

HP C++バージョン A.12.14 リリース・ノート

HP 9000 コンピュータ

第 1 版



HP Part No. 5969-0302

1999 年 2 月

Printed in USA

ご注意

- (1) 本書に記載した内容は、予告なしに変更することがあります。
- (2) 本書は内容について細心の注意をもって作成いたしました。万が一不審な点や誤り、記載もれなど、お気づきの点がございましたら当社までお知らせください。
- (3) お客様の誤った操作に起因する損害については、当社は責任を負いかねますのでご了承ください。
- (4) 当社では、本書に関して特殊目的に対する適合性、市場性などについては、一切の保証をいたしかねます。また、備品、パフォーマンス等に関連した損傷についても保証いたしかねます。
- (5) 当社提供外のソフトウェアの使用や信頼性についての責任は負いかねます。
- (6) 本書の内容の一部または全部を、無断でコピーしたり、他のプログラム言語に翻訳することは法律で禁止されています。
- (7) 本製品パッケージとして提供した本マニュアル、フレキシブル・ディスクまたはテープ・カートリッジ等の媒体は本製品用だけにお使いください。プログラムをコピーする場合はバックアップ用だけにしてください。プログラムをそのままの形で、あるいは変更を加えて第三者に販売することは固く禁じられています。

© copyright 1999 Hewlett-Packard Company.

© copyright 1999 日本ヒューレット・パカード株式会社

本書には著作権によって保護されている内容が含まれています。本書の内容の一部または全部を著作者の許諾なしに複製、改変、および翻訳することは、著作権法下での許可事項を除き、禁止されています。

All rights reserved.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY

3000 Hanover Street

Palo Alto, California 94304 U.S.A.

日本ヒューレット・パカード株式会社

〒168-0072 東京都杉並区高井戸東3丁目29番21号

TEL 03-3331-6111(大代表)

© copyright 1999 Hewlett-Packard Company.

UNIX は、X/Open カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です。

出版履歴

出版の日付は、最新版ができるたびに更新します。内容の小さな変更に対しては、増刷の際に対応し、出版日の更新は行いません。マニュアルのパーツ番号は、改訂が行われるたびに更新します。

新版の作成は、記載内容の訂正もしくはドキュメント製品の変更に伴って行われます。お手元のマニュアルが最新のものか否かは、当社の営業担当に確認してください。

- 1999年2月... 第1版

原典

本書は『HP C++ Version A.12.14 Release Notes』(HP Part No. 5967-0038)を翻訳したものです。

納入後の保証について

- ・保証の期間は、ご購入時に当社よりお出しした見積書に記載された期間とします。保証サービスは、当社の定める休日を除く月曜日から金曜日までの、午前8時45分から午後5時30分の範囲で無料で行います。当社で定めたシステム製品については出張修理を行い、その他の製品については当社にご返却いただいた上での引取り修理となります。当社が定める地域以外における出張修理対象製品の修理は、保証期間中においても技術者派遣費が有料となります。
 - ・ソフトウェア製品の保証は上記にかかわらず、下記に定める範囲とさせていただきます。
 - ・ソフトウェア製品およびマニュアルは当社が供給した媒体物の破損、資料の落丁およびプログラム・インストラクションが実行できない場合のみ保証いたします。
 - ・バグおよび前記以外の問題の解決は、別に締結するソフトウェア・サポート契約に基づいて実施されます。
 - ・次のような場合には、保証期間内でも修理が有料となります。
 - ・取扱説明書等に記載されている保証対象外部品の故障の場合。
 - ・当社が供給していないソフトウェア、ハードウェア、または補用品の使用による故障の場合。
 - ・お客様の不適合または不十分な保守による故障の場合。
 - ・当社が認めていない改造、酷使、誤使用または誤操作による故障の場合。
 - ・納入後の移設が不適切であったための故障または損傷の場合。
 - ・指定外の電源（電圧、周波数）使用または電源の異常による故障の場合。
 - ・当社が定めた設置場所基準に適合しない場所での使用、および設置場所の不十分な保守による故障の場合。
 - ・火災、地震、風水害、落雷、騒動、暴動、戦争行為、放射能汚染、およびその他天災地変等の不可抗力的事故による故障の場合。
 - ・当社で取扱う製品は、ご需要先の特定目的に関する整合性の保証はいたしかねます。また、そこから生じる直接的、間接的損害に対しても責任を負いかねます。
 - ・当社で取扱う製品を組み込みあるいは転売される場合は、最終需要先における直接的、間接的損害に対しては責任を負いかねます。
 - ・製品の保守、修理用部品の供給期間は、その製品の製造中止後5年間とさせていただきます。
- 本製品の修理については取扱説明書に記載されている最寄の事業所へお問い合わせください。

