HP OpenVMS

V8.2-1 for Integrity Servers
新機能およびリリース・ノート【翻訳版】

注文番号: BA322-90035

2005 年 10 月

本書では， OpenVMS I64 Version 8.2-1 の新機能と使用上の注意事項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 8.2-1

日本ヒューレット・パッカード株式会社

ZK6676

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

目次

図

表

まえがき

対象読者

本書は， HP OpenVMS I64 Version 8.2-1 オペレーティング・システムのすべてのユーザを対象にしています。 OpenVMS Version 8.2-1のインストレーション，アップグレード，およびご使用の前に，本書をお読みください。

本書の構成

本書の構成は以下のとおりです。

- 第 1 章では，リリースの概要，弊社のテクニカル・サポート方針，およびネットワーク・オプションについて説明します。
- 第 2 章では， OpenVMS Version 8.2-1 for Integrity Servers で提供される新機能について説明します。
- 第 3 章では， Version 8.2-1 に固有の注意事項について説明します。
- 第 4 章では， InfoServer ユーティリティの新機能について説明します。この機能により， LAN 上の仮想ディスク・デバイス用のサービスを作成することができます。
- 第 5 章では， Linker ユーティリティの概要と， OpenVMS I64 システム上でプログラムをリンクする際の留意事項について説明します。
- 第 6 章では， OpenVMS Version 8.2-1 メディア・キットに含まれる Operating Environment (OE) DVD およびその他のメディア内のディレクトリ名と製品の格納場所について説明します。

1
はじめに

この章では， Integrity サーバ製品用の HP OpenVMS Version 8.2-1 を紹介し，Version 8.2-1 のインストールまたはアップグレードを行う前に理解すべき情報について説明します。

また， OpenVMS Version 8.2-1 のインストールまたはアップグレードを行う前に，以下のすべてのドキュメントを読むことをお勧めします。

- 『HP OpenVMS V8.2-1 新機能およびリリース・ノート[翻訳版]』（本書）
- 『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』
- HP OpenVMS Version 8.2-1 for Integrity Servers **のカバー・レター**
- 『HP OpenVMS V8.2 リリース・ノート[翻訳版]』

1.1 OpenVMS Version 8.2-1 for Integrity Servers の概要

OpenVMS Version 8.2-1 for Integrity Servers (OpenVMS I64 とも呼ばれる) は，OpenVMS Version 8.2 for Integrity Servers の後継リリースです。 OpenVMS Version 8.2-1 では，前リリースのすべての機能の他に，以下の新しい機能とハードウェア・サポートが追加されました。

主な新機能には以下のものがあります。

- 以下の Integrity サーバ・プラットフォームのサポート
  - — HP Integrity rx7620 サーバ
  - — HP Integrity rx8620 サーバ
  - — HP Integrity Superdome サーバ

  HP Integrity サーバ用の HP sx1000 チップセットには，これらのサーバ用の CPU，メモリおよび I/O サブシステムが含まれます。
- InfoServer ユーティリティ

  このユーティリティを使うと， LAN 上の仮想ディスク・デバイス用のサービスを作成することができます。また， LAN 上の仮想 DVD ドライブからブートして，そこからシステムを Version 8.2-1 にアップグレードすることもできます。

  rx1600， rx1620， rx2600， rx2620，および rx4640 Integrity サーバのコア LAN I/O デバイスでサポートされます。

- nPartitions のサポート

  弊社のセル型サーバ (rx7620, rx8620, Integrity Superdome など) では, 各nPartitionsはサーバ・ハードウェア・リソースのサブセットを定義しており, それぞれが独立したシステム環境として使用されます。**セル型サーバ**は, 1 つ以上のオペレーティング・システムを実行でき, ハードウェア・リソースを複数の nPartitions に分割することができるハードウェア・コンプレックスです。セル型サーバでは, すべてのプロセッサとメモリはセルと呼ばれる回路基板上に実装され, それらを nPartitions に排他的に割り当てて使用することができます。 nPartitions は専用のシステム・ブート・インタフェースを備えており, 各 nPartitions は独立にブートやリブートを行うことができます。

- 省電力機能

  アイドル時に CPU を省電力モードにすることによって, 電力消費を抑えて, エネルギー・コストを削減することができます。

- 低コスト 2 ノード・マルチホスト SCSI クラスタ

  2 ノードの OpenVMS I64 Cluster システム用に共用 SCSI ストレージがサポートされ, 2 ノード・クラスタ環境 (エントリ・クラスの Integrity サーバのみ) で手頃な価格の共用ストレージを利用することができます。

- I64 クラスタでのノード数の増加

  OpenVMS Cluster システムでは最大 96 ノードまでサポートされるようになり, Integrity サーバ・ノードと AlphaServer ノードを混在させる際の制約もなくなりました。

- 新しいシステム・ダンプ分析コマンドを使用した Fibre Channel の性能データ

  新しいシステム・ダンプ分析コマンド FC PERFORMANCE を使用して, Fibre Channel の性能データが取得できるようになりました。

- PPL ユニット割り当てツール

  OpenVMS Per Processor Licenses (PPL) 用のライセンス・ユニットの管理に役に立つツールです。

1.2 弊社ソフトウェアのテクニカル・サポート方針

弊社との特別の契約を締結しない限り, 弊社では, 弊社ソフトウェアの最新版 (現在出荷されているバージョン) と, 直前のバージョンの製品についてのみ, ソフトウェアのテクニカル・サポートを行います。さらに, 弊社の指定する構成に含まれるハードウェアでソフトウェアを使用する場合に限ります。ここでいうバージョンとは, 新機能, 機能拡張, またはメンテナンス・アップデートを含んだソフトウェア製品の各リリースと定義されます。

各 OpenVMS オペレーティング・システム・ソフトウェアの現行バージョン・レベルのサポート (標準サポート (SS)) または旧バージョンのサポート (PVS) が，それぞれのガイドラインに基づいて提供されます。 OpenVMS I64， OpenVMS Alpha，OpenVMS VAX の最近のバージョンに対する現在のサポート・レベルは，次の Web サイトから入手してください。

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

以下の OpenVMS 関連製品は，これらが付属しているオペレーティング・システムと同一レベル (SS または PVS) で同一期間サポートされます。

- HP Advanced Server for OpenVMS
- HP DECnet (Phase IV)
- HP DECnet-Plus for OpenVMS
- HP OpenVMS Cluster Client Software
- HP OpenVMS Cluster Software for OpenVMS
- HP RMS Journaling for OpenVMS
- HP TCP/IP Services for OpenVMS
- HP Volume Shadowing for OpenVMS

ただし，これらの製品にはそれぞれ専用のサポート契約が必要です。オペレーティング・システムのサポート契約には含まれません。

1.3 アプリケーションの互換性

OpenVMS では，公開された API は，以降のすべてのリリースで一貫してサポートされます。通常，公開された API を使用しているアプリケーションであれば，OpenVMS の新しいリリースをサポートするために変更が必要になることはありません。一般に API の中にはサポートが廃止されマニュアルから削除されるものもありますが， OpenVMS ではドキュメントに記載のない API として引き続き使用できます。

1.4 パッチ・キットの入手方法

弊社製品のパッチ・キット (修正キットとも呼ぶ) は， HP IT リソース・センタ (ITRC) からオンラインで入手できます。 ITRC パッチ・ダウンロード・サイトを使用するためには，ユーザ登録とログインが必要です。すべてのユーザが登録可能で，サービス契約は不要です。次の URL で，登録とログインができます。

http://www2.itrc.hp.com/service/patch/mainPage.do

また，FTP を使用して，次の場所からパッチを入手することもできます。

```
ftp://ftp.itrc.hp.com/openvms_patches
```

1.5 ネットワーク・オプション

OpenVMS では，使用するネットワーク・プロトコルを柔軟に選択できます。DECnet が必要な場合も，TCP/IP が必要な場合も，OpenVMS ではネットワークにとって最適なプロトコルあるいは複数のプロトコルの組み合わせを選択できます。OpenVMS では，弊社のネットワーク製品と他社製ネットワーク製品のどちらも使用できます。

OpenVMS Version 8.2-1 のインストール時に，サポートされている次の HP ネットワーク・ソフトウェアをインストールすることができます。

- HP DECnet-Plus Version 8.2-1 for OpenVMS または HP DECnet Phase IV Version 8.2-1 for OpenVMS (これらの DECnet 製品を同時に実行することはできません)

  DECnet-Plus には，DECnet フェーズ IV 製品のすべての機能に加えて，TCP/IP または OSI プロトコル上で DECnet を動作させる機能も含まれています。

  Prior Version Support 契約を結ばれているお客様に対しては，DECnet Phase IV がサポートされます。Prior Version Support サービスについての詳細は，第 1.2 節を参照してください。

- HP TCP/IP Services for OpenVMS Version 5.5

  TCP/IP Services と DECnet は，システム上で同時に実行できます。システムに HP DECnet-Plus for OpenVMS と TCP/IP Services をインストールすると，DECnet アプリケーション，OSI アプリケーション，またはその両方を，TCP/IP ネットワーク上で実行できます。DECnet over TCP/IP の実行 (RFC 1859)，OSI over TCP/IP の実行 (RFC 1006) についての詳細は，『DECnet-Plus for OpenVMS Management Guide』を参照してください。

または，OpenVMS をインストールした後で，OpenVMS Version 8.2-1 で動作する他社製ネットワーク製品をインストールすることもできます。

HP ネットワーク・ソフトウェアをインストールした後の構成および管理方法については，TCP/IP，DECnet-Plus，または DECnet の各マニュアルを参照してください。マニュアルは OpenVMS Documentation CD-ROM，Online Documentation Library CD，および次の OpenVMS ドキュメント Web サイトから，オンライン形式で入手できます。

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

1.6 古いバージョンの OpenVMS と互換性のないディスク

OpenVMS Version 8.2-1 のインストール手順では，ターゲット・ディスクをボリューム拡張付き (INITIALIZE/LIMIT) で初期化します。これにより， Version 7.2 よりも前の OpenVMS とはディスクの互換性がなくなります。この新しいディスクを Version 7.2 よりも前の OpenVMS にマウントする場合は，そのバージョンのオペレーティング・システムと互換性を持つようにそのディスクを設定しなければなりません。設定手順の詳細は，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。

この手順を実行すると，(/CLUSTER_SIZE で定義される) 新しいシステム・ディスクの最小割り当てサイズは，以前より大きくなることがあります。その結果，小さなファイルが必要以上のスペースを専有するようになります。そのため，この手順は，Version 7.2 より前の OpenVMS にマウントしなければならないシステム・ディスクに対してのみ実行してください。

注意

ODS-5 ディスクも，Version 7.2 より前の OpenVMS とは互換性がありません。

1.7 Integrity サーバでのシステム・イベント・ログ (SEL)

HP Integrity サーバは，システム・コンソールのストレージ内にシステム・イベント・ログ (SEL) を保存しており， OpenVMS I64 システムは SEL の内容を自動的に OpenVMS エラー・ログに転送します。コンソールから操作しているとき，正常なブート処理中に BMC (Baseboard Management Controller) SEL が満杯である旨のメッセージが表示されることがあります。 BMC SEL が満杯の場合でも，プロンプトに従って操作すれば，運用を継続することができます。 OpenVMS は SEL の内容を自動的に処理してクリアします。

1.8 Integrity サーバ用のファームウェア

OpenVMS I64 Version 8.2-1 は，サポートされている各 Integrity サーバ用の最新のファームウェアでテストされています。

エントリ・クラスの Integrity サーバの場合には，最新のシステム・ファームウェアを使用することをお勧めします。エントリ・クラスの Integrity サーバのシステム・ファームウェアをアップデートする方法については，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。 rx7620， rx8620， Superdome サーバで，ファームウェアのアップデートが必要な場合には，弊社のカスタマ・サポートにお問い合わせください。

次の表に，エントリ・クラスの Integrity サーバで推奨するファームウェア・バージョンの一覧を示します。

表 1–1 エントリ・クラスの Integrity サーバのファームウェア・バージョン

システム	システム ファームウェア	BMC ファームウェア	MP ファームウェア	DHCP ファームウェア
rx1600	2.18	3.48	E.03.15	N/A
rx1620	2.18	3.48	E.03.15	N/A
rx2600	2.31	1.53	E.03.15	N/A
rx2620	3.17	3.47	E.03.15	N/A
rx4640	3.17	3.49	E.03.15	1.10

セル型サーバの場合には， MP コマンドのメニューにアクセスしてsysrevコマンドを実行し， MP ファームウェアのリビジョン・レベルを表示します。sysrevコマンドは， MP を持つすべての HP Integrity サーバで利用できます。セル型 Integrity サーバでは， EFI のinfo fwコマンドを実行しても Management Processor (MP) ファームウェアのバージョンは表示されません。

MP を持たないエントリ・クラスの Integrity サーバでファームウェアのバージョン情報を調べるには，次のように， EFI プロンプトでinfo fwコマンドを実行します。

```
Shell> info fw

FIRMWARE INFORMATION

   Firmware Revision: 2.13 [4412]          1

   PAL_A Revision: 7.31/5.37
   PAL_B Revision: 5.65
   HI Revision: 1.02

   SAL Spec Revision: 3.01
   SAL_A Revision: 2.00
   SAL_B Revision: 2.13

   EFI Spec Revision: 1.10
   EFI Intel Drop Revision: 14.61
   EFI Build Revision: 2.10

   POSSE Revision: 0.10

   ACPI Revision: 7.00

   BMC Revision: 2.35                      2
   IPMI Revision: 1.00
   SMBIOS Revision: 2.3.2a
   Management Processor Revision: E.02.29  3
```

1 システムのファームウェア・リビジョンは， 2.13 です。

2 BMC のファームウェア・リビジョンは， 2.35 です。

3 MP のファームウェア・リビジョンは， E.02.29 です。

注意

この例で表示されているバージョンは，ご使用中のシステムのバージョンとは一致しません。

HP Integrity rx4640 サーバには，アップグレード可能なファームウェアを備えた， Dual Hot Plug Controller (DHPC) ハードウェアが含まれています。 DHPC ファームウェアの現在のバージョンを確認するには，次の例のように， EFI のinfo chiprevコマンドを使用します。ホット・プラグ・コントローラのバージョンが表示されます。 0100 という表示は， Version 1.0 を意味します。 0110 という表示は， Version 1.1 を意味します。

```
Shell> info chiprev

CHIP REVISION INFORMATION

   Chip                  Logical     Device       Chip
   Type                     ID         ID       Revision
   -------------------   -------     ------     --------
   Memory Controller         0       122b         0023
   Root Bridge               0       1229         0023
     Host Bridge          0000       122e         0032
     Host Bridge          0001       122e         0032
     Host Bridge          0002       122e         0032
     Host Bridge          0004       122e         0032
      HotPlug Controller     0          0         0110
     Host Bridge          0005       122e         0032
      HotPlug Controller     0          0         0110
     Host Bridge          0006       122e         0032
     Other Bridge              0          0         0002
       Other Bridge            0          0         0008
         Baseboard MC          0          0         0235
```

EFI のアクセス方法および使用方法については，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。詳細は，ご使用中のサーバに付属しているハードウェア・ドキュメントを参照してください。

エントリ・クラスの Integrity サーバでファームウェアをアップグレードする方法は，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。 rx7620， rx8620， Superdome のファームウェアのアップグレードが必要な場合には，弊社のカスタマ・サポートにお問い合わせください。

1.9 アップグレードを行う前に TIE キットの削除が必要

TIE (Translated Image Environment) は， OpenVMS I64 Version 8.2-1 に統合されました。詳細は， HP OpenVMS システムの移行ソフトウェア Web サイトを参照してください。

```
http://www.hp.com/go/openvms/products/omsais
```

OpenVMS I64 Version 8.2 に TIE の PCSI キット (HP-I64VMS-TIE) をインストールしてある場合には， OpenVMS I64 Version 8.2-1 にアップグレードする前に， TIE の旧バージョンを手作業で削除する必要があります。

TIE 製品のキットを削除するには，次のコマンドを実行します。

```
$ PRODUCT REMOVE TIE
```

OpenVMS I64 Version 8.2-1 には， TIE 製品のキット HP I64VMS TIE V1.0 はインストールしないでください。

1.10 システムのブートに関する注意事項

OpenVMS I64 システムのブートに関する注意事項は，以下のとおりです。

1.10.1 インストール DVD からのブート

サポートする最小メモリ構成の I64 システムで，インストール DVD からブートすると次のようなメッセージが表示されます。

```
********* XFC-W-MemmgtInit Misconfigure Detected ********
XFC-E-MemMisconfigure MPW_HILIM + FREEGOAL > Physical Memory and no reserved memory for XFC
XFC-I-RECONFIG Setting MPW$GL_HILIM to no more than 25% of physical memory XFC-I-RECONFIG
Setting FREEGOAL to no more than 10% of physical memory
********* XFC-W-MemMisconfigure AUTOGEN should be run to correct configuration ********
********* XFC-I-MemmgtInit Bootstrap continuing ********
```

このメッセージは，システム・キャッシュ (XFC) の初期設定で，インストール中のキャッシュ処理を有効にするために， SYSGEN パラメータの MPW_HILIM と FREEGOAL を正常に調整したことを意味しています。インストールは続行できます。

1.10.2 I64 システムのリブート設定

OpenVMS I64 システムでは， EFI または OpenVMS I64 Boot Manager ユーティリティによる設定を行わない限り，自動リブートは行われません。クラスタ環境では，Alpha システムはリブートされますが， I64 システムはシャットダウン後，コンソール・プロンプトの状態になります。

I64 システムをリブートするように設定する方法については，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。

1.10.3 共通クラスタ・システム・ディスクを使用するブート

追加のノードが OpenVMS Cluster システムの共通クラスタ・ディスクを使用してブートするように設定するためには， OpenVMS I64 Boot Manager (BOOT_OPTIONS.COM) を実行する必要があります。

1 つのシステムまたはスタンドアロン・システムでクラスタリングを有効にした後，以下のようにして，追加の I64 ノードが共用ディスクを使ってブートするように設定することができます。

