HP OpenVMS
V8.2 新機能説明書

BA322-90009

2005 年 4 月

本書では， OpenVMS Alpha および I64 V8.2 オペレーティング・システムの新機能と，このソフトウェアのドキュメントの概要について説明します。

改訂/更新情報: 新規マニュアルです。

ソフトウェア・バージョン: OpenVMS I64 Version 8.2
OpenVMS Alpha Version 8.2

日本ヒューレット・パッカード株式会社

© Copyright 2005 Hewlett-Packard Development Company, L.P.

本書の著作権は日本ヒューレット・パッカード株式会社およびその子会社が保有しており，本書中の解説および図，表は日本ヒューレット・パッカードの文書による許可なしに，その全体または一部を，いかなる場合にも再版あるいは複製することを禁じます。

また，本書に記載されている事項は，予告なく変更されることがありますので，あらかじめご承知おきください。万一，本書の記述に誤りがあった場合でも，弊社は一切その責任を負いかねます。

本書で解説するソフトウェア (対象ソフトウェア) は，所定のライセンス契約が締結された場合に限り，その使用あるいは複製が許可されます。

日本ヒューレット・パッカードは，弊社または弊社の指定する会社から納入された機器以外の機器で対象ソフトウェアを使用した場合，その性能あるいは信頼性について一切責任を負いかねます。

Adobe および Acrobat は，米国 Adobe Systems 社の登録商標です。

Intel および Itanium は，米国およびその他の国における， Intel Corporation またはその関連会社の登録商標です。

Java は，米国 Sun Microsystems 社の米国における商標です。

Linux は， Linus Torvalds 氏の米国における登録商標です。

Macintosh は，米国アップルコンピュータ社の商標です。

Microsoft および Windows は，米国 Microsoft 社の米国ならびに他の国における商標です。

UNIX は， The Open Group の登録商標です。

原典： HP OpenVMS Version 8.2 New Features and Documentation Overview
© 2005 Hewlett-Packard Development Company, L.P.

本書は，日本語 VAX DOCUMENT V 2.1を用いて作成しています。

目次

例

図

表

まえがき

本書の対象読者

本書は， HP OpenVMS オペレーティング・システムを使用する一般ユーザ，システム管理者，プログラマを対象としています。

本書では OpenVMS Version 8.2 の新機能を説明します。新機能が使用中のシステムに与える影響について， Version 8.2 をインストール，アップグレード，使用する前にリリース・ノートをお読みください。

本書の構成

本書の構成は以下のとおりです。

- 第 1 部， OpenVMS Version 8.2 の新機能
  - 第 1 章， OpenVMS の新機能について概要を説明しています。
  - 第 2 章， OpenVMS オペレーティング・システムの一般ユーザ向けの新機能について説明しています。
  - 第 3 章，システム管理作業に関する新機能について説明しています。
  - 第 4 章，プログラミングの新機能について説明しています。
  - 第 5 章，重要なレイヤード・プロダクトの新機能について説明しています。
  - 第 6 章， Volume Shadowing for OpenVMS でのホスト・ベース・ミニマージ (HBMM) について説明しています。
  - 第 7 章， OpenVMS Version 8.2 での Linker の新機能の概要，および OpenVMS I64 システムでプログラムをリンクする前に検討すべき相違点や考慮点についても説明しています。
- 第 2 部， OpenVMS のドキュメント
  - 第 8 章， OpenVMS ドキュメントの変更点について説明しています。
  - 第 9 章，ドキュメントの入手方法について説明しています。
  - 第 10 章， OpenVMS ドキュメント・セットに含まれている各ドキュメントについて説明しています。

関連資料

HP OpenVMS 製品とサービスに関するその他の情報については，次の Web サイトをご覧ください。

http://www.hp.com/jp/openvms
http://www.hp.com/go/openvms

本書で使用する表記法

本書では，次の表記法を使用しています。

表記法	意味
Ctrl/x	Ctrl/xという表記は， Ctrl キーを押しながら別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
PF1 x	PF1 xという表記は， PF1 に定義されたキーを押してから，別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
Return	例の中で，キー名が四角で囲まれている場合には，キーボード上でそのキーを押すことを示します。テキストの中では，キー名は四角で囲まれていません。 HTML 形式のドキュメントでは，キー名は四角ではなく，括弧で囲まれています。
. . .	例の中の水平方向の反復記号は，次のいずれかを示します。 • 文中のオプションの引数が省略されている。 • 前出の 1 つまたは複数の項目を繰り返すことができる。 • パラメータや値などの情報をさらに入力できる。
. . .	垂直方向の反復記号は，コードの例やコマンド形式の中の項目が省略されていることを示します。このように項目が省略されるのは，その項目が説明している内容にとって重要ではないからです。
( )	コマンドの形式の説明において，括弧は，複数のオプションを選択した場合に，選択したオプションを括弧で囲まなければならないことを示しています。
[ ]	コマンドの形式の説明において，大括弧で囲まれた要素は任意のオプションです。オプションをすべて選択しても，いずれか 1 つを選択しても，あるいは 1 つも選択しなくても構いません。ただし， OpenVMS ファイル指定のディレクトリ名の構文や，割り当て文の部分文字列指定の構文の中では，大括弧に囲まれた要素は省略できません。
[|]	コマンド形式の説明では，括弧内の要素を分けている垂直棒線はオプションを 1 つまたは複数選択するか，または何も選択しないことを意味します。
{ }	コマンドの形式の説明において，中括弧で囲まれた要素は必須オプションです。いずれか 1 つのオプションを指定しなければなりません。
太字	太字のテキストは，新しい用語，引数，属性，条件を示しています。また，変数を示す場合にも使用されています。

表記法	意味
italic text	イタリック体のテキストは，重要な情報を示します。また，システム・メッセージ (たとえば内部エラー*number*)，コマンド行 (たとえば /PRODUCER=*name*)，コマンド・パラメータ (たとえば*device-name*) などの変数を示す場合にも使用されます。
UPPERCASE TEXT	英大文字のテキストは，コマンド，ルーチン名，ファイル名，ファイル保護コード名，システム特権の短縮形を示します。
`Monospace type`	モノスペース・タイプの文字は，コード例および会話型の画面表示を示します。 C プログラミング言語では，テキスト中のモノスペース・タイプの文字は，キーワード，別々にコンパイルされた外部関数およびファイルの名前，構文の要約，または例に示される変数または識別子への参照などを示します。
-	コマンド形式の記述の最後，コマンド行，コード・ラインにおいて，ハイフンは，要求に対する引数がその後の行に続くことを示します。
数字	特に明記しない限り，本文中の数字はすべて 10 進数です。 10 進数以外 (2 進数， 8 進数， 16 進数) は，その旨を明記してあります。

第1部
OpenVMS Version 8.2の新機能

1

HP OpenVMS Version 8.2 の新機能の概要

OpenVMS Version 8.2 は，新しいプラットフォームをサポートする OpenVMS オペレーティング・システムのメジャー・リリースであり，新たに HP Integrity サーバをサポートします。 OpenVMS は， 1 年 365 日 24 時間体制 (24 × 365) で休みなく稼働する環境にとって不可欠な，最高レベルの可用性，拡張性，柔軟性，性能，セキュリティを提供します。 OpenVMS には 20 年以上にわたって高い信頼性を提供してきた実績があり，さらに新しいテクノロジをベース・オペレーティング・システムおよび OpenVMS Cluster ソフトウェア環境に組み込むことにより可用性と性能を強化しています。

1.1 新機能のまとめ

OpenVMS Version 8.2 では， OpenVMS Version 7.3–2 のすべての機能に加えて，OpenVMS オペレーティング・システムに新機能が追加されています。表 1–1 では，OpenVMS Version 8.2 に用意されている各機能の概要を機能要素 (一般ユーザ，システム管理，プログラミング，および関連製品) 別に示します。

表 1–1 OpenVMS Version 8.2 ソフトウェアの機能概要

一般ユーザ機能	
ドキュメント	ドキュメントの最新版がインターネット上で入手できるようになりました。 このリリースでは， 1 冊の新しいマニュアル『HP OpenVMS Alpha から OpenVMS I64 へのアプリケーション・ポーティング・ガイド』が作成されました。 OpenVMS Volume Shadowing がサポートするホスト・ベース・ミニマージ (HBMM) は，第 6 章で説明します。 Linker ユーティリティの新機能は，第 7 章で説明します。
DCL コマンドとレキシカル関数	Version 8.2 には，新規または変更された DCL コマンド，修飾子，レキシカル関数があります。
EV7 チップの速度向上	AlphaServer ES47， ES80，および GS1280 システムでは，速度が向上した新しい EV7 チップがサポートされます。
LMF の機能拡張	Integrity サーバの OE パッケージと I64 システムでのプロセッサごとのライセンス管理が追加されました。

(次ページに続く)

表 1–1 (続き) OpenVMS Version 8.2 ソフトウェアの機能概要

一般ユーザ機能	
Monitor ユーティリティの機能拡張	Monitor ユーティリティの移植時に，いくつかの機能が拡張されました。
オペレーティング環境	OpenVMS I64 は， 3 種類のオペレーティング環境として提供されます。 Foundation Operating Environment (FOE)， Enterprise Operating Environment (EOE)， Mission Critical Operating Environment (MCOE) です。
システム管理機能	
クラスタ	OpenVMS Cluster ソフトウェアでは，現在， OpenVMS Alpha システムと OpenVMS VAX システムで提供されている機能とほぼ同様の機能が OpenVMS I64 システムで提供されます。
Performance Data Collector (TDC)	OpenVMS Version 7.3-2 以降が稼働しているシステムの性能データを収集します。
OpenVMS I64 Boot Manager ユーティリティ	メニュー方式のユーティリティで， OpenVMS I64 が稼働する Integrity サーバでの EFI ブート・オプションの管理を容易にします。
OpenVMS 用 EFI ユーティリティ	I64 システムの EFI コンソールでデバイス管理機能を提供します。
Fibre Channel HBA サポート	OpenVMS Version 8.2 では， 2 つの新しい Fibre Channel ホスト・ベース・アダプタ (HBA) をサポートします。 1 つは OpenVMS I64 システム用で，もう 1 つは OpenVMS Alpha 用です。
ホスト・ベース・ミニマージ (HBMM)	HBMM では，マージ操作に必要な比較の回数を減らすことにより，マージ操作の性能が向上しました。
System Dump Analyzer	新しいコマンドと修飾子が追加され，変更も行われました。変更のうちの一部は I64 システムに対応するために行われました。
ボリューム・シャドウイングのマージ操作とコピー操作の優先順位付け	SET SHADOW コマンドの 2 つの新しい修飾子と新しいシステム・パラメータを使って，システムごとにシャドウ・セットのマージ操作とコピー操作に優先順位を付けることができるようになりました。また，マージ操作とコピー操作を行なう場所も操作できます。

(次ページに続く)

表 1–1 (続き) OpenVMS Version 8.2 ソフトウェアの機能概要

システム管理機能	
新しいシステム・パラメータ	— ERLBUFFERPAG_S2 : 各 S2 空間エラー・ログ・バッファに割り当てる S2 空間メモリのサイズを指定します。 — ERRORLOGBUFF_S2 : システム・エラー・ログ・エントリ用に確保する S2 空間エラー・ログ・バッファの数を指定します。 — SCSI_ERROR_POLL : OpenVMS が各 SCSI ディスクに 1 時間ごとに SCSI Test Unit Ready コマンドを発行するように指示します。 — SHADOW_ENABLE : 弊社で使用するために予約されている特別なパラメータです。 — SHADOW_HBMM_RTC : ホスト・ベース・ミニマージ (HBMM) ビットマップを持つシャドウ・セットのリセットしきい値をシステムがチェックする時間間隔を制御します。 — SHADOW_PSM_DLY : シャドウイングで追加される遅延時間を，システム管理者が調整するために使います。 — SHADOW_REC_DLY : シャドウ・セットの回復操作の管理を試みるまでにシステムが待つ時間を決定するのを助けます。 — SHADOW_SITE_ID : Volume Shadowing が読み込み操作を行う最適のデバイスを決定するためのサイトの値を，システム管理者が定義するために使います。 — SYSSER_LOGGING : プロセスに対するシステム・サービス要求のロギングを有効にします。 — VHPT_SIZE : システム内の各 CPU の仮想ハッシュ・ページ・テーブル (VHPT) に割り当てるサイズを KB 単位で制御します。
タイムゾーン	追加タイムゾーンをデータベースに追加して，サポートするようになりました。
OpenVMS I64 での Ultra SCSI アダプタのサポート	OpenVMS I64 は 2 つの Ultra-SCSI ホスト・ベース・アダプタ (HBA) をサポートします。 Ultra-160 デュアル・チャネルと Ultra-320 デュアル・チャネルです。
プログラミング機能	
ANALYZE ユーティリティの機能拡張 (I64 のみ)	ELF (Executable and Linkable Format) オブジェクト・ファイルとイメージ・ファイルを解析します。
HP C 実行時ライブラリ (CRTL) の機能拡張	— ファイル・ロック機能 — 標準に準拠した stat 構造体 — ファイル・システムの統計サポート — fcntl のファイル・ステータス・フラグ — UNIX スタイルのパイプのサポート — 新しい論理名 DECC$POPEN_NO_CRLF_REC_ATTR — glob と globfree の 64 ビット・サポート — socketpair — stat 関数の機能拡張
呼び出し規則	OpenVMS I64 システムでの Intel Itanium 呼び出し規則をサポート。ただし，いくつかの例外があります。
Checksum ユーティリティ	CHECKSUM/OBJECT コマンドを I64 オブジェクト用に機能強化しました。
DCE RPC	OpenVMS Alpha システムと OpenVMS I64 システムの両方で， G_FLOAT と IEEE の浮動小数点型をサポートします。

(次ページに続く)

表 1–1 (続き) OpenVMS Version 8.2 ソフトウェアの機能概要

プログラミング機能	
拡張ロック値ブロック	OpenVMS ロック・マネージャに 64 バイトのロック値ブロックのサポートが追加されました。ただし，既存のアプリケーションでは 16 バイトの値ブロックを使います。拡張値ブロックを使うためには，ソースを変更する必要があります。
HP OpenVMS 移行ソフトウェア	Alpha の実行イメージまたは共有イメージを， OpenVMS I64 システムで動作可能なトランスレート形式のイメージに変換します。
新しい RTL LIB$ルーチン	ルーチンとその機能についての詳細は，第 4.12 節を参照してください。
新しい RTL OTS$ルーチン	ルーチンとその機能についての詳細は，第 4.13 節を参照してください。
Patch ユーティリティ	Patch ユーティリティが OpenVMS Alpha システムと OpenVMS I64 システムでも使用できるようになりました。
POSIX スレッド機能	– プロセス共用のミューテクス変数と条件変数 – SET コマンドと SHOW コマンドの機能拡張 (I64 のみ) – スレッド独立のサービスに新しいルーチン tis_mutex_init_type を追加
新しいシステム・サービス	— $CLEAR_UNWIND_TABLE — $GET_UNWIND_ENTRY_INFO — $GOTO_UNWIND_64 — $IEEE_SET_ROUNDING_MODE — $IEEE-SET PRECISION_MODE — $RPCC_64 — $SET_RETURN_VALUE — $SET_UNWIND_TABLE
Traceback Application Programming Interface for OpenVMS I64	ユーザ・アプリケーションからトレースバック情報にアクセスするために使用します。

(次ページに続く)

表 1–1 (続き) OpenVMS Version 8.2 ソフトウェアの機能概要

	関連製品の機能
Common Data Security Architecture (CDSA) Version 2.1	新しい暗号タイプをサポートします。
Kerberos for OpenVMS	Kerberos Version 2.1 for OpenVMS は， MIT Kerberos V5 Release 1.2.6 with CERT patches up to Release 1.2.8 をベースにしています。 OpenVMS I64， OpenVMS Alpha，および OpenVMS VAX 上で Kerberos のクライアントとサーバをサポートします。 Kerberos for OpenVMS Version 2.1 の新機能として，管理者が Kerberos V5 keytab ファイルや V4 srvtab ファイルのエントリの読み取り，書き込み，編集を行うためのメニューを呼び出す`ktutil`コマンドが含まれています。
HP OpenVMS Management Station Version 3.2-D	OpenVMS Alpha Version 8.2 には， OpenVMS Management Station Version 3.2-D が含まれています。
HP SSL for OpenVMS	HP SSL Version 1.2 は OpenSSL 0.9.7d をベースとし， openssl.org によって報告されたセキュリティの脆弱性に関する不具合を修正したものです。 OpenVMS I64， OpenVMS Alpha，および OpenVMS VAX 用に HP SSL のサポートが用意されています。 HP SSL Version 1.2 の新機能には， OCSP (Online Certificate Status Protocol)， AES (Advanced Encryption Standard)，および楕円曲線暗号化機能が含まれます。
HP TCP/IP Services for OpenVMS	OpenVMS Version 8.2 では，この製品の Version 5.5 をサポートします。
UNIX ポータビリティ機能	UNIX アプリケーションの OpenVMS への移植を簡単にするために，UNIX ポータビリティ機能が追加されました。
Open Source Tools for OpenVMS	OpenVMS Version 8.2に添付されている Open Source Tools CD には，UNIX から OpenVMS への移植を効率的に行なうためのユーティリティやソースが含まれています。

1.2 OpenVMS Alpha システムと OpenVMS I64 システムの主な相違点

以下に， Alpha プラットフォームと I64 プラットフォームで違いが顕著な項目をいくつかリストアップし，簡単に説明します。

- コンソールとコンソール回線接続

  Integrity サーバでは，現在の Integrity プラットフォームにおいて 2 つの異なるシリアル回線でのコンソール回線接続に対応しています。 1 つのシリアル回線は BMC コンソールに接続し，もう 1 つはオプションで Management Processor (MP) に接続します。 Management Processor では，ネットワーク接続と TELNET アクセスができ，より柔軟な管理が可能です。

  どちらのコンソールもハードウェアの管理機能を持ち，たとえば，システムのリセットやシステムの電源の制御 (ON/OFF) を行うことができます。 Management Processor は， BMC コンソールより管理機能が豊富なので， Management Processor の使用をお勧めします。

- システム・コンソールへの接続

BMC コンソールと MP コンソールはどちらもシステム・コンソールと接続可能です。 Integrity サーバでは， EFI シェルと呼ばれます。ブート・デバイスとブート・オプションのセットアップ機構は， Alpha の SRM コンソールとは全く異なります。さらに，いったんシステムがブートしたら， EFI シェルを使用することはできなくなります。 Alpha システムの場合は，システムをブートした後でも SRM コンソールに戻ることができます。

- Integrity サーバのブート可能デバイス

  Integrity サーバでブート可能なデバイスは，初期ブートストラップ・プログラムが格納されている FAT パーティションを持っています。 EFI シェルは，そのパーティションを検索して読み込むことができます。システムは， FAT パーティション内にある vms_loader.efi という名の初期ブートストラップ・プログラムを実行することで，ブートされます。

  ブート・デバイスを指定する環境変数はありません。その代わり，特定のディスクに対して vms_loader.efi を実行するメニュー項目を EFI シェルで作成することができます。

- Integrity サーバのコンソール環境変数

  ブート・フラグは，*vms_flags*で指定可能です。 Alpha では，名前 BOOT_OSFLAGS を使用します。 SCSI デバイスや LAN デバイスの構成を指定する環境変数はありません。 LAN 構成の設定は， LANCP プログラムで行う必要があります。

- ライセンス方式

  LMF 管理機能は基本的には変更はありません。ただし， OpenVMS I64 での新しいライセンス方式は， OpenVMS Alpha および OpenVMS VAX での方式とは異なります。主な相違点は，以下のとおりです。

  — ライセンス単位の割り当て方法とカウント方法

  — 提供されるライセンスの種類

  — アップデート・ライセンス:

    — I64 システムでは，アップデート・ライセンスは提供されません。有効な Software Updates Service をお持ちのお客様は，ソフトウェアの新しいリビジョンの提供を受けることができます。お持ちでないお客様は，製品を再度購入していただく必要があります。 Software Updates Service およびその他の HP Service に関する情報は，次の Web サイトを参照してください。

      `http://www.hp.com/hps/software`

    — Alpha システムでは，これまでどおり，サービス契約をしていないお客様もサービス契約が切れているお客様も，アップデート・ライセンスを入手可能です。

  OpenVMS I64 システムのライセンス管理に関する詳細は，『OpenVMS License Management Utility Manual』を参照してください。

- 特権アーキテクチャ・ライブラリ・コード (PALcode) 機能

  OpenVMS Alpha では， PALcode を呼び出して，メモリ・バリア (MB)，停止 (Halt)，バグ・チェック (Bugcheck) といったアトミック機能を実行します。これらの機能は，コンソール・ファームウェアに組み込む代わりに， OpenVMS I64 に組み込まれました。

- OpenVMS 呼び出し標準規則

  Intel® Itanium®プロセッサ・ファミリでの OpenVMS 呼び出し標準規則の実装は， Intel Calling Standard をベースに行われました。 VAX および Alpha のコードとの互換性を確保するために OpenVMS 用に拡張されています。

  現在のアプリケーションで，呼び出し標準規則の内部フォーマットの詳細を意識している場合は， OpenVMS Version 8.2 での OpenVMS 呼び出し標準規則の実装について説明された『OpenVMS Calling Standard』を参照してください。

- IEEE 浮動小数点フォーマット

  Alpha Server では， VAX 浮動小数点データ型と IEEE 浮動小数点データ型をハードウェアでサポートしています。 Integrity サーバでは， IEEE 浮動小数点はハードウェアでサポートしますが， VAX 浮動小数点データ型はソフトウェアでサポートします。 OpenVMS I64 コンパイラには， VAX 浮動小数点データ型を使用するための/FLOAT_D_FLOAT 修飾子と/FLOAT_G_FLOAT 修飾子が用意されています。これらの修飾子のいずれも指定しない場合， IEEE 浮動小数点データ型が使用されます。

  OpenVMS Version 8.2 で使用できる IEEE 浮動小数点データ型については，『HP OpenVMS Alpha から OpenVMS I64 へのアプリケーション・ポーティング・ガイド』を参照してください。

Integrity サーバ・システムの使用を開始する前に，『HP OpenVMS Version 8.2 リリース・ノート[翻訳版]』，『HP OpenVMS Version 8.2 Upgrade and Installation Manual』および本書をお読みください。

2
一般ユーザ機能

この章では，HP OpenVMS Alpha オペレーティング・システムと OpenVMS I64 オペレーティング・システムのすべてのユーザに関連する，新機能について説明します。

2.1 DCL コマンドとレキシカル関数

表 2–1 と表 2–2 に，OpenVMS Version 8.2 の新規または変更された DCL コマンド，修飾子，レキシカル関数の概要を示します。詳細は，オンライン・ヘルプ，または『OpenVMS DCL ディクショナリ』を参照してください。

表 2–1 DCL コマンドと DCL ドキュメントのアップデート

DCL コマンド	ドキュメントのアップデート
ANALYZE/IMAGE	新しい修飾子: /FLAG_VALUES，/SECTIONS，/SEGMENTS，/SELECT
ANALYZE/OBJECT	新しい修飾子: /FLAG_VALUES，/SECTIONS，/SELECT
ANALYZE/SSLOG	新しいコマンド。詳細は，『OpenVMS System Analysis Tools Manual』を参照してください。
APPEND	新しい/BLOCK_SIZE 修飾子
ASSIGN	新しい/CLUSTER_SYSTEM 修飾子
CHECKSUM	新しいコマンド
COPY	新しい/BLOCK_SIZE 修飾子
CREATE/MAILBOX	新しいコマンド
DEASSIGN	新しい/CLUSTER_SYSTEM 修飾子
DEFINE	新しい/CLUSTER_SYSTEM 修飾子
DELETE ’file’	新しい/GRAND_TOTAL 修飾子
DELETE/BITMAP	新しいコマンド
DELETE/MAILBOX	新しいコマンド
DIRECTORY	/SELECT 用の新しいキーワード VERSION
INITIALIZE	新しい/GPT 修飾子，/ERASE 用の新しいキーワード，/LIMIT の変更，および/CLUSTER_SIZE の省略時の指定を 16 に引き上げ
OPEN	新しい/NOSHARE 修飾子
PATCH	VAX でのみ使用できるコマンドだったが，3 つのプラットフォームすべてで実行できるようになりました。
PURGE	新しい/GRAND_TOTAL 修飾子

(次ページに続く)

表 2–1 (続き) DCL コマンドと DCL ドキュメントのアップデート

DCL コマンド	ドキュメントのアップデート
SEARCH	新しい修飾子: /LIMIT，/SKIP，/WILDCARD_MATCHING
SET BOOTBLOCK	新しいコマンド (I64 のみ)
SET DISPLAY	IPv6 (Internet Protocol version 6) をサポートするために/TRANSPORT と /PMTRANSPORT へ値を追加
SET IMAGE	新しいコマンド
SET PROCESS	新しい/TOKEN 修飾子と/SSLOG 修飾子，およびアップデートされた /RESOURCE_WAIT 修飾子
SET SERVER	/CONFIGURE の説明の改訂。 SET SERVER は独立した 3 つのコマンドごとにドキュメント化され，オンライン・ヘルプがわかりやすくなりました。
SET SHADOW	ホスト・ベース・ミニマージをサポートするためのいくつかの機能追加
SET TERMINAL	新しい/BACKSPACE 修飾子
SHOW DEVICES	/FULL を指定した場合にデータをバイト単位で表示するためのサポートを追加
SHOW FASTPATH	新しいコマンド
SHOW IMAGE	新しいコマンド
SHOW LICENSE	新しい修飾子/HIERARCHY と/OE
SHOW LOGICAL	新しい/CLUSTER 修飾子，/FULL に対する表示の拡張
SHOW PROCESS	新しい/CASE_LOOKUP 修飾子と/TOKEN 修飾子。 Red Sox ファンの例の改訂
SHOW SERVER	SHOW SERVER は独立した 2 つのコマンドとしてドキュメント化され，オンライン・ヘルプがわかりやすくなりました。
SHOW SHADOW	ホスト・ベースのミニマージをサポートするためのいくつかの機能を追加
SHOW SYSTEM	新しい/IMAGE 修飾子
SHOW TERMINAL	表示の末尾の新しいフィールド
SHOW WORKING_SETS	バイト単位で表示するための新しい機能
WRITE	新しい/WAIT[/NOWAIT]修飾子

表 2–2 DCL レキシカル関数とそのドキュメントのアップデート

DCL レキシカル関数	ドキュメントのアップデート
F$FID_TO_NAME	新しいレキシカル関数
F$GETDVI	新しいpathname引数と多数の新しい項目コード
F$GETJPI	新しい TOKEN アイテム・コード
F$GETSYI	アップデートされたアイテム・コードのリスト
F$LICENSE	新しいレキシカル関数
F$MULTIPATH	新しいレキシカル関数
F$UNIQUE	新しいレキシカル関数

2.2 LMF の機能拡張

LMF (License Management Facility) がアップデートされ， I64 システム用の OpenVMS で 2 つの販売方法がサポートされました。オペレーティング環境 (OE) とプロセッサ単位のライセンス (PPL) です。

オペレーティング環境には，統合パッケージとして，オペレーティング・システムとバンドル・アプリケーションが含まれます。現在， OpenVMS I64 システムには，次の 3 つのオペレーティング環境が用意されています。

- HP OpenVMS Foundation Operating Environment (FOE)
- HP OpenVMS Enterprise Operating Environment (EOE)
- HP OpenVMS Mission Critical Operating Environment (MCOE)

LMF コマンドの新しい修飾子を使用することで, オペレーティング環境のライセンスを管理できます。ご購入いただいた製品に応じて， 1 ライセンスで FOE， EOE，または MCOE を使用できるようになりました。この変更により， LMF 管理の複雑さが軽減され，運用の自由度が向上しました。

オペレーティング環境を使用するためには，新しいライセンス形態である PPL (プロセッサ単位のライセンス) が必要です。 I64 システムのアクティブ・プロセッサごとにライセンスが必要です。

ライセンス管理についての詳細は，『OpenVMS License Management Utility Manual』を参照してください。

2.3 Monitor ユーティリティの機能拡張

Monitor ユーティリティが OpenVMS I64 に移植され， OpenVMS Alpha システムと OpenVMS I64 システムの両方でネイティブ・ユーティリティとして提供されています。移植の際に，次のようにいくつかの機能拡張が行なわれました。

- いくつかのデータ・クラス (たとえば， DISK) で，情報を表示する際に，画面の高さ全体が使われるようになりました。
- SYSTEM クラスでは， Process States に「現在の」プロセスの数が表示されるようになりました。これまでは，現在のプロセスは，「その他の」カテゴリにグループ化されていました。
- 記録データ・ファイルのフォーマットは，今回のリリースで変更されました。フォーマットの変更は，記録データのアラインメントを改善するために行われました。 MONITOR ユーティリティのレコード・フォーマットの詳細は，『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル』の付録を参照してください。ただし，この変更を行なったために， V8.2 より前のシステムでは，新しいフォーマットの記録データ・ファイルは読み込めません。そのため，新しいフォーマットのデータ・ファイルを V8.2 より前のフォーマットに変換するユーティリティ SYS$EXAMPLES:MONITOR_CONVERT.C が，SYS$EXAMPLES に用意されました。実行プログラムも用意されています。このユーティリティを使用するには，次のコマンドを実行します。

```
$ mc system$examples:monitor_convert input-file output-file
```

- MONITOR データの収集ルーチンには，各種の性能改良が行なわれ， Monitor ユーティリティを使用する際のシステム・オーバヘッドが減少しました。

2.4 OpenVMS I64 のオペレーティング環境 (OE)

OpenVMS Version 8.2 では， OpenVMS I64 システム用の OpenVMS オペレーティング・システムとレイヤード・プロダクトに，新しい提供方法が導入されました。OpenVMS I64 オペレーティング・システムでは， OpenVMS Alpha オペレーティング・システムとは異なり， Integrity サーバで使用可能な次の 3 つのパッケージが用意されています。

- HP OpenVMS Foundation Operating Environment (FOE) は，価格と性能の両面を重視した，インターネット対応の高機能パッケージです。
- HP OpenVMS Enterprise Operating Environment (EOE) は，管理容易性と，シングル・システムでの可用性および性能をさらに高めたパッケージです。
- HP OpenVMS Mission Critical Operating Environment (MCOE) は，マルチ・システムでの可用性とワークロードの管理において最高の機能を提供する最上位パッケージです。

これら 3 つのオペレーティング環境が， 1 枚の DVD に収められています。ライセンス契約書に応じて，アクセス可能なオペレーティング環境が決まります。

注意

これらの OpenVMS I64 オペレーティング環境では，システム処理能力に対するライセンスではなく，プロセッサごとのライセンス (PPL) となります。Alpha でのライセンス方式には変更はありません。

技術仕様の総覧は，次の Software Products Description の Web サイトを参照してください。

http://www.hp.com/info/spd

HP OpenVMS Operating Environment に関しては，弊社営業担当者までお問い合わせください。

3
システム管理機能

この章では，システム管理に関連する，新機能，変更点，拡張された機能について説明します。

3.1 OpenVMS I64 Boot Manager (BOOT_OPTIONS.COM) ユーティリティ

OpenVMS I64 Boot Manager (BOOT_OPTIONS.COM) ユーティリティは，メニュー方式のユーティリティで， OpenVMS I64 が稼働する Integrity サーバでの EFI ブート・オプションの管理を容易にします。このユーティリティでは次の作業を行えます。

- Integrity サーバに，システム・ディスク，ダンプ・デバイス，デバッグ・デバイスのブート・オプションを設定する。
- OpenVMS ブート・エントリを表示する。
- EFI Boot Manager でのエントリの位置を決める。たとえば，システム・ディスクをオプション・リストの 1 番に指定すれば，システムの電源投入時やリブート時に自動的にそのディスクからブートされるようになります。
- エントリのブート・フラグを設定する。
- エントリを削除する。
- EFI タイムアウトを有効/無効にする (EFI Boot Manager が，ブート・リストの 1 番目または 2 番目の有効なエントリをブートする前の待機時間)。
- ブート・エントリを検証する。
- OpenVMS の動作中にブート・デバイス，ダンプ・デバイス，デバッグ・デバイスを設定する。 Alpha システムと異なり，ブート・デバイスを設定するために，オペレーティング・システムをシャットダウンしてコンソールでコマンドを入力する必要はありません。

OpenVMS I64 のインストール後に，このユーティリティを使用してシステム・ディスクを EFI Boot Manager リスト内の最初のブート・エントリとして追加することをお勧めします。 Fibre Channel ストレージ・デバイスからのブートを設定するためには，このユーティリティを必要とします。他のデバイスではすべて，このユーティリティはオプションです。このユーティリティは使いやすいので， EFI Boot Manager ではなくこのユーティリティをできる限り使用することをお勧めします。 Fibre Channel デバイスの設定に関する情報は，『OpenVMS Cluster 構成ガイド』を参照

してください。 OpenVMS Boot Manager ユーティリティについては，『OpenVMS システム管理者マニュアル』を参照してください。

3.2 OpenVMS I64 システムのクラスタリング

OpenVMS Cluster ソフトウェアでは，現在， OpenVMS Alpha システムと OpenVMS VAX システムに提供されている機能とほとんど同じ機能が OpenVMS I64 システムに提供されます。

OpenVMS Cluster の主要な機能は，以下のとおりです。

- 完全共用型の，複数ノードからのディスクへの読み書きアクセス
- クラスタ単位のファイル・システム
- クラスタ単位のバッチ/印刷キュー・サブシステム
- 分散ロック・マネージャ
- ボート/クォーラム・ベースのメンバシップ管理
- 単一セキュリティ・ドメイン
- 単一システム管理ドメイン
- 豊富なクラスタ単位 API
- 複合アーキテクチャ・クラスタ
- ローリング・アップグレードのサポート
- 複数のインタコネクトのサポート (第 3.2.1 項を参照)
- 複合アーキテクチャ・クラスタでの，最大 16 システムのサポート。このうち，最大 8 システムは， I64 システムとすることができます。
- フェイルオーバと負荷分散
- クラスタのネットワーク別名
- ディスクとテープのサービス
- DTCS (Disaster-Tolerant Cluster Services) を使った最大 800 Km (500 マイル) の距離をサポートするディザスタ・トレラント機能

サテライト・ブートは，今回のリリースではサポートしていません。この機能は将来のリリースでサポートする予定です。

3.2.1 OpenVMS I64 Cluster のインタコネクト・サポート

OpenVMS I64 システムでは，クラスタ間通信 (SCS トラフィック) 用に Ethernet, Fast Ethernet， Gigabit Ethernet を使用することができます。ただし， OpenVMS Alpha システムのクラスタ間通信でサポートされている FDDI と ATM は，OpenVMS I64 システムではサポートされません。

FDDI アダプタと ATM アダプタは， OpenVMS I64 システムでクラスタ・インタコネクトとしてサポートされませんが，マルチ・サイト・クラスタ内でのサイト間のインタコネクトとしてはサポートされます。 OpenVMS I64 ノードの FastEthernet /GigabitEthernet NIC と WAN サプライヤが提供する任意のサイト間のインタコネクト (たとえば T3， E3， SONET， ATM， FDDI， DWDM，他) の接続にはブリッジやスイッチを使用できます。

OpenVMS Cluster ソフトウェアは， Alpha システムでは，次の 3 つの独自のクラスタ・インタコネクトをサポートしていますが， OpenVMS I64 システムではサポートされません。 DSSI (DIGITAL Systems Storage Interconnect)， CI (Cluster Interconnect) と MEMORY CHANNEL です。

DSSI と CI は OpenVMS I64 システムではサポートされませんが， Alpha システムに接続された DSSI ディスクと CI ディスクに格納されたデータは，同一クラスタ内の OpenVMS I64 システムで使用することができます。

OpenVMS I64 システムでは， Fibre Channel を，共用ストレージ・クラスタ・インタコネクト用にサポートしていますが， SCSI はサポートしていません。(OpenVMS Alpha システムでも，最新の SCSI アダプタは共用ストレージ・クラスタ・インタコネクト用の SCSI としてはサポートしていません。)

ただし， OpenVMS I64 システム，または OpenVMS Alpha システムに直接接続された SCSI ディスクに格納されたデータは，クラスタ内の任意のメンバから使用することができます。これは， OpenVMS Cluster システム内でローカルに接続されたすべてのディスクについていえます。