まえがき

本書では次の事項について説明します。

- ・新機能と変更された機能
- ・インストール情報
- ・関連マニュアル
- ・問題点とバグ修正

注記

リリース・ノートのタイトルに印刷されているソフトウェア・コードは、リリース時のソフトウェア製品バージョンを表します。製品やオペレーティング・システムのマイナーな変更点はマニュアルに反映されていない場合があります。リリース・ノートは製品やオペレーティング・システムの変更点に厳密に 1 対 1 の対応をしているわけではありませんので、ご注意ください。

本書は、オンライン版も提供されています

(`/opt/CC/newconfig/RelNotes/CXX.release.notes`)。このファイルは、次のように `lp` コマンドを使用して出力できます。

```
lp -dprinter_name /opt/CC/newconfig/RelNotes/CXX.release.notes
```

目次

第 1 章 新しい機能と変更された機能 . . . 9
HP C++バージョン A.12.14 の概要 . . . 10
新しい機能と変更された機能 . . . 10
第 2 章 インストール情報 . . . 13
UNIX システム V リリース 4 (V.4) ファイルシステムへの移行 . . . 14
第 3 章 関連マニュアル . . . 15
オンライン・マニュアル . . . 16
オンライン・サンプル・ソース・ファイル . . . 18
印刷文書 . . . 19
その他の印刷文書 . . . 20
例外処理機能について . . . 21
例外処理機能使用時にリンク非互換性を検出する . . . 21
共有ライブラリ内のリンク非互換性を検出する . . . 21
例外処理言語の説明 . . . 22
例外のスロー . . . 22
例外処理 . . . 24
例外指定 . . . 24
terminate() と unexpected() . . . 25
その他の問題 . . . 26
第 4 章 問題点の詳細、バグ修正、既知の制限事項 . . . 27
既知の問題点 . . . 28
既知の制限事項 . . . 29
ライブラリ・ルーチンを使用する前に浮動小数点例外をトリガする . . . 29
setjmp/longjmp と +eh オプション . . . 29
サポートされていないカーネル・スレッド . . . 29

表一覧

表 4 - 1 スレッド・セーフ・レベル・・・30

第 1 章 新しい機能と変更された機能

本章では、HP C++バージョン A.12.14 の新機能と変更された機能の概要を説明します。

HP C++バージョン A.12.14 の概要

HP C++バージョン A.12.14 の新機能と変更された機能には、コンパイラの拡張機能、問題点の修正、オンライン・ヘルプの改訂版が含まれます。

新しい機能と変更された機能

HP C++バージョン A.12.14 の新機能と変更された機能を次に示します。これらの項目は、『HP C++ Online Programmer's Guide』に記載されています (アクセス方法については第 3 章を参照)。

- 浮動小数点演算を使用するプログラムは、ライブラリ・ルーチンを指定する前に、浮動小数点例外をトリガする必要があります。「ライブラリ・ルーチンを使用する前に浮動小数点例外をトリガする」の項を参照してください。
- `-ptb` オプションは、テンプレートを含む他の共有ライブラリに依存している共有ライブラリのビルドには必要ありません。コンパイラのデフォルトの動作では、このオプションを不要なものと思えますが、後方互換のために保持しています。
- HP C++は、 2^{29} バイトを上回る大容量データ・オブジェクトをサポートしていません。このようなデータを必要とするオブジェクトを利用する場合は、HP aC++コンパイラを使用してください。
- HP C++は PA-RISC 64 ビット・モードをサポートしていません。64 ビット・モードの機能を使用するには、HP aC++コンパイラを使用してください。
- HP C++ (cfront) には `libcodelibs.a` が含まれていませんが、HP Codelibs Library のソース・コードは含まれています。このライブラリを使用する場合は、ソース・コードからビルドする必要があります。ライブラリをビルドする方法については、`/usr/contrib/codelibs/README` を参照してください。
- HP-UX 上の次のような変更によって、HP C++の旧バージョンで作成されたプログラムに非互換性が生じています。
 - `typedef size_t` に対応する基本の型は、`unsigned int` から `unsigned long` に変更されました。同様に、`ptrdiff_t` は、`int` から `long` に変更されました。これらの変更によって、`size_t` が非外部の C インタフェースで使用される場合に、マングリングされた符号が異なるため、互換性の問題が生じます。これらの変更によって、任意のオブジェクト・ファイルを再コンパイルするかリンクすると、すべての C++ ファイルを再コンパイルしなければなりません。つまり、アーカイブ形式の他社のライブラリも、再コンパイルするか、更新する必要があります。
 - HP-UX では、`time_t` が `long` 型に変更されました。これによって、HP-UX 10.10 または 10.20 用の HP C++を使用するとエラーを出さずにコンパイルされるソース・ファイルが、リリース 11.0 ではコンパイルできなくなります。次に示す例は、この場合に生じる結果の例です。

```
1: #include <time.h>
```

```
2: time_t ff (time_t t) { return t; }
3: time_t ff (long t) { return t; }
4: int main () { long tt = ff (1L); return 0; }
```

