenMP 3.0 Wait Policy* – The OpenMP 3.0 `OMP_WAIT_POLICY` environment variable affects the behavior of idle threads, in particular whether they spin or sleep when idle. Unemployed threads during a serial region can either busy wait using the barrier (`ACTIVE`) or politely wait using a mutex (`PASSIVE`). The choice is set by `OMP_WAIT_POLICY`. The default is `ACTIVE`.
- *Support for SSSE3, SSE4a, and ABM* – Support for Intel Core 2 SSSE3 instructions and AMD SSE4a and ABM instruction is now incorporated into the C and C++ inline intrinsics packages. These instructions are also available through extended asm and, on Windows, in the `as32` and `as64` assemblers. The Windows assemblers support GNU-style syntax. On Linux, these instructions are available through the platform `binutils` `as`.

- *Improved Performance* – Code-generation optimizations include propagation and elimination of sign-extension, removal of useless zero extension, dead-store elimination, use of two-byte returns when branching to a return, optimized 32- and 64-bit shift operations, and better allocation of registers.
- *Fortran/C/C++ Performance* – High-level optimizations include serially-nested redundant conditional elimination, better analysis that creates more opportunities for vectorization and auto-parallelization, LRE enabled with pointer variables, extending the range of local common-subexpression elimination, adding LRE to the C++ compiler, and using an exit heuristic where certain function names are considered to be invoked only on a rare path and a branch to a target that has a low probability.
- *Auto-parallelization for multi-core processors* – Enhanced heuristic for auto-parallelization gives less consideration to loops with few iterations and more consideration to loops which contain nested loops when generating serial alt-code for auto-parallelized loops.
- *Enhanced vectorization* – Further tuning of the vectorizer provides additional idiom recognition and vectorization of loops with type conversions. For 64-bit targets, `-fast` now includes SSE code generation and vectorization.
- *ACML 3.6* – This latest edition of the AMD Core Math Library is bundled with the PGI 7.0 compilers on Linux and Windows.
- *Expanded OS support* – Added support for Fedora Core 6 and SuSE 10.2. On Windows, support is added for Vista and for native compilation and execution under SFU and SUA.
- *Visual Studio 2005 SP1* – Added support for Visual Studio 2005 SP1.
- *Environment Modules* – For users of the environment modules package, a script is included to set up the appropriate module files.
- *Quad-Core AMD Opteron Processor support* – The new options `-tp barcelona` and `-tp barcelona-64` implement code generation and tuning for the 32- and 64-bit AMD Barcelona processors.
- *Apple Core 2 and Core 2 Duo support*– PGI Workstation for *Mac OS* is available from <http://www.pgroup.com> as a preview release.

## 3.4 Compiler Options

### 3.4.1 Getting Started

By default, the PGI 7.0 compilers generate code that is optimized for the type of processor on which compilation is performed, the compilation host. If you are unfamiliar with the PGI compilers and tools, a good option to use by default is *-fast* or *-fastsse*.

#### 3.4.1.1 Using *-fast*, *-fastsse*, and Other Performance-Enhancing Options

These options create a generally optimal set of flags for targets that support SSE/SSE2 capability. These options incorporate optimization options to enable use of vector streaming SIMD (SSE/SSE2) instructions for 64-bit targets. They enable vectorization with SSE instructions, cache alignment, and flushz.

**Note.** The contents of the *-fast* and *-fastsse* options are host-dependent.

- *-fast* and *-fastsse* typically include these options:

<i>-O2</i>	Specifies a code optimization level of 2.
<i>-Munroll=c:1</i>	Unrolls loops, executing multiple instances of the loop during each iteration.
<i>-Mnoframe</i>	Indicates to not generate code to set up a stack frame
<i>-Mlre.</i>	Indicates loop-carried redundancy elimination

- For 64-bit targets, *-fast* and *-fastsse* typically also include these options:

<i>-Mvect=sse</i>	Generates SSE instructions
<i>-Mscalarsse</i>	Generates scalar SSE code with xmm registers; implies <i>-Mflushz</i>
<i>-Mcache_align</i>	Aligns long objects on cache-line boundaries
<i>-Mflushz</i>	Sets SSE to flush-to-zero mode
<i>-M[no]vect</i>	Controls automatic vector pipelining.

**Note.** For best performance on processors that support SSE instructions, use the *PGF95* compiler, even for FORTRAN 77 code, and the *-fastsse* option.

In addition to *-fast* and *-fastsse*, the *-Mipa=fast* option for inter-procedural analysis and optimization can improve performance. You may also be able to obtain further performance improvements by experimenting with the individual *-Mpgflag* options detailed in the *PGI User's Guide*, such as *-Mvect*, *-Munroll*, *-Minline*, *-Mconcur*, *-Mphi/-Mpfo*, and so on. However, increased speeds using these options are typically application- and system-dependent. It is important to time your application carefully when using these options to ensure no performance degradations occur.

### 3.4.2 New or Modified Compiler Options

Unknown options are now treated as errors instead of warnings. This change makes it a compiler error to pass switches that are not known to the compiler; however, you can use the new switch *-noswitcherror* to issue warnings instead of errors for unknown switches.

