PGI® Server 7.0
PGI® Workstation 7.0

Installation & Release Notes

The Portland Group™
STMicroelectronics, Inc
Two Centerpointe Drive
Lake Oswego, OR 97035
www.pgroup.com

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

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

Table of Contents

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* and Windows* 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 for 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 for 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.

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`, `pghpf`, `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` are identical.

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.

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.

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.

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.

x64 – collectively, all AMD64 and EM64T processors 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, enabling 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.

SSSE3 – an extension of the sSE3 instruction set found on the Intel Core 2 .

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

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.

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.

SUA – the Subsystem for Unix-base 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.

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

–mcmodel=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 is the default and only possible format for *linux86* 32-bit executables. This 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.

–mcmodel=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. Supported *only* in *linux86-64* environments. This option 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 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 *–mcmodel=medium* can incorporate objects compiled using *–mcmodel=small* as long as the *small* objects are from a shared library.

Large Arrays – arrays with aggregate size larger than 2GB**,** which requires 64-bit index arithmetic for accesses to elements of arrays. Program units that use *Large Arrays* must be compiled using *–mcmodel=medium*. If *–mcmodel=medium* is not specified, but *–Mlarge_arrays* is specified, the default small memory model is used but 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.

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

Static 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.

Static linking – 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 to link.

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.

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 2nd *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.

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 4, will on a Dual-core system with 4 CPUs enable process creations up to a maximum of 8 (4 CPUs * 2 Cores per CPU or 8 total processes).

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.

2 PGI Release 7.0 Installation Notes

2.1 Introduction

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

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. On Windows, only FLEXlm-style licensing is supported. 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 for *PGI Workstation* allows any user of the system to use the compilers; however, only a single simultaneous copy of the compiler or tools is allowed.

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.

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 choose to make a network install, adding another machine to the group running the compilers is a much simpler process now, adjusting only to the unique characteristics of the newly added system with a customization script that must be executed on each machine in the group.

If you require FLEXlm-style licensing, you must request FLEXlm-style license keys when generating your keys, install the PGI compilers and tools according to the instructions in the following sections, 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.

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

NOTE

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

Note: Trial licensing is a deprecated feature and will be removed from a subsequent release. You must specifically request trial keys from the PGI web site at *http://www.pgroup.com*.

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 will have access to e-mail support for an additional 12 months and will be notified by e-mail when maintenance releases occur and are available for electronic download and installation. Phone support is not currently available. Contact us at *sales@pgroup.com* if you would like information regarding the subscription service for the PGI products you have purchased.

Section 2.2 below describes how to install *PGI Workstation* or *PGI Server* in a generic manner on Linux. Section 2.3 describes how to install and run a FLEXlm license daemon on Linux. Section 2.4 describes how to install *PGI Workstation* on Windows systems and how to install and run a FLEXlm license daemon on Windows.

2.2 Installing on Linux

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

Name of directory	Contents
`/opt/pgi/linux86/7.0/bin`	*linux86* 32-bit compilers & tools
`/opt/pgi/linux86/7.0/lib`	*linux86* 32-bit runtime libraries
`/opt/pgi/linux86/7.0/liblf`	*linux86* 32-bit large-file support libs (used by *–Mlfs*)
`/opt/pgi/linux86/7.0/include`	*linux86* 32-bit header files
`/opt/pgi/linux86-64/7.0/bin`	*linux86-64* compilers & tools

Name of directory	Contents
`/opt/pgi/linux86-64/7.0/lib`	*linux86-64 –mcmodel=small* libs
`/opt/pgi/linux86-64/7.0/libso`	*linux86-64 –fpic* shared libraries for *–mcmodel=medium* development
`/opt/pgi/linux86-64/7.0/include`	*linux86-64* 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 *PGDBG* and *PGPROF* 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:

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

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 both the *linux86-64* and *linux86* versions of the PGI compilers and tools. However, the 32-bit *linux86* tools will be installed on a 64-bit *x64* system only if the 32-bit *gcc* development package is already installed on the system. The 32-bit and 64-bit compilers, tools and supporting components have the same command names, and the environment you target by default (*linux86-64* or *linux86*) will depend on the version of the compiler that comes first in your path 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 filesystem for the installation base directory. You must also specify a second directory name that will be local to each of the systems where the PGI compilers and tools will be used. This local directory will have the libraries to use when compiling and running on that machine. This allows a network installation to support a network of machines running different versions of Linux.