1. ターゲット・ノードで，インストレーション・メディアから HP OpenVMS Version 8.2-1 をブートします。
2. OpenVMS のインストール・メニューから[Option 7, Execute DCL commands and procedures]を選択し， DCL プロンプトにアクセスします。
3. DCL プロンプトで次のコマンドを実行し， OpenVMS I64 Boot Manager ユーティリティを起動します。

   ```
   $$$ @$SYS$MANAGER:BOOT_OPTIONS
   ```

4. ユーティリティを起動するとメイン・メニューが表示されます。システム・ディスクをブート・オプションとして追加するには， 1 ((1) ADD an entry to the Boot Options list) を入力します。
5. ユーティリティはプロンプトを表示して，デバイス名， EFI ブート・オプション・リスト内の場所，ブート・フラグの設定，ブート・オプションの説明のような追加情報の入力を求めます。
6. ブート・オプションの追加が終ったら，プロンプトに対して E を入力してユーティリティを終了します。
7. DCL プロンプトからログアウトして， I64 システムをシャットダウンします。

I64 Boot Manager の Boot Options Management Utility についての詳細は，『HP OpenVMS システム管理マニュアル (上巻)』を参照してください。

1.10.4 Fibre Channel ストレージ・デバイスからのブート

多くのユーザは，処理速度の点と， SAN 内で共通クラスタ・システム・ディスクとしてサービスできるという点から， Fibre Channel (FC) ストレージ・デバイスからのブートを選びます。 OpenVMS I64 システムで FC ストレージ・デバイスからブートする方法は， OpenVMS Alpha システムで FC ストレージ・デバイスからブートする方法と大きな違いがあります。

OpenVMS I64 システムで FC デバイスを構成してブートする方法については，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』の Fibre Channel の章を参照してください。

1.10.5 OpenVMS I64 Boot Manager ユーティリティ: マルチパス Fibre Channel ディスク・デバイスの追加

OpenVMS の Boot Manager ユーティリティ BOOT_OPTIONS.COM は， SAN 内のブート・ディスクとダンプ・ディスクのリストを指定するために使用します。このリストにマルチパス Fibre Channel ディスク・デバイスを追加するときには，冗長パスも含め， SAN 内で検出されるこのデバイスへのすべてのパスがリストされます。冗長パスは Remove オプションを使用してリストから削除できます。

1.10.6 Fibre Channel ブート・ディスク: 簡単になった設定手順

OpenVMS I64 Version 8.2 の Fibre Channel ブート・デバイスの設定手順では， OpenVMS I64 Boot Manager の BOOT_OPTIONS.COM を使って EFI Boot Manager に値を指定する必要がありました。この手順は OpenVMS I64 Version 8.2-1 のインストール手順では自動化されました。ただし，必要な場合には，手動で設定することもできます。

OpenVMS I64 Version 8.2-1 のインストール手順では， Fibre Channel ディスクの名前がブート・デバイスとして表示され，このデバイスをブート・オプションとして追加するかどうかが問い合わされます。このデフォルト設定は受け入れてください。受け入れない場合には，インストールまたはアップグレードの終了後に，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』の説明に従って OpenVMS I64 Boot Manager を実行することになります。

注意

ご使用中のシステムが rx1600，rx2600，rx4600 **ファミリのサーバのメンバで，** EFI **ブート・メニュー内に** Fibre Channel **ブート・デバイスが表示されない場合には，** SAN **全体がスキャンされるため** EFI **での初期化に時間がかかります。**

SAN **の大きさによっては，この遅延は数秒から数分に及ぶことがあります。セル型システム** (rx7620，rx8620，Superdome **ファミリのサーバ**) **ではこの**

遅延はありません。この遅延は，すべての OpenVMS I64 システムで，最初にインストール DVD から OpenVMS をブートするときに発生します。

Fibre Channel ブート・デバイスからブートしたり， Fibre Channel アダプタのファームウェアをアップデートする方法は，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。

1.11 HP DECwindows Motif

この節の HP DECwindows Motif に関する注意事項は， OpenVMS I64 ユーザを対象としています。

1.11.1 サーバ起動前の周辺デバイスの接続

システムを DECwindows X ディスプレイ・サーバとして適切に構成するためには，起動前に，次のすべての周辺コンポーネントが接続されていなければなりません。

- モニタ
- USB マウス
- USB キーボード

接続されていない場合，サーバ・システムはデバイスの初期化処理を正しく完了できないことがあります。たとえば，入力デバイス (マウスおよびキーボード) なしでサーバ・システムを起動すると，ブランク・スクリーンになり， DECwindows が起動しません。

この問題を解決するには，すべての周辺デバイスを接続してから，サーバの再起動，またはシステムのリブートを行ってください。

1.11.2 起動中に表示されるカウントダウン・メッセージ

クライアント・オンリー・モード (サーバ無しの構成，またはマウスやキーボードが接続されていない場合) で DECwindows Motif を実行すると，起動中に次のようなメッセージが表示されることがあります。

```
Waiting for mouse...
Waiting for keyboard...
```

これらのメッセージは，デバイス・ポーリングの実行中だという情報を示しているにすぎません。 15 秒経過すると，これらのメッセージは消えます。これは通常，Server Management オプションで組み込みグラフィックスを指定したサーバで，マウスやキーボードを接続していなかった場合に発生します。

このメッセージが表示されないようにして，15 秒間の遅延が発生しないようにするには，システムを起動する前に，入力デバイス (USB マウスおよび USB キーボード) を接続してください。

システムでローカル・グラフィックスを使用しない場合は，SYSTARTUP_VMS.COM 内で論理名を次のように定義します。

```
$ DEFINE/SYSTEM DECW$IGNORE_WORKSTATION TRUE
```

これによって DECwindows の起動時にローカル・グラフィックスに対する操作が開始されないようにすることができます。

1.11.3 ブート時にサポートされない VGA コンソール

VGA 文字型コンソールは，OpenVMS のブート時にはサポートされません。Integrity サーバのファームウェアは VGA からの入力を許していますが，BOOT コマンドに続いて，デバイスが使用できないことを示すエラー・メッセージが表示されます。コンソール出力をシリアル・ポート (または Server Management オプションのシリアル・コンソール・ポート) にリダイレクトして，ブートは続行されます。DECwindows が起動された後，グラフィック・ディスプレイで通常どおりに動作します。会話型ブートを利用するためには，端末または端末エミュレータを実行しているシステムが，システムのシリアル・ポートに接続されている必要があります。LAN 接続のコンソールなどの，その他のオプションも利用することができます。

1.11.4 オプションのグラフィックス

エントリ・クラスの Integrity サーバでは，2D マルチヘッド操作や 3D 操作が可能な ATI Radeon 7500 PCI オプション (HP 部品番号 AB551A) がサポートされています。構成方法と使用方法については，このデバイスの『Installation Guide』または Quickspecs を参照してください。

1.11.5 キーボードのサポート

OpenVMS USB キーボード (HP 部品番号 AB552A) は，DECwindows のグラフィックをサポートしているすべての Integrity システムでサポートされています。このキーボードには 3 ボタンのホイール・マウスが付属しています。

2

OpenVMS Version 8.2-1 の新機能

この章では， OpenVMS I64 Version 8.2-1 で導入された新機能について説明します。また，第 4 章では InfoServer ユーティリティの機能，そして第 5 章では Linker ユーティリティの機能について説明します。

2.1 新しい Integrity サーバのサポート

Version 8.2-1 から， OpenVMS I64 がサポートする Integrity サーバ・プラットフォームに以下の 3 つが追加されました。

- HP Integrity rx7620 サーバ

  1 台の 2 ウェイ～ 8 ウェイの Intel® Itanium® 2 プロセッサ SMP サーバとして，または 2 つの独立したハード・パーティション (nPartitions) に分割して動作させることができます。

- HP Integrity rx8620 サーバ

  rx7620 サーバをベースに，クロスバー・バックプレーンと 2 枚のセル基板を追加して構築されています。 2 ウェイ～ 16 ウェイの Intel® Itanium® 2 プロセッサ SMP サーバとして，または複数のより小さな独立した nPartitions に分割して動作させることができます。

- HP Integrity Superdome サーバ

  ハイエンドの Integrity サーバであり，以下のモデルで構成されます。

  – SD16A (最大 4 枚のセルをサポートするサーバ・コンプレックス)

  – SD32A (最大 8 枚のセルをサポートするサーバ・コンプレックス)

  – SD64A (最大 16 枚のセルをサポートするサーバ・コンプレックス)

2.1.1 HP Integrity サーバ用の HP sx1000 チップセットがサポートする構成

HP Integrity サーバ用の HP sx1000 チップセットは， HP Integrity rx7620 サーバ， HP Integrity rx8620 サーバ， HP Integrity Superdome サーバに CPU，メモリ， I/O サブシステムを提供します。 HP sx1000 チップセット・アーキテクチャでは，セル・コントローラが 4 基の CPU と結合され，コンピューティング・セルを形成します。セル・コントローラ・チップは， I/O デバイスおよびオフセル・メモリへのパスも提供します。

rx7620, rx8620, および Superdome サーバはそれぞれ異なる数の sx1000 チップセット・セルで構成されます。 rx7620 は最大 2 枚のセル (8 CPU), rx8620 は最大 4 枚のセル (16 CPU), Superdome は最大 16 枚のセル (64 CPU) で構成されます。

OpenVMS I64 Version 8.2-1 では, rx7620, rx8620, Superdome で以下の構成をサポートします。

- CPU

  各セルには最大で 4 基の CPU を搭載できます。 OpenVMS は以下のリビジョンの CPU をサポートします。

  — rx7620

    * Madison 9, 1.6 GHz, 6 MB キャッシュ CPU
    * Madison 9, 1.5 GHz, 4 MB キャッシュ CPU

  — rx8620

    * Madison 9, 1.6 GHz, 6 MB キャッシュ CPU
    * Madison 9, 1.5 GHz, 4 MB キャッシュ CPU

  — Superdome

    * Madison 9, 1.6 GHz, 9 MB キャッシュ CPU

- メモリ

  — rx7620 のセル当たりの最大メモリ (セル当たり 16 DIMM スロット)

    * 2 GB DIMM, セル当たり 32 GB, システム当たり最大 64 GB (2 セル基板)
    * 4 GB DIMM, セル当たり 64 GB, システム当たり最大 128 GB

  — rx8620 のセル当たりの最大メモリ (セル当たり 16 DIMM スロット)

    * 2 GB DIMM, セル当たり 32 GB, システム当たり最大 128 GB (4 セル基板)
    * 4 GB DIMM, セル当たり 64 GB, システム当たり最大 256 GB

  — Superdome のセル当たりの最大メモリ (セル当たり 32 DIMM スロット, nPartitions 当たり最大 4 セル)

    * 1 GB DIMM, セル当たり 32 GB, 4 セル nPartitions 当たり 128 GB
    * 2 GB DIMM, セル当たり 64 GB, 4 セル nPartitions 当たり 256 GB

- コア I/O

  — rx7620

    1 枚または 2 枚のコア I/O 基板を備え, 各基板は, 1 つの Ultra 160 LVD SCSI ポート, 1 つの 10/100/1000Base-T LAN ポート, 2 つの RS-232 シリアル・ポート (1 つはコンソール用で, もう 1 つは UPS (無停電電源装置) 用), 1 つの 10/100Base-T ポート (Web および LAN コンソール接続用) を装備。

— rx8620

1 ～ 4 枚のコア I/O 基板を備え (2 つのシステム拡張ユニット・キャビネットを使用)，各基板は，1 つの Ultra3 LVD SCSI ポート，1 つの 10/100/1000Base-T LAN ポート，2 つの RS-232 シリアル・ポート (1 つはコンソール用で，もう 1 つは UPS 用)，1 つの 10/100Base-T ポート (Web および LAN コンソール接続用) を装備。

— Superdome

1 ～ 16 枚のコア I/O 基板 (汎用 10/100/Base-T LAN カード) をそれぞれ PCI シャーシのスロット 0 に備えています。パーティションには複数のコア I/O を割り当てることができますが，同時に複数のコア I/O をアクティブにすることはできません。

- PCI シャーシ

  — 最大 PCI I/O: rx7620 と rx8620 の場合には，コア I/O 当たり 8 スロット (セル基板を実装する必要がある)

  — 最大 PCI I/O: Superdome の場合には，セル当たり 12 スロット (パーティションのプライマリ・セルの場合には，スロット 0 にコア I/O 基板を実装する必要がある)

- ハード・パーティション

  — rx7620

  1 ～ 2 個の nPartitions (2 nPartitions の場合，システム内の各セルに 1 個)

  — rx8620

  1 ～ 4 個の nPartitions (4 nPartitions の場合，システム内の各セルに 1 個)。3 nPartitions 以上構成する場合には，システム拡張ユニット (SEU) が必要。

  — Superdome

  4 ～ 16 個の nPartitions (16 nPartitions の場合，システム内の各セルに 1 個)。 OpenVMS Version 8.2-1 を動作させる場合には， 1 個の Superdome nPartitions で最大 4 セル (最大 16 CPU) をサポート。

2.2 InfoServer からの OpenVMS I64 Operating Environment DVD のブート

OpenVMS I64 Version 8.2-1 では， OpenVMS Version 8.2-1 オペレーティング・システムで提供される InfoServer ソフトウェアを使用すれば， LAN 上の仮想 DVD ドライブからブートすることができます。ただし，この機能が使用できるのはオペレーティング・システムのアップグレードを行うときだけで，以下の Integrity サーバでのみサポートされます。

- rx1600
- rx1620

- rx2600
- rx2620
- rx4640

InfoServer ソフトウェアには，ディストリビューション・キットの 1 つのコピーからネットワーク上の複数のシステムをブートできるという利点もあります。

InfoServer を使用するネットワーク・ブートでは， OpenVMS V8.2-1 に固有のいくつかの構成手順が必要です。 InfoServer ハードウェアを使用した OpenVMS Alpha システム用のネットワーク・ブート構成は， OpenVMS I64 システムには適用できません。 OpenVMS I64 InfoServer を使用して，システムを LAN 上でブートする方法については，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。 InfoServer ユーティリティについての詳細は，第 4.1 節を参照してください。

2.3 OpenVMS Cluster システム

ここでは，今回のリリースで導入された OpenVMS Cluster のいくつかの新機能について説明します。

2.3.1 OpenVMS Cluster に構成できる OpenVMS I64 システムの数の増加

Version 8.2-1 から OpenVMS は， OpenVMS I64 システムと OpenVMS Alpha システムで構成される複合アーキテクチャの OpenVMS Cluster システム，または，OpenVMS I64 システムだけで構成される単一アーキテクチャ・クラスタで， 96 台の OpenVMS システムをサポートするようになりました。従来のリリースでは，どちらのタイプの OpenVMS Cluster システムでも， OpenVMS I64 システムの最大サポート台数は 8 台でした。 OpenVMS I64 システムと OpenVMS Alpha システムを混在させた複合アーキテクチャのクラスタでは，合計 16 システムまでサポートされていました。

大規模 OpenVMS Cluster 環境では，以下のことが実現できます。

- これまで 16 メンバに制限されていた複合アーキテクチャの OpenVMS Cluster システムに，さらにシステム (I64 または Alpha) を追加する。
- すでに 16 システム以上が組み込まれている Alpha のみのクラスタに OpenVMS I64 システムを追加する。

OpenVMS Cluster のサポートについての詳細は，『OpenVMS Cluster 構成ガイド』と『OpenVMS Cluster システム』を参照してください。

2.3.2 2 ノード OpenVMS I64 クラスタ・システム用の共用 SCSI ストレージのサポート

2 ノード OpenVMS I64 クラスタ・システム用の共用 SCSI ストレージが OpenVMS Version 8.2-1 でサポートされるようになりました。これまでのリリースでは，共用 SCSI ストレージは，以前の SCSI ホスト・バス・アダプタ (HBA) を使用して OpenVMS Alpha システムだけでサポートされていました。

OpenVMS クラスタ・システムで共用 SCSI ストレージを使用すると， 1 つの SCSI バスに接続された複数のコンピュータ間で， SCSI ストレージ・デバイスへのアクセスを直接共用することができます。この機能によって， SCSI ストレージへの共用アクセスを使用した高可用性システムを構成することができます。

OpenVMS I64 クラスタ・システムの共用 SCSI ストレージには，以下の制約があります。

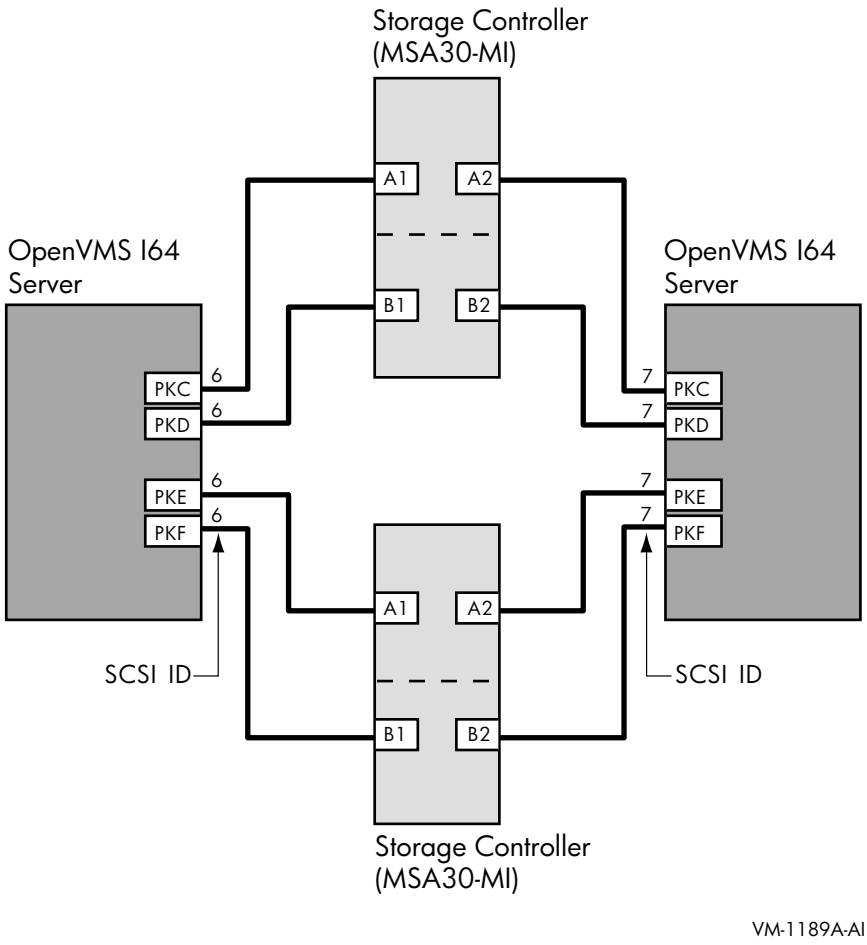
- 1 つの SCSI バスに接続できる OpenVMS I64 システムは 2 台まで。
- 各システムに接続できる共用 SCSI バスは 4 本まで。
- サポートされるシステムは， rx1600 ファミリ， rx2600 ファミリ， rx4640。
- サポートされる HBA は A7173A HBA のみ。
- サポートされる SCSI ストレージ・タイプは MSA30-MI ストレージ筐体のみ。
- サポートされるディスク・ファミリは Ultra320 SCSI ディスク・ファミリのみ。

