PGI® Release Notes

PGPROF & PGDBG Release 5.2

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

PGI PGPROF & PGDBG 5.2 Release Notes
Copyright © 2004

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

First Printing: Release 5.2, September 2004

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

Table of Contents

1 PGI PGPROF/PGDBG 5.2 Release Notes

1.1 PGDBG and PGPROF

PGI PGPROF & PGDBG 5.2 includes several new features in the *PGDBG* parallel debugger and *PGPROF* performance profiling tools. In particular, both of these tools include completely new graphical user interfaces (GUIs).

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

PGPROF is supported as a graphical and command line profiler in both the *linux86* and *linux86-6* environments. The same version works in either the *linux86 or linux86-64* environment to process a trace file of profile data created by executing the instrumented program. Program instrumentation is either line-level (*–Mprof=lines*), function-level (*–Mprof=func*), or *gprof*-style (*–pg*) sample based and trace profiling.

The new *PGDBG* and *PGPROF* graphical user interfaces (GUIs) are invoked by default. To use the command-line interfaces, invoke either tool with the *–text* option. To use the old GUI interfaces (included in *PGI Workstation 5.1* and prior releases), invoke either tool with the *–motif* option.

1.2 PGDBG and PGPROF New Features

Following are the new features included in the *PGI PGPROF & PGDBG 5.2* versions of *PGDBG* and *PGPROF*:

- *Fortran 95 support* – *PGDBG* and *PGPROF* both support the language, syntax, and context of *Fortran 95*.

- *New Graphical User Interfaces (GUIs)* – all-new graphical user interfaces provide easier, more intuitive and effective ways to access the debugger and functionality. The *PGDBG* graphical interface supports single-threaded, multi-threaded, and distributed applications. The *PGPROF* graphical user interface supports either PGI-style pgprof.out trace files or *gprof*-style gmon.out trace files, including source correlation for *gprof*-style traces.

- *Process attach* – *PGDBG* now supports the *attach* and *detach* commands to attach and detach the debugger to or from running processes. This functionality works for MPI applications, allowing attach to all processes in the MPI application with a single attach command.

- *AMD64* call *command* – *PGDBG* now supports the call command for *linux86-64* environments, with some minor limitations (see below) in passing *F90* deferred shape array arguments.

- *Large Arrays* – *PGDBG* now supports *linux86-64* applications built with *–mcmodel=medium –Mlarge_arrays*.

- *Dynamic threads support* – In previous releases, *PGDBG* was unable to debug multi-threaded parallel programs built on some Linux distributions unless the programs were statically linked. *PGDBG* can now debug such programs even if they are dynamically linked.

- Command Mode - the default command mode has been changed to ***PGI*** mode from ***DBX*** mode. ***DBX*** commands are no longer available by default. Command modes can be switched by using the *pgienv* command.

- ***OpenMP*** Changes - the ***pgdbg*** **5.2** ***OpenMP*** event handler is disabled by default. The ***pgdbg OpenMP*** event handler sets breakpoints at the beginning and end of each parallel region, and at each thread synchronization point. This leads to a noticeable slowdown in the performance of the program when it runs under the control of the debugger. The new behavior is far less intrusive.
    - *Parallel Regions*: when the target program stops in a parallel region, ***pgdbg*** allows threads to stop asynchronously and a prompt is available immediately after a control command is issued. Line level debugging is maintained, but some threads may be running (e.g. spinning at a barrier point) while others are stopped.
    - *Serial Regions*: when the target program stops in a serial region, ***pgdbg*** instructs threads to stop synchronously and a prompt is available after all threads have stopped. In this way all worker (non-initial) threads are halted each time the initial thread is halted.

  The ***pgienv*** command can be used to define when ***pgdbg*** accepts new commands (relative to the execution state of running processes), and which threads/processes are halted when a subset of threads/processes trigger an event.

- **I/O** Redirection: you no longer need a space between **I/O** redirection specifiers and filenames; e.g. ***run >out.dat*** now works as well as ***run > out.dat***.
- *Typecast Evaluation*: ***the pgdbg expression evaluator*** now evaluates ***C/C++* typecasting correctly**.
- **NPTL** Threads - ***pgdbg*** 5.2 supports debugging NPTL thread-parallel programs. **NPTL** is a new implementation of the *pthreads* library for ***GNU/Linux,*** which is the default on some newer Linux distributions. (There are some minor issues with **NPTL** and debugger support (see below).)..