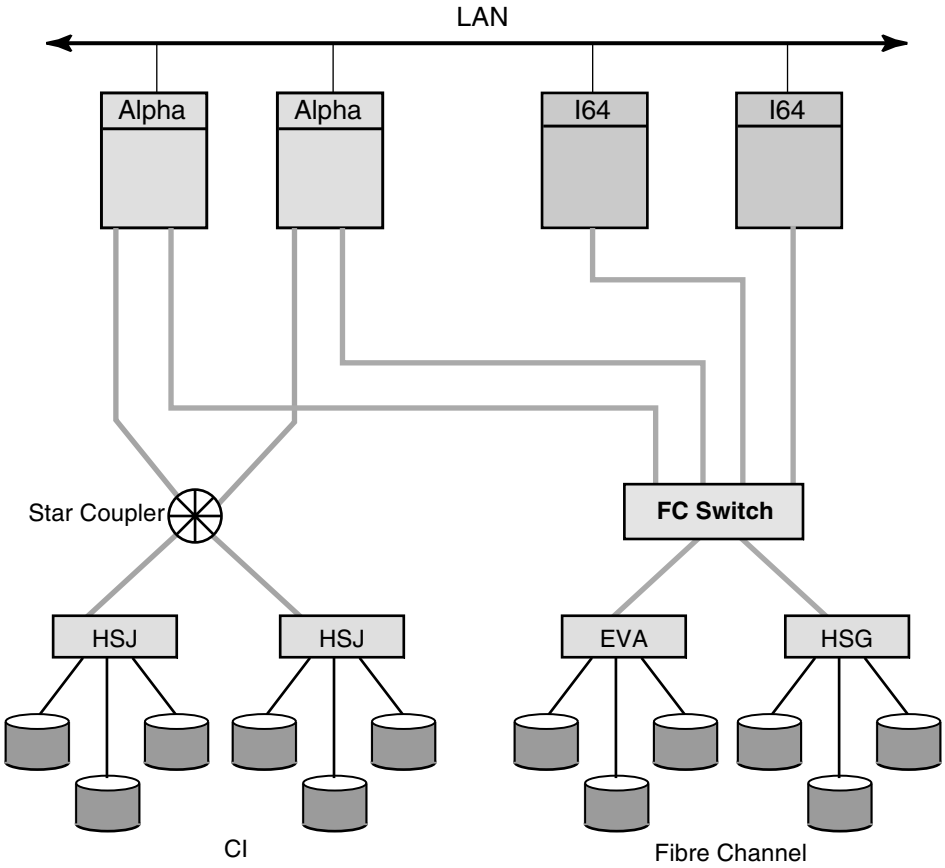
3.2.2 複合アーキテクチャ・クラスタ

OpenVMS は，複合アーキテクチャ・クラスタで OpenVMS Alpha システムと OpenVMS I64 システムの両方をサポートします。この構成でサポートする OpenVMS Alpha のバージョンは， OpenVMS Alpha Version 7.3–2 です。これらのバージョンが混在したクラスタをサポートするためには，『HP OpenVMS Version 8.2 リリース・ノート[翻訳版]』で説明されている修正キットをインストールする必要があります。次の Web サイトにある HP OpenVMS Version 8.2 のドキュメント・セットを参照してください。

```
http://www.hp.com/go/openvms/doc
```

図 3–1 に， OpenVMS I64 システムを追加した OpenVMS Cluster システムを示します。

図 3–1　Alpha システムと I64 システムが組み込まれた OpenVMS Cluster システム



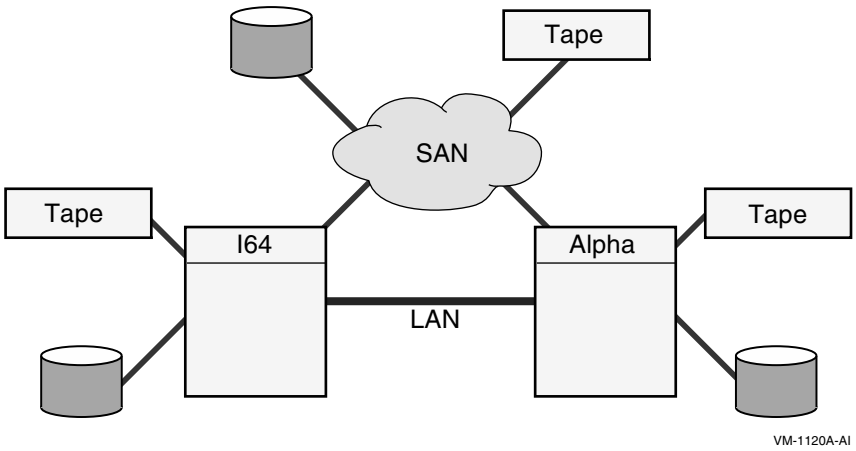
クラスタ内のすべてのシステムで LAN インタコネクトをクラスタ間通信用に使用しています。この構成では， OpenVMS Alpha システムと OpenVMS I64 システムの両方から同じ Fibre Channel ストレージに同時にアクセスできます。 Fibre Channel ディスクに直接接続された I64 システムは， CI ディスクのデータを使用できることに注意してください。 OpenVMS の複合アーキテクチャ・クラスタでは，各アーキテクチャに対して少なくとも 1 つのシステム・ディスクが必要です。今回のリリースでは，クラスタで最大 8 台の I64 システムをサポートします。複合アーキテクチャ・クラスタでは，合計が最大 16 システムとなるように，最大 8 台の I64 システムを Alpha システムに混在させることができます。

3.2.2.1 複合アーキテクチャ・クラスタでのストレージ

この項では， OpenVMS I64 システムと OpenVMS Alpha システムで構成した複合アーキテクチャ・クラスタでの，システム・ディスクを含むストレージに関連する規則について説明します。

図 3–2 に，ローカルに接続したストレージと共用 SAN (Storage Area Network) を持つ， OpenVMS I64 システムと OpenVMS Alpha システムの複合アーキテクチャ・クラスタの単純な例を示します。

図 3–2 複合アーキテクチャ OpenVMS Cluster のストレージ



複合アーキテクチャ OpenVMS Cluster システム内の I64 システムの特徴は，次のとおりです。

- ローカル・ディスク，または共用 Fibre Channel ディスクとして， 1 つの I64 システム・ディスクを持つこと。
- サーバがサービスする Alpha ディスクと Alpha テープを使用できる。
- SAN のディスクとテープを使用できる。
- 同じ SAN データ・ディスクを Alpha システムと共用できる。
- I64 システムと Alpha システムの両方のクラスタ・メンバに，ディスクとテープをサービスすることができる。

複合アーキテクチャ OpenVMS Cluster システム内の Alpha システムの特徴は，次のとおりです。

- クラスタ内の他の Alpha システムと共用できる， 1 つの Alpha システム・ディスクを持つ必要がある。
- ローカルに接続されたテープとディスクを使用できる。

- I64 システムと Alpha システムの両方に，ディスクとテープをサービスすることができる。
- I64 システムがサービスするデータ・ディスクを使用できる。
- SAN のディスクとテープを使用できる。
- 同じ SAN データ・ディスクを I64 システムと共用できる。

3.3 OpenVMS 用 EFI ユーティリティ

EFI ユーティリティは， OpenVMS I64 が動作する Integrity サーバ・システム向けのデバイス管理機能です。これらのユーティリティは EFI シェルでの対話形式の操作になります。各ユーティリティを起動するコマンドは， OpenVMS オペレーティング・システムをシャットダウンしてから， EFI Shell>プロンプトで\efi\vms から実行する必要があります。以下のコマンドがあります。

- VMS_BCFG: EFI Boot Manager にブート・エントリを追加します。これにより，エントリに OpenVMS デバイス名を指定できます。(この操作には，OpenVMS I64 Boot Manager (BOOT_OPTIONS.COM) ユーティリティの使用をお勧めします。)
- VMS_SET: 指定した OpenVMS デバイス名に，ダンプ・デバイスとデバッグ・デバイスを設定します。
- VMS_SHOW: EFI コンソールでマップされたデバイスに対する OpenVMS デバイス名を表示します。

詳細は，『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル』の EFI ユーティリティの章を参照してください。

3.4 HP Performance Data Collector (TDC)

OpenVMS Version 8.2 では， HP Performance Data Collector for OpenVMS (TDC V2.1) を使用することができます。 Performance Data Collector (TDC) を使用すると，約 1100 システムの性能メトリックを Alpha システムと I64 システムから収集することができます。このメトリックは，他のアプリケーション・ソフトウェアで解析することができます。

収集するメトリックは，以下のとおりです。

- キャッシュとメモリの使用率と性能
- クラスタ構成と通信
- CPU 使用率
- ディスクの使用率と性能
- 分散ロック・マネージャの性能

- Distributed Transaction Manager の性能
- ファイル・システムの性能
- ネットワーク・ハードウェアとソフトウェアの性能
- プロセス・メトリック
- その他のシステム性能のメトリック (たとえば，ページング，スワッピング，ページ・フォルト)
- システム・パラメータの設定値

OpenVMS Version 8.2 には， Performance Data Collector のランタイム・バージョン (TDC_RT V2.1) がインストールされます。ランタイム・バージョンは，データ収集アプリケーションとサポート・ファイルで構成されます。データ収集アプリケーションは自動的には起動されませんが，適切な特権を持ったユーザであれば手操作で起動や停止を行なうことができます。

ダウンロードできるキットには，サポートするすべてのシステム構成用の実行時環境の他に， SDK (Software Developer Kit) が含まれています。

プラットフォーム	OpenVMS のバージョン
Alpha システム	Version 7.3-2 または Version 8.2
I64 システム	Version 8.2

SDK には， TDC アプリケーション・プログラミング・インタフェース (API) と C ヘッダ・ファイル，およびサンプル・コードを説明したプログラマ・マニュアルが用意されています。 API を使うと，各種の方法で TDC が他のアプリケーションと統合された，次のようなソフトウェアを開発できます。

- TDC データ・ファイルから解析用のデータを抽出する
- TDC が収集したデータを，ファイルに格納することなく，リアルタイムに別のアプリケーションに渡す
- TDC が収集したメトリックを，他の解析対象のメトリックで，完全に統合されてサポートされた形態で補完する

SDK を使って作成したソフトウェアは， OpenVMS とともに配布されインストールされる TDC_RT キット，または完全な TDC キットをインストールした実行時環境で動作します。

ダウンロード可能な完全なキットと追加ドキュメントは，次の Web サイトから入手できます。

```
http://h71000.www.7hp.com/openvms/products/tdc/
```

3.5 Ethernet LAN ドライバ: 全二重モードと半二重モードの不一致

Ethernet LAN ドライバは，次のいずれかの条件で，全二重モードまたは半二重モードで動作します。

- Alpha システムでは，コンソール環境変数の設定に従う
- Alpha システムと I64 システムでは， LANCP デバイス・データベースの設定に従う
- オート・ネゴシエーションを有効にしている場合は，スイッチまたはリンク・パートナとのネゴシエーションで決まる

これらの条件のいずれかが満たされている場合に，二重モードを誤って設定すると，二重モードの不一致が発生します。

二重モードの不一致の例は，次のとおりです。

LAN デバイスを， 100 M ビット/秒の全二重モードで動作するように設定しています。しかし，スイッチ・ポートにはオート・ネゴシエーションが設定されています。スイッチ・ポートは，速度は正しく判定しますが，半二重モードを選択します。

3.5.1 二重モード不一致の結果

二重モードの不一致が発生すると，全二重モードのリンク終端では，送信データがあれば必ず送信します。受信パケットでリンクが占有されていないか確認することなく送信するので，その結果送信エラーと受信エラーが発生します。

パケットが失われるエラーに対しては，アプリケーションでは次のいずれかの対処が必要になります。

- エラーを検出してパケットを再送する，エラー回復を実行する。
- リンク・パートナにパケットの再送を依頼する。

その結果，アプリケーションによっては，性能が著しく低下します。したがって，このような状況を見つけて，修正しておくことが重要です。

3.5.2 二重モードの不一致の検出と修正

すべての全二重モード対応の LAN デバイス用の Ethernet LAN ドライバは，上記の状況を検出して報告するように変更されました。したがって，システム管理者は，この報告に基づいて修正することができます。各ドライバはエラー・カウンタを定期的にチェックします。二重モード不一致の状況が存在していると判断した場合には，ドライバは次のコンソール・メッセージを表示します。

```
%EWA0, Possible duplex mode mismatch condition detected
```

また，LAN ドライバは，エラー・ログ・エントリを作成します。このエントリのエラー・タイプは OxDD です (エラー・ログ・ビューアがエントリを英語に変換しない場合は，このエラー・タイプで検索できます)。LAN ドライバはリンクが停止したり，再び稼働したときもエラー・ログ・エントリを作成します。

LAN ドライバのエラー・ログは，次のエラー・タイプで検索して，エラー・ログ・エントリを解読することができます。

エラー・タイプ	説明
OxCA	接続が有効 (リンクは稼働中)
OxCD	接続が停止中 (リンクが停止中)
OxDD	疑わしい二重モード (おそらく二重モードの不一致)

各エラー・ログ・エントリは同じ形式になっており，エラー・タイプ・コードは同じ位置にあります。

エラー・ログ・エントリとコンソール・メッセージは，状況が改善しない限り，1 時間ごとに繰り返されます。

LANCP または ANALYZE/SYSTEM を使うと，デバイス・カウンタからさらに詳しい情報を取得できます。

3.6 ホスト・ベース・アダプタ (HBA) のサポート

以降の項では，OpenVMS Version 8.2 での次の HBA のサポートについて説明します。

- Fibre Channel ホスト・ベース・アダプタ
- Ultra SCSI ホスト・ベース・アダプタ

3.6.1 OpenVMS I64 システムと OpenVMS Alpha システムでの Fibre Channel HBA のサポート

OpenVMS I64 Version 8.2 では，Fibre Channel ストレージの外部接続用に，デュアル・ポート 2GB/1GB Fibre Channel Universal PCI-X HBA (A6826A) だけをサポートします。この HBA は，新しいドライバ PGQDRIVER でサポートされますが，これは既存のドライバ DKDRIVER へのポート・ドライバとして実装されています。Integrity サーバ用の OpenVMS I64 V8.1 評価用リリースでサポートしていた PCI-X 1 ポート FCA2404 2GB アダプタ (AB232A または KGPSA-EA，AlphaServers では LP9802) は，OpenVMS V8.2 以降ではサポートされません。したがって，ユーザは外部 Fibre Channel ストレージ用には，FC アダプタ AB232A または KGPSA-EA を A6826A アダプタで置き換える必要があります。

OpenVMS Alpha Version 8.2 では，新しい Fibre Channel HBA である LP10000 をサポートしています。 LP10000 は， AlphaServer の DS および ES サーバ・ファミリでサポートされています。 LP10000 には，シングル・チャネル・バージョンとデュアル・チャネル・バージョンがあります。

LP10000 は，パッチ・キットとコンソール・ファームウェア V6.6 によって，旧バージョンの OpenVMS Alpha (Version 7.3-1，および Version 7.3-2) でもサポートされます。

FIBRE_SCSI というルート名を持つ各バージョンのパッチ・キットは，次の Web サイトから入手できます。

```
http://h71000.www7.hp.com/serv_support.html
```

「Service tools」から「Patches for OpenVMS」，または「FTP site for OpenVMS patches」のいずれかを選択してください。

これらの HBA についての詳細は， HP Integrity サーバまたは使用するアダプタのハードウェア・マニュアルを参照してください。

3.6.2 OpenVMS I64 システムでの Ultra SCSI HBA のサポート

OpenVMS I64 V8.2 では，次に示す， HP Integrity サーバ・システムの Ultra SCSI HBA をサポートしています。

- Ultra-160 SCSI デュアル・チャネル (A6829A)—HP rx4640 Integrity サーバの場合
- Ultra-320 SCSI デュアル・チャネル (A7173A)— HP Integrity rx2600 サーバと HP Integrity rx1600 サーバに内蔵

外部 SCSI ストレージを使用するためには， DS2100， MSA30，または 4200/4300 シリーズの筐体に， Ultra-160 デュアル・ポート SCSI アダプタ (A6829A) を追加する必要があります。

注意

OpenVMS は， OpenVMS I64 システムだけで構成される OpenVMS Cluster システム，または OpenVMS I64 システムと OpenVMS Alpha システムが混在する OpenVMS Cluster システムでは，これらのアダプタを使った共用 SCSI ストレージをサポートしません。

弊社では，弊社が提供する HP-UX， Linux，および Microsoft XP 64 ビット・オペレーティング・システムなどのオペレーティング・システムでもこれらのアダプタをサポートします。

詳細は， HP Integrity サーバに付属するハードウェア・ドキュメントを参照してください。

3.7 System Analysis Toolsの機能拡張

System Analysis Tools には， Alpha システムと I64 システムの両方で使用できる，いくつかの新しいコマンドと新機能が追加されました。新しいコマンドと機能を使うと， OpenVMS システムを簡単に解析できるようになります。

詳細は，『OpenVMS System Analysis Tools Manual』を参照してください。

3.7.1 追加または機能拡張された SDA コマンド

I64 で使うために，追加または機能拡張されたコマンドは次のとおりです。

- EVALUATE
- EXAMINE
- FLT
- FORMAT
- READ
- SET CPU
- SET OUTPUT
- SHOW
  - CALL_FRAME
  - CBB
  - CEB
  - CPU
  - CRASH
  - DEVICE
  - EXCEPTION_FRAME
  - EXECUTIVE
  - GLOBAL_SECTION_TABLE
  - GST
  - IMAGE
  - KFE
  - PAGE_TABLE

  - PARAMETER
  - PROCESS
  - STACK
  - SWIS
  - TQEIDX
  - UNWIND
- VALIDATE TQEIDX
- WAIT

3.7.2 System Service Logging 機能

System Service Logging 機能によって，プロセス内のシステム・サービスのアクティビティが記録されます。これはシステムのトラブルシューティングに使用するために用意されています。

ロギング機能は， SET PROCESS/SSLOG=(STATE=ON) コマンドで有効にできます。ロギング機能は， SET PROCESS/SSLOG=(STATE=UNLOAD) または (STATE=OFF) コマンドで停止できます。ログに記録された情報は， ANALYZE /SSLOG コマンドを使って，表示できます。ログ情報が記録されるファイルのデフォルトは， SSLOG.DAT です。

詳細は，『OpenVMS System Analysis Tools Manual』を参照してください。

3.8 システム・パラメータ

以降の項で説明するシステム・パラメータは， Version 8.2 での新しいパラメータです。最後の項に， Version 8.2 で変更されたシステム・パラメータのリストを示します。詳細は，『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル』を参照してください。

3.8.1 追加されたシステム・パラメータ

次のシステム・パラメータが新しく OpenVMS Version 8.2 で追加されました。

- ERLBUFFERPAG_S2

  ERLBUFFERPAG_S2 は， ERRORLOGBUFF_S2 パラメータで要求された S2 空間エラー・ログ・バッファに割り当てる S2 空間メモリの量を指定します。

- ERRORLOGBUFF_S2

ERRORLOGBUFF_S2 は，システム・エラーのログ・エントリ用として予約された S2 空間エラー・ログ・バッファの数を指定します。各バッファは ERLBUFFERPAG_S2 で指定されるサイズを持ちます。 ERRORLOGBUFF_S2 のサイズが小さすぎる場合には，メッセージはエラー・ログ・ファイルに書き込まれません。 ERRORLOGBUFF_S2 のサイズが大きすぎる場合には，バッファによって不必要に多くの物理ページが専有される可能性があります。

- SCSI_ERROR_POLL

  SCSI_ERROR_POLL を指定すると， OpenVMS は各 SCSI ディスクに対して 1 時間ごとに SCSI Test Unit Ready コマンドを発行します。これは，ラッチされたエラーがあれば，それを解除して，即座に報告させるためです。 SCSI_ERROR_POLL にはデフォルトで 1 が設定されていますが，ユーザは， 0 を設定することで，エラー・ポーリング動作を停止させることができます。

- SHADOW_ENABLE

  弊社で使用するために予約されている特別なパラメータです。

- SHADOW_HBMM_RTC (Alpha と I64 のみ)

  SHADOW_HBMM_RTC は，システムがホスト・ベース・ミニマージ (HBMM) ビットマップを持つシャドウ・セットのリセットしきい値をチェックする時間間隔を制御します。リセットしきい値を超えている場合には，ビットマップがゼロ・クリアされます。

- SHADOW_PSM_DLY

  多くのシステムにマウントされているシャドウ・セットで，コピー操作やマージ操作が必要な場合，シャドウィング機能では，全シャドウ・セット・メンバへのローカル接続を持つシステム上でこの操作を実行しようとします。シャドウィングでは， MSCP でサービスされるシャドウ・セット・メンバの数に基づいて時間遅延を加えることで，コピー操作やマージ操作を実行します。ローカル・メンバに対しては遅延が加算されないため， 1 つ以上のメンバが MSCP でサービスされているシステム (したがって遅延が発生する) よりも，すべてのシャドウ・セット・メンバにローカルにアクセスできるシステムのほうがコピー操作やマージ操作に適しています。

  SHADOW_PSM_DLY パラメータを使用することで，システム管理者はシャドウィングによる遅延時間を調整することができます。 MSCP でサービスされる各シャドウ・セット・メンバの省略時の遅延時間は 30 秒で，遅延として指定できる範囲は 0 ～ 65,535 秒です。

  シャドウ・セットがマウントされているシステムが 1 つの場合は， SHADOW_PSM_DLY の値は，そのシャドウ・セットでの省略時のシャドウ・セット・メンバ回復遅延として使用されます。既存のシャドウ・セットでの SHADOW_PSM_DLY の変更方法は， SET SHADOW/RECOVERY_OPTIONS=DELAY_PER_SERVED_MEMBER=n コマンドを参照してください。

- SHADOW_REC_DLY (Alpha と I64 のみ)

  SHADOW_REC_DLY の値が RECNXINTERVAL パラメータの値に加算され，システムがシャドウ・セットの優先順位に従って，システムにマウントされているシャドウ・セットに対する回復操作の管理を始めるまでに待つ時間が決定されます。

- SHADOW_SITE_ID (Alpha と I64 のみ)

  SHADOW_SITE_ID を使用すると，システム管理者は， Volume Shadowing が，読み込み操作を実行するための最適のデバイスを選択し，性能を改善させるために使用する値のサイト値を定義できます。システム管理者は，システムにマウントされているすべてのシャドウ・セットに対するサイト値を定義できるようになりました。

- SYSSER_LOGGING (Alpha と I64 のみ)

  SYSSER_LOGGING に 1 を設定すると，プロセスのシステム・サービス要求のロギングが有効になります。デフォルトでは 1 が設定されています。このパラメータは動的に変更することができます。

- TTY_DEFCHAR3 (Alpha と I64 のみ)

  TTY_DEFCHAR3 を使用すると， OpenVMS のターミナル・ドライバが Ctrl/H を削除キーに対応付ける機能を有効にすることができます。この機能は，システム全体のデフォルトとして設定**しない**ようにしてください。

特性	値 (16 進数字)	機能
TT3$M_BS	10	このビットを設定すると，OpenVMS のターミナル・コンソールは CTRL/H を削除キーに対応付けます。

- VHPT_SIZE (I64 のみ)

  VHPT_SIZE には，システム内の各 CPU の仮想ハッシュ・ページ・テーブル (VHPT) 用に割り当てるメモリ・サイズを KB 単位で指定します。

  - 0 の場合は， VHPT は割り当てられません。
  - 1 の場合には， OpenVMS はシステム構成に合わせて適切なデフォルト・サイズを選択します。

3.8.2 変更されたシステム・パラメータ

OpenVMS Version 8.2 では，次のシステム・パラメータの定義が変更されました。

- BALSETCNT
- CHANNELCNT
- CRD_CONTROL
- DEVICE_NAMING

- ERLBUFFERPAGES
- ERRORLOGBUFFERS
- FAST_PATH_PORTS
- GALAXY
- GLX_SHM_REG
- LAN_FLAGS
- LCKMGR_MODE
- MMG_CTLFLAGS
- MULTITHREAD
- NISCS_MAX_PKTSZ
- PQL_MASTLM
- PQL_MENQLM
- SECURITY_POLICY
- SHADOW_MBR_TMO
- VCC_FLAGS

このリストで示したパラメータの他に，以前はAlpha **のみ**となっていた多くのパラメータが，今回からはAlpha **と** I64となりました。

詳細は，『HP OpenVMS Version 8.2 リリース・ノート[翻訳版]』と『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル』を参照してください。

3.9 データベースに追加されたタイムゾーン

OpenVMS Version 8.2 には，ftp://elsie.nci.nih.gov/pub/にあるタイムゾーン公用データベース tzdata2003e に基づいた 540 種類のタイムゾーンが用意されています。既存のタイムゾーンはアップデートされ，データベースには新たに 204 種類のタイムゾーンが追加されました。新しいタイムゾーン名の一覧については，『OpenVMS システム管理者マニュアル』を参照してください。

3.10 Volume Shadowing for OpenVMS の新機能

今回のリリースで Volume Shadowing for OpenVMS には，次の新機能が追加されました。

- ホスト・ベースのミニマージ (HBMM)

HBMM は，シャドウ・セットのマージ操作の効率を向上させるように設計されています。 HBMM では，ビットマップを使って，変更されたブロックをトレースします。 HBMM では，ビットマップがリセットされた後に変更されたシャドウ・セットのセクションだけを比較して一致させます。

HBMM のポリシー・キーワード RESET_THRESHOLD は，ビットマップのリセット条件となるまでに変更できるブロックの数を指定するために使います。新しいシステム・パラメータ SHADOW_HBMM_RTC は，変更ブロック数が RESET_THRESHOLD に設定された値を超えているかどうかを調べる頻度を指定するために使います。超えている場合には，ビットマップはリセットされます。 RESET_THRESHOLD の値を適切に設定すれば， HBMM の操作はフルマージよりもはるかに高速になります。

- マージ操作とコピー操作の優先順位付け

  システムで障害が発生した場合に，クラスタに残っているシステムで実行されるマージ操作とコピー操作の順序を制御できるようになりました。 SET SHADOW コマンドに新しい/PRIORITY=*n*修飾子を指定することで，マウントされているすべてのシャドウ・セットに異なる優先順位を割り当てることができます。最も重要なボリュームに高い優先順位を割り当てると，低い優先順位が割り当てられた重要度の低いボリュームよりも先にマージ操作とコピー操作が実行されます。

  SHADOW_MAX_COPY の設定値は，クラスタ内の状況の変化に応じて，動的に増減させることができます。シャドウ・セットをディスマウントすることなく，SET SHADOW コマンドの新しい EVALUATE_RESOURCES 修飾子を使用して，システムのワークロードを，新しい SHADOW_MAX_COPY の値，または SET SHADOW/PRIORITY=*n* DSA*n*コマンドで指定した新しい優先順位に適応させることができます。

  新しいシステム・パラメータ SHADOW_REC_DLY を使うと，システム障害が発生した後で各システムで実行されるマージ操作とコピー操作の順序を指定することができます。最適な回復操作を実行できるシステムに最小の遅延時間を割り当てることができます。

- Volume Shadowing for OpenVMS のライセンス方式

  OpenVMS I64 システムの場合は， CPU 単位となるキャパシティ・ライセンスの購入が可能です。 OpenVMS Alpha Version 8.2 の場合は，引き続き，キャパシティ・ライセンスとディスク単位のラインセンスが提供されます。

  OpenVMS I64 コンピュータ用のボリューム・シャドウィングのライセンスは，単独製品として，または EOE (Enterprise Operating Environment) と呼ばれる OpenVMS 製品群に含まれて提供されています。 EOE (Enterprise Operating Environment) の詳細は，『HP Operating Environments for OpenVMS Industry Standard 64 Version 8.2 for Integrity Servers Software Product Description (SPD 82.34.xx)』を参照してください。

ボリューム・シャドウィングのライセンス方式に関する詳細は，『HP Volume Shadowing for OpenVMS Software Product Description (SPD 27.29.xx)』を参照してください。ライセンス管理機能に関する詳細は， OpenVMS オペレーティング・システム SPD または『OpenVMS License Management Utility Manual』を参照してください。

HBMM の新機能についての詳細は，第 6 章を参照してください。

4
プログラミング機能

この章では， HP OpenVMS オペレーティング・システムのこのバージョンで追加されたアプリケーション・プログラミングおよびシステム・プログラミングに関連する新機能について説明します。

4.1 Analyze ユーティリティの機能拡張 —(I64 のみ)

OpenVMS I64 システムの Analyze ユーティリティは， ELF (Executable and Linkable Format) オブジェクト・ファイルおよびイメージ・ファイルが解析できるように機能拡張されました。この機能拡張についての説明は，『OpenVMS DCL ディクショナリ』の ANALYZE/IMAGE コマンドと ANALYZE/OBJECT コマンドに追加されています。

OpenVMS I64 システムの Analyze ユーティリティは，ファイル内のレコード形式とランドマーク値を使って，アーキテクチャ・タイプと，ファイルがオブジェクト・ファイルとイメージ・ファイルのどちらであるかを調べます。 ANALYZE では/OBJECT 修飾子と/IMAGE 修飾子が指定できますが，これらの修飾子を指定しても解析対象が，指定したファイル・タイプのファイルに限定されるわけではありません。

Analyze ユーティリティの機能拡張についての詳細は，『OpenVMS DCL ディクショナリ』を参照してください。

4.2 OpenVMS I64 での OpenVMS 呼び出し規則の変更点

OpenVMS 呼び出し規則が， OpenVMS I64 を実行している Intel Itanium プロセッサのシステムで使用できるように変更されました。

Intel Itanium プロセッサ・ファミリでの OpenVMS 呼び出し規則は， OpenVMS VAX や Alpha での規約とは，ユーザに見える違いはないようにする一方で，Itanium ソフトウェア規約にできる限り従うように設計されています。 Itanium の規約は，従来の OpenVMS の設計との互換性の維持が必要な場合に限って変更されました。目標はアプリケーションと OpenVMS 自身の Itanium アーキテクチャへの移植のコストと難しさを最小化することでした。

詳細は，『OpenVMS Calling Standard』を参照してください。

4.3 Checksum ユーティリティ

Checksum ユーティリティは， OpenVMS のファイルについて，ファイル，イメージまたはオブジェクトのチェックサムを計算します。このユーティリティは，CHECKSUM コマンドで起動されます。このユーティリティは， I64， Alpha，および VAX プラットフォームで動作するようになりました。計算結果のチェックサムは， DCL シンボル CHECKSUM$CHECKSUM に設定されます。

このユーティリティについての詳細は，『OpenVMS DCL ディクショナリ』の CHECKSUM コマンドを参照してください。

4.3.1 I64 オブジェクトに対する CHECKSUM/OBJECT の機能強化

I64 オブジェクト (ELF オブジェクト) に対する CHECKSUM/OBJECT では，チェックサムの計算に以下の情報も含めるようになりました。

- EIDC (entity identification consistency check) 情報
- FPMODE (whole programming floating-point mode) 情報

EIDC フィールドまたは FPMODE フィールドがオブジェクト・ファイル内に存在している場合，以前のバージョンの Checksum ユーティリティによる I64 オブジェクトのチェックサム計算結果と OpenVMS Version 8.2 の Checksum での計算結果が異なります。

違いは".note"セクションのチェックサムからわかります。

以前のバージョンでは，このセクションのチェックサムは計算されていませんでしたが，新しいバージョンでは，この情報が存在する場合にそのチェックサムが計算されるようになりました。この動作の結果を表示するには/SHOW=SECTIONS 修飾子を使用してください。

CHECKSUM/IMAGE には影響しません。イメージでは，".note"セクションに EIDC 情報や FPMODE 情報はありません。

ファイル・チェックサム (CHECKSUM に/IMAGE や/OBJECT の指定なし) にも影響はありません。ファイル・チェクサムでは，ファイル全体またはレコード構造に従ったデータを使用してチェックサムが計算されます。

4.4 C ランタイム・ライブラリの機能拡張

以降の項では， OpenVMS Version 8.2 に含まれる C ランタイム・ライブラリ (RTL) の機能拡張について説明します。この機能拡張により， UNIX への移植性，標準規格への準拠性，追加ユーザ制御機能の選択における柔軟性が改善されます。また，新し

い C RTL 関数が追加されました。詳細は，『OpenVMS HP C ランタイム・ライブラリ・リファレンス・マニュアル (上下巻)』を参照してください。

4.4.1 ファイル・ロック関数

以下の X/Open のファイル・ロック関数が追加されました。これらの関数を使用することで，ファイルのロックとアンロックを実行できます。また，スレッドを使ったプログラムの間で，アクセスの同期をとることができます。

```
flockfile
ftrylockfile
funlockfile
clearerr_unlocked
getc_unlocked
getchar_unlocked
feof_unlocked
ferror_unlocked
fgetc_unlocked
fputc_unlocked
putc_unlocked
putchar_unlocked
```

4.4.2 標準に準拠した stat 構造体

X/Open の標準に準拠したstat構造体の定義とそれに関連する定義が追加されました。これらの新しい定義を使うには，次の新しいマクロ定義を組み込んで，アプリケーションをコンパイルする必要があります。

_USE_STD_STAT

標準に準拠したstat構造体をサポートするために，次の新しいマクロも追加されています。

S_INO_NUM(ino)
S_INO_SEQ(ino)
S_INO_RVN(ino)
S_INO_RVN_RVN(ino)
S_INO_RVN_NMX(ino)

4.4.3 ファイル・システム統計情報のサポート

UNIX への移植性をサポートするために，次に示す，ファイル・システムの情報を返す X/Open の関数が追加されました。

```
statvfs
fstatvfs
```

4.4.4 fcntl ファイル・ステータス・フラグ

ファイル・ステータス・フラグを設定したり，取得するためのコマンド・オプション F_SETFL と F_GETFL が，fcntl関数に追加されました。

4.4.5 UNIX スタイルのパイプのサポート

UNIX への移植性を高めるために， C RTL のパイプの実装では，新しい機能論理名 DECC$STREAM_PIPE の制御のもとで，レコード入出力だけではなく，ストリーム入出力も使用するようになりました。

従来のレコード入出力を使用した動作がデフォルトです。

ストリーム入出力によるパイプのサポートを有効にするには，次のように，機能論理名 DECC$STREAM_PIPE に ENABLE を定義します。

```
$ DEFINE DECC$STREAM_PIPE ENABLE
```

4.4.6 DECC$POPEN_NO_CRLF_REC_ATTR

popen関数でオープンされたパイプのレコード属性には， CR/LF キャリッジ制御が設定されています (fab$b_rat |= FAB$M_CR)。 UNIX システムでは，パイプに CR/LF が挿入されません。

UNIX 互換の動作をサポートするために，新しい機能論理名が追加されました。デフォルトの動作を無効にして， CR/LF キャリッジ制御がパイプのレコードに追加されないようにするには， DECC$POPEN_NO_CRLF_REC_ATTR に ENABLE を定義します。

```
$ DEFINE DECC$POPEN_NO_CRLF_REC_ATTR ENABLE
```

ただし，この機能を有効にすると，キャリッジ・リターン文字を前提としている他の関数 (たとえば，gets) が望ましくない動作をする可能性があります。

4.4.7 glob と globfree での 64 ビット・サポート

glob関数とglobfree関数に， 64 ビット・サポートが追加されました。その結果，それぞれ 32 ビットのポインタと 64 ビットのポインタに対応する次の関数エントリ・ポイントが追加されました。

```
_glob32          _glob64
_globfree32      _globfree64
```

4.4.8 socketpair

TCP/IP ソケット・ルーチンのsocketpairが追加されました。

このルーチンは，接続状態のソケットのペアを作成します。 TCPIP$SOCKETPAIR 関数を使うためには，ベースとなる TCP/IP 製品が必要です。

4.5 DCE RPC での IEEE 浮動小数点型のサポート

DCE RPC for OpenVMS は， OpenVMS Alpha プラットフォームと OpenVMS I64 プラットフォームで， G_FLOAT と IEEE の両方の浮動小数点型をサポートするようになりました。 Alpha プラットフォームでのデフォルトの浮動小数点型は G_FLOAT のままです。 I64 プラットフォームでのデフォルトの浮動小数点型は IEEE_FLOAT です。

DCE RPC アプリケーションを開発するときにデフォルト以外の浮動小数点型を使用する場合には，アプリケーションの中で rpc_set_local_float_drep を呼び出す必要があります。

注意

OpenVMS VAX プラットフォームの DCE RPC は， G_FLOAT 型だけをサポートしています。

4.6 OpenVMS Debugger

以降の項では， OpenVMS I64 システムの OpenVMS Debugger の新機能について説明します。

4.6.1 Intel® Itanium®ハードウェアのサポート

OpenVMS I64 Debugger は，以下のハードウェア・レジスタをサポートしています。

- 汎用レジスタ R0 ～ R127
- 浮動小数点レジスタ F0 ～ F127
- 分岐レジスタ B0 ～ B7
- プレディケート・レジスタ P0 ～ P63。すべてのプレディケート・レジスタ値は，シンボル PR を使って調べることができます。
- アプリケーション・レジスタ: AR16 (RSC)， AR17 (BSP)， AR18 (BSPSTORE)， AR19 (RNAT)， AR25 (CSD)， AR26 (SSD)， AR32 (CCV)， AR36 (UNAT)， AR64 (PFS)， AR65 (LC)， AR66 (EC)