The following compiler options have been added or modified in PGI 7.0:

- *-fast* – For 64-bit targets, *-fast* now includes the same options as the *-fastsse* option. The new *-fast* enables vectorization with SEE instructions, cache alignment, and flushz. The old *-fast* behavior is available as *-nfast*.
- *-fast* – The C/C++ compilers enable *-Mautoinline* with *-fast* or *-fastsse*.
- *-tp* – The *-tp* switch now allows a list of comma-separated targets. Previous releases allowed just one. If multiple targets are given, a unified binary is generated with code optimized for each of the targets.
- *-O4* – A new optimization level, *-O4*, enables hoisting of guarded invariant floating point expressions.
- *-d[D|I|M|N]* – The *-d* option prints additional information from the preprocessor:
  - dD*     Print macros and values from source files.
  - dI*     Print include file names.
  - dM*     Print macros and values, including predefined and command-line macros.
  - dN*     Print macro names from source files.
- *-soname library.so* – The compiler recognizes the *-soname* option and passes it to the linker. (Linux only.)
- *-Mdll* – The *-Mdll* option implies *-D\_DLL*, which defines the pre-processor symbol *\_DLL*. . (Windows only.)

- `—flagcheck` – This options returns zero status if all flags are ok.
- `–E –pgcc –E` now preprocesses .h files.
- `–M[no]dse` – Enable [disable] a dead-store elimination phase that is useful for C++ programs that rely on extensive use of inline function calls for performance. The default is `–Mnodse`.
- `—keeplnk` – If the compiler generates a temporary indirect file for long linker command, the `—keeplnk` option instructs the compiler to preserve the temporary file instead of deleting it. (Windows only.)
- `–Mmakeimplib` – Using `–Mmakeimplib` without `–def:deffile` passes the `–def` switch to the librarian without a deffile. (Windows only.)
- `–Mallocatable=[95|03]` – The `–Mallocatable` option controls how the compiler treats assignment of allocatables. The default behavior is to use Fortran 95 semantics; the `03` option instructs the compiler to use Fortran 2003 semantics.
- `–cyglibs` and `–mslib` – These switches for older releases of PGI Workstation for Win32 are no longer supported.
- `–noswitcherror` – Issue warnings instead of errors for unknown switches. This behavior can be configured in the `siterc` file with `set NOSWITCHERROR=1`.
- `–[no]compress_names` – Compress C++ mangled names to fit into 1024 characters. The current default is `--no_compress_names`. All C++ user code must be recompiled when using this switch.
- `–M[no]asmkeyword` – This switch is useful only if the target device is a Pentium 3 or older CPU type (`–tp piii|p6|k7|athlon|athlonxp|px`). The current default is to support gcc's extended asm, where the syntax of extended asm includes asm strings.
- `–fPIC` – If you use `–fPIC` with `–lacml` or with `–lacml_mp`, you must add the `–lacml_mv` library to the link line.

## 3.5 PGI Workstation 7.0 for Windows

*PGI Workstation 7.0* for *Windows* supports most of the features of the 32- and 64-bit versions for *linux86* and *linux86-64* environments. The product optionally provides a familiar and somewhat compatible development environment for Linux or RISC/UNIX users porting to or developing programs for Windows systems. Except where noted in the *PGI User's Guide* and these release notes, the PGI compilers and tools on *Windows* function identically to their *Linux* counterparts.

*PGI Workstation* includes the *Microsoft Open Tools* tools, libraries, and include files. *Open Tools* includes C header files that use C++ style comments. Use the *-B* option to accept C++ style comments in C programs. Also, some Microsoft header files generate warnings about things such as multiple definitions of types. In most cases these warnings may be safely ignored.

### 3.5.1 The Windows Command Environment

A UNIX-like shell environment, *Cygwin*, is bundled with *PGI Workstation 7.0* for *Windows* to provide a familiar development environment for Linux or UNIX users. *PGI Workstation* for SFU and SUA does not include *Cygwin*; it runs in the SFU/SUA shell environment.

After installation, a double-left-click on the *PGI Workstation* icon on your desktop will launch a *Cygwin* `bash` shell command window with pre-initialized environment settings. Many familiar UNIX commands are available, such as `vi`, `sed`, `grep`, `awk`, `make`, and so on. If you are unfamiliar with the `bash` shell, refer to the user's guide included with the online HTML documentation.

On *Win64*, the desktop icon starts a `bash` shell configured for building 64-bit programs. To start a `bash` shell configured for building 32-bit programs, launch *PGI Workstation (32-bit)* from the Start menu.

Alternatively, by selecting the appropriate option from the *PGI Workstation* program group accessed in the usual way through the "Start" menu, you can launch a standard Windows command window that is pre-initialized to enable use of the PGI compilers and tools.

The command window launched by *PGI Workstation* can be customized using the “Properties” selection on the menu accessible by right-clicking the window’s title bar.

### 3.5.2 MKS Toolkit Compatibility

The MKS Toolkit is a commercially available product providing a suite of UNIX and Windows utilities and is available for Win32 and Win64. You can use *PGI Workstation* compilers and tools in any of the MKS toolkit shells. To use *PGI Workstation* in an MKS shell, you must first configure your environment.

In the following example, the Windows system drive is C: and default installations were selected for the Java JRE and *PGI Workstation*.

Open a Windows command prompt and execute the following commands:

```
> set PGI=C:\Program Files\PGI
> PATH=C:\Program Files (x86)\Java\j2re1.5.0_05\bin;%PATH%
> PATH=%PGI%\win64\7.0-7\bin;%PATH%
> set TMPDIR=C:\temp
```

Invoke an MKS shell. For example, to start the bash shell, type:

```
> bash.exe
```

For more information or to obtain the MKS Toolkit, visit the MKS website at <http://www.mkssoftware.com/>.

### 3.5.3 Using Shared object files in SFU and SUA

*PGI Workstation* for SFU and SUA now uses the GNU ld for its linker, unlike previous versions that used the Windows LINK.EXE. With this change, the PGI compilers and tools are now able to generate shared object (.so) files. You use the -shared switch to generate a shared object file.

The following example creates a shared object file, “hello.so”, and then creates a program called “hello” that uses it.