図 2–1 にこの構成の例を示します。ホスト対ホストの OpenVMS クラスタ通信用に， 2 番目のインターコネクト (LAN) が必要であることに注意してください。 OpenVMS クラスタ通信は SCA (System Communications Architecture) 通信とも呼ばれます。

また，この構成では， SCSI ID 6 と 7 を使用する必要があることも注意してください。システムの一方では，共用 SCSI バスに接続される A7173A アダプタの各ポートの SCSI ID として，工場出荷時のデフォルトの 7 でなく， 6 を設定する必要があります。 IPF Offline Diagnostics and Utilities CD に収録されている U320_SCSI pscsi.efi ユーティリティを使用すれば， SCSI ID を変更することができます。この手順は次の URL から入手できる『HP A7173A PCI-X Dual Channel Ultra320 SCSI Host Bus Adapter Installation Guide』で説明されています。

```
http://docs.hp.com/en/netcom.html
```

図 2–1 2 ノード OpenVMS I64 Cluster システム



2.4 Integrity サーバのコンソールで Ctrl/P を押すと IPC が起動

OpenVMS Version 8.2 では， Integrity サーバのコンソールで Ctrl/P を押すと，システムをクラッシュさせるかどうかを問い合わせるプロンプトが表示されました。

```
Crash (y/n):
```

しかし， XDELTA デバッガがシステムにロードされている場合には， Ctrl/P を押すとデバッガが起動されます。 OpenVMS Version 8.2-1 から，コンソールの Ctrl/P の動作はクラッシュ・プロンプトではなく， IPC ユーティリティの起動に変更されました。 XDELTA がロードされている場合には，これまでどおり Ctrl/P によって制御が XDELTA デバッガに渡されます。システム管理者は， Ctrl/P ユーティリティを使用して，ディスクのクォーラムを調整したり，マウント確認をキャンセルすることができます。

IPC ユーティリティは以下のようにアップデートされました。

- 小さなヘルプ画面が表示される。
- IPC を起動しても 45 秒以内に入力がない場合には，ユーティリティは終了する。
- IPC が Ctrl/P への応答に失敗した場合には，コンソールでもう一度 Ctrl/P を押すと，Crash (y/n):プロンプトが表示される。
- IPC 内で Ctrl/P を押すと，Crash (y/n):プロンプトが表示される。このプロンプトに対する応答にタイムアウトはない。

IPC ユーティリティは，プライマリ CPU 上で高い割り込み優先順位で動作します。このユーティリティが動作していてもクラスタ通信は実行できますが，このユーティリティは短時間のうちに使って，使い終わったらすぐに終了させるようにしてください。

2.5 HP OpenVMS Debugger

ここでは， OpenVMS V8.2-1 Integrity システム上の OpenVMS Debugger の新機能について説明します。

2.5.1 C++ の演算子名のサポートの改善

C++ の演算子名のサポートが改善されました。特に，ユーザ定義の演算子名がサポートされるようになりました。この変更が行われる前は，演算子名を%NAMEで囲む必要がありました。

例は次のとおりです。

```
DBG> SHOW SYMBOL /FULL operator ==
routine C::operator==
    type signature: bool operator==(C)
    code address: 198716, size: 40 bytes
    procedure descriptor address: 65752
DBG> SET BREAK operator==
```

2.5.2 SET MODULE コマンドの使用はオプションになった

OpenVMS Debugger はモジュールの設定を自動的に行うようになったため，明示的な SET MODULE コマンドの使用はオプションになりました。このように機能拡張を行った理由は，次の 2 つです。

- デバッガがグローバル・シンボル名を認識するため，シンボルから，宣言されているモジュールが判定できる。

- ユーザがデバッガ・コマンドでモジュール名を指定したときに，デバッガがモジュール情報を自動的にロードする。たとえば，次のように入力したとします。

  ```
  DBG> SET BREAK X\Y
  ```

  デバッガはモジュールXのモジュール情報をロードして，Yという名前のルーチンの情報を探し出します。これまで，デバッガは，モジュールXの情報がロードされていない場合はエラーにしていましたが，今回のリリースからはそれを自動的にロードするようになりました。

2.5.3 Heap Analyzer が使用できるようになった

OpenVMS I64 でも Heap Analyzer が使用できるようになりましたが，起動方法は Alpha システムとは異なります。新しい起動コマンド START HEAP_ANALYZER をデバッガのコマンド行プロンプト (DBG>) から入力することによって， OpenVMS I64 デバッガから Heap Analyzer を起動できます。詳細は，『HP OpenVMS デバッガ・マニュアル』のヒープ・アナライザの章を参照してください。

2.5.4 SHOW STACK コマンドの新しい修飾子

SHOW STACK コマンドでは/START_LEVEL=*n*修飾子が使用できるようになりました。この修飾子は SHOW STACK に対して，コール・フレーム・レベル*n*の位置からスタック情報の表示を開始するように指示します。

たとえば，フレーム 3 のスタック情報だけを表示したい場合には，次のコマンドを実行します。

```
DBG> SHOW STACK/START=3 1
```

4 番目と 5 番目のスタック・フレームの詳細情報を表示するには，次のコマンドを実行します。

```
DBG> SHOW STACK/START=4 2
```

2.5.5 型のないストレージ位置のデフォルト・データ型の変更

型のないストレージ位置のデフォルト・データ型がロングワード (32 ビット) からクォドワード (64 ビット) に変更されました。データ型情報があるストレージ位置は，関連する型に応じて，これまでどおり表示されます。

2.5.6 SHOW SYMBOL コマンドでのオーバーロード・シンボル・サポートの改善

SHOW SYMBOL コマンドが機能拡張されて，オーバーロード・シンボル名を認識するようになりました。これまでは，オーバーロード名が表示されるだけでした。この機能拡張によって，名前とその名前に関連付けられた情報が表示されるようになりました。たとえば，次のとおりです。

```
DBG> show symbol/full g
overloaded name C::g
    routine C::g(char)
        type signature: void g(char)
        address: 132224, size: 128 bytes
    routine C::g(long)
        type signature: void g(long)
        address: 132480, size: 96 bytes
```

2.5.7 Ada 言語の予備的サポート

OpenVMS I64 デバッガは機能拡張されて， Ada 言語をサポートするようになりました。ただし，今回のサポートは予備的なものであり， OpenVMS の将来のリリースでは機能拡張される予定です。現在サポートされている機能は次のとおりです。

- 単純な型，配列，レコード
- バイアス型
- 密封型
- パッケージ
- オーバーロード・ルーチンとデータ・シンボル
- ユーザ定義演算子

2.6 利用可能になったデバッガ

OpenVMS Version 8.2-1 Integrity サーバ・オペレーティング・システムには，以下のデバッガが用意されています。

- HP DELTA デバッガが OpenVMS I64 システムでも利用可能になりました。

  OpenVMS I64 で DELTA デバッガを使用する方法は， OpenVMS Alpha システムで DELTA デバッガを使用する方法と同じです。 OpenVMS I64 の DELTA デバッガで受け付けられるコマンドは， OpenVMS Alpha の場合と同じです。ただし， OpenVMS I64 の DELTA で認識されるレジスタ名は， OpenVMS I64 の XDELTA で認識されるのと同じ Intel® Itanium®のレジスタ名です。

- System Code Debugger (SCD)

SCD を使用すると，任意の割り込み優先順位レベル (IPL) で動作する，ページングされないシステム・コードとデバイス・ドライバをデバッグすることができます。

- System Dump Debugger (SDD)

  SDD を使用すると， SCD のコマンドとセマンティクスを使用して，システム・ダンプを解析することができます。

2.7 新しい SDA コマンドによる Fibre Channel のデータ取得

OpenVMS Version 8.2-1 の新しい System Dump Analyzer (SDA) のコマンド FC PERFORMANCE を使用して， DGA デバイスの I/O 性能特性を表示することができます。この機能は， OpenVMS Alpha Version 8.2 および OpenVMS I64 Version 8.2 でも利用できます。また，このコマンドは OpenVMS Alpha Version 7.3-2 でも，FIBRE_SCSI_V400 Fibre Channel パッチ・キットをインストールすれば利用できます。これらでサポートされるバージョンの Fibre Channel ドライバは，構成されている各ディスクに関する性能データを配列形式で記録します。

この新しい SDA コマンドを使用して，指定した DGA デバイスやシステム内に構成されている全 DGA デバイスの I/O 性能特性を表示することができます。デバイス名を省略した場合には，デバイスが現在マウントされているかどうかにかかわらず， 0 以外のデータを持つすべての DGA ディスクの性能データが表示されます。 FC PERFORMANCE 配列には， LUN ごとの入出力待ち時間，入出力サイズ，入出力方向 (読み込み/書き込み) が記録されます。

入出力待ち時間は，リクエストがホスト FC アダプタのキューに入れられてから完了割り込みが発生するまでの時間で計測されます。最近のディスク・ドライブの平均入出力待ち時間は 5 ～ 10 ミリ秒です。ディスク・コントローラ (HSG/EVA/MSA/XP) 内のキャッシュと物理ディスク・ドライブ内のキャッシュによって，アクセス時間が大幅に短くなることがあります。

FC PERFORMANCE コマンドは，デフォルトでは，/SYSTIME 修飾子を使用して待ち時間を計測します。/SYSTIME 修飾子では EXE$GQ_SYSTIME が使用されますが，これは 1 ミリ秒ごとに更新されます。入出力が 1 ミリ秒より短い時間で完了した場合には， 0 秒で完了したように見えます。/SYSTIME を使用したときには， 1 ミリ秒より短い時間で完了した入出力操作は<2us の欄に表示されます。ただし， us はマイクロ秒を表わしています。

精度を高めるためには/RSCC 修飾子を指定します。この場合には，システム・サイクル・カウンタ・タイマが使用されます。コマンド修飾子 (このタイマも含む) を，表 2–1 に示します。

表 2–1 FC PERFORMANCE コマンドの修飾子

修飾子	説明
[device-name]	性能特性を表示するデバイス。1 つの DGA デバイス名だけを指定できます。名前を省略した場合には，システムに構成されているすべての DGA デバイスの性能データが表示されます。ただし，配列に 0 以外の値が含まれている場合に限ります。これには，現在マウントされていない DGA デバイスの性能データも含まれます。
[/RSCC | /SYSTIME]	2 種類の時間修飾子を指定できます。/RSCC 修飾子の場合は，PAL 呼び出し Read System Cycle Counter が使用されて，CPU サイクル・カウンタが読み取られます。これで正確な時間が計測できますが，時間のかかる PAL 呼び出しを 1 回の入出力操作につき 2 回行うため，処理時間が長くなります。1 ミリ秒より短い精度で時間を表示する必要がある場合には，この修飾子を使用してください。/SYSTIME 修飾子の場合は，1 ミリ秒ごとに更新される OpenVMS システム時間が使用されます。これがデフォルトです。
[/COMPRESS]	0 だけからなる欄の画面表示を抑止します。
[/CSV]	出力を CSV (comma-separated value) 形式でファイルに書き込みます。このファイルは Microsoft® Excel のスプレッドシートに読み込んだり，グラフに加工することができます。
[/CLEAR]	性能配列をクリアします。この修飾子は入出力性能を試験している場合に便利です。新しい試験を開始する前に，計測するデバイスの性能配列をクリアし，試験が終了したらすぐに性能配列の内容をダンプします。

次に示す例は，SDA コマンドの FC PERFORMANCE に/COMPRESS 修飾子を指定して，デバイス1DGA4321 の読み書き性能を測定したときの出力です。

この出力では，LBC (最初の列の見出し) は論理ブロック数です。各行の LBC は 2 のべき乗になっていることに注意してください。各行は次の行に示される LBC 数までの LBC 数を含んでいます。たとえば，LBC 8 の行は，LBC 8 から LBC 15 までの入出力の回数を含んでいます。256 を超える LBC を持つ入出力はこのマトリックスには表示されませんが，マトリックスの上に表示される "total blocks"の値には含まれています。

各列に表示される回数は，列の見出しに示される時間よりも短い時間で完了した入出力の回数を示しています。たとえば，<2ms の列はこの列内の入出力が完了するまでにかかった時間が 2 ミリ秒より小さく，かつ 1 ミリ秒より大きかったことを意味します。同様に，<4ms の列の入出力は，完了するまでにかかった時間が 4 ミリ秒より小さく，かつ 2 ミリ秒より大きかったことを意味します。例外は，<2us の列で，/SYSTIME 修飾子を指定した場合に，1 ミリ秒より短い時間で完了したすべての入出力の回数が<2us の回数に含められます。

列の見出し<2us，<2ms，<4ms などが，この表示に示されています。いずれかの列で見出しに該当する値がない場合には，/COMPRESS 修飾子が指定されているため，その列は表示されません。デフォルトの/SYSTIME 修飾子の代わりに，/RSCC 修飾子を指定した場合には，追加の見出し<4us，<8us，<16us，<256us が表示されます。

```
SDA> FC PERFORMANCE $1$DGA4321/COMPRESS

Fibre Channel Disk Performance Data
-----------------------------------

$1$dga4321 (write)

Using EXE$GQ_SYSTIME to calculate the I/O time

    accumulated write time = 2907297312us

    writes = 266709

    total blocks = 1432966

I/O rate is less than 1 mb/sec

LBC  <2us  <2ms  <4ms  <8ms <16ms <32ms <64ms <128ms <256ms <512ms <1s
===  ===== ====  ====  ==== ===== ===== ===== ====   =====  ===== ====
  1   46106 20630 12396 13605 13856 15334 14675 8101    777     8    -   145488
  2      52    21     8     9     5     5     6    1      2     -    -      109
  4   40310 13166  3241  3545  3423  3116  2351  977     88     -    -    70217
  8    2213  1355   360   264   205   225   164   82      5     -    -     4873
 16   16202  6897  3283  3553  3184  2863  2323 1012    108     -    1    39426
 32     678   310    36    39    47    44    33   27      6     -    -     1220
 64     105    97    18    26    41    43    42   24      7     -    -      403
128     592  3642   555    60    43    31    23    9      2     -    -     4957
256       -     9     7     -     -     -     -    -      -     -    -       16

     106258 46127 19904  21101 20804 21661 19617 10233  995     8    1   266709

Fibre Channel Disk Performance Data
-----------------------------------

$1$dga4321 (read)

Using EXE$GQ_SYSTIME to calculate the I/O time

    accumulated read time = 1241806687us

    reads = 358490

    total blocks = 1110830

I/O rate is less than 1 mb/sec

LBC  <2us  <2ms  <4ms  <8ms  <16ms <32ms <64ms <128ms <256ms <512ms <2s
=== ===== =====  ===== ===== ===== =====  ===== ===== ====== ====== ====
  1  46620 12755  6587  7767  3758  2643   1133   198     5    -    -    81466
  2    574   134    66   158    82    20     21     4     1    -    -     1060
  4 162060 35896 20059 18677 15851 11298   5527  1300    25    2    1   270696
  8    355    79    46    97    59    36     28    10     -    -    -      710
```

```
 16    241   103    32   150    77    24     13     1     -     -     -      641
 32    916   355    76   302   316    61     25    10     -     -     -     2061
 64    725   380    64   248   140    17     10     3     -     -     -     1587
128     13    22    13    36    21     6      -     -     -     -     -      111
256     10    41    28    15    49    13      2     -     -     -     -      158

    211514 49765 26971 27450 20353 14118   6759  1526    31    2    1   358490
SDA>
```

2.8 メモリ常駐セクションに対する，より大きな粒度ヒントの指定

I64 プラットフォームでの OpenVMS のメモリ常駐セクションでは，ハードウェアでサポートされるすべてのページ・サイズを使用できるようになりました。 OpenVMS のデフォルトである 8 KB より大きなページ・サイズを使用することで，アプリケーションの性能を大幅に改善することができます。これは，大量のデータをカバーするために必要な変換キャッシュのエントリが少なくてすむからです。

Alpha システムと I64 システムでは， OpenVMS は粒度ヒント機能を使って，デフォルトのシステム・ページ・サイズより大きなページ・サイズを実現します。 Alpha システムでは，最大有効ページ・サイズは 4 MB で，現在の I64 システムでは，最大有効ページ・サイズは 4 GB です。ユーザは， SYSMAN RESERVED_MEMORY コマンドを使用して，前もってメモリを割り当てることにより，メモリ常駐セクションに対して粒度ヒント機能を適用することができます。粒度ヒント機能を使用するときには， Alpha システムまたは I64 システムでメモリ常駐セクションを使用する必要があります。 SYSMAN RESERVED_MEMORY コマンドを使用して次回のシステム・ブート時のメモリを前もって割り当てます。

