

# PGI Compiler Option 一覧

## PGI コンパイラのコンパイル・オプション (2011年9月版 PGI 2011 対応)

PGI の F77, F95, HPF, C, C++ のコンパイラを使用する際のオプションを以下に示しました。以下は、pgfortran を使用した場合の例ですが、コンパイラのオプションの設定方法は、他の言語コンパイラでも同じです。なお、各オプションの詳細は、[PGI User's Guide](#) をお読みください。

```
{コマンド名} -[options] [path] filename
```

(Fortran : 例) pgf95/pgf90/pgfortran は全て同じコンパイラです。

```
pgf95 -fastsse -Minfo=all -L/opt/lib -lmylib test.f
```

```
pgf90 -fastsse -Minfo=all -L/opt/lib -lmylib test.f
```

```
pgfortran -fastsse -Minfo=all -L/opt/lib -lmylib test.f
```

(C : 例)

```
pgcc -fastsse -Mipa=fast,inline -L/opt/lib -lmylib test.cpp
```

(C++ : 例)

```
pgCC -fastsse -Mipa=fast,inline -L/opt/lib -lmylib test.cpp
```



必要とする各オプションを `-[option]` 形式でブランクを空けて指定します。また、`-M` オプションは、最適化オプションを詳細に指定するものであり、`-M` に引き続きブランクを空けずにフラグを指定します。なお、`-M` にさらに、サブ・フラグがある場合は、`-M[flag]={subflag}` の形式で指定します。**サブ・フラグを指定しない場合は、コンパイラの default 設定のサブ・フラグが使用されます。**

[options] 各コンパイル・オプションを指定する。指定順序は基本的に制約はない  
但し、ライブラリパス等の順序は重要であり、その順位で反映される

[path] リンカへのライブラリ等のパスを指定する

[filename] ソースファイル、オブジェクト・ファイル、アセンブリ・ファイル等を指定する

コンパイル・オプションの概要について以下の表に纏めました。表の中の「白抜き」の行は、最適化において、よく使用されるオプションを表しています。

- [PGI コンパイル・オプションの説明](#)
- [-M オプション\(最適化詳細オプション\)の各種フラグについての説明](#)
- [C、C++ 特有のコンパイル・オプション](#)

Copyright (C) 株式会社ソフテック

## PGI Compiler のコンパイル・オプション

オプション	記述
-#	コンパイラ手続きの呼出し情報を表示します。
-###	ドライバコマンドを表示しますが、実行しません(-dryrunと同じ)。
-Bdynamic	(Linux only) 明示的にコンパイラのドライバが、shared object library をリンクするように指示する。 (PGI 7.1 以降) Windows にも対応。PGI のランタイムライブラリの DLL をリンクします。このオプションは、コンパイル時とリンク時の両方で必要です。
-Bstatic	(Linux only) 明示的に、static libraries を使用して静的リンクを行うように指示する。 (PGI 7.1 以降) Windows にも対応。Windows 上では、実行形式モジュールにリンクされる全てのファイルは、同じオプションでコンパイル/リンクされなければなりません。また、本オプションは、コンパイル時にも必要です。Windows のデフォルトは、-Bstatic となりました。
-Bstatic_pgi	(Linux only) PGI 用の share library を静的にリンクし、システム依存のライブラリは、ダイナミック・ローディングする形式の実行モジュールを生成する(PGI 6.2 以降)。 このオプションは、-Mnorpath 機能を含む。
-byteswapio	(Fortran only) アンフォーマット Fortran データ・ファイルの入出力時にビッグエンディアンからリトルエンディアンにあるいはその逆に、バイトをスワップします。生成された実行モジュールは、自動的に read/write 処理中にこのエンディアン変換を行います。
-C	実行時の配列の境界チェックを有効にする実行モジュールを作成するように指示する。
-c	アセンブリフェーズの後で止まり、オブジェクトコードを filename.o にセーブ。

-D<arg>	プリプロセッサマクロを定義します。
-d[D M N]	<b>【PGI 7.0 以降 新設】</b> プリプロセッサからの追加情報を出力させるためのものです。 -dD : ソースファイルからマクロと値をプリントします。 -dI : インクルードファイル名をプリントします。 -dM : 前以って定義された、コマンドラインマクロを含むマクロと値をプリントします。 -dN : ソースファイルからマクロ名をプリントします。
-dryrun	コンパイル手続き上のドライバコマンドを表示しますが、実行しません。
-drystdin	(PGI 7.2新設) 標準インクルード・ディレクトリを出力して終了します。
-E	プリプロセスフェーズの後で止まり、標準出力にプリプロセスされたファイルを表示。 (PGI 7.0 以降) pgcc -E は、.h ファイルの前処理を行うようになりました。
-F	(pgf77、pgfortranとpgghpfのみ) プリプロセスフェーズの後で止まり、プリプロセスされたファイルをfilename.f にセーブ。
-f	無視されます。
-fast	一般的に最適化セット・フラグ。x86 並びに AMD64 ターゲットに対するフラグ -O2 -Munroll -Mnoframe -Mlre と同等。PGI のバージョンによって異なるため、pgf90 -fast -help でオプションの内容を確認すること。 <b>【PGI 7.0 以降の C/C++ 環境】</b> C/C++コンパイラは、-fast あるいは -fastsse の複合オプションの中に、-Mautoinline 機能も有効になるように変更されました。 <b>【PGI 7.0 以降の 64 ビット環境】</b> 64 ビットシステムのターゲットに対して、-fast オプションは、従来の -fastsse オプションと同じ機能を有するオプションに変更しました。新しい -fast オプションは、SSE 命令を伴うベクトル化、キャッシュ整理、flushz (SSEのflush-to-zeroモード) 機能を有効にします。従来の -fast と等価な機能として、-nfast というオプションが新設されました。
-fastsse	SSE/SSE2 インストラクションを有するマシンターゲットへの一般的な最適化フラグセット。x86 並びに AMD64 ターゲットに対するフラグ、-O2 -Munroll -Mnoframe -Mscalarsse -Mvect=sse -Mchache_align -Mflushz と同等 <b>【PGI 7.0 以降の C/C++ 環境】</b> C/C++コンパイラは、-fast あるいは -fastsse の複合オプションの中に、-Mautoinline 機能も有効になるように変更されました。
-flags	有効なドライバオプションとその内容を表示します。この場合は、コンパイルの実行は行われません。
-fpic	(Linux only) 他のコンパイラとの互換性を有するポジション独立のコードを生成します。Dynamic Shared Library を作成する際に使用することができる。-mcmode=medium と共には使用できません。
-fPIC	(Linux only) -fpicと同じ。
-G	リンカに共有オブジェクトファイル(ダイナミック・リンク・ライブラリ)を生成するように指示する
-g	オブジェクトモジュールにデバッグ情報を含ませます。
-gopt	オブジェクトモジュールにデバッグ情報を含ませます。最適化されたコードのデバッグが可能とするような方法でモジュールが生成されます。-goptはシンボリックデバッグ情報をオブジェクトモジュールの中に付加し、さらに、-gが指定されない時と同じ最適化コードを生成するように、コンパイラに対して指示するものです。
-g77libs	(Linux only) g77 によって生成されたオブジェクトファイルを pgfortran を使用してコンパイルされたメインプログラムにリンクする場合、このオプションを指定することで、g77 でコンパイルされたプログラム内で使用している未解決な g77 サポートライブラリを検索できるようにします。
-help	ドライバが認識する全てのオプションを標準出力に表示します。また、オプションとサブオプションの内容も表示します。-help と他のコンパイルオプションを同時に付けた場合、そのオプションの意味・内容を表示します。
-I<dirname>	ディレクトリを #include ファイルのためのサーチパスに加えます。
-i	リンカに渡されます。
-i2	2 バイトとしてINTEGER変数を扱います。(明示的にバイト数を指定しない整数宣言変数)
-i4	4 バイトとしてINTEGER変数を扱います。(明示的にバイト数を指定しない整数宣言変数)
-i8	8 バイトとしてINTEGER変数を扱い、INTEGER*8 オペレーションに 64ビットを使います。
-i8storage	INTEGER変数を 4 バイトとして扱うが、ストアする際に 8 バイトワード(64bit) としてストアする。
	特別なセマンティックをコンパイルに指示します。<flag> は多種類あるため、詳細は、User's Guide を参照。例えば、IEEE 754 に準拠するように浮動小数点演算を行う、あるいは、 <b>浮動小数点演算の例外処理の方式等の指定が可能</b> です(default は例外が起きても実行続行する)。