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

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

1.3 PGDBG Thread Library Limitations

PGDBG 5.2 includes support for the new NPTL threads library supported as the default pthreads library in most recent versions of Linux: ***RH 9.0***, ***SuSE 9.1***, ***RHEL3.0***, and ***Fedora Core 2***.

As with everything new, there are problems that are being worked out or around. Here are ones specific to running pgdbg with programs that use threads. There are a few issues found in using ***pgdbg*** **5.2** with this new technology:

Pgdbg Thread Support Limitations and Workarounds		
Area	**Behavior**	**Workaround**
NPTL	***"No more processes":*** when a multi-threaded process that uses **NPT**L and spawns a lot of threads is run under control of a debugger, the process may fail with an "out of processes" error. This is due to an **NPTL** issue whereby, in some cases **NPTL** threads are not cleaned up properly, leaving "*zombie*" processes (processes which are no longer running, but whose resources have not been released).	Force link with the old Linuxt pthread library by setting the LD_ASSUME_KERNEL environment variable as follows (example assumes csh): • on SuSE 9.1 **setenv LD_ASSUME_KERNEL 2.4.21** • on RH EL 3.0: on RH 9.0: **setenv LD_ASSUME_KERNEL 2.4.1** **setenv LD_ASSUME_KERNEL 2.4.19** Example: Append the name of the program to use old Linux pthread library only with that program. **setenv LD_ASSUME_KERNEL \\ 2.4.19 ./realplay**
NPTL Thread Exit	PGDBG may not recognize exit of initial thread: • *Linux86-64*: ***SuSE 9.1***, *RHEL 3.0, Fedora Core 2* • ***Linux86***: *RH 9.0, RHEL 3.0, Fedora Core 2* When the debugger traces a thread intensive NPTL program, the initial thread does not report its exit to the debugger. As noted in the previous item, the operating system does not always clean up NPTL threads properly, resulting in "zombie" processes.	To avoid this problem, exit the debugger before program exit, or set **LD_ASSUME_KERNEL** as above to avoid the use of the NPTL thread library.
Threads Library	on some **Linux** systems, a **Linux** installation bug results in installing a *pthreads* library that has been *stripped*; that is, the ***strip*** command has been used to remove symbol information from the library.	• This has been observed on *Red Hat 7.1* and *Red Hat 8.0*. **Red Hat** is aware of the problem; the bugzilla number is **110038**. • To check if your system has this problem, execute the command:

	On such systems, ***pgdbg*** cannot debug multi-threaded programs.	**`file /lib/libpthread*`** and check to see if the output says that the library has been stripped.
reading **F90/F95** *derived types*	• ***pgdbg* 5.2** does not support using the ***%*** character to access a member of a *derived type*. • if you select a member of a *derived type variable* and attempt to print it via the right-click pop-up menu, ***pgdbg*** will return an error.	• Use the *period character* (**.**) instead, e.g. use ***x.y*** instead of ***x%y*** **.** • type the ***print*** command in the *command* pane, using the (**.**) character to specify a member of a derived type variable.
GUI *Command* pane ***source***	From the **GUI** *command pane*, the *source* command does not wait for execution to stop, but continues reading commands, even if the target is running. For instance, if the *source script* contains commands to *set* a *breakpoint*, run, and print a stacktrace, the expectation might be that the stacktrace would print at the breakpoint. In fact, it might return an error, since the target could be running when *stacktrace* was executed.	Insert a ***wait*** command after each control command in the script **.**
GUI ***refresh***	***pgdbg* 5.2** *GUI* may fail to refresh the display if it is running remotely across a slow or congested network.	Hit ***<CTRL>-L*** in the *main* window or select the ***Refresh*** item under the **GUI**'s ***Options*** menu to manually refresh the display.
GUI panel disassem.	When main function or routine of the target program is not compiled with ***-g***, the **GUI** does not automatically display disassembly in the source pane.	Select ***Window->Disassembly*** and enter the name of the *main* routine...