#### Example:

First create a shared object file called "hello.so"

```
pgcc -shared hello.c -o hello.so
```

Then create a program that uses the shared object, in this example, "hello.so":

```
pgcc hi.c hello.so -o hello
```

When running a program that uses a shared object, you may encounter an error message similar to the following:

```
hello: error in loading shared libraries
hello.so: cannot open shared object file: No such file or
directory
```

This error message either means that the shared object file does not exist or its location is not specified in your `LD_LIBRARY_PATH` variable. To specify the location in your `LD_LIBRARY_PATH` variable, add the shared object's directory to your variable.

### Example:

The following example adds the current directory to your `LD_LIBRARY_PATH` variable under C Shell, enter:

```
setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH": "./"
```

The following list is a summary of compiler switches that support shared objects:

-shared	Used to produce shared libraries
-Bdynamic	Passed to linker; specify dynamic binding
-Bstatic	Passed to linker; specify static binding
-Bstatic_pgi	Use to link static PGI libraries with dynamic system libraries; implies -Mnorpath
-L<libdir>	Passed to linker; Add directory to library search path
-Mnorpath	Don't add -rpath paths to link line
-Mnostartup	Do not use standard linker startup file
-Mnostdlib	Do not use standard linker libraries
-R<ldarg>	Passed to linker; just link symbols from object, or add directory to run time search path

## 3.6 PGI Workstation 7.0 for Mac OS

*PGI Workstation 7.0 for MAC OS* supports most of the features of the 32- and 64-bit versions for *linux86* and *linux86-64* environments. Except where noted in these release notes, the PGI compilers and tools on *MAC OS* function identically to their *Linux* counterparts.

### 3.6.1 Mac OS Debugging Requirements

Both the `-g` and `-Mkeepobj` switches play important roles when compiling a program on Apple Mac OS for debugging.

- To debug a program with symbol information on the Mac OS, the user must compile with the `-g` switch to keep the program's object files, the files with a ".o" extension. Further, these object files must remain in the same directory in which they were created.
- If the user builds a program with separate compile and link steps, by compiling with the `-c` switch which generates the ".o" object files, then using the `-g` switch guarantees the required object files are available for debugging.
- If a user chooses to compile and link a program in one step, then add the `-Mkeepobj` switch to the compile line. The `-Mkeepobj` switch keeps the object files available even after the linking process occurs.

Let's look at the following two scenarios for compiling and linking a program on MAC OS.

#### Scenario 1: Separate compile and link steps

Use the following command sequence to compile and then link your code.

To compile the programs, use these commands:

```
pgcc -c -g main.c
pgcc -c -g foo.c
pgcc -c -g bar.c
```

To link, use this command:

```
pgcc -g main.o foo.o bar.o
```

#### Scenario 2: Compiling and linking in one step

Use the following command to compile and link your code in a single step:

```
pgcc -g -Mkeepobj main.c foo.c bar.c
```

## 3.7 Generating PGI Unified Binaries

All PGI compilers can produce PGI Unified Binary programs containing code streams fully optimized and supported for both AMD64 and Intel EM64T processors using the `-tp` target option. The compilers generate and combine into one executable multiple binary code streams each optimized for a specific platform. At runtime, this one executable senses the environment and dynamically selects the appropriate code stream.

Different processors have differences, some subtle, in hardware features such as instruction sets and cache size. The compilers make architecture-specific decisions about such things as instruction selection, instruction scheduling, and vectorization. PGI unified binaries provide a low-overhead means for a single program to run well on a number of hardware platforms.

Executable size is automatically controlled via unified binary culling. Only those functions and subroutines where the target affects the generated code will have unique binary images, resulting in a code-size savings of 10-90% compared to generating full copies of code for each target.

Programs can use PGI Unified Binary even if all of the object files and libraries are not compiled as unified binaries. Like any other object file, you can use PGI Unified Binary object files to create programs or libraries. No special start up code is needed; support is linked in from the PGI libraries.

The `-Mppi` option disables generation of PGI Unified Binary. Instead, the default target auto-detect rules for the host are used to select the target processor.

### 3.7.1 Unified Binary Command-line Switches

The PGI Unified Binary command-line switch is an extension of the target processor switch, `-tp`, which may be applied to individual files during compilation.

The target processor switch, `-tp`, accepts a comma-separated list of 64-bit targets and generates code optimized for each listed target.

The following example generates optimized code for three targets.

```
-tp k8-64,p7-64,core2-64
```

A special target switch, `-tp x64`, is the same as `-tp k8-64,p7-64`.

## 3.7.2 Unified Binary Directives and Pragmas

Unified binary directives and pragmas may be applied to functions, subroutines, or whole files. The directives and pragmas cause the compiler to generate PGI Unified Binary code optimized for one or more targets. No special command line options are needed for these pragmas and directives to take effect.

The syntax of the Fortran directive is

```
!pgi${g|r| } pgi tp [target]...
```

where the scope is *g* (global), *r* (routine) or blank. The default is *r*, routine.

For example,

```
!pgi$g pgi tp k8_64 p7_64
```

indicates that the whole file, represented by *g*, should be optimized for both *k8\_64* and *p7\_64*.

The syntax of the C/C++ pragma is

```
#pragma [global|routine|] tp [target]...
```

where the scope is *global*, *routine*, or blank. The default is *routine*.

For example, the following syntax indicates that the next function should be optimized for *k8\_64*, *p7\_64*, and *core2\_64*.

```
#pragma routine tp k8_64 p7_64 core2_64
```

## 3.8 Using Environment Modules

On Linux, if you use the Environment Modules package (e.g., the `module load` command), PGI 7.0 includes a script to set up the appropriate module files.