-K<flag>	<p>ieee / noieee : 厳密なIEEE 754に準拠する浮動小数点演算</p> <p>pic : ポジション独立のコードを生成</p> <p>trap=[subflag] : 例外が生じた場合、実行を停止させます。</p> <p>trap=none : 全てのトラップを抑制 (PGI 6.2)</p> <p>(例) <b>pgfortran -Ktrap=fp test.f</b></p>
-L<dirname>	ライブラリ・ディレクトリを指定します。これをライブラリ・サーチ・パスに加えます。
-l<library>	指定された<library>ライブラリをロードします。
-M<pgflag>	コード生成と最適化の各種のフラグ<pgflag>を指定します。フラグの指定方法は、-M<pgflag>,<pgflag>, ... or -M<pgflag>=xxxx
-m	標準出力にリンクマップを表示します。
-m32	デフォルトのプロセッサタイプとして、32ビットコンパイラを使用することをコンパイラに指示する。(PGI 10.3 新設)
-m64	デフォルトのプロセッサタイプとして、64ビットコンパイラを使用することをコンパイラに指示する。(PGI 10.3 新設)
-module <moduledir>	(F90/F95/HPF only) ディレクトリ<moduledir>にモジュールファイル(.mod)を保存/検索します。
-mcmmodel=medium	(linux86-64のみ) linux86-64 環境において、medium memory model をサポートするコードを生成します。(2GB 超えのプログラム) このオプションは、 <b>linux86-64 (64bit Linux) のみ</b> となります。Win64/osx86-64 では使用できません。
-mp[=align,[no]numa allcores,bind]	<p>ユーザーによって挿入された共有メモリ並列プログラミングディレクティブを解釈、処理します。</p> <p><b>align</b> サブオプションは、並列化と SSE によるベクトル化の両方が適用されるループにおいて、ベクトル化のためのアライメント(整列)を最大化するようなアルゴリズムを使用して、OpenMP スレッドにループ回数を割り当てるようにするものです。この機能は、このような特性を帯びた多くのループがプログラムに存在する時に性能が向上します。しかし、一方、各ループの中で、非常に大きなタスク・ワークを含むループで、そのループ長が相対的に短いプログラムにおいては、結果としてロードバランスの問題が生じて大きく性能を落とす場合がありますので注意が必要です。このオプションを適用し性能を確認してから使用してください。(この align サブオプションは、PGI 6.1 以降のオプションです)</p> <p><b>-mp=nonuma</b> (libnumaライブラリをリンクしない) PGI 6.1 以降 PGI 6.2 から libnumaを有しないシステムには、その stub(仲介)ライブラリを提供する。 (<b>PGI 8.0 以降のサブオプション</b>)</p> <p>allcores 環境変数OMP_NUM_THREADS あるいは NCPUSIにセット していない場合、すべての有効なコアを使用する (リンク時に指定すること)</p> <p>bind スレッドをコアあるいはプロセッサにバインドする (リンク時に指定すること)</p>
-noswitcherror	コマンドライン上に、コンパイラに有効なオプションではないものが指定された場合、エラーで終了させる代わりに警告レベルに変更する。この挙動は、コンパイラのサイト初期設定ファイル siterc ファイルに set NOSWITCHERROR=1 を指定することで可能となります。(PGI 7.0-4 以降) siterc ファイルは、一般に \$PGI/linux86{-64}/7.1/bin の配下にあります。 PGI 7.1 以降は、未知のオプションが指定された場合、コンパイル・エラーとなります。
-O<level>	<p>コード最適化レベルを指定します。level は 0、1、2、3 あるいは 4。</p> <p>0 : 各ステートメントに対し基本ブロックを生成します。しかし、スケジューリング並びにグローバルな最適化は行いません。</p> <p>1 : 基本ブロック内でのスケジューリング並びにいくつかのレジスタ・割当の関する最適化を行います。しかし、グローバルな最適化は行いません。</p> <p>2 : 全ての上記 レベル 1 の最適化を行います。さらに、基本ブロック間の制御フローとデータフロー解析を実施し、グローバル最適化を行います。導入変数の削除や問題のないループの移動、グローバルレジスタの割り当て等のグローバル最適化を行います。</p> <p>3 : アグレッシブなグローバル最適化を行います。全てのレベル 1, 2 の最適化だけでなく、効果のあるなしに関わらず、スカラの置き換え、より積極的な最適化を行います。</p> <p>4 : 4 レベルの最適化は、浮動小数点演算式の中で不変変数に対する巻上げ最適化を行うようになります。(PGI 7.0 以降で新設) (PGI 7.1) algebraic transformations とレジスタ・アロケーション最適化を追加しました。</p>
-o	オブジェクトファイルの名前を指定します。
-nomp	(PGI 11.0以降)PGI 11.0 から、リンク時のオプションとして、常に-mmpオプション(マルチスレッドライブラリ)がデフォルトして付加されます。これは、リンケージの時の動作ですので、コンパイル時の動作ではありません。もし、このデフォルトを変更したい場合は、新しいオプション -nomp をリンク時に指定して下さい。
-pgcppibs	PGF77 あるいは pgfortran でオブジェクトをビルドする際に、C++ ランタイムライブラリをリンクす

	るために使用します。( pgf77 あるいは pgfortran で指定する)
-pgf77libs	PGF77 でコンパイルされたオブジェクトを C あるいは C++ のメインプログラムにリンクする際に、PGF77 ランタイムライブラリをリンクするために使用します。( pgcc あるいは pgCC で指定する) (PGI 6.0~)
-pgf90libs	pgfortran でコンパイルされたオブジェクトを F77 あるいは、C、C++ のメインプログラムにリンクする際に、pgfortran ランタイムライブラリをリンクするために使用します。(pgf77 あるいは、pgcc、pgCC で指定する) (PGI 6.0~)
-P	(pgccとpgCCのみ)プリプロセスフェーズの後で止まり、プリプロセスされたファイルをfilename.ilにセーブします。
-pc	(CPU target が、px/p5/p6/piii のみ) 浮動小数点計算時の x86 アーキテクチャ上のレジスタビット長の使用精度の制御を行います。プログラムの誤差感度の評価に有効です。 -pc 32 : 単精度 (32bit) -pc 64 : 倍精度 (64bit) -pc 80 : x87 natice (80bit) このモードがデフォルトです -Kieee も参照のこと(厳密の IEEE 754 準拠)
-pg	gprof-style のサンプルベースのプロファイルデータを生成する。生成されたプロファイルデータ gmon.out ファイルは、pgprof で分析可能となる。
-Q	コンパイルステップの変化を選択します。
-R<directory>	(Linux only) リンカへ渡されます。リンク時の共有オブジェクトファイルのサーチパスの中に <directory> を入れます。これは、環境変数 LD_LIBRARY_PATH の内容を変えるものではありません。
-r	リロケータブルなオブジェクトファイルを作成します。
-r4	DOUBLE PRECISION 変数を REAL と解釈します。
-r8	REAL 変数を DOUBLE PRECISION と解釈します。
-rc file	ドライバのスタートアップファイルの名前を指定します。
-S	コンパイルフェーズの後で止まり、アセンブリ言語コードを filename.s にセーブします。
-s	オブジェクトファイルからシンボルテーブル情報を除去します。
-shared	(Linux only) リンカへ渡されます。共有オブジェクトファイルを生成するようにリンカに指示します。
-show	コンパイル起動時の各ドライバの設定パラメータ、引数の詳細を表示します。
-silent	警告メッセージをプリントしません。
-soname <library.so>	(Linux only) shared オブジェクトを生成する時、library.so(一例) というシェアードライブラリを内部の DT_SONAME フィールドへセットするようにリンカーに指示します。
-stack=nocheck	(PGI 7.1 以降) (Windows only) -stack オプションは、Windows 上で自動ランタイムスタック拡張を行わないようにすることができるように変更されました。もし、researve と commit サブオプションが、十分なスタック量を確保できるようにセットされたなら、自動的な拡張チェックは必要ありませんし、スタックのチェックを避けることができます。デフォルトは、-stack=checkです。Win64 では、デフォルトの researve 値あるいは commit 値はありません。Win32 では、researve、commit それぞれのデフォルト値は、2,097,152byteです。
-time	様々なコンパイルステップの実行時間を表示します。
-ta=nvidia (,suboption),host	(pgfortranとpgcc にのみ有効、PGI 2010以降、アクセラレータのみ) GPGPU 用のターゲット・アーキテクチャを意味します。Fortranにおける !\$ACC ディレクティブ、C における #pragma acc ディレクティブをコンパイラに認識させ、コード生成するためのです。 -ta=nvidia : NVIDIAアクセラレータをターゲットとして選択します。さらに、以下の nvidia 用のサブオプションがあります。 analysis ループの解析のみ行い、コードの生成を行いません。 cc10: compute capability 1.0 のコードを生成 cc11: compute capability 1.1 のコードを生成 cc12: compute capability 1.21 のコードを生成 cc13: compute capability 1.3 のコードを生成 cc20: compute capability 2.0 のコードを生成 (PGI 10.4以降) cuda2.3 or 2.3: CUDA toolkit 2.3 バージョンを使用 (PGI 10.4以降) cuda3.0 or 3.0: CUDA toolkit 3.0 バージョンを使用 (PGI 10.4以降) cuda3.1 or 3.1: CUDA toolkit 3.1 バージョンを使用 (PGI 10.8以降) cuda3.2 or 3.2: CUDA toolkit 3.2 バージョンを使用 (PGI 11.0以降) cuda4.0 or 4.0: CUDA toolkit 4.0 バージョンを使用 (PGI 11.6以降) nofma: fused-multiply-add命令を生成しない time: アクセラレータ領域の単純な時間情報を集積するためにプロファイル・ライブラリをリンクする fastmath : fast mathライブラリを使用。

[no]flushz: GPU上の浮動小数点演算の flush-to-zero モードを制御。デフォルトは noflushz。  
(PGI 11.5以降)  
keepgpu : kernelソースファイルを保持する。  
keepptx : GPUコードのためのportable assembly(.ptx)ファイルを保持する  
maxregcount:n : GPU上で使用するレジスタの最大数を指定。ブランクの場合は、  
制約が無いと解釈する。  
mul24 : 添字計算のために 24ビット乗算を使用  
[no]wait ホスト側での実行継続を行う際に、各カーネルが終了するまで待つ。  
(nowaitは待たない) (PGI 10.8以降)

-ta=nvidia,host : host は、アクセラレータがターゲットとして存在しないコード生成を行う。アクセラレータ領域をホスト側で実行するようにコンパイルする。PGI Unified Binaryコードを生成する。

**ターゲットプロセッサのタイプ**を指定し、そのアーキテクチャに沿ったコードを生成します。**ターゲットの default は、コンパイルを実行するシステムの「プロセッサ・タイプ」にターゲットが設定されます。**

32 ビットプロセッサ上の Pentium 4 プロセッサでは -tp p7、Pentium Pro/II/III /AthlonXP プロセッサでは -tp p6、piii/p6の混合コード生成ではジェネリック・フラグ -tp px を指定します。64ビット Athlon64/Opteron プロセッサ上では、-tp k8-64 の指定で64bit コードが生成され、-tp k8-32 を指定した場合は、32bit コードが生成される。また、インテルの64bit Xeon EM64T プロセッサに対しては、-tp -p7-64 を指定する。**コンパイルの方法は、こちらへ**