アラインメントの要件が次のように変更されたことに注意してください。すなわち，より大きなページ・サイズを使用する場合には，仮想アドレスと物理アドレスの両方を適切にアラインメントしなければなりません。 Alpha システムでは，セクションの開始アドレスをページ・テーブルの境界にマッピングすれば，十分なアラインメントが保証されました。しかし， I64 システムではこれだけでは不十分です。

OpenVMS I64 システムでは，メモリ常駐セクションを使用するときには，次の手順に従うことをお勧めします。

1. 必要に応じて， SYSMAN コマンドの RESERVED_MEMORY を使用して，メモリを前もって割り当てます。
2. SYS$UPDATE:AUTOGEN GETDATA REBOOT プロシージャを実行して，少なくなった汎用メモリの量に合わせてシステムをチューニングします。

3. すべてのメモリ常駐セクションで，共用ページ・テーブル領域を使用します。共用ページ・テーブルがブート時に割り当てられていなくても，共用ページ・テーブル領域を使用することで，仮想アドレス空間内の領域の最適なアラインメントが保証されます。

   共用ページ・テーブル領域には開始アドレスを指定しないでください。そうしておけば OpenVMS は，この領域内にマッピングされるセクションに対して，最大の粒度ヒント・ファクタを設定できる開始アドレスを選択できます。

4. セクションをマッピングするときには， SEC$M_EXPREG フラグを使用します。これによって， OpenVMS はそのセクションに対して最適なアラインメントの開始アドレスを選択することができます。

5. メモリ常駐セクションが作成できたら，必ずセクション全体をマッピングします。セクションの一部しかマッピングしなかった場合には，共用ページ・テーブルや，大きなページ・サイズを使用できません。

6. 共用ページ・テーブル領域やメモリ常駐セクションの開始アドレスを指定しているアプリケーションを変更できない場合には，アプリケーションが失敗する可能性があります。回避策としては，動的システム・パラメータ MMG_CTLFLAGS のビット 4 (10 進数で 16) をセットして， I64 システムのページ・サイズを Alpha システムでサポートされる最大の粒度ヒント・サイズに制限する，特殊な動作モードを選択します。

以降の項では，メモリ常駐セクションに対するより大きな粒度ヒントの指定に関する追加の変更を説明します。

2.8.1 MMG_CTLFLAGS システム・パラメータに追加されたフラグ

次の表には，今回のリリースで導入されたビット 4 も含め，このパラメータで現在定義されているすべてのビット・マスク値を示しています。

ビット	意味
0	このビットがセットされていると，メモリの再利用が有効になります。すなわち，定期的に実行され，それ以外はアイドル状態のプロセスをトリミングすることで，再利用が行われます。再利用は，空きリストと修飾リストの合計サイズが，FREEGOAL の値の 2 倍より小さくなったときに行われます。この機能は，このビットがセットされていない場合には，無効になっています。
1	このビットがセットされていると，メモリの再利用が有効になります。すなわち， LONGWAIT の秒数を超えてアイドル状態になっているプロセスをスワップアウトすることで，再利用が行われます。再利用は，空きリストのサイズが，FREEGOAL の値より小さくなったときに行われます。この機能は，このビットがセットされていない場合には，無効になっています。
2	AlphaServer 4100 システムでの遅延メモリ・テストを制御します。このビットがセットされていない場合 (デフォルト) には， OpenVMS はメモリをバックグラウンドでテストします。このビットがセットされていると，ブートストラップ・プロセスの際にすべてのメモリがテストされます。
3	OpenVMS 用に予約。 0 でなければならない。

ビット	意味
4	このビットがセットされていない場合 (デフォルト) には， I64 のハードウェアでサポートされているすべてのページ・サイズをメモリ常駐セクションのマッピングで使用することができます。このビットがセットされていると， I64 システムのページ・サイズは Alpha システムで使用できる最大粒度ヒント・ファクタ (512 *システム・ページ・サイズ) に制限されます。
5-7	将来のために予約。

2.8.2 システム・サービスの変更点

以下のリストには，メモリ常駐セクションが増加したことに伴って項目コードが変更されたシステム・サービスが含まれます。

- $CREATE_REGION_64
  - length_64

    以下の情報が，説明の最初の段落の後に挿入されます。

    複数のメモリ常駐セクションをこの領域にマッピングする場合には，すべてのセクションを収容できるだけでなく，次のセクションを最大の有効ページ・サイズ (粒度ヒント) に合わせてアラインメントするために必要となる空間も収容できるだけの十分に大きな長さを指定します。この要件は，マッピングするすべてのセクションの合計の 2 倍の領域を割り当てるだけで満たすことができます。
  - start_va_64

    説明の最後の段落は次のように変更されました。

    フラグ VA$M_SHARED_PTS をセットして，この引数を指定する場合には，指定する開始アドレスは，自然なページ・テーブル境界と，セクションのマッピングで使用する最大のページ・サイズのうち，大きい方にアラインメントされている必要があります。アラインメントがページ・テーブル境界よりも小さい場合には，$CREATE_REGION_64 サービスはエラーを返します。アラインメントが，セクションで使用する最大のページ・サイズよりも小さい場合には，セクションをマッピングしようとするとエラーになります。

    開始アドレスを指定しない場合には， OpenVMS は自動的に正しいアラインメントを保証します。
- $CRMPSC_GDZRO_64
  - section_offset_64

    最後の段落は次のように変更されました。

    共用ページ・テーブル領域を region_id_64 引数で指定する場合には，section_offset_64 は，少なくとも CPU 固有のページ・テーブルのページによってマッピングできるバイト数の整数倍の大きさでなくてはなりません。 I64 システムの場合には，セクション・オフセットのアラインメントも，仮想アド

レス空間をこのオフセットにマッピングするために使用されるページ・サイズの整数倍でなければなりません。

I64 システムで共用ページ・テーブルを使用する場合には，メモリ常駐セクションの部分的なマッピングは避けるようにしてください。これを避けることができない場合には，システム・パラメータ MMG_CTLFLAGS のビット 4 をセットして，有効ページ・サイズを，ページ・テーブルのページでマッピングできるバイト数に制限してください。

- start_va_64

  説明の最後の段落は次のように変更されました。

  フラグ VA$M_SHARED_PTS をセットして，この引数を指定する場合には，指定する開始アドレスは，自然なページ・テーブル境界と，セクションのマッピングで使用する最大のページ・サイズのうち，大きい方にアラインメントされている必要があります。アラインメントがページ・テーブル境界よりも小さい場合には，$CREATE_REGION_64 サービスはエラーを返します。アラインメントが，セクションで使用する最大のページ・サイズよりも小さい場合には，セクションをマッピングしようとするとエラーになります。

  開始アドレスを指定しない場合には， OpenVMS は自動的に正しいアラインメントを保証します。

- $MGBLSC_64

  - section_offset_64

    最後の段落は次のように変更されました。

    共用ページ・テーブル領域を region_id_64 引数で指定する場合には，section_offset_64 は，少なくとも CPU 固有のページ・テーブルのページによってマッピングできるバイト数の整数倍の大きさでなくてはなりません。 I64 システムの場合には，セクション・オフセットのアラインメントも，仮想アドレス空間をこのオフセットにマッピングするために使用されるページ・サイズの整数倍でなければなりません。

    I64 システムで共用ページ・テーブルを使用する場合には，メモリ常駐セクションの部分的なマッピングは避けるようにしてください。これを避けることができない場合には，システム・パラメータ MMG_CTLFLAGS のビット 4 をセットして，有効ページ・サイズを，ページ・テーブルのページでマッピングできるバイト数に制限してください。

  - start_va_64

    説明の最後の段落は次のように変更されました。

    フラグ VA$M_SHARED_PTS をセットして，この引数を指定する場合には，指定する開始アドレスは，自然なページ・テーブル境界と，セクションのマッピングで使用する最大のページ・サイズのうち，大きい方にアラインメントされている必要があります。アラインメントがページ・テーブル境界よりも小さい場合には，$CREATE_REGION_64 サービスはエラーを返します。アライ

ンメントが，セクションで使用する最大のページ・サイズよりも小さい場合には，セクションをマッピングしようとするとエラーになります。

開始アドレスを指定しない場合には， OpenVMS は自動的に正しいアラインメントを保証します。

2.9 Partition Manager

Partition Manager (parmgr) は，すべての nPartitions 管理機能を 1 ヶ所で実行します。 Partition Manager は， nPartitions の動的再構成，電源投入，電源切断，作成，削除，変更を，スムーズかつ管理の行き届いた操作のもとで実行するのに必要なツールをシステム管理者に提供します。

Partition Manager は， HP サーバ・システム上の nPartitions を構成し，管理するための便利な GUI をシステム管理者に提供します。 Partition Manager を使用すれば，コンプレックスの構成作業を，コマンドやパラメータを覚えることなく実行できます。 nPartitions，セル， I/O シャーシ，その他のコンポーネントをグラフィカル・ディスプレイで選択して，メニューからアクションを選択するだけです。

Partition Manager は， HP-UX システムと Microsoft® Windows®システムの両方で動作します。どちらかのバージョンの Partition Manager を使用すれば， OpenVMS I64 Version 8.2-1 用の nPartitions を管理することができます。

Partition Manager を使用すると，以下の作業が実行できます。

- nPartitions の作成，変更，削除
- コンプレックスの nPartitions 構成の検証
- コンプレックスの構成とハードウェアに関する潜在的な問題点の確認
- コンプレックス上のハードウェア・リソースの管理

Partition Manager は，次の Web サイトからダウンロードできる無償の製品です。

```
http://www.docs.hp.com/en/PARMGR2/download.html
```

Partition Manager の詳細は，『HP **システムパーティションガイド**: nPartitions **の管理作業**』を参照してください。 Partition Manager Web サイトのメイン・ページまたは次の Web サイトから，オンライン・ヘルプをダウンロードできます。

```
http://www.docs.hp.com/en/PARMGR2/features2.html
```

URL では大文字と小文字が区別されるため，アドレスはここに示したとおりに入力してください。

2.10 新しい省電力機能

I64 システムでは，アイドル時に CPU を「省電力モード」にすることができます。これによって電力消費を抑えて，システムのエネルギー・コストを削減することができます。 OpenVMS I64 Version 8.2-1 は，システム・パラメータ CPU_POWER_MGMT と CPU_POWER_THRSH の設定にもとづいて，この機能をサポートします。これらの 2 つのパラメータの説明は以下のとおりです。

CPU_POWER_MGMT (**動的**)
この動的パラメータに 1 をセットするとオン (デフォルト)， 0 をセットするとオフを意味します。

CPU_POWER_THRSH パラメータの値を超過して， I64 プロセッサがアイドル状態であれば，オペレーティング・システムは I64 プロセッサを省電力モードにします。 OpenVMS I64 がこの操作を行うのは， CPU_POWER_MGMT がオンになっているときだけです。 CPU は割り込みを受け付けると，通常の電力モードに戻ります。

CPU_POWER_THRSH (**動的**)
CPU_POWER_THRSH は，パーセントで表現される動的パラメータです。

OpenVMS I64 は各 CPU が動作しているかどうかを一定時間にわたって監視します。 CPU_POWER_MGMT がオンになっていて CPU_POWER_THRSH で示される期間 CPU がアイドルの場合には， CPU は省電力モードになります。 CPU は割り込みを受け付けると，通常の電力モードに戻ります。

迅速な応答が必要なリアルタイム処理をサポートしているシステムでは，この機能はオフにしておくことをお勧めします。この機能を使用すると，性能が多少低下する可能性があるためです。

詳細は，次の Web サイトにある『Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture』を参照してください。

`http://www.intel.com/design/itanium/manuals/iiasdmanual.htm`

2.11 PPL ユニット割り当てツール

License Management Facility には， OpenVMS の Per Processor Licenses (PPL) のライセンス・ユニットの管理に役に立つユニット割り当てツールが用意されています。このユニット割り当てツールでは， CSV (comma-separated values) 形式のテキスト・ファイルを使用して，ライセンス要件とクラスタ内での割り当てを記述します。

CSV 形式のファイルが使用されているため，任意のテキスト・エディタやスプレッドシート・プログラムを使用して更新した後，このツールに戻すことができます。このユニット割り当てツールを起動するには，次のコマンドを実行します。

```
$@SYS$EXAMPLES:LMF$PPL_UNITS_ASSIGNMENT.COM
```

詳細は， PPL のオンライン・ヘルプを参照してください。

2.12 System Dump Analyzer (SDA) ユーティリティ

ここでは， OpenVMS I64 Version 8.2-1 オペレーティング・システムに用意された SDA の新機能について説明します。 Fibre Channel の性能情報を取得するための SDA コマンドの説明は，第 2.7 節を参照してください。

2.12.1 SDA COPY コマンドの補足情報

ダンプ解析の際には，システム・クラッシュ時にダンプ・ファイルに書き込むことができないデータが利用できると役に立ちます。このようなデータには，ファイルの識別に関連する完全なファイル指定や， OpenVMS I64 の場合には，プロセス内で起動されていたイメージのアンワインド・データなどがあります。

ダンプが書き出されたシステムでダンプを解析している場合には， COLLECT コマンドを使ってこれらのデータを収集し，現在の SDA セッションの中で使用することができます。ダンプを別のシステムにコピーして解析する場合には， COPY /COLLECT コマンドを実行してデータを収集し，書き込まれるダンプのコピーにそのデータを追加することができます。 COLLECT コマンドを実行した後で COPY /COLLECT コマンドを実行した場合には，収集済みのデータがダンプのコピーに追加されます。

デフォルトでは，システム・クラッシュ時に書き込まれたもとのダンプのコピーには，収集データが追加されます。この動作を無効にするには COPY/NOCOLLECT を使用します。逆に，以前に収集データ無しで SDA を使用してコピー (COPY /NOCOLLECT) したダンプのコピーには，収集データは含まれていません。この動作を無効にするには COPY/COLLECT を使用します。

収集データがすでに追加されているダンプをコピーすると，収集データは必ず含まれます。

すべてのファイル・データやアンワインド・データを正しく収集するには，システム・クラッシュ発生時にマウントされていたすべてのディスクを再マウントして， SDA を実行するプロセスからアクセスできるようにしておく必要があります。起動の最初の段階でダンプの内容を保存するために SDA を起動していて (たとえば， CLUE$SITE_PROC (『HP OpenVMS System Analysis Tools Manual』の第 2.2.3 項を参照) を使用)，バッチ・ジョブが実行されるまでディスクをマウントしていなかった場合には， CLUE$SITE_PROC コマンド・プロシージャ内で COPY/NOCOLLECT コマンドを使ってください。すべてのディスクがマウントされれば COPY/COLLECT コマンドを実行して，ファイル・データやアンワインド・データを保存することができます。

2.12.2 COPY コマンドの新しい修飾子

SDA COPY コマンドの新しい修飾子は以下のとおりです。

- /[NO]COLLECT

  /COLLECT 修飾子を指定すると，SDA はファイル識別データとアンワインド・データ (OpenVMS I64 システムの場合) を現在のシステムから収集し，書き込まれるダンプのコピーに追加します。/NOCOLLECT 修飾子を指定すると，SDA がデータを収集して追加するのを禁止します。デフォルトなどの詳細は，コマンドの説明を参照してください。

- /LOG

  COPY コマンドの進行状況を表示します。たとえば，選択ダンプでコピーされるプロセスの名前や，I64 で COPY/COLLECT を実行している場合にはアンワインド・データがダンプ・コピーに追加されているイメージの名前などが，表示されます。

2.12.3 新しい SDA コマンド

ここでは新しい SDA COLLECT コマンドについて説明します。

COLLECT

Alpha と I64 でファイル識別情報をファイル名に変換するためのデータを収集し，I64 でのみプロセスのアンワインド・データを収集します。

フォーマット

COLLECT *[/qualifier...]*

パラメータ

なし

修飾子

/LOG
COLLECT コマンドの進行状況を表示します。たとえば，スキャン中のプロセスの名前や， I64 ではアンワインド・データを収集しているイメージの名前などが表示されます。

/UNDO
以前に実行した COLLECT コマンドで収集されたすべてのファイル・データとアンワインド・データを SDA メモリから削除します (COLLECT/UNDO は解析中のダンプ・ファイルにすでに追加されているファイル・データやアンワインド・データには影響を与えません)。

説明

ダンプを解析する際には，システム・クラッシュ時にダンプ・ファイルに書き込むことができないデータが利用できると役に立ちます。このようなデータには，ファイルの識別に関連する完全なファイル指定や， OpenVMS I64 の場合には，プロセス内で起動されていたイメージのアンワインド・データなどがあります。

ダンプが書き出されたシステムでダンプを解析している場合には， COLLECT コマンドを使ってこれらのデータを収集し，現在の SDA セッションの中で使用することができます。ダンプを別のシステムにコピーして解析する場合には， COPY /COLLECT コマンドを実行してデータを収集し，書き込まれるダンプのコピーにそのデータを追加することができます。 COLLECT コマンドを実行した後で COPY

/COLLECT コマンドを実行した場合には，収集済みのデータがダンプのコピーに追加されます。

すべてのファイル・データやアンワインド・データを正しく収集するには，システム・クラッシュ発生時にマウントされていたすべてのディスクを再マウントして，SDA を実行するプロセスからアクセスできるようにしておく必要があります。

例

```
SDA>  COLLECT
%SDA-W-DISKNOACC, no access to _$30$DKB100: for file and/or unwind data
%SDA-W-FILENOACC, no access to _$30$DKB0: (7709,1,0) for unwind data
-SYSTEM-W-NOSUCHFILE, no such file
```

この例では，ファイル・データやアンワインド・データを収集するときには，システム・クラッシュの発生時にマウントされていたディスク30DKB100 が利用できません。また，_30DKB0 上のファイル識別情報 (7709,1,0) を持つイメージのアンワインド・データは，残されていないため収集できません。

2.12.4 SHOW CALL_FRAME のパラメータの変更

SHOW CALL_FRAME コマンドのstarting addressパラメータの説明は次のように変更されました。

Alpha の場合には，プロシージャ呼び出しフレームの開始アドレスを表現する式が表示されます。 pthread を使用しているプロセスの場合は，次の SDA コマンドを使用してすべての pthread の開始アドレスを表示することができます。

```
pthread thread -o u
```