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

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, `<tarfile>` needs to be replaced with the name of the file that was downloaded.

The PGI products cannot be installed into the same directory where the tar file is unpacked. 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 – The install script ***must*** be run to properly install the software. 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
```

The install script will list 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 to select an installation directory.

NOTE: For a network installation, you will also be asked for a common local directory; this is a directory that will be created once for each platform utilizing the network installation, and must be created on each platform before adding that platform to the network using the compilers.

After the software is installed, the script will do some system-specific customization and then initialize the licensing, which is covered in step 3 below.

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

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

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

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

Step 3 – All of the PGI compilers and tools are license-managed. *PGI Workstation* products that are node-locked and limited to a single user are not required to run a FLEXlm license daemon. If you want the *PGI Workstation* compilers to be usable by any one user rather than locked to a specific username, or if you are installing a multi-user *PGI Server* product, you must use FLEXlm and must specifically request FLEXlm-style keys when generating license keys over 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 should have received an order acknowledgement e-mail with instructions on how to generate your license keys through the *pgroup.com* web page.

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

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


Name:   <your name>
User:   <your username>
Email:  <your e-mail address>
Hostid: PGI=9BF378E0131FF0C3CD37F6
FLEXlm hostid: 00a024a3dfe7
Hostname: yourhost.yourdomain.com
Installation: /opt/pgi
PGI Release: 7.0-2
```

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

Note: Trial licensing is a deprecated feature and will be removed from a subsequent release. You must specifically request trial keys from the PGI web site at *http://www.pgroup.com*.

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

Step 4 – You can view the online HTML and PDF documentation using any web browser 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 – With either the temporary or permanent license file in place, execute the following commands to make the products you have purchased accessible. Note that the path settings below assume that a 64-bit Linux product has been installed.

Assuming `csh` and installation in the default `/opt/pgi` directory:

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

Or, assuming `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
```

If you install only the *linux86* versions of the compilers or wish to target *linux86* as the default, use the same setup with an alternate path setting:

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

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

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

All users of the PGI products should add these commands to their startup files to ensure they have access to the PGI compilers and tools upon future logins.

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

- For Fortran 77, use "`pgf77 -V x.f`"
- For Fortran 95, use "`pgf95 -V x.f`"
- For HPF, use "`pghpf -V x.f`"
- For C++, use "`pgCC -V x.c`" or "`pgcpp -V x.c`"
- For ANSI C, use "`pgcc -V x.c`"

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

Step 7 – For a network installation, you must run the local installation script on each machine on the network where you will use the compilers and tools. If your installation base directory is /opt/pgi and the common local directory is /usr/pgi/shared/7.0, run the commands

```
/opt/pgi/linux86/7.0-2/bin/makelocalrc \
       -x /opt/pgi/linux86/7.0-2 \
       -net /usr/pgi/shared/7.0

/opt/pgi/linux86-64/7.0-2/bin/makelocalrc \
       -x /opt/pgi/linux86-64/7.0-2 \
       -net /usr/pgi/shared/7.0
```

on each machine on the network. This will create a machine-dependent file 'localrc.machinename' in /opt/pgi/linux86/7.0-2/bin and /opt/pgi/linux86-64/7.0-2/bin, and create directories /usr/pgi/shared/7.0/{lib,liblf,lib64} containing libraries and shared objects specific to the operating system and system libraries on that machine.

Note that the `makelocalrc` command does allow the flexibility to have a different named local directory on different machines. However, using the same directory on different machines allows users to easily move binaries that load PGI-supplied shared libraries between systems.

2.2.1 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. 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 `csh`,

```
  % 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)
```

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

```
  % 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
```

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

In `csh`,

```
  % 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)
```

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