-tp <target>

amd64	AMD64 Processor	PGI6.0以前
athlon	AMD Athlon Processor	PGI6.0以前
athlonxp	AMD Athlon XP Processor	PGI6.0以前
k8-32	AMD Athlon64/Opteron 32-bit mode	
k8-64	AMD Athlon64/Opteron 64-bit mode	
k8-64e	AMD Opteron Rev.E/F Turion 64-bit mode	PGI6.1以降
barcelona	AMD barcelona/Quad-Core AMD64	PGI 7.0-3以降
barcelona-32	AMD barcelona/Quad-Core AMD64 32-bit mode	PGI 7.0-3以降
barcelona-64	AMD barcelona/Quad-Core AMD64 64-bit mode	PGI 7.0-3以降
shanghai	AMD shanghai/Quad-Core AMD64	PGI 8.0以降
shanghai-32	AMD shanghai/Quad-Core AMD64 32-bit mode	PGI 8.0以降
shanghai-64	AMD shanghai/Quad-Core AMD64 64-bit mode	PGI 8.0以降
istanbul	AMD istanbul/six-Core AMD64	PGI 9.0以降
istanbul-32	AMD istanbul/six-Core AMD64 32-bit mode	PGI 9.0以降
istanbul-64	AMD istanbul/six-Core AMD64 64-bit mode	PGI 9.0以降
bulldozer	AMD bulldozer AMD64	PGI 11.9以降
bulldozer-32	AMD bulldozer AMD64 32-bit mode	PGI 11.9以降
bulldozer-64	AMD bulldozer AMD64 64-bit mode	PGI 11.9以降
piii	Intel PentiumIII with SSE1 only	
p6	Intel Pentium Pro, II, III, AthlonXP	
p7	Intel Pentium 4/Xeon 32-bit mode	
px	Intel generic x86 mode	
p7-64	Intel Xeon/Pentium4 EM64T 64-bit mode	PGI5.2以降
core2	Intel Core 2 (Duo) 32-bit mode	PGI6.2以降
core2-64	Intel Core 2 (Duo) EM64T 64-bit mode	PGI6.2以降
penryn	Intel Penryn 32-bit mode	PGI7.2以降
penryn-64	Intel Penryn 64-bit mode	PGI7.2以降
nehalem	Intel Core i7/i5/i3(Nehalem)	PGI9.0以降
nehalem-32	Intel Core i7/i5/i3(Nehalem) 32-bit mode	PGI9.0以降
nehalem-64	Intel Core i7/i5/i3(Nehalem) 64-bit mode	PGI9.0以降

	<table border="1"> <tr> <td>sandybridge</td> <td>Intel Core i7/i5/i3(SandyBridge)</td> <td>PGI11.6以降</td> </tr> <tr> <td>sandybridge-32</td> <td>Intel Core i7/i5/i3(SandyBridge) 32-bit mode</td> <td>PGI11.6以降</td> </tr> <tr> <td>sandybridge-64</td> <td>Intel Core i7(SandyBridge) 64-bit mode</td> <td>PGI11.6以降</td> </tr> <tr> <td>x64</td> <td>AMD64/EM64Tの両方に対応可能とした最適化を施す Unified Bynary の生成(-tp p7-64,k8-64 と同意)</td> <td>PGI6.1以降</td> </tr> </table> <p>64-bit Unified Bynary を作成する場合は、-tp x64 あるいは、以下のように明示的な複数のターゲット名を指定しなければならない。</p> <p><b>【PGI 7.0 以降の 64 ビット環境】</b></p> <p>-tp オプションは、コンマ(,)で区切られた複数の64ビット・ターゲットを記述する方式を採用しました。以前のバージョンでは、これは一つのターゲットのみの記述方式でした。もし、複数のターゲットが指定された場合、Unified binary は、各ターゲットに対して最適化されたコードを生成します。</p>	sandybridge	Intel Core i7/i5/i3(SandyBridge)	PGI11.6以降	sandybridge-32	Intel Core i7/i5/i3(SandyBridge) 32-bit mode	PGI11.6以降	sandybridge-64	Intel Core i7(SandyBridge) 64-bit mode	PGI11.6以降	x64	AMD64/EM64Tの両方に対応可能とした最適化を施す Unified Bynary の生成(-tp p7-64,k8-64 と同意)	PGI6.1以降
sandybridge	Intel Core i7/i5/i3(SandyBridge)	PGI11.6以降											
sandybridge-32	Intel Core i7/i5/i3(SandyBridge) 32-bit mode	PGI11.6以降											
sandybridge-64	Intel Core i7(SandyBridge) 64-bit mode	PGI11.6以降											
x64	AMD64/EM64Tの両方に対応可能とした最適化を施す Unified Bynary の生成(-tp p7-64,k8-64 と同意)	PGI6.1以降											
-[no]traceback	環境変数 PGI_TERM のスイッチにより異常終了時のトレースバックの処理を制御できますが、その際に必要なデバッグ情報を加えます。なお、FortranコンパイラのデフォルトはONであり、C/C++のデフォルトは OFF として設定されています。												
-U symbol	プリプロセッサマクロを #undef します。												
-u symbol	リンカーにとって未定義なものとしてシンボルテーブルを symbol で初期化します。未定義シンボルは、アーカイブライブラリ上の最初のメンバーのローディングを引きおこします。												
-V{Release_Number}	バージョンメッセージ、及び、他の情報を表示します。-V に続けてシステムにインストールしている過去のバージョンを指定した場合、デフォルトバージョンではなく、そのバージョンのコンパイラを使用してコンパイルされます。 (例) pgfortran -V5.2 test.f (PGI 7.1 以降) プロセッサ名をプリントするようになりました。例えば、Core 2 Duo上でコンパイルすると-Vオプションは、-tp core2-64と表示します。												
-v	コンパイラ、アセンブラ、及び、リンカフェーズ呼出しを表示します。												
-W	引数を特定のフェーズ(コンパイル、アセンブラ、リンカ)に渡します。 -W{0,1,l}, <option>, <option> 形式:0 はコンパイラ、1 はアセンブラ、l はリンカ												
-w	警告メッセージを表示しません。												

### -M オプションの各種フラグ

pgflag	記述	カテゴリ
allocatable=[95/03]	<b>【PGI 7.0 以降 pgfortranで 新設】</b> -Mallocatable= オプションは、コンパイラがメモリ割付(allocatable)に係る意味合いを制御します。デフォルトは Fortran 95 に準拠します。=03 オプションは、Fortran 2003 に準拠します。	Fortran95 言語
anno	アセンブリコードと共にソースコードを注釈する。-Manno -S の指定により、アセンブラ・リスティング・ファイル xxxx.s の中にソース・リストとそれに対するアセンブラアセンブラ・リストが両方表示される。	その他
[no]asmkeyword	(pgccとpgCCのみ)コンパイラが C/C++ ソースファイルにのちに asm キーワードの挿入を許すかどうかを指定。asm キーワードの構文は以下のとおり。 asm("statement") ; statement はアセンブラ言語による文であり、ダブル・クォツで囲むことが必要。	C / C++ 言語
[no]autoinline[=levels:n   maxsize:n   totalsize:n ]	(PGI 6.2 以降)最適化オプション -O2 以上において、C/C++ コンパイラはインラインキーワードで定義されたもの、あるいはクラス実体(class body)で定義された関数をインライン化する。-Mnoautoinline は、このインライン化を抑制する。levels:n は、インラインの段数(レベル)の数の制約値を指定します。そのデフォルトは4です。 (PGI 2010 以降の C/C++ コンパイラ) -O2 オプション時にインライン化をコンパイラに指示する。 levels:n : インラインを行うレベル階層を最大 n まで行うことを指示。 デフォルトは10 です。 maxsize:n : nサイズを超える関数のインラインを行わない。 デフォルトは100。	インライン C / C++ 言語

	totalsize:n : インライン対象が、n サイズ時にインラインを止めることを指示。デフォルトは800。	
[no]backslash	(pgf77、pgfortranとpgphpfのみ) backslash キャラクタが quote された文字列において escape キャラクタとして扱うかを決定。	Fortran 言語
[no]bounds	実行時の配列の境界チェックを有効にするか、無効するかを指定。プログラムのデバッグ時に非常に有効である。例えば、配列境界外のアクセスを行った場合、以下のような形式で出力される。 PGFTN-F-Subscript out of range for array a (a.f: 2) subscript=3, lower bound=1, upper bound=2, dimension=2	その他
[no]builtin	(pgccとpgCCのみ) 数学サブルーチンのビルトインサポート(選択された数学ライブラリルーチンをインライン化する)を用いてコンパイルする[しない]。	最適化
byteswapio	FortranアンフォーマットデータのI/O時にバイトオーダーをスワップ(リトルエンディアンからビッグエンディアンに、またその逆)する。	その他
cache_align	可能な限り、16バイト以上のデータオブジェクトをキャッシュラインに整列させる。特に、SSE/SSE2 のベクトル化を行う際に有効(必須)である。	最適化
chkfpstk	関数の開始時と、関数またはサブルーチン呼び出しから戻った後での x86 FPスタックの内部の一貫性についてチェック。 実行時に環境変数 PGI_CONTINUE=verbose のセットを行うと警告メッセージが出る。	その他
chkptr	(pgfortranとpgphpfのみ) NULLポインタについてチェック。	その他
chkstk	パラレル領域のエントリ時と、パラレル領域の開始前にエントリ上のスタックの利用可能なスペースをチェック。多くのプライベートな変数が宣言されるときに有益。 (PGI 7.1 以降) -Mchkstk オプションでコンパイルされたプログラムは、スタック high-water mark の情報を収集できるように指示できます (Windows版のみ)。もし、環境変数 PGI_STACK_USAGE が実行時にセットされた場合、スタックの high-water mark が実行終了時に印字されます。	その他
concur[=flag[,flag,...]]	ループの <b>自動並行化</b> を有効にします。複数のプロセッサにより並列化できるループの並列性を確認し、可能な限り並列化する(共有メモリアルチCPUシステムのみで有効)。以下のサブ・フラグがありますので、詳細は User's Guide を参照のこと。 altcode:n / noaltcode dist:block / dist:cyclic cncall assoc/noassoc [no]innermost (最内側ループの並列化) PGI 6.1以降 nonuma (libnumaライブラリをリンクしない) PGI 6.1 以降 <b>(PGI 8.0 以降)</b> allcores 環境変数OMP_NUM_THREADS あるいは NCPUSにセットしていない場合、すべての有効なコアを使用する (リンク時に指定すること) bind スレッドをコアあるいはプロセッサにバインドする (リンク時に指定すること)	最適化
cpp=[option]	後続のコンパイル手続きを行わずに、PGI cppライクのプリプロセッサを実行する。このオプションは、makefileの中を含む各ルーチンの依存情報を生成する際に有効です。optionは、以下に示す一つあるいは複数の文字列(m, md, mm, mmd)からなる。もし、これらの複数のオプションが指定された場合は、最後にリストされたオプションのみが有効となる m : makefile dependenciesをstdoutに出力する。 md : makefile dependenciesをfilename.dと言うファイルに出力します。ここでfilenem.dとは、コンパイルする入力ファイル名のルート部分の名前が採用される。 mm : makefile dependenciesをstdoutに出力しますが、システムincludeファイルは無視する。 mmd : makefile dependenciesをfilename.dと言うファイルに出力します。ここでfilenem.dとは、コンパイルする入力ファイル名のルート部分の名前が採用される。なお、システムincludeファイルは無視する。 [no]comment : プリプロセス処理の出力のコメントを残す(さない)。 [suffix:] : makefile dependenciesを含むファイルの添字として<suff>を使	その他