この例では、time_t パラメータまたは long パラメータを取るため、ff がオーバーロードします。10.10 システムまたは 10.20 システムでは、time_t が int であるため、コードがコンパイルされます。11.0 システムでは、time_t が long であるため、コードはコンパイルに失敗します。

```
CC: "tm.c", line 4: error: two definitions of ff() (1034)
```

第 2 章 インストール情報

インストールを開始する前に、本書とその他の関連リリース・ノートや `readme` ファイルをすべてお読みください。

ソフトウェアをインストールするには、SD-UX の `swinstall` コマンドを実行します。これにより、ユーザー・インタフェースが起動するので、画面の指示に従ってインストールを行います。インストール手順や関連事項についての詳細は、HP-UX 10.x オペレーティング・システム・パッケージに付属または記載されている『SD-UX による HP-UX ソフトウェア管理』、他の `readme` ファイル、インストールやアップグレードに関するマニュアルを参照してください。

HP C++には約 46MB のディスク・スペースが必要です。/opt/CC のファイル用に 16MB、DDE, Blink Link, HP/PAK 用に 30MB を使用します。使用する環境によっては、システムの他の要素(ネットワーク、システム・セキュリティ、ウィンドウ・システムなど)に関するマニュアルも必要になる場合があります。

UNIX システム V リリース 4 (V.4) ファイルシステム への移行

本リリースでは、2種類の移行ツールが提供されています。システムの Upgrade Tools または `tlink_install` スクリプトのいずれを使用しても、HP-UX 9.x から HP-UX 10.x へ移行できます。

システムに Upgrade Tools がインストールされている場合 (`/usr/sbin/upgrade` がある場合) は、製品のインストール時に自動的に移行リンクが生成されます。これらのリンクの削除方法も提供されています。自動移行リンクについての詳細は、オペレーティング・システムのアップグレードに関するマニュアルを参照してください。

システムに Upgrade Tools がインストールされていない場合は、C++の `tlink_install` スクリプトを移行ツールとして使用して、HP-UX 9.x から HP-UX 10.x へ移行する際に HP C++製品の実行可能ファイルとインクルード・ファイルのシンボリック・リンクを作成します。このスクリプトは `/opt/CC/newconfig/tlink_install` にあります。リンクを削除するには、`/opt/CC/newconfig/unlink_install` にあるスクリプトを使用してください。これらのスクリプトは必ず、スーパー・ユーザーが実行してください。

移行プロセスを逆順に実行する場合は、インストールに使用したツールに対応するアンインストール・ツールを使用する必要があります。つまり、システムの Upgrade Tools を使用してリンクをインストールした場合は必ず、Upgrade Tools を使用してリンクをアンインストールしなければなりません。同様に、`tlink_install` スクリプトを使用してリンクをインストールした場合は必ず、`unlink_install` スクリプトを使用してリンクをアンインストールしなければなりません。

第 3 章 関連マニュアル

以降の項では、HP C++の関連マニュアルについて説明します。

オンライン・マニュアル

X ウィンドウ・ユーザーは、次のいずれかの方法で『HP C++ Online Programmer's Guide』を起動できます。

- `cc` コマンド行で `+help` オプションを使用します。
- HP CDE フロント・パネルの “?” アイコンをクリックした後、HP C++ アイコンをクリックします。
- 次に示すように、`/usr/dt/bin` で `dthelpview` コマンドを実行します。

```
/usr/dt/bin/dthelpview -h /opt/CC/dt/appconfig/help/C/CXX.sdl
```

HP コンパイラが別のシステムにインストールされているか、またはシステムが HP CDE を実行していない場合に、このコマンドを使用すると便利です。

HP C++には次のようなオンライン・マニュアルが含まれています。

- 『HP DDE デバッガ オンライン・ヘルプ』
『HP DDE デバッガ オンライン・ヘルプ』内の、最適化コードの (高度なデバッグではなく) 基本デバッグに関する説明を参照してください。
- 『HP C++ Online Programmer's Guide』を起動するには、次のコマンドを使用します。

```
CC +help
```

『HP-UX Linker and Libraries Online User Guide』を起動するには、次のコマンドを使用します。

```
ld +help
```

オンライン・マニュアル『HP-UX Linker and Libraries Online User Guide』は、『Programming on HP-UX』の代わりに使用できます。

注記

文字ベースの端末または端末エミュレータを使用するユーザーは、`charhelp` プログラムを使用して、C++ やリンカに付属するオンライン・ヘルプを表示したり印刷できます。