デフォルトの開始アドレスは， SDA の現在のプロセスのフレーム・ポインタ (FP) レジスタの内容です。

I64 の場合には，このアドレスは以下のいずれかになります。

- フレームの呼び出しコンテキスト・ハンドル。
- 例外フレームのアドレス。これは次のコマンドを実行したのと同じです。

  ```
  SHOW CALL_FRAME /EXCEPTION_FRAME=address
  ```

- TEB (Thread Environment Block) のアドレス。プロセスのすべての TEB のリストに対して，次の SDA コマンドを実行します。

  ```
  pthread thread -o u
  ```

デフォルトの開始アドレスは， SDA の現在のプロセス内の現在のプロシージャの呼び出しコンテキスト・ハンドルです。

2.13 Traceback 機能

I64 システムの Traceback 機能には，新しいシンボル化ルーチン TBK$I64_SYMBOLIZE が含まれます。このルーチン (いつでも呼び出すことができる) を使用することにより，実行可能 pc 値とそのプログラム・コードを生成したソース・コードをアプリケーションで対応付けることができます。

このルーチンの説明は以下のとおりです。

TBK$I64_SYMBOLIZE
Traceback シンボル化ルーチン TBK$I64_SYMBOLIZE を使用すると，アプリケーション・プログラムでは，現在実行中のアプリケーション内の特定のアドレスの実行可能プログラム・コードを生成するために使用された，アプリケーション・プログラムのソース・コードを確認したりシンボル化することができます。

呼び出し規則

#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	「トレースバック」するプロセス内の実行可能命令。値渡し。

出力

filename_desc	文字列ディスクリプタ。プログラム・コードが格納された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。プログラム・コードが格納されたテキスト・ライブラリ・ファイルの名前が返される。参照渡し。該当する場合のみ返される。
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 で指定されるファイルの n 番目の行を示す) 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

正常終了時には TBK$_NORMAL が返されます。エラーが発生したときには，他の失敗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$I64_SYMBOLIZE が追加されました。このルーチンは，いつでも呼び出し可能で，例外によって通知された状況をアプリケーションが別の方法で処理できるようにします。通常のトレースバック処理では「トレースバック・スタック」情報を生成し，SYS$OUTPUT に出力します (つまり，一連の PC 値をスタック・レベルごとに 1 つ)。アプリケーションで独自にトレースバック情報を処理したい場合には，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の一部またはすべて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・コード行のリスト上の行番号やレコード番号が設定されます。戻り情報には，イメージ名，モジュール名，ルーチン名も設定できます。イメージ相対値またはモジュール相対値を示す相対 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，その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ときには，** Linker **オプション・ファイル内に次の記述を含めてください。**

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

3

OpenVMS V8.2-1 リリース・ノート

この章では OpenVMS I64 Version 8.2-1 の注意事項について示します。 OpenVMS Version 8.2 のリリース・ノートの大半は本リリースでも適用されるため，システムのインストールまたはアップグレードを行う前に，『HP OpenVMS V8.2 リリース・ノート[翻訳版]』の中の OpenVMS I64 に関係する注意事項(つまり， Alpha のみと書かれていないすべての注意事項) と，この章の注意事項をお読みください。

ただし，以下に示す OpenVMS I64 Version 8.2 のリリース・ノートは， Version 8.2-1 には適用されません。

- 第 1.9.8 項「DECnet-Plus の新しいバージョンが必要」
- 第 2.12 節「HP DECnet-Plus for OpenVMS: X.25 データ・リンクがサポートされていない」
- 第 4.10 節「EFI ツール: VMS_SHOW DUMP_DEV エラー」
- 第 5.28.1 項「全般的な問題点と回避方法 (I64 のみ)」

  以下の OpenVMS Debugger の問題点は修正されました。

  – デフォルトの基数による配列のインデックス値のプリント

  – ローテートするレジスタの表示

3.1 アダプタについての注意事項

ここでは OpenVMS Version 8.2-1 でサポートされるアダプタについての注意事項を説明します。

3.1.1 ゾーンが構成された SAN 環境での A9782A， A9784A， AB465A の制約

Fibre Channel 環境では，ゾーン構成を使用して SAN 内の種々の環境にわたってパーティションを配置することができます。コンボ・アダプタ A9782A， A9784A, AB465A を他の Fibre Channel アダプタを持つゾーンにマッピングすると，システムはブートできなくなります。

3.1.2 Fibre Channel の EFI ドライバとファームウェアの要件

OpenVMS I64 Version 8.2-1 では， HP A6826A Fibre Channel ホストベース・アダプタとその改良版を使用するためには， EFI ドライバのバージョン 1.46 以降， RISC ファームウェアのバージョン 3.03.154 以降が必要です。ファームウェアのアップデート方法については，次の Web サイトを参照してください。

```
http://www.hp.com/support/itaniumservers
```

Fibre Channel デバイスの構成とブート方法については，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』の付録 C を参照してください。

3.2 関連製品のサポート

Software Public Rollout Reports for OpenVMS には， OpenVMS I64 用の Layered Product Library キット (DVD) で提供されている弊社のソフトウェア製品の情報が記載されています。このレポートには製品名とバージョン，製品をサポートするために必要なオペレーティング・システムのバージョン，製品の出荷日が示されています。レポートの情報は今後も追加され，変更される可能性があります。このレポートは公開されており，毎月更新されます。レポートの情報はたえず変化するため，このリリース・ノートには含まれません。

Software Public Rollout Reports for OpenVMS は，次の Web サイトで参照できます。

```
http://h71000.www7.hp.com/openvms/os/swroll/
```

インターネットにアクセスできない場合は，四半期ごとに提供される Software Products Libraries (SPL) でオペレーティング・システムのサポート情報を参照できます。この情報は， SPL の次のファイルにあります。

```
[README]SW_COMPAT_MATRIX.PS
[README]SW_COMPAT_MATRIX.TXT
```

また， Software Public Rollout Reports は弊社のサポート担当者から入手することもできます。

3.3 クラスタ互換性パッチ・キット

OpenVMS Version 8.2-1 システムを既存の OpenVMS Cluster システムに導入する場合は，事前に旧バージョンの OpenVMS で動作しているシステムにいくつかのパッチ・キット (修正キット) を適用する必要があります。これらのキットは特定のバージョンに対応していることに注意してください。

表 3–1 にリストしたバージョンは，動作が保証された構成でサポートされます。動作が保証された構成についての詳細は，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。

表 3–1 に，パッチ・キットを必要とする機能とパッチ・キットのファイル名をリストします。各パッチ・キットには対応する readme ファイルが付属しています。readme ファイルの名前はパッチ・キットの名前にファイル拡張子.README を付加した形式になっています。

パッチ・キットは次の Web サイトからダウンロードすることができます。または，弊社のサポート担当に連絡して，ご使用中のシステムに適したメディアに格納されたパッチ・キットを入手してください。

http://www2.itrc.hp.com/service/patch/mainPage.do

注意

パッチ・キットは必要に応じて定期的に更新されます。必ず，機能ごとの最新のパッチ・キットを使用してください。バージョン番号はキットの readme **ファイルに記載されています。各キットの最新バージョンは，** Web **サイトで提供されているバージョンです。**

表 3–1 クラスタ互換性のために必要なパッチ・キット

機能	パッチ・キットのファイル名
OpenVMS Alpha Version 7.3-2	
ここにリストされているパッチ・キットを除く大部分のパッチ・キットを含むアップデート・キット	VMS732_UPDATE-V0400
C RTL	VMS732_ACRTL-V0100
DCL	VMS732_DCL-V0300
ドライバ	VMS732_DRIVER-V0200
Fibre Channel と SCSI	VMS732_FIBRE_SCSI-V0600
グラフィックス	VMS732_GRAPHICS-V0300
入出力デバイス構成	VMS732_IOGEN-V0100
ジョブ制御	VMS732_JOBCTL-V0100
LAN	VMS732_LAN-V0300
Math RTL	VMS732_AMATHRTL-V0100
PCSI	VMS732_PCSI-V0100
SYSMAN その他	VMS732_MANAGE-V0300
シャドウィング	VMS732_SHADOWING-V0400[1]

[1]ご使用中の構成でこのソフトウェアを使用している場合には，このキットが必要です。

(次ページに続く)

表 3–1 (続き) クラスタ互換性のために必要なパッチ・キット

機能	パッチ・キットのファイル名
OpenVMS Alpha Version 7.3-2	
システム	VMS732_SYS-V0700
VERIFY	VMS732_VERIFY_V0100
OpenVMS Alpha Version 8.2	
CPU270F 機能	VMS82A_CPU270F-V0100
DECnet-Plus for OpenVMS Alpha ECO1	DNVOSIECO01_V82[1]
ドライバ	VMS82A_DRIVER-V0100
グラフィックス	VMS82A_GRAPHICS-V0100
入出力デバイス構成	VMS82A_IOGEN-V0100
OpenVMS I64 Version 8.2	
アダプタ・サポート (U320 SCSI および Gb Ethernet)	VMS821_AB290A-V0100
DDTM	VMS82I_DDTM-V0100
ドライバ	VMS821_DRIVER-V0100
グラフィックス	VMS82I_GRAPHICS-V0100
入出力デバイス構成	VMS82I_IOGEN-V0100
ライセンス管理機能	VMS82I_LMF-V0100
MANAGE	VMS821_PERFORMANCE-V0100
MX2CPU 機能	VMS821_MX2-V0100
実行時ライブラリ汎用ルーチン	VMS82I_LIBOTS-V0100
Secure Server	VMS82I_SECSRV-V0100
VMS ローダ	VMS821_IVMSLOA-V0100

[1]ご使用中の構成でこのソフトウェアを使用している場合には，このキットが必要です。

3.4 HP OpenVMS Debuggerの Heap Analyzerの問題点と回避策

問題点: Heap Analyzer の起動時，また I64 用のデバッガ上で Heap Analyzer の START コマンド (START HEAP_ANALYZER) を実行したとき，アップコールを有効にしているマルチスレッド・アプリケーションがあるとハングします。

回避策:スレッドまたは AST を使っているアプリケーションに対しては，デバッグ・イベントを設定する前，またはデバッグ・イベントを無効またはキャンセルした後に Heap Analyzer を起動するようにしてください (Heap Analyzer を起動した後，START コマンドがデバッガの制御をユーザに戻した後にイベントを有効化/リセットすることができます)。

3.5 ドキュメントの訂正

ここではOpenVMSドキュメント・セット内の各種マニュアルの訂正について説明しています。

3.5.1 『HP OpenVMS **デバッガ・マニュアル**』: **クライアント/サーバ・インタフェースの訂正**

Version 8.2の『HP OpenVMSデバッガ・マニュアル』には，PCクライアント・インタフェース・キットに関して間違った情報が記載されています。第11.1節と第11.2節に対する訂正を以下に示します。

デバッグ・サーバはOpenVMSシステム上で動作します。クライアントはデバッガに対するユーザ・インタフェースであり，そこからユーザがデバッガ・コマンドを入力しサーバに送られます。サーバはコマンドを実行し，結果がクライアントに送り返されて表示されます。クライアントは，Microsoft Windows 95，Windows 98，Windows NT ®，Windows 2000，Windows XP ®上で動作します。

上記のクライアント・プラットフォームに対する正しいクライアント・キットは次のとおりです。

```
[DEBUG_CLIENTS011.KIT] DEBUGX86011.EXE
```

OpenVMS上で動作するコンポーネントについては，特別なインストール手順はありません。システム管理者は上記のOpenVMSデバッグ・クライアント・キットをOpenVMSディストリビューション・メディアからPATHWORKSシェアやFTPサーバなどPCユーザがアクセスできる場所に移動する必要があります。クライアント・キットは自己解凍形式の.EXEファイルです。

適切な実行可能ファイルをPCに転送したら，そのファイルを実行してデバッグ・クライアントをPCにインストールします。InstallShieldのインストール・プロシージャによって，インストール手順が案内されます。

デフォルトでは，デバッグ・クライアントは`\Program Files\OpenVMS`のDebuggerフォルダにインストールされますが，[参照]をクリックして，別の場所を選択することもできます。

インストール手順によって，OpenVMS Debuggerのプログラム・フォルダが作成されます。そこには，以下の項目へのショートカットが格納されます。

- デバッグ・クライアント
- デバッグ・クライアントのヘルプ・ファイル
- `README`ファイル
- アンインストール・プロシージャ

3.5.2 『HP OpenVMS I/O User's Reference Manual』: PTD$READ の明確化

『HP OpenVMS I/O User's Reference Manual』では，読み込みリクエストの動作が十分に説明されていません。次の段落は，第 6.5.1 項の最初の段落を置き換えるものです。

制御プログラムは，データを擬似端末から読み込むために， PTD$READ ルーチンを使います。 PTD$READ ルーチンが呼び出されると，オペレーティング・システムによって，読み込み操作がキューに登録されます。読み込み操作は，擬似端末が出力しようとする文字があれば終了します。読み込みリクエストは TTDRIVER に対して，返すべきデータがあるかどうかを照会します。データがあった場合には，その文字列が返されます。読み込みリクエストが発行されてもデータが存在しない場合には，読み込みリクエストはキューに登録され，終了するのは後になります。この場合には，ルーチンは少なくとも 1 文字を返します。読み込みリクエストは，出力として返す文字がなくても終了することがあります。 TTDRIVER が出力として返すデータがないことを示し，実際にデータがないという稀な状況では，読み込み操作は 0 バイトのデータで終了します。

以下の段落は，第 D.4.4 項を置き換えるものです。

PTD$READ ルーチンは擬似端末からデータを読み込みます。読み込みリクエストは最小で 1 文字，そして最大でreadbuf_len引数で指定した数の文字を読み込んで終了します。

PTD$READ ルーチンが呼び出されると，オペレーティング・システムによって，読み込み操作がキューに登録されます。読み込み操作は，擬似端末が出力しようとする文字があれば終了します。読み込みリクエストは TTDRIVER に対して，返すべきデータがあるかどうかを照会します。データがあった場合には，その文字列が返されます。読み込みリクエストが発行されてもデータが存在しない場合には，読み込みリクエストはキューに登録され，終了するのは後になります。この場合には，ルーチンは少なくとも 1 文字を返します。読み込みリクエストは，出力として返す文字がなくても終了することがあります。 TTDRIVER が出力として返すデータがないことを示し，実際にデータがないという稀な状況では，読み込み操作は 0 バイトのデータで終了します。

3.5.3 『HP OpenVMS V8.2 新機能説明書』: Librarian ユーティリティの訂正

以下のリリース・ノートは， OpenVMS I64 Librarian ユーティリティに関する訂正情報です。

3.5.3.1 /REMOVE 修飾子の訂正

『HP OpenVMS V8.2 新機能説明書』の第 4.8.2.3 項にある，Librarian の拡張された /REMOVE 修飾子についての説明は誤っています。正しい説明は次のとおりです。

I64 Librarian ユーティリティでは，/REMOVE 修飾子の機能が拡張されました。拡張された形式では削除するシンボルのモジュール・インスタンスを指定できるようになりました。拡張された/REMOVE 修飾子では，LIBRARY コマンドに対して，オブジェクト・ライブラリのグローバル・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。

3.5.3.2 ELF オブジェクト・ライブラリへのアクセスについての訂正

『HP OpenVMS V8.2 新機能説明書』の第 4.8.3.2 項には誤った説明があります。以下の文章はその項の説明を置き換えるものです。

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_MODULE を呼び出すとき，または 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。

3.5.4 『HP OpenVMS RTL Library (LIB$) Manual』の訂正

ここでは， Version 8.2 の『HP OpenVMS RTL Library (LIB$) Manual』に対する追加と訂正を説明します。

3.5.4.1 『HP OpenVMS RTL Library (LIB$) Manual』： LIB$CVT_DX_DX の丸め規則の明確化

『HP OpenVMS RTL Library (LIB$) Manual』の LIB$CVT_DX_DX ルーチンの説明では，「Guidelines for Using LIB$CVT_DX_DX」の下にある下記の段落に丸め規則に関する具体的な説明を追加する必要があります。

結果は，常に，(切り捨てられるのではなく) 丸められます。ただし，次に述べる状況は例外です。精度や範囲が失われることは，変換先のデータ型や NBDS 変換先のサイズによっては本質的に避けられないことに注意してください。変換先のデータ型のせいで精度や範囲が失われてもエラーは表示されません。

この段落は次のように変更する必要があります。

結果は，常に，(切り捨てられるのではなく) 丸められます。ただし，変換元と変換先の両方が NBDS で，スケーリングが要求されていない場合は例外です。この場合については，後述する規則で詳しく説明されます。 LIB$CVT_DX_DX は VAX_ROUNDING 規則を使用します。精度や範囲が失われることは，変換先のデータ型や NBDS 変換先のサイズによっては本質的に避けられないことに注意してください。変換先のデータ型のせいで精度や範囲が失われてもエラーは表示されません。

VAX_ROUNDING 規則の詳細は，CVT$CONVERT_FLOAT の説明を参照してください。

3.5.5 『HP OpenVMS RTL Library (LIB$) Manual』: プラットフォームの制限の明確化

『HP OpenVMS RTL Library (LIB$) Manual』では，以下のルーチンが Alpha と I64 の両方で利用できると説明されていますが，これは誤りです。これらのルーチンは Alpha でしか利用できません。

- LIB$GET_CURR_INVO_CONTEXT
- LIB$GET_INVO_CONTEXT
- LIB$GET_INVO_HANDLE
- LIB$GET_PREV_INVO_CONTEXT
- LIB$GET_PREV_INVO_HANDLE
- LIB$PUT_INVO_REGISTERS

また，『HP OpenVMS RTL Library (LIB$) Manual』では，LIB$GET_UIB_INFO ルーチンは I64 でのみ利用できることを説明する必要があります。