	<stuff> 用する	
cray	(pgf77、pgfortranとpghpfのみ)Cray Fortran(CF77)互換性を強制。	最適化
cuda=[option]	(pgfortranのみ、PGI 2010以降、アクセラレータのみ) コンパイラに CUDA Fortranを使用できるように指示する。以下のサブオプションを有する。 cc10 :compute capability 1.0のコード生成。 cc11 :compute capability 1.1のコード生成。 cc12 :compute capability 12のコード生成。 cc13 :compute capability 1.3のコード生成。 cc20: compute capability 2.0 のコードを生成 (PGI 10.4以降) cuda2.3 or 2.3: CUDA toolkit 2.3 バージョンを使用 (PGI 10.4以降) cuda3.0 or 3.0: CUDA toolkit 3.0 バージョンを使用 (PGI 10.4以降) cuda3.1 or 3.1: CUDA toolkit 3.1 バージョンを使用 (PGI 10.8以降) cuda3.2 or 3.2: CUDA toolkit 3.2 バージョンを使用 (PGI 11.0以降) cuda4.0 or 4.0: CUDA toolkit 4.0 バージョンを使用 (PGI 11.6以降) fastmath : fast mathライブラリを使用 (PGI 10.4以降) [no]flushz: GPU上の浮動小数点演算の flush-to-zero モードを制御。デフォルトは noflushz。(PGI 11.5以降) nofma: fused-multiply-add命令を生成しない (PGI 10.4以降) emu : CUDA Fortran エミュレーションモード keepbin: kernelバイナリファイルを保持し、ファイル(.bin)として出力する。(PGI 10.3 以降) keepgpu : kernelソースファイルを保持する(xxxx.gpuファイル)。(PGI 10.3 以降) keepptx : GPUコードのためのportable assembly(.ptx)ファイルを保持する。 maxregcount:n : GPU上で使用するレジスタの最大数を指定。空白の場合は、制約が無いと解釈する。 ptxinfo : コンパイル時にPTXAS情報メッセージを表示する。(PGI 11.0以降)	CUDA Fortran言語
cuda <sub>x86</sub>	(PGI 11.5新設、pgcpp のみ) CUDA C++ プログラムを PGI C++ コンパイラでコンパイルして、この実行バイナリをインテルやAMDの x86 プロセッサ上で実行できる <b>PGI CUDA C for Multi-core x86</b> 機能を有効にする。	C++言語
[no]daz	IEEE 754 正規化されていない数字(内部表現)に対して、zero セットすることを許可する(しない)オプション。(PGI6.0) このオプションは、メインプログラムに対して適用しなければ有効とならない。 PGI 6.2 以降、64ビット EM64T の場合は、-Mdazがデフォルトとし、AMD64の場合は、-Mnodaz がデフォルトとなる。	最適化
[no]dclchk	(pgf77、pgfortranとpghpfのみ) 全てのプログラム変数が宣言されていることを前提としてチェックする(しない)。	Fortran言語
[no]defaultunit	(pgf77、pgfortranとpghpfのみ) どのようにアスタリスクキャラクタ"*"が(I/O ユニット 5 と 6 の状態に関係なく)標準入力、および、標準出力と関連して扱われるかを決定。	Fortran言語
[no]depchk	潜在的なデータ依存性が実際に存在することをコンパイラに指示してチェックを行う。一方、nodepchk は依存性がないことをコンパイラに指示する(もし、この場合、存在した場合は不正確な結果となる)。	最適化
[no]dlines	(pgf77、pgfortranとpghpfのみ)コンパイラが実行可能なステートメントとしてカラム11に"D"を含む行を扱うかどうかを決定。	Fortran言語
dll	(Windows only) ランタイムライブラリの DLL バージョンとリンクする。 (PGI 7.0以降)-Mdll オプションは、-D_DLL 機能を含意します。-D_DLL は、プリプロセッサ・シンボル _DLL を定義するものです。 ((PGI 7.1以降)-Mdll オプションが削除されました。その代わりに、-Mdynamic オプションを使用します。	その他
dollar, char	コンパイラがドル記号コードをマップする際の文字(char)を指定。ドル記号を名前として使用することを許す。ANSI C は許さない。	Fortran言語
[no]dse	【PGI 7.0 以降】 参照しない変数の保存を排除(dead store eliminations)する最適化を有効[無効]にするオプションです。これは、C++プログラムのようなパフォーマンス向上のために、関数呼び出しをインライン化することが多い場合に有効となります。	最適化
dwarf1   dwarf2   dwarf3	DWARF1 あるいは DWARF2、DWARF3 フォーマットのいずれかのデバッグ情報を生成する。デフォルトは、DWARF2 である。-g とともに使用する。	コード生成

nodwar	(PGI 8.0以降) デバッグ情報を生成バイナリに付け加えないように指示する。	コード生成
eh_frame, noeh_frame	(PGI 2010以降) リンカーに、executable内のen_frameのcall frame を保持/非保持することを指示する。(注意) このオプションは、システムunwindライブラリを持つ、最新のLinux、Windowsシステムでのみ有効です。	コード生成
extend	(pgf77、pgfortranとpghpfのみ) コンパイラは、132 カラムソースコードを受け付けます。デフォルトでは 72 カラムコードを受け付けます。	Fortran言語
extract[= <i>flag</i> [, <i>flag</i> ,...]]	関数エキストラクタを起動。コマンドライン上で指定されたファイルから関数を抽出し、指定した外部 directory へその関数ファイルを生成、追加する。インライン(-Minline) とともに使用する場合が多い。以下のフラグがありますので、詳細は User's Guide を参照のこと。 name:func size:number lib:dirname	インライン化
fcon	(pgccとpgCCのみ) 浮動小数点定数を倍精度型の代わりに、float 型として扱うようにコンパイラに指示。	C / C++言語
fixed	(pgfortranとpghpfのみ) F77スタイルの固定フォーマットのソースであると認識する	Fortran言語
[no]flushz	SSE/SSE2 を flush-to-zero モードにセットする。浮動小数点のアンダーフローが生じた場合、これを 0 にセットする。このオプションは、メインプログラムに対して適用しなければ有効とならない。	最適化
free	(pgfortranとpghpfのみ) コンパイラは F90 形式のフリーフォーマットのソースコードであると仮定する。	コード生成
[no]func32	32 Byte 上で全ての関数をアライン(整列)させる。	その他
[no]fpapprox [= <i>div</i>   <i>sqrt</i>   <i>rsqrt</i> ]	(PGI 7.1 以降) 特定の単精度浮動小数点演算を低精度近似方を使用して実行します。このオプションは結果の差異が生じる可能性がありますので、十分注意して使用してください。 div : 浮動小数点除算近似 sqrt : 浮動小数点平方根近似 rsqrt : 浮動小数点逆数平方根近似 デフォルトでは、-Mfpapprox は使用されません。もし、サブ・オプションを指定しない -Mfpapprox のみの場合は、上記の全てのサブ・オプションが指定されたものとして扱います。 (PGI 8.0 追加) -Mnofpapprox : 低い精度の浮動小数点演算を使用しないように指示する。	最適化
[no]fpmisalign	(PGI 7.1 以降) AMD barcelonaプロセッサに対して、16-byte境界に整列されていないアドレスを持つメモリ・オペランドのベクトル演算命令の使用を許可します。デフォルト設定は、Barcelonaを含めて、全てのプロセッサにおいて-Mnofpmsalignです。本オプションは、-tp barcelona-64あるいは、-tp barcelonaの設定時、あるいは、barcelona上でコンパイルされたときにのみ効果があります。また、このオプションでコンパイルされたコードは、barcelonaプロセッサ上だけで実行できるものとなりますのでご注意ください。	最適化
[no]fprelaxed= [ <i>div</i> , <i>order</i> , <i>rsqrt</i> , <i>sqrt</i> : <i>recip</i> ]	いくつかの内部組み込み関数 (div/sqrt/rsqrt) の計算において緩い精度で行うことをコンパイラに指示する。性能は向上するが、計算精度は劣る。(PGI 6.1 以降) デフォルトは、-Mnofprelaxed。 PGI 6.2 以降、細かな制御を行うためのサブオプションを導入。サブオプションは以下のとおりです。-Mfprelaxed=[ <i>div</i> , <i>rsqrt</i> , <i>sqrt</i> ] div : 緩い精度で除算処理を行う。 order : $a*b+a*c$ を $a*(b+c)$ と変換する方式も含め、演算の順序の変更 noorder : 上記 order を行わない。 rsqrt : 緩い精度でsqrtの逆数近似(1/sqrt)の処理を行う sqrt : 緩い精度でsqrtの処理を行う。 なお、サブオプションを付加しない場合(-Mfprelaxedのみ)は、そのターゲットプロセッサに応じて、顕著な性能向上が行える処理に緩い精度での処理を行うかを選択し適用される。 (PGI 9.0 新設) recip : 緩和した精度で逆数近似	最適化
[no]i4	(pgf77、pgfortranとpghpfのみ) どのようにコンパイラが INTEGER 変数を扱うかを決定。i4 の場合、INTEGER*4、noi4 の場合は、INTEGER*2 として扱う。	最適化