- ハードウェアの IP レジスタと PSR レジスタの ri フィールドを合成したプログラム・カウンタ PC。
- その他のレジスタ: CFM (カレント・フレーム・マーカー)， UM (ユーザ・マスク)， PSP (以前のスタック・ポインタ)，および IIPA (この前に実行されたバンドルのアドレス)
- 出力レジスタ OUT0 ～ OUT7。これらの名前は，整数型 (integer) の引数を現在のルーチンから呼び出し先ルーチンへ渡すのに使用するレジスタの識別を容易にするために用意されています。詳細は，『OpenVMS Calling Standard』を参照してください。

4.6.2 OpenVMS I64 の言語サポート

OpenVMS I64 Debugger は，以下の言語で記述されたプログラムをサポートします。

BASIC
BLISS
C
C++
COBOL
IMACRO
Fortran
Intel Assembler (IAS)
Pascal

4.6.3 ヒープ・アナライザが OpenVMS I64 システムで使用可能

ヒープ・アナライザが OpenVMS I64 システムで使用可能になりましたが，起動方法は Alpha システムの場合と異なります。 OpenVMS I64 では，ヒープ・アナライザはデバッガから起動します。起動するには，デバッガ・コマンド行プロンプト (DBG>) で新しい起動コマンド START HEAP_ANALYZER を入力します。詳細は，『OpenVMS デバッガ説明書』の「ヒープ・アナライザ」の章を参照してください。

4.7 拡張ロック値ブロック

ロック値ブロックは，同期型のプロセス間/クラスタ内通信をサポートするための，OpenVMS ロック・マネージャの機能です。これまでは，ロック・マネージャを使っている OpenVMS アプリケーションでは，ロック操作で最大 16 バイトのロック値ブロック・データの格納と回復を行なうことができました。 OpenVMS Version 8.2 では，アプリケーションは，オプションで，ロック値ブロックに最大 64 バイトのデータを格納できるようになりました。

$GETLKI システム・サービスが変更されて，この新しい機能に必要な 2 つの新しいアイテム・コードが追加されました。アイテム・コードの用途は，次のとおりです。

- LKI$_XVALBLK 拡張値ブロックを 64 バイトのバッファに取り出す。
- 最後の書き込みで， LKI$_XVALNOTVALID 短縮値ブロックが書き込まれたかどうかを，クォドワード・バッファ内に論理値で示す。

詳細は，『OpenVMS Programming Concepts Manual』と『OpenVMS System Services Reference Manual』を参照してください。

4.8 Librarian ユーティリティとライブラリ・ルーチン (I64 のみ)

以降の項では， OpenVMS I64 システムの Librarian ユーティリティとライブラリ・ルーチンについて説明します。 OpenVMS Alpha システムでは， Librarian ユーティリティとライブラリ・ルーチンの変更はありません。

注意

Librarian ユーティリティは，単に Librarian と呼ばれることもあります。また，ライブラリ・ルーチンは，ライブラリ・サービス， LBR ルーチン，または LBR$ルーチンと呼ばれることがあります。

4.8.1 Librarian の使用方法の概要

DCL コマンドの LIBRARY を使って Librarian ユーティリティを起動して (または，ライブラリ [LBR] ルーチンを使って)，次に示す種類のライブラリを作成できます。

- I64 (ELF) オブジェクト・ライブラリ
- I64 (ELF) 共有イメージ・ライブラリ
- マクロ・ライブラリ
- ヘルプ・ライブラリ
- テキスト・ライブラリ

LIBRARY コマンドを実行すると OpenVMS Librarian ユーティリティが起動できます。このユーティリティでは，ライブラリ・モジュールの保守を行なうことができます。または，単にライブラリやモジュールについての情報を表示するために使うこともできます。 I64 Librarian ユーティリティには， Alpha Librarian と同様の機能が用意されていますが，『HP OpenVMS Version 8.2 リリース・ノート [翻訳版]』に示す変更点と制限があります。

表 4–1 に，各 OpenVMS プラットフォームの Librarian ユーティリティで作成されるライブラリを示します。

表 4–1 OpenVMS プラットフォームで作成されるライブラリ

OpenVMS VAX	OpenVMS Alpha	OpenVMS I64
VAX オブジェクト	Alpha オブジェクト	I64 オブジェクト
VAX 共有イメージ	Alpha 共有イメージ	I64 共有イメージ
Alpha オブジェクト	VAX オブジェクト	
Alpha 共有イメージ	VAX 共有イメージ	
マクロ	マクロ	マクロ
テキスト	テキスト	テキスト
ヘルプ	ヘルプ	ヘルプ

4.8.2 Librarian ユーティリティの変更点

この項では， OpenVMS I64 システムの Librarian ユーティリティの変更点と制限について説明します。

4.8.2.1 Librarian のデフォルトが Intel® Itanium®アーキテクチャになった

OpenVMS I64 Librarian ユーティリティにはアーキテクチャを指定するスイッチがありません。次の修飾子を指定すると， Librarian の処理対象は， OpenVMS ELF オブジェクト・ライブラリまたはイメージ・ライブラリになります。

/OBJECT—OpenVMS ELF オブジェクト・ライブラリの処理 (デフォルト)
/SHARE—OpenVMS ELF 共有イメージ・ライブラリの処理

OBJECT 修飾子と SHARE 修飾子のどちらも指定しなかった場合にデフォルトで作成されるライブラリの種類は，オブジェクト・ライブラリです。

4.8.2.2 /ALPHA 修飾子と/VAX 修飾子はサポートされない

/ALPHA 修飾子と/VAX 修飾子は， I64 Librarian ユーティリティではサポートしていません。この Librarian ユーティリティは，次の種類のライブラリだけを処理します。

MACRO
TEXT
HELP
ELF OBJECT
ELF SHARABLE IMAGE

4.8.2.3 拡張された/REMOVE 修飾子

I64 Librarian では，/REMOVE 修飾子の機能が拡張されました。拡張された形式では，シンボルのインスタンスをどのモジュールから削除するかを指定できるようになりました。

LIBRARY コマンドに対して，オブジェクト・ライブラリ内のグローバル・シンボル・テーブルから 1 つまたは複数のエントリを削除するように要求します。

/REMOVE=(symbol[:module] [, ...])

symbol

グローバル・シンボル・テーブルから削除するシンボル。

module

グローバル・シンボル・テーブルからシンボルのインスタンスを削除するモジュール。 UNIX スタイルの弱いシンボルと ELF グループ・シンボルをサポートするようになったため，シンボルは複数のモジュールで定義されている可能性があります。この拡張構文を使用すると，他のモジュールのシンボルのインスタンスはインデックスから削除せずに，特定のモジュールのシンボルだけをインデックスから削除することができます。

説明

複数のシンボルを指定する場合には，シンボルはコンマで区切り，リストを括弧で囲みます。シンボルはワイルドカード文字を使って指定することもできます。削除したグローバル・シンボルを表示させたいときには，/LOG 修飾子も指定する必要があります。

4.8.3 ライブラリ (LBR) ルーチンの変更点

以降の項で， OpenVMS I64 システムのライブラリ・ルーチンの変更点と制限について説明します。ここにリストされていないすべてのルーチンは， Alpha システムの場合と同じです (説明は，『OpenVMS Utility Routines Manual』にあります)。

4.8.3.1 新しく追加されたライブラリ・タイプ

LBR$OPEN ルーチンに， 2 つの新しいライブラリ・タイプが追加されました。

LBR$C_TYP_ELFOBJ (9)—ELF オブジェクト・ライブラリであることを示す
LBR$C_TYP_ELFSHSTB (10)—ELF 共有イメージ・ライブラリであることを示す

さらに， LBR$OPEN ルーチンの次のライブラリ・タイプは， OpenVMS I64 ではサポートされません。

LBR$C_TYP_OBJ (1)—VAX オブジェクト・ライブラリであることを示す
LBR$C_TYP_SHSTB (5)—VAX 共有イメージ・ライブラリであることを示す
LBR$C_TYP_EOBJ (7)—Alpha オブジェクト・ライブラリであることを示す
LBR$C_TYP_ESHSTB (8)—Alpha 共有イメージ・ライブラリであることを示す

注意

これらのライブラリ・タイプを使用して， OpenVMS Alpha または VAX のオブジェクト・ライブラリや共有イメージ・ライブラリを作成したり，オープンすることはできません。

4.8.3.2 ELF オブジェクト・ライブラリへのアクセス

OpenVMS Alpha オブジェクト，テキスト・モジュールなどは，シーケンシャル・アクセス・モジュールですが， ELF オブジェクト・モジュールは，本質的に，ランダム・アクセス・モジュールです。ランダムにアクセスできるように， 1 つの新しいライブラリ・ルーチンが作成されました。このルーチンを使うと， ELF オブジェクト・モジュールがプロセスの P2 空間にマップされ，アプリケーションはランダム・アクセスのクエリを実行できるようになります。このマッピングから仮想アドレス空間を解放するために，このマッピングを削除するための別のライブラリ・ルーチンも作成されました。これらの新しいルーチン (LBR$MAP_MODULE と LBR$UNMAP_MODULE) は， ELF オブジェクト・ライブラリの処理にのみ使用できます。これらのルーチンは P2 空間を参照するため，エントリ・ポイントは 64 ビット・インタフェースです。

ELF オブジェクト・ファイルはランダムにアクセスされるものなので，次に示す操作は ELF オブジェクト・ライブラリに対しては実行できません。

LBR$GET_RECORD
LBR$SET_LOCATE
LBR$SET_MOVE

ライブラリにモジュールを挿入する操作はシーケンシャル操作なので， ELF オブジェクト・ライブラリに対して LBR$PUT_RECORD を実行することはできます。 ELF オブジェクト・モジュールはレコード単位にセグメント化されていないので，モジュールをライブラリに書き込む際に， LBR$PUT_RECORD を最初に呼び出すときに，ディスク上でのモジュールのサイズを指定する必要があります。

オブジェクト・モジュールを挿入するために LBR$PUT_RECORD を使用する方法を次の C コードの例に示します。

```
bufdesc->dsc$a_pointer = &p0_buffer ;
  bytes_to_transfer = module_size ;

  while ( bytes_to_transfer )  {
     transfer = MIN ( bytes_to_transfer ,
                      ELBR$C_MAXRECSIZ ) ;

     bufdesc->dsc$w_length = transfer ;

     status = lbr$put_record (  library_index ,
                              & bufdesc ,
                              & txtrfa ,
                                module_size ) ;
     if ( (status & 1) == 0 )
        break ;

     bytes_to_transfer      -= transfer ;
     bufdesc->dsc$a_pointer += transfer ;
     } ;

  if ( (status & 1) == 1 )
     status = lbr$put_end ( library_index ) ;
```

LBR$PUT_RECORD を何度も呼び出さなくてよいように，新しいライブラリ・ルーチン LBR$PUT_MODULE が作成されました (第 4.8.4 項を参照)。

4.8.4 ELF オブジェクト・ライブラリ用の新しいライブラリ (LBR) ルーチン

この項では， ELF オブジェクト・ライブラリ用の次の 4 つの新しいライブラリ・ルーチンについて説明します。

- LBR$LOOKUP_TYPE
- LBR$MAP_MODULE
- LBR$PUT_MODULE
- LBR$UNMAP_MODULE

LBR$LOOKUP_TYPE—インデックスでモジュール (RFA) のキーを検索

LBR$LOOKUP_TYPE ルーチンは，インデックスで特定のモジュール (RFA) のキーを検索し，そのモジュールのキーの型を返します。

フォーマット

LBR$LOOKUP_TYPE *library_index, key_name, txtrfa, ret_types*

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

key_name

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

引数key_nameは，次の引数特性を持つキーを指す文字列記述子のアドレスです。

引数特性	エントリ
OpenVMS 用法	char_string
データ型	文字列
アクセス	読み取り専用
受け渡し方	記述子渡し

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

モジュールのライブラリ・モジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数txtrfaは，モジュール・ヘッダの RFA を指定している 2 つのロングワードからなる配列のアドレスです。

ret_types

OpenVMS 用法: mask_longword
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 参照渡し

指定したモジュール (txtrfa) に対して見つかったシンボルの型を受け取るためのロングワードのアドレス。返される値のビットの意味は，次のとおりです。

LBR$M_SYM_NGG = 1
LBR$M_SYM_UXWK = 2
LBR$M_SYM_GG = 4
LBR$M_SYM_GUXWK = 8

説明

このルーチンは，インデックスで特定のモジュール (RFA) のキーを検索し，そのモジュールのキーがあれば，その型を返します。キーが見つからない場合には，LBR$_KEYNOTFND を返します。

LBR$MAP_MODULE—モジュールをマップする

LBR$MAP_MODULE ルーチンは，モジュールをプロセスの P2 空間にマップします。

フォーマット

LBR$MAP_MODULE *library_index, ret_va_addr, ret_mod_len, txtrfa*

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

ret_va_addr

OpenVMS 用法: address
データ型: クォドワード・アドレス
アクセス: 書き込み専用
受け渡し方: 32 ビットまたは 64 ビットの参照渡し

自然にアラインされたクォドワードの 32 ビットまたは 64 ビットの仮想アドレス。このルーチンは，ライブラリ・モジュールをマップした仮想アドレスをこのクォドワードに返します。

ret_mod_len

OpenVMS 用法: byte_count
データ型: クォドワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 32 ビットまたは 64 ビットの参照渡し

自然にアラインされたクォドワードのアドレス。ライブラリ・ルーチンはモジュールの長さをこのクォドワードに返します。

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

モジュールのライブラリ・モジュール・ヘッダのレコード・ファイル・アドレス(RFA)。引数txtrfaは，モジュール・ヘッダの RFA を指定している 2 つのロングワードからなる配列のアドレスです。

説明

このルーチンは，指定されたtxtrfaを使って，プロセスの P2 メモリ空間にモジュールをマップし，モジュールをマップした仮想アドレスとモジュール・サイズを返します。

他の LBR サービスが RMS サービスを使用するのとは異なり， LBR$MAP_MODULE はシステム・サービスも使用します。そのため，エラーで戻る場合の第 2 のステータスが LBR$$GL_SUBSTS に置かれます。エラーが返されたときは，これを使用してステータスの詳細を調べてください。

LBR$PUT_MODULE—モジュールをメモリから現在のライブラリへ置く

LBR$PUT_MODULE ルーチンは，モジュールのレコード・ファイル・アドレス (RFA) も含めてモジュール全体を，メモリ空間から現在のライブラリへ書き込みます。

フォーマット

LBR$PUT_MODULE *library_index, mod_addr, mod_len, txtrfa*

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・サービスから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

mod_addr

OpenVMS 用法: address
データ型: クォドワード・アドレス
アクセス: 読み取り専用
受け渡し方: 32 ビットまたは 64 ビットの参照渡し

メモリにマップされているモジュールの 64 ビット・アドレスを，ライブラリ・サービスが取得するアドレス。引数mod_addrは，自然にアラインされたクォドワードの 32 ビットまたは 64 ビットの仮想アドレスです。このクォドワードにはライブラリへ書き込むモジュールの仮想アドレスが格納されています。

mod_len

OpenVMS 用法: byte count
データ型: クォドワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 32 ビットまたは 64 ビットの参照渡し

ライブラリ・サービスによってライブラリへ書き込まれるモジュールの長さが格納された，自然にアラインされたクォドワードの 64 ビット仮想アドレス。

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)

アクセス: **書き込み専用**
受け渡し方: **参照渡し**

モジュールのライブラリ・モジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数txtrfaは，新たに作成されたモジュール・ヘッダの RFA を受け取る 2 つのロングワードからなる配列のアドレスです。

説明

LBR$PUT_MODULE ルーチンは，モジュールのレコード・ファイル・アドレス (RFA) も含めてモジュール全体を，メモリ空間から現在のライブラリへ書き込みます。モジュール全体を現在のライブラリに書き込む際に， LBR$PUT_END を使用する必要はありません。

LBR$UNMAP_MODULE—プロセスの P2 空間からモジュールをアンマップする

LBR$UNMAP_MODULE ルーチンは，プロセスの P2 空間からモジュールをアンマップします。

フォーマット

LBR$UNMAP_MODULE *library_index, txtrfa*

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

モジュールのライブラリ・モジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数txtrfaは，モジュール・ヘッダの RFA を指定している 2 つのロングワードからなる配列のアドレスです。

説明

txtrfa内のレコード・ファイル・アドレスを使って，プロセスの P2 メモリ空間からモジュールをアンマップします。これにより，モジュールをマップするために使われていたリソースは解放されます。

他の LBR サービスが RMS サービスを使用するのとは異なり， LBR$UNMAP_MODULE はシステム・サービスも使用します。そのため，エラーで戻る場合の第 2 のステータスが LBR$GL_SUBSTS に置かれます。エラーが返されたときは，これを使用してステータスの詳細を調べてください。

4.8.5 ELF オブジェクト・ライブラリ用の拡張ライブラリ (LBR) ルーチン

OpenVMS I64 システムでは，以降の項で説明するライブラリ・ルーチンが変更されています。 ELF グループ・シンボルと UNIX スタイルの弱いシンボルの定義の追加によって追加情報を渡す必要が生じたために，ライブラリ・ルーチンへ追加パラメータを渡すように機能が拡張されました。ライブラリ・ルーチンへの機能拡張は， ELF オブジェクト・ライブラリと ELF 共有イメージ・ライブラリにだけ影響します。

この項では，以下のライブラリ・ルーチンをリストします。

- LBR$DELETE_DATA
- LBR$DELETE_KEY
- LBR$GET_INDEX
- LBR$INSERT_KEY
- LBR$LOOKUP_KEY
- LBR$PUT_RECORD
- LBR$REPLACE_KEY
- LBR$SEARCH

このリストに入っていないライブラリ・ルーチンは， OpenVMS I64 システムと OpenVMS Alpha システムの間で，変更されていません。説明は，『OpenVMS Utility Routines Manual』にあります。

LBR$DELETE_DATA—モジュール・データの削除

LBR$DELETE_DATA ルーチンは，ライブラリからモジュール・データを削除します。

フォーマット

LBR$DELETE_DATA　*library_index, txtrfa, [flags]*

戻り値

OpenVMS 用法:　cond_value
データ型:　ロングワード (符号なし)
アクセス:　書き込み専用
受け渡し方:　値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法:　longword_unsigned
データ型:　ロングワード (符号なし)
アクセス:　読み取り専用
受け渡し方:　参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

txtrfa

OpenVMS 用法:　vector_longword_unsigned
データ型:　ロングワード (符号なし)
アクセス:　読み取り専用
受け渡し方:　参照渡し

削除したいモジュールのモジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数txtrfaは， RFA が格納された 2 つのロングワードからなる配列のアドレスです。モジュール・ヘッダの RFA は， LBR$LOOKUP_KEY または LBR$PUT_RECORD を呼び出すことで取得できます。

flags

OpenVMS 用法: mask_longword
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 値渡し

フラグの内容は無視されます。フラグを指定する目的は，アプリケーションが， ELF オブジェクト・ライブラリと ELF 共有イメージ・ライブラリの新しいインデックス構造体を認識していることを，このルーチンに知らせることです。

説明

ライブラリ・モジュールを削除する場合には，最初に LBR$DELETE_KEY を呼び出して，そのモジュールを指しているすべてのキーを削除する必要があります。どのライブラリのインデックス・キーもそのモジュール・ヘッダを指していない場合には，LBR$DELETE_DATA はモジュール・ヘッダと，関連付けられているデータ・レコードを削除します。それ以外の場合には，このルーチンは LBR$_STILLKEYS エラーを返します。他のライブラリ・ルーチンが，データを含まないデータ・ブロックを再利用する可能性があることに注意してください。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_INVRFA	指定した RFA は無効。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。
LBR$_STILLKEYS	インデックス内のキーが，モジュール・ヘッダを指したままである。この場合には，指定したモジュールは削除されません。

LBR$DELETE_KEY—現在のインデックスからのキーの削除

LBR$DELETE_KEY ルーチンは，現在のインデックスからキーを削除します。

フォーマット

LBR$DELETE_KEY *library_index, key_name [, txtrfa] [, flags]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

key_name

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

ライブラリ・インデックスから削除するキー。バイナリ・キーを持つライブラリの場合には，引数key_nameはキー番号が格納された符号なしロングワードのアドレスです。

ASCII キーを持つライブラリの場合には，引数key_nameは，次の引数特性を持つキーを指す文字列記述子のアドレスです。

引数特性	エントリ
OpenVMS 用法	char_string
データ型	文字列
アクセス	読み取り専用
受け渡し方	記述子渡し

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: **ロングワード (符号なし)**
アクセス: **読み取り専用**
受け渡し方: **参照渡し**

この引数を指定して，**なおかつ**引数flagsを指定しなかった場合は，このルーチンは，指定したtxtrfaのすべての型のキーを検索し，そのエントリを削除します。

flags

OpenVMS 用法: mask_longword
データ型: **ロングワード (符号なし)**
アクセス: **読み取り専用**
受け渡し方: **値渡し**

この引数を指定した場合には，特定の型のキー，またはすべての型のキーが削除されます。フラグ・ビットの意味は，次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 0x1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 0x2	グループ・シンボル属性
LBR$M_SYM_ALL = 0x80000000	すべてのシンボル

txtrfaを指定しない場合，または 0 を指定した場合には，flagsで示される型が削除されます。txtrfaに 0 以外の値を指定すると，txtrfaで指定した型のエントリが削除されます。 1 つの型，またはすべての型を指定することに注意してください。

説明

LBR$DELETE_KEY は，現在のインデックス内で，key_nameで指定されたキーを検索し，そのキーを削除します。ライブラリ・モジュールを削除する場合には，最初に LBR$DELETE_KEY を使って，そのモジュールを指しているすべてのキーを削除し，その後で LBR$DELETE_DATA を使って，モジュール・ヘッダと，関連付けられているデータを削除する必要があります。 LBR$SEARCH または LBR$GET_INDEX で指定したユーザ定義ルーチンの中から LBR$DELETE_KEY を呼び出すことはできません。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_KEYNOTFND	指定したキーが見つからない。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。
LBR$_UPDIRTRAV	LBR$SEARCH または LBR$GET_INDEX で指定したユーザ定義ルーチン内でインデックスをアップデートすることはできない。

LBR$GET_INDEX—指定したインデックス・キーを検索しルーチンを呼び出す

LBR$GET_INDEX ルーチンは，インデックス内で指定したキーを検索しユーザ定義ルーチンを呼び出します。

フォーマット

LBR$GET_INDEX *library_index,index_number, routine_name [, match_desc] [, flags]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

index_number

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

ライブラリ・インデックスの番号。引数index_numberは，インデックス番号が格納されたロングワードのアドレスです。これはユーザ定義ルーチンへの入力として使用するキーに関連付けられたインデックス番号です。

routine_name

OpenVMS 用法: procedure
データ型: **プロシージャ値**
アクセス: **読み取り専用**
受け渡し方: **参照渡し**

指定したインデックス・キーに対して呼び出されるユーザ定義ルーチン。引数routine_nameは，このユーザ定義ルーチンのプロシージャ値のアドレス。

LBR$GET_INDEX は，このルーチンに 3 つの引数を渡します。

- キーの名前。
  - ASCII キーを持つライブラリの場合は，引数key_nameは，そのキーを指す文字列記述子のアドレスです。ユーザ・ルーチンに渡される文字列と文字列記述子は，呼び出されている間だけ，有効であることに注意してください。その文字列をその後の処理でも使用する場合には，ローカルにコピーしておく必要があります。
  - バイナリ・キーを持つライブラリの場合は，引数key_nameはキー番号が格納された符号なしロングワードのアドレスです。
- このキーの名前に対応するモジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数 RFA は， RFA が格納された 2 つのロングワードからなる配列のアドレスです。
- キーの型。各フラグ・ビットの意味は次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 2	グループ・シンボル属性

ユーザ・ルーチンは，正常終了，または異常終了を示す戻り値を返す必要があります。ユーザ・ルーチンが偽の値 (最下位ビット= 0) を返した場合には， LBR$GET_INDEX はインデックスの検索を中止し，ユーザ定義ルーチンが生成したステータス値を，呼び出し元プログラムに返します。

ユーザ・ルーチンからは， LBR$DELETE_KEY や LBR$INSERT_KEY を呼び出すことはできません。

match_desc

OpenVMS 用法: char_string
データ型: **文字列**
アクセス: **読み取り専用**
受け渡し方: **記述子渡し**

キー照合識別子。引数match_descは，どのキーについてユーザ定義ルーチンを呼び出すかを指定するために使われる文字列を指す文字列記述子のアドレスです。この文字列ではワイルドカードを使うことができます。この引数を省略した場合には，

インデックス内のすべてのキーについて，ユーザ・ルーチンが呼び出されます。引数match_descは， ASCII キーを持つライブラリの場合にのみ有効です。

flags

OpenVMS 用法: mask_longword
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 値渡し

指定してあり，しかも 0 ではない場合には，指定したキーの型，またはすべての型を示します。フラグ・ビットの意味は，次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 0x1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 0x2	グループ・シンボル属性
LBR$M_SYM_ALL = 0x80000000	すべてのシンボル

ユーザ・ルーチンには，追加された 3 番目のパラメータによって，キーの型が渡されます。

説明

LBR$GET_INDEX は指定されたインデックス内で，引数match_descと一致するキーを検索します。一致するキーが見つかるたびに，引数routine_nameで指定されたユーザ・ルーチンを呼び出します。引数match_descを指定しなかった場合には，インデックス内のすべてのキーに対して，ユーザ・ルーチンを呼び出します。

たとえば，match_descを TR*とし，index_numberに 1 (モジュール名テーブル) を設定してオブジェクト・ライブラリに対して LBR$GET_INDEX を呼び出した場合には， LBR$GET_INDEX は，名前が TR で始まる各モジュールに対して，routine_nameを呼び出します。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_ILLIDXNUM	指定したインデックス番号は無効。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。
LBR$_NULIDX	指定したライブラリは空である。

LBR$INSERT_KEY—新しいキーの追加

LBR$INSERT_KEY ルーチンは，現在のライブラリのインデックスに新しいキーを追加します。

フォーマット

LBR$INSERT_KEY *library_index ,key_name ,txtrfa, [flags]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

key_name

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

追加する新しいキーの名前。ライブラリでバイナリ・キーが使われている場合には，引数key_nameは，キーの値が格納された符号なしロングワードのアドレスです。

ライブラリで ASCII キーが使われている場合には，引数key_nameは，次の引数特性を持つキーを指す文字列記述子のアドレスです。

引数特性	エントリ
OpenVMS 用法	char_string
データ型	文字列
アクセス	読み取り専用
受け渡し方	記述子渡し

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 変更
受け渡し方: 参照渡し

追加する新しいキーに関連付けられるモジュールのレコード・ファイル・アドレス (RFA)。引数txtrfaは， RFA が格納された 2 つのロングワードからなる配列のアドレスです。最初の LBR$PUT_RECORD の呼び出しで返される RFA を使うことができます。

flags

OpenVMS 用法: mask_longword
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 値渡し

指定した場合には，キーの型を示します。フラグ・ビットの意味は，次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 0x1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 0x2	グループ・シンボル属性

このパラメータを指定しない場合には，通常の NonGroup-Global 型を指定したものと見なされます。

説明

LBR$SEARCH や LBR$GET_INDEX で指定したユーザ定義ルーチンから LBR$INSERT_KEY を呼び出すことはできません。

戻される状態値

LBR$_DUPKEY	指定したキーはインデックスに既に含まれている。
LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_INVRFA	指定した RFA は有効なデータを指していない。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。
LBR$_UPDURTRAV	LBR$INSERT_KEY が，LBR$SEARCH または LBR$GET_INDEX で指定されたユーザ定義ルーチンから呼び出された。

LBR$LOOKUP_KEY—ライブラリ・キーの検索

LBR$LOOKUP_KEY ルーチンは，ライブラリの現在のインデックスからキーを検索し，そのキーに関連付けられたモジュール内のデータへのアクセスを準備します。

フォーマット

LBR$LOOKUP_KEY *library_index ,key_name ,txtrfa, [flags]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

key_name

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

ライブラリのキーの名前。ライブラリでバイナリ・キーが使われている場合には，引数key_nameは，キーの値が格納された符号なしロングワードのアドレスです。

ライブラリで ASCII キーが使われている場合には，引数key_nameは，次の引数特性を持つキーの文字列記述子のアドレスです。

引数特性	エントリ
OpenVMS 用法	char_string
データ型	文字列
アクセス	読み取り専用
受け渡し方	記述子渡し

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 参照渡し

ライブラリ・モジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数txtrfaは，モジュール・ヘッダの RFA を受け取る 2 つのロングワードからなる配列のアドレスです。

flags

OpenVMS 用法: mask_longword
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 参照渡し

この引数を指定し，しかも 0 でない場合には，返されるキーの型を指定します。フラグ・ビットの意味は，次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 0x1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 0x2	グループ・シンボル属性

返されるキーは，最高位の優先順位が定義された型を持つキーです。

説明

LBR$LOOKUP_KEY は，指定されたキーを検出すると，内部テーブルを初期化して，関連付けられたデータへアクセスできるようにします。

このルーチンは，txtrfaが指す 2 つのロングワードからなる配列に RFA を返します。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。

LBR$_INVRFA	取得された RFA が無効。
LBR$_KEYNOTFND	指定したキーが見つからない。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。

LBR$PUT_RECORD—データ・レコードの書き込み

LBR$PUT_RECORD ルーチンは，ライブラリ内の，空き領域が始まる位置にデータ・レコードを書き込みます。

フォーマット

LBR$PUT_RECORD *library_index ,bufdes ,txtrfa [,mod_size]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

bufdes

OpenVMS 用法: char_string
データ型: 文字列
アクセス: 読み取り専用
受け渡し方: 記述子渡し

ライブラリに書き込まれるレコード。引数bufdesは，レコードが格納されたバッファを指す文字列記述子のアドレスです。 VAX ライブラリの最大レコード・サイズは，シンボル LBR$C_MAXRECSIZ で定義されています。 Alpha と I64 のライブラリの場合は，最大レコード・サイズを示すシンボルは ELBR$_MAXRECSIZ です。

txtrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 参照渡し

モジュール・ヘッダのレコード・ファイル・アドレス (RFA)。引数txtrfaは，最初の LBR$PUT_RECORD の呼び出しで新しく作成されたモジュール・ヘッダの RFA を受け取る 2 つのロングワードからなる配列のアドレスです。

mod_size

OpenVMS 用法: byte count
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 値渡し

mod_sizeの値は，このルーチンを最初に呼び出したときにだけ読み取られ，それ以外では無視されます。この値には，書き込むモジュールのサイズを指定し，ライブラリ内の連続した領域がそのモジュールに割り当てられるようにします。この引数は ELF 以外のオブジェクト・ライブラリとデータ縮小型の ELF オブジェクト・ライブラリでは無視されます。バイト・ストリームを閉じて，モジュールを閉じるためには，従来どおりに， LBR$PUT_END ルーチンを呼び出す必要があります。

説明

LBR$PUT_RECORD を初めて呼び出した場合には，このルーチンは最初にモジュール・ヘッダを作成し，その RFA をtxtrfaが指す 2 つのロングワードからなる配列に返します。その後 LBR$PUT_RECORD は指定されたデータ・レコードをライブラリに書き込みます。続いて LBR$PUT_RECORD を呼び出した場合には，このルーチンは，ライブラリ内の，空き領域が始まる位置 (前のレコードの直後) にデータ・レコードを書き込みます。モジュールの最後のレコードを書き込んだ後で，LBR$PUT_END を呼び出す必要があります。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。

LBR$REPLACE_KEY—ライブラリ・キーの置換

LBR$REPLACE_KEY ルーチンは，ライブラリのキーの変更または追加を行います。

フォーマット

LBR$REPLACE_KEY *library_index ,key_name ,oldrfa ,newrfa [,flags]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

key_name

OpenVMS 用法: char_string
データ型: 文字列
アクセス: 読み取り専用
受け渡し方: 参照渡し

ライブラリで ASCII キーが使われている場合には，引数key_nameは，キーの文字列記述子のアドレスです。

ライブラリでバイナリ・キーが使われている場合には，引数key_nameは，キーの値が格納された符号なしロングワードのアドレスです。

oldrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

元のレコード・ファイル・アドレス (RFA)。引数oldrfaは，置換しようとしているキーに関連付けられたモジュール・ヘッダの元の RFA (LBR$LOOKUP_KEY で返される) が格納された 2 つのロングワードからなる配列のアドレスです。

newrfa

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

新しい RFA。引数newrfaは，新しいキーに関連付けられたモジュール・ヘッダの RFA (LBR$PUT_RECORD で返される) が格納された 2 つのロングワードからなる配列のアドレスです。

flags

OpenVMS 用法: mask_longword
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

指定した場合には，置換するキーの型を示します。フラグ・ビットの意味は，次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 0x1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 0x2	グループ・シンボル属性

このパラメータを指定しない場合には， NonGroup-Global を指定したものと見なされます。この場合には，すべての型のリストが検索され，対応するエントリが削除されます。新しいシンボルが，newrfaを定義モジュールとする，新しい NonGroup-Global 定義として格納されます。

このパラメータを指定した場合には，このパラメータは置換するシンボルの型のフラグを表わします。置換は元の位置で行われ，型リスト内での位置は変わりません。このルーチンを呼び出したときにシンボルが存在しなかった場合には，新しい定義は，指定した型の型リストの最後に付け加えられます。

これで異なるシンボル定義型が存在することになりますが，古いキーと新しいキーで定義型が異なる場合には， LBR$DELETE_KEY ルーチンを使った後で LBR$INSERT_KEY ルーチンを使うことをお勧めします。

説明

LBR$REPLACE_KEY は，現在のインデックス内にキーを検出できなかった場合には， LBR$INSERT_KEY ルーチンを呼び出して，キーを追加します。LBR$REPLACE_KEY がキーを検出した場合には，インデックスのキー・エントリが新しいモジュール・ヘッダを指すように変更します。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_INVRFA	指定した RFA は無効。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。

LBR$SEARCH—インデックスの検索

LBR$SEARCH ルーチンは，指定したデータを指すインデックス・キーを検索します。

フォーマット

LBR$SEARCH *library_index ,index_number ,rfa_to_find ,routine_name [, flags]*

戻り値

OpenVMS 用法: cond_value
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 値渡し

ロングワードの状態値。大部分のユーティリティ・ルーチンは，状態値を返します。このルーチンから戻される状態値は，「戻される状態値」の項に示します。

引数

library_index

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

LBR$INI_CONTROL ライブラリ・ルーチンから返されたライブラリ制御インデックス。引数library_indexは，インデックスが格納されたロングワードのアドレスです。

index_number

OpenVMS 用法: longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 読み取り専用
受け渡し方: 参照渡し

ライブラリのインデックス番号。引数index_numberは，検索対象のインデックス番号が格納されたロングワードのアドレスです。

rfa_to_find

OpenVMS 用法: vector_longword_unsigned
データ型: ロングワード (符号なし)
アクセス: 書き込み専用
受け渡し方: 参照渡し

キーを検索するモジュールのレコード・ファイル・アドレス (RFA)。引数rfa_to_findは，モジュール・ヘッダの RFA (前もって， LBR$LOOKUP_KEY または LBR$PUT_RECORD で返される) が格納された 2 つのロングワードからなる配列のアドレスです。

routine_name

OpenVMS 用法: procedure
データ型: **プロシージャ値**
アクセス: **読み取り専用**
受け渡し方: **参照渡し**

キーを処理するユーザ定義ルーチンの名前。引数routine_nameは， RFA を持つキー・エントリ (すなわち，同じモジュール・ヘッダを指すキー) が見つかるたびに呼び出すユーザ定義ルーチンのプロシージャ値のアドレスです。

このユーザ定義ルーチンでは， LBR$DELETE_KEY や LBR$INSERT_KEY を呼び出すことはできません。

flags

OpenVMS 用法: mask_longword
データ型: **ロングワード (符号なし)**
アクセス: **読み取り専用**
受け渡し方: **参照渡し**