`charhelp` を起動するには、絶対パス (または、環境変数 `$PATH` で `/opt/langtools/bin` が設定されている場合は `charhelp` のみ) を入力します。画面に `charhelp` の使用方法が表示されます。

```
$ /opt/langtools/bin/charhelp
charhelp: Usage: charhelp {cc CC f77 ld -helpVolume file}
```

例えば、C++に関するヘルプを表示するには、`charhelp CC` と入力し、以降は画面の指示に従って操作します。詳細は、`charhelp(1)` のマニュアル・ページを参照してください (環境変数 `$MANPATH` で `/opt/langtools/share/man/man1.z` を必ず設定しておいてください)。

`+help` オプションが正しく機能しない場合は、環境変数 `DTHELPPSEARCHPATH` が設定されているか確かめてください (`rlogin` を使用してシステムにリモート・ログインしている場合は、

この環境変数が設定されていない可能性があります。設定されていない場合は、次のコマンドを使用して設定してください。

```
eval $(dtsearchpath)
```

必ず環境変数 LANG を設定してください (通常は LANG=C)。

設定の際に、HP CDE フロント・パネルの?アイコンをクリックするか、次のコマンドのいずれかを入力すると、リンクのオンライン・ヘルプを表示できます。

```
/usr/dt/bin/dthelpview -helpVolume linker
```

または

```
/usr/dt/bin/dthelpview -h
```

```
/opt/langtools/lib/linker/dt/appconfig/help/C/linker.sdl
```

- 『HP C++ Templates Technical Addendum』では、HP C++に実装されているテンプレートについて説明しています。このマニュアルへは、『HP C++ Online Programmer's Guide』内でアクセスできます。また、ポストスクリプト形式のファイル `/opt/CC/newconfig/TecDocs/templates.ps` や ASCII ファイル `/opt/CC/newconfig/TecDocs/templates.ascii` としても提供されています。

- 『HP C++ Troubleshooting Notes』では、発生する可能性のある問題の診断方法や解決方法を重点的に説明しています。このマニュアルに含まれる「トラブルシューティング・マトリクス」とツールのリストへは、ディレクトリ `/opt/CC/contrib/Tools` からオンラインでアクセスできます。

このマニュアルは、ポストスクリプト形式のファイル

`/opt/CC/newconfig/TecDocs/templates.ps` や ASCII ファイル

`/opt/CC/newconfig/TecDocs/templates.ascii` でも入手できます。ASCII ファイルへは『HP C++ Online Programmer's Guide』からアクセスできます。

- 『HP C++リリース・ノート』(本書)のオンライン版は、ASCII ファイル形式の `/opt/CC/newconfig/RelNotes/CXX.release.notes` で入手できます。
- 『HP PA-RISC Compiler Optimization Technology White Paper』では、最適化を使用する利点を説明しています。ポストスクリプト形式のファイル `/opt/langtools/newconfig/white_papers/optimize.ps` で入手できます。
- CC, C++filt, nm++, gprof++と標準ライブラリ (stream, task, complex, codelibs, および標準コンポーネント) のマニュアル・ページは `/opt/CC/share/man` にあります。

関連マニュアル
オンライン・サンプル・ソース・ファイル

オンライン・サンプル・ソース・ファイル

HP C++には、『HP C++ Programmer's Guide』に記載されているサンプル・ソース・ファイルが
付属しています。このファイルは/opt/CC/contrib/Examples ディレクトリにあります。

印刷文書

- ・ 『HP C++リリース・ノート』(本書)。英文のリリース・ノートは、前述のとおりオンラインでも提供されています。
 - ・ 『HP C++ Programmer's Guide』(92501-90005)に記載されている情報は、『HP C++ Online Programmer's Guide』とほぼ同じですが、多少古い場合があります。このマニュアルは、Hewlett-Packard's Support Materials Organization (SMO)でお求めになれます。ご注文の際は、最寄りの当社営業所まで、発注番号と併せてご連絡ください。
 - ・ 『C++クイック・リファレンス・カード』(B1637-97007)
 - ・ 『HP/DDE デバッガ ユーザーズ ガイド』(B5462-90001)では、HP 9000 の HP Distributed Debugging Environment (分散デバッグ環境)でのプログラムのデバッグ方法を説明しています。
- HP 社の印刷文書をご注文の際は、*manuals(5)*を参照してください。