Assuming your installation base directory is `/opt/pgi`, and your `MODULEPATH` environment variable is

`/usr/local/Modules/modulefiles`, execute this command:

```
/opt/pgi/linux86/7.0-7/etc/modulefiles/pgi.module.install \  
-all -install /usr/local/Modules/modulefiles
```

This command creates module files for all installed versions of the PGI compilers. You must have write permission to the `modulefiles` directory to enable the module commands:

```
module load pgi32/7.0
module load pgi64/7.0
module load pgi/7.0
```

where "pgi/7.0" uses the 32-bit compilers on a 32-bit system and uses 64-bit compilers on a 64-bit system.

To see what versions are available, use this command:

```
module avail pgi
```

The `module load` command sets or modifies the environment variables as follows:

PGI	the base installation directory
CC	full path to pgcc
FC	full path to pgf90
F90	full path to pgf90
F77	full path to pgf77
CPP	full path to pgCC
CXX	path to pgCC
C++	path to pgCC
PATH	prepends the PGI compiler and tools bin directory
MANPATH	prepends the PGI man page directory
LD_LIBRARY_PATH	prepends the PGI library directory

PGI does not support the Environment Modules package. For more information about the package, go to: <http://modules.sourceforge.net>.

### 3.9 PGDBG and PGPROF

*PGDBG* is supported as a graphical and command line debugger in the *linux86*, *linux86-64*, *Win32*, and *Win64* execution and development environments. Like the compilers, *PGDBG* for *linux86-64* must run in a *linux86-64* execution environment. *PGDBG* for *linux86* environments is a separate version; and although it will run in the *linux86-64* execution environment, it only debugs *linux86* executables. The *linux86-64* version of *PGDBG* only debugs executables built to run as *linux86-64* executables.

*PGPROF* is supported as a graphical and command line profiler in the *linux86*, *linux86-64*, *Win32*, and *Win64* environments. The same version works in any of these environments to process a trace file of profile data created by executing the instrumented program. Program instrumentation is either line-level (*-Mprof=lines*) or function-level (*-Mprof=func*). Additionally, on Linux, *PGPROF* supports *gprof*-style (*-pg*) sample-based and trace profiling, and hardware counters (*-Mprof=hwcts*).

The *PGDBG* and *PGPROF* graphical user interfaces (GUIs) are invoked by default. To use a command line interface, invoke either tool with the *-text* option.

### 3.9.1 PGDBG New Features

*PGI Workstation 7.0* includes several new features and enhancements in the *PGDBG* parallel debugger.

- *PGDBG 7.0* now supports attachment to a running process on Windows. This support is available through the `attach` command in *PGDBG*, or through the *File->Attach to Target* menu item in the *PGDBG* GUI.
- *PGDBG 7.0* has a new command line argument, *-attach*, that automatically attaches to a running process at start-up. For example, entering the following command from a shell or command window invokes *PGDBG*, which then tries to attach to the process whose PID is 1234.

```
pgdbg -attach 1234
```

- Symbolic debugging is available for C/C++ programs compiled with Microsoft Visual C++. Files compiled with VC++ or Microsoft CL may be linked with a PGI Fortran main program, for example, and debugged using *PGDBG*.
- The bundled Java(TM) 2 Runtime Environment, Standard Edition on Windows is now version 1.5.0\_10.

For a description of the usage and capabilities of *PGDBG* and *PGPROF*, see the *PGI Tools Guide*. For limitations and workarounds, see <http://www.pgroup.com/support/faq.htm>.

## 3.10 The REDIST Directories

Programs built with PGI compiler may depend on run-time library files. These library files must be distributed with such programs to enable them to execute on systems where the PGI compilers are not installed. There are PGI redistributable file for all platforms. On Windows, PGI also supplies Microsoft redistributable files.

### 3.10.1 PGI Redistributables

The PGI 7.0 release includes these directories:

- *\$PGI/linux86/7.0/REDIST*
- *\$PGI/linux86-64/7.0/REDIST*
- *\$PGI/win64/7.0-7/REDIST*
- *\$PGI/win32/7.0/REDIST*

These directories contain all of the PGI Linux runtime library shared object files or Windows dynamically linked libraries that can be re-distributed by PGI 7.0 licensees under the terms of the PGI End-user License Agreement (EULA). For reference, a copy of the PGI EULA is included in the 7.0 directory in text form.

The Linux REDIST directories contain the PGI runtime library shared objects for all supported targets. This enables users of the PGI compilers to create packages of executables and PGI runtime libraries that will execute successfully on almost any PGI-supported target system, subject to these requirements:

- End-users of the executable have properly initialized their environment
- On Linux, users have set `LD_LIBRARY_PATH` to use the relevant version of the PGI shared objects.

### 3.10.2 Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named “redist”. PGI 7.0 licensees may redistribute the files contained in this directory in accordance with the terms of the PGI End-User License Agreement.

## 3.11 Customizing With `siterc` and User `rc` Files

The PGI 7.0 release for Linux platforms includes a `siterc` file in the `bin` directory to enable site-specific customization of the PGI compiler drivers. Using `siterc`, you can control how the compiler drivers invoke the various components in the compilation tool chain. In addition to the `siterc` file, user `rc` files can reside in a given user's home directory: `.mypgf77rc`, `.mypgf90rc`, `.mypgccrc`, `.mypgcprc`, and `.mypghpfrc` can be used to control the respective PGI compilers. All of these files are optional.

Following are some examples that show how these `rc` files can be used to tailor a given installation for a particular purpose.