<p>iface=unix   cref   mixed_str_len_arg   nomixed_str_len_arg</p>	<p>(PGI 7.2 新設 Windowsのみ)サブオプション-MifacはFortranのための呼び出しルール(コンベンション)を調整するものです。          unix - Use UNIX calling conventions, 語末のアンダースコアがないタイプ          cref - Use CREF calling conventions, 語末のアンダースコアがないタイプ          mixed_str_len_arg - 文字列の長さをその対応する引数の直後に置くタイプ          nomixed_str_len_arg - 文字列の長さを引数リストの最後に置くタイプ。</p>	Fortran言語
<p><a href="#">info[=flag[,flag,...]]</a></p>	<p>コンパイル時に最適化並びにコード生成に関するコンパイル・メッセージを標準出力に表示する。以下のサブ・フラグがありますので、詳細は User's Guide を参照のこと。</p> <p>all          inline          ipa          loop          mp          opt          time          unroll</p> <p>(PGI 7.2新設)          intensity ループ内の「演算密度」(Computational Intensity)を表示します。デフォルトは、最内側ループの情報が表示されます。演算密度とは、一般にループ内の演算数とメモリのロード・ストア数との比率を表し、演算とメモリ参照のバランスを見るための指標です。このような情報はパフォーマンス・チューニングにおいて特に重視されます。</p> <ul style="list-style-type: none"> <li>ループ内の演算が浮動小数点演算である場合、演算密度は、浮動小数点演算総数を浮動小数点データのメモリロードとストアの総和で割った比率として定義します。</li> <li>ループ内の演算が整数演算である場合、演算密度は、整数演算総数を整数データのメモリロードとストアの総和で割った比率として定義します。</li> </ul> <p>(PGI 8.0 以降新設)          all 以下のサブオプションをすべて指定したものと解釈します。          -Minfo=accel,inline,ipa,loop,lre,mp,opt,par,unified,vect</p> <p>accel アクセラレータ情報の有効化          ccff オブジェクトファイルに最適化情報を追加します          ftm Fortran特有な情報の有効化          hpf HPF特融な情報の有効化 information          inline インライン情報の有効化          lre LRE情報の有効化          par 並列化の情報の有効化          pfo プロファイル・フィードバックに関する情報の有効化          vect ベクトル化の情報の有効化</p> <p>(PGI 9.0 新設)          accel アクセラレータ領域をGPU Kernel に翻訳することが成功したかどうかの情報を示す</p>	その他
<p>inform,<i>level</i></p>	<p>指定した <i>level</i> 以上のエラー・メッセージを表示するように指示。</p> <p>fatal : fatal error messages.          severe : severe and fatal error messages.          warn : warning, severe and fatal error messages          inform : all error messages          (inform, warn, severe and fatal)</p>	その他
<p><b>inline</b>  <a href="#">[=func   filename.ext   number   levels:number],...</a></p>	<p>関数のインライン展開を行う。以下のサブ・フラグがありますので、詳細は User's Guide を参照のこと。</p> <p>except:func : IPA(-Mipa) のインライン機能にも影響する。          [name:]func          filename.ext          Number          levels:number</p> <p>PGI 7.1以降) 配列の形態(Array shape) が一致しない場合でも Fortran におけるインライン処理を許可(抑止)する。-Mconcur あるいは -mp の場合を除いたデフォルトは、-Minline=noreshape。-Mconcur あるいは -mp の場合のデフォルトは、Minline=reshape。</p>	インライン化
<p>instrument [=functions]</p>	<p>(PGI 9.0以降、linux86-64 にのみ)          Common Compiler Feedback Format (CCFF)を使用して、PGI コンパイラは、どのようにプログラムの最適化を行ったら良いか、あるいは、特定の最適化がなされないのか等の関数レベルの instrument 情報をオブジェクトに保持することを可能とする。-Minstrument=functions の指定も -Minstrumentと同じ意</p>	その他

	<p>味なる。このオプションは、-Minfo=ccff -Mframe の二つを指定したことに同意です。</p>	
ipa[= <i>flags</i> ]	<p>関数、サブルーチン間のグローバルな最適化を行うために、内部手続き間の最適化を行うように指示。version 5.2 から 1パスで行うことが可能。この ipa は同時に -O2 のレベルで行うことを前提にしている。以下のサブ・フラグ <i>flags</i> の詳細は、User's Guide を参照のこと。<b>一般的には、-Mipa=fast を指定すると良い。</b></p> <p>[no]align  [no]arg  [no]const Interprocedural constant propagation  [no]cg  except:&lt;func&gt;  [no]f90ptr  fast  force  [no]globals  inline:&lt;n&gt;  inline  ipofile  [no]keepobj  [no]libc  [no]libinline  [no]libopt  [no]localarg  main:&lt;func&gt;  noerror  [no]ptr  [no]pure  required  safe:[&lt;function&gt; &lt;library&gt;]  [no]safeall  [no]shape  summary  [no]vestigial</p> <p>-Mipa Default enables constant propagation</p> <p><b>複合フラグ fast の意味は</b></p> <p>-Mipa=align,arg,const,f90ptr,shape,globals,localarg,ptr</p> <p><b>PGI 6.0 New flag:</b></p> <p>-Mipa=[...,safe:&lt;libname&gt;,safeall,...] — IPA機能を使用してコンパイルして生成していない、ライブラリ名 libname 中のプログラムユニットへの呼び出しが安全であると仮定する、あるいは呼び出し側におけるIPA最適化を禁止しないことをコンパイラに指示するためのオプションです。-Mipa=safeall は実行モジュールの中にリンクされる全てのライブラリが安全であることコンパイラに指示します。</p> <p><b>PGI 6.1 New flag:</b></p> <p>-Mipa=cg <b>スイッチ</b>を設定することで、プログラムのコール・グラフ情報を出力できるようになりました。これは、新規に提供されたpgicgコマンド・ユーティリティを使用して、出力可能です。</p> <p>-Mipa=except:&lt;func&gt; — IPA最適化において、インラインすべきでない関数funcを指定します。-Mipa=inlineと共に指定します。デフォルトは、内部的に検出されたすべての関数がインライン対象となります。</p> <p><b>PGI 6.2 New flag:</b></p> <p>-Mipa=[no]libc は、システム標準Cライブラリ内で、あるルーチンへの呼び出しを最適化するために使用します。-fastオプション時のデフォルトは-Mipa=libcです。nolibcは、その機能を抑止します。</p> <p><b>PGI 7.1 New flag:</b></p> <p>-Mipa=[no]reshape は、配列の形態(Array shape) が一致しない場合でも Fortran におけるインライン処理を許可(抑止)します。</p> <p><b>PGI 7.2 New flag:</b></p> <p>-Mipa=jobs:&lt;n&gt; — jobs:[n] サブオプションを指定できるようになりました。このサブオプションは、並列に n ジョブで再コンパイルを行うように指示するものです。</p> <p><b>PGI 9.0 New flag:</b></p> <p>-Mipa=nopfo は、プロファイル・フィードバック情報の引用回数情報を無視する。このサブオプションは、inlineサブオプションの次に指定されているときのみ有効。-Mipa=inline,nopfo は、IPA手続きに対して、PFO情報が有効な状</p>	最適化

	態において、インラインされる関数を決める際に、PFO情報を無視するように伝えます。	
noipa	内部手続き間解析と最適化機能を抑制します。機能複合オプションの後に、このオプションを指定した場合、他の機能に関しては影響せず、IPA最適化のみを抑制することができます。(PGI 6.0)	最適化
[no]iomutex	(pgf77、pgfortranとpghpfのみ) クリティカルセクションがFortran I/Oコールの周辺で生成されるかどうかを決定。	Fortran言語
[no]large_arrays	(AMD64、EM64T & Linux86-64) 2GBを超える「単一の静的なデータオブジェクト」を扱うことができるコードを生成します。PGI 5.2 の場合は、pgfortran、PGF77、PGCC でサポートします。一般的には、-mcmode=medeium と同時に使用します。 <b>PG 6.0 以降</b> では、2GB以上の単一の静的なデータオブジェクトをサポートするために指定する有効化(無効化)フラグです。pgfortran、PGF77、PGCC、PGC++ の言語でサポートします。なお、このオプションは、PGI 6.0 から -mcmode=medium の複合オプションの中に加えられました。2GB以上の単一の静的なデータオブジェクトを使用するアプリケーションでは必要とされるオプションです。	コード生成
largeaddressaware[=no]	(PGI 7.2新設: Windowsのみ) Windows x64 用に64bitアドレス・インデックシングと単一のデータオブジェクトのサイズが2GB以上使用できるようにする	その他
[no]loop32	(PGI 7.1 以降) barcelona上での 32-byte 境界上にある最内側ループを整列します。barcelona上で 32-byte 境界で整列されている場合、小さなループは性能が向上する可能性があります。しかし、実際には、ほとんどのアセンブラが、まだ効果的なパディング(padding)を実装していません。その結果、このオプションで遅くなる可能性もあります。Barcelonaに対して最適化されたアセンブラを有するシステム上でこのオプションを使用してください。デフォルトは、-Mnloop32 です。	最適化
lsf	(32-bit Linux) 32ビットシステム上で 2GB 以上のファイル I/O を扱うためのライブラリをリンクする。	環境
lre[=array   assoc   noassoc] [no]lre	ループ内での冗長性を削除する最適化の有効化 [無効化]。 array : 個々の配列要素の参照を冗長性削減の対象として扱う。デフォルトは、2以上のオペランドを含む冗長式のみが対象となる。 assoc : 冗長性削減の対象を増やすことができる、演算式の再結合を許す最適化。結果の差異が生じる可能性がある。 noassoc: 上記を許さない最適化	最適化
keepasm	アセンブリファイルを保持するようにコンパイラに命令。ファイル名は、<filenema>.s。	その他
[no]list	コンパイラがリスティング・ファイルを作成するかどうかを指定。ファイル名は、<filenema>.lst。	その他
[no]m128	(PGI 9.0 新設 pgcc のみ) __m128, __m128d, __m128iデータ型を認識するためオプション	その他
makedll[=export_all]	(Windows only) Dynamic Link Library (DLL) を生成する。=export_all は、DLL内の全ての関数をエクスポートする。 Windows 上での DLL の作成に関しては、PGI User's Guide の 8 章を参考のこと。 (PGI 7.1 以降) -Mmakedll オプションは、-Mdynamic オプションを内包します。	その他
makeimpdll[=export_all]	(Windows only) DLL を生成することなしに、import ライブラリを生成する。 =export_all は、DLL内の全ての関数をエクスポートする。	その他
makeimplib	(Windows only : PGI 7.0 以降) DLLを生成することなしに、インポートライブラリを生成します。これは、まだ、それ自身のDLLライブラリが構築される前に、DLLのためにインポートライブラリを生成したい時に使用します。	その他
mpi[=option]	(MPI 使用可能ライセンスのみ:PGI 7.1 以降) プログラムのビルドに使用する MPI ライブラリの指定を行う。 -Mmpi のみの場合は、MPICH-1ライブラリを使用。 mpi=mpich1 : MPICH-1 ライブラリ mpi=mpich2 : MPICH-2 ライブラリ mpi=mvapich1: MVAPICH-1 ライブラリ	コード生成