```
  % 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
```

2.3 Installing FLEXlm on Linux

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

IMPORTANT NOTE: Release 7.0 includes a newer version of the Macrovision FLEXlm software. The *lmgrd* and *pgroupd* daemons are updated and 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 use any of these versions of the compilers and tools with a single Release 6.2 license file and the new versions of *lmgrd* and *pgroupd*.

You must modify your `lmgrd.rc` file in the `/etc/rc.d` or `/etc/init.d` directory to use the new *lmgrd* if you use this file to start *lmgrd* automatically after a reboot of your system. For example:

```
## 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
```

where `<target>` is replaced appropriately with `linux86` or `linux86-64`. See ***Step 4*** below for complete details on setup and usage of these files.

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

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

```
SERVER <hostname> <hostid> 7496
DAEMON pgroupd 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
```

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

NOTE: In the feature line component `VENDOR_STRING=107209:16` is the Product ID Number (PIN) for this installation. You will have a similar unique PIN number for your installation. Please include your PIN number when sending mail to us regarding technical support for the products you have purchased.

Step 3 – When the license file is in place, execute the following commands to make the products you have purchased accessible. Issue the following commands to initialize your environment for use of FLEXlm:

In `csh`,

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

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

```
% 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 you have access to the PGI products upon future logins.

Step 4 – You must now start the license manager daemon. In the following paragraphs, if you have only installed *linux86-64*, please substitute *linux86-64* for *linux86*.

Edit the shell script template `/opt/pgi/linux86/7.0/bin/lmgrd.rc`. If you have installed the compilers in a directory other than `/opt/pgi`, substitute the correct installation directory for '/opt/pgi' on line 3 of the script. Now exit the editor and 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 above, 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 that your system's default runlevel may be something other than '3', and if it is, that number should be used above in setting the correct subdirectory. Run `/sbin/runlevel` to check the system's runlevel. Note also that your `rc` files may be in a directory other than `/etc/initd.d` such as `/etc/rc.d/init.d`.

Some Linux distributions such as 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. If you have difficulties with the installation, send e-mail to *trs@pgroup.com* for assistance.

2.4 Installing on Windows

Note: 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*.

Step 1 – Uninstall any PGI pre-release software

Prior to installing *PGI Workstation*, please be sure to uninstall any PGI pre-release installation and, in particular, remove any license.dat file associated with such an installation. In a typical configuration, this file would be found in:

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

where `C:\` is the system drive.

Step 2 – Log on as Administrator

Administrator privileges are required to install *PGI Workstation*.

Step 3 – Start the installation

If you are installing *PGI Workstation* from a CD-ROM, insert the CD-ROM into the CD-ROM drive on the system on which the install is to take place. An installation script will automatically be invoked and the installation process will begin. Follow the directions printed to your screen. If you have configured your system so that CD-ROM autorun is disabled, run the *PGI Workstation* installation executable from the CD. If you obtained *PGI Workstation* from PGI electronically, execute this file on the target machine.

Step 4 – Install the license keys

The *PGI Workstation* compilers and tools on Windows are license-managed using the FLEXlm software license management system from Macrovision Software. This system requires that you possess valid license keys for the licensed product. Your order acknowledgement will contain instructions on how to obtain license keys. These keys must be put in the file license.dat. In a typical configuration, this file would be found in:

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

where `C:\` is the system drive.

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.

Step 5 – 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 is in place (refer to Step 4), this service can be started. Open the Services dialog (Start:Control Panel:Administrative Tools:Services), select "PGI License Server", and select "Start." The PGI License Server service will also start automatically on system reboot provided that the license.dat file contains valid keys.

2.4.1 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 pre-initialized with environment and path settings for use of the *PGI Workstation* compilers and tools. If you prefer different background or text colors, font style, window size, or scrolling capability, you can customize the "shortcut" that creates the *PGI Workstation* command window. Right-click on the *PGI Workstation* desktop icon, and left-click "Properties" from the pop-up menu. Modify the features mentioned above by selecting the appropriate tabs in the pop-up window and making modifications as desired.