この引数を指定していて， 0 でない場合には，指定したキーの型，またはすべての型を示します。フラグ・ビットの意味は，次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 0x1	UNIX スタイルの弱いシンボル属性
LBR$M_SYM_GROUP = 0x2	グループ・シンボル属性
LBR$M_SYM_ALL = 0x80000000	すべてのシンボル

ユーザ・ルーチンには，追加された 3 番目のパラメータで，シンボルの型が渡されます。

説明

ライブラリ・インデックスで，指定された RFA を持つシンボルを検索し，そのシンボルに対してユーザ定義ルーチンを呼び出します。

LBR$SEARCH を使うと，同じモジュール・ヘッダを指しているインデックス・キーが検索できます。通常，インデックス番号 1 (モジュール名テーブル) では， 1 つのキーだけが特定のモジュールを指しています。したがって，このルーチンは，複数のキーが 1 つのモジュールを指している場合にライブラリ・インデックスを検索するために使います。たとえば，オブジェクト・ライブラリ内の 1 つのオブジェクト・モジュ

ールに関連付けられたシンボルを，シンボル・インデックス内ですべて見つけるために， LBR$SEARCH を呼び出します。

LBR$SEARCH は，指定された RFA に関連付けられたインデックス・キーを検出した場合には，以下の 3 つの引数を渡してユーザ定義ルーチンを呼び出します。

- キー引数。次のいずれかのアドレスです。
  - キーの名前に対応した文字列記述子 (ASCII のキー名を持つライブラリ)
  - キーの値に対応した符号なしロングワード (バイナリ・キーを持つライブラリ)
- RFA 引数。モジュール・ヘッダの RFA が格納された 2 つのロングワードからなる配列のアドレスです。
- キーの型。各フラグ・ビットの意味は次のとおりです。

フラグ・ビット	意味
LBR$M_SYM_WEAK = 1	UNIX スタイルの弱いシンボルの属性
LBR$M_SYM_GROUP = 2	グループ・シンボルの属性

ユーザ・ルーチンは，正常終了，または異常終了を示す戻り値を返す必要があります。ユーザ・ルーチンが偽の値 (最下位ビット= 0) を返した場合には，インデックスの検索を中止します。

LBR$SEARCH が検出したキーは，ユーザ定義ルーチンを呼び出している間だけ有効です。このキーを後で使用する場合は，コピーしておく必要があります。

戻される状態値

LBR$_ILLCTL	指定したライブラリ制御インデックスは無効。
LBR$_ILLIDXNUM	指定したライブラリ・インデックス番号は無効。
LBR$_KEYNOTFND	ライブラリ・ルーチンは指定した RFA のキーを検出できなかった。
LBR$_LIBNOTOPN	指定したライブラリは，オープンされていない。

4.8.6 新しい UNIX スタイルの弱いシンボルの導入によるライブラリ形式の変更

Intel C++ コンパイラの要件に従うため，ライブラリの形式は，新しい UNIX スタイルの弱いシンボルに対応するように拡張されました。新しい UNIX スタイルの弱いシンボルのキー名が一致している複数のモジュールが，同じライブラリ内に存在できるようになりました。 Librarian ユーティリティは， OpenVMS スタイルの弱いシンボルの定義は，従来どおり，無視します。

UNIX スタイルの弱いシンボルの定義は， OpenVMS での弱い転送アドレスと同様の働きをします。すなわち，それらのシンボルの定義は一時的です。リンク操作時に，より強力なバインディング・タイプの定義が見つからなかった場合に，一時的な定義が最終的な定義になります。

4.8.6.1 弱いシンボルのための新しい ELF タイプ

2 種類の弱いシンボルの定義を区別するために，新しい ELF (Executable and Linkable Format) タイプが作られました。

バージョンが 2 以上の ABI を持つモジュールの場合に，次のタイプが追加されました。

- タイプ STB_WEAK は， UNIX スタイルの弱いシンボルを表わします (以前は，ABI バージョン 1 ELF 形式での OpenVMS スタイルの弱いシンボル定義でした)。
- タイプ STB_VMS_WEAK は， OpenVMS スタイルの弱いシンボル定義を表わします。

Librarian は，同じライブラリ内で， ELF ABI のバージョン 1 とバージョン 2 の両方のオブジェクト・ファイル形式とイメージ・ファイル形式をサポートしています。

注意

新しいライブラリ形式 (Version 6.0) は， ELF のオブジェクト・ライブラリと共有イメージ・ライブラリに対してだけ適用されます。他のライブラリは，Version 3.0 のままです。

4.8.6.2 Version 6.0 のライブラリ・インデックス形式

新しい Version 6.0 のライブラリを使うことをお勧めします。

ただし，新しいライブラリ・インデックス形式では， LBR ルーチンは， Version 3.0 と Version 4.0 の ELF のオブジェクト・ライブラリと共有イメージ・ライブラリを読み取り専用モードでオープンします。そのため，そのライブラリを LBR ルーチンや Librarian ユーティリティで変更することはできません。ただし，次の LIBRARY コマンドを使うと，旧バージョンのライブラリを Version 6.0 のライブラリに変換できます。

```
$  LIBRARY/COMPRESS library-name
```

Librarian ユーティリティを使って， Version 3.0， Version 4.0 と Version 5.0 の ELF のオブジェクト・ライブラリや共有イメージ・ライブラリを変更しようとすると， Librarian はこれらのライブラリを自動的に Version 6.0 のライブラリに変換します。

注意

ライブラリを少なくとも Version 6.0 レベルのライブラリ形式に変換すると，旧バージョンのライブラリ・ルーチンや Librarian ユーティリティではアクセスできなくなります。アクセスしようとすると， LBR$_UNSUPLVL ステータスが返され， Librarian ユーティリティは次のメッセージを表示します。

```
%LIBRAR-F-OPENIN, error opening library-name as input
 -LBR-E-UNSUPLVL, unsupported library format level
```

4.8.6.3 新しいグループ・セクションのシンボル

シンボルは， GROUP と呼ばれる ELF エンティティ内に含まれるセクションに関連付けることができます。これらのグループとそれに関連付けられるシンボルは，新しい UNIX スタイルの弱いシンボルの定義と同様の働きをします。すなわち，一時的な定義として扱われます。ライブラリではこれらのタイプのシンボルに対して，ライブラリのシンボル名インデックス内に複数のシンボル定義を許すようになりました。

4.8.6.4 優先順位規則

次に示すリストでは， GROUP 属性と UNIX スタイルの WEAK 属性が組み合わされたシンボルの型について，優先順位が高い型から低い型の順で説明します。これらのシンボルの型では，シンボルの結合の順番は，追加された時期に応じて，古いものから新しいものへの順番になります。この順番は重要です。 OpenVMS I64 Linker は，ライブラリからのシンボルの解決を求める際に，優先順位が最も高いシンボルで最も早く追加されたモジュールのインスタンスを受け取るからです。

- *NonGroup-Global* **シンボル**—ELF グループのメンバではなく， UNIX スタイルの弱い定義でもないシンボルです。これらのシンボルは，最も高い優先順位を持ちます。この型の定義は 1 つしか許されません。ライブラリ内にこの型のシンボルが既に存在していた場合には，新しいシンボル定義がエラーになります。
- *Group Global* **シンボル**—1 つのグループに関連付けられた ELF セクションのメンバで， UNIX スタイルの弱い定義ではないシンボルです。ライブラリには，この型の定義が複数あっても構いません。これらの定義は，追加した順に，リスト内に格納されます。リスト内のモジュール定義を置き換えた場合でも，リスト内の位置はそのままであることに注意してください。
- *UNIX-style weak* **シンボル**—UNIX スタイルの弱いバインディングを持ち，グループに属さないシンボルです。ライブラリには，この型の定義が複数あっても構いません。これらの定義は，追加した順に，リスト内に格納されます。リスト内のモジュール定義を置き換えた場合でも，リスト内の位置はそのままであることに注意してください。

- *Group weak* シンボル—Group Global 属性と UNIX スタイルの弱い属性が組み合わされたようにふるまうシンボルです。ライブラリには，この型の定義が複数あっても構いません。これらの定義は，追加した順に，リスト内に格納されます。リスト内のモジュール定義を置き換えた場合でも，リスト内の位置はそのままであることに注意してください。

4.9 Linker ユーティリティ

Linker ユーティリティの情報は第 7 章を参照してください。

4.10 HP OpenVMS Migration Software

バイナリ・トランスレータとも呼ばれる HP OpenVMS Migration Software for Alpha to Integrity (OMSAI) は， OpenVMS Alpha イメージを OpenVMS I64 で動作する機能的に同等なイメージに変換し， OpenVMS Alpha アプリケーションを OpenVMS I64 システムへ移行するのを容易にします。 OpenVMS I64 で，変換後のイメージを実行する際には，あたかも OpenVMS Alpha システム上で実行しているかのような環境が透過的にサポートされます。 OMSAI は， Alpha Environment Software Translator (AEST) ユーティリティと変換プロセスを容易にするよう設計されたプログラムやコマンド・ファイル群で構成されています。

この機能のリリース状況については，次の Web サイトで確認してください。

http://h71000.www7.hp.com/openvms/products/omsva/omsais.html

4.11 POSIX スレッド機能

以降の項では， POSIX スレッド・ライブラリに追加された新機能について説明します。

4.11.1 /NAMES=AS_IS でコンパイルするための小文字のシンボル名

POSIX スレッド・ライブラリ PTHREAD$RTL.EXE は，多くのシンボル名 (pthread API ルーチン，定義済みの例外オブジェクトなど) をエクスポートします。以前のリリースでは，これらのシンボルの英字はすべて大文字でした。 OpenVMS Version 8.2 では，既存の大文字シンボルのバージョンに加えて小文字シンボルのバージョンが提供されます。これらのシンボルにより，多くの HP コンパイラ製品に含まれている/NAMES=AS_IS 修飾子の使用が簡単になります。

4.11.2 プロセス共用型のミューテクスと条件変数のサポート

POSIX スレッド・ライブラリでは，プロセス共用型のミューテクスと条件変数をサポートするようになりました。これまでは，これらの機能は UNIX システムでのみサポートされていました。(プロセス共用型の読み書きロック機能は，今でも，UNIX システムでのみサポートされています。) OpenVMS Version 8.2 で，次に示す pthread ルーチンがサポートされるようになりました。

- pthread_condattr_getpshared
- pthread_condattr_setpshared
- pthread_mutexattr_getpshared
- pthread_mutexattr_setpshared

4.11.3 SET コマンドと SHOW コマンドの機能拡張 (I64 のみ)

OpenVMS I64 システムでは， DCL コマンドの SET と SHOW が機能拡張され，メイン・イメージのヘッダ・フラグ (UPCALLS と MKTHREADS の 2 つ) の照会や変更が可能になりました。さらに， Alpha システムと I64 システムの両方で， SET IMAGE コマンドと SHOW IMAGE コマンドを使って I64 イメージのみの設定や表示が可能になりました。

DCL コマンドの THREADCP は，引き続き OpenVMS Alpha システムで使用可能です。

4.11.4 Thread Independent Services API に追加された新しいルーチン

Thread Independent Services API (TIS) に，新しいルーチンtis_mutex_init_typeが追加されました。このルーチンは，既存のルーチンtis_mutex_initと似ています。

tis_mutex_init_type

指定されたミューテクス・オブジェクトを初期化し，呼び出し元が指定したミューテクスのタイプと名前を設定します。

フォーマット

tis_mutex_init_type *(mutex,type,name):*

引数	データ型	アクセス
mutex	opaque pthread_mutex_t	書き込み
type	integer	読み取り
name	char	読み取り

C バインディング

```
#include <tis.h>

int
tis_mutex_init_type (
    pthread_mutex_t    *mutex,
    int   type,
    const char    *name );
```

引数

mutex
初期化するミューテクス・オブジェクトへのポインタ (参照渡し)。

type
ミューテクスのタイプ属性の値。type引数には，作成されるミューテクスのタイプを指定します。有効な値は，次のとおりです。

- PTHREAD_MUTEX_NORMAL
- PTHREAD_MUTEX_DEFAULT
- PTHREAD_MUTEX_RECURSIVE
- PTHREAD_MUTEX_ERRORCHECK

name
ミューテクスに関連付けるテキスト名。

説明

このルーチンは，呼び出し元で指定するミューテクスのtypeとnameを除き，DECthreads のデフォルトのミューテクス属性で，ミューテクス・オブジェクトを初期化します。ミューテクスは，複数のスレッドが共用データへのアクセスをシリアル化するために使用する，同期用のオブジェクトです。

ミューテクス・オブジェクトは初期化されて，アンロックの状態になります。ミューテクスのタイプについては，『*Guide to POSIX Threads Library*』を参照してください。

ミューテクスのname引数は， C 言語の文字列であり， DECthreads マルチスレッド・アプリケーションのデバッグを行なう際に意味のある識別子です。 name 文字列は，文字列リテラル，またはミューテクスが存続している間存在する記憶域のいずれかである必要があります。文字列の内容は，ミューテクスの初期化では，コピーされません。

戻り値

エラーが発生すると，このルーチンはエラーの種類を示す整数値を返します。ミューテクスは初期化されず，**ミューテクス**の内容は不定です。戻り値は，以下のとおりです。

0	正常終了。
[EAGAIN]	ミューテクスを初期化するために必要なシステムのリソースが不足している。
[EBUSY]	すでに初期化されていて，破棄されていない**ミューテクス**を再度初期化しようとした。
[EINVAL]	mutexで指定した値が正しいミューテクスではない，またはtypeで指定したタイプが正しいミューテクス・タイプではない。
[ENOMEM]	ミューテクスを初期化するためのメモリが不足している。
[EPERM]	この操作を行うための権限が呼び出し元にない。

関連ルーチン

```
tis_mutex_destroy()
tis_mutex_init()
tis_mutex_lock()
tis_mutex_trylock()
tis_mutex_unlock()
```

4.12 新しい RTL LIB ルーチン

表 4–2 に， OpenVMS 8.2 Version 8.2 で用意された新しいルーチンの一覧を示します。これらのルーチンは，特に断らない限り OpenVMS Alpha と OpenVMS I64 の両方のシステムで使用できます。

表 4–2 RTL LIB ルーチン

ルーチン名	説明
LIB$CVTS_FROM_INTERNAL_TIME	内部時刻を外部時刻に変換 (S-floating 値)
LIB$CVTS_TO_INTERNAL_TIME	外部時刻を内部時刻に変換 (S-floating 値)
LIB$EMODS	拡張乗算を実行し， S-floating 値を整数化
LIB$EMODT	拡張乗算を実行し， T-floating 値を整数化
LIB$I64_CREATE_INVO_CONTEXT	呼び出しコンテキスト・ブロックの割り当てと初期化 (I64 のみ)
LIB$I64_FREE_INVO_CONTEXT	呼び出しコンテキスト・ブロックの割り当て解除 (I64 のみ)
LIB$I64_GET_INVO_CONTEXT	任意のアクティブ・プロシージャの呼び出しコンテキストを取得 (I64 のみ)
LIB$I64_GET_CURR_INVO_CONTEXT	任意のアクティブ・プロシージャの現在の呼び出しコンテキストを取得 (I64 のみ)
LIB$I64_GET_CURR_INVO_HANDLE	現在の呼び出しハンドルを取得 (I64 のみ)
LIB$I64_GET_FR	浮動小数点レジスタの値を取得 (I64 のみ)
LIB$I64_GET_GR	汎用レジスタの値を取得 (I64 のみ)
LIB$I64_GET_INVO_HANDLE	呼び出しハンドルを取得 (I64 のみ)
LIB$I64_GET_PREV_INVO_CONTEXT	直前の呼び出しコンテキストを取得 (I64 のみ)
LIB$I64_GET_PREV_INVO_HANDLE	直前の呼び出しハンドルを取得 (I64 のみ)
LIB$I64_GET_UNWIND_HANDLER_FV	指定された pc_value に対して，条件ハンドラの関数値 (プロシージャ記述子のアドレス) を探し，見つかった場合は，それを handler_fv に書き込む (I64 のみ)
LIB$I64_INIT_INVO_CONTEXT	割り当て済みの呼び出しコンテキスト・ブロックを初期化 (I64 のみ)
LIB$GET_UIB_INFO	unwind 情報ブロックの情報を返す
LIB$I64_GET_GR	汎用レジスタの値を取得 (I64 のみ)
LIB$I64_GET_UNWIND_LSDA	unwind 情報ブロックの言語固有のデータのアドレスを見つける (I64 のみ)
LIB$I64_GET_UNWIND_OSSD	unwind 情報ブロックのオペレーティング・システム固有領域のアドレスを見つける (I64 のみ)
LIB$I64_IS_AST_DISPATCH_FRAME	指定された PC 値が AST かどうかを判定する (I64 のみ)
LIB$I64_IS_EXC_DISPATCH_FRAME	指定された PC 値が例外ディスパッチ・フレームかどうかを判定する (I64 のみ)
LIB$I64_PREV_INVO_END	unwind 記述子を処理するために使用したメモリを解放 (I64 のみ)

(次ページに続く)

表 4–2 (続き) RTL LIB ルーチン

ルーチン名	説明
LIB$I64_PUT_INVO_REGISTERS	呼び出しレジスタを設定 (I64 のみ)
LIB$I64_SET_FR	呼び出しコンテキスト・ブロックのコンテキストを書き込む (I64 のみ)
LIB$I64_SET_GR	呼び出しブロックの汎用レジスタの値を書き込む (I64 のみ)
LIB$I64_SET_PC	呼び出しコンテキスト・ブロックの pc_copy 値を書き込む (I64 のみ)
LIB$LOCK_IMAGE	イメージをプロセス・ワーキング・セット内にロックする
LIB$MULTS_DELTA_TIME	デルタ時間に S-floating のスカラー値を掛ける
LIB$POLYS	多項式を評価する (S-floating 値)
LIB$POLYT	多項式を評価する (T-floating 値)
LIB$UNLOCK_IMAGE	プロセス・ワーキング・セットからイメージをアンロックする

詳細は，『OpenVMS RTL Library (LIB$) Manual』を参照してください。

4.12.1 LIB$GETDVI ルーチンの変更 (I64 のみ)

OpenVMS I64 システム用に， RTL LIB ルーチン LIB$GETDVI に新しい引数*pathname*が追加されました。詳細は，『OpenVMS RTL Library (LIB$) Manual』を参照してください。

4.13 新しい RTL OTS ルーチン

表 4–3 に，新しい OTS ルーチンの一覧を示します。これらのルーチンは，特に断らない限り OpenVMS Alpha と OpenVMS I64 の両方のシステムで使用できます。

表 4–3 RTL OTS ルーチン

ルーチン名	説明
OTS$CNVOUT_S	S-floating 値を文字列に変換
OTS$CNVOUT_T	T-floating 値を文字列に変換
OTS$CVT_T_S	数値テキストを S-floating 値に変換
OTS$CVT_T_T	数値テキストを T-floating 値に変換
OTS$DIVCS_R3	複素数の除算の結果の S-floating 複素数を返す
OTS$DIVCT_R3	複素数の除算の結果の T-floating 複素数を返す

(次ページに続く)

表 4–3 (続き) RTL OTS ルーチン

ルーチン名	説明
OTS$MULCT_R3	2 つの複素数値から複素数の積を計算し， T-floating 複素数を返す
OTS$POWCSCS_R3	複素数の基底を， S-floating 複素数の指数に上げる
OTS$POWCSCT_R3	複素数の基底を， T-floating 複素数の指数に上げる
OTS$POWCSJ	S-floating 複素数の基底を整数の指数に上げた結果の複素数を返す
OTS$POWCTJ	T-floating 複素数の基底を整数の指数に上げた結果の複素数を返す
OTS$POWSJ	S-floating の基底を，ロングワードの指数に上げる
OTS$POWSLU	S-floating の基底を，符号なしロングワードに上げる
OTS$POWSS	S-floating の基底を， S-floating またはロングワード整数の指数に上げる
OTS$POWTLU	T-floating の基底を，符号なしロングワードに上げる
OTS$POWTJ	T-floating の基底を，ロングワード整数の指数に上げる
OTS$POWTT	T-floating の基底を， T-floating またはロングワード整数の指数に上げる

詳細は，『OpenVMS RTL General Purpose (OTS$) Manual』を参照してください。

4.14 パッチ・ユーティリティが OpenVMS Alpha と OpenVMS I64 で使用可能

これまでは OpenVMS VAX システムでのみ使用可能だったパッチ・ユーティリティが OpenVMS Alpha および OpenVMS I64 システムでも使用できるようになりました。デフォルトで PATCH/ABSOLUTE が含まれます。 PATCH/ABSOLUTE は絶対仮想アドレスでファイルにパッチをあてます。 PATCH/ABSOLUTE と関連パラメータの詳細は，『OpenVMS DCL ディクショナリ』を参照してください。パッチ・ユーティリティの説明は，『OpenVMS VAX Patch Utility Manual』にも記載されています。このドキュメントは， OpenVMS Documentation Web サイトの[Archived documents]またはパッチ・ユーティリティに付属のオンライン・ヘルプにあります。

4.15 新しいシステム・サービスと改訂されたシステム・サービス

表 4–4 に， OpenVMS Version 8.2 で新たに追加されたシステム・サービスの概要を示します。

表 4–4 新しいシステム・サービス

システム・サービス名	説明
SYS$CLEAR_UNWIND_TABLE	アンワインド・テーブル (UT) 情報をクリアする
SYS$GET_UNWIND_ENTRY_INFO	I64 システムで，修正されたアンワインド・エントリ情報を取得する
SYS$GOTO_UNWIND_64	Alpha システムと I64 システムで，コール・スタックをアンワインドする
SYS$IEEE_SET_PRECISION_MODE	I64 システムで， IEEE 精度モードを変更し，オプションで，以前の値を返す
SYS$IEEE_SET_ROUNDING_MODE	I64 システムで， IEEE 丸めモードを変更し，オプションで，以前の値を返す
SYS$RPCC_64	Alpha システムと I64 システムで， 64 ビット，プロセス・ベースの高精度時刻カウンタを返す
SYS$SET_RETURN_VALUE	Alpha システムと I64 システムで，アーキテクチャとは独立に， Mechanism Array に戻り値または条件コードを設定する
SYS$SET_UNWIND_TABLE	I64 システムで，アンワインド・テーブル (UT) 情報を登録する，または拡張する

詳細は，『OpenVMS System Services Reference Manual』を参照してください。UNWIND システム・サービス・ルーチンについての詳細は，『OpenVMS Calling Standard』を参照してください。

表 4–5 に， OpenVMS Version 8.2 で修正されたシステム・サービスの概要を示します。

表 4–5 修正されたシステム・サービス

システム・サービス名	説明
SYS$CHECK_FEN	I64 システムで，ビットマスクに 2 つのビットが用意された (下位浮動小数点バンクに対応するビット 0 と，上位浮動小数点バンクに対応するビット 1)
SYS$CREATE_GPFN	SEC$M_UNCACHED フラグは I64 システムの場合にのみ有効
SYS$CRELNT	LNM$M_NO_ALIAS はクラスタ単位の論理名テーブルでは使用不可
SYS$CREMBX	prmflg の値を変更
SYS$CRMPSC_GDZRO_64	SS$_INSF_SHM_REG の状態値を追加
SYS$CRMPSC_GPFN_64	Alpha システムと I64 システムの新しい動作に対応して修正
SYS$CRMPSC_PFN_64	SEC$M_UNCACHED フラグは I64 システムの場合にのみ有効
SYS$DEQ	拡張ロック値ブロック情報を追加
SYS$ENQ	拡張ロック値ブロック情報を追加

(次ページに続く)

表 4–5 (続き) 修正されたシステム・サービス

システム・サービス名	説明
SYS$GETLKI	拡張ロック値ブロック情報を追加
SYS$GETDVI	以下に示す新しい項目コードを追加 ACCESSTIMES_RECORDED AVAILABLE_PATH_COUNT ERASE_ON_DELETE ERROR_RESET_TIME HARDLINKS_SUPPORTED MOUNT_TIME MOUNTVER_ELIGIBLE MPDEV_AUTO_PATH_SW_CNT MPDEV_MAN_PATH_SW_CNT MVSUPMSG NOCACHE_ON_VOLUME NOHIGHWATER NOSHARE_MOUNTED ODS2_SUBSET0 ODS5 PATH_AVAILABLE PATH_NOT_RESPONDING PATH_POLL_ENABLED PATH_SWITCH_FROM_TIME PATH_SWITCH_TO_TIME PATH_USER_DISABLED PROT_SUBSYSTEM_ENABLED SCSI_DEVICE_FIRMWARE_REV TOTAL_PATH_COUNT VOLUME_EXTEND_QUANTITY VOLUME_MOUNT_GROUP VOLUME_MOUNT_SYS WRITETHRU_CACHE_ENABLED
SYS$GETRMI	多くのバッファ長フィールドを 8 バイトに変更
SYS$IEEE_SET_FP_CONTROL	Alpha システムと I64 システムの新しい動作に対応して修正
SYS$INIT_VOL	新しいアイテム・コード INIT$_ERASE_ON_DELETE, INIT$_ERASE_ON_INIT, INIT$_VOLUME_LIMIT を追加
SYS$LKWSET	Alpha システムと I64 システムの新しい動作に対応して修正
SYS$LKWSET_64	Alpha システムと I64 システムの新しい動作に対応して修正
SYS$MGBLSC_GPFN_64	SEC$M_UNCACHED フラグは I64 システムでは無視される
SYS$MOUNT	アイテム・コードMNT_DENSITY を修正
SYS$SETFLT	FLT$M_EXECUTABLE の軽微な変更
SYS$SETFLT_64	FLT$M_EXECUTABLE の軽微な変更
SYS$SETRWN	追加されたシステム・リソースとプロセス制限は，リソースの待ち状態モードの影響を受ける

(次ページに続く)

表 4–5 (続き) 修正されたシステム・サービス

システム・サービス名	説明
SYS$SET_DEVICE	SDV$_MP_SWITCH_PATH に対して返される新しい状態値を追加
SYS$ULWSET	Alpha システムと I64 システムの新しい動作に対応して修正
SYS$ULWSET_64	Alpha システムと I64 システムの新しい動作に対応して修正

詳細は，『OpenVMS System Services Reference Manual』の説明を参照してください。

4.16 Time Zone Information Compiler (zic) のアップデート

Rule 行の AT フィールドに新しい時刻インディケータが追加されました。英字 "u" (または "g"または "z") は， AT フィールドの時刻が UTC であることを示します。 zic の詳細は，以下のとおりです。

AT　規則が適用される時刻を指定します。
指定できる形式は，次のとおりです。

2	時間で指定する時刻
2:00	時間と分で指定する時刻
15:00	24 時間制の時刻（正午以降の時刻用）
1:28:14	時間，分，秒で指定する時刻
マイナス記号(-)	0 と同値

ただし，時刻 0 は，一日の始まりを意味し，時刻 24 は一日の終わりを意味します。

これらの時刻表現には，次のサフィックスを付けることができます。

- w: 時刻はローカルな「壁時計」時刻
- s: 時刻はローカルな「標準」時刻
- u (または g または z): 時刻はグリニッジ標準時

サフィックスがない場合には「壁時計」時刻と見なされます。

4.17 Traceback 機能

I64 システムの Traceback 機能に，新しいシンボル化ルーチン TBK$I64_SYMBOLIZE が加わりました。このルーチンは，いつでも呼び出し可能で，例外によって起動された条件をアプリケーションで処理できるようにします。

以下に，このルーチンの詳細を説明します

TBK$I64_SYMBOLIZE

Traceback シンボル化ルーチン TBK$I64_SYMBOLIZE は，アプリケーション・プログラムで例外条件を処理できるようにします。

呼び出し方法

```
#pragma pointer_size save
#pragma pointer_size 64
int32 tbk$i64_symbolize(
   uint64 const pc,
   struct dsc64$descriptor * const filename_desc,
   struct dsc64$descriptor * const library_module_desc,
   uint64 * const record_number,
   struct dsc64$descriptor * const image_desc,
   struct dsc64$descriptor * const module_desc,
   struct dsc64$descriptor * const routine_desc,
   uint64 * const listing_lineno,
   uint64 * const rel_pc);
#pragma pointer_size restore
```

入力

pc	プロセス空間で「トレースバック対象」とする実行可能な命令。値渡し。

出力

filename_desc	文字列ディスクリプタ。コードを含むファイルの名前が返される。参照渡し。
library_module_desc	文字列ディスクリプタ。コードを含むテキスト・ライブラリ・ファイルの名前が返される。参照渡し。 該当する場合のみ。
record_number	64 ビット符号なし整数型。レコード番号 (レコード番号 n は filename_desc で特定されたファイルの n 行目を示します) が返される。参照渡し。
image_desc	文字列ディスクリプタ。イメージ名が返される。参照渡し。
module_desc	文字列ディスクリプタ。モジュール名が返される。参照渡し。
routine_desc	文字列ディスクリプタ。ルーチン名が返される。参照渡し。
listing_lineno	64 ビット符号なし整数型。コンパイラ・リスティングの行番号が書き込まれる。参照渡し。
rel_pc	64 ビット符号なし整数型。相対 PC 値が書き込まれる。参照渡し。

戻り値

TBK$_NORMAL は，正常終了時に返されます。エラーが発生すると，他の異常終了コードが返されます。

説明

I64 システムの Traceback 機能に，新しいシンボル化ルーチン TBK$I64_SYMBOLIZE が加わりました。このルーチンは，いつでも呼び出し可能で，例外によって起動された条件をアプリケーションが別の方法で処理できるようにします。通常のトレースバック処理では「トレースバック・スタック」情報を生成し，SYS$OUTPUT に出力します (一連の PC 値，スタック・レベルごとに 1 つ)。アプリ

ケーションで独立してトレースバック情報を処理したい場合に，トレースバック処理を直接呼び出すことができます。

アプリケーションでは，イメージ・アドレス空間内の任意の実行可能 PC 値を指定でき，上記の「出力」の項に記載された戻り引数を受け取ることができます。戻り情報には，指定した PC 値を含むオブジェクト・コードを生成したソース・コード行のレコード番号やリスティング行の場所が記述されます。戻り情報には，イメージ名，モジュール名，ルーチン名も記述されます。イメージ相対値またはモジュール相対値のいずれかを指定する，相対 PC 値も返されます。

要求された値を Traceback が返せるかどうかは，イメージ生成のコンパイル段階とリンク段階にトレースバック情報を要求したかどうかによります。

注意

いずれの出力引数 (値) も，ゼロ (0) を指定すると，引数が無視されます。

TBK$I64_SYMBOLIZE を呼び出すアプリケーションをリンクするには，Linker オプション・ファイルに以下の記述を含めてください。

```
SYS$SHARE:TRACE.EXE/shareable
```

4.18 XDELTA の新機能

以下に， OpenVMS Alpha システムと I64 システムで動作する OpenVMS XDELTA Debugger の新機能の要約をリストします。

- 新しいコマンド: ;D と;T
- 記号の使い方の改善
- XDELTA は，以下に示す Itanium®レジスタへのアクセスをサポートします。
  - 汎用レジスタ: R0 ～ R127
  - 浮動小数点レジスタ: FP0 ～ FP127
  - アプリケーション・レジスタ: AR0 ～ AR127
  - 分岐レジスタ: BR0 ～ BR7
  - 制御レジスタ: CR0 ～ CR63
  - プレディケート・レジスタ: P0 ～ P63
  - その他のレジスタ: PC， PS， CFM
  - Alpha のハードウェア・レジスタのソフトウェアによる実装

詳細は，『OpenVMS Delta/XDelta Debugger Manual』を参照してください。

5
関連製品の新機能

この章では， OpenVMS オペレーティング・システムの関連製品の重要な新機能について説明します。 OpenVMS 関連製品の一覧とディレクトリ情報については，『HP OpenVMS Version 8.2 CD-ROM/DVD ユーザーズ・ガイド』を参照してください。

5.1 ATI RADEON 7500 グラフィック

OpenVMS Version 8.2 では， Integrity サーバでの 2D マルチヘッドおよび 3D グラフィックのサポートは，オペレーティング・システムのリリース後に提供する予定です。 2005 年前半のサポートを予定しており，次の OpenVMS Web サイトにてお知らせします。

```
http://h71000.www7.hp.com/new/index.html
```

5.2 Common Data Security Architecture (CDSA) での新しい暗号タイプのサポート

CDSA では， CDSA Version 2.1 (OpenVMS Version 8.2) で新しい暗号タイプをサポートするようになりました。 Advanced Encryption Standard (AES) が， CDSA SSLeay Cryptographic Service Provider でサポートする標準暗号タイプの 1 つとして有効になりました。

以下に， CDSA を使用する AES 暗号/復号プログラムの簡単な例を示し， OpenVMS 上で CDSA をビルドするのに必要なファイルについて説明します。プログラムは，2 つのソース・ファイル (AES.C と DO_AES.C) および 2 つのビルド・ファイル (BUILD_AES.COM と AES.OPT) で構成されます。これらのファイルは， CDSA AES サンプル・ディレクトリ (SYS$COMMON:[SYSHLP.EXAMPLES.CDSA.AES]) にあります。

このプログラムを実行する前に， CDSA を初期化する必要があります。初期化するには， 1 回だけ次のコマンドを実行します。

```
$ @SYS$STARTUP:CDSA$INITIALIZE
```

AES サンプル・プログラムをビルドするには，サンプル・ファイルをローカルのビルド領域にコピーし， BUILD_AES コマンド・ファイルを実行します。次のように行います。

```
$ copy SYS$SYSROOT:[SYSHLP.EXAMPLES.CDSA.AES]*.* local_build_area
$ SET DEF local_build_area
$ @AES_BUILD
```

その結果生成される AES.EXE ファイルは，フォーリン・コマンドとして次のように指定して，実行することができます。

```
$ AES :== $ local_build_area AES.EXE
```

このプログラムには，次のオプションを指定して実行することができます。

-e : 指定したキーで暗号化する (-k スイッチが必要)
-d : 指定したキーで復号化する (-k スイッチが必要)
-h : キーが 32， 48，または 64 文字 (16 進キー) であることを指定する
-k key : キー "key"を使用する (-h を指定した場合は，単一引用符が必要)

MYFILE.TXT を暗号化するのに， AES サンプル・ファイルを使用してキーを ASCII で指定する場合は，次のコマンドを実行します。

```
$ aes -e -k "xyzzy" MYFILE.TXT MYFILE.AES
```

上記ファイルを復号化するには，次のコマンドを実行します。

```
$ aes -d -k "xyzzy" MYFILE.AES MYFILE.TXT
```

暗号/復号するのに， 16 進 (ヘキサ) キーを使用するには，正確に 64 文字 (32 バイト) の長さでキーを入力し， -h スイッチを使用します。例を以下に示します。

```
$ aes -e -k 0123456789abcdefghijklmnopqrstuvwxyz0123456789abcdefghijklmnopqr -h MYFILE.TXT MYFILE.AES
$ aes -d -k 0123456789abcdefghijklmnopqrstuvwxyz0123456789abcdefghijklmnopqr -h MYFILE.AES MYFILE.TXT
```

注意

16 進数キーには，以下のものがあります。

- **256-bit AES (32 バイト， 256 ビット) では 64 文字指定します。 (CDSA のサンプル・プログラムはこの指定になっています。)**
- **192-bit AES (24 バイト， 192 ビット) では 48 文字指定します。**
- **128-bit AES (16 バイト， 128 ビット) では 32 文字指定します。**

サンプルを 128-bit AES または 192-bit AES サンプル用に変更するには，次のようにします。

1. aes.cをエディタで開きます。
2. キー・サイズを key[32]から以下のように変更します。

   key[24] (192-bit AES の場合)
   key[16] (128-bit AES の場合)

3. do_aesをエディタで開きます。
4. key.KeyHeader.AlgorithmId = CSSM_ALGID_EVP_AES_256;を以下のように変更します。

   key.KeyHeader.AlgorithmId = CSSM_ALGID_EVP_AES_192; (192-bit AES の場合)
   key.KeyHeader.AlgorithmId = CSSM_ALGID_EVP_AES_128; (128-bit AES の場合)

5. リビルドします。

5.3 Kerberos for OpenVMS

Kerberos for OpenVMS Version 2.1 は， MIT Kerberos V5 Release 1.2.6 with CERT patches up to Release 1.2.8 をベースにしています。 OpenVMS I64, OpenVMS Alpha，および OpenVMS VAX で， Kerberos クライアントと Kerberos サーバの両方をサポートしています。

Kerberos for OpenVMS Version 2.1 の新機能には，ktutilコマンドが含まれます。このコマンドを実行するとメニューが表示され，管理者は Kerberos V5 keytabファイル，または V4 srvtabファイルのエントリの読み取り，書き込み，編集を行うことができます。