その他の印刷文書

C++関連の入手可能なマニュアルの一部を次に示します。

- 『Codelibs Library Reference』(B2617-90600) では、HP Codelibs クラス・ライブラリに関する情報をすべて記載しています。この文書は、Hewlett-Packard's Support Materials Organization (SMO) でお求めになれます。ご注文の際は、最寄りの当社営業所まで、発注番号と併せてご連絡ください。また、*codelibs(3X)*のマニュアル・ページも参照してください(Man page could not be formattedまたはNo manual entry for codelibsというメッセージが表示される場合は、マニュアル・ページがインストールされていて、環境変数 `MANPATH` で `/opt/CC/share/man` が設定されているかどうか確認してください)。
- 『C++ Programming Language』(second edition、Bjarne Stroustrup (ISBN 0-201-53992-6)) は C++ の入門書で、C++ 言語の完全なリファレンスと、オブジェクト指向の設計やソフトウェア開発に関する情報が記載されています。この文書はコンピュータ専門書の取り扱い店でお求めになれます。
- 『C++ Primer』(second edition, Stanley Lippman (ISBN 0-201-54848-8)) は、C++の入門書で、コンピュータ専門書の取り扱い店でお求めになれます。
- 『注解 C++リファレンス・マニュアル』(Margaret Ellis、Bjarne Stroustrup (ISBN 4-8101-8027-1)) は、C++ 言語の完全なリファレンスに加えて、ある機能がなぜそのように定義されているのかなどに関する詳しい注解と解説部が記載されています。コンピュータ専門書の取り扱い店でお求めになれます。
- USL/Novell 社のマニュアルには、C++に関する有用な情報が記載されています。これらのマニュアルの中には、HP C++がベースとしている cfront コンパイラだけを扱っているものもあります。

これらのマニュアルの最新版に関するお問い合わせ先は、次の通りです。

- 米国 — (電話) 1-800-336-5989
- 米国以外 — (FAX) 1-801-431-4045

Basic SDK Documentation Set も必要となります。このドキュメンテーション・セットの内容は次の通りです。

- 『Software Development Tools』
- 『Programming in Standard C and C++』
- 『Debugging and Analyzing C and C++ Programs』
- 『Programming with System Calls and Libraries』
- 『Porting and Integration Guide』
- 『C++ Standard Components Programmer's Guide』
- 『Programming with the C++ Standard Libraries』

例外処理機能について

旧バージョンのリリース・ノートに記載されている例外処理機能に関する有用な情報を次に示します。

例外処理機能は、コンパイラ・モードと翻訳モードの両方で使用できるので、モードの異なるオブジェクト・ファイルを混在させることができます。コンパイルとリンクの両方で例外処理機能を使用可能にするには、`+eh` オプションを使用します。翻訳モードで`+eh` オプションを使用した場合は、性能が一部低下します。

例外処理機能使用時にリンク非互換性を検出する

HP C++の本リリースでは、`+eh` オプションを指定した場合の例外処理機能をサポートしていません。`+eh` オプションを使用してコンパイルされたコードは、このオプションを使用せずにコンパイルされたコードとリンク互換性がないことに注意してください。これには次のような3つの理由があります。

1. `+eh` オプションを使用可能にすると、コンストラクタはヒープ・オブジェクトにメモリを割り当てなくなります。これは、ヒープ・オブジェクトに対するメモリの割り当てが、コンストラクタが呼び出される前に行われるようになったためです。例えば、`+eh` オプション不使用のコードが`+eh` オプション使用コンストラクタを呼び出してヒープ・オブジェクトを構築した場合、ヒープ・オブジェクトにはメモリが割り当てられません。
2. `+eh` オプションを使用可能にすると、すべてのコンストラクタは、オブジェクトの構築がどれくらい進行しているかをある程度記録します。この作業が必要なのは、例外発生時に、一部だけ構築が完了したオブジェクトをクリーンアップする必要があるためです。`+eh` オプション使用コードが、`+eh` オプション不使用のコンストラクタを呼び出した場合は、記録は行われません。従って、例外が発生した場合、`+eh` オプション不使用のコンストラクタを呼び出したプロシージャに含まれるオブジェクトの状態について誤った情報が記録されます。
3. `+eh` オプション使用プロシージャはすべて、各プロシージャ内で構築されるオブジェクトのリストに関する情報を記録します。`+eh` オプション不使用のプロシージャはこの記録を行わないため、例外が発生しても、このプロシージャにはオブジェクトのクリーンアップが適用されません。

共有ライブラリ内のリンク非互換性を検出する

CC ドライバを使用して共有ライブラリを生成した場合 (`-b` を使用)、`c++ patch` は前述と同じ規則に従ってリンクの非互換性を検出します。共有ライブラリを持つリンクを実行すると、HP C++はランタイムまで待機して、リンクまたは明示的にロードされる各共有ライブラリが必ずメインの実行可能ファイルと互換性を持つようにします。非互換性が検出された場合は、デフォルトでは警告メッセージが `stderr` (標準エラー) に出力されます。このデフォルト動作を無効にするには、ユーザー独自のルーチン `__link_incompatibility` にリンクします。