To perform this task....	Do this:
Make the libraries found in <code>/opt/newlibs/64</code> available to all <code>linux86-64</code> compilations	Add the line: <pre>set SITELIB=/opt/newlibs/64; to /opt/pgi/linux86-64/7.0/bin/siterc</pre>
Make the libraries found in <code>/opt/newlibs/32</code> available to all <code>linux86</code> compilations.	Add the line: <pre>set SITELIB=/opt/newlibs/32; to /opt/pgi/linux86/7.0/bin/siterc</pre>
Add a new library path <code>/opt/local/fast</code> to all <code>linux86-64</code> compilations.	Add the line: <pre>append SITELIB=/opt/local/fast; to /opt/pgi/linux86-64/7.0/bin/siterc</pre>
Make the include path <code>/opt/acml/include</code> available to all compilations; <code>-I/opt/acml/include</code> .	Add the line: <pre>set SITEINC=/opt/acml/include; to /opt/pgi/linux86/7.0/bin/siterc and to /opt/pgi/linux86-64/7.0/bin/siterc</pre>
Change <code>-Mmpi</code> to link in <code>/opt/mympi/64/libmpix.a</code> with <code>linux86-64</code> compilations.	Add the line: <pre>set MPILIBDIR=/opt/mympi/64; set MPILIBNAME=mpix; to /opt/pgi/linux86-64/7.0/bin/siterc</pre>

To perform this task....	Do this:
Have <i>linux86-64</i> compilations always add <i>-DIS64BIT -DAMD</i>	Add the line: <pre>set SITEDEF=IS64BIT AMD;</pre> to <i>/opt/pgi/linux86-64/7.0/bin/siterc</i>
A user builds an F90 executable for <i>linux86-64</i> or <i>linux86</i> that resolves PGI shared objects in the relative directory <i>./REDIST</i>	Add the line: <pre>set RPATH=./REDIST;</pre> to <i>~/mypgf95rc</i> . <i>NOTE:</i> this will only affect the behavior of PGF95 for the given user.

### 3.12 Known Limitations

The frequently asked questions (FAQ) section of the *pgroup.com* web page at <http://www.pgroup.com/support/index.htm> provides more up-to-date information about the state of the current release.

- Object and module files created using *PGI Workstation 7.0* compilers are incompatible with object files from *PGI Workstation 5.x* and prior releases.
- Object files compiled with *-Mipa* using *PGI Workstation 6.1* and prior releases must be recompiled with *PGI Workstation 7.0*.
- Windows programs compiled with *PGF90* must be run with the *PATH* environment variable set to include the directory that contains *pg.dll*. This *dll* is redistributable and must be present on any Windows system where a program built with *PGF90 7.0-7* is to be run.
- *PGI C++ 7.0* for *Windows* template instantiation has changed considerably to match the *PGI C++* compiler on *Linux*. All C++ sources on *Windows* must be recompiled and all template instantiation flags must be removed from *Windows* makefiles.
- On *Windows*, the version of *vi* included in *cygwin* can have problems when the *SHELL* variable is defined to something it does not expect. In this case, the following messages appear when *vi* is invoked:

```
E79: Cannot expand wildcards
E79: Cannot expand wildcards
E79: Cannot expand wildcards
Hit ENTER or type command to continue
```

To workaroud this problem, set SHELL to refer to a shell in the cygwin bin directory, e.g. /bin/bash.

- The `-i8` option can make programs incompatible with MPI, use of any INTEGER\*8 array size argument can cause failures with these libraries.
- The `-i8` option can make programs incompatible with the bundled ACML library. Visit [developer.amd.com](http://developer.amd.com) to check for compatible libraries.
- Programs that incorporate object files compiled using `-mcmodel=medium` cannot be statically linked. This is a limitation of the *linux86-64* environment, not a limitation specific to the PGI compilers and tools.
- Using `-Mipa=vestigial` in combination with `-Mipa=libopt` with *PGCC*, you may encounter unresolved references at link time. This problem is due to the erroneous removal of functions by the *vestigial* sub-option to `-Mipa`. You can work around this problem by listing specific sub-options to `-Mipa`, not including *vestigial*.
- Using `-Mprof=func`, `-mcmodel=medium` and `-mp` together on any of the PGI compilers can result in segmentation faults by the generated executable. These options should not be used together.
- Programs compiled and linked for *gprof*-style performance profiling using `-pg` can result in segmentation faults on system running version 2.6.4 Linux kernels. In addition, the time reported for each program unit by *gprof* and *PGPROF* for such executables run under some Linux distributions can be a factor of 10 higher than the actual time used. This behavior is a bug in certain shared object libraries included with those Linux distributions.
- *OpenMP* programs compiled using `-mp` and run on multiple processors of a *SuSE 9.0* system can run very slowly. These same executables deliver the expected performance and speed-up on similar hardware running *SuSE 9.1* and above.
- ACML 3.6 is built using the `-fastsse` compile/link option, which includes `-Mcache_align`. When linking with ACML using the `-lacml` option on 32-bit targets, all program units must be compiled with `-Mcache_align`, or an aggregate option such as `-fastsse` which incorporates `-Mcache_align`. This process is not an issue on 64-bit targets where the stack is 16-byte aligned by default. The lower-performance, but fully portable, *libblas.a* and *liblapack.a* libraries can be used on CPUs that do not support SSE instructions.