Kerberos は，従来の (共有秘密鍵) 暗号化方式を使って，信頼できる第三者認証サービスとして認証を行ないます。 Kerberos は，ネットワーク上を流れるパケットは，意のままに，読まれたり，変更または挿入される可能性があるという前提で，ホスト・オペレーティング・システムによる認証に依存せず，ホスト・アドレスの信頼性に依存せず，ネットワーク上のすべてのホストの物理的なセキュリティを必要とせずに，本人確認をする機能を備えています。クライアントとサーバは， Kerberos を使って互いに相手を確認した後は，すべての通信を暗号化して機密性とデータ完全性を確保します。

詳細は，『HP Open Source Security for OpenVMS, Volume 3: Kerberos』を参照してください。

Kerberos for OpenVMS の最新版をダウンロードするための情報は，次の Web サイトを参照してください。

```
http://h71000.www7.hp.com/openvms/products/kerberos/
```

Kerberos についての追加情報は，次に示す MIT の Kerberos Web サイトを参照してください。

```
http://web.mit.edu/kerberos/www/
```

5.4 HP SSL for OpenVMS

HP SSL Version 1.2 は， OpenSSL 0.9.7d をベースにしています (以前のバージョンの HP SSL は， OpenSSL 0.9.6g をベースにしていました)。今回のリリースには，http://www.openssl.org/news/で， 2003 年の 9 月 30 日と 11 月 4 日，そして 2004 年 3 月 17 日に報告されたセキュリティの脆弱性に対応する不具合の修正が含まれています。

HP SSL は， OpenVMS I64， OpenVMS Alpha， OpenVMS VAX でサポートされます。

HP SSL Version 1.2 の新機能には， OCSP (Online Certificate Status Protocol)，AES (Advanced Encryption Standard)，楕円曲線暗号化機能が含まれています。これらの機能について，次に説明します。

- OCSP (Online Certificate Status Protocol)

  OCSP (Online Certificate Status Protocol) を使うと，アプリケーションは，CRL (証明書失効リスト) を使用した場合よりも，迅速に証明書のステータスを調べることができます。これは，サーバやクライアントのアプリケーションが， CA (Certificate Authority) から定期的 (毎週または毎月) に発行される CRL 情報ではなく， VA (Validation Authority) からリアルタイムに取得する証明書ステータス情報を使うことができるためです。 VA と CA は，同じ団体であっても構いませんが，同じ団体である必要はありません。

- AES (Advanced Encryption Standard)

  AES (Advanced Encryption Standard) は，新しい FIPS (Federal Information Processing Standard) 文書であり，慎重な扱いが必要な (機密扱いを受けていない) 情報を保護するために，米国政府機関によって使用される暗号化アルゴリズムを規定しています。 AES は，米国内の他の組織，機関，個人，および米国外でも，自由意志で広く使われています。 AES の暗号化アルゴリズムとして Rijndael が採用されています。

  AES は DES に代わるものとして開発されましたが， Triple DES は当面は承認されたアルゴリズム (米国政府が使用) として残ります。 Single DES は使用されなくなっています。

  AES には， 3 通りのキー・サイズがあります。 128 ビット， 192 ビット，および 256 ビットです。 56 ビットの DES キーの解読に 1 秒かかるコンピュータを使うと， 128 ビット AES キーの解読には 149 兆年かかる計算になります。

- 楕円曲線暗号化方式

  楕円曲線は， (x,y) 座標で緩やかにループする曲線を描く単純な関数です。楕円曲線は，公開鍵暗号化方式として使うことができます。この方法は高速で，小さな鍵を使っても，他の方法と同等のセキュリティ強度を持つという特徴があります。楕円曲線の利点は，公開鍵の計算に異なる種類の数学体系を使うことです。

RSA，SPEKE，Diffie-Hellman など多くの公開鍵暗号化方式を，楕円曲線と組み合わせて使うことができます。

SSL (Secure Sockets Layer) は，インターネットを使って機密情報を安全に送信するための，オープン標準のセキュリティ・プロトコルです。HP SSL ではインターネットその他の TCP/IP ネットワークを使った通信に関わる 3 つの基本的なセキュリティの課題について，次のように対処しています。

- SSL のサーバ認証

  SSL のサーバ認証機能を使うと，ユーザはサーバの身元確認を行なうことができます。SSL 対応のクライアント・ソフトウェアは，標準的な手法の公開鍵暗号化方式を使って，サーバの証明書と公開 ID が，正当であり，クライアントが持っている信頼できる CA (Certificate Authority) リスト中の CA によって発行されたものであることが確認できます。たとえば，PC ユーザが Web 上で買い物をするためにクレジット・カードの番号を送信するときに受信サーバの身元確認を行ないたい場合に，サーバ認証が使われます。

- SSL のクライアント認証

  SSL のクライアント認証機能を使うと，サーバはユーザの身元確認を行なうことができます。サーバ認証の場合と同様の手法を使って，SSL 対応のサーバ・ソフトウェアは，クライアントの証明書と公開 ID が，正当であり，サーバが持っている信頼できる CA (Certificate Authority) リスト中の CA によって発行されたものであることが確認できます。たとえば，銀行が顧客に機密金融情報を送信するときに，受信者の身元確認を行ないたい場合に，クライアント認証が使われます。

- 暗号化 SSL 接続

  暗号化 SSL 接続では，クライアントとサーバ間で送信されるすべての情報を送信ソフトウェアで暗号化して，受信ソフトウェアで復号化する必要があり，これによって高度の機密性を確保できます。機密性は機密を要するトランザクションの両当事者にとって重要です。また，暗号化 SSL 接続を通じて送信されるすべてのデータは，転送中の改竄を自動的に検出するメカニズムによって保護されます。

詳細は，『HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS』を参照してください。

最新バージョンの HP SSL for OpenVMS をダウンロードするための情報は，次の Web サイトを参照してください。

```
http://h71000.www7.hp.com/openvms/products/ssl/
```

OpenSSL についての詳細は，次のアドレスの OpenSSL Web サイトを参照してください。

```
http://www.openssl.org/
```

5.5 HP TCP/IP Services for OpenVMS Version 5.5

本リリースの OpenVMS には， I64 システムをサポートする新しいバージョンの TCP/IP Services for OpenVMS が含まれています。

TCP/IP Services Version 5.5 は，今回リリースされる OpenVMS V8.2 でのみサポートされ， Alpha システムと I64 システムの両方で動作します。 VAX システムでは，従来どおり TCP/IP Services Version 5.3 がサポートされます。このため，クラスタ環境では，使用しているオペレーティング・システムのバージョンに合ったバージョンの TCP/IP を使用するように注意してください。クラスタ構成とサポートが保証されている組み合わせについては，『OpenVMS Cluster 構成ガイド』を参照してください。

次の新機能が， Version 5.5 に追加されました。

Libpcap Library と TCPDUMP のアップデート	– libpcap API のサポート – TCPDUMP Version 3.8.3 のサポート
IPv6 構成のサポートと機能強化	– IPv6 の構成手順は機能強化され，構成オプションが増えました。 – FailSAFE IP での IPv6 のサポート (新しい ifconfig コマンドを含む)。 – SSH での IPv6 のサポート – PWIP (PATHWORKS Internet Protocol) ドライバでの IPv6 のサポート
NTP (Network Time Protocol) のサポート	– NTP Version 4.2.0 をサポートします。 NTP Version 3 および NTP Version 2 とは互換性がありますが， NTP Version 1 とは互換性がありません。 – NTP Version 1 よりも優れたセキュリティを備えています。

TCP/IP Services V5.5 の強化された機能と不具合の修正についての詳細は，『TCP /IP Services V5.5 Release Notes』を参照してください。

注意

なお， OpenVMS I64 対応の TCP/IP Services では，日本語版のキットは提供されません。 OpenVMS I64 上で TCP/IP Services をご利用になる場合は，英語版の TCP/IP Services をお使いください。

6

Volume Shadowing for OpenVMS におけるホスト・ベース・ミニマージ (HBMM)

Volume Shadowing for OpenVMS におけるホスト・ベース・ミニマージ (HBMM) については，オンライン・ドキュメント『Volume Shadowing for OpenVMS におけるホスト・ベース・ミニマージ』を参照してください。『Volume Shadowing for OpenVMS ホスト・ベース・ミニマージ』では，以下のような内容について説明しています。

- HBMM の構成要件
- HBMM の制限事項
- 複合バージョンまたは複合アーキテクチャの OpenVMS Cluster システムでの HBMM
- フル・マージ操作とミニマージ操作の概要
- ホスト・ベース・ミニマージ (HBMM) の概要
- HBMM ポリシー指定の構文
- HBMM ポリシーに適用される規則
- HBMM ポリシーを確立するためのガイドライン
- HBMM の構成と管理
- HBMM に影響を与える新しいシステム・パラメータ
- HBMM が有効な場合の/DEMAND_MERGE の使用
- マージ操作とコピー操作の優先順位付け
- 一時状態イベントの目に見える影響

7
Linker ユーティリティ

この章では， OpenVMS I64 システムでプログラムをリンクする前に検討すべき相違点と考慮事項の概要について説明します。主なトピックは，次のとおりです。

- VAX および Alpha システムでのリンクと I64 システムでのリンクの相違点 (第 7.1.1 項)
- OpenVMS I64 でのイメージのリンクの特徴 (第 7.1.2 項)
- Linker の新しい修飾子とオプション (第 7.1.3 項，第 7.1.4 項)
- Linker の既存の修飾子とオプションの新しい使い方 (第 7.1.5 項)
- I64 で拡張された Linker マップ・ファイル情報 (第 7.1.6 項)

Linker のリリース・ノートは，『HP OpenVMS Version 8.2 リリース・ノート[翻訳版]』を参照してください。

7.1 Linker ユーティリティの新機能

Linker の目的は，イメージ (バイナリのコードとデータを含むファイル) を作成することです。 OpenVMS I64 の Linker は， OpenVMS I64 オブジェクト・ファイルを受け付け OpenVMS I64 イメージを生成するという点で， OpenVMS VAX および Alpha システムの Linker とは異なります。 OpenVMS VAX および Alpha システムの Linker と同様に，作成するイメージの基本タイプは，**実行イメージ**です。このイメージは， DCL コマンド行で RUN コマンドを入力することで起動できます。

さらに， OpenVMS I64 Linker は**共有イメージ**も作成します。共有イメージは SYMBOL_VECTOR オプションを介してエクスポートされたプロシージャとデータの集まりで，実行イメージや他の共有イメージから呼び出すことができます。共有イメージは，実行イメージや共有イメージを作成するリンク操作で入力ファイルとして含めます。

OpenVMS I64 Linker でモジュールをリンクする際の入力ファイルの基本タイプは，**オブジェクト・ファイル**です。オブジェクト・ファイルは，コンパイラやアセンブラなどの言語処理プロセッサによって作成されます。これらのコンパイラによって作成されたオブジェクト・ファイルは， Intel® Itanium®アーキテクチャに固有のため， OpenVMS I64 システム上でモジュールをリンクする方法は， VAX や Alpha システム上でリンクする方法とは異なる点があります。ただし， Linker 全体で見ると， OpenVMS I64 Linker のインタフェースや機能 (たとえば，シンボルの解決，仮

想メモリ割り当て，イメージの初期化) は， OpenVMS VAX および Alpha システムの Linker と類似しています。

7.1.1 OpenVMS I64 システムでのリンク時の相違点

OpenVMS I64 システムでのリンクは， OpenVMS Alpha システムでのリンクと類似していますが，相違点もあります。 OpenVMS I64 Linker では，以下の修飾子とオプションは無視されます。

- /REPLACE
- /SECTION_BINDING
- /HEADER
- PER_PAGE—/DEMAND_ZERO=PER_PAGE の per_page キーワード (per_page キーワードは，若干異なった意味合いで将来サポートされます。)
- DZRO_MIN
- ISD_MAX

OpenVMS I64 Linker では，以下の修飾子とオプションは指定できません。

- /SYSTEM
- /DEBUG=修飾子でのファイル指定キーワード
- CLUSTER=cluster_name,base_address . . . でのヌルでないベース・アドレス (ベース・アドレスは，ヌルでなければならない)
- BASE=
- UNIVERSAL=

OpenVMS I64 Linker でサポートされる新しい修飾子を以下に示します。

- /BASE_ADDRESS
- /SEGMENT_ATTRIBUTE
- /FP_MODE
- /EXPORT_SYMBOL_VECTOR
- /PUBLISH_GLOBAL_SYMBOLS
- /FULL 修飾子の GROUP_SECTIONS および SECTION_DETAILS キーワード

これらの修飾子とオプションについて，以降の項で説明します。

7.1.1.1 I64 ではシンボルのデータ・タイプが一致する必要がある

OpenVMS Alpha では，シンボルをデータの 1 部として定義し，あたかも関数であるかのように参照することができます。 Alpha オブジェクト言語を使用しているときには， Linker がこの状態を検出する手段がありません。

一方， OpenVMS I64 では， Linker はシンボルのデータ・タイプの不一致を検出できます。 OpenVMS I64 Linker は，関数 (FUNC)，データの 1 部 (OBJECT)，または不明 (NOTYPE) としてシンボルをマークする情報を受け取ります。また， Linker は，関数として定義または参照されるシンボルの関数記述子をいつ作成するかを示す再配置情報も受け取ります。

シンボル・タイプが不明でない (NOTYPE でない) 場合，シンボル・タイプは一致する必要があります。シンボルの定義側が不明としてタイプを設定した場合には，参照側のタイプを関数にすることはできません。

例， 2 つのモジュール FIRST.C と SECOND.C がある場合

```
FIRST.C

#include <stdio.h>

int a;
int aa;
extern int second ();

void main () {

        printf ("The address of 'a'  is %x\n", &a);
        printf ("The address of 'aa' is %x\n", &aa);
        second ();
}

SECOND.C

#include <stdio.h>

extern int a();
extern int aa();

void second () {

        printf ("The address of 'a'  is %x\n", &a);
        printf ("The address of 'aa' is %x\n", &aa);
}
```

FIRST と SECOND をリンクすると次の情報メッセージ (シンボル記述子に基づく) と警告メッセージ (再配置情報に基づく) が Linker から出力されます。

```
$ link first,second
%ILINK-I-DIFTYPE, symbol AA of type OBJECT cannot be referenced as type FUNC
        module: SECOND
        file: DISK$USER:[JOE]SECOND.OBJ;5
%ILINK-I-DIFTYPE, symbol A of type OBJECT cannot be referenced as type FUNC
        module: SECOND
        file: DISK$USER:[JOE]SECOND.OBJ;5
%ILINK-W-RELODIFTYPE, relocation requests the linker to build a function
descriptor for a non-function type of symbol
        symbol: A
        relocation section: .rela$CODE$ (section header entry: 19)
        relocation type: RELA$K_R_IA_64_LTOFF_FPTR22
        relocation entry: 1
        module: SECOND
        file: DISK$USER:[JOE]SECOND.OBJ;5
%ILINK-W-RELODIFTYPE, relocation requests the linker to build a function
descriptor for a non-function type of symbol
        symbol: AA
        relocation section: .rela$CODE$ (section header entry: 19)
        relocation type: RELA$K_R_IA_64_LTOFF_FPTR22
        relocation entry: 4
        module: SECOND
        file: DISK$USER:[JOE]SECOND.OBJ;5
```

これらの情報メッセージと警告メッセージが出力されないようにするには，シンボル "A"と "AA"のタイプを関数またはデータに変更します。たとえば， FIRST.C での宣言を関数に変更します。

```
#include <stdio.h>

int a();
int aa();
extern int second ();

void main () {

        printf ("The address of 'a'  is %x\n", &a);
        printf ("The address of 'aa' is %x\n", &aa);
        second ();
}

int a () {
        return 1;
}

int aa () {
        return 1;
}
```

この変更を行えば，次のように，情報メッセージと警告メッセージは出力されなくなります。

```
$ link first,second
$
```

他の方法としては， SECOND での "A"と "AA"への参照をデータへの参照に変更することもできます。

7.1.1.2 ベース付き CLUSTER オプションの指定は OpenVMS プラットフォームによって異なる

CLUSTER オプションにベース・アドレスを指定することは， VAX システムと Alpha システムでのみ許されます。 Alpha システムでは，ベース付き CLUSTER の指定はメイン・イメージにのみ許されます。 VAX システムでは，さらに柔軟で共有イメージでもベース付き CLUSTER が許されます。 I64 システムでは， CLUSTER オプションのベース・アドレスの指定は，すべてのイメージで許されません。

7.1.1.3 OpenVMS I64 システムでの初期化されたオーバレイ・プログラム・セクションの取り扱い

Alpha システムと VAX システムでは，オーバレイ・プログラム・セクションの一部を初期化することができます。その後発生する同じ部分への初期化では，前のモジュールによる初期値を上書きします。任意のバイトに対する最後の初期値は，リンクされるイメージのそのバイトの最終的なデータとして使用されます。

I64 システムの ELF オブジェクト言語では，プログラム・セクションの一部を初期化する Alpha および VAX のオブジェクト言語の機能が現時点では実装されていません。初期化ではセクション全体が初期化されます。そのセクションに対するその後の初期化は，ゼロでない部分の値が一致するときのみ実行されます。これは，互換性のある初期化と呼ばれます。

たとえば以下の条件では， OpenVMS I64 システムと Alpha システムで結果が異なります。

それぞれで 2 つのロングワードを宣言している 2 つのプログラム・セクションがオーバレイされます。 1 番目のプログラム・セクションは 1 番目のロングワードを， 2 番目のプログラム・セクションは 2 番目のロングワードをそれぞれゼロ以外の値で初期化します。

Alpha システムでは， Linker は， 1 番目と 2 番目のロングワードが初期化されたイメージ・セクションを作成します。 VAX および Alpha のオブジェクト言語では，セクション・サイズとそれぞれのロングワードを初期化する Text Information Relocation (TIR) コマンドが Linker に渡されます。

I64 システムでは， Linker はコンパイラからの初期化データを含むセクションを受け取ります。 ELF には TIR コマンドが存在しないため， Linker は初期化を実行しません (初期化について関知しません)。 Linker は，最後に処理したプログラム・セクションの初期値を使用してイメージ・セグメントを作成します。すなわち，リンクされた順番に従って，イメージには， 1 番目または 2 番目のロングワードにゼロでない値が格納されます。 Linker はイメージを作成しますが，エラーと見なしメッセージを出力します。

I64 システムでは， Linker はすべての初期化情報を読み込み，互換性をチェックします。 2 つの初期化情報が，ゼロ以外の値で同値であれば互換性があります。初期化情報に互換性がない場合には， Linker は次のエラー・メッセージを出力します。

```
%ILINK-E-INVOVRINI, incompatible multiple initializations for
 overlaid section
        section:  <section name>
        module:   <module name for first overlaid section>
        file:  <file name for first overlaid section>
        module:  <module name for second overlaid section>
        file:  <file name for second overlaid section>
```

このエラー・メッセージには，ゼロ以外の初期化がなされた最初のモジュールと互換性のない初期化が検出された最初のモジュールが表示されます。すべての互換性のない初期化がリストされるわけではないことに注意してください。 Linker が検出した最初の互換性のないモジュールが表示されるだけです。

Linker マップの Program Section Synopsis では，ゼロ以外の初期化がされた各モジュールには "Initializing Contribution"のフラグが付けられます。この情報を使用して，すべての互換性のない初期化を識別し解決してください。

次の例では，マップ・ファイルの追加情報を示します (例 7–1 を参照)。

```
$ cre one.c
int common_data[]={0,0,47,11};
[Exit]
$ cc /extern=common one
$
$ cre two.c
int common_data[]={0,0,47,11};
[Exit]
$ cc /extern=common two
$
$ cre three.c
int common_data[]={0,0,0,0,0,0,0,0};
[Exit]
$ cc /extern=common three
$
$ link/map one,two,three
.
.
.
```

例 7–1 には，上記の例の Linker マップでの Program Section Synopsis を示します。 Align および Attributes フィールドは，通常 Length フィールドに続いて表示されますが，ページ内に収まるように変更してあります。

例 7–1 Program Section Synopsis を示す Linker マップ

```
                                +--------------------------+
                                ! Program Section Synopsis !
                                +--------------------------+

Psect Name      Module/Image     Base     End          Length
----------      ------------     ----     ---          ------
COMMON_DATA                    00010000 0001001F 00000020 (      32.)
                ONE            00010000 0001000F 00000010 (      16.)
                TWO            00010000 0001000F 00000010 (      16.)
                THREE          00010000 0001001F 00000020 (      32.)

Align                Attributes
-----               ------------
OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,  MOD
OCTA  4 Initializing Contribution
OCTA  4 Initializing Contribution
OCTA  4
```

次の例では，互換性のない初期化とその結果の Linker メッセージを示します。

```
$ cre four.c
int common_data[]={0,0,47,11,0,0,17,4};
[Exit]
$ cc /extern=common four
$
$link one,two,three,four
%ILINK-E-INVOVRINI, incompatible multiple initializations for
 overlaid section
        section: COMMON_DATA
        module: ONE
        file: DISK$USER:[JOE]ONE.OBJ;1
        module: FOUR
        file: DISK$USER:[JOE]FOUR.OBJ;1
```

この例は，外部共通モデルでコンパイルされています。VMS では，デフォルトの外部モデルは，緩やかな refdef モデルです。このモデルでは，明示的初期化が 1 つのみ許されます。つまり，同一の初期化でも，重複定義の警告メッセージが出力されます。上記の例と同じソース・ファイルを使用した例を次に示します。

```
$ cc one
$ cc two
$ cc three
$
$ link one, two, three
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: TWO
        file: DISK$USER:[JOE]TWO.OBJ;2
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: THREE
        file: DISK$USER:[JOE]THREE.OBJ;2
```

他の互換性のない初期化に対しては， Linker は INVOVRINI エラーと MULDEF 警告の両方を出力します。そのようなモジュールでは， INVOVRINI メッセージ，MULDEF メッセージの順に出力されます。このような場合には， INVOVRINI エラーを取り除くために MULDEF 警告を解決する必要があります。

```
$ cc four
$
$ link one,two,three,four
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: TWO
        file: DISK$USER:[JOE]TWO.OBJ;2
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: THREE
        file: DISK$USER:[JOE]THREE.OBJ;2
%ILINK-E-INVOVRINI, incompatible multiple initializations for
 overlaid section
        section: COMMON_DATA
        module: ONE
        file: DISK$USER:[JOE]ONE.OBJ;2
        module: FOUR
        file: DISK$USER:[JOE]FOUR.OBJ;2
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: FOUR
        file: DISK$USER:[JOE]FOUR.OBJ;2
```

7.1.1.4 ELF 共通シンボルのリンク時の動作の相違点

ELF 共通シンボル (または， Alpha の緩やかな refdef シンボル) が同じシンボルの定義を含むイメージに対して選択的にリンクされるときには， I64 Linker は異なる動作をします。 Alpha では， Linker は誤ってモジュールの緩やかな refdef シンボルから定義を取り出します。 I64 Linker は，共有イメージから定義を取り出します。

たとえば，共有イメージが下記のモジュール (my_int.c) からリンクされるとします。

```
#include ints
uint64 my_int ;
$cc/extern=relaxed my_int.c
$link/map/full/cross/share my_int,sys$input/opt
symbol_vector=(my_int=data)
```

次に別の C モジュール (x.c) が my_$int.exe に対して選択的にリンクされるとします。オブジェクトは緩やかな外部モデルでコンパイルされていることに注意してください。その結果， my_int に対して条件付き参照/定義が生成されます。

```
#include ints
uint64 my_int;
main()
{
my_int = 1;
return;
}
```

```
$cc/extern=relaxed x.c
$link/map/full/cross sys$input/opt
cluster=myclu,,,x.obj
my_int.exe/share/select
```

Alpha の Linker は，間違ってモジュールの条件付き定義の my_int から my_int を定義します。 I64 Linker は，正しく my_int.exe から定義を取り出します。

Alpha の Linker では，この動作に依存するプログラムを保護するために，変更は実施されていません。 I64 の Linker では，選択動作は正しく実行されます。

7.1.1.5 /TRACEBACK，/DEBUG，/DSF が使用されたときのフラグ設定

Linker デバッグ修飾子の/TRACEBACK，/DEBUG，および/DSF を指定すると，下記のフラグがイメージ・アクティベータ用に設定されます。これらのフラグは，タグ値 DT_VMS_LNKFLAGS の下で動的セグメントに設定されます。/TRACEBACK，/DEBUG および/DSF 修飾子の意味は， Alpha の Linker から変更されていませんが，フラグの意味と名称が変更されています。

フラグ	意味
VMS_LF_IMGSTA	イメージの実行は， SYS$IMGSTA を呼び出すことで開始されます。イメージ・アクティベータは，転送ベクタ (従来の VMS 形式) の最初のアドレスとして SYS$IMGSTA をインクルードします。
VMS_LF_CALL_DEBUG	SYS$IMGSTA は，デバッガを呼び出すかどうかを判断するためにこのフラグをチェックします。
VMS_LF_TBK_IN_IMG	トレースバック・レコードはイメージ・ファイル内に存在します。
VMS_LF_DBG_IN_IMG	デバッグ情報はイメージ・ファイル内に存在します。
VMS_LF_TBK_IN_DSF	トレースバック・レコードは DSF ファイル内に存在します。
VMS_LF_DBG_IN_DSF	デバッグ情報は DSF ファイル内に存在します。

フラグは，次の表に従って設定されます。

修飾子	IMGSTA	CALL_ DEBUG	TBK_IN _IMG	DBG_IN _IMG	TBK_IN _DSF	DBG_IN _DSF
/NoTrace/NoDebug /NoDSF	0	0	0	0	0	0
/Trace/NoDebug /NoDSF	1	0	1	0	0	0
/NoTrace /Debug /NoDSF	1	1	1	1	0	0
/Trace /Debug /NoDSF	1	1	1	1	0	0
/NoTrace /NoDebug /DSF	0	0	0	0	1	1
/Trace /NoDebug /DSF	1	0	1	0	1	1
/NoTrace /Debug /DSF	1	1	1	0	1	1
/Trace /Debug /DSF	1	1	1	0	1	1

注意

- SYS$IMGSTA の値は，イメージの転送配列に組み込まれなくなりました。SYS$IMGSTA の呼び出しを示すフラグのみが設定されます。イメージ・アクティベータは， SYS$IMGSTA の値を知っています。
- これらのフラグは， DSF ファイル中には現れません。 DSF ファイルは，イメージ・アクティベータによってアクティブにされません。(DSF ファイルには動的セグメントがないため， DT_VMS_LNKFLAGS フィールドがありません。)
- Linker は， I64 システムでは /DEBUG=ファイル名をサポートしなくなりました。別のデバッガを独立したイメージとしてリンクした後， LIB$DEBUG を定義してそのイメージをポイントするようにします。 SYS$IMGSTA は，LIB$DEBUG によってポイントされたものを常に呼び出します。
- /TRACEBACK または/DEBUG とともに/DSF を指定すると， VMS_LF_TBK_IN_IMG (イメージのトレースバック) フラグが設定されます。これは， Alpha での動作から変更されています。 Alpha での動作では，/TRACEBACK/DSF または/DEBUG/DSF を指定したときには，イメージにトレースバック・レコードが組み込まれませんでした。/DEBUG/DSF を指定したときにはデバッガ・レコードがイメージにコピーされないことに注意してください。/DEBUG のみを指定した場合は， IMGSTA ビットがイメージに設定されます。

次の表は， SYMBOL_TABLE や SYMBOL_VECTOR オプションを使用しないイメージのリンク操作で，どのような場合にグローバル・シンボル定義が書き込まれるかを示します。

修飾子	イメージのグローバル・シンボル	DSF ファイルのグローバル・シンボル
/NoTrace/NoDebug/NoDSF	0	0
/Trace/NoDebug/NoDSF	0	0
/NoTrace/Debug/NoDSF	1	0
/Trace /Debug/NoDSF	1	0
/NoTrace/NoDebug/DSF	0	1
/Trace/NoDebug/DSF	0	1
/NoTrace/Debug/DSF	0	1
/Trace/Debug/DSF	0	1

7.1.2 OpenVMS I64 システムでのリンクの特徴

以下の項では，I64 システムでイメージをリンクする際の I64 プラットフォームに特有の事項について説明します。トピックは，次のとおりです。

- リンケージ・メッセージの意味 (第 7.1.2.1 項)
- 縮小浮動小数点モデルを使ってコンパイルしたイメージについての留意事項 (第 7.1.2.2 項)
- ELF グループおよび UNIX スタイルの弱いシンボルとリンクする場合の留意事項 (第 7.1.2.3 項)
- 新しい/BASE_ADDRESS 修飾子の使用 (第 7.1.3.1 項)
- 新しい/SEGMENT_ATTRIBUTE 修飾子の使用 (第 7.1.3.2 項)
- 新しい/FP_MODE 修飾子の使用 (第 7.1.3.3 項)
- 新しい/EXPORT_SYMBOL_VECTOR 修飾子と/PUBLISH_GLOBAL_SYMBOLS 修飾子の使用 (第 7.1.3.4 項)
- /FULL 修飾子の新しい GROUP_SECTIONS および SECTION_DETAILS キーワードの使用 (第 7.1.3.5 項)
- PSECT_ATTRIBUTE オプションの新しいアラインメント (第 7.1.4.1 項)

7.1.2.1 リンケージ・メッセージの意味

一部の HP コンパイラは，呼び出し元や呼び出し先のルーチンで複数のモジュールにわたる汎用レジスタの使い方に一貫性があることが確認できるように，呼び出しリンケージ情報を生成します。リンケージ情報は，汎用レジスタ 0 ～ 31 についてだけ生成されます。リンケージ情報に問題がある場合には，衝突が発生しています。

Linker がリンケージの衝突を検出した場合に表示される警告メッセージの例を次に示します。

```
%ILINK-W-LNKGERR, linkage to routine X is not compatible with
 linkage of caller
        calling module: MOD_SRC
                file: DISK:[DIR]SOURCE.OBJ;1
        target  module: MOD_TARG
                file: DISK:[DIR]TARGET.OBJ;1
        source type of JSB to target type of CALL
        register AI not provided (needed at target)
        register GP not provided (needed at target)
        IA64 register R19 (Alpha R21) -- call=PRESERVE, target=VOLATILE
```

ソース情報とターゲット情報に続いて，警告メッセージが表示される原因となった衝突の原因を示すメッセージが表示されます。ターゲット・ルーチンでは引数情報を必要としているのに，呼び出し元では引数情報レジスタ (AI) に引数情報を設定していない場合には， Linker は上記の例のようなメッセージを出力します。

また，必要な情報が欠けている場合の他に，一貫性や互換性がない方法で汎用レジスタを使った場合にも，リンケージの衝突が発生することがあります。リンケージ情報には，次のような，汎用レジスタに関するレジスタ・ポリシーが含まれています。

- Volatile: このポリシーを持つレジスタは，プロシージャ間で情報を渡すための入力と出力のいずれにも使用できません。
- Scratch: このポリシーを持つレジスタは，呼び出し先のプロシージャで変更される可能性があります。
- Output: このポリシーを持つレジスタは，呼び出し元のプロシージャに情報を返すために使用することができます。
- Preserve: このポリシーを持つレジスタは，ターゲット・ルーチンが使う場合には，最初に内容を保存し，最後に復元しなければなりません。

リンケージ呼び出し規則のレジスタ・ポリシーは，次のとおりです。

Intel® Itanium®レジスタ	ポリシー
R2	Volatile
R3	Scratch
R4 - R7	Preserve
R8 - R9	Output
R10 -R11	Scratch
R12 - R18	Volatile
R19 - R24	Scratch
R26 - R31	Scratch

プロシージャ呼び出し規則では CALL メカニズムが使われ，グローバル・ポインタ (GP) の値と AI レジスタには引数情報が格納されます。

次の表では，呼び出し元のレジスタ・ポリシー (列の見出し) とターゲット・ルーチンのレジスタ・ポリシー (行の見出し) の間で互換性があるポリシーを示しています。

呼び出し元	ターゲット			
	Volatile	Scratch	Output	Preserve
Volatile	X			
Scratch		X		X
Output			X	X
Preserve				X

前のサンプル・メッセージを考えます。

```
IA64 register R19 (Alpha R21) - call-PRESERVE, target=VOLATILE
```

このメッセージでは， Intel® Itanium®レジスタ R19 は，呼び出し元では "Preserve"のレジスタ・ポリシーを持っていますが，ターゲット・ルーチンでは "Volatile"のレジスタ・ポリシーを持っています。上記の表に従えば， R19 は呼び出し元とターゲット・ルーチンの間で互換性がありません。

また，ルーチン間の呼び出しメカニズムにも互換性がありません。ターゲット・ルーチンでは CALL メカニズムが使用されることを期待しているにもかかわらず，呼び出し元ではターゲット・ルーチンへの JumptoSuBroutine (JSB) を実行しているからです。

Intel® Itanium®レジスタの中には固有のポリシーを設定しなければならないものがあります。これらのレジスタに正しいポリシーが設定されていない場合には， Linker は，ターゲット・リンケージが不正であることを示す警告メッセージ (1)，または呼び出し元が不正であることを示すメッセージ (2) を出力します。

```
(1) %ILINK-W-INVLNKG, invalid target linkage for symbol INV_LNKG
(2) %ILINK-W-INVRLNKG, invalid call linkage for symbol INV_LNKG
```

次の表に固有のポリシーを持つ Intel® Itanium®の汎用レジスタを示します。

Intel® Itanium®レジスタ	ポリシー
R2	Volatile
R12 ～ R18	Volatile

リンケージ・メッセージを生成する可能性がある衝突の説明

異なる言語間で呼び出しを実行 (たとえば， IMACRO から BLISS を呼び出す) した場合に， OpenVMS の呼び出し規則が Alpha システムと I64 システムで異なるときに，衝突が発生します。 Intel® Itanium®版の呼び出し規則では，デフォルトで保存されるレジスタの数が少なくなっています。そのため，ターゲット・ルーチンで保存されるレジスタと保存されないレジスタについての前提があります。

リンケージ文の不一致も Linker が警告メッセージを生成する原因になります。たとえば，呼び出し元のルーチンがレジスタ R12 ～ R15 (Alpha のレジスタ番号) が保存されることを期待しているのに，ターゲットが保存しない場合には， Linker はレジスタ R20， R21， R30，および R31 にリンケージの問題があることを示し，括弧内に Alpha のレジスタ番号も表示します。

また，レジスタを使っている関数でレジスタが正しく宣言されていない場合にも，問題が発生する可能性があります。たとえば，戻り値のためのレジスタに OUTPUT ではなく， NOPRESERVE を宣言しているような場合です。

このような衝突の可能性がある部分を修正するには， Linker が指摘するすべてのリンケージの定義と宣言を調べて，これらの問題点を修正し，問題点のないリンク操作が実行できるようにします。 Linker はこれらの問題を警告として扱うので，イメージの完了コードのステータスに SUCCESS は設定されません。共有イメージの場合は，これは，問題がある共有イメージとリンクされるイメージには， SUCCESS 以外の完了ステータスが与えられることを意味します。

7.1.2.2 縮小浮動小数点モデルを使ってコンパイルしたイメージについての留意事項

OpenVMS I64 システムでは，縮小浮動小数点モデルを使用しているイメージは，割り込みが発生した際の実行速度が速くなります。これは，割り込みが発生した際に，限定された数の浮動小数点レジスタだけを保存し復元すればよいからです。整数演算でも浮動小数点レジスタを使う場合があることに注意してください。イメージを構成するすべてのオブジェクト・モジュールを縮小浮動小数点モデルでコンパイルした場合にだけ，生成されたイメージを縮小浮動小数点モードで実行できます。 Linker マップ・ファイルの「Object and Image Synopsis」セクションには，モジュールが縮小浮動小数点モデルかどうかが表示されます。

7.1.2.3 ELF グループおよび UNIX スタイルの弱いシンボルとリンクする場合の留意事項

Intel C++ コンパイラでは， ELF (Executable and Linking Format) グループ (すなわち COMDAT) と UNIX スタイルの弱いシンボルを使用できるようになったため，Linker と Librarian はそれらを正しく処理する必要があります。

オブジェクト・モジュール内の ELF グループは，変数とプロシージャの定義を含む，一時的なコードとデータのコントリビューションと見なすことができます。 C++ コンパイラは C++ のテンプレートにこの機能を利用します。 C++ コンパイラは単にテンプレート (グループ) のためのコードとデータを生成します。そのような環境では，複数のオブジェクト・モジュールが複数の同一のコントリビューション (コードとデータが同じ) を持つことになります。イメージの場合は， Linker が，各グループから 1 つのコントリビューションを選択します。そのため，複数の同一グループを持つオブジェクト・モジュールを複数リンクすると，グループごとにコードとデータの 1 つのインスタンスがイメージに取り込まれます。