例えば、このような種類の警告を一切表示しない場合は、次に示すルーチンにリンクします。

関連マニュアル

例外処理機能について

```
extern "C" void __link_incompatibility
    (const char* libname, int lib_mode) {
    //libname is the name of the library
    //lib_mode == 0 for a non +eh library
    //lib_mode == 1 for a +eh library

    //You can provide your own version to override the
    //default behavior
    //This is an empty body which does nothing
}
```

例外処理言語の説明

本項では、さまざまな例外処理言語の問題について説明します。これらは、『注解 C++ リファレンス・マニュアル』の詳しい説明と考えてください。ここでは、HP 社が実装した例外処理機能の動作を説明します。

本項の内容は次の通りです。

- 例外のスロー
- 例外処理
- 例外指定
- terminate() と unexpected()
- その他の問題

例外のスロー

1. 曖昧な基底クラスを持つクラスはスローできますか。つまり、次のようなスローは正しいですか。

```
struct A { ... };
struct B1 : A { ... };
struct B2 : A { ... };
struct C : B1, B2 { ... };
void f()
{
    C c;
    throw c; // legal?
}
```

正しくありません。曖昧な基底クラスを持つクラスはスローできません。

2. 同じ基底クラスの複数インスタンスを持つクラスは、基底クラスのいずれか 1 つのインスタンスがアクセス可能であればスローできますか。

できません。同じ基底クラスの複数インスタンスを持つクラスは、基底クラスのいずれか 1

つのインスタンスだけがアクセス可能であってもスローできません。

3. リファレンスをスローするとどうなりますか。

一時領域が割り当てられて、スロー引き数が参照するオブジェクトがここにコピーされ、適切なハンドラの検索が開始されます。

ハンドラが見つかると、引き数がリファレンス型でない場合は、ローカル変数は一時領域から初期化されます。ハンドラのローカル変数がリファレンス型の場合は、一時領域へのリファレンスが作成されます。

ここで注目すべき点は、グローバル変数へのリファレンスがスローされた場合で、ハンドラのローカルがリファレンス型であれば、ハンドラは、グローバル変数へのリファレンスではなく、一時ディレクトリへのリファレンスを獲得することです。

4. 多重定義関数名はスローできますか。

できません。多重定義関数の名前 (実際にはそのアドレス) はスローできません

5. スローの優先順位はどのようになっていますか。

throw 式は代入式です。

6. スローは条件式に入れることは可能ですか。例えば、次のような式は正しいですか。

```
void f()  
{  
  int x;  
  x ? throw : 12;  
}
```

```
void g()  
{  
  int x;  
  x ? 12 : throw;  
}
```

可能です。スローは条件式に入れても構いません。

7. ネストしたスローは可能ですか。

可能です。ネストしたスローが発生すると、前回の例外処理は破棄され、新規の例外処理が行われます。

8. ハンドラの動的コンテキスト外で再スローが発生した場合、どうなりますか。

ハンドラの動的コンテキスト外での再スロー動作は定義されていません。

9. シグナル・ハンドラ内で例外をスローするとどうなりますか。

シグナル・ハンドラ内での例外のスローはサポートされていません。シグナル・ハンドラの実行時期を予測することはできないため、例外処理機構が矛盾した状態になった場合にシグナル・ハンドラが呼び出される可能性があります。

10. シグナル・ハンドラ内で `longjmp` が発行された場合どうなりますか。

これは、シグナル・ハンドラ内での例外スローがサポートされないのと同じ理由により、推奨できません。シグナル・ハンドラはコードの処理を中断するため、データ構造体が定義されず予期しない結果が生じることになります。

例外処理

1. マスクされた `catch` 文節に対する警告メッセージまたはハード・エラーは、実装側で生成する必要がありますか。

マスクされた `catch` 文節を指定した場合はエラーとなります。

2. リンク指定を行った場合、関数 (のアドレス) を受け取るハンドラに影響を与えますか。

与えません。この型の関数はリンク指定による影響は受けません。

次に例を示します。

```
extern "C" {  
void f(int);  
};
```

```
void g()  
{  
throw f;  
}
```

is catchable by:

```
catch (void (*)())
```

3. `catch` 文節内に不完全な型エラーが現れる可能性はありますか。

ありません。 `catch` 文節内に不完全な型が現れることはできません。

4. 例外処理はいつ行われますか。

例外処理は、次のいずれかが発生した時点で行われます。

- 例外用ハンドラが起動された
- `terminate` が起動された
- `unexpected` が起動された

例外指定

1. 関数の定義と宣言ではすべての例外指定が一致する必要がありますか。

はい。関数の定義と宣言ではすべての例外指定が一致する必要があります。

2. 曖昧な基底クラスを持つクラスは指定リストに記載できますか。つまり、次に示す `bar` に対する例外指定は正しいですか。

```
struct A { ... };  
struct B1 : A { ... };  
struct B2 : A { ... };  
struct C : B1, B2 { ... };
```

```
void foo (C* cp)
{
w *cp;    //error according to ANSI
}
```

```
void bar () throw(C); // legal?
```