2.4.2 PGI Workstation Directory Structure

On Win32, the default installation directory is

```
%SYSTEMDRIVE%\Program Files\PGI\win32\7.0-2\
```

On Win64 platforms, the default installation directories are

```
%SYSTEMDRIVE%\Program Files\PGI\win64\7.0-2\
```

```
%SYSTEMDRIVE%\Program Files (x86)\PGI\win32\7.0-2\
```

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

```
%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 2.2.1, 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`
	PGI Workstation 7.0 compilers and tools binaries

`C:\Program Files\PGI\win64\7.0\lib`
 	
`C:\Program Files (x86)\PGI\win32\7.0\lib`
	PGI Workstation 7.0 runtime and support libraries

`C:\Program Files\PGI\win64\7.0\include`
 	
`C:\Program Files (x86)\PGI\win32\7.0\include`
	PGI Workstation 7.0 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

Name of directory	**Contents**
`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`	*PGI Workstation 7.0* compilers and tools binaries
`C:\Program Files\PGI\win32\7.0\lib`	*PGI Workstation 7.0* runtime and support libraries
`C:\Program Files\PGI\win32\7.0\include`	*PGI Workstation 7.0* 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`	Man pages for commands
`C:\Program Files\PGI\Microsoft Open Tools 8`	Microsoft tools
`C:\cygwin`	Cygwin tools

2.4.3 Using LM_LICENSE_FILE

The system environment variable `LM_LICENSE_FILE` is not required by *PGI Workstation* but can be used to override the default location 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, open the System Properties dialog (Start:Control Panel:System). Select the 'Advanced' tab. Click on 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.

2.4.4 Common Installation Problems

The most common installation problems 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. The environment variable FLEXLM_BATCH prevents interactive pop-ups from appearing. Set the environment variable FLEXLM_BATCH to 1 to prevent interactive pop-up windows.

3 PGI Release 7.0 Release Notes

This document describes changes between Release 7.0 of the PGI compilers and previous releases, as well as late-breaking information not included in the current printing of the *PGI User's Guide*. There are seven 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.

These versions are distinguished in these release notes where necessary.

3.1 PGI Release 7.0 Contents

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

- *PGF95* native OpenMP and auto-parallelizing Fortran 95 compiler.
- *PGF77* native OpenMP and auto-parallelizing FORTRAN 77 compiler.
- *PGHPF* data parallel High Performance Fortran compiler. **Note:** ***PGHPF 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

Release 7.0 of the PGI compilers and tools is supported on the following processors. The *–tp <target>* command-line option is used to generate 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.

CPUs available and supported in multi-core versions are noted here as well.

Processors Supported by PGI 7.0

Brand	CPU	Cores	*<target>*	Memory Address	Floating Point HW			
					x87	SSE1	SSE2	SSE3
AMD	Opteron/Athlon64	2	k8-64	64-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

Processors Supported by PGI 7.0

Brand	CPU	Cores	*<target>*	Memory Address	Floating Point HW			
					x87	SSE1	SSE2	SSE3
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

Release 7.0 of the PGI compilers and tools is supported on the operating systems listed in the table below, and their equivalents. To determine if Release 7.0 will install and run under a Linux equivalent version (Mandrake*, Debian*, Gentoo*, etc), look to see if a supported system with the same *glibc* and *gcc* versions is in the table. 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 will be installed. If you attempt to install the 64-bit Windows version on a system running 32-bit Windows, the installation will fail.

Most newer Linux distributions support the *Native POSIX Threads Library* (NPTL), a new threads library that can be utilized in place of the *libpthreads* 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 will automatically make use of NPTL on distributions where 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*. See *Terms and Definitions* for more information on these terms.

Operating Systems and Features Supported in PGI 7.0

Distribution	**Type**	**64-bit**	**HT**	**pgC++**	**pgdbg**	**NPTL**	**NUMA**	**glibc**	**GCC**
RHEL 4.0	***Linux***	Yes	Yes	Yes	Yes	Yes	No	2.3.4	3.4.3
Fedora C-6	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.5	4.1.1
Fedora C-5	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
Fedora C-4	***Linux***	Yes	Yes	Yes	Yes	Yes	No	2.3.5	4.0.0
Fedora C-3	***Linux***	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.4.2