ELF グループは，名前を持っており，それはグループ署名とも言われます。グループは，名前が同じであれば，同一と見なされます。シンボルはグループに属することができ，そのようなシンボルをグループ・シンボルと言います。

通常どおり，グループ・シンボルは定義や参照に使用することができます。 UNIX スタイルの弱い定義は， VMS スタイルの弱い定義 (Alpha と VAX) とは異なり，重複して定義することができます。 UNIX スタイルの弱い参照は， Alpha および VAX スタイルの参照と似ていて，未解決の場合にエラーや警告は表示されません。

ELF グループの UNIX スタイルの弱いシンボルでは，特定のグループの最初に出現したシンボルを，定義インスタンスとして扱います。そして，そのグループの他の部分に出現する UNIX スタイルの弱いシンボルでその後に検出された定義は，すべて最初のインスタンスへの参照と見なされます。 UNIX スタイルの弱いシンボルは，本質的に一時的なシンボルです。弱いシンボルが存在し，リンクするその他のモジュールやイメージに強い定義が存在しない場合には，最初に出現した UNIX スタイルの弱いシンボルが，定義インスタンスと見なされます。

強い定義は， UNIX の弱い定義を上書きし，弱い定義は参照となります。同じグループの他の UNIX の弱いシンボルも参照になります。これは問題にはなりません。強い定義も含めすべてのシンボルはコンパイラによって生成されるか，または実行時環境で定義されるからです。このメカニズムを理解すると， Linker マップを理解する手助けになります。

グループ内に定義された UNIX の弱いシンボルは，そのグループ外の弱くないシンボルによって参照できます。

マップの相互参照リストでは， UNIX の弱い定義と参照には，その前に UNIX の弱いシンボルであることを示すタグ (UNIX weak タグ) が表示されます。通常は， UNIX weak タグの付いている定義は，オブジェクト・モジュールからグループを選択した結果であることを意味します。 UNIX weak タグの付いている参照は，通常，同じグループが 2 度目に出現した結果です。 UNIX weak タグの付いている参照を持つ定義でタグが付いていないものは，通常，強い定義がオブジェクト・モジュールのグループ・シンボルを上書きした結果として発生します。相互参照テーブルのタグを見れば，シンボルを定義しているインスタンスを所有するモジュールとしてどのモジュールが選択されたかがわかります。

相互参照には，グループ署名 (つまりグループ名) はリストされません。 ANALYZE ユーティリティを使用すれば，グループに属するすべてのシンボルを知ることができます。また， Linker に/MAP/FULL=GROUP_SECTIONS 修飾子を指定することで，すべてのグループと，それらを定義しているモジュールとファイルのリストを得ることができます。

ユーザが記述したテンプレートは，共有イメージとしてリンクできます。共有イメージによって (ユニバーサルに) エクスポートされたシンボルは常に強い定義になります。強い定義は UNIX の弱い定義より優先されるため，共有イメージのコードとデータは Linker によってコントリビューションとして選択されます。 OpenVMS I64 では，シンボルのグループ情報は，同じように共有イメージ内に保持されます。この場合，シンボルはすべて強い定義となるため，共有イメージからのグループはオブジェクト・モジュール内のすべての同一グループより常に優先されます。

共有イメージ内のグループとオブジェクト・モジュール内のグループには，相違点が1つあります。他の共有イメージに同じグループがあった場合， Linker はそのグループが2度目に出現したときに警告します。この場合，生成されたイメージは期待どおりの動作をしない可能性があります。グループの最初に出現したコードとデータが，定義元共有イメージ内のすべてのモジュールによって使用されます。ただし，他の場所に出現するコードとデータは，他の定義元共有イメージ内で使用されます。共有イメージをリンクする際には，テンプレートの実装に必要なすべてのグループのすべてのシンボルを必ずエクスポートしてください。そうしないと，冗長なコードとデータがイメージに組み込まれてしまったり，(前述のように) 間違ったコードとデータが使用されてしまうことになります。

以前からある OpenVMS スタイルの弱いシンボルも，従来どおり，使うことができます。

7.1.3 OpenVMS I64 用 Linker の新しい修飾子

OpenVMS I64 システムでのリンクをサポートするために， Linker に新しい修飾子がいくつか追加されました。ここでは，これらの修飾子について説明します。

7.1.3.1 新しい/BASE_ADDRESS 修飾子

新しい修飾子/BASE_ADDRESS が I64 システムに用意されました。ベース・アドレスは， Linker に割り当てを指示する，実行イメージの開始アドレスです。この修飾子は， OpenVMS イメージ・アクティベータによってアクティベートされないイメージ (たとえば，ブート処理で使用されるイメージ) に仮想アドレスを割り当てるためのものです。 OpenVMS イメージ・アクティベータは， Linker が割り当てた開始アドレスを無視することもできます。この修飾子は主にシステム開発者が使用します。

/BASE_ADDRESS 修飾子は， I64 では使用できない CLUSTER=,[base-address]オプションの代わりとなるものではありません。第 7.1.1.2 項を参照してください。

7.1.3.2 新しい/SEGMENT_ATTRIBUTE 修飾子

OpenVMS I64 Linker に，新しい修飾子/SEGMENT_ATTRIBUTE が追加されました。この修飾子には2つのキーワード SHORT_DATA=WRITE と DYNAMIC_SEGMENT=P0 または P2 を指定できます。/SEGMENT_ATTRIBUTE 修飾子の構文は次のとおりです。

```
/SEGMENT_ATTRIBUTE=([DYNAMIC_SEGMENT=(P0|P2) ], [SHORT_DATA=WRITE])
```

デフォルトでは， Linker は，イメージ・アクティベータ用の情報を持つ動的セグメントを P2 空間に配置します。しかし， DYNAMIC=P0 を指定すると， Linker はこの動的セグメントを P0 空間に配置するようになります。したがって， DYNAMIC_SEGMENT キーワードを指定する必要はほとんどありません。

SHORT_DATA=WRITE キーワードは，読み書き可能なショート・データ・セクションと読み取り専用/書き込み専用のショート・データ・セクションを1つのセグメントに結合することで，最大 65,535 バイトの未使用の読み取り専用領域を再利用できる

ようにします。SHORT_DATA に WRITE を指定すると，以前は読み取り専用だったデータにプログラムが誤って書き込みをする可能性があります。したがって，この修飾子は，ショート・データ・セグメントが 4 MB の限界に達してしまった場合にのみ使用することをお勧めします。

7.1.3.3 新しい/FP_MODE 修飾子

OpenVMS I64 Linker は，メインの遷移アドレスを提供するモジュールに設定された浮動小数点モードを使用して，プログラムの初期浮動小数点モードを決定します。メインの遷移アドレスを提供するモジュールに初期浮動小数点モードが設定されていない場合に限り，/FP_MODE 修飾子を使用して，初期浮動小数点モードを設定してください。/FP_MODE 修飾子は，メインの遷移モジュールに設定された初期浮動小数点モードより優先されることはありません。

OpenVMS I64 Linker では以下のキーワードを浮動小数点モードとして指定可能です。

- D_FLOAT, G_FLOAT—VAX の浮動小数点モードに設定
- IEEE_FLOAT[=ieee_behavior]—IEEE 浮動小数点モードに設定。動作を指定するか，指定しなければデフォルトの動作になります。

OpenVMS I64 Linker に指定可能な IEEE の動作キーワードには，以下のものがあります。

- FAST
- UNDERFLOW_TO_ZERO
- DENORM_RESULTS (デフォルト)
- INEXACT

OpenVMS I64 Linker には，浮動小数点モードの動作リテラルを指定することも可能です。初期浮動小数点モードの詳細は，『OpenVMS Calling Standard』を参照してください。

7.1.3.4 Linker の新しい修飾子: /EXPORT_SYMBOL_VECTOR と/PUBLISH_GLOBAL_SYMBOLS

/EXPORT_SYMBOL_VECTOR 修飾子と/PUBLISH_GLOBAL_SYMBOLS 修飾子が Linker に追加されました。これらの修飾子は，共有イメージを作成したいが，どのシンボルを SYMBOL_VECTOR オプションでエクスポートすればよいかわからないような場合に便利です。(UNIX からアプリケーションを移植しようとしていて，そのアプリケーションに精通していなければ，エクスポートすべきシンボルがわからないといった場合があります。他には， C++ でコーディングしていて，マングル化された名前がどのように見えるかがわからないといった場合があります。)

新しい/PUBLISH_GLOBAL_SYMBOLS 修飾子は，オブジェクト・モジュール内のすべてのグローバル・シンボルをシンボル・ベクタ・オプション・ファイルにエクスポート可能とするために，オブジェクト・モジュールにマーク付けを行います。新し

い/EXPORT_SYMBOL_VECTOR 修飾子は，/PUBLISH_SYMBOL_VECTOR でマーク付けされたモジュール内の各グローバル・シンボルに対するシンボル・ベクタ・オプションを出力ファイルに書き出します。/EXPORT_SYMBOL_VECTOR を指定したときには，オプション・ファイルへの書き込みが行われるだけで，イメージ・ファイルは生成されません。生成されたオプション・ファイルは，開発者が今後のリンクで用いるために， GSMATCH 情報を付加して完結させておく必要があります。

どちらの修飾子も，/SHAREABLE 修飾子が指定されている場合にのみ，有効です。/EXPORT_SYMBOL_VECTOR は，コマンド行でのみ指定できる修飾子です。/PUBLISH_GLOBAL_SYMBOLS は，オプション・ファイルでの使用も可能です。 Linker は，/EXPORT_SYMBOL_VECTOR 修飾子が指定されているときに，/PUBLISH_GLOBAL_SYMBOLS 修飾子が指定されていないと警告メッセージを出力します。

/EXPORT_SYMBOL_VECTOR=[file-spec]
/EXPORT_SYMBOL_VECTOR は， I64 Linker に対し，/PUBLISH_GLOBAL_SYMBOLS 修飾子で指定されたシンボル・ベクタ・オプションと， GSMATCH テンプレート・オプションが格納されたオプション・ファイルを作成するように指示します。

/EXPORT_SYMBOL_VECTOR 修飾子を使用すると，イメージの生成は抑止されます。

/EXPORT_SYMBOL_VECTOR を指定すると，入力の symbol_vector オプションは受け付けられません。

/PUBLISH_GLOBAL_SYMBOLS を少なくとも 1 つ，コマンド行またはオプション・ファイル内に指定する必要があります。

この修飾子には，リンク操作で作成するオプション・ファイルの名前として使用したい文字列を指定することもできます。文字列にファイル・タイプを指定しないと，デフォルトで，ファイル・タイプ.OPT が割り当てられます。/EXPORT_SYMBOL_VECTOR 修飾子にファイル名を指定しないと， Linker は，最初の入力ファイルのファイル名を使用してオプション・ファイルを作成します。

入力ファイル指定に/EXPORT_SYMBOL_VECTOR 修飾子を付加すると， Linker は，修飾子を付加したファイルのファイル名を使用してオプション・ファイルを作成します。

生成されたオプション・ファイルは， GSMATCH テンプレート・オプションを書き込んだ後，共有イメージ・ファイルを作成するために他のリンク操作で使用されます。

/PUBLISH_GLOBAL_SYMBOLS
/PUBLISH_GLOBAL_SYMBOLS 修飾子は，オブジェクト・モジュールやオブジェクト・モジュール・ライブラリにマーク付けし，/EXPORT_SYMBOL_VECTOR 修飾子で指定されたシンボル・ベクタ・オプション・ファイルへすべてのグローバル・シンボルをエクスポートするように， I64 Linker に指示します。

この修飾子は，/SHAREABLE 修飾子および/EXPORT_SYMBOL_VECTOR 修飾子と一緒に使用する必要があります。

この修飾子は，/INCLUDE 修飾子および/SELECTIVE_SEARCH 修飾子と互換性があります。この修飾子は，コマンド行と Linker オプション・ファイルで使用できます。

この修飾子をオブジェクト・ファイルに適用すると，そのオブジェクト・ファイル内のすべてのモジュールのすべてのグローバル・シンボルが，シンボル・ベクタ・オプションとしてエクスポートされることになります。

この修飾子を/SELECTIVE_SEARCH 修飾子と組み合わせて，オブジェクト・ファイルに適用すると，そのオブジェクト・ファイル内のすべてのモジュールのグローバル・シンボルのうち，参照されているもののみが，シンボル・ベクタ・オプションとしてエクスポートされることになります。

この修飾子をオブジェクト・ライブラリに適用すると，イメージ内に暗黙的に組み込まれた，オブジェクト・ライブラリのすべてのモジュールのすべてのグローバル・シンボルが，シンボル・ベクタ・オプションとしてエクスポートされることになります。

この修飾子を/INCLUDE 修飾子と組み合わせて，オブジェクト・ライブラリに適用すると，オブジェクト・ライブラリのインクルード・リストに記載されたモジュールのすべてのグローバル・シンボルが，シンボル・ベクタ・オプションとしてエクスポートされることになります。

/SELECTIVE_SEARCH 修飾子を使用してビルドされたオブジェクト・ライブラリにこの修飾子を適用すると，イメージに組み込まれた，オブジェクト・ライブラリのすべてのモジュールのグローバル・シンボルのうち，参照されているもののみが，シンボル・ベクタ・オプションとしてエクスポートされることになります。(選択的検索が可能なライブラリにモジュールを挿入する方法については，『OpenVMS Command Definition, Librarian, and Message Utilities Manual』を参照してください。)

グローバル・シンボルが，複数のモジュールで一時的な定義 (つまり， C の緩やかな refdef 外部モデルから生成) となっている場合，モジュールのいずれかが/PUBLISH でマーク付けされていれば，公開されています。同様に， UNIX の弱いシンボルの場合は，いずれのモジュールによっても，グローバル・シンボルを，シンボル・ベクタ・オプションとしてエクスポートする候補とすることができます。ただし，どちら

のシンボル・タイプの場合も，強い定義によってシンボルが上書きされる場合は，定義元モジュールに/PUBLISH があるかどうかによって，シンボルをシンボル・ベクタ・オプションとしてエクスポートする候補とするかどうかが決まります。

例:

```
$ link/SHARE public/PUBLISH,implementation/EXPORT=public
$ link/SHARE plib/LIBRARY/PUBLISH/INCLUDE=public/EXPORT=public
$ LINK/SHAREABLE public/PUBLISH, implementation/EXPORT=public
```

PUBLIC.OBJ 内のモジュール中のグローバル・シンボルがすべて，シンボル・ベクタ・オプションとしてファイル PUBLIC.OPT にエクスポートされます。

```
$ LINK/SHAREABLE api_table,implementation/PUBLISH/SELECTIVE/EXPORT=public
```

IMPLEMENTATION.OBJ 内のモジュール中のグローバル・シンボルのうち， API_TABLE.OBJ 内のモジュールから参照されているものだけが，シンボル・ベクタ・オプションとしてファイル PUBLIC.OPT にエクスポートされます。

```
$ LINK/SHAREABLE api_table,plib/LIBRARY/PUBLISH/EXPORT
```

共有イメージに組み込まれている，ライブラリ PLIB.OLB のすべてのモジュールのすべてのグローバル・シンボルが，シンボル・ベクタ・オプションとしてファイル PLIB.OPT にエクスポートされます。

```
$ LINK/SHAREABLE plib/LIBRARY/PUBLISH/INCLUDE=public/EXPORT
```

ライブラリ PLIB.OLB 内のモジュール public 中のグローバル・シンボルがすべて，シンボル・ベクタ・オプションとしてファイル PLIB.OPT にエクスポートされます。

```
$ LINK/SHAREABLE api_table, plib/LIBRARY/PUBLISH/SELECTIVE/EXPORT=public
```

api_table 内のすべてのモジュールから参照されている，ライブラリ PLIB.OLB のすべてのモジュール中のすべてのグローバル・シンボルが，シンボル・ベクタ・オプションとして，ファイル PUBLIC.OPT にエクスポートされます。また，同様の例でオプション・ファイルに/PUBLISH が指定されているものを次に示します。

```
$ link/SHAREABLE TT:/opt/EXPORT  public/PUBLISH, implementation
```

PUBLIC.OBJ 内のモジュール中のグローバル・シンボルがすべて，シンボル・ベクタ・オプションとして， TT: (端末への出力) にエクスポートされます。

7.1.3.5 /FULL 修飾子の新しいキーワード: GROUP_SECTIONS と SECTION_DETAILS

OpenVMS I64 Linker で，フル・イメージ・マップの作成を指示する/FULL 修飾子に 2 つのキーワードが追加されました。/FULL 修飾子のフォーマットは，次のとおりです。

/FULL [=(keyword [,...])]

OpenVMS I64 Linker は，マップの表示を調整するための次のキーワードを受け付けます (デフォルトは/FULL=SECTION_DETAILS)。

キーワード	意味
GROUP_SECTIONS	OpenVMS I64 Linker に対して，処理されたすべてのグループ (ELF COMDAT) をリストするように指示します。
NOSECTION_DETAILS	OpenVMS I64 Linker に対して，「Program Section Synopsis」での，長さがゼロのコントリビューションの出力を抑止するように指示します。
ALL	OpenVMS I64 Linker では， ALL キーワードを指定すると， GROUP_SECTIONS キーワードと SECTION_DETAILS キーワードの両方を指定したのと同じになります。

最初のキーワード， GROUP_SECTIONS は，使用されたグループをすべてマップに出力します。現時点で，グループを活用できるコンパイラは， C++ だけです。このキーワードを C++ 以外の言語で使用しても効果はありません。

/FULL=NOSECTION_DETAILS を指定すると， OpenVMS I64 Linker は，マップの「Program Section Synopsis」での長さがゼロのコントリビューションの出力を抑制します。修飾子/FULL を使用すると，デフォルトの/FULL=SECTION_DETAILS になり， VAX， Alpha，および I64 システムでのフル Linker マップとして，すべてのモジュール・コントリビューションが「Program Section Synopsis」にリストされます。

7.1.4 OpenVMS I64 の Linker の新しいオプション

OpenVMS I64 システムでのリンクをサポートするために， Linker に新しいオプションがいくつか追加されました。ここでは，これらの機能について説明します。

7.1.4.1 PSECT_ATTRIBUTE オプションの新しいアラインメント

PSECT_ATTRIBUTE オプションには，アラインメント属性として， 5， 6， 7，および 8 が追加されました。この整数値は， 2 のべき乗として示されるバイト・アラインメントを表します。たとえば， 6 の場合には， 2 ** 6 で， 64 バイト・アラインメントを表します。 2 ** 5 に対して，キーワード HEXA (16 ワードの意味) が追加されました (これは 32 バイト・アラインメント， 16 ワード・アラインメントを意味します)。

7.1.5 Linker の既存の修飾子とオプションの新しい使用方法

以下の項では， I64 システムでの， Linker の既存の修飾子とオプションの新しい使用方法を説明します。トピックは，次のとおりです。

- Linker オプションでの大文字と小文字が混在した引数の使用 (第 7.1.5.1 項)

- イメージ名を指定する場合の規約 (第 7.1.5.2 項)
- アラインメントを指定する PSECT_ATTRIBUTE オプションの使用 (第 7.1.5.3 項)

7.1.5.1 I64 システムの Linker オプションでの大文字と小文字が混在した引数

OpenVMS I64 システムでは，コンパイラが大文字と小文字が混在した名前を生成することがあります。オプション・ファイル内の大文字と小文字が混在した名前を処理させる必要がある場合 (たとえば，ライブラリのインクルード文を使用していて，モジュールの名前に大文字と小文字が混在している場合) は，デフォルトの動作 (すべて大文字に変換してから処理する) を使用しないで， Linker のオプションを使って，大文字と小文字が混在した名前を処理するようにします。そのオプションは，CASE_SENSITIVE で，次のように指定します。

```
CASE_SENSITIVE=YES
```

CASE_SENSITIVE オプションに YES を設定すると， (行の左端からみて) 最初の等号より右側にあるすべての文字 (たとえば，オプションの引数) は大文字/小文字の区別が維持されます。つまり，これらの文字は (大文字に) 変換されずにそのまま処理されます。ファイル名，モジュール名，シンボル名，キーワードが対象です。 Linker のデフォルト動作の行全体を大文字に変換する設定に戻すには，次のように NO キーワード付きの CASE_SENSITIVE オプションを指定します。

```
CASE_SENSITIVE=NO
```

NO キーワードを小文字 (no) で指定すると， Linker によって認識されないため，大文字で指定する必要があることに注意してください。

たとえば，次の大文字と小文字が混在した名前を含むモジュールを扱う際に， Linker に大文字/小文字を区別するモードを設定して，大文字と小文字をそのまま残す場合を考えます。

```
case=Yes
My_Lib/library/include=(Add_Func, Sub_Func)
symbol_vector=(Add_Func=PROCEDURE,PAGE_COUNT=DATA)
case=NO
```

Linker によって処理されると，このテキストは次のようになります。

```
CASE=YES
MY_LIB/LIBRARY/INCLUDE=(Add_Func,Sub_Func)
SYMBOL_VECTOR=(Add_Func=PROCEDURE,PAGE_COUNT=DATA)
CASE=NO
```

各オプションの最初の等号より右側のすべての文字は大文字/小文字を区別したまま読み込まれていることがわかります。

VAX と Alpha での動作を保持するために， CASE_SENSITIVE=YES は，必要な場合にのみ使用することをお勧めします。

7.1.5.2 イメージ名を指定する場合の規約

次に示す規約は，イメージに適用される各種の名前の説明です。

- イメージには，イメージ・ファイル指定が付けられます (たとえば， FOO.EXE)。これは， DCL の RENAME コマンドで変更できます。
- イメージ名は NAME=オプションで指定され， NT_VMS_IMGNAM 型の注釈としてイメージに格納されます。この名前はイメージ・ファイル指定の名前とは異なっても構いませんが， NAME=オプションを使わない場合には，この名前はデフォルトでイメージ・ファイル指定の名前と同じになります。 Analyze ユーティリティではこの名前を「イメージ名」として表示します。この名前は DCL の RENAME コマンドでは変更できません。
- これ以外に，イメージにはグローバル・シンボル・テーブル (GST) に関連付けられる名前があり， NT_VMS_GSTNAM 型の注釈としてイメージに格納されます。 Linker はこの名前としてイメージ・ファイル指定の名前と同じものを設定します。この名前はイメージをイメージ・ライブラリに追加する際に， Librarian によって使われます。 Analyze ユーティリティでは，「グローバル・シンボル・テーブル名」として表示されます。この名前は DCL の RENAME コマンドでは変更できません。

Alpha システムでは， NAME=に指定したイメージ名は，共有イメージ・リスト内での自己参照を識別するために使われます。自己参照は，シンボル・ベクタ内の別名を使った，イメージ内部からの自身の呼び出しです。

I64 システムでは，共有イメージ・リスト内には現在のイメージに対するエントリがありません。自己参照は，共有イメージ・リスト (つまり， DT_NEEDED エントリの集まり) に対する特殊なインデックス値 (DT_VMS_FIXUP_NEEDED フィールドが -1) によって参照されます。

7.1.5.3 PSECT_ATTRIBUTE オプションによるアラインメントの指定

コンパイラがセクションに割り当てたアラインメントより小さいセクション・アラインメントを PSECT_ATTRIBUTE で指定しないでください。

そのセクションに組み込まれたすべてのコントリビューションを元に，コンパイラが割り当てたアラインメントより小さいアラインメントをプログラム・セクションに指定すると， Linker は警告を出力するようになりました。例を以下に示します。

```
$ link hi,sys$input/opt
psect_attr=$literal$,byte
%ILINK-W-CONFALGN, PSECT option alignment (1) less than compiler
assigned (16);
alignment ignored
        section: $LITERAL$
        module: HI
        file: DISK$USER:[JOE]HI.OBJ;3
```

VAX システムと Alpha システムでは， Linker は指定された境界 (上記のコード例では "byte") でプログラム・セクションを不適切にアラインし，コンパイラの指定とは異なる境界で，そのプログラム・セクションのすべてのコントリビューションを (上記の例では "hi"とリンクされている，他のモジュールから) 配置します。 Linker はエラー・メッセージは出力しません。

I64 システムでは， Linker は常にコンパイラが指定した境界以上の境界でセクションをアラインします。

PSECT_ATTRIBUTE オプションは，指定された境界が，コンパイラの指定以上のときに，セクションをアラインします。このオプションは，セクションの個々のコントリビューションをアラインするのではなく，セクション全体をアラインします。PSECT_ATTRIBUTE オプションは，個々のコントリビューションをアラインする際には，コンパイラのアラインメント指定に従います。

7.1.5.4 存在しないファイルがあった場合の Linker の特別な処理

Linker オプションの RMS_RELATED_CONTEXT がオンになっていて (デフォルトは RMS_RELATED_CONTEXT=YES)， LINK コマンドに指定したファイル・リストに存在しないファイルを指定していると， Linker が呼び出した LIB$FIND_FILE が完了するのに時間がかかり， Linker がハングしているように見えることがあります。リンクされているファイルの数と，ファイル指定で論理名を使用しているかどうかによりますが， LIB$FIND_FILE は，見つからないファイルを接頭辞のあらゆる組み合わせで探してから "file not found"メッセージを表示するため， Linker の完了に数時間かかることがあります。 Linker が LIB$FIND_FILE を呼び出した後は，Ctrl/Y で Linker プロセスを終了させることはできません。

どれが存在しないファイル指定かを素早く見つけるには， LINK コマンドに /OPTION を指定します。[Return]を押すと， Linker はオプション・ファイルへの情報の入力待ち状態になります。オプションの指定が終わったら， Ctrl/Z を入力します。 LIB$FIND_FILE に関連する問題を回避するには，次の項目をオプション・ファイルに含めます。

- 1 行目に，次のオプションを指定します。

  ```
  RMS_RELATED_CONTEXT=NO
  ```

  RMS_RELATED_CONTEXT オプションに NO を設定しておくと，このオプション・ファイルにリストされたファイルが見つからない場合は，すぐに "file not found"メッセージが出力されます。

- すぐ次の行に，リンクするファイルを，完全なファイル指定の形式 (*disk*:[*dir*]*filename.ext*) で，ファイル**毎**に指定します。完全なファイル指定が必要になる理由は， RMS_RELATED_CONTEXT=NO を指定したときには，ファイル名の「スティッキー・モード」が無効になるためです。

次の LINK コマンドを例に説明します。

```
$ LINK DSK:[TEST]A.OBJ, B.OBJ
```

このコマンドに RMS_RELATED_CONTEXT=NO を指定したい場合は，以下のように/OPTION を指定し，さらに，リンクするファイルを完全ファイル指定で入力します。

```
$  LINK SYS$INPUT:/OPTION
RMS_RELATED_CONTEXT=NO
DSK:[TEST]A.OBJ, DSK:[TEST]B.OBJ Ctrl/Z
$
```

例

次の例は， Linker がハングしたように見える様子を示しています。ファイル DOES_NOT_EXIST.OBJ がリストに含まれているが存在せず， RMS_RELATED_CONTEXT オプションは指定していない (デフォルトで YES となる) ものとします。

```
$  DEFINE DSKD$ WORK4:[TEST.LINKER.OBJ.]
$  DEFINE RESD$ ROOT$, ROOT2$, ROOT3$, ROOT4$, ROOT5$, DISK_READ$:[SYS.] 1
$  DEFINE ROOT$ WORK4:[TEST.PUBLIC.TEST]
$  DEFINE ROOT2$ WORK4:[TEST.LINKER.]
$  DEFINE ROOT3$ WORK4:[TEST.UTIL32.]
$  DEFINE ROOT4$ WORK4:[TEST.PUBLIC.]
$  DEFINE ROOT5$ WORK4:[TEST.PUBLIC.TMP]
$  LINK/MAP/FULL/CROSS/EXE=ALPHA.EXE  RESD$:[TMPOBJ] A.OBJ,-
_$  RESD$:[SRC]B.OBJ,C,DSKD$:[OBJ]D.OBJ,E,RESD$:[TMPSRC]F.OBJ,-
_$  RESD$:[TEST]G.OBJ,RESD$:[SRC.OBJ]H,RESD$:[COM]DOES_NOT_EXIST.OBJ
Ctrl/T NODE6::_FTA183: 15:49:46 LINK CPU=00:02:30.04 PF=5154 IO=254510 MEM=134 2
Ctrl/T NODE6::_FTA183: 15:49:46 LINK CPU=00:02:30.05 PF=5154 IO=254513 MEM=134
Ctrl/T NODE6::_FTA183: 15:50:02 LINK CPU=00:02:38.27 PF=5154 IO=268246 MEM=134
Ctrl/T NODE6::_FTA183: 15:50:02 LINK CPU=00:02:38.28 PF=5154 IO=268253 MEM=134
Ctrl/T NODE6::_FTA183: 15:50:14 LINK CPU=00:02:44.70 PF=5154 IO=278883 MEM=134
```

1 このコマンドは論理名と等価名を定義しています。

2 Ctrl/T を入力するたびに， CPU と IO の値は増加していますが， MEM と PF の値は変化していないため， LIB$FIND_FILE が呼び出されていることがわかります。

以下に示す例のように，オプション・ファイルを使用して， RMS_RELATED_CONTEXT に NO を設定すると，存在しないファイルに遭遇したときに，直ちにリンク操作を終了させることができます。

```
$  DEFINE DSKD$ WORK4:[TEST.LINKER.OBJ.]
$  DEFINE RESD$ ROOT$, ROOT2$, ROOT3$, ROOT4$, ROOT5$, DISK_READ$:[SYS.]
$  DEFINE ROOT$ WORK4:[TEST.PUBLIC.TEST.]
$  DEFINE ROOT2$ WORK4:[TEST.LINKER.]
$  DEFINE ROOT3$ WORK4:[TEST.UTIL32.]
$  DEFINE ROOT4$ WORK4:[TEST.PUBLIC.]
$  DEFINE ROOT5$ WORK4:[TEST.PUBLIC.TMP.]
$  LINK/MAP/FULL/ CROSS /EXE=ALPHA.EXE SYS$INPUT:/OPTION
RMS_RELATED_CONTEXT=NO
```

```
RESD$:[TMPOBJ]A.OBJ,RESD$:[SRC]B.OBJ,RESD$:[SRC]C,DSKD$:[OBJ]D.OBJ
DSKD$:[OBJ]E,RESD$:[TMPSRC]F.OBJ,RESD$:[TEST]G.OBJ
RESD$:[SRC.OBJ]H,RESD$:[COM]DOES_NOT_EXIST.OBJ Ctrl/Z

%LINK-F-OPENIN, error opening DISK_READ$:[SYS.][COM]DOES_NOT_EXIST.OBJ; as input
-RMS-E-FNF, file not found
$
```

7.1.6 新しい OpenVMS I64 Linker マップ

Linker マップは拡張され， OpenVMS I64 Linker 用の新しい情報が追加されました。 Linker マップには，次の情報が表示されます。サンプル・マップを図 7–1，図 7–2，図 7–3，図 7–4，図 7–5，図 7–6，図 7–7，図 7–8 に示します。

- オブジェクトとイメージの概要
- クラスタの概要
- イメージ・セグメントの概要
- プログラム・セクションの概要
- シンボルの相互参照
- シンボル一覧 (値順)
- イメージの概要
- リンク処理の統計情報

Linker マップに新たに追加された部分と，変更された部分を示す，白抜きの黒丸数字部分についての説明は， Linker マップ例の後にあります。

図 7–1 オブジェクトとイメージの概要，クラスタの概要

```
28-OCT-2004 13:34        Linker I02-17

+-------------------------------+
! Object and Image Synopsis ! 1
+-------------------------------+

Module/Image 2  File                    Ident          Attributes 3        Bytes  Creation Date      Creator
-------------   ----                    -----          ----------          -----  -------------      -------
GETJPI          SYS$SYSROOT:[SYSMGR]GETJPI.OBJ;1  V1.0            Lkg    Dnrm      360  28-OCT-2004 13:32  HP C V7.1-005
DECC$SHR        SYS$COMMON:[SYSLIB]DECC$SHR.EXE;1  V8.2-00            Lkg              0  21-OCT-2004 11:23  Linker T02-17
SYS$PUBLIC_VECTORS  SYS$COMMON:[SYSLIB]SYS$PUBLIC_VECTORS.EXE;1  X-3            Sel Lkg              0  21-OCT-2004 11:23  Linker T02-17

  Key for Attributes
  +-----------------------------------------+
  ! Sel  - Module was selectively searched  !
  ! Lkg  - Contains call linkage information !
  ! Dnrm - Denormal IEEE FP model           !
  +-----------------------------------------+

                                  +-------------------+
                                  ! Cluster Synopsis ! 4
                                  +-------------------+

Cluster 5       Match 6     Majorid   Minorid
-------         -----       -------   ---------
MYCLU
DEFAULT_CLUSTER             LESS/EQUAL      1          1
DECC$SHR                    EQUAL           9114  3906113603
SYS$PUBLIC_VECTORS
```

VM-1173A-AI

図 7–2 イメージ・セグメントの概要

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34     Linker I02-17

                                  +-------------------------+
                                  ! Image Segment Synopsis ! 7
                                  +-------------------------+

 Seg#  Cluster          Type     Pglts   Base Addr  Disk VBN  PFC  Protection  Attributes
 ----  -------          ----     -----   ---------  --------  ---  ----------  ----------
    0  MYCLU            LOAD         1   00010000          2    0  READ WRITE
    1                   LOAD         1   00020000          0    0  READ WRITE  DEMAND ZERO
    2                   LOAD         1   00030000          3    0  READ ONLY   EXECUTABLE,SHARED
    3                   LOAD         1   00040000          4    0  READ ONLY   SHARED
    4                   LOAD         1   00050000          5    0  READ ONLY   [UNWIND]
    5  DEFAULT_CLUSTER  LOAD         1   00060000          6    0  READ ONLY   SHORT
    6                   DYNAMIC      2  Q-00000000
                                        80000000          7    0  READ ONLY
 8

                  Key for special characters above
                    +-------------------+
                    !  Q  - Quadword    !
                    +-------------------+
```

VM-1174A-AI

図 7–3 プログラム・セクションの概要

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1                28-OCT-2004 13:34      Linker I02-17
                                  +-----------------------------+
                                  ! Program Section Synopsis !
                                  +-----------------------------+

  Psect Name     Module/Image      Base     End         Length          Align      Attributes (9)
  ----------     ------------      ----     ---         ------          -----      ----------

  ITMLST                       00010000 0001000F 00000010 (      16.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,  MOD
                 GETJPI        00010000 0001000F 00000010 (      16.) OCTA  4 Initializing Contribution (11)

  FILLEN                       00020000 00020003 00000004 (       4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker> (10) 00020000 00020003 00000004 (       4.) OCTA  4

  FILLM                        00020010 00020013 00000004 (       4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker>      00020010 00020013 00000004 (       4.) OCTA  4

  IOSB                         00020020 00020027 00000008 (       8.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker>      00020020 00020027 00000008 (       8.) OCTA  4

  STATUS                       00020030 00020033 00000004 (       4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker>      00020030 00020033 00000004 (       4.) OCTA  4
```

VM-1175A-AI

図 7–4 プログラム・セクションの概要 (続き)

```
$CODE$                  00030000 0003015F 00000160 (   352.) OCTA  4 CON,REL,LCL,  SHR,  EXE,NOWRT,NOVEC,  MOD
        GETUPI          00030000 000300FF 00000100 (   256.) OCTA  4
        <Linker>  12    00030100 0003015F 00000060 (    96.) OCTA  4

$LINK$                  00040000 00040000 00000000 (     0.) OCTA  4 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,NOMOD
        GETUPI          00040000 00040000 00000000 (     0.) OCTA  4

$LITERAL$               00040000 00040017 00000018 (    24.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
        GETUPI          00040000 00040017 00000018 (    24.) OCTA  4

$READONLY$              00040020 0004002F 00000010 (    16.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
        GETUPI          00040020 0004002F 00000010 (    16.) OCTA  4

$LINKER_UNWIND$         00050000 00050017 00000018 (    24.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
        GETUPI          00050000 00050017 00000018 (    24.) QUAD  3

$LINKER_UNWINFO$        00050018 0005002F 00000018 (    24.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
        GETUPI          00050018 0005002F 00000018 (    24.) QUAD  3

$LINKER_SYMBOL_VECTOR$  00060000 00060007 00000008 (     8.) OCTA  4 CON,REL,GBL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD,SHORT
        <Linker Option> 00060000 00060007 00000008 (     8.) OCTA  4

$LINKER_SDATA$          00060008 000600AF 000000A8 (   168.) OCTA  4 CON,REL,GBL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD,SHORT
        <Linker>        00060008 000600AF 000000A8 (   168.) OCTA  4
```