	<p>mpi=msmpi : Windowsのみ MSMPIライブラリ  mpi=hpmpi : HP-MPI ライブラリ (PGI 8.0 以降)</p>	
<p>neginfo [= flags]</p>	<p>なぜ、最適化が行われないかに関する情報を生成するようにコンパイラに指示。  all : 全てのメッセージ出力  concur : 自動並列化できない理由  loop : メモリ階層型の最適化ができない理由  (PGI 8.0 以降)  all 以下のサブオプションをすべて指定したものと解釈します。  -Mneginfo=accel,inline,ipa,loop,lre,mp,opt,par,vect  accel アクセレータ情報の有効化  ftn Fortran特有な情報の有効化  hpf HPF特有な情報の有効化 information  inline インライン情報の有効化  ipa IPA 情報の有効化  lre LRE情報の有効化  mp OpenMP情報の有効化  opt 最適化の情報の有効化  par 並列化の情報の有効化  pfo プロファイル・フィードバックに関する情報の有効化  vect ベクトル化の情報の有効化</p>	その他
<p>names=lowercase uppercase</p>	<p>Fortran外部関数名の大文字/小文字を指定する。Lowercaseの場合は、小文字を使用すると言う意味となり、uppercaseは大文字を使用すると言う意味となる。(PGI 7.2新設)</p>	その他
<p>noframe</p>	<p>関数の真のスタック・フレームポインタのセットアップ処理を消去するように指示。このオプションを有効化すると traceback 機能を使用することができない。</p>	最適化
<p>nomain</p>	<p>(pgf77、pgfortranとpghpfのみ)リンクステップ時に、Fortran のメインプログラムを呼ぶオブジェクトファイルを含めない形でリンクする。C プログラムと Fortran プログラムのオブジェクトの混在したものをリンクする時、C プログラムにメインプログラムが存在している場合で、かつ pgf77、pgfortran でリンクする時に使用する。</p>	コード生成
<p>[no]movnt</p>	<p>non-temporal ストア並びにプリフェッチの生成を強制するオプション。これまで使用してきた -Mnontemporal を置き換えるものです。(PGI 6.1)</p>	コード生成
<p>nontemporal</p>	<p>non-temporal ストア並びにプリフェッチの生成を強制するオプション。</p>	最適化
<p>noopenmp</p>	<p>-mp オプションと同時に使用した場合、強制的に OpenMP directives を無視するようにコンパイラに指示する。但し、SGI スタイルの並列 directive は解釈する。</p>	その他
<p>[no]prefetch (PGI5.2まで)  [no]prefetch  [=d:&lt;m&gt;[,n:&lt;p&gt;[, {nta   t0   w}]]] (PGI6.0以降)</p>	<p>prefetch インストラクションの生成を有効化/無効化する。-Mvect (-fastsse) オプションと共に使用する。  <b>PG 6.0 以降</b>では、メモリデータのプリフェッチ命令を生成することを有効化(無効化)します。このオプションは、-Mvect あるいは、-Mfastsse (-Mvectを含む複合オプション)と組み合わせて使用する場合のみ有効です。新たなサブオプションである、d:&lt;m&gt; <b>距離サブフラグ</b>は、アクセスしようとするデータの先にある m キャッシュラインの長さをプリフェッチするようにコンパイラに指示します。n:&lt;p&gt; <b>数サブフラグ</b>は、プリフェッチが使用されている場所において、最大 p プリフェッチ命令まで出すことができるようにコンパイラに指示するものです。また、新たな nta   t0   w の各サブオプションは、プリフェッチのために、prefetchnta、prefetch0、prefetchw 命令を使うようにコンパイラに指示するものです。なお、prefetchw は、IA32 あるいは EM64T プロセッサではサポートしません。</p>	最適化
<p>nopgdllmain</p>	<p>(Windows only) デフォルトの DllMain() を DLL の中に含んでいるモジュールをリンクしない。このフラグは、pgfortran によるDLL の構築に対して適用される。</p>	その他
<p>norpath</p>	<p>(Linux only) PGI の 共有ライブラリ・オブジェクトを含むディレクトリパス名を -rpath オプションをリンク時のコマンド行に付加しない。(デフォルトは -Mrpath で含む)</p>	その他
<p>nosgimp</p>	<p>-mp オプションと同時に使用した場合、強制的に SGI スタイルの並列 directive を無視するようにコンパイラに指示する。但し、OpenMP directives は解釈する。</p>	その他
	<p>(pgf77、pgfortranとpghpfのみ) 標準のスタートアップルーチンをリンクしな</p>	

nostartup	い。	環境
nostddef	標準のプリプロセッサマクロを認識しないようにコンパイラに指示。	環境
nostdinc	インクルードファイルの標準の場所を検索しないようにコンパイラに指示。	環境
nostdlib	標準のライブラリをリンクしないようにリンクに指示。	環境
[no]onetrip	(pgf77, pgfortranとpghpfのみ) 各 DOループが少なくとも 1 回実行させるかさせないかの指示。	言語
novintr	イディオム認識を抑制し、最適化されたベクトル関数の呼び出しを行う。	最適化
pfi	-Mpfo 最適化オプションを含む後続のコンパイル時において使用されるプロファイルとデータ・フィードバック情報を集めるための実行モジュールを生成するためのオプションです。-Mpfi を伴う実行モジュールはこの情報を集積するためのオーバーヘッドが発生するため、実行時間が多く掛かります。(PGI 6.0) (PGI 7.2 新設) -Mpfi[=indirect] -Mpfiオプションは、間接的(indirect)な関数呼び出しターゲットを保持することを指示する	最適化
pfo	強化されたブロック・リオーダーリング機能を含む特定の性能最適化を有効にするために、pgfi.outプロファイル・フィードバック・トレースファイルのデータを使用して最適化を行います。(PGI 6.0) (PGI 7.2 新設) -Mpfo[=indirect nolayout] Indirect サブオプションは、間接的な関数呼び出しのインライン化を有効にするもので、nolayout は、動的なコード配置を抑制する	最適化
pre[=all], nopre	(PGI 7.2新設)サブオプションを付けない -Mpre オプションは、一部の冗長部削除を有効にする。サブオプション all を付けた場合、よりアグレッシブな pre 処理を行う。 (PGI 9.0 以降) =all サブオプションが廃止された廃止された。 (PGI 2010以降) -Mnopre 冗長部削除の最適化を抑制する。	最適化
preprocess	cpp 形式の前処理をアセンブラ言語と Fortran ソースファイル上で行う。	その他
prof[=flags[,flags,.]]	プロファイルオプションをセット。関数レベルと、行レベルのプロファイリングがサポートされます。-Mprof=func、あるいは-Mprof=lines を指定する。これは、リンク時にも指定が必要である(特に Makefile 等でコンパイルとリンク処理を別々に行う際に注意) <b>PGI 6.0 New feature:</b> プロファイル・オプションをセットします。-ql, -qp, -pg スイッチは、通常、プロファイルのために使用されますが、プロファイリングのデフォルトの方法を再セット(上書き)するために以下のオプションを指定します。詳細は、PGI User's Guide をご覧ください。 <b>dwarf</b> : サードパーティのプロファイリング・ツールによって、ソース相関を有効にするため、DWARF 情報を生成する。 <b>func</b> : PGI スタイルの関数レベルのプロファイリングを実行する <b>hwcts</b> : ハードウェア・カウンタを用いた PAPI ベースのプロファイリングを使用する場合に指定する (linux x86-64ベースのシステムのみ) <b>lines</b> : PGI スタイルのソースレベルのプロファイリングを実行する <b>time</b> : サンプリングベースのインストラクション・ベースのプロファイリングを実行 (PGI 7.1 以降) プロファイルするアプリケーションにリンクするための mpich1 と mpich2 ライブラリ名を、-Mprof オプションに指定する。 mpich1 : プロファイル用 MPICH-1 ライブラリを使用。-Mmpi=mpich1 を包含します。 mpich2 : プロファイル用 MPICH-2 ライブラリを使用。-Mmpi=mpich2 を包含します。 mvapich1 : プロファイル用 MVAPICH-1 ライブラリを使用。-Mmpi=mvapich1 を包含します。 (PGI 8.0以降) [no]ccff : CCFF情報の有効化 [無効化] hpmmpi : プロファイル用 HP-MPI ライブラリを使用。-Mmpi=hpmmpi を包含します。	コード生成
[no]propcond	(PGI 7.1 新設) equality conditionalsから派生するassertions からのconstant propagation	最適化