- On SUA and SFU, programs compiled with `-Mchkfpstk` may fail when executing hyperbolic math functions such as `tanh` or `sinh` because the x87 stack is erroneously reported as not empty. The workaround is to not compile routines that call the hyperbolic functions with `-Mchkfpstk`.
- Times reported for multi-threaded sample-based profiles, that is, profiling invoked with `-pg` or `-Mprof=time` options, are for the master thread only. PGI-style instrumentation profiling with `-Mprof={lines | func}` or hardware counter-based profiling using `-Mprof=hwcts` must be used to obtain profile data on individual threads.
- *PGDBG* – On Windows platforms, program arguments are passed incorrectly from the *PGDBG* command line, such that the executable is listed as both `argv[0]` and `argv[1]` in a C program. Program arguments are passed from the *PGDBG run* command correctly.
- *PGDBG* GUI – From the command pane, the `source` command does not wait for execution to stop. It continues to read commands, even if the target is running.
- For example, if the `source` script contains commands to set a breakpoint, run and print a stack trace, the expectation might be that the stack trace would print at the breakpoint. In fact, it might return an error, since the target could be running when `stacktrace` is executed.

The only workaround is to insert a `wait` command after each control command in the script.

- *PGDBG* – The `watch` family of commands is unreliable when used with local variables. Calling a function or subroutine from within the scope of the watched local variable may cause missed events and/or false positive events. Local variables may be watched reliably if program scope does not leave the scope of the watched variable. Using the `watch` family of commands with global or static variables is reliable.
- *PGDBG* – The `stacktrace` command may skip a frame in the call stack if it encounters a routine compiled without `-g`. This behavior is most noticeable when an exception is encountered in a library routine such as `memset()`, and `stacktrace` does not show the calling routine. The current routine may not be identified by name, showing only `unknownaddr`. There is no known workaround for this problem.
- *PGDBG* – The `call` command does not support the following F90/F95 features: array-valued functions, pointer-valued functions, assumed-shape array arguments, or pointer arguments. There is no known workaround to this limitation.

- *PGDBG* – If you execute a `run` or `rerun` command with no arguments after a previous `run` or `rerun` that specified I/O re-direction, I/O redirection continues as specified in the previous `run` or `rerun`. This behavior can cause unexpected results to be appended to a file specified as `stdout`, and can cause unexpected program failures due to erroneous program input from `stdin` which is not reset to the start of the intended input file. This limitation also applies to a `shell` command issued after a `run` or `rerun` with I/O redirection.
- *PGDBG* – Before *PGDBG* can set a breakpoint in code contained in a shared library, `.so` or `.dll`, the shared library must be loaded.
- *PGDBG* – Debugging of unified binaries, that is, programs built with the `-tp=x64` option, is not fully supported. The names of some subprograms are modified in the creation of the unified binary, and *PGDBG* does not translate these names back to the names used in the application source code. For detailed information on how to debug a unified binary, see <http://www.pgroup.com/support/tools.htm>.
- *PGDBG Win* – In Windows, file path names use the backslash (`\`) character to delimit directory names. *PGDBG* uses C Language notation for expressions, which means the backslash character is the escape character.
- In *PGDBG* on the *Windows* platform, use the forward slash (`/`) character to delimit directory names in file path names.  
*Note.* This requirement does not apply to the `DEBUG` command or to target executable names on the command line, although this convention will work with those commands.
- *PGPROF Windows* – Profiling of DLLs is not supported.
- Using `-Mppi` and `-mp` together is not supported. The `-Mppi` flag will disable `-mp` at compile time, which can cause run-time errors in programs that depend on interpretation of OpenMP directives or pragmas. Programs that do not depend on OpenMP processing for correctness can still use profile feedback. The `-Mpfo` flag does not disable OpenMP processing.
- *PGDBG* does not currently support debugging core files on SFU/SUA systems.
- Due to lack of operating system support, *PGDBG* does not support hardware watchpoints, that is, the "hwatch" command, on SFU/SUA systems.

- Due to operating system limitations, PGDBG supports multi-thread debugging only with 32-bit SUA programs, with one restriction: once stopped, the process may be continued by continuing all threads, or a single thread, but not a partial set of the threads. Attempts to continue a partial set of threads results in the entire process, all threads, being continued.
- Dynamic Link libraries built on the *Windows* platform by the *PGI Workstation 7.0* compilers have the following known limitations:
  - DLLs cannot be produced with the *PGI Workstation C++* compiler.
  - 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.
  - If an executable is linked with any *PGI Workstation*-compiled DLL, the *PGI Workstation* runtime library DLLs must be used; this means the static libraries cannot be used. To accomplish this, use the compiler option *-Mdll* when creating the executable.
  - Do not use *-Mprof* with *PGI Workstation* runtime library DLLs. To build an executable for profiling, use the static libraries. When the compiler option *-Mdll* is *not* used, the static libraries are the default.

### 3.13 Corrections

The following problems have been corrected in the *PGI Workstation 7.0* release. Most were reported in *PGI Workstation 6.2* or previous releases. Problems found in *PGI Workstation 6.2* may not have occurred in previous releases. A table is provided that describes the summary description of the problem. 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. For a complete and up-to-date list of TPRs fixed in recent releases of the PGI compilers and tools, see [http://www.pgroup.com/support/release\\_tprs.htm](http://www.pgroup.com/support/release_tprs.htm).

### 3.13.1 Corrections in 7.0-7

The following problems have been corrected in 7.0-7:

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
2932	pgf90	-M[no]asmkeyword not working
3974	pgcpp	PGI does not return small struct/union in registers
4135	pgf90	Example with PARAMETER declaration causes ICE 'module:new_dtype, dt nfd'
4152	pgf90	Pgf90 example ICE – unexpected data type at assignment
4155	pgcpp	C++ overload issue
4209	pgf90	Customer Claims TPR 4157 fix caused ICE – 'ipa_import: loop directive record out of order'
4221	pgf90	SPR 738851/738791 – Another ICE with 'module: new_dtype, dt nfd' error esperanto
4236	pgf90	Transfer function in 64-bit fails, 32-bit works
4238	pgcc	PGI Compilers fails to compile ternary operator expressions
4240	pgf90	Legal f90 fails with 'illegal use of symbol – must have the Target or Pointer attribute'
4255	All	MCNP5 application gets runtime failure when compiled with the 32-bit FORTRAN (pgf90) Linux compiler
4258	pgcpp	Pgcpp1 is freshness dated
4261	pgf90	WHERE construct incorrectly generates severe error
4262	pgf90	Matmul function does not produce correct answers
4267	All	PGI EVAL license prevents shared library creation - we need an -fPIC '__pgio_ini'