VM-1176A-AI

図 7–5 シンボルの相互参照

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34    Linker I02-17


                    +-----------------------------+
                    ! Symbol Cross Reference !
                    +-----------------------------+

Symbol          Value           Defined By          Referenced By ...
------          -----           ----------          ----------------
DECC$TXPRINTF   00000496-X (13)   DECC$SHR            GETJPI
ELF$TFRADR      00060050-R      WK-GETJPI
FILLEN          00020000-R        GETJPI            GETJPI
FILLM           00020010-R        GETJPI            GETJPI
GETJPI (U)      00000000          <Linker Option>
INTERNAL_GETJPI 00060098-R        GETJPI
IOSB            00020020-R        GETJPI            GETJPI
ITMLST          00010000-R        GETJPI
STATUS          00020030-R        GETJPI            GETJPI
SYS$GETJPIW     0000009A-X        SYS$PUBLIC_VECTORS  GETJPI
```

VM-1177A-AI

図 7–6 シンボル一覧 (値順)

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34    Linker I02-17

                                        +--------------------+
                                        ! Symbols By Value !
                                        +--------------------+

  Value       Symbols...
  -----       ----------
  00000000    GETJPI (U)
  0000009A  X-SYS$GETJPIW
  00000496  X-DECC$TXPRINTF
  00010000  R-ITMLST
  00020000  R-FILLEN
  00020010  R-FILLM
  00020020  R-IOSB
  00020030  R-STATUS
  00060050  R-ELF$TFRADR
  00060098  R-INTERNAL_GETJPI

                                        Key for special characters above (14)
                                        +-------------------------+
                                        !   *  - Undefined        !
                                        ! (U) - Universal         !
                                        !   R  - Relocatable      !
                                        !   X  - External         !
                                        !   C  - Code Address     !
                                        !  WK  - Weak             !
                                        ! UxWk - Unix-Weak        !
                                        +-------------------------+
```

VM-1178A-AI

図 7–7 イメージの概要

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34      Linker I02-17

                                        +-----------------+
                                        ! Image Synopsis !
                                        +-----------------+

  Virtual memory allocated:               00010000 0006FFFF 00060000 (393216. bytes, 768. pages)
  64-Bit Virtual memory allocated:        00000000 00000000 00000000
                                          80000000 80010000 00010000 (65536. bytes, 128. pages)
    Stack size:                                      0. pages
    Image header virtual block limits:               1.        1. (    1. block)
    Image binary virtual block limits:               2.        8. (    7. blocks)
    Image name and identification:                GETJPI V1.0
    Number of files:                                    5.
    Number of modules:                                  3.
    Number of program sections:                         9.
    Number of global symbols:                        3338.
    Number of cross references:                        17.
    Number of image segments:                           7.
    Transfer address from module:                    GETJPI
    User transfer FD address:               00000000 00060050
    User transfer code address:             00000000 00030000
    Initial FP mode:                        00000000 09800000 (IEEE DENORM_RESULTS)
  Number of code references to shareable images:       3332.
  Image type:                                     SHAREABLE. Global Section Match=EQUAL, Ident, Major=9120, Minor=2068313277
  Reduced Floating Point model (RFP):             Image does not use RFP model
  Map format:                                     FULL WITH CROSS REFERENCE in file SYS$SYSROOT:[SYSMGR]GETJPI.MAP;1
  Estimated map length:                           441. blocks
```

VM-1179A-AI

図 7–8 リンク処理の統計情報

```
+-------------------------+
! Link Run Statistics !
+-------------------------+

Performance Indicators                Page Faults      CPU Time       Elapsed Time
----------------------------          ------------     --------       ------------
  Command processing:                          55     00:00:00.00    00:00:00.00
  Pass 1:                                     173     00:00:00.06    00:00:00.05
  Allocation/Relocation:                        5     00:00:00.02    00:00:00.02
  Pass 2:                                      32     00:00:00.01    00:00:00.00
  Symbol table output:                          4     00:00:00.00    00:00:00.00
  Map data after object module synopsis:        5     00:00:00.00    00:00:00.07
Total run values:                             274     00:00:00.09    00:00:00.17

Quota usage (15)                      ByteCount  FileCount  PgFlCount
------------                          ---------  ---------  ---------
  Available:                             127808        100     512000
  Command processing:                       384          2       7888
  Pass 1:                                   384          2      10240
  Allocation/Relocation:                    576          3      10240
  Pass 2:                                   576          3      18624
  Symbol table output:                      384          2      18624
  Map data after object module synopsis:    384          2      18624

Using a working set limited to 16384 pages and 11029 pages of data storage (excluding image)

Number of modules extracted explicitly          = 0
   with 0 extracted to resolve undefined symbols


1 library searches were for symbols not in the library searched

A total of 8 global symbol table entries was written

LINK/DEB/MAP/FULL/CROSS/SHARE GETJPI.OPT/OPT
<SYS$SYSROOT:[SYSMGR]GETJPI.OPT;2>
cluster=myclu,,,getjpi.obj
symbol_vector=(getjpi/internal_getjpi=procedure
```

VM-1180A-AI

以下の説明は，上記の Linker マップの例の中の数字のついた項目に対応しています。

1 Object and Image Synopsis。 Alpha システムで Object Module Synopsis というタイトルが付いていたセクションは， I64 システムでは Object and Image Synopsis というタイトルに変更されました。

2 Module/Image。 Alpha システムで Module Name というタイトルが付いていた欄は， I64 システムでは Module/Image というタイトルに変更されました。

3 Attributes。 4 つの欄で構成される Attributes というタイトルが付いた新しい情報が追加されました。最初の欄には，モジュールの検索が選択型 (selective) かどうかが示されます。選択型の場合には，「Sel」が表示されます。選択型でない場合には，この欄は空白です。

2 番目の欄は，モジュールに呼び出しリンケージ情報が含まれているかどうかが示されます。モジュールにリンケージ情報が含まれている場合には，「Lkg」が表示されます。リンケージ情報が含まれていない場合には，この欄は空白です。

3 番目の欄には，モジュールが縮小浮動小数点モデルでコンパイルされたかどうかが示されます。縮小浮動小数点モデルでコンパイルされた場合には，「RFP」が表示されます。縮小浮動小数点モデルでコンパイルされなかった場合には，この欄は空白です。共有イメージの場合には，この欄は表示されません。

4 番目の欄には，全体のプログラム・モードが表示されます。この欄では， Key for Attributes セクションにリストされているいくつかの省略文字が使われます。次に示す例では， Key for Attributes セクションにリストされる可能性があるすべての省略文字を示します。ただし，この例では Creation Date and Creator 欄は省略されています。 Object and Image Synopsis 全体については，以下のマップの例を参照してください。

```
Module/Image    File           Ident            Attributes        Bytes
------------    ----           -----          ----------------    -----
NONE                            V1.0              Lkg               568
                DISK1:[JOE]NONE.OBJ;1
DNORM_CASE                                        Lkg RFP Dnrm      504
                DISK1:[JOE]DENORM_W.OBJ;1
FAST _CASE                                        Lkg RFP Fast      504
                DISK1:[JOE]FAST_W.OBJ;1
NEPCT_CASE                                        Lkg RFP Inex      504
                DISK1:[JOE]INEXACT_W.OBJ;1
SPCL _CASE                                        Lkg RFP Spcl      504
                DISK1:[JOE]SPECIAL_W.OBJ;1
UNDER_CASE                                        Lkg RFP Undr      504
                DISK1:[JOE]UNDERFLOW_W.OBJ;1
DG_FL_CASE                                        Lkg RFP VXfl      504
                DISK1:[JOE]VAXFLOAT_W.OBJ;1
DECC$SHR                      V8.2-00             Lkg                 0
                RESD$:[SYSLIB]DECC$SHR.EXE;1
SYS$PUBLIC_VECTORS            X-2             Sel Lkg                 0
                RESD$:[SYSLIB]SYS$PUBLIC_VECTORS.EXE;1
```

```
 Key for Attributes
+----------------------------------------+
! Sel  - Module was selectively searched  !
! Lkg  - Contains call linkage information !
! RFP  - Conforms to the reduced FP model  !
! VXfl - VAX Float FP model                !
! Dnrm - Denormal IEEE FP model            !
! Fast - Fast IEEE FP model                !
! Inex - Inexact IEEE FP model             !
! Undr - Underflow-to-zero IEEE FP model   !
! Spcl - Special FP model                  !
+----------------------------------------+
```

4 Cluster Synopsis。 Alpha システムで Image Section Synopsis というタイトルが付いていたセクションは， I64 システムでは 2 つのセクション， Cluster Synopsis と Image Segment Synopsis に分割されました。 Cluster Synopsis セクションには，イメージ・セクションが含まれなくなりました。これは， I64 システムでは**セグメント**と呼ばれます。また，共有イメージのイメージ・セクションも表示されなくなりました。これは， VAX マップ・ファイルでは，ベース付きの共有イメージを持つ可能性がある場合には，必ず表示されていました (ベース付きの共有イメージは， Alpha システムや I64 システムでは許されていません)。

5 Cluster。この欄には， Linker によって作成され，使用されるクラスタが，処理された順番で表示されます。

6 Match, Majorid, Minorid。この欄には，バージョン基準が表示されます (存在する場合)。

7 Image Segment Synopsis。このセクションには，作成されたイメージ・セグメントが表示されます。ここには， OpenVMS Alpha の Image Section Synopsis の残りの欄も含まれます。最初の欄 Seg #には，イメージ・セグメントの番号が表示されます。この番号はそのセグメントに適用される再配置で使用されます (再配置でのセグメント番号の表示については，イメージの解析データを参照してください)。 Alpha システムで Protection and Paging というタイトルが付いていたセクションは， I64 システムでは 2 つの欄， Protection と Attributes に分割されました。 Global Section Name 欄は廃止されました。

8 モジュールが/TIE でコンパイルされ，イメージが/NONATIVE だけでリンクされており，イメージに非標準の署名が含まれている場合には， (SHORT で示される) ショート・データ・セグメントの直後に非標準の署名を含む独立したセグメントが表示されます。

9 セクション属性 PIC と NOPIC は I64 システムでは有効な属性ではないので，削除されました。

10 Linker は，共通シンボル，または緩やかな refdef シンボルに記憶域を割り当てます。これは Module/Image ヘッダの下で<Linker>とマークされます。セクション名は常にシンボルの後に付けられます。(このモジュールは，デフォルトのスイッ

チ/EXTERN=RELAXED でコンパイルされており，変数 ITMLST， FILLEN，FILLIM，および IOSB は緩やかな refdef シンボルです。）

11 Linker は，属性 OVR， REL，および GBL を持つセクションの初期化を行なうモジュールがあれば，それに "Initializing Contribution"を付けて表示します。初期化を行なうモジュールを複数指定したエラーの場合には， Linker は複数のセクションに "Initializing Contribution"を付けるので，初期化を行うモジュールが複数あるインスタンスのデバッグが容易になります。

12 Linker は，トランポリン (遠くへ分岐する命令) または別のセグメント (イメージの中または外) へ分岐するコードを含むコード・セグメントへのコントリビューションにマークを付加します。その場合には， Module/Image ヘッダの下に<Linker>を表示します。

13 外部シンボルの表示が Alpha マップとは異なる表示になりました。 Alpha システムでは，再配置可能と外部を意味した，プレフィックスまたはサフィックス RX が使われましたが， I64 システムの Linker では外部シンボルが再配置可能かどうかがわからないため，プレフィックスまたはサフィックスは X (外部) に変更されました。

14 Keys for Special Characters。特殊文字のキーが，次のように変更されました。

- I64 システムでは，コード・アドレスに対して特殊文字 C が表示されます。 Linker によって割り当てられた関数記述子を関数が持たない場合には，その値はコード・アドレスになります。
- OpenVMS Alpha では，ユニバーサル・シンボルは，内部の値とともに 1 回だけ表示されました。マップには，外部のユニバーサル値 (I64 システムではシンボル・ベクタへのインデックス) は表示されませんでした。今回 I64 システムでのユニバーサル・シンボルは，<Linker Option>で定義されるサフィックス (U) を付けて 1 回だけ表示され，外部の値であることが表示されるようになりました。ただし，プレフィックスまたはサフィックス R が付けられた場合には，内部の値であることを示しています。別名を持ったシンボル・ベクタがある場合には，別名にはユニバーサル値が表示され，内部名は内部の値とともに表示されます。 OpenVMS Alpha のプレフィックスおよびサフィックス A と I (それぞれ，別名と内部の意味) は廃止されました。

  たとえば， symbol_vector=(getjpi/internal_getjpi=procedure) と指定すると，次の出力が表示されます。

```
00000000        GETJPI (U)
00050098      R-INTERNAL_GETJPI
```

- UxWk で示される UNIX の弱いシンボルが， OpenVMS に新しく追加されました。これは， OpenVMS の弱いシンボルに似ていますが， UNIX の弱い定義を持った複数のシンボルは，複数のモジュールをリンクするときに，多重定義エラーとならずに処理できます。 UNIX の弱いシンボルは，現在は C++ コンパイラによって生成されます。

15 Quota Usage。 I64 Linker によって使われる制限値を追跡できるように Link Run Statistics セクションに， Quota Usage というタイトルの新しいセクションが追加されました。制限値の問題が発生した場合， Linker はそれを回避できます。ただし，その場合には Linker は Quota Usage セクションに特殊なメッセージを表示し，性能を向上させるための最適な制限値の増加方法を示します。例を次に示します。

```
Performance of this link operation could be improved by increasing quotas
   Quota related to status return:  %SYSTEM-SECTBLFUL, process or global
   section table is full
2688 extra file I/O operations performed due to current process quota(s)
36 performed on object files; 2652 performed on library files
```

第2部
OpenVMSの英語版ドキュメント

注意

ここでは OpenVMS の英語版ドキュメントについて説明します。日本語ドキュメントについては『日本語 OpenVMS リリース・ノート』で説明しています。

8

OpenVMS の英語版ドキュメントの概要

表 8–1 に OpenVMS Version 8.2 の英語版ドキュメント・セットにおける変更点を示します。本リリースでは， OpenVMS 英語版ドキュメント・セットに 2 つの新しいマニュアルが追加されており， 3 つのマニュアルがアーカイブ扱いとなっています。また，以下のマニュアルが再びハードコピー版として提供されています。

- 『HP OpenVMS Management Station Overview and Release Notes』
- 『COM, Registry, and Events for OpenVMS Developer's Guide』

表 8–1 OpenVMS Version 8.2 における英語版ドキュメント・セットの変更点

新しいマニュアル	
『Porting Aplications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers』(翻訳版は『HP OpenVMS Alpha から OpenVMS I64 へのアプリケーション・ポーティング・ガイド』)	このマニュアルは， OpenVMS Alpha から OpenVMS I64 へのアプリケーションのポーティングを検討しているアプリケーション・プログラマのためのドキュメントです。
『HP MACRO Compiler Porting and User's Guide』	このドキュメント自体は改訂版のドキュメントですが，OpenVMS のドキュメント・セットで提供するのは初めてになります。
アーカイブ・マニュアル	
『VAX MACRO and Instruction Set Reference Manual』	このマニュアルは，VAX MACRO命令セットおよびアセンブラの機能について説明しています。 Version 8.2 以降，このマニュアルはアーカイブ扱いとなります。下記の URL の OpenVMS のドキュメント Web サイトにアーカイブ・ドキュメントへのリンクがあります。 http://www.hp.com/go/openvms/doc/
『OpenVMS VAX RTL Mathematics (MTH$) Manual』	このマニュアルは， OpenVMS VAX ランタイム・ライブラリの MTH$機能に含まれている算術ルーチンについて説明しています。 V8.2 以降，このマニュアルはアーカイブ扱いとなります。下記の URL の OpenVMS のドキュメント Web サイトにアーカイブ・ドキュメントへのリンクがあります。 http://www.hp.com/go/openvms/doc/
『OpenVMS VAX System Dump Analyzer Utility Manual』	このマニュアルは，システムのエラーを調査し実行中のシステムをテストするための System Dump Analyzer (SDA) の使用方法について説明しています。 V8.2 以降，このマニュアルはアーカイブ扱いとなります。下記の URL の OpenVMS のドキュメント Web サイトにアーカイブ・ドキュメントへのリンクがあります。 http://www.hp.com/go/openvms/doc/

9

OpenVMS の英語版ドキュメント (印刷およびオンライン)

OpenVMS のドキュメントは次の形式で提供されます。

- 印刷物

  印刷されたドキュメントが必要な場合は，ほとんどの OpenVMS ドキュメントをこの形式で購入できます。個々の印刷マニュアルを個別に購入することはできませんが， OpenVMS 印刷ドキュメント・キットにはすべてのマニュアルが含まれています。ただし『Porting Applications from HP OpenVMS Alpha to OpenVMS I64 for Integrity Servers』の印刷版は単独で購入可能です。

- CD に収録されたオンライン・ドキュメント

  すべての OpenVMS ドキュメントは CD に収録されたオンライン形式で提供され，多くの関連製品に関するドキュメントも含まれています。ドキュメンテーション CD は OpenVMS メディア・キットに同梱されています。

- OpenVMS Web サイトで提供されるオンライン・ドキュメント

  OpenVMS のドキュメントは，アーカイブされたドキュメントを含めて，OpenVMS Web サイトで閲覧できます。

- オンライン・ヘルプ

  タスク関連情報が必要な場合は， OpenVMS コマンド，ユーティリティ，システム・ルーチンに関するオンライン・ヘルプを簡単に表示できます。

ここでは， OpenVMS ドキュメントが提供される形式と，各形式で提供されるドキュメントの名称を示します。

9.1 印刷ドキュメント

OpenVMS メディア・キット (インストレーション・キット) には，一部のドキュメントの印刷版が同梱されています。それ以外の印刷ドキュメントは別売のドキュメント・キットで注文可能です。ここでは， OpenVMS の印刷ドキュメントについて次のカテゴリに分類して説明します。

- メディア・キット
- ドキュメンテーション・セット: 基本セットとフル・セット，およびオペレーティング環境拡張
- システム統合製品

- アーカイブされたドキュメント

9.1.1 OpenVMS メディア・キットのドキュメント

OpenVMS Alpha および OpenVMS I64 システムのメディア・キット (インストレーション・キット) には， OpenVMS オペレーティング・システムの最新のバージョンを使用するのに必要なドキュメントが同梱されています。表 9–1 には， OpenVMS メディア・キットに含まれているドキュメントを示しています。提供されるドキュメントは，新規カスタマであるのか，サービス・カスタマであるのかに応じて異なります。新規カスタマにはすべてのドキュメントが提供されます。サービス・カスタマには，前回のリリース以降に更新されたドキュメントと新しいドキュメントだけが提供されます。

注意

『OpenVMS License Management Utility Manual』，『Guide to HP OpenVMS Version 8.2 Media』，および『HP OpenVMS Version 8.2 Upgrade and Installation Manual』は，メディア・キットでのみ提供されます。第 9.1.2 項で説明する OpenVMS フル・ドキュメント・セットには含まれていません。

表 9–1 OpenVMS メディア・キットに含まれるドキュメント

ドキュメント	
『OpenVMS License Management Utility Manual』	AA-PVXUG-TK
『Guide to HP OpenVMS Version 8.2 Media』	BA322-90001
『HP OpenVMS Version 8.2 New Features and Documentation Overview』	BA322-90003
『HP OpenVMS Version 8.2 Upgrade and Installation Manual』	BA322-90002
『HP OpenVMS Version 8.2 Release Notes』	BA322-90004

9.1.2 OpenVMS ドキュメンテーション・セット

OpenVMS のドキュメントは次のドキュメンテーション・セットで提供されます。

ドキュメンテーション・セット	説明	Alpha パーツ番号	I64 パーツ番号
フル・セット	主要なすべての OpenVMS リソースの広範囲にわたる説明情報が必要なユーザを対象にしている。すべての OpenVMS ドキュメントが 1 つのセットとして提供される。基本ドキュメンテーション・セットも含まれている。	QA-001AA-GZ.8.2	BA554MN

ドキュメンテーション・セット	説明	Alpha パーツ番号	I64 パーツ番号
基本セット	フル・ドキュメンテーション・セットの一部。小規模なスタンドアロン・システムのシステム管理者や一般ユーザを対象にしている。最も一般的に使用される OpenVMS のドキュメントが含まれている。	QA-09SAA-GZ.8.2	BA555MN

OpenVMS Alpha および OpenVMS I64 システムで，ドキュメント・セットの内容はほぼ共通です。 OpenVMS Alpha ドキュメント・セットと OpenVMS I64 ドキュメント・セットに含まれるドキュメントは 1 冊の例外を除き同一です。 OpenVMS Alpha ドキュメント・セットには， Alpha 専用ドキュメントの『COM, Registry, and Events for OpenVMS Developer's Guide』が含まれています。

表 9–2 は， OpenVMS のフル・ドキュメンテーション・セットまたは基本ドキュメンテーション・セットに含まれているドキュメントを示しています。各ドキュメントの詳細については，第 10.2 節を参照してください。

表 9–2 OpenVMS フル・ドキュメンテーション・セット (QA-001AA-GZ.8.2/BA554MN)

ドキュメント	
OpenVMS 基本ドキュメンテーション・セット	**QA-09SAA-GZ.8.2 /BA555MN**
『OpenVMS DCL Dictionary:A-M』[1]	AA-PV5KK-TK
『OpenVMS DCL Dictionary:N-Z』[1]	AA-PV5LK-TK
『HP OpenVMS Guide to System Security』[1]	AA-Q2HLG-TE
『OpenVMS System Management Utilities Reference Manual:A-M』[1]	AA-PV5PJ-TK
『OpenVMS System Management Utilities Reference Manual:N-Z』[1]	AA-PV5QJ-TK
『OpenVMS System Manager's Manual, Volume 1:Essentials』[1]	AA-PV5MJ-TK
『OpenVMS System Manager's Manual, Volume 2:Tuning, Monitoring, and Complex Systems』[1]	AA-PV5NJ-TK
『OpenVMS User's Manual』[1]	AA-PV5JG-TK
『OpenVMS Alpha V 8.2 New Features and Documentation Overview』[2]	BA322-90003
『OpenVMS Alpha V 8.2 Release Notes』[2]	BA322-90004
フル・ドキュメンテーション・セットに含まれている追加ドキュメント	**QA-001AA-GZ.8.2**
『Availability Manager User's Guide』[1]	AA-RNSJD-TE
『COM, Registry, and Events for OpenVMS Developer's Guide』[3]	AA-RSCWC-TE
『Compaq C Run-Time Library Reference Manual for OpenVMS Systems』[2]	AA-RSMUC-TE
『HP C Run-Time Library Utilities Reference Manual』	AA-R238C-TE

[1]バージョン 8.2 で変更されたドキュメント

[2]バージョン 8.2 で新規に提供されるドキュメント

[3]Alpha のみ - QA-001AA-GZ.8.2 に含まれる。

(次ページに続く)

表 9–2 (続き) OpenVMS フル・ドキュメンテーション・セット (QA-001AA-GZ.8.2/BA554MN)

ドキュメント	
フル・ドキュメンテーション・セットに含まれている追加ドキュメント	QA-001AA-GZ.8.2
『Compaq Portable Mathematics Library』	AA-PV6VE-TE
『DECamds User's Guide』	AA-Q3JSE-TE
『DEC Text Processing Utility Reference Manual』	AA-PWCCD-TE
『Extensible Versatile Editor Reference Manual』	AA-PWCDD-TE
『Guidelines for OpenVMS Cluster Configurations』[1]	AA-Q28LH-TK
『Guide to Creating OpenVMS Modular Procedures』	AA-PV6AD-TK
『Guide to OpenVMS File Applications』[1]	AA-PV6PE-TK
『Guide to the POSIX Threads Library』	AA-QSBPD-TE
『Guide to the DEC Text Processing Utility』	AA-PWCBD-TE
『Open Source Security for OpenVMS Alpha, Volume 1: Common Data Security Architecture』[2]	AA-RSCUB-TE
『HP Open Source Security for OpenVMS,Volume 2: HP SSL for OpenVMS』[2]	AA-RSCVC-TE
『HP Open Source Security for OpenVMS, Volume 3: Kerberos』[1]	AA-RUEBB-TE
『OpenVMS Alpha Partitioning and Galaxy Guide』[1]	AA-REZQD-TE
『OpenVMS Guide to Upgrading Privileged-Code Applications』	AA-QSBGE-TE
『OpenVMS System Analysis Tools Manual』[1]	AA-REZTE-TE
『OpenVMS Calling Standard』	AA-QSBBE-TE
『OpenVMS Cluster Systems』[1]	AA-PV5WF-TK
『OpenVMS Command Definition, Librarian, and Message Utilities Manual』	AA-QSBDE-TE
『OpenVMS Debugger Manual』	AA-QSBJE-TE
『OpenVMS Delta/XDelta Debugger Manual』	AA-PWCAE-TE
『OpenVMS I/O User's Reference Manual』[1]	AA-PV6SG-TK
『OpenVMS Linker Utility Manual』	AA-PV6CE-TK
『OpenVMS MACRO-32 Porting and User's Guide』	AA-PV64E-TE
『OpenVMS Management Station Overview and Release Notes』	AA-QJGCG-TE
『OpenVMS Performance Management』	AA-R237C-TE
『Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers』[2]	BA442-90001
『OpenVMS Programming Concepts Manual, Volume I』[1]	AA-RNSHD-TK
『OpenVMS Programming Concepts Manual, Volume II』[1]	AA-PV67H-TK
)	
『OpenVMS Record Management Services Reference Manual』[1]	AA-PV6RE-TK
『OpenVMS Record Management Utilities Reference Manual』	AA-PV6QD-TK
『OpenVMS RTL General Purpose (OTS$) Manual』	AA-PV6HE-TK
『OpenVMS RTL Library (LIB$) Manual』	AA-QSBHE-TE

[1]バージョン 8.2 で変更されたドキュメント

[2]バージョン 8.2 で新規に提供されるドキュメント

(次ページに続く)

表 9–2 (続き) OpenVMS フル・ドキュメンテーション・セット (QA-001AA-GZ.8.2/BA554MN)

ドキュメント	
フル・ドキュメンテーション・セットに含まれている追加ドキュメント	QA-001AA-GZ.8.2
『OpenVMS RTL Screen Management (SMG$) Manual』	AA-PV6LD-TK
『OpenVMS RTL String Manipulation (STR$) Manual』	AA-PV6MD-TK
『OpenVMS System Messages: Companion Guide for Help Message Users』	AA-PV5TD-TK
『OpenVMS System Services Reference Manual: A–GETUAI』[1]	AA-QSBMG-TE
『OpenVMS System Services Reference Manual: GETUTC–Z』[1]	AA-QSBNG-TE
『OpenVMS Utility Routines Manual』[1]	AA-PV6EF-TK
『OpenVMS VAX RTL Mathematics (MTH$) Manual』	AA-PVXJD-TE
『OpenVMS VAX System Dump Analyzer Utility Manual』	AA-PV6TD-TE
『POLYCENTER Software Installation Utility Developer's Guide』	AA-Q28MF-TK
『VAX MACRO and Instruction Set Reference Manual』	AA-PS6GD-TE
『Volume Shadowing for OpenVMS』[1]	AA-PVXMK-TE

[1]バージョン 8.2 で変更されたドキュメント

9.1.3 オペレーティング環境拡張ドキュメント・セット (I64 のみ)

オペレーティング環境拡張ドキュメント・セットには，OpenVMS OE に含まれる製品のマニュアルが含まれています。このドキュメント・セットに含まれるドキュメントの一覧は第 10.5 節を参照してください。

9.1.4 システム統合製品のドキュメント

システム統合製品 (SIP) は，OpenVMS ソフトウェアに含まれていますが，これらの製品を使用するには個別のライセンスを購入する必要があります。表 9–3 は，システム統合製品に関連するドキュメントを示しています。

表 9–3 システム統合製品のドキュメント

システム統合製品	関連ドキュメント
HP Galaxy Software Architecture on OpenVMS Alpha	このドキュメントは OpenVMS フル・ドキュメンテーション・セットに含まれている。
OpenVMS Clusters	OpenVMS Cluster ドキュメントは OpenVMS フル・ドキュメンテーション・セットに含まれている。
RMS Journaling for OpenVMS	RMS Journaling for OpenVMS のマニュアルは，下記の URL の OpenVMS ドキュメント Web サイトで HTML 形式で提供されています。http://www.hp.com/go/openvms/doc
Volume Shadowing for OpenVMS	このドキュメントは OpenVMS フル・ドキュメンテーション・セットに含まれている。

9.1.5 アーカイブされた OpenVMS ドキュメント

OpenVMS では， OpenVMS オペレーティング・システムのドキュメントを継続的に更新，変更，拡張しています。必要に応じてドキュメントはアーカイブに移されます。アーカイブされたドキュメントは，次の Web サイトからアクセスすることができます。

```
http://www.hp.com/go/openvms/doc
```

アーカイブされた OpenVMS のドキュメントの一覧については，第 10.6 節を参照してください。

9.2 OpenVMS ドキュメントの開発ツール

OpenVMS ドキュメント・チームは SGML (Standard Generalized Markup Language) ベースのツールを使用してマニュアルを作成しています。 SGML は業界標準の記述言語であり，お客様にとっても OpenVMS ドキュメンテーションにとっても多くの利点があります。下記の Version 8.2 ドキュメントはこの新しいツールを使用して作成されています。

SGML で作成されたドキュメントとその他のドキュメントでは見た目が異なります。 HTML， PDF，および印刷ドキュメントのすべてでこの違いが見られますが，これは使用したツールの違いによるものです。

- 『HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS』
- 『HP Open Source Security for OpenVMS, Volume 3: Kerberos』
- 『HP OpenVMS Version 8.2 Upgrade and Installation Manual』
- 『Guide to HP OpenVMS Version 8.2 Media』
- 『OpenVMS I/O User's Reference Manual』
- 『HP OpenVMS System Manager's Manual, Volume 1:Essentials』
- 『HP OpenVMS System Manager's Manual, Volume 2:Turning,Monitoring,and Complex Systems』

9.3 CD に収録されているオンライン・ドキュメント

OpenVMS オペレーティング・システムおよび多くの関連製品のオンライン・ドキュメントは， 2 枚の CD に収録されて提供されます。 1 枚は ISO9660 Level 2 CD で，Windows および Macintosh システムで読むことができます。もう 1 枚はFiles–11 CD で， OpenVMS システムで読むことができます。 2 枚の CD の内容は同じですが，次の点が異なっています。

- ISO9660 Level 2 の CD-ROM には， Adobe Acrobat Reader バージョン 5.0.5 が格納されています。
- Files–11の CD-ROM には， HP Secure Web Browser for OpenVMS Alpha (Mozilla ベース) と，ブラウザを起動するためのコマンド・プロシージャが格納されています。

9.3.1 オンライン形式

ドキュメンテーション CD-ROM には，ドキュメントがさまざまな形式で収録されています。

ドキュメント	提供される形式
現在のバージョンの OpenVMS のドキュメント	HTML, PDF
『HP OpenVMS Version 8.2 Upgrade and Installation』	HTML, PDF
『HP OpenVMS Version 8.2 Release Notes』	HTML, PDF
『HP OpenVMS V 8.2 New Features and Documentation Overview』	HTML, PDF
レイヤード製品のドキュメント	HTML, PDF

以前のバージョンで提供していた Bookreader ファイルは提供されなくなりました。

9.3.2 PDF Reader

PDF ファイルを表示するために， Adobe Acrobat Reader が提供されます。この自己解凍形式のファイルは， Windows が稼動するコンピュータにインストールできます。 ISO9660 Level 2 の CD に収録されています。

ドキュメンテーション CD のドキュメントにアクセスする方法と， PDF Reader の詳細については，『HP OpenVMS Version 8.2 Upgrade and Installation Manual』を参照してください。

9.4 OpenVMS Web サイトで提供されるオンライン・ドキュメント

次の OpenVMS Web サイトでは， OpenVMS のドキュメントがさまざまなオンライン形式で提供されています。

```
http://www.hp.com/gp/openvms/doc
```

このサイトには， OpenVMS フル・ドキュメンテーション・セットに含まれるドキュメントの最新のバージョン，および特定のレイヤード製品のドキュメントへのリンクが掲載されています。

9.5 オンライン・ヘルプ

OpenVMS オペレーティング・システムでは，フル・ドキュメンテーション・セットに説明されているコマンド，ユーティリティ，システム・ルーチンのオンライン・ヘルプが提供されます。

システム・メッセージのオンライン説明にアクセスするには，ヘルプ・メッセージ機能を使用します。さらに，ヘルプ・メッセージ・データベースに書き込んだメッセージ・ドキュメンテーションなど，独自のソース・ファイルを追加することができます。『OpenVMS System Messages: Companion Guide for Help Message Users』では，ヘルプ・メッセージ機能の使い方が説明されています。また，次のように入力して，ヘルプ・メッセージに関する DCL ヘルプにアクセスすることもできます。