	最適化を有効にします。これは、デフォルトで有効となります。	
[no]r8	(pgf77, pgfortranとpghpfのみ)コンパイラが REAL 変数と定数をDOUBLE PRECISION に変換する(しない)。	最適化
[no]r8intrinsic	(pgf77, pgfortranとpghpfのみ)コンパイラが 組み関数のCMPLX and REAL 型を DCMLX and DBLEとして扱う(扱わない)	最適化
[no]recursive	(pgf77, pgfortranとpghpfのみ)ローカル変数をスタックに割当てます(割当てません)。これは再帰を可能にします。SAVEされた、データ初期化された、または、namelist メンバは、このスイッチの設定に関係なく常にスタティックに割当てられます。	コード生成
[no]reentrant	コンパイラがコードをリエントラントとしない最適化を回避するかどうかを指定。	コード生成
[no]ref_externals	(pgf77, pgfortranとpghpfのみ) EXTERNAL 文に現れる名前の参照を強制(強制しない)。	コード生成
safepr[= <i>flags</i> ]	(pgcc と pgCC のみ)ポインタと配列の間のデータ依存関係を以下のサブフラグの内容でオーバーライドするようにコンパイラに指示します。以下のサブフラグ <i>flags</i> の詳細は、User's Guide を参照のこと。 all all is safe arg Argument pointers are safe auto Local pointers are safe dummy Argument pointers are safe local Local pointers are safe static Static local pointers are safe global Global pointers are safe -Msafepr All pointers are safe	最適化
safe_lastval	スカラがループの後で使用され、しかし、ループの全ての反復に関しては定義されない場合、コンパイラはデフォルトではループを並列化しません。しかし、このオプションは、コンパイラにループを並列化することが安全であると告げます。特定のループについて、全てのスカラの最後に計算された値がループの並列化を安全にします。	コード生成
[no]save	(pgf77, pgfortranとpghpfのみ)コンパイラが全てのローカルな変数がSAVE ステートメントと同等な状況に強いるように仮定するかどうかを決定。	Fortran言語
[no]scalarsse	スカラの浮動小数点演算において、xmm レジスタを使用した SSE/SSE2 のインストラクションを使用するか否かを指示。このオプションは、-tp { p7 / p7-64 / k8-32 / k8-64 以降の target } 時に有効。	最適化
schar	(pgccとpgCCのみ) "plain" character を signed char ととして扱う。--uchar を参照。	C / C++言語
[no]second_underscore	(pgf77, pgfortranとpghpfのみ) Fortran のグローバルなシンボル名が、既にその名前の suffix にアンダースコアを有しているものが存在している場合に、内部シンボル名として 2 つめのアンダースコアを加えます(加えません)。Fortran Module 間のシンボル名の競合が起きる場合にも便利です。また、g77プログラムとのリンク時に有効です。	コード生成
[no]signextend	コンパイラがサインビットを拡張するかどうかを指定します。	コード生成
[no]single	(pgccとpgCCのみ) float パラメータを double パラメータキャラクタに変換するかどうかを指示。	C / C++言語
nosizelimit sizelimit:n	ベクトライザにループ中のステートメント数に拘らず、全てのループに対してベクトル化最適化の対象とするように指示する。 PGI 6.2 から nosizelimit が、デフォルトとなった。一方、そのステートメントのサイズは、-Mvect=sizelimit:n (nはループ内のステートメントの数)によって制限される。	最適化
[no]smart	AMD64専用 post-pass instruction スケジューリングを行うか否かのスイッチ。(デフォルトは no)	最適化
	メインルーチン中に最適化された malloc ルーチンのコールを加えます。これを有効にするためには、Fortran、C、C++のメインプログラムを含むファイルをコンパイルする際に、このオプションを付する必要がある。デフォルトは、-Mnosmartalloc。(PGI 6.2 以降) <b>PGI 7.1 New feature:</b> -Msmartalloc オプションは、Linux 並びに Windows 上での large TLBs をサポートするために強化されました。このオプションは、最適な malloc ルーチンを有効にするために、メインプログラムをコンパイルする際に使用することが	

[no]smartalloc[=option]	<p>必要です。サブ・オプション huge は、シングルプログラムで使用される大きな 2MB ページを有効にするために指定します。これは、実行するために必要な TLB エントリ数を削減する効果があります。このオプションは、AMD の Barcelona やインテル(r)の Core2 システムで特に有効です。古いプロセッサ・アーキテクチャでは、TLB エントリの数が少ないため、大きな効果は期待できない可能性もあります。サポートするサブ・オプションは、以下のとおりです。</p> <p>huge : huge page のランタイムライブラリをリンクします。  huge:&lt;n&gt; : 使用されるページの数の限度を n に設定します。  hugebss : huge page の中に BSS セクションを置きます。</p> <p>huge サブ・オプションは、それ自身、必要とされる huge page をアロケートしようとし、Huge page の数は、:n サブ・オプションで制限を設けることができ、あるいは、環境変数 PGI_HUGE_SIZE でも設定できます。Hugebss は、プログラムの初期化されていないデータセクションを huge page の中に置きます。  (PGI 8.0 以降)  hugebss : huge ページの中に BSS セクションを置く  (PGI 9.0 新設)  nohuge : -Msmartalloc=huge を上書き(無効化)するサブオプション</p>	環境
standard	<p>(pgf77, pgfortran と pghpf のみ) ANSI 標準に適合しないソースコードを検出します。  (PGI 7.1 以降) -Mstandard は、-Mbackslash を内包しました。これは、-Mstandard が現れたときにバックスラッシュ・エスケープ・シーケンスを認識することを禁止します。例えば、バックスラッシュは標準的なキャラクタとして扱います。</p>	Fortran 言語
[no]stride0	<p>(pgf77, pgfortran と pghpf のみ) コンパイラは、増分がゼロであるかもしれない誘導変数を含むループのために代替のコードを生成します(生成しません)。</p>	コード生成
[no]traceback	<p>環境変数 \$PGI_TERM を使用することにより、ランタイム traceback のためにデバッグ情報が追加されました。また、デフォルトでのトレースバック機能は、f77、f90/f95 では有効となっておりますが、C/C++ では無効となっております。コンパイラへの初期設定ファイル siterc あるいは、.mypg*rc ファイルに TRACEBACK=OFF をセットすることで、デフォルトのトレースバック機能を無効にすることができます。反対に OFF の代わりに ON と指定することによって、有効にすることができます。</p>	最適化制御
uchar	<p>(pgcc と pgCC のみ) "plain character" を unsigned char として扱う。-- schar も参照。</p>	C/C++ 言語
unix	<p>(pgf77, pgfortran for Win32) Fortran サブプログラムに対して、UNIX の呼び出し、名前のコンベンションを使用することを指示。</p>	コード生成
[no]unixlogical	<p>(pgf77, pgfortran と pghpf のみ) 論理値 .TRUE. と .FALSE. が、unixlogical 非ゼロ(TRUE)、ゼロ(FALSE)と決定されるかどうかを決定します。デフォルトの unixlogical では、non-zero 値が TRUE で、0 の値が FALSE です。  nounixlogical は、VMS convention スタイルを使用する。</p>	Fortran 言語
[no]unroll[=flags]	<p>アンロール展開を制御。-Munroll=flags という形態でサブフラグを設定できる。以下のサブ・フラグ flags の詳細は、User's Guide を参照のこと。  c : <i>m</i>  n : <i>u</i></p> <p><b>PGI 7.1 New feature:</b>  -M[no]unroll[=c:&lt;n&gt; n:&lt;n&gt; m:&lt;n&gt;] : マルチ・ブロックを持ったループをアンロールする機能を追加しました。特に、条件文を伴ったこのようなループで、アンロールできるようになりました。新しいオプション -Munroll="m" は、この機能を制御するために導入されました。  n:&lt;n&gt; : シングル・ブロックを n 回アンロール  m:&lt;n&gt; : マルチ・ブロックを n 回アンロール  デフォルトでは、-Munroll=m は有効となっております。また、-Munroll=m の場合のデフォルトの n 値は 4 です。</p>	最適化
[no]upcase	<p>(pgf77, pgfortran と pghpf のみ) コンパイラがプログラム識別子に大文字を許すかどうかを決定します。upcase の場合、大文字も識別されます。デフォルトは、noupcase で全てが小文字として識別されます。特に、リンク時のモジュール名の識別において重要です。</p>	Fortran 言語

unsafe_par_align	並列化ループでの配列の参照において、その配列の最初の要素が「整列」されている限り、「整列移動 (aligned moves)」を行うことは安全であるとみなします。NOTE: このオプションは、コンパイラがその安全性を疑った場合でも、「整列移動」で行うコードを生成します。このオプションは、特に、 <u>STREAM Benchmark</u> やメモリ・インテンシブなメモリアクセスを含むループブロックの <u>並列化</u> で効果を発揮します。	最適化
vect	コードベクタライザを起動。プログラムのベクトル化を行います。-Mvect の指示だけでも良い。以下のサブ・フラグ <i>flags</i> の詳細は、User's Guide を参照のこと。 altcode: <i>n</i> / noaltcode : 代替スカラコードの生成 assoc / noassoc : ループの結合の許可 cachesize: <i>n</i> : cache tiling の最適化における cache size の仮定 nosizelimit : 全てのループに対して、そのソースコード数の制限なしで、ベクトル化の適用を行うように指示する prefetch : ベクトル可能なコードの可能な限りの prefetch 操作 smallvect[: <i>n</i> ] : 最大のベクトル長の定義 sse : SSE/SSE2 インストラクションの使用によるベクトル化 (PGI 7.1 以降) gether : 配列のgather (ギャザー) 間接参照を有するループのベクトル化ができるようになりました。コンパイラのデフォルトは、-Mvect=gether です。 (PGI 7.2 新設) partial : 最内側ループの分離によるループのベクトル化を有効にするように指示するサブオプション (PGI 8.0 以降) [no]short : 短いベクトル演算を有効化[無効化] -Mvect=short は、ループ外のスカラコードから生じる、あるいは、ループ・イテレーションの中から生じる短ベクトル演算のためのパックSSE演算の生成を有効化します。 (PGI 11.6 新設) simd:{128 256} : SIMD命令とデータを使用してベクトル化する際、そのデータ幅を 128bit / 256bit のどちらを使用するかを選択する。256bit を使用できるかはプロセッサに依存する。	最適化
novector	ベクトル化を抑制します。-fastsse のような機能複合オプションの後に、このオプションを指定した場合、他の機能に関しては影響せず、ベクトル化のみを抑制することができます。 (PGI 6.0)	最適化
novintr	コンパイラに、イディオム認識を実行しないように指示する、あるいは、手製の最適化ベクトル関数を導入することを指示する。	最適化
varargs	(pgf77 と pgfortran のみ) Fortran ユニットに対して、C ルーチンが vararg 型のインタフェースを有すると仮定する場合に指定します。	コード生成
writable-strings	(pgcc/pgCC: PGI 7.2新設) 書き込み可能なデータセグメント内に string constant をストアできるようにします。(注意) 既存の -Xt 並びに -Xs は、本オプションを含む	