I64 でのみ利用できる呼び出しコンテキストと呼び出しハンドルに関連するルーチンは，次のとおりです。

- LIB$I64_CREATE_INVO_CONTEXT
- LIB$I64_FREE_INVO_CONTEXT
- LIB$I64_GET_CURR_INVO_CONTEXT
- LIB$I64_GET_CURR_INVO_HANDLE
- LIB$I64_GET_INVO_CONTEXT
- LIB$I64_GET_INVO_HANDLE
- LIB$I64_GET_PREV_INVO_CONTEXT
- LIB$I64_GET_PREV_INVO_HANDLE

これらのルーチンについての詳細は，『HP OpenVMS Calling Standard』を参照してください。

3.5.6 『HP OpenVMS システム管理者マニュアル』: IPC コマンドの制限

『HP OpenVMS システム管理マニュアル (上巻)』の第 9.15 節「IPC (割り込み優先順位レベル C) の使用」では，I64 とすべての Alpha で IPC コマンドを使用できると説明していますが，これは誤りです。このドキュメントは訂正され，次の文章が追加されました。

OpenVMS Versions 8.2 と 8.2-1 では，グラフィック・コンソールからブートした場合には， I64 システム，または ES47 あるいは GS1280 Alpha システムで IPC コマンドを使うことはできません。

3.5.7 『HP OpenVMS System Services Reference Manual』の追加と訂正

ここでは， Version 8.2 の『HP OpenVMS System Services Reference Manual』に対する追加と訂正を説明します。

3.5.7.1 $GETRMI の項目コード —RMI$_MODES

Version 8.2 の『HP OpenVMS System Services Reference Manual』では，新しい$GETRMI の項目コード RMI$_MODES について以下の情報が抜けています。

RMI$_MODES を使用すると，マルチ CPU システムの各 CPU のモードを監視することができます (単に，ノード全体での CPU 使用率ではありません)。

RMI$_MODES は，システムのブート以降に，現在アクティブなすべての CPU のすべてのプロセッサ・モードで消費された時間を 10 ミリ秒単位の値で返します。各モードで返される時間の 1 単位は，そのモードで CPU 時間が 10 ミリ秒消費されたことを表わしています。アクティブな CPU とは OpenVMS インスタンスが実行しているプロセッサ・スケジューリングに含まれる CPU のことです。

項目記述子内のバッファ長フィールドは，システムに搭載されている CPU の数に依存します。渡すバッファのサイズは，次の式のとおりでなければなりません。

```
n * sizeof ( CPU_struct ) + 4
```

ただし，*n*は利用可能な CPU の個数です。$GETSYI の項目コード SYI$_POTENTIALCPU_CNT を使用して，利用可能な CPU の個数*n*を取得します。(詳細は，$GETSYI サービスの説明を参照してください。)

個々の CPU のデータは， CPU_struct 構造体で返されます。 CPU_struct 構造体は，次の例に示すように，どのメンバもアラインメントなしで宣言します。

```
#pragma member_alignment save
#pragma _nomember_alignment
typedef struct _CPU_struct
{
    unsigned char   cpu_id;     // physical cpu id
    unsigned int    interrupt;  // is actually the sum of interrupt and idle  1
    unsigned int    mpsynch;    // multi-processor synchronization
    unsigned int    kernel;     // kernel mode
    unsigned int    exec;       // executive mode
    unsigned int    super;      // supervisor mode
    unsigned int    user;       // user mode
    unsigned int    reserved;   // reserved, will be zero
    unsigned int    idle;       // CPU idle
}CPU_struct;
#pragma member_alignment restore
```

1 既存のアプリケーションとの互換性を確保するために， interrupt に返される時間は，割り込み時間とアイドル時間の合計になっています。実際の割り込み時間は， interrupt に返される時間から idle に返される時間を差し引いて算出してください。

マルチ CPU システムの場合には，すべてのアクティブな CPU のデータが CPU_struct 型の配列として返されます。配列内の要素の数は，利用できる CPU の数と同じです。 CPU が非アクティブの場合には， CPU_struct 配列内のその CPU に対応する要素では， CPU_struct のすべてのメンバの値は 0 になっています。

すべての利用可能な CPU のカウンタを格納できるだけの大きさを持つバッファを，確実に割り当ててください。

次に示すコード例は， CPU モードを収集する方法の 1 例です。

```
#include <starlet.h>
#include <syidef.h>
#include <rmidef.h>
#include <efndef.h>
#include <iledef.h>
#include <iosbdef.h>

CPU_struct   *cpu_counters = NULL;
IOSB          myiosb = { 0 };
char         *buffer;
unsigned long buffer_size;
int           status;

// Set up the $GETSYI item list: potential CPUs and active CPU bitmask
// unsigned long
CPU_Count = 0;
unsigned long long ActiveCPUs = 0;

ILE3 SYIitmlst[] = { {sizeof CPU_Count, SYI$_POTENTIALCPU_CNT, &CPU_Count, 0},
                     {sizeof ActiveCPUs, SYI$_ACTIVE_CPU_MASK, &ActiveCPUs, 0},
                     {0,0}};
```

```
// Get the available CPU count and a bitmask of active CPUs
status = SYS$GETSYIW( EFN$C_ENF, NULL, NULL, SYIitmlst, &myiosb, NULL, 0 );
buffer_size = CPU_Count * sizeof( CPU_struct ) + 4;
buffer = malloc( buffer_size );

// Set up the $GETRMI item list: CPU modes
ILE3 RMIitmlst[] = {{buffer_size, RMI$_MODES, buffer, 0},
                    {0,0}};

// Call the service
status = SYS$GETRMI( EFN$C_ENF, 0, 0, RMIitmlst, &myiosb, NULL, 0 );

// THE DATA COUNTERS BEGIN 4 BYTES OFF THE START OF THE BUFFER;
// The first 4 bytes of the buffer are reserved for internal use.
cpu_counters = (CPU_struct *) ( buffer + 4 );

// Use the counters for all active CPUs
for ( int i = 0; i < CPU_Count; i++)
{
    if ( 1 == ( 1 & (ActiveCPUs >> i) ) )
    {
       cpu_counters[i].interrupt
       ---
       ---
       ---
    }
}

free( buffer );
```

3.5.7.2 $PUTMSG システム・サービスの訂正

Version 8.2 の『HP OpenVMS System Services Reference Manual』では，$PUTMSG のプロトタイプの説明に誤りがあります。正しいプロトタイプは次のとおりです。

C プロトタイプ

```
int sys$putmsg (void *msgvec, int (*actrtn)(__unknown_params),
void *facnam, unsigned __int64 actprm);
```

*actrtnからの戻り値は実際にチェックされ，メッセージが入力されたかどうかの判定に使用されることに注意してください。

ドキュメントのソース・ファイルが訂正され，その訂正の内容は次版の『HP OpenVMS System Services Reference Manual』とオンライン・ヘルプでリリースされる予定です。

3.5.8 『HP Volume Shadowing for OpenVMS 説明書』：メモリ要件の訂正

Version 7.3-2 の『HP Volume Shadowing for OpenVMS 説明書』の第 1 章の第 1.3.1 項「メモリ要件」で説明されているように， OpenVMS Version 7.3 からは， Volume Shadowing for OpenVMS を実行するには追加メモリが必要になりました。以下の訂正があることに注意してください。

- 「たとえば，メンバごとに 200 GB のストレージがあるシャドウ・セットでは，クラスタ内の各々のノードの書き込みビットマップには， 420 KB のメモリが必要になります。」という説明がありますが， 420 KB ではなく 400 KB が正しい値です。
- ここから 3 段落先に，「たとえば， 10 個のシャドウ・セットがマウントされているシステムで，各々のシャドウ・セットに 50 GB のメンバ・ディスクがある場合，追加で 1,119 KB のメモリが必要ですが，」という説明がありますが，その下に示されている計算方法に従って計算されるように， 1,119 KB ではなく 1,069 KB が正しい値です。

これらは，『HP Volume Shadowing for OpenVMS 説明書』の次版で訂正される予定です。

3.6 EFI シェルで共用システム・ディスクまたはシャドウ・システム・ディスクを操作する場合の注意事項

Integrity サーバのシステム・ディスクには， OpenVMS ブート・ローダ， EFI アプリケーション，ハードウェア診断ツールを含む FAT パーティションが最大 2 つ存在します。 OpenVMS のブートストラップ・パーティションと診断パーティション (存在する場合) は，それぞれ OpenVMS システム・ディスク内の次のコンテナ・ファイルにマップされます。

```
SYS$LOADABLE_IMAGES:SYS$EFI.SYS
SYS$MAINTENANCE:SYS$DIAGNOSTICS.SYS
```

これらの FAT パーティションの内容は，コンソールの EFI Shell>プロンプトに，fs*n*: デバイスとして現れます。これらの fs*n*: デバイスは， EFI Shell>プロンプトでのユーザ・コマンド入力，または EFI コンソール・アプリケーションや EFI 診断アプリケーションによって直接変更することができます。システム・ディスクを共用する OpenVMS 環境または EFI コンソール環境のいずれにも，パーティションの変更は通知されません。 OpenVMS 環境と EFI コンソール環境は，これらのコンソールによる変更を全く意識しません。そのため， OpenVMS コンソールや使用する他の任意の EFI コンソールでは，この変更に適切に連携して同期を取る必要があります。

次のどちらかまたは両方の手段で，構成内のコンソールを変更するときには注意が必要です。

- OpenVMS I64 システム・ディスクに対する OpenVMS のホスト・ベースのボリューム・シャドウイング
- システム・ディスクを共用している Integrity 環境間での，共用システム・ディスクと EFI コンソールへの同時アクセス

このような OpenVMS システム・ディスク環境は，前もって，単一メンバのホスト・ベース・ボリューム・シャドウセット，または非シャドウ・システム・ディスクに移行し，さらに，次のような操作で fs*n*: デバイスをシェル・レベルで変更するときには， Shell>プロンプトへの同時アクセスを行わないようにアクセスを調整する必要があります。

- 診断パーティション内での，診断ツールのインストールまたは操作
- パーティション内またはリムーバブル・メディアから実行する診断ツールに，OpenVMS I64 システム・ディスク上のブート・パーティションまたは診断パーティションの変更を許す
- あるいは，これらの環境の Shell>プロンプトから，ブート・パーティションまたは診断パーティションを直接的または間接的に変更

上記の予防措置をとらなかった場合には，ブート・パーティションに対応する fs*n*デバイスでの変更や，診断パーティションに対応するデバイスでの変更は直ちに，または次回の OpenVMS のホスト・ベース・ボリューム・シャドウイングのフル・マージ操作の後に，書き換えられて失われます。

たとえば，シャドウ・システム・ディスクのいずれかの物理メンバでコンテナ・ファイルの内容が EFI コンソールのシェルによって変更されても，ボリューム・シャドウイング・ソフトウェアは物理デバイスへの書き込みがあったことは認識できません。システム・ディスクが複数のメンバからなるシャドウ・セットの場合には，シャドウ・セットのメンバである他の物理デバイスのすべてに対して同じ変更を行う必要があります。そうしないと，システム・ディスクでフル・マージ操作が行われたときに，これらのファイルの内容が元に戻ってしまいます。マージ操作は， EFI での変更が行われてから数日後，または数週間後に行われることもあります。

また，シャドウ・システム・ディスクでフル・マージがアクティブになっている場合には，いずれのファイルもコンソールの EFI シェルを使って変更してはなりません。

進行中のフル・マージ操作を停止する方法や，シャドウ・セットのメンバ構成を調べる方法については，『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。

これらの注意事項は，ホスト・ベースのボリューム・シャドウイング用に構成されている Integrity システム・ディスク，または複数の OpenVMS I64 システムで構成されて共用されているシステム・ディスクにのみ適用されます。コントローラ・ベースの RAID を使用している構成，システム・ディスクでホスト・ベースのシャドウイングを使用していない構成，別の OpenVMS I64 システムと共用していない構成では影響を受けません。

3.7 Librarian: オブジェクト・モジュール名のキー長の拡大

OpenVMS Version 8.2 では， Librarian は I64 (ELF) オブジェクト・イメージ・ライブラリと共用イメージ・ライブラリのオブジェクト・モジュール名のキーの長さを，誤って， 31 文字に制限していました。この問題は修正されました。 Librarian は最大で 1024 文字のオブジェクト・モジュール名のキー長を受け付けるようになりました。

3.8 Linker ユーティリティ

『HP OpenVMS V8.2 リリース・ノート[翻訳版]』内の大部分の注意事項は，OpenVMS I64 Version 8.2-1 の Linker ユーティリティにも当てはまりますが，OpenVMS Version 8.2-1 では， HP OpenVMS Version 8.2 のリリース・ノートに対して以下の変更点があります。

- 第 5.24 節のすべての注意事項は， Alpha だけでなく，すべての OpenVMS プラットフォームに当てはまるようになりました。
- 第 5.24.4 項「スタックのエレメント数は最大 25 に制限」は， Alpha と VAX にのみ当てはまります。
- 第 5.25.4 項「マップの Image Synopsis セクションの誤り」は，誤りが修正されたため， V8.2-1 には適用されません。
- 第 5.25.5 項「DIFTYPE および RELODIFTYPE メッセージ」の TYPE1.C と TYPE2.C の例とこの例の前文は，次のように読み替えてください。

  次の 2 つのモジュールからなる例は，これらの問題点を修正する方法を示しています。

```
TYPE1.C

 #include <stdio>

 int status ;   // Defines status as data.
 extern int sub();

 main ()
 {
     printf ("Hello World\n");
     sub();
 }

TYPE2.C

 extern int status (int x) ; //Refers to status as a procedure.

 sub ()
 {
 int x;
         x = (int)status;
         return status (x);
 }
```

Linker の新機能についての章 (第 5 章) には，理解しておく必要がある以下の情報が記載されています。

- 第 5.2.2 項，「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」

  問題点: 0 で初期化されたオーバレイ・プログラム・セクションは，誤って，互換性のある初期化と見なされます。 OpenVMS V8.2 と V8.2-1 の I64 Linker では 0 で初期化されたセクションは 0 以外の初期値を持つセクションと互換性があると見なされます。そのため，リンク操作時のモジュールの順番によっては，イメージに異なる初期値が設定されます。

  解決策:この問題は I64 Linker の将来のリリースで修正される予定です。

- 第 5.4.4 項，「Linker の新しい修飾子: /EXPORT_SYMBOL_VECTOR と /PUBLISH_GLOBAL_SYMBOLS」

  問題点:すべてのグローバル・シンボルをエクスポートするだけで共用可能イメージを作成しても，それだけではうまくいきません。/EXPORT_SYMBOL_VECTOR 修飾子と/PUBLISH_GLOBAL_SYMBOLS 修飾子を使用すれば，すべてのグローバル・シンボルのシンボル・ベクタ・オプションをモジュール単位に作成できます。ただし，これらの修飾子は，従来の方法でシンボル・ベクタを作成する場合には，別の問題を起こします。

  解決策:エラーが発生する可能性があるため，両方の修飾子は I64 Linker の将来のリリースでは廃止される予定です。

3.9 セル型システムでの複数の nPartitions

HP Integrity rx7620， HP Integrity rx8620， HP Integrity Superdome などのサーバに複数の nPartitions を構成し，マルチ・オペレーティング・システム環境で動作させ，そのうち 1 つの nPartitions で OpenVMS を動作させる場合， OpenVMS のブート時に他のオペレーティング・システムのいずれかが SEL (システム・イベント・ログ) にエラーまたはイベントを記録することがあります。 OpenVMS は FRU (Field Replaceable Unit) テーブルの生成が完了するまで SEL を保持し続けるため，それが原因で，他のオペレーティング・システムがエラーまたはイベントを記録することがあります。

3.10 Version 8.2 から Version 8.2-1 へのネットワーク・アップデートの制限

OpenVMS Version 8.2-1 は， Version 8.2 から Version 8.2-1 へのネットワーク・アップデートをサポートしています。ネットワーク・アップデートは OpenVMS Version 8.2 でサポートされるシステム上のコア I/O LAN カードを使用する形態だけがサポートされています。詳細は，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。

また，ネットワーク・ブートにはハードウェア構成上の制限もあります。ネットワーク・アダプタの速度や二重モードの設定をコンソールから実行できる Alpha コンソールとは異なり， Integrity サーバのコンソールとネットワーク・ブート・ドライバは自動ネゴシエーションしか実行できません。正常にネットワーク・ブートを実行するためには， Integrity サーバのブート・クライアントの最も近くにあるネットワーク・スイッチに自動ネゴシエーションを設定する必要があります。スイッチを自動ネゴシエーションに設定しないと，ネットワーク・ブート・プロセスを実行することができません。

3.11 NPAG_AGGRESSIVE と NPAG_GENTLE のデフォルト値

OpenVMS Version 8.2 以降は，システム・パラメータ NPAG_AGGRESSIVE と NPAG_GENTLE のデフォルト値には， 100 が設定されるようになりました。 100 を設定すると，非ページング・プールのルックアサイド・リストの控えめな再生と積極的な再生の両方が無効になります。多くの場合，プールを再生するために小さなパケットをルックアサイド・リストからバリアブル・リストに戻すと，バリアブル・リストで断片化が生じます。断片化の結果，バリアブル・リストの先頭部分には多くの小さなパケットが入り，バリアブル・リストの末尾部分には少数の大きなパケットが入るようになります。

どのルックアサイド・リストよりも大きなパケットの割り当てを行うときには，システムはバリアブル・リスト上で十分な大きさを持つパケットを検索する必要があります。過度に断片化されている場合には，上述のように，十分な大きさを持つパケットを見つけるために，バリアブル・リスト全体の検索が必要になります。バリアブル・リストはアドレスの順番で並んでいるため，大きなパケットの割り当てが解除されたときにも，リスト全体を検索して，パケットの割り当てを解除する必要があります。

このような状況では，システムの性能は大きく低下する可能性があります。そのため，プールの再生をオフにして，システム・パラメータの NPAG_GENTLE と NPAG_AGGRESSIVE をともに 100 のままにしておくことをお勧めします。