```
$ HELP HELP/MESSAGE
```

OpenVMS ユーティリティ・ルーチンに関する参照情報は，オンライン・ヘルプで提供されるようになりました。

10

OpenVMS のドキュメントの説明

この章では，次の OpenVMS ドキュメントについて簡単に説明します。

- OpenVMS メディア・キットに含まれるドキュメント (第 10.1 節)
- OpenVMS ベース・ドキュメンテーション・セットおよびフル・ドキュメンテーション・セットに含まれるドキュメント (第 10.2 節と第 10.3 節)
- RMS Journaling のドキュメント (第 10.4 節)
- OpenVMS I64 OE 拡張キットに含まれているドキュメント
- アーカイブされたドキュメント (第 10.6 節)

10.1 OpenVMS メディア・キットに含まれるドキュメント

『Guide to HP OpenVMS Version 8.2 Media』
（翻訳版は『HP OpenVMS Version 8.2 CD-ROM/DVD ユーザーズ・ガイド』）
OpenVMS オペレーティング・システムおよびドキュメント CD に関する情報を提供します。 OpenVMS Alpha および OpenVMS I64 バージョン8.2メディア・キットの内容を示し，インストール情報へのポインタを示し，ドキュメント CD-ROM に収録されているドキュメントへのアクセス方法を示します。

『OpenVMS License Management Utility Manual』
OpenVMS のライセンス管理ツールである LMF (License Management Facility) について説明します。 LMF には， License Management ユーティリティ (LICENSE) と，ソフトウェア・ライセンスの登録，管理，追跡に使用するコマンド・プロシージャ VMSLICENSE.COM が含まれています。

『HP OpenVMS Version 8.2 Upgrade and Installation Manual』
OpenVMS Alpha および OpenVMS I64 オペレーティング・システムのインストレーション手順を示します。ブート，シャットダウン，バックアップ，ライセンス・プロシージャに関する情報が記載されています。

『OpenVMS Version 8.2 New Features and Documentation Overview』
（翻訳版は『HP OpenVMS Version 8.2 新機能説明書』）
OpenVMS I64 および Alpha オペレーティング・システム 8.2 リリースの新しいコンポーネントと拡張されたコンポーネントについて説明します。バージョン 8.2 で変更された OpenVMS ドキュメントについて説明し， OpenVMS ドキュメントの印刷形式およびオンライン形式についても説明します。

『OpenVMS Version 8.2 Release Notes』
（翻訳版は『HP OpenVMS Version 8.2 リリース・ノート[翻訳版]』）
ソフトウェアの変更点，インストール，アップグレード，互換性情報，ソフトウェアの新しい問題点と制約事項および既存の問題点と制約事項，ソフトウェアとドキュメントの修正について説明します。

注意

日本語 OpenVMS のメディア・キット (インストレーション・キット) には，上記の他に『日本語 OpenVMS リリース・ノート』および『日本語 OpenVMS インストレーション・ガイド』も含まれます。

10.2 OpenVMS 基本ドキュメント・セットのドキュメント

『OpenVMS DCL Dictinary』
（翻訳版は『OpenVMS DCL ディクショナリ』）
DCL (DIGITAL Command Language) について説明し，すべての DCL コマンドとレキシカル関数の詳細な参照情報と例をアルファベット順に示します。このドキュメントは 2 分冊になっています。

『HP OpenVMS Guide to System Security』
(翻訳版は『OpenVMS システム・セキュリティ・ガイド』)
OpenVMS オペレーティング・システムで提供されるセキュリティ機能について説明します。具体的なセキュリティ・ニーズを示し，それぞれの状況に応じた各機能の目的と適切な応用を示します。

『OpenVMS System Management Utilities Reference Manual』
（翻訳版は『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル』）
システム管理作業を実行するためのユーティリティや，システムへのアクセスとリソースの制御と監視に使用するツールについて参照情報を示します。 AUTOGEN コマンド・プロシージャについても説明します。このドキュメントは 2 分冊になっています。

『OpenVMS System Manager's Manual, Volume 1: Essentials』
（翻訳版は『OpenVMS システム管理者マニュアル (上巻)』）
システムの起動，ソフトウェアのインストール，プリント・キューとバッチ・キューの設定など，日常的に行う操作の設定と管理の方法について説明します。また，日常的に行うディスク操作と磁気テープ操作についても説明します。

『OpenVMS System Manage's Manual, Volume 2: Tuning, Monitoring, and Complex Systems』
（翻訳版は『OpenVMS システム管理者マニュアル (下巻)』）
ネットワークの構成と制御，システムの監視，システム・パラメータの管理の方法について説明します。また， OpenVMS Cluster システム，ネットワーク環境，DECdtm 機能についても説明します。

『OpenVMS User's Manual』
（翻訳版は『OpenVMS ユーザーズ・マニュアル』）
オペレーティング・システムの概要を示し，基本的な概念，タスク情報，日常のコンピューティング・タスクを実行するための参照情報も示します。ファイルとディレクトリの操作方法についても説明します。また，次の追加トピックも含まれています。

- Mail ユーティリティと Phone ユーティリティによるメッセージの送信
- Sort/Merge ユーティリティの使用
- 論理名とシンボルの使用
- コマンド・プロシージャの作成
- EVE および EDT テキスト・エディタによるファイルの編集

『OpenVMS Version 8.2 New Features and Documentation Overview』
（翻訳版は『HP OpenVMS Version 8.2 新機能説明書』）
バージョン 8.2 リリースの新しいコンポーネントと拡張されたコンポーネントについて説明します。バージョン 8.2 で変更された OpenVMS ドキュメントについて説明し， OpenVMS ドキュメントの印刷形式およびオンライン形式についても説明します。

『OpenVMS Version 8.2 Release Notes』
（翻訳版は『HP OpenVMS Version 8.2 リリース・ノート[翻訳版]』）
ソフトウェアの変更点，インストール，アップグレード，互換性情報，ソフトウェアの新しい問題点と制約事項および既存の問題点と制約事項，ソフトウェアとドキュメントの修正について説明します。

10.3 OpenVMS フル・ドキュメンテーション・セットの追加ドキュメント

『Availability Manager User's Guide』
OpenVMS Alpha または Windows ノードから HP Availability Manager システム管理ツールを使用して，拡張ローカル・エリア・ネットワーク (LAN) で 1 つ以上の OpenVMS ノードを監視する方法と，詳細な分析のために特定のノードまたはプロセスを監視する方法について説明します。

『COM, Registry, and Events for OpenVMS Developer's Guide』
OpenVMS 環境と Windows NT 環境の間で簡単に移行できるアプリケーションを開発するプログラマを対象にしたドキュメントです。既存の OpenVMS アプリケーションやデータをカプセル化する場合や， OpenVMS システム用に新しい COM アプリケーションを開発する場合は，このドキュメントを参照してください。また，OpenVMS Registry を使用して OpenVMS システムだけに関する情報を格納する場合や， OpenVMS レジストリ情報と Windows NT レジストリ情報の両方を格納するための共用の格納場所として OpenVMS Registry を使用する場合も，このドキュメントを参照してください。このドキュメントは，以前は『OpenVMS Connectivity Developer Guide』（翻訳版は『OpenVMS コネクティビティ開発者ガイド』）という名称でオンラインで提供されていました。

『HP C Run-Time Library Reference Manual for OpenVMS Systems』
（翻訳版は『HP C ランタイム・ライブラリ・リファレンス・マニュアル』）
I/O 操作，文字および文字列操作，算術演算，エラー検出，サブプロセスの生成，システム・アクセス，画面管理などを実行する HP C RTL の関数とマクロに関する参照情報を示します。オペレーティング・システム間での移植に関する問題点， TCP/IP プロトコル用にインターネット・アプリケーション・プログラムを作成するために使用する HP C for OpenVMS ソケット・ルーチンについても説明します。

『Compaq C Run-Time Library Uilities Reference Manual』
（翻訳版は『Compaq C 国際化ユーティリティ・リファレンス・マニュアル』）
国際化ソフトウェア・アプリケーションでローカリゼーションとタイム・ゾーン・データを管理するための C ランタイム・ライブラリ・ユーティリティの詳細な使い方と参照情報を示します。

『Compaq Portable Mathematics Library』
DPML (Compaq Portable Mathematics Library) の算術演算ルーチンについて説明します。これらのルーチンは OpenVMS Alpha システムでのみ提供されます。 VAX プログラマは『OpenVMS VAX RTL Mathematics (MTH$) Manual』を参照してください。

『DECamds User's Guide』
DECamds ソフトウェアのインストールと使用方法について説明します。 DECamds は， OpenVMS システムおよび OpenVMS Cluster 環境でイベントの監視，診断，追跡を行うためのシステム管理ツールです。

『DEC Text Processing Utility Reference Manual』
DECTPU (DEC Text Processing Utility) について説明し， DECTPU に対する EDT キーパッド・エミュレータ・インタフェースに関する参照情報を示します。

『Extensible Versatile Editor Reference Manual』
EVE テキスト・エディタに関するコマンド参照情報を示します。また， EDT コマンドと EVE コマンドの間の相互参照も示します。

『Guidelines for OpenVMS Cluster Configurations』
（翻訳版は『OpenVMS Cluster 構成ガイド』）
このドキュメントでは，システム，インターコネクト，ストレージ・デバイス，ソフトウェアを選択するのに役立つ情報を示します。高い可用性，拡張性，パフォーマンス，容易なシステム管理を実現するためにこれらのコンポーネントを構成するのに役立ちます。このドキュメントでは， OpenVMS Cluster システムで SCSI および Fibre Channel を使用する方法についても詳しく説明しています。

『Guide to Creating OpenVMS Modular Procedures』
プログラムを複数のモジュールに分割し，各モジュールを個別のプロシージャとしてコーディングすることにより，複雑なプログラミング作業を実行する方法について説明します。

『Guide to OpenVMS File Applications』
RMS (Record Management Services) を使用して，効率のよいデータ・ファイルの設計，作成，管理を行うためのガイドラインを示します。このドキュメントは， RMS ファイルを使用するプログラム，特にパフォーマンスが重要視されるプログラムを取り扱うアプリケーション・プログラマおよび設計者を対象にしています。

『Guide to the POSIX Threads Library』
弊社のマルチスレッド・ランタイム・ライブラリである POSIX Threads Library (以前の名称は DECthreads) パッケージについて説明します。このパッケージに含まれているルーチンは，1 つのプロセスで提供されるアドレス空間内で複数の実行スレッドを作成し，制御することができます。このドキュメントでは使い方のヒントと参照情報の両方を示し，3 つのインタフェースについて説明しています。3 つのインタフェースとは，IEEE POSIX 1003.1c 標準規格に準拠したルーチン (pthreadと呼びます)，非スレッド・アプリケーションでスレッド関連サービスを提供するルーチン (スレッド独立サービスまたはtisと呼びます)，上位互換性のある安定したインタフェースを提供する弊社固有のルーチン (cmaと呼びます) です。

『Guide to the DEC Text Processing Utility』
DECTPU プログラムの開発の概要について説明します。

『Open Source Security for OpenVMS Alpha, Volume 1: Common Data Security Architecture』
CDSA (Common Data Security Architecture) を使用して，プログラムにセキュリティ機能を追加するアプリケーション開発者を対象にしています。CDSA について説明し，インストールと初期化について説明し，サンプル・プログラムも提供します。CDSA アプリケーション・プログラミング・インタフェース・モジュールが含まれています。

『HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS』
HP SSL (HP Secure Sockets Layer) for OpenVMS Alpha で OpenVMS アプリケーションへの通信リンクを保護することを検討しているアプリケーション開発者を対象にしています。インストールの方法とリリース・ノートを示し，サンプル・プログラムを提供します。OpenSSL アプリケーション・プログラミング・インタフェース・モジュールのプログラミング情報と参照情報が示されています。

『OpenVMS Alpha Partitioning and Galaxy Guide』
（翻訳版は『OpenVMS Alpha パーティショニングおよび Galaxy ガイド』）
OpenVMS Alpha バージョン 7.3-2 で提供されるすべての OpenVMS Galaxy 機能の使い方について詳しく説明します。AlphaServer 8400，8200，4100 システムで OpenVMS Galaxy コンピューティング環境を作成，管理，使用する手順も示します。

『OpenVMS Guide to Upgrading Privileged-Code Applications』
OpenVMS Alpha バージョン 7.0 で OpenVMS Alpha の 64 ビット仮想アドレッシングおよびカーネル・スレッドがサポートされるようになった結果，Alpha の特権付きコード・アプリケーションおよびデバイス・ドライバに影響を与える可能性のある OpenVMS Alpha バージョン 7.0 の変更点について説明します。

OpenVMS Alpha バージョン 7.0 より前のバージョンで作成された特権付きコード・アプリケーションは，このガイドの説明に従ってソース・コードを変更する必要があります。

『OpenVMS System Analysis Tools Manual』
次のシステム分析ツールについて詳しく説明します。また， DOSD (dump off system disk) 機能と DELTA/XDELTA デバッガの概要も示します。

- System Dump Analysis (SDA)
- System code debugger (SCD)
- System dump debugger (SDD)
- Watchpoint ユーティリティ

このドキュメントは，システム障害の原因を調べ，デバイス・ドライバなどのカーネル・モード・コードをデバッグしなければならないシステム・プログラマを対象しています。

『OpenVMS Calling Standard』
OpenVMS オペレーティング・システムの呼び出し標準規約について説明します。

『OpenVMS Cluster Systems』
（翻訳版は『OpenVMS Cluster システム』）
OpenVMS Cluster システムの構成と管理の手順およびガイドラインについて説明します。また，クラスタに接続されたシステムで高い可用性，構築ブロックの拡張，統一されたシステム管理を実現する方法についても説明します。

『OpenVMS Command Definition, Librarian, and Message Utilities Manual』
次のユーティリティについて説明し，参照情報も示します。

- Command Definition ユーティリティ
- Librarian ユーティリティ
- Message ユーティリティ

『OpenVMS Debugger Manual』
（翻訳版は『OpenVMS デバッガ説明書』）
プログラマを対象に OpenVMS Debugger の機能について説明します。

『OpenVMS Delta/XDelta Debugger Manual』
特権付きプロセッサ・モードまたは引き上げられた割り込み優先順位レベルで動作するプログラムをデバッグするために使用する Delta/XDelta ユーティリティについて説明します。

『OpenVMS I/O User's Reference Manual』
オペレーティング・システムに付属しているデバイス・ドライバを使用して，システム・プログラマが I/O 操作をプログラミングするのに必要な情報を示します。

『OpenVMS Linker Utility Manual』
Linker ユーティリティを使用して， OpenVMS システムで動作するイメージを作成する方法について説明します。また，リンク修飾子とリンク・オプションを使用してリンク操作を制御する方法についても説明します。

『HP OpenVMS MACRO Compiler Porting and User's Guide』
MACRO-32 コンパイラの機能を使用して，既存のVAX MACROアセンブリ言語コードを OpenVMS Alpha システムに移植する方法について説明します。既存の OpenVMS Alpha のコードを OpenVMS I64 システムへ移植する方法についても説明してます。また，コンパイラの 64 ビット・アドレッシングのサポート機能を使用する方法についても説明します。

『OpenVMS Management Station Overview and Release Notes』
OpenVMS Management Station の概要とリリース・ノートを示し，このソフトウェアの使い方の概要も示します。 OpenVMS Management Station は， OpenVMS システムでユーザ・アカウントやプリンタの管理作業を行うシステム管理者やその他の人を対象した， Microsoft Windows ベースの強力な管理ツールです。

『OpenVMS Performance Management』
OpenVMS システムでパフォーマンスを最適化するために使用する手法について説明します。

『Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers』
（翻訳版は『HP OpenVMS Alpha から OpenVMS I64 へのアプリケーション・ポーティング・ガイド』）
HP OpenVMS Alpha から HP OpenVMS I64 へ移行しようとしているアプリケーション開発者に移行計画の枠組みを提供します。

『OpenVMS Programming Concepts Manual』
プロセスの生成，カーネル・スレッドとカーネル・スレッド・プロセス構造，プロセス間通信，プロセス制御，データの共用，条件処理， AST などの概念について説明します。この 2 分冊のドキュメントでは，システム・サービス，ユーティリティ・ルーチン，ランタイム・ライブラリ (RTL) ルーチンを使用して， OpenVMS の機能を利用する方法を説明します。

『OpenVMS Record Management Services Reference Manual』
RMS データ・ファイルを使用するすべてのプログラマを対象に，参照情報と使用方法を示します。

『OpenVMS Record Management Utilities Reference Manual』
次の RMS ユーティリティに関する説明と参照情報を示します。

- Analyze/RMS_File ユーティリティ
- Convert and Convert/Reclaim ユーティリティ
- File Definition Language 機能

『OpenVMS RTL General Purpose (OTS$) Manual』
OpenVMS ランタイム・ライブラリの OTS$機能に含まれる汎用ルーチンについて説明します。 I64， Alpha， VAX 固有のルーチンを示し，各システムで異なる動作を実行するルーチンについても説明します。

『OpenVMS RTL Library (LIB$) Manual』
OpenVMS ランタイム・ライブラリの LIB$機能に含まれる汎用ルーチンについて説明します。 I64， Alpha， VAX 固有のルーチンを示し，各システムで異なる動作を実行するルーチンについても説明します。

『OpenVMS RTL Screen Management (SMG$) Manual』
OpenVMS ランタイム・ライブラリの SMG$機能に含まれる画面管理ルーチンについて説明します。 Alpha 固有のルーチンと VAX 固有のルーチンについて説明し，各システムで異なる動作を実行するルーチンについても説明します。

『OpenVMS RTL String Manipulation (STR$) Manual』
OpenVMS ランタイム・ライブラリの STR$機能に含まれる文字列操作ルーチンについて説明します。 Alpha 固有のルーチンと VAX 固有のルーチンを示し，各システムで異なる動作を実行するルーチンについても説明します。

『OpenVMS System Messages: Companion Guide for Help Message Users』
ヘルプ・メッセージを表示するためのツールであるヘルプ・メッセージ機能について説明します。 HELP/MESSAGE コマンドとその修飾子について説明し，ヘルプ・メッセージ・データベースのカスタマイズに関する詳細情報も示します。また，システムおよびヘルプ・メッセージ機能が完全に動作しないときに表示される可能性のあるメッセージの説明も示します。

『OpenVMS System Services Reference Manual』
リソースの制御，プロセス通信の実行， I/O の制御，その他のオペレーティング・システム機能を実行するためにオペレーティング・システムで使用するルーチンについて説明します。このドキュメントは 2 分冊になっています。

『OpenVMS Utility Routines Manual』
プログラムで特定の OpenVMS ユーティリティの呼び出し可能インタフェースを使用するためのルーチンについて説明します。

『OpenVMS VAX RTL Mathematics (MTH$) Manual』
OpenVMS ランタイム・ライブラリの MTH$機能に含まれる算術演算ルーチンについて説明します。このドキュメントは OpenVMS VAX を使用するプログラマを対象にしています (Alpha のプログラマは『Compaq Portable Mathematics Library』を参照してください)。

『OpenVMS VAX System Dump Analyzer Utility Manual』
System Dump Analyzer ユーティリティを使用して，システム障害を調べ，稼動中の OpenVMS VAX システムを確認する方法について説明します。 VAX のプログラマはこのドキュメントを参照してください。 Alpha および I64 のプログラマは『VMS System Dump Analyzer Utility Manual』を参照してください。

『POLYCENTER Software Installation Utility Developer's Guide』
POLYCENTER Software Installation ユーティリティを使用してインストールされるソフトウェア製品を開発する場合の手順とガイドラインを示します。このドキュメントは， OpenVMS オペレーティング・システムのレイヤード・ソフトウェア製品のインストール手順を設計する開発者を対象にしています。

『VAX MACRO and Instruction Set Reference Manual』
VAX MACROのアセンブラ・ディレクティブと VAX 命令セットの両方について説明します。

『Volume Shadowing for OpenVMS』
フェーズ II のボリューム・シャドウイングで高いデータ可用性を提供する方法について説明します。

10.4 RMS Journalingのドキュメント

『RMS Journaling for OpenVMS Manual』
3 種類の RMS Journaling について説明し， RMS Journaling をサポートする他の OpenVMS コンポーネントについても説明します。このドキュメントでは， RMS Recovery ユーティリティ (ジャーナリングを使用して保存したデータを回復するために使用します)，トランザクション処理システム・サービス， RMS Journaling を使用するときに必要なシステム管理タスクについても説明します。

10.5 OpenVMS I64 OE 拡張キットに含まれているドキュメント

以下に示すのは OpenVMS I64 オペレーティング環境に関係するドキュメントです。

- HP DECwindows Motif for OpenVMS Installation Guide
- HP DECwindows Motif for OpenVMS New Features
- HP DECwindows Motif for OpenVMS Documentation Overview
- HP DECwindows Motif for OpenVMS Management Guide
- HP DECnet-Plus for OpenVMS Installation and Configuration
- HP DECnet-Plus for OpenVMS Introduction and User's Guide
- HP DECnet-Plus Network Management
- HP DECnet-Plus for OpenVMS DECdts Programming Reference
- HP DECnet-Plus for OpenVMS DECdts Management
- HP DECnet-Plus for OpenVMS DECdns Management
- HP DECnet-Plus for OpenVMS Network Management Quick Reference Guide
- HP DECnet-Plus for OpenVMS OSAK Programming
- HP DECnet-Plus for OpenVMS OSAK Programming Reference
- HP DECnet-Plus for OpenVMS OSAK SPI Programming Reference
- HP DECnet-Plus for OpenVMS Problem Solving Manual
- HP DECnet-Plus for OpenVMS Programming Manual
- HP DECnet-Plus for OpenVMS FTAM and Virtual Terminal User and Management

- HP DECnet-Plus for OpenVMS Problem Solving
- HP DECnet-Plus for OpenVMS Network Control Language Reference
- HP DECnet-Plus for OpenVMS Planning Guide
- HP TCP/IP Services for OpenVMS Installation and Configuration
- HP TCP/IP Services for OpenVMS Sockets API and System Services Programming
- HP TCP/IP Services for OpenVMS Concepts and Planning
- HP TCP/IP Services for OpenVMS SNMP Programming Reference
- HP TCP/IP Services for OpenVMS ONC RPC Programming
- HP TCP/IP Services for OpenVMS Tuning and Troubleshooting
- HP TCP/IP Services for OpenVMS Guide to SSH for OpenVMS
- HP TCP/IP Services for OpenVMS Management
- HP TCP/IP Services for OpenVMS Management Command Reference
- HP TCP/IP Services for OpenVMS Management Command Quick Reference Card
- HP TCP/IP Services for OpenVMS User's Guide
- HP TCP/IP Services for OpenVMS UNIX Command Equivalents Reference Card
- HP TCP/IP Services for OpenVMS Guide to IPv6
- HP DECprint Supervisor (DCPS) for OpenVMS User's Guide
- HP DECprint Supervisor (DCPS) for OpenVMS Software Installation
- HP DECprint Supervisor (DCPS) for OpenVMS Manager's Guide
- HP DCE for OpenVMS Product Guide
- HP DCE for OpenVMS Reference Guide
- HP DCE for OpenVMS Installation and Configuration Guide

10.6 アーカイブされたドキュメント

表 10–1 は，アーカイブされた OpenVMS のドキュメントを示しています。アーカイブされたドキュメントのほとんどの情報は，他のドキュメントまたはオンライン・ヘルプに統合されているという点に注意してください。

表 10–1 アーカイブされた OpenVMS のドキュメント

ドキュメント	
『A Comparison of System Management on OpenVMS AXP and OpenVMS VAX』	AA-PV71B-TE
『Building Dependable Systems: The OpenVMS Approach』	AA-PV5YB-TE
『Creating an OpenVMS Alpha Device Driver from an OpenVMS VAX Device Driver』	AA-R0Y8A-TE
『Creating an OpenVMS AXP Step 2 Device Driver from a Step 1 Device Driver』	AA-Q28TA-TE
『Creating an OpenVMS AXP Step 2 Device Driver from an OpenVMS VAX Device Driver』	AA-Q28UA-TE
『Guide to OpenVMS AXP Performance Management』	AA-Q28WA-TE
『Guide to OpenVMS Performance Management』	AA-PV5XA-TE
『Migrating an Application from OpenVMS VAX to OpenVMS Alpha』	AA-KSBKB-TE
『Migrating an Environment from OpenVMS VAX to OpenVMS Alpha』	AA-QSBLA-TE
『Migrationg to an OpenVMS AXP System: Planning for Migration』	AA-PV62A-TE
『Migrating to an OpenVMS AXP System: Recompiling and Relinking Applications』	AA-PV63A-TE
『OpenVMS Alpha Guide to 64-Bit Addressing and VLM Features』	AA-QSBCC-TE
『OpenVMS Alpha System Dump Analyzer Utility Manual』	AA-PV6UC-TE
『OpenVMS AXP Device Support: Developer's Guide』	AA-Q28SA-TE
『OpenVMS AXP Device Support: Reference』	AA-Q28PA-TE
『OpenVMS Bad Block Locator Utility Manual』	AA-PS69A-TE
『OpenVMS Compatibility Between VAX and Alpha』	AA-PYQ4C-TE
『OpenVMS Developer's Guide to VMSINSTAL』	AA-PWBXA-TE
『OpenVMS DIGITAL Standard Runoff Reference Manual』	AA-PS6HA-TE
『OpenVMS EDT Reference Manual』	AA-PS6KA-TE
『OpenVMS Exchange Utility Manual』	AA-PS6AA-TE
『OpenVMS Glossary』	AA-PV5UA-TK
『OpenVMS Guide to Extended File Specifications』	AA-REZRB-TE
『OpenVMS Master Index』	AA-QSBSD-TE
『OpenVMS National Character Set Utility Manual』	AA-PS6FA-TE
『OpenVMS Obsolete Features Manual』	AA-PS6JA-TE
『OpenVMS Programming Environment Manual』	AA-PV66B-TK
『OpenVMS Programming Interfaces: Calling a System Routine』	AA-PV68B-TK
『OpenVMS RTL DECtalk (DTK$) Manual』	AA-PS6CA-TE
『OpenVMS RTL Parallel Processing (PPL$) Manual』	AA-PV6JA-TK
『OpenVMS Software Overview』	AA-PVXHB-TE
『OpenVMS SUMSLP Utility Manual』	AA-PS6EA-TE

(次ページに続く)

表 10–1 (続き) アーカイブされた OpenVMS のドキュメント

ドキュメント	
『OpenVMS System Messages and Recovery Procedures Reference Manual: A–L』	AA-PVXKA-TE
『OpenVMS System Messages and Recovery Procedures Reference Manual: M–Z』	AA-PVXLA-TE
『OpenVMS Terminal Fallback Utility Manual』	AA-PS6BA-TE
『OpenVMS VAX Card Reader, Line Printer, and LPA11–K I/O User's Reference Manual』	AA-PVXGA-TE
『OpenVMS VAX Device Support Manual』	AA-PWC8A-TE
『OpenVMS VAX Device Support Reference Manual』	AA-PWC9A-TE
『OpenVMS VAX Patch Utility Manual』	AA-PS6DA-TE
『OpenVMS Wide Area Network I/O User's Reference Manual』	AA-PWC7A-TE
『PDP–11 TECO User's Guide』	AA-K420B-TC
『POLYCENTER Software Installation Utility User's Guide』	AA-Q28NA-TK
『TCP/IP Networking on OpenVMS Systems』	AA-QJGDB-TE
『Standard TECO Text Editor and Corrector for the VAX, PDP–11, PDP-10, and PDP-8』	CD-ROM でのみ提供

表 10–2 は，アーカイブされたネットワーキング・ドキュメントおよびインストール補足資料を示しています。

表 10–2 アーカイブされたネットワーキング・ドキュメントおよびインストール補足資料

ドキュメント	
『OpenVMS VAX Upgrade and Installation Supplement: VAX 8820, 8830, 8840』	AA-PS6MA-TE
『OpenVMS VAX Upgrade and Installation Supplement: VAX 8200, 8250, 8300, 8350』	AA-PS6PA-TE
『OpenVMS VAX Upgrade and Installation Supplement: VAX 8530, 8550, 8810 (8700), and 8820–N (8800)』	AA-PS6QA-TE
『OpenVMS VAX Upgrade and Installation Supplement: VAX 8600, 8650』	AA-PS6UA-TE
『VMS Upgrade and Installation Supplement: VAX–11/780, 785』	AA-LB29B-TE
『VMS Upgrade and Installation Supplement: VAX–11/750』	AA-LB30B-TE

ここでは，アーカイブされた OpenVMS ドキュメントについて説明します。

『A Comparison of System Management on OpenVMS AXP and OpenVMS VAX』
システム管理ツール， Alpha のページ・サイズがシステム管理操作に与える影響，システム・ディレクトリ構造，相互運用性に関する問題点，パフォーマンス情報について説明します。このドキュメントは， OpenVMS Alpha システムの管理方法を短時間に学習する必要のあるシステム管理者を対象にしています。

『Building Dependable Systems: The OpenVMS Approach』
ビジネス・アプリケーションで必要とされる信頼性を分析し，コンピューティング・システムを使用して，信頼性の達成目標をサポートする方法を判断するための，実際的な情報を示します。この情報の他に， OpenVMS や関連ハードウェア，レイヤード・ソフトウェア製品の信頼性機能の技術概要も補足されています。

『Creating an OpenVMS Alpha Device Driver from an OpenVMS VAX Device Driver』
OpenVMS VAX で使用されているデバイス・ドライバを OpenVMS Alpha で動作するデバイス・ドライバに変換する手順について説明します。このドキュメントには，Macro-32 で作成された Alpha ドライバを操作するためのデータ構造，ルーチン，マクロも含まれています。

『Creating an OpenVMS AXP Step 2 Device Driver from a Step 1 Device Driver』
Step 1 デバイス・ドライバ (OpenVMS AXP の初期のバージョンで使用) を Step 2 デバイス・ドライバにアップグレードする方法について説明します。 OpenVMS AXP バージョン 6.1 では， Step 2 のデバイス・ドライバが必要です。

『Creating an OpenVMS AXP Step 2 Device Driver from an OpenVMS VAX Device Driver』
OpenVMS VAX で使用されているデバイス・ドライバを， OpenVMS AXP バージョン 6.1 で使用される Step 2 デバイス・ドライバに移行する方法について説明します。

『Guide to OpenVMS AXP Performance Management』
OpenVMS Alpha システムでパフォーマンスを最適化するために使用される手法について説明します。

『Guide to OpenVMS Performance Management』
OpenVMS VAX システムでパフォーマンスを最適化するために使用される手法について説明します。

『Migrating an Application from OpenVMS VAX to OpenVMS Alpha』
（翻訳版は『OpenVMS VAX から OpenVMS Alpha へのアプリケーションの移行』）
OpenVMS VAX アプリケーションの OpenVMS Alpha バージョンを作成する方法について説明します。 VAX から Alpha への移行プロセスの概要を示し，移行の計画に役立つ情報も示します。移行計画で必要になる判断と，これらの判断を下すのに必要な情報の入手方法を説明します。さらに，このドキュメントでは，使用できる移行方法について説明し，各方法で必要な作業量を見積もり，各アプリケーションに最適な方法を選択できるようにします。

『Migrating an Environment from OpenVMS VAX to OpenVMS Alpha』
OpenVMS VAX システムから OpenVMS Alpha システムまたは複合アーキテクチャ・クラスタにコンピューティング環境を移行する方法について説明します。 VAX から Alpha への移行プロセスの概要を示し， VAX コンピュータと Alpha コンピュータでのシステム管理およびネットワーク管理の相違点について説明します。

『Migrating to an OpenVMS AXP System: Planning for Migration』
（翻訳版は『OpenVMS AXP オペレーティング・システムへの移行: システム移行の手引き』）
RISC アーキテクチャの一般的な特性を示し，Alpha アーキテクチャと VAX アーキテクチャを比較し，移行プロセスの概要を示し，弊社が提供している移行ツールの概要を示します。このドキュメントの内容は，アプリケーションにとって最適な移行方法を定義するのに役立ちます。

『Migrating to an OpenVMS AXP System: Recompiling and Relinking Applications
（翻訳版は『OpenVMS AXP オペレーティング・システムへの移行: 再コンパイルと再リンク』）
高級言語アプリケーションを OpenVMS Alpha に移行するプログラマを対象に，詳細な技術情報を示します。アプリケーションの移行を容易にするための開発環境の設定方法を示し，プログラマが VAX アーキテクチャの要素に対するアプリケーションの依存性を識別するのに役立つ情報を提供し，これらの依存性を解決するのに役立つコンパイラ機能を紹介します。このドキュメントの各セクションでは，VAX アーキテクチャ機能に対する特定のアプリケーションの依存性，データ移植の問題点 (アライメントの問題点など)，VAX 共有メージの移植プロセスなどについて説明します。

『OpenVMS Alpha Guide to 64-Bit Addressing and VLM Features』
（翻訳版は『OpenVMS Alpha 64 ビット・アドレッシングおよび VLM 機能説明書』）
OpenVMS Alpha オペレーティング・システムでの 64 ビット仮想アドレッシングおよび VLM (Very Large Memory) のサポートについて説明します。このドキュメントはシステム・プログラマおよびアプリケーション・プログラマを対象にしており，OpenVMS Alpha の 64 ビットおよび VLM 機能について，その特徴と利点を中心に説明します。また，これらの機能を利用して，64 ビット・アドレスをサポートし，非常に大きい物理メモリを効率よく利用できるようにアプリケーション・プログラムを拡張する方法についても説明します。

『OpenVMS Alpha System Dump Analyzer Utility Manual』
System Dump Analyzer ユーティリティを使用して，システム障害を調べ，動作中の OpenVMS Alpha システムを確認する方法について説明します。Alpha のプログラマはこのドキュメントを参照してください。VAX のプログラマは『OpenVMS VAX System Dump Analyzer Utility Manual』を参照してください。

『OpenVMS AXP Device Support: Developer's Guide』
弊社が提供していないデバイス用に OpenVMS Alpha のドライバを開発する方法について説明します。

『OpenVMS AXP Device Support: Reference』
『Writing OpenVMS Alpha Device in C』用の参照情報を提供します。デバイス・ドライバのプログラミングで使用するデータ構造，マクロ，ルーチンについて説明します。

『OpenVMS Bad Block Locator Utility Manual』
Bad Block Locator ユーティリティを使用して，古いタイプのメディアで不良ブロックを検索する方法について説明します。

『OpenVMS Compatibility Between VAX and Alpha』
エンド・ユーザ，システム管理者，プログラマに提供される機能を中心に， VAX コンピュータと Alpha コンピュータで稼動する OpenVMS を比較します。

『OpenVMS Developer's Guide to VMSINSTAL』
VMSINSTAL コマンド・プロシージャについて説明し，弊社が推奨している標準に準拠したインストール・プロシージャを設計する場合のガイドラインを示します。このドキュメントは， OpenVMS オペレーティング・システムでレイヤード・ソフトウェア製品のインストール・プロシージャを設計する開発者を対象にしています。

『OpenVMS DIGITAL Standard Runoff Reference Manual』
DSR テキスト生成ユーティリティについて説明します。

『OpenVMS EDT Reference Manual』
EDT エディタの詳細な参照情報を示します。

『OpenVMS Exchange Utility Manual』
Exchange ユーティリティを使用して，外部フォーマットのボリュームと OpenVMS ネイティブ・ボリュームの間でファイルを転送する方法について説明します。

『OpenVMS Glossary』
ドキュメンテーション全体で使用している OpenVMS 固有の用語を定義します。

『OpenVMS Guide to Extended File Specifications』
（翻訳版は『OpenVMS Extended File Specifications の手引き』）
拡張ファイル指定の概要を示し，全体的な相違点，および拡張ファイル指定を OpenVMS 環境に導入した場合の影響について説明します。

『OpenVMS Master Index』
OpenVMS フル・ドキュメンテーション・セットに含まれるドキュメントから抽出した索引情報を提供します。

『OpenVMS National Character Set Utility Manual』
National Character Set ユーティリティを使用して NCS 定義ファイルを作成する方法について説明します。

『OpenVMS Obsolete Features Manual』
VMS バージョン 4.0 ～バージョン 5.0 で廃止された DCL コマンド，システム・サービス， RTL ルーチン，ユーティリティを示します。 VMS バージョン 4.0 以降で廃止された DCL コマンド， RTL ルーチン，ユーティリティをまとめた付録もあります。

『OpenVMS Programming Environment Manual』
プログラミング環境を定義する弊社の製品およびツールの全般的な説明を示します。コンパイラ，リンカ，デバッガ， System Dump Analyzer，システム・サービス，ルーチン・ライブラリなどの機能やツールについて紹介します。

『OpenVMS Programming Interfaces: Calling a System Routine』
OpenVMS プログラミング・インタフェースについて説明し，ユーザ・プロシージャから OpenVMS システム・ルーチンを呼び出すための標準規約を定義します。さまざまな高級言語での Alpha および VAX データ・タイプのインプリメンテーションについても，このドキュメントに示しています。

『OpenVMS RTL DECtalk (DTK$) Manual』
OpenVMS ランタイム・ライブラリの DTK$機能に含まれる DECtalk サポート・ルーチンについて説明します。

『OpenVMS RTL Parallel Processing (PPL$) Manual』
OpenVMS ランタイム・ライブラリの PPL$機能に含まれる並列処理ルーチンについて説明します。 Alpha 固有のルーチンと VAX 固有のルーチンを示し，各システムで異なる動作を実行するルーチンについても説明します。

『OpenVMS Software Overview』
OpenVMS オペレーティング・システムの概要と，提供される一部の製品の概要を示します。

『OpenVMS SUMSLP Utility Manual』
SUMSLP バッチ指向エディタを使用して，操作ファイルを更新する方法について説明します。

『OpenVMS System Messages and Recovery Procedures Reference Manual』
オペレーティング・システムから出されるエラー・メッセージ，警告メッセージ，情報メッセージをアルファベット順に示します。また，各メッセージの意味と，各メッセージに対するユーザの対処法も示します。このドキュメントは 2 分冊になっています。

『OpenVMS Terminal Fallback Utility Manual』
Terminal Fallback ユーティリティを使用して，このユーティリティで使用できるライブラリ，文字変換テーブル，ターミナル・パラメータを管理する方法について説明します。

『OpenVMS VAX Card Reader, Line Printer, and LPA11–K I/O User's Reference Manual』
OpenVMS VAX でのカード・リーダー，ラボラトリ周辺アクセラレータ，ライン・プリンタのドライバについて説明します。

『OpenVMS VAX Device Support Manual』
弊社が提供していないデバイス向けの OpenVMS VAX ドライバを開発する方法について説明します。

『OpenVMS VAX Device Support Reference Manual』
『OpenVMS VAX Device Support Manual』用の参照情報を提供します。デバイス・ドライバのプログラミングで使用されるデータ構造，マクロ，ルーチンについて説明します。

『OpenVMS VAX Patch Utility Manual』
Patch ユーティリティを使用して， OpenVMS VAX の実行イメージおよび共有イメージを調べ，変更する方法について説明します。

『OpenVMS Wide Area Network I/O User's Reference Manual』
OpenVMS VAX での DMC11/DMR11， DMP11， DMF32，DR11–W， DRV11-WA， DR32，非同期 DDCMP インタフェース・ドライバについて説明します。

『PDP–11 TECO User's Guide』
PDP–11 TECO (Text Editor and Corrector) プログラムの操作手順について説明します。

『POLYCENTER Software Installation Utility User's Guide』
POLYCENTER Software Installation ユーティリティについて説明します。このユーティリティは，ユーティリティと互換性のあるソフトウェア製品のインストールと管理のために使用される新しいコンポーネントです。

『TCP/IP Networking on OpenVMS Systems』
TCP/IP ネットワーキングの概要を示し， TCP/IP 機能に対する OpenVMS DCL のサポートについて説明します。

索引

M

O

P

R

S

T

U

V

W

X

ア

オ

カ

ク

コ

シ

タ

テ

ト

ネ

ハ

ヘ

ホ

マ

ラ

レ

ロ

HP OpenVMS V8.2 新機能説明書

2005 年 4 月　発行

日本ヒューレット・パッカード株式会社

〒 140-8641 東京都品川区東品川 2-2-24 天王洲セントラルタワー

電話　(03)5463-6600 (大代表)

BA322-90009