### -C と C++ 特有のオプション

オプション	記述
-alias=[ansi traditional]	(PGI 7.1 以降) C、C++プログラムにおける、「型」ベースのポインタ・エイリアス規則に基づき、最適化方法を選択します。 ansi : ANSI C型ベースのポインタの一義化(disambiguation)を使用した最適化を有効化 traditional : 型ベースのポインタ一義化を無効にする C コンパイラでは、デフォルトは -alias=ansi で、C++ においては、-alias=traditionalとして います。
-A	(pgCCのみ) プログラムが Proposed ANSI C++ に合致していることを指示する
--no_alternative_tokens	(pgCCのみ) 代替トークンの認識を Enable/disable する。These are tokens that make it possible to write C++ without the use of the , , [ , ] , # , & , and ^ and characters. The alternative tokens include the operator keywords (e.g., and, bitand, etc.) and digraphs. デフォルトは、..no_alternative_tokens.
-B	C ソース内における // を使用したC++ 形式のコメントを許可する。
-b	(pgCCのみ) cfront2.1 互換でコンパイルを行う

-b3	(pgCCのみ) cfront3.0 互換でコンパイルを行う。See -babove.
-c89	(pgccのみ) C ソース言語として、C89 standard (C89) を使用する(PGI 6.2 以降) PGI 6.1 以前のデフォルト
-c8x	(pgccのみ) -c89 と同じ機能
-c99	(pgccのみ) C ソース言語として、C99 standard (C99) を使用する(PGI 6.2 以降 のデフォルト)
-c9x	(pgccのみ) -c99 と同じ機能
-[no]compress_names	(PGI 7.1 以降) C++ マングル名を 1024 キャラクタにフィットするように圧縮します。高度にネストされたテンプレート・パラメータは、非常時長い関数名が作成されるようになります。これらの長い名前は、古いアセンブラでは問題を引き起こす原因になります。現在のデフォルトは、-no_compress_names です。全ての C++ ユーザコードは、このスイッチを使用する際に、再度コンパイルされなければなりません。PGI によって提供されるライブラリは、-compress_namesと共に動作します。
--[no]bool	(pgCCのみ) bool の認識をするかどうかを指示する。デフォルトは、--bool.
--[no]builtin	数学関数ルーチンをビルトインでコンパイルするかどうかを指示する。選択された数学ライブラリルーチンをコンパイル時にインライン化する。デフォルトは、--builtin. コンパイルオプションは -M[no]builtin
--cfront_2.1	(pgCCのみ) cfront version 2.1互換でコンパイルするかどうかを指示する。
--cfront_3.0	(pgCC only) cfront version 3.0 互換でコンパイルするかどうかを指示する。
--create_pch filename	(pgCCのみ) filename を伴った プリコンパイルされたヘッダーファイルを生成する。
--dependencies	(pgCCのみ)makefile 依存性を標準出力に出力する (-M を参照)。
--dependencies_to_file filename	(pgCCのみ)makefile 依存性を filename ファイルに出力する。
--diag_error tag	(pgCCのみ) 指定されたダイアグ・メッセージの標準的なエラーレベルの内容を tag を使用して上書きする。
--diag_remark tag	(pgCCのみ) 指定されたダイアグ・メッセージの標準的なエラーレベルの内容を tag を使用して上書きする。
--diag_suppress tag	(pgCCのみ) 指定されたダイアグ・メッセージの標準的なエラーレベルの内容を tag を使用して上書きする。
--diag_warning tag	(pgCCのみ) 指定されたダイアグ・メッセージの標準的なエラーレベルの内容を tag を使用して上書きする。
--display_error_number	(pgCCのみ) 生成されたダイアグ・メッセージの中にエラーメッセージ番号を表示する。
--enumber	(pgCCのみ) C++ front-end error の数の上限を指定した数にセットする。
--[no_]exceptions	(pgCCのみ) Disable/enable例外処理のサポートを許可するかどうかを指示する。デフォルトは、--exceptions
--gnu_version	(pgCCのみ、PGI 2010以降) コンパイル時に使用するGNU C++ 互換性をセットする。デフォルトは、最新のバージョン番号がセットされる。使用例: --gnu_version 4.3.4。
--gnu_extensions	(pgCCのみ) Linux system header files をコンパイルする必要がある“include next” のような GNU 拡張を許す。
--instantiation_dir	(pgCCのみ) If --one_instantiation_per_object is used, define dirname as the instantiation directory.
--[no]lalign	(pgCCのみ) 整数の境界で、long long integersの整列を行うかどうかを指示する。デフォルトは --lalign.
-M	make 依存性リストを生成する。
-MD	make 依存性リストを生成する。
-MD,filename	(pgCCのみ) make 依存性リストを生成して、それらを filename へ出力する。
--microsoft_version	(pgCCのみ、PGI 2010以降) コンパイル時に使用するMicrosoft C++ 互換性をセットする。デフォルトは、最新のバージョン番号がセットされる。使用例: --microsoft_version 1.5。
--one_instantiation_per_object	(pgCCのみ) 各 template instantiation (function or static data member) を個々のオブジェクトファイル上に置く。
--optk_allow_dollar_in_id_chars	(pgCCのみ) 識別子としてドル記号を許す。

--pch	(pgCCのみ) 自動的にプリコンパイルされたヘッダファイルを使用する、あるいは生成することを指示する。
--pch_dir directoryname	(pgCCのみ) プリコンパイルされたヘッダファイルの置かれたディレクトリをサーチパスに加える。
--[no_]pch_messages	(pgCCのみ) 現在のコンパイルフェーズで、プリコンパイルされたヘッダファイルが生成され/使用されたかと言うメッセージを表示するかどうかを指示する。
--pedantic	(pgCCのみ) PGI 8.0 新設 含まれたシステムヘッダーファイルに関わる警告メッセージを印刷
+p	(pgCCのみ) 全ての anachronistic construct を抑制する。
-P	プリプロセスフェーズの後で止まり、プリプロセスされたファイルを filename.i にセーブ。
--preinclude=<filename>	(pgCCのみ) コンパイル時の始めにインクルードされるファイルの名前を指定する。このオプションは、システム依存のマクロ、型をセットする時に使われる。
--prelink_objects	(pgCCのみ) このオプションが指定された場合、テンプレート・ライブラリにしようとするオブジェクト・セットのために、template instantiations を作成する。
-t [arg]	テンプレート関数の instantiation を制御する。[arg] は以下の引数が存在する。 all local none used
--use_pch filename	(pgCCのみ) 現在のコンパイルフェーズで、指定された名前のプリコンパイルされたヘッダファイルを使用する。
--[no_]using_std	(pgCCのみ) 標準ヘッダーファイルがインクルードされた時に、std namespace の使用を暗黙に使用するか否かを指示する。
-X	(pgCCのみ) クロス・リファレンス情報を生成し、指定されたファイルに書き込む。
-Xm	(pgCCのみ) 名前として \$ を許す。 PGI 7.1 以降、このオプションは削除されました。現在、ほとんどの場合、ドルサインは許可されています。
-Xs	(PGI 7.1以降) C/C++ において、レガシーな標準モードを使用する。 これは、-alias=traditional オプションを内包します。
-Xt	(PGI 7.1以降) C/C++ において、レガシーな移行モードを使用する。 これは、-alias=traditional オプションを内包します。
-xh	(pgCCのみ) 例外処理を enable にする。
--zc_eh	(PGI 7.1以降) ゼロ・オーバーヘッド例外領域を生成します。本オプションは、実際の例外処理が起こるまで、例外ハンドリングのコストを遅らせる措置を行います。多くの例外領域を有しながら、あまり例外が起こらないプログラムでは、このためのコンパイル・オプションによって、ランタイム性能の向上に繋がるかもしれませんが、デフォルトは、--zc_eh を使用しませんが、その代わりに、setjmp と longjmp と共に例外ハンドリングを実装する -sjlj_eh を使用します。このオプションは、PGI C++ の以前のバージョンでコンパイルされた C++ コードにも互換性があります。--zx_eh オプションは、libgcc_eh 内のシステム unwind ライブラリを提供している新しい Linux システムと Windows 上でのみ有効です。 このオプションは、PGI 2011(11.0)以降、C++コンパイラのデフォルトとなりました。
-suffix (see -P)	(pgCCのみ) -E、-F、-P の機能で指定された中間ファイルをセーブする。

### PGI 6.0 以降でのC++のテンプレートのインスタント化の変更について

C++ テンプレートのインスタント化は、32-bit 並びに 64-bit Linux システムにおいて変更されました。新しい方法では、全てのテンプレート参照を解決するためにGNUリンカーを使用し、重複を回避することでテンプレートの使用の単純化に大きな効果を発揮します。この新しい方法は、PGI コンパイラの前のバージョンと互換性はありません。C++ プログラムを PGC++ 6.0 用に移行するためには、全ての C++ プログラムを再コンパイルすることが必要です、また、makefile 上の全てのテンプレート・インスタント化フラグを削除することが必要です。次のテンプレートのインスタント化に関係するコマンドオプションが削除する対象となります。

```
-one_instantiation_per_object
-instantiation_dir
-instantiate
-[no]auto_instantiation
```

```
-prelink_objects  
-Wc, -tlocal  
-Wc, -tused  
-Wc, -tall
```