Fedora C-2	***Linux***	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.3.3
SuSE 10.2	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.5	4.1.0
SuSE 10.1	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
SuSE 10.0	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.3.5	4.0.2
SuSE 9.3	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.3.4	3.3.5
SuSE 9.2	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.3.3	3.3.4
SLES 10	***Linux***	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
SLES 9	***Linux***	Yes	Yes	Yes	Yes	No	Yes	2.3.3	3.3.3

Operating Systems and Features Supported in PGI 7.0									
Distribution	**Type**	**64-bit**	**HT**	**pgC++**	**pgdbg**	**NPTL**	**NUMA**	**glibc**	**GCC**
SuSE 9.1	***Linux***	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.3.3
RHEL 3.0	***Linux***	Yes	Yes	Yes	Yes	Yes	No	2.3.2	3.2.3
SuSE 9.0	***Linux***	Yes	Yes	Yes	Yes	No	No	2.3.2	3.3.1
SuSE 8.2	***Linux***	Yes	Yes	Yes	Yes	No	No	2.3.2	3.3
RedHat 9.0	***Linux***	No	No	Yes	Yes	Yes	No	2.3.2	3.2.2
Red Hat 8.0	***Linux***	No	No	Yes	Yes	No	No	2.2.93	3.2
SLES8 SP2	***Linux***	Poor	Yes	Yes	Yes	No	No	2.2.5	3.2.2
SuSE 8.1	***Linux***	Poor	Yes	Yes	Yes	No	No	2.2.5	3.2.2
SuSE 8.0	***Linux***	No	No	Yes	Yes	No	No	2.2.5	2.96
Red Hat 7.3	***Linux***	No	No	Yes	Yes	No	No	2.2.5	2.96
Microsoft Windows	*XP*	No	Yes	Yes	Yes	NA	Yes	NA	NA
	2003	No	No	Yes	Yes	NA	Yes	NA	NA
	2000	No	No	Yes	Yes	NA	Yes	NA	NA
	XP x64	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	2003 x64	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	SFU	No	Yes	Yes	Yes	NA	Yes	SFU	3.3
	SUA x86	No	Yes	Yes	Yes	NA	Yes	SUA	3.3
	SUA x64	Yes	Yes	Yes	Yes	NA	Yes	SUA	3.3

NOTE: *http://www.pgroup.com/support/install.htm* lists any new Linux 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 table above, the installation may still be successful.

3.2.3 New System Calls

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

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

aio_error
aio_read
aio_return
aio_suspend
aio_write
calloc
getrlimit

pthread_attr_init	pthread_mutex_unlock
pthread_mutex_init	setrlimit
pthread_mutex_lock	sleep

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* –PGI compilers produce PGI Unified Binary programs containing code streams fully optimized and supported for both AMD and Intel x64 CPUs using the *–tp x64* target option.
- *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, resulting 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 module

ISO_C_BINDING has been 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. The other procedures in *ISO_C_BINDING*, for example, *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, 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* is provided to enabled the Fortran 2003 assignment semantics.
- *Fortran 2003 Asynchronous Input/Output* – Fortran 2003 asynchronous i/o has been partially implemented in *PGF77* and *PGF95* compilers. 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* – Implemented *sleep3f* and *sleepqq3f* for *Win32*, 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 its preceded by '$' because a ‘C’ identifier can contain a '$'. When the source being preprocessed is an assembly file, '$' is excluded from an identifier in the preprocessor if the input file suffix is .s or .S.
- *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. It allows the compiler to perform additional optimizations.
- *Expanded gcc compatibility – PGC++ now supports* extended asm, the inline intrinsic libraries, GNU statement expressions and redefinable predefined macros *__PGIC__, __PGIC_MINOR__,* and *__PGIC_PATCHLEVEL__*. On Linux, now using the *.ctor* and *.dtor* sections for constructors and destructors instead of the *.init* and *.fini* sections.
- *Expanded cl compatibility*– On Windows, defined *__STDC__* to be compatible with Microsoft *CL*, that is, *-D__STDC__=0*. Added support for the integral constant suffixes *i64* and *ui64* and *__declspec(align(n))*, where n can be 2, 4, 8, 16, or 32. Added the GNU *__extension__* keyword 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. The 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 have been 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 the heuristic for auto-parallelization to give 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 for 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* – The latest edition of the *AMD Core Math Library,*

ACML 3.6, is bundled with the PGI 7.0 compilers on Linux and Windows.

- *Expanded OS support* – Support added for Fedora Core 6 and SuSE 10.2. On Windows, support has been added for Vista and for native compilation and execution under SFU and SUA.