3.12 同期型データ・リンクはサポートされていない

OpenVMS Version 8.2-1 では， Integrity サーバの同期型データ・リンク・ハードウェアはサポートしていません。

3.13 HP TCP/IP Services for OpenVMS

OpenVMS Version 8.2-1 へアップグレードしたときには，最新のソフトウェアが確実にインストールされるように TCP/IP Services for OpenVMS Version 5.5 に必ず最新の ECO (エンジニアリング・チェンジ・オーダ) を適用してください。

3.14 Traceback API の問題の修正

OpenVMS I64 Version 8.2 では，引数pc_rel (相対 PC 値) またはimage_desc (イメージ名文字列記述子) に 0 を指定すると，エラーが発生していました。 Traceback 機能ではこのような引数を正しく無視するようになったため， Traceback 処理が続行可能になりました。

4

InfoServer ユーティリティ

この章では， InfoServer ユーティリティの新機能について説明します。この章には，InfoServer ハードウェアと InfoServer アプリケーションの比較と InfoServer ユーティリティ・コマンドのリファレンスが含まれています。

4.1 InfoServer ユーティリティの概要

InfoServer アプリケーションを使用すると， LAN 上の仮想ディスク・デバイス用のサービスを作成することができます。

仮想ディスク・デバイスには，以下のものがあります。

- DVD ドライブ
- 特定のディスク・ドライブ: SCSI および Fibre Channel
- CD ドライブ
- パーティション (コンテナ・ファイルとも言う)

InfoServer ハードウェアと InfoServer アプリケーションの比較

OpenVMS の新しい InfoServer アプリケーションは，以前の InfoServer ハードウェアとは，いくつかの点で大きく異なっています。最も大きな違いは，以下のとおりです。

- DCL スタイルのコマンド構文の使用
- デバイス用のサービスを作成する前に，そのデバイスをマウントしておかなくてはならない点
- DVD ドライブ用サービスの作成のサポート
- テープ・デバイスはサポートされない
- CD-R (CD-recordable) ドライブはサポートされない
- 自動マウントはサポートされない

4.1.1 InfoServer の使用方法の概要

InfoServer ユーティリティ・コマンドを使用すると，以下の作業が実行できます。

- LAN 上の仮想ディスク・デバイス用のサービスの作成と削除
- アクティブな InfoServer サービスのリストの保存

- 既存のサービスの属性の変更
- サービスに接続されているサーバとノードについての情報の表示
- 1 つまたは複数のサービスについて，サービス固有の情報の表示
- LASTport/Disk サーバの起動とサーバおよびキャッシュの各種の特性の設定

InfoServer は以下の方法で起動できます。

- **RUN コマンドの使用**

  RUN コマンドを使用して InfoServer を起動するには， DCL コマンド・プロンプトで次のように入力します。

```
$ RUN SYS$SYSTEM:ESS$INFOSERVER
```

  InfoServer ユーティリティからプロンプトが表示されたら，次のように InfoServer のコマンドを入力します。

```
InfoServer> SHOW SERVER
```

  InfoServer がコマンドの実行を完了すると，再び InfoServer>プロンプトが表示されます。ユーティリティを終了するまで，この動作が繰り返されます。

- **InfoServer をフォーリン・コマンドとして定義**

  DCL プロンプト，または起動コマンド・ファイルまたはログイン・コマンド・ファイルで，次のように入力することで， InfoServer をフォーリン・コマンドとして定義することができます。

```
$ InfoServer :== $ESS$INFOSERVER
```

  ログイン・コマンド・ファイルを実行した後は， DCL プロンプトで INFOSERVER と入力するだけで，このユーティリティを起動できます。

```
$ INFOSERVER
```

  以下の点に注意してください。

  – InfoServer をフォーリン・コマンドとして使用し，また InfoServer のコマンドを入力すると，次のように，このユーティリティはコマンドを実行した後に終了し， DCL コマンド・プロンプトの状態に戻ります。

```
$ InfoServer SHOW SERVER
$
```

  – InfoServer のコマンドを指定せずに InfoServer をフォーリン・コマンドとして使用すると，このユーティリティは InfoServer>プロンプトを表示するので，その段階で，次のように，コマンドが入力できます。

```
$ InfoServer
InfoServer>  SHOW SERVER
```

注意

すべての InfoServer **コマンドは，** SYSPRV **および** OPER **の権限を必要とします。**

InfoServer ユーティリティを終了するには， InfoServer>プロンプトで EXIT コマンドを入力するか Ctrl/Z を押します。

InfoServer ユーティリティについての情報を表示するには， InfoServer>プロンプトで HELP コマンドを入力します。

4.1.2 InfoServer コマンド

以降の項で， InfoServer コマンドの説明と例を示します。

CREATE SERVICE

指定したデバイスまたはパーティション用のサービスを作成します。

使用方法:

- すべてのデバイスは，システムワイドにマウントし，プロセスがログアウトしたときにマウント解除されないようにする必要があります。
- 読み書きサービスを持つデバイスは，/FOREIGN 修飾子を付けてマウントし，OpenVMS から見えないようにする必要があります。
- 読み取り専用サービスを持つデバイスは，/NOWRITE 修飾子または/FOREIGN 修飾子を付けてマウントし，誰もローカルに変更できないようにする必要があります。
- 読み取り専用または読み書きアクセスで OpenVMS にマウントされるディスクでは，パーティションのサービスはオフにできます。
- 今回のリリースでは，パーティションのサポートは制限されています。

フォーマット

CREATE SERVICE　*serviceName device-or-partitionName*

パラメータ

serviceName
LAN 上で使用されるサービスの名前です。サービス名は英数字とドル記号 ($) で構成されます。長さは最大で 255 文字です。

device-or-partitionName
LAN 上で使用される OpenVMS のディスク・デバイスまたはパーティションの名前です。入力するデバイス名またはパーティション名は，作成済みのものでなければなりません。

デバイス名とパーティション名について，以下に説明します。

- デバイス名

  LAN にサービスを提供するデバイスは， OpenVMS ディスク・デバイスです。InfoServer デバイス名を指定するときには， OpenVMS デバイス名を使用します。デバイス名は， SHOW SERVICES コマンドで表示される名前に完全に一致するか，またはワイルドカードを含むかのいずれかでなければならないことに注意してください。

InfoServer ユーティリティでワイルドカードが使用できるところでは，OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

ディスク指定はコロンで終わる必要があります。

- パーティション名

  パーティションは，ネットワークにサービスが提供されるコンテナ・ファイルです。したがって，デフォルトのファイル・タイプが.ESS$PARTITION の OpenVMS ファイル名を持っています。デバイス名，ディレクトリ名，ファイル名を含むパーティション名の長さは，最大で 242 文字です。

  今回のバージョンでは，パーティションのサポートは制限されています。ハード・ドライブの分割利用には， LD デバイスを使用することをお勧めします。詳細は， DCL コマンドの LD HELP を参照してください。

修飾子

/CLASS=className
完全な LASTport Disk (LAD) 名前空間のサブセットを指定します。

クラス名の目的は，名前空間を分割して，クライアントに対して意味のある名前だけを示すことです。クラス名を使用すると， 2 つのサービスに同じ名前を付けて，互いに競合しないようにすることもできます。

たとえば，いくつかのクライアント・システムで使用される異なるオンディスク構造に対して，異なるクラス名をつけることができます。あるクライアント・システムでは SERVICEA/CLASS=ODS-2 を使用し，別のクライアント・システムでは SERVICEA/CLASS=ISO_9660 を使用します。サービス名は同じ (SERVICEA) ですが，クラス名は異なります。

使用するクラス名は，作成するサービスに接続するクライアント・システムに依存します。デフォルトのクラス名は ODS_2 です。たとえば， OpenVMS システムは， InfoServer デバイスをマウントするとき， ODS_2 名前空間を使用します。OpenVMS クライアントは， ODS_2 サービス・クラスに属しているサービスのみ利用できることに注意してください。

有効なクラス名は，次のとおりです。

V2.0	PCSA MS-DOS クライアントが理解する名前
Unformatted	仮想ディスクは形式を持っていません
MSDOS	MSDOS 仮想ディスク
ODS_2	VMS 仮想ディスク
UNIX	UNIX 仮想ディスク
ISO_9660	ISO 9660 CD 形式
HIGH_SIERRA	MS-DOS CD 形式
APPLE	Macintosh HFS 形式
SUN	Sun 形式

/ENCODED_PASSWORD=hexstring
SAVE コマンドでこの修飾子が作成されます。パスワードは平文で保存されず，ハッシュ化されたパスワードの値が SAVE 操作の一環として書き込まれるため，パスワードを人目にさらすことなく，サービスを作成しなおすことができます。

SAVE コマンドが作成するコマンド・プロシージャを編集してサービス名を変更した場合には，エンコードされたパスワード値が有効ではなくなってしまうことに注意してください。その場合には，/PASSWORD 修飾子を使用して，サービスに別のパスワードを設定する必要があります。

/PASSWORD=passwordString
/NOPASSWORD (**デフォルト**)
オプションのサービス・アクセス制御パスワードを指定します。パスワードが設定されたサービスにクライアント・システムがアクセスするためには，パスワードを指定する必要があります。

パスワードは最大 39 文字の英数字文字列です。パスワードを指定しなかった場合には，クライアント・システムではこのサービスにアクセスする場合に，パスワードの入力を求められません。

テキストのパスワードは暗号化された形式で，他のサービス情報とともにメモリに保存されます。

/RATING=DYNAMIC
/RATING=STATIC=value
該当するサービスが複数ある場合には，クライアントはサービス評価点を使用してサービスを選択します。最高のサービス評価点を持ったサービスが選択されます。

システムは動的サービス評価点を負荷に応じて調整します。静的評価点を 0 ～ 65535 の範囲で指定することもできます。システムは静的評価点は調整しません。

静的評価点を使用する場合の例は，サービスのコピーの 1 つから，別のコピーにクライアントを移行させる場合です。クライアントを移動させたいサービスに静的評価点 0 を設定しておけば，新しいクライアントが評価点 0 のサービスに接続することはありません。代わりに，それよりも高い評価点を持つサービスに接続します。すべての現在のクライアントがサービスとの接続を解除したときに，そのサービスを安全に削除することができます。

/READAHEAD (**デフォルト**)
/NOREADAHEAD
キャッシュ・ブロックを埋めるためのディスクの読み取りが必要になったときに，/READAHEAD 修飾子は，読み取り範囲が，要求された最初のブロックから，バケット境界の最後までであることを指示します。先読みすることによって，必要なディスク・ブロックが前もってキャッシュにロードされるので，シーケンシャル操作の速度が向上します。

/READAHEAD 修飾子と/READBEHIND 修飾子の両方を指定しておくと，キャッシュ・バケット内のブロックが要求されたときに，そのブロックのバケット範囲全体がキャッシュに読み込まれます。

/READBEHIND
/NOREADBEHIND (**デフォルト**)
キャッシュ・ブロックを埋めるためのディスクの読み取りが必要になったときに，/READBEHIND 修飾子は，読み取り範囲が，キャッシュ・バケット境界の先頭から要求されたブロックまでであることを指示します。

/READAHEAD 修飾子と/READBEHIND 修飾子の両方を指定すると，キャッシュ・バケット内のブロックが要求されたときに，そのブロックのバケット範囲全体がキャッシュに読み込まれます。

/READERS=number (**デフォルトは，**READERS=1000)
/NOREADERS
クライアントの読み取りアクセスで許される同時接続の最大数を指定します。デフォルトは 1000 接続です。値 0 は書き込み専用であることを意味します。

クライアントがサービスに対して読み取り専用アクセスまたは読み書きアクセスを要求した場合には，システムは読み取り接続としてカウントします。

/WRITERS
/NOWRITERS (**デフォルト**)
サービスが 1 つの書き込み接続だけを許すことを指定します。

例

1. `$ SHOW DEVICE MOVMAN$DQA0:/full`

```
Disk MOVMAN$DQA0:, device type Compaq  CRD-8322B, is online, file-oriented
    device, shareable, served to cluster via MSCP Server, error logging is
    enabled.
```

```
Error count                0    Operations completed
Owner process             ""    Owner UIC                  [SYSTEM]
Owner process ID     00000000    Dev Prot        S:RWPL,O:RWPL,G:R,W
Reference count             0    Default buffer size            512
Total blocks         16515072    Sectors per track               63
Total cylinders         16384    Tracks per cylinder             16

$ MOUNT/SYSTEM dqa0 OVMSIPS11

Volume is write locked
OVMSIPS11 mounted on _MOVMAN$DQA0:

$ InfoServer
InfoServer> CREATE SERVICE VMS_SIPS_V11 _MOVMAN$DQA0:

%INFOSRVR-I-CRESERV, service VMS_SIPS_V11 [ODS-2] created for
_MOVMAN$DQA0:.
```

この例は， CD デバイス用のサービスの作成方法を示しています。

- SHOW DEVICE . . . /FULL コマンドは，_MOVMAN$DQA0 CD について完全な情報リストを表示します。
- MOUNT/SYSTEM コマンドは， OVMSIPS11 ボリュームを_MOVMAN$DQA0: CD にマウントします。
- InfoServer の CREATE SERVICE コマンドは，_MOVMAN$DQA0 CD 上に VMS_SIPS_V11 サービスを作成します。

2.
```
$LD CREATE KIT1/SIZE-100000
$DIRECTORY KIT1




Directory DKB0:[DISKS]

KIT1.DSK;1       100000/100008   29-APR-2005 14:14:43.49

Total of 1 file, 100000/100008 blocks.

 $ LD CONNECT KIT1

%LD-I-UNIT, Allocated device is MOVMAN$LDA1:

 $ CREATE SERVICE TEST_KIT_1 MOVMAN$LDA1:

%INFOSRVR-I-CRESERV, service TEST_KIT_1 [ODS-2] created for
 _MOVMAN$LDA1:
```

この例は，論理ディスク (LD) デバイス用のサービスの作成方法を示しています。

- LD CREATE KIT1 コマンドは，論理ディスクとして使用できる連続ファイル KIT1 を作成します。

- DIRECTORY KIT1 コマンドは KIT1 に関する情報を表示します。
- LD CONNECT KIT1 は，論理ディスク・ファイル KIT1 を論理ディスク・デバイス MOVMAN$LDA1: に接続します。
- INITIALIZE コマンドは， MOVMAN$LDA1: LD デバイスをフォーマットします。
- MOUNT コマンドは， LD デバイスを利用できるようにします。
- CREATE SERVICE コマンドは，_MOVMAN$LDA1 LD デバイス上に TEST_KIT_1 サービスを作成します。

DELETE SERVICE

1 つまたは複数のサービスを削除します。

フォーマット

DELETE SERVICE *serviceName [device-or-partitionName]*

パラメータ

serviceName
LAN 上で使用されるサービスの名前です。サービス名は英数字とドル記号 ($) で構成されます。長さは最大で 255 文字です。ワイルドカードも許されています。

InfoServer ユーティリティでワイルドカードが使用できるところでは， OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

device-or-partitionName
デバイス名またはパーティション名は， LAN 上で使用される OpenVMS ディスク・デバイスまたはパーティションの名前です。入力するデバイス名またはパーティション名は，作成済みのものでなければなりません。

デバイス名とパーティション名について，以下に説明します。

- デバイス名

  LAN にサービスを提供するデバイスは， OpenVMS ディスク・デバイスです。InfoServer デバイス名を指定するときには， OpenVMS デバイス名を使用します。デバイス名は， SHOW SERVICES コマンドで表示される名前に完全に一致するか，またはワイルドカードを含むかのいずれかでなければならないことに注意してください。

  InfoServer ユーティリティでワイルドカードが使用できるところでは，OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

  ディスク指定はコロンで終わる必要があります。

- パーティション名

  パーティションは，ネットワークにサービスが提供されるコンテナ・ファイルです。したがって，デフォルトのファイル・タイプが.ESS$PARTITION の

OpenVMS ファイル名を持っています。デバイス名，ディレクトリ名，ファイル名を含むパーティション名の長さは，最大で 242 文字です。

パーティション名は選択するサービスを特定する場合に使用できます。

今回のバージョンでは，パーティションのサポートは制限されています。パーティションに分割されたハード・ドライブをサポートする場合には， LD デバイスを使用することをお勧めします。詳細は， DCL コマンドの LD HELP を参照してください。

修飾子

/CLASS=className
完全な LASTport Disk (LAD) 名前空間のサブセットを指定します。

クラス名の目的は，名前空間を分割して，クライアントに対して意味のある名前だけを示すことです。クラス名を使用すると， 2 つのサービスに同じ名前を付けて，互いに競合しないようにすることもできます。

たとえば，いくつかのクライアント・システムで使用される異なるオンディスク構造に対して，異なるクラス名をつけることができます。あるクライアント・システムでは SERVICEA/CLASS=ODS-2 を使用し，別のクライアント・システムでは SERVICEA/CLASS=ISO_9660 を使用します。サービス名は同じ (SERVICEA) ですが，クラス名は異なります。

使用するクラス名は，作成するサービスに接続するクライアント・システムに依存します。デフォルトのクラス名は ODS_2 です。たとえば， OpenVMS システムは， InfoServer デバイスをマウントするとき， ODS_2 名前空間を使用します。OpenVMS クライアントは， ODS_2 サービス・クラスに属しているサービスのみ利用できることに注意してください。

有効なクラス名は，次のとおりです。

V2.0	PCSA MS-DOS クライアントが理解する名前
Unformatted	仮想ディスクは形式を持っていません
MSDOS	MSDOS 仮想ディスク
ODS_2	VMS 仮想ディスク
UNIX	UNIX 仮想ディスク
ISO_9660	ISO 9660 CD 形式
HIGH_SIERRA	MS-DOS CD 形式
APPLE	Macintosh HFS 形式
SUN	Sun 形式

/CONFIRM (**デフォルト**)
/NOCONFIRM
サービスを削除するときに確認が求められます。/NOCONFIRM を指定した場合でも，接続が残っている場合には，確認が求められます。

各サービスの削除操作の前に確認を行うかどうかを制御します。以下の応答が有効です。

```
YES      NO       QUIT
TRUE     FALSE    Ctrl/Z
1        0        ALL
    Return キー
```