### 3.13.2 Corrections in 7.0-6

The following problems have been corrected in 7.0-6:

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
4144	pgcpp	pgCC/pgcpp has unrealistic hard limit of 8K bytes for mangled names
4178	All	32-bit libpgc.so does not define __pgi_trace as 'weak'

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
4216	pgcpp	pgCC/pgcpp crashes on BOOST PYTHON component
4220	pgf90	pgf90 gets lowering error for mixing logical, real data types
4230	pgdbg	GUI File selector combo box not available
4231	pgdbg	pgdbg fails to find debugging symbols when directory names contains a "."
4232	pgdbg	SUA: GUI Open Target Action fails
4239	All	SFU32 "div" function does not appear to work

### 3.13.3 Corrections in 7.0-5

The following problems have been corrected in 7.0-5:

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
3692	pgf90	Performance and correctness changes with loop count size w/wo -Kieeee
4082	pgcc	'-fastsse -Minline=levels:10' causes ICE 'sym_is_refd: bad sprt' on 32-bit platform
4085	pgcc	Aborts with 'Source file too large to compile at this opt level'
4112	pgf90	ICE 'mismatched carry-around expression' with -Mlre
4136	pgf90	SHAPE intrinsic fails with -i8
4177	pgf90	Switch combination causes compiler to hang
4211	pgf90	Modfile compiled with 7.0-2 in file compiled 7.0-3 causes ICE 'Errors in ILM file'

### 3.13.4 Corrections in 7.0-4

The following problems have been corrected in 7.0-4:

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
3556	pgcc	y.c does not generate a fp exception at runtime at -O2
3620	pgcc	pgcc fails to act same at -O2 as -O1 and -O3
3899	pgf90	-Mbounds triggers error for allocating character array of size zero, but not for integer array

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
4003	pgf90	Defective program breaks compiler rather than emit error info
4115	pgf90	pgf90 example causes compiler crash with -Mstandard
4117	pgf90	-Mstandard flags variable name length after decorating it
4130	All	GMON_OUT_PREFIX, to work the way it does with gprof
4137	All	New driver behavior should have switch to go back to old behavior
4141	All	Feature request -gfortranlibs
4146	pgf90	TARGET attribute disables optimizations
4157	pgf90	Overlap warnings portend difficult ICE 'module:new_dtype, dt nfd'

### 3.13.5 Corrections in 7.0-3

The following problems have been corrected in 7.0-3:

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
2978	pgcc	Bad Code in cast of int to char
3668	pgf90	pgf90 program causes TS-11
3715	All	NUMA support for RHEL 4?
3743	pgf90	Long compilation time for large array data initialization
3757	pgf90	Dummy args as allocatables
3802	pgcc	Can pgi preprocessor display predefined macros like ' cpp - dM /usr/include/stdio.h'
3874	pgcpp	C++ code fails to print 'long double' data type using cout
3900	All	OpenMP limit set to minimum when main is not compiled with PGI compilers
3904	All	makelocalrc needs to be French (and other langs) aware
3908	pgf90	Fortran I/O on 64-bit systems much slower with -mcmodel=medium
3919	pgf90	Fortran example misses opportunity to hoist costly dsin from loop (add -O4)
3923	pgf90	Loop bounds passed in struct, stored locally, aren't countable

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
3940	pgf90	memory leak in allocatable components
3945	All	Please add large array coding considerations to the user guide
4011	pgf90	2.0**66 causes a floating exception with -Ktrap=fp
4016	pgcpp	fails with ' <code>__attribute__((aligned(16)))</code> ' in code
4026	pgf90	IMPLIED DO error where there is none, and compiler crash where there is an error
4044	pgf90	Seg Fault executing a write with multiply by array section
4045	pgf90	Forall gives wrong answer when lhs is an allocatable array in a derived type
4060	pgf90	Program seg faults with character length >64, and Transfer function
4065	pgf90	Simple fortran OPEN test works with pgf77, fails with pgf90
4069	CDK	mpif90, mpif77, etc scripts need to act exactly like pgf90 -Mmpi or -Mmpi2, etc
4083	pgcpp	32-bit pgCC does not read back 'long long' from file properly
4087	pgf90	pgf90 bug for Pointers in COMMONs - storage order messed up
4099	All	Stack command shows only current routine on win32
4110	All	Stack overflow on SFU with <code>-mp -Mchkstk</code>
4116	CDK	CDK installations do not properly support 'mpixxx -V'
4118	pgf90	Release 7.0-2 pgf90 no longer supports !DEC\$ directives the way 6.2-5 did
4145	All	OpenMP Directive NUM_THREADS extends outside scope of current section

### 3.13.6 Corrections in 7.0-2

The following problems have been corrected in 7.0-2:

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
2978	pgcc/ pgCC	Bad code in cast of int to char
3693	All	Windows lacks erf and erfc routines
3741	pgf90	TS-11 with -tp piii and -O2
3768	pgf90	Loop interchange runs much slower than manual loop interchange
3795	pgf90	ICE with -fastsse -tp p6
3799	All	Nodes are sharing the same scratch file name
3805	pgcc	Missing C99 Macro variable "__STDC_VERSION__"
3869	pgcc/ pgCC	Request for -dN and/or -dD to list predefs
3892	pgcc	C99 math.h prototypes not being used.
3911	pgf90	-Mchkptr mistakenly decides associated(x) as a dereference of x
3922	All	Reference to sched_yield in non-mp programs
3925	pgf90	Initializer Alt2 = ((3.0*i, i=1,nn)) 'Illegal implied DO expression' wrong
3930	pgf90	ICHAR problem in the initialization process
3946	pgf90	Example causes ICE 'flowgraph: node is zero' in -tp x64
3947	pgf90	PGF90 NIMROD ORNL example -- ICE 'Errors in Lowering'
3955	pgcc/pgCC	Driver does not recognize suffix .h file for preprocess
3965	pgf90	Optimization opportunity
3969	pgf90	__builtin_stinit is not available when linking vc++ with a PGI-generated shared lib on Win32
3972	pgcc/pgCC	DWARF info in 6.2-3 looks wrong
3975	pgf90	64-bit pgf90 error "suffix or operands invalid for `movsd'"
3977	pgf90	Compiled code should detect error -Mstandard

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
3980	pgf90	Program causes pgf90 to terminate with signal 11
3982	pgf90	Customer example with mod files of same name, but only one used, stymies pgf90
3985	All	Set up default options for a site.
3988	All	Code example should vectorize
3990	pgf90	User code seg faults when using a passed procedure in an internal function
3991	pgf90	Save attribute ignored in pgf90 64-bit
3992	pgf90	DIM argument to minval and maxval appears broken
3993	pgf90	Program fails because of -Mchkptr
3995	pgf90	Vectorization results in wrong answers
3997	pgf90	Higher opt levels causes errors
3998	pgf90	Program fails to link when module compiled -g
4000	pgf90	\$TMPDIR set to invalid directory causes compiler hang
4005	pgcc	Variable-length array fails
4006	pgf90	Customer reports compilation failure for a valid FORTRAN program
4007	All	Installation instructions about rc.d/init.d are not consistent across Linux versions
4008	pgf77	Program compiled -Minline crashes pgf77
4009	pgf90	Code with strings fails on execution
4010	pgcpp/pgCC	Typo in cpprc causes "--use_pch" to fail
4012	pgf90	CE with -Mvect " wrong loop indices when computing distances"
4014	pgf90	64-bit code compiled -g causes compiler crash
4015	pgf90	Code causes ICE 'Lowering Error: logical result for arithmetic operation'
4018	pgf90	WRF V2.2 Fails to compile
4019	pgf90	Using .o on Windows with Fortran compilers causes link failure

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
4020	All	Create distribution directory for catamount libraries
4021	pgf90	Error with lbound of an assumed-shape array
4024	pgf90	ICE - transform_call:Array Expression can't be here
4025	pgf90	OPEN and INQUIRE handle file names with a trailing NULL character differently
4027	pgf90	Errors with threadprivate common block containing a pointer
4028	pgf90	Incorrect example cause compiler ICE - 'sym_of_ast: unexpected ast'
4029	pgf90	Example with RECORD problem generates ICE
4031	pgf90	Program causes seg fault in compiler
4032	pgcc	OpenMPI compile fails with ICE because of struct copy containing bool
4034	pgf90	ICE ' unexpected ast type in initialization expr'
4036	pgf90	Threadprivate global variable not used in contained subprograms
4038	pgcc	-MM should only give user include header file dependencies, instead gives all headers
4041	pgf90	Module variables not imported correctly when used in a DLL
4043	pgf90	Failures when running with multiple threads on a RHEL3 Opteron machine
4048	All	Manuals shouldn't talk about Netscape
4050	pgf90	libpgc.a in the 64-bit libso directory has non-fpic compiled objects
4052	pgf90	Call exit(code) fails to echo 'code'
4053	pgcc	PGI include file stdlib not properly adding include_next
4054	pgcc	C example gives correct answer at -O1, fails at -O2
4055	All	Release notes confusing w.r.t. new fast math intrinsic
4056	pgcc	Inline assembly fails with this example
4057	pgf90	Async I/O library routines need to be removed from libqk_pgf90.a
4059	pgf90	User has problem linking application
4062	pgcc	OpenMPI fails to build in their nightly regression system. filename too long

<b>TPR</b>	<b>Lang/ Tool</b>	<b>Description</b>
4067	pgf90	Example gives different answers than gfortran, requires USE of a module that seems unnecessary
4072	pgcc	User code causes pgcpp2 to seg fault when compiled with "-Mipa"
4073	All	"-lrt" missing from link line when -pgf90libs or -pgf77libs is used with pgcc
4076	pgcc	VLA after a block that reads size gives ICE on RHEL 4.
4077	pgf90	Link with -mcmodel=medium 'Could not resolve generic procedure fu_explain_neq_shape'
4078	All	/bin/strip breaks SFU/SUA executables
4080	All	liblapack and libblas missing from Windows products



# 4 Contact Information and Documentation

---

You can contact The Portland Group at:

*The Portland Group  
STMicroelectronics, Inc.  
Two Centerpointe Drive  
Lake Oswego, OR 97035 USA*

The PGI User Forum is monitored by members of the PGI engineering and support teams as well as other PGI customers. The forum newsgroups may contain answers to commonly asked questions. Log in to the PGI website to access the forum:

<http://www.pgroup.com/userforum/index.php>

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 by e-mail or submissions using an online form at <http://www.pgroup.com/support>. Phone support is not currently available.

Many questions and problems can be resolved at our frequently asked questions (FAQ) site at <http://www.pgroup.com/support/faq.htm>.

Online documentation is available at <http://www.pgroup.com/doc> or in your local copy of the documentation in the release directory [doc/index.htm](http://www.pgroup.com/doc/index.htm).