正しくありません。曖昧な基底クラスを持つクラスは例外指定には指定できません。

3. 例外指定リストにあるクラスの派生クラスは、同じ例外指定リストにも指定できますか。
はい。例外指定リストにあるクラスの派生クラスは同じ例外指定リストにも指定できます。
4. 例外指定リスト内の基底クラスへのポインタをリストする関数は、このクラスの派生クラスへのポインタもスローできますか。
はい。例外指定リスト内の基底クラスへのポインタをリストする関数は、このクラスの派生クラスへのポインタもスローできます。
5. リファレンスは例外指定リスト内で指定できますか。
はい。リファレンスは例外指定リスト内で指定できます。
6. 例外指定リストに同じ型を 2 回以上指定できますか。
つまり、次のような宣言は正しいですか。

```
void baz() throw(A,A,A);    // legal?
```

はい。例外指定型リストでは型が重複しても構いません。

7. 例外指定リストには不完全な型を指定できますか。例えば、次のような文は正しいですか。

```
struct A;
void f() throw(A) { }
```

はい。例外指定リストには不完全な型を指定できます。

8. 例外指定はどこに指定できますか。

例外指定は、宣言または定義されている関数に対してのみ、関数宣言または関数定義内で指定できます。例外指定は引き数リストや typedef 内には指定できません。

terminate() と unexpected()

1. スローされた例外が処理されない場合はどうすればよいですか。

クリーンアップは行わず、`terminate` を呼び出します。

静的オブジェクトの構築中に例外の未処理が発生した場合は、`terminate` を呼び出します。`terminate` が `exit` を呼び出した場合は、完全または部分的に構築済みになっている静的オブジェクトを破壊する必要があります。

静的オブジェクトの破壊中に例外の未処理が発生した場合は、`terminate` を呼び出します。`terminate` が `exit` を呼び出した場合は、破壊が完了していない静的オブジェクトを破壊する必要があります。例外の原因になるため、オブジェクトは再度破壊しないようにしてください。

2. `terminate()` は `exit()` を呼び出すことができますか。

はい、できます。

3. `unexpected()`は制御を戻すことができますか。

いいえ、できません。

4. `unexpected()`はスローまたは再スローを行えますか。

はい、できます。

5. `unexpected()`は何を再スローしますか。

`unexpected()`は、`unexpected()`が呼び出される原因となった例外を再スローします。

その他の問題

1. `try` ブロックとハンドラへの制御の移行は可能ですか。

いいえ。`try` ブロックとハンドラへの制御の移行はできません。

2. デストラクタは最初の文に到達する直前に完了したとみなされますが、オブジェクトは、最後の文に到達した時点で構築されたとみなしてよいですか。

オブジェクトは、コンストラクタ内のすべての動作が完了するまで完全に構築されたとはみなされません。またオブジェクトは、デストラクタ内で何らかの動作が発生する前に部分的に破壊されているとみなされます。

3. EH ランタイムは、新旧メモリの交換によって割り当てられるメモリを削除する必要がありますか。

いいえ、ありません。

4. 処理不可能な例外がスローされた場合、ローカル変数とグローバル変数をクリーンアップするべきですか。

いいえ、クリーンアップしてはいけません。

5. デストラクタが呼び出されたオブジェクトは、EH ランタイムによってクリーンアップする必要がありますか。

デストラクタが暗黙的に呼び出されたオブジェクト上で、デストラクタを明示的に呼び出してはいけません。従って、EH ランタイムは、オブジェクトに対してデストラクタの明示的呼び出しが発行されたかどうかを考慮する必要はありません。

6. `exit()`は標準例外をスローして、確実に `auto` 変数をクリーンアップするべきですか。

いいえ、`exit()`は例外をスローしてはいけません。

7. `atexit()`を使用して登録されている関数から例外がスローされた場合、どうなりますか。

`terminate()`が呼び出されます。

8. ユーザー・プログラムが `alloca()`を呼び出した場合、どうなりますか。

`alloca()`を使用できるのは翻訳モードだけです。ただし、この関数を使用しないことをお勧めします。

第 4 章 問題点の詳細、バグ修正、既知の制限事項

本章では、特に指定のない限り、HP C++の現行バージョンについての既知の問題点と制限事項を記載します。

HP-UX 10.10 は PA-RISC 1.0 アーキテクチャのマシンをサポートする最後の OS です。HP-UX 10.30 では、PA-RISC 1.0 コードの実行をサポートしません。また、10.30 コンパイラは PA-RISC 1.0 コードのコンパイルをサポートしません。