1.4 PGBDG Command Limitations

Pgdbg Known Limitations and Workarounds

cmd	Behavior	Workaround
watch ***watchi*** ***track*** ***tracki*** ***hwatch*** ***hwatchr*** ***hwatchb***	The "watch" family of commands is unreliable when used with local variables in ***pgdbg*** 5.2. ***Note*** that calling a function or subroutine from within the scope of the watched local variable may cause missed events and/or *false positive* events.	Local variables may be watched reliably as long as program scope does not leave the scope of the watched variable. Using the "watch" commands with global or static variables is reliable.
stacktrace	*Stacktrace* can be very slow. The method used by ***pgdbg*** to retrieve function or subroutine arguments in the *stacktrace* command is currently non-optimal.	For a fast traceback without argument information, use the undocumented ***calltrace*** command. Note that *calltrace* may not be available in future releases.
stacktrace	*Stacktrace* may skip a frame in the stack if it encounters a routine compiled without **-g**. This may be most noticeable when an exception is encountered in a library routine such as ***memset***, and the *stacktrace* does not show the calling routine. The current routine may not be identified by name, showing only *unknown-addr*.	No workaround.
step	***step*** into a function in a shared library compiled **–g** fails; the command steps over the function..	set a breakpoint on the function name, then use the *cont* command to run to the breakpoint in the shared library **.** .
call	The ***call*** command does not support the following **Fortran90/95** features: • Functions returning arrays, • Functions returning pointers, • Assumed shape array arguments, • Pointer arguments.	Be careful
print	***OpenMP*** **PRIVATE** Variables: ***print*** or other means of accessing OpenMP **PRIVATE** variables may give incorrect results. Accessing **PRIVATE** variables only works in limited circumstances.	See the FAQ at http://www.pgroup.com for accessing limitations in ***OpenMp***.

Target Platform	On an **AMD64** system, you can debug either a **linux86-64** application or a linux86 application using ***pgdbg 5.2***. However, by default ***pgdbg5.2*** will assume you are debugging a **linux86-64** application	Debugging a linux86 application will not work unless you use the ***-tp k8-32*** pgdbg command line option.
run or ***rerun***	***run*** or ***rerun*** with no arguments, after a previous ***run*** or ***rerun*** that specified **I/O** redirection, continues to use the **I/O** redirection set up in the previous command. Thus • ***run* > out.dat** creates a new file, **out.dat**, containing program output. A subsequent ***run*** command without arguments will append the output of that ***run*** to **out.dat**. • ***run* < in.dat** followed by run will likely fail, since the second ***run*** will continue using **in.dat**, but the file pointer for **in.dat** will be set to the location at the point in the previous ***run*** when the new ***run*** was executed.	Care should be taken when rerunning code.
shell	• ***shell***, used after a ***run***or ***rerun*** that specified **I/O** redirection, will continue to use the I/O redirection set for the previous ***run*** or ***rerun***. Thus ***run*> out.dat** followed by ***shell ls*** will produce a file **out.dat** containing the output of the ***run***command followed by the output of the ***shell ls*** command. • The same holds true when issuing the ***shell*** command in the command prompt of the ***pgdbg*** **GUI**. However, if you are not redirecting **I/O**, then the output of the ***shell***command is sent to the **GUI**'s *Program I/O window*.	Care should be taken when using the shell command

1.5 PGPROF Features

The following table summarizes the new ***pgprof*** features in ***5.2 PGI Workstation***, as compared to previous releases.

pgprof **5.2 New Features**	
Feature	**Description**
Java *pgprof* GUI	***pgdbg***, User Interface implementation
gprof*-style sample-based* profiling	***pgprof*** supports *gprof*-style *sample-based* profiling
pgprof *gprof*-style traces	***pgprof*** supports *gprof*-style traces including routine and line-level correlation
instrumentation profiling	***pgprof*** supports *pgprof*-style instrumentation profiling
Pgprof custom GUI	***pgprof*** supports *a* user-customizable graphical user interface GUI

1.6 Tools Problems Corrected in 5.2

The following problems were corrected in the current release. Most were reported in *PGI Workstation 5.1-6* or previous releases. Problems found in *PGI Workstation 5.1-6* may not have occurred in the previous releases. A description of the problem is given, but some problems can only be described in general terms because of complexity or confidentiality. An *Internal Compiler Error* (ICE) is usually the result of checks the compiler components make on internal data structures, discovering inconsistencies that could lead to faulty code generation. The messages accompanying any ICE are cryptic and of little use to users.

Tools and compiler problems corrected are shown together.

Tool Technical Problem Reports (TPRs) Corrected in PGI Release 5.2-4				
TPR	**Rel**	**Lang/ tool**	**Description**	**Symptom**
2093	5.1	*pgdbg*	***pgdbg*** now supports **F90** pointer variables.	`Pgdbg error msgs`
2773	5.1	***pgdbg***	1. Printing allocatable arrays. 2. Source pane not updated after reload	`Display not right`
2806	5.1	***pgdbg***	print of F90 array pointer that is a field of a derived type.	`Prints only the first element`
3224	5.1	***pgdbg***	Documentation corrected to say core files are NOT supported.	`Documentation said core files supported`