- *Visual Studio 2005 SP1* – Support added 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.

3.4 Compiler Options

3.4.1 Getting Started

By default, the PGI 7.0 compilers generate code 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*. This option incorporates optimization options to enable use of vector streaming SIMD (SSE/SSE2) instructions for 64-bit targets. The contents of the *–fast* switch are host-dependent, but usually includes the options *–O2 –Munroll –Mnoframe -Mlre*. For 64-bit targets, *-fast* also includes *–Mvect=sse –Mscalarsse –Mcache_align –Mflushz*. For best performance on processors that support SSE instructions, you will want to use the *PGF95* compiler (even for FORTRAN 77 code) and the *–fastsse* option.

In addition to *–fast*, the *–Mipa=fast* option for inter-procedural analysis and optimization can improve performance. You may be able to obtain further performance improvements by experimenting with the individual *–Mpgflag* options detailed in the *PGI User's Guide* (*–Mvect*, *–Munroll*, *–Minline*, *–Mconcur*, *–Mpfi*/*–Mpfo*, etc). However, speed-ups using these options are typically application and system-dependent, so it is important to time your application carefully when using these options to ensure no performance degradations occur.

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.

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* – Introduced a new optimization level, *–O4*, that enables hoisting of guarded invariant floating point expressions.

- *–d[D|I|M|N]* – The *–d* option prints addition 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 – Simply return zero status if all flags are ok.

- *–E – pgcc –E* now preprocess .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 the librarian without a defile. (Windows only.)

- *–Mallocatable=[95|03]* – The *–Mallocatable* option controls how the compiler treats assignment of allocatables. The default 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.

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*, 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 *-B* option to accept C++ style comments in C programs. Some Microsoft header files also generate warnings about things like multiple definitions of types. In most cases these warnings may be safely ignored.

3.6.1 The Windows Command Environment

A UNIX-like shell environment (called 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 (`vi`, `sed`, `grep`, `awk`, `make`, etc). 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, you can launch a standard Windows command window pre-initialized to enable use of the PGI compilers and tools by selecting the appropriate option from the *PGI Workstation* program group accessed in the usual way through the "Start" menu.

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.

3.6.2 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. *PGI Workstation* compilers and tools can be used in any of the MKS toolkit shells. To use *PGI Workstation* in an MKS shell, you must first configure your environment. For this example, assume 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\7.0-2\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/.

3.6.2 Creating and Using Dynamic-Link Libraries

The *PGI User's Guide* for a complete description of how to build and use DLLs on Windows using the PGI compilers and tools.

Dynamic-link libraries built 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 (in particular 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. The static libraries will be used by default in the absence of *–Mdll*.

3.6 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 subtle and not-so-subtle differences 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. PGI Unified Binary object files can be use to create programs or libraries like any other object file. No special start up code is needed; support is linked in from the PGI libraries.

The *-Mpfi* 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 will generate code optimized for each listed target. For example,

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

generates optimized code for three targets.

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 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 or routine or blank. The default is routine.

For example,

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

indicates that the next function should be optimized for *k8_64*, *p7_64*, and *core2_64*.

3.7 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 modulefiles.

Assuming your installation base directory is /opt/pgi, and your MODULEPATH environment variable is /usr/local/Modules/modulefiles, execute the command:

```
/opt/pgi/linux86/7.0-2/etc/modulefiles/pgi.module.install \
        -all -install /usr/local/Modules/modulefiles
```

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

```
        module load pgi32/7.0

        module load pgi64/7.0

        module load pgi/7.0
```

where "pgi/7.0" will use the 32-bit compilers on a 32-bit system and 64-bit compilers on a 64-bit system. To see what version are available, use the command

```
module avail pgi
```

The `module load` command will set or modify 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

The Environment Modules package itself is not supported by PGI. More information about the package can be found at *http://modules.sourceforge.net*.

3.8 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 will only debug *linux86* executables. The *linux86-64* version of *PGDBG* will only debug executables built to run as *linux86-64* executables.

PGPROF is supported as a graphical and command line profiler in both 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.

3.8.1 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 is issued 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 will automatically attach to a running process at start-up. For example, entering "pgdbg -attach 1234" from a shell or command window will invoke PGDBG, which will then try to attach to the process whose PID is 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.

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

3.9 The REDIST Directories

Program 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.

3.9.1 PGI Redistributables

The PGI 7.0 release includes directories named *$PGI/linux86/7.0/REDIST* and *$PGI/linux86-64/7.0/REDIST*, *$PGI/win64/7.0-2/REDIST* and *$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), a copy of which is included in the 7.0 directory in text form for reference.

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 the requirement that end-users of the executable have properly initialized their environment and, on Linux, set LD_LIBRARY_PATH to use the relevant version of the PGI shared objects.

3.9.2 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.

3.10 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`, `.mypgcpprc`, 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.

Make the libraries found in */opt/newlibs/64* available to all *linux86-64* compilations	Add the line: `set SITELIB=/opt/newlibs/64;` to */opt/pgi/linux86-64/7.0/bin/siterc*
Make the libraries found in */opt/newlibs/32* available to all *linux86* compilations.	Add the line: `set SITELIB=/opt/newlibs/32;` to */opt/pgi/linux86/7.0/bin/siterc*
Add a new library path `/opt/local/fast` to *all linux86-64 compilations.*	add the line: `append SITELIB=/opt/local/fast;` to */opt/pgi/linux86-64/7.0/bin/siterc*
Make the include path `/opt/acml/include` available to all compilations; *–I/opt/acml/include*.	add the line: `set SITEINC=/opt/acml/include;` to */opt/pgi/linux86/7.0/bin/siterc* and */opt/pgi/linux86-64/7.0/bin/siterc*

Change *–Mmpi* to link in */opt/mympi/64/libmpix.a* with *linux86-64* compilations.	add the lines:
	
	`set MPILIBDIR=/opt/mympi/64;`
	`set MPILIBNAME=mpix;`
	
	to */opt/pgi/linux86-64/7.0/bin/siterc*

Have *linux86-64* compilations always add *–DIS64BIT –DAMD*	add the line: `set SITEDEF=IS64BIT AMD;` to */opt/pgi/linux86-64/7.0/bin/siterc*
A user wishes to build an F90 executable for *linux86-64* or *linux86* that resolves PGI shared objects in the relative directory *./REDIST*	add the line: `set RPATH=./REDIST;` to *~/.mypgf95rc*. *NOTE:* this will only affect the behavior of PGF95 for the given user.

3.11 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 7.0-2 is going 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 include 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 workaround 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* 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 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 is due to 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 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, multi-threaded programs compiled with *-Mchkstk* and -mp or *-Mconcur* will erroneously report a stack limit overflow with a message similar to stack overflow: thread 0, max 10561KB, used 2KB, request 64B The workaround is not to use *-Mchkstk* for any file in a multi-threaded program.

- 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 *–Mchkfstk*.

- Times reported for multi-threaded sample-based profiles (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 `watch`ed local variable may cause missed events and/or false positive events. Local variables may be `watch`ed reliably if program scope does not leave the scope of the `watch`ed 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 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, 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 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 (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. See http://www.pgroup.com/support/tools.htm for detailed information on how to debug a unified binary.

- *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 that 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 *-Mpfi* and *-mp* together is not supported. The *-Mpfi* 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 (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 will result in the entire process (all threads) being continued.

3.12 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 the 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*.

The following problems have been corrected in 7.0-2:

TPR	**Lang/ Tool**	**Description**
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.

TPR	Lang/ Tool	Description
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
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

TPR	Lang/ Tool	Description
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

TPR	Lang/ Tool	Description
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

TPR	**Lang/ Tool**	**Description**
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
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 by pointing your browser at either your local copy of the documentation in the release directory *doc/index.htm* or online at *http://www.pgroup.com/doc*.