システム・レベルのバイナリ互換性情報については、『Release Notes for HP-UX 10.30』を参照してください。

既知の問題点

HP C++に関する問題とバグ修正のリストは、HP エレクトロニック・サポート・センタの問題解決情報データベース (Technical Knowledge Database) にあります。URL は次の通りです。

<http://us-support.external.hp.com/>

<http://europe-support.external.hp.com/>

問題解決情報データベースは、サポート契約を結んでいる場合に限りご利用いただけます。

手順 1. Web ページにアクセスし、User ID と Password を入力した後、[Okay]をクリックします。

手順 2. 次のページに移動したら、**Technical Knowledge Database** をクリックします。

手順 3. **Search Options** の欄で、検索条件を指定します。**System Type** に HPUX が含まれていることと、**Document Type** に Service Request が含まれていることを確認してください。

手順 4. **Search String** フィールドに HPCPLUSPLUSA を入力し、[Search]をクリックします。問題点とバグ修正のリストが表示されます。

HP C++コンパイラの製品番号とバージョンを確認するには、次の HP-UX コマンドを実行します。

```
what /opt/CC/lbin/cfront
```

```
what /opt/CC/bin/CC
```

既知の制限事項

既知の制限事項の一部とその対処法については、本書の中で詳しく説明しています。制限事項の中には特定のプラットフォームだけに適用されるものがあることに注意してください。

既知の制限事項の完全なリストについては、『HP C++ Online Programmer's Guide』で“limitations”というキーワードを使用して検索してください。

ライブラリ・ルーチンを使用する前に浮動小数点例外をトリガする

浮動小数点演算を使用するプログラムでは、浮動小数点例外をトリガしてからライブラリ・ルーチンを指定する必要があります。例えば、浮動小数点除算の後に浮動小数点のストアがなければなりません。そうでないと、ライブラリ内のコードが浮動小数点例外をトリガし、記述したコードではなくライブラリ・コードが割り込まれることがあります。

本リリースでこのような動作が引き起こされる原因は、libcl.a および libcl.sl の Unwind コンポーネント内で、旧バージョンの同ライブラリの場合よりも浮動小数点演算を多く使用しているためです。Unwind ライブラリ・ルーチンを指定する前に浮動小数点をトリガしないプログラムでは、Unwind ルーチンで例外がトリガされます。

setjmp/longjmp と+eh オプション

コンパイラ・モードで+eh オプションを使用してコンパイルされたコードは、setjmp/longjmp を使用してはいけません。翻訳モードで+eh オプションを付けて setjmp/longjmp を使用するには、すべての setjmp/longjmp 呼び出しを Setjmp/Longjmp に置き換えてください。また、#include を <setjmp.h> から <Setjmp.h> へ変更することも必要です。

サポートされていないカーネル・スレッド

HP C++ が生成するコードのスレッド・セーフ・レベルは、+eh 使用コードまたは関数スコープ静的変数に対する静的コンストラクタの例外により、どの libC ルーチンが呼び出されたかによって異なります。

また、スレッド・セーフ・レベルは、特定のインタフェースとスレッドのタイプにも依存しています。

表 4-1 スレッド・セーフ・レベル

	カーネル・スレッド	ユーザー・スレッド
生成されるコード		
関数スコープ静的変数	スレッド制限付き C。ユーザーはローカル変数を持ちます。	スレッド制限付き C。ユーザーはローカル変数を持ちます。
ファイル・スコープ静的変数とグローバル変数	共有ライブラリの動的ローディングに対するスレッド制限付き C。初期化の順序設定は問題となる可能性があります。	共有ライブラリの動的ローディングに対するスレッド制限付き C。初期化の順序設定は問題となる可能性があります。
+eh コード	スレッド制限付き A。他のスレッドが C でない限り、スレッドは使用できません。	スレッド・セーフの性能が低下されます。
libC インタフェース		
+eh ((再)スロー)	スレッド制限付き A。C++ で書かれたスレッドが 1 つの場合に限り、スレッド制限付き B です。	スレッド・セーフの性能が低下されます。
I/O (iostreams, ostream など)	スレッド制限付き C。	スレッド・セーフの性能が低下されます。定義済みストリーム (cin, cout, cerr, clog) を使用している場合は調整されます。これ以外の場合は、スレッド制限付き C です。
vec new/delete	スレッド制限付き C。	スレッド・セーフの性能が低下されます。
cxxshl_load と cxxshl_unload	スレッド制限付き C。	スレッド制限付き C。
その他	おそらく、静的/グローバル変数への参照はないので、完全に安全です。libC は fork セーフではありません。取り消しは不可能とみなされます。	おそらく、静的/グローバル変数への参照はないので、完全に安全です。libC は fork セーフではありません。取り消しは不可能とみなされます。