使用上の注意:

- 単語で答える場合には，大文字と小文字を混在させて構いません。単語による応答は， 1 文字以上の英字に省略できます (たとえば， TRUE に対して， T， TR，TRU)。ただし，省略する場合には，一意である必要があります。
- 肯定的な応答は， YES， TRUE，および 1 です。否定的な応答は， NO，FALSE， 0，および Return キーを押すことです。
- QUIT の入力または Ctrl/Z の押下は，その時点でコマンドの処理を停止させたいことを意味します。
- ALL を入力して応答した場合には，コマンドは処理を続行して，その後，プロンプトは表示されなくなります。

/DISCONNECT
/NODISCONNECT (**デフォルト**)
接続されているセッションが残っているサービスを削除しようとしたときのデフォルトの確認プロンプト処理を変更します。サービスに接続しているセッションがあって/DISCONNECT 修飾子を指定していない場合には，サービスの削除に確認が求められます。

サービスを削除する際に確認を求められないようにするには，/NOCONFIRM 修飾子と/DISCONNECT 修飾子の両方を指定します。

例

```
$ SHOW SERVICES




    Service Name         [Service Class] Device or File
    -------------------- --------------- -------------------------
    HUDSON               [ODS-2]         _MOVERS$LDA1: [ 1 Connection]
    BAFFIN               [ODS-2]         _MOVERS$LDA1:
    FUNDY                [ODS-2]         _MOVERS$LDA1:
    3 services found.

$ DELETE SERVICE HUDSON
```

```
Service HUDSON has 1 session connected!
Delete service HUDSON [ODS-2] for _MOVERS$LDA1:? [N]:
```

最初のコマンドでは，1つの接続中のセッションを含む，3つのサービスを表示しています。2番目のコマンドではHUDSONサービスを削除しています。このコマンドは，HUDSONに1つのセッションが接続されていることを示すメッセージを表示し，そのサービスを削除してよいか確認するプロンプトを表示しています。

EXIT

InfoServer の実行を終了します。 Ctrl/Z を押しても，終了することができます。

フォーマット

EXIT

HELP

InfoServer のオンライン・ヘルプを表示します。

ESS$INFOSERVER は， OpenVMS のアプリケーションとして実装された LASTport/Disk サーバのユーザ・インタフェースです。動作はハードウェア InfoServer 製品と似ていますが，完全に同じではありません。

フォーマット

HELP *[topic]*

パラメータ

topic
ヘルプを要求するトピックを指定します。

例

```
$ INFOSERVER HELP SHOW SESSIONS
```

このコマンドは， InfoServer の SHOW SESSIONS コマンドについてのヘルプを表示します。

SAVE

現在アクティブなサービスのセットを，コマンド・プロシージャ内の一連のコマンドとして保存します。システムをリブートしたときにこのコマンド・プロシージャを実行すれば，現在のサービスのセットを再現することができます。

フォーマット

SAVE *procedureName*

パラメータ

procedureName
現在のサーバの状態を復活させるためのコマンド・プロシージャを作成します。プロシージャ名は，作成するコマンド・プロシージャの OpenVMS ファイル名です。ファイル・タイプを指定しない場合には，デフォルトで.COM になります。

デフォルトのプロシージャ名は ESS$LAD_SERVICES.COM です。

例

```
$ SHOW SERVICES


    Service Name         [Service Class] Device or File
    -------------------- --------------- --------------
    BASELEVEL_A          [ODS-2]         _INFOS$LDA1:
    BASELEVEL_B          [ODS-2]         _INFOS$LDA2:
    BASELEVEL_C          [ODS-2]         _INFOS$LDA3:
    BASELEVEL_D          [ODS-2]         _INFOS$LDA4:

    FIELD_TEST_BASELEVEL [ODS-2]         _INFOS$LDA2:
    CURRENT_BASELEVEL    [ODS-2]         _INFOS$LDA3:
    EXPERIMENTAL_BASELEVEL
                         [ODS-2]         _INFOS$LDA4:
    %INFOSRVR-I-FOUND, 7 services found.

$ SAVE BASELEVELS
```

```
$! Created by the OpenVMS InfoServer SAVE command on 22-APR-2005
14:34:02.48
$ Set NoOn
$ Infoserver := $ESS$INFOSERVER
$!
$! The comment for each service includes the current device name.
$!
$!****************************************************************
$!  BASELEVEL_A [ODS_2] - _BILBO$LDA1: 1
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_A.DSK;1 2
$ LD_UNIT_1 := LDA'LD_UNIT': 3
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_1' BASELEVELA 4
 $ INFOSERVER Create Service BASELEVEL_A 'LD_UNIT_1' - 5
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  BASELEVEL_B [ODS_2] - _BILBO$LDA2:
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_B.DSK;1
$ LD_UNIT_2 := LDA'LD_UNIT':
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_2' BASELEVELB
$ INFOSERVER Create Service BASELEVEL_B 'LD_UNIT_2' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  BASELEVEL_C [ODS_2] - _BILBO$LDA3:
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_C.DSK;1
$ LD_UNIT_3 := LDA'LD_UNIT':
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_3' BASELEVELC
$ INFOSERVER Create Service BASELEVEL_C 'LD_UNIT_3' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  BASELEVEL_D [ODS_2] - _BILBO$LDA4:
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_D.DSK;1
$ LD_UNIT_4 := LDA'LD_UNIT':
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_4' BASELEVELD
$ INFOSERVER Create Service BASELEVEL_D 'LD_UNIT_4' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic -
        /Encoded_Password=481C6B9081E742C2
           ! Invalid if service name changes 6
$!****************************************************************
$!  FIELD_TEST_BASELEVEL [ODS_2] - _BILBO$LDA2:
$!****************************************************************
$ INFOSERVER Create Service FIELD_TEST_BASELEVEL 'LD_UNIT_2' - 7
        /Class=ODS_2/Readers=1000/NoWriters -
```

```
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!*************************************************************
$ INFOSERVER Create Service CURRENT_BASELEVEL 'LD_UNIT_3' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!*************************************************************
$!  EXPERIMENTAL_BASELEVEL [ODS_2] - _BILBO$LDA4:
$!*************************************************************
$ INFOSERVER Create Service EXPERIMENTAL_BASELEVEL 'LD_UNIT_4' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic -
        /Encoded_Password=01F1D7374C0B81EC
          ! Invalid if service name changes 8
$ Exit
```

この例の SHOW SERVICES コマンドは，現在このサーバが提供しているサービスを表示します。ここには一連のソフトウェア・ベースレベルがあり，それぞれの論理ディスクと LAN に提供しているサービスが示されています。ベースレベルには a ～ d のラベルが付けられていますが，対応する英字をユーザが覚えなくてもいいように名前も付けられています。

デバイス LDA2， LDA3， LDA4 にはそれぞれ 2 つのサービスが割り当てられていることに注意してください。

例の中の数字は，以下の説明の番号に対応しています。

1 各デバイスのコメントには， SAVE コマンドを実行したときのデバイス名が含まれています。 LD デバイスは擬似ディスク・デバイスであり，ユニット番号は接続するたびに変わります。

2 このコマンドは， LD デバイスをコンテナ・ファイルに接続して，ユニット番号を DCL のシンボル LD_UNIT に代入します。

3 コンテナ・ファイルに割り当てられる各デバイスに対して，一意のシンボルが作成されます。

4 このコマンドは， SAVE コマンドの実行時にデバイスに付いていたボリューム・ラベルを指定して，デバイスをマウントします。

5 当該デバイスに対する InfoServer サービスが再作成されます。

6 EXPERIMENTAL_BASELEVEL サービスはパスワードで保護されています。セキュリティを確保するために，パスワードは暗号化された形式でコマンド・プロシージャ内に格納されています。 2 つのサービスは同じパスワードを持っていますが，暗号化されたパスワードは異なっていることに注意してください。

7 FIELD_TEST_BASELEVEL と BASELEVEL_B は同じ LD デバイスを指しているため，別のデバイスは作成されず，作成済みのユニットを参照するために，正しいユニット (シンボルは LD_UNIT_2) が使用されます。

8 6 番の説明を参照してください。

SET SERVICE

既存のサービスの属性を変更します。

フォーマット

SET SERVICE　*serviceName [device-or-partitionName]*

パラメータ

serviceName
LAN 上で使用されるサービスの名前です。サービス名は英数字とドル記号 ($) で構成されます。長さは最大で 255 文字です。

device-or-partitionName
LAN 上で使用される OpenVMS のディスク・デバイスまたはパーティションの名前です。入力するデバイス名またはパーティション名は，作成済みのものでなければなりません。

デバイス名とパーティション名について，以下に説明します。

- デバイス名

  LAN にサービスを提供するデバイスは， OpenVMS ディスク・デバイスです。InfoServer デバイス名を指定するときには， OpenVMS デバイス名を使用します。デバイス名は， SHOW SERVICES コマンドで表示される名前に完全に一致するか，またはワイルドカードを含むかのいずれかでなければならないことに注意してください。

  InfoServer ユーティリティでワイルドカードが使用できるところでは，OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

  ディスク指定はコロンで終わる必要があります。

- パーティション名

  パーティションは，ネットワークにサービスが提供されるコンテナ・ファイルです。したがって，デフォルトのファイル・タイプが.ESS$PARTITION の OpenVMS ファイル名を持っています。デバイス名，ディレクトリ名，ファイル名を含むパーティション名の長さは，最大で 242 文字です。

  パーティション名は選択するサービスを特定する場合に使用できます。

今回のバージョンでは，パーティションのサポートは制限されています。パーティションに分割されたハード・ドライブをサポートする場合には， LD デバイスを使用することをお勧めします。詳細は， DCL コマンドの LD HELP を参照してください。

修飾子

/CLASS=className
完全な LASTport Disk (LAD) 名前空間のサブセットを指定します。

クラス名の目的は，名前空間を分割して，クライアントに対して意味のある名前だけを示すことです。クラス名を使用すると， 2 つのサービスに同じ名前を付けて，互いに競合しないようにすることもできます。

たとえば，いくつかのクライアント・システムで使用される異なるオンディスク構造に対して，異なるクラス名をつけることができます。あるクライアント・システムでは SERVICEA/CLASS=ODS-2 を使用し，別のクライアント・システムでは SERVICEA/CLASS=ISO_9660 を使用します。サービス名は同じ (SERVICEA) ですが，クラス名は異なります。

使用するクラス名は，作成するサービスに接続するクライアント・システムに依存します。デフォルトのクラス名は ODS_2 です。たとえば， OpenVMS システムは， InfoServer デバイスをマウントするとき， ODS_2 名前空間を使用します。OpenVMS クライアントは， ODS_2 サービス・クラスに属しているサービスのみ利用できることに注意してください。

有効なクラス名は，次のとおりです。

V2.0	PCSA MS-DOS クライアントが理解する名前
Unformatted	仮想ディスクは形式を持っていません
MSDOS	MSDOS 仮想ディスク
ODS_2	VMS 仮想ディスク
UNIX	UNIX 仮想ディスク
ISO_9660	ISO 9660 CD 形式
HIGH_SIERRA	MS-DOS CD 形式
APPLE	Macintosh HFS 形式
SUN	Sun 形式

/PASSWORD=passwordString
/NOPASSWORD
オプションのサービス・アクセス制御パスワードを指定します。パスワードが設定されたサービスにクライアント・システムがアクセスするためには，パスワードを指定する必要があります。

パスワードは最大 39 文字の英数字文字列です。パスワードを指定しなかった場合には，クライアント・システムではこのサービスにアクセスする場合に，パスワードの入力を求められません。

テキストのパスワードは暗号化された形式で，他のサービス情報とともにメモリに保存されます。

/RATING=DYNAMIC
/RATING=STATIC=value
該当するサービスが複数ある場合には，クライアントはサービス評価点を使用してサービスを選択します。最高のサービス評価点を持ったサービスが選択されます。

システムは動的サービス評価点を負荷に応じて調整します。

静的評価点を 0 ～ 65535 の範囲で指定することもできます。システムは静的評価点は調整しません。

/READAHEAD
/NOREADAHEAD
キャッシュ・ブロックを埋めるためのディスクの読み取りが必要になったときに，/READAHEAD 修飾子は，読み取り範囲が，要求された最初のブロックから，バケット境界の最後までであることを指示します。先読みすることによって，必要なディスク・ブロックが前もってキャッシュにロードされるので，シーケンシャル操作の速度が向上します。

/READAHEAD 修飾子と/READBEHIND 修飾子の両方を指定しておくと，キャッシュ・バケット内のブロックが要求されたときに，そのブロックのバケット範囲全体がキャッシュに読み込まれます。

/READBEHIND
/NOREADBEHIND
キャッシュ・ブロックを埋めるためのディスクの読み取りが必要になったときに，/READBEHIND 修飾子は，読み取り範囲が，キャッシュ・バケット境界の先頭から要求されたブロックまでであることを指示します。

/READAHEAD 修飾子と/READBEHIND 修飾子の両方を指定すると，キャッシュ・バケット内のブロックが要求されたときに，そのブロックのバケット範囲全体がキャッシュに読み込まれます。

/READERS=number
クライアントの読み取りアクセスで許される同時接続の最大数を指定します。

例

```
$ INFOSERVER SET SERVICE FUNDY/NOPASSWORD




    Service FUNDY [ODS-2] modified.

$ INFOSERVER SHOW SERVICES FUNDY/FULL

    FUNDY [ODS-2]                          Access: Read-only
      File or device: _MOVERS$LDA1: [750000 blocks]
      Flags: 00000000D2 {No Writers,Static Rating,Readbehind,Readahead}
      Rating:        Static,    42         Password:         Disabled
      Max Readers:             1000        Max Writers:             0
      Curr Readers:               0        Curr Writers:            0
      Reads:                      0        Writes:                  0
      Blocks Read:                0        Blocks Written:          0
```

この例の最初のコマンドは FUNDY サービスを変更して，クライアントがサービスにアクセスするときにパスワードを入力しなくてすむようにします。2 番目のコマンドは FUNDY サービスを表示します。パスワードの使用が無効になったことが示されています (SHOW SERVICES コマンドの例では， FUNDY サービスでパスワードの使用が有効になっていたことに注意してください)。

SHOW SERVER

サーバ(すなわち，サービスを提供するシステム)に関する情報を表示します。

フォーマット

SHOW SERVER

例

```
$ INFOSERVER SHOW SERVER


   Node MOVERS [COMPAQ Professional Workstation XP1000] running OpenVMS XALD-BL2

   LASTport/Disk Server Version 1.2

   Max Services:            64              Write Quota:          0
   Cache Buckets:         4096              Cache Bucket Size:   32 blocks
   Cache Size:        67108864 bytes
   Hits:                   478              Hit Percentage:      59%
   Misses:                 328

   Current Sessions:        0               Peak Sessions:          1

                          Read                    Write
   Requests:               40                      0
   Blocks:                319                      0
   Errors:                  0                      0
   Aborted:                 0                      0
   Conflicts:               0                      0
```

このコマンドは，クライアントにサービスを提供しているサーバに関する情報を表示します。表示される情報には，以下のものがあります。

- このサーバが同時に提供できるサービスの最大数
- 現在のキャッシュ・サイズ
- キャッシュの有効度についての統計情報
- 現在の同時接続クライアント数と過去の最大の同時接続クライアント数
- I/O 統計情報

SHOW SERVICES

SHOW SERVICES コマンドは，サーバが提供している 1 つまたはすべてのサービスについて，サービス固有の情報を表示します。この情報には，サービスに関連付けられたデバイス番号と，接続されているセッション数が含まれます。

SHOW SERVICES コマンドでは，ワイルドカード表現が使用できます。 InfoServer ユーティリティでワイルドカードが使用できるところでは， OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

フォーマット

SHOW SERVICES *[serviceName] [options...]*

パラメータ

serviceName
LAN 上で使用されるサービスの名前です。サービス名は英数字とドル記号 ($) で構成されます。長さは最大で 255 文字です。省略した場合には，サービス名のデフォルトは全サービスです。

InfoServer ユーティリティでワイルドカードが使用できるところでは， OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

修飾子

/BRIEF (**デフォルト**)
BRIEF オプションでは，選択したそれぞれのサービスについて，省略されたオンライン・サマリ情報が表示されます。 BRIEF がデフォルトです。

/FULL
FULL オプションでは，選択したそれぞれのサービスについて，サービス固有のすべての情報が表示されます。

例

1. INFOSERVER> SHOW SERVICES

```
Service Name         [Service Class] Device or File
-------------------- --------------- -------------------------
HUDSON                [ODS-2]         _MOVERS$LDA1:
BAFFIN                [ODS-2]         _MOVERS$LDA1:
FUNDY                 [ODS-2]         _MOVERS$LDA1:
3 services found.
```

このコマンドは，接続中のすべてのサービスについて，デフォルトの BRIEF オンライン・サマリを表示しています。

2. INFOSERVER> SHOW SERVICES/FULL

```
HUDSON [ODS-2]                          Access: Read-only
  File or device: _MOVERS$LDA1: [750000 blocks]
  Flags: 0000000082 {No Writers,Readahead}
  Rating:       Dynamic, 65535         Password:         Disabled
  Max Readers:            1000         Max Writers:             0
  Curr Readers:              0         Curr Writers:            0
  Reads:                     0         Writes:                  0
  Blocks Read:               0         Blocks Written:          0

BAFFIN [ODS-2]                          Access: Read-only
  File or device: _MOVERS$LDA1: [750000 blocks]
  Flags: 0000000082 {No Writers,Readahead}
  Rating:       Dynamic, 65535         Password:         Disabled
  Max Readers:            1000         Max Writers:             0
  Curr Readers:              0         Curr Writers:            0
  Reads:                     0         Writes:                  0
  Blocks Read:               0         Blocks Written:          0

FUNDY [ODS-2]                           Access: Read-only
  File or device: _MOVERS$LDA1: [750000 blocks]
  Flags: 00000000D2 {No Writers,Static Rating,Readbehind,Readahead}
  Rating:        Static,    42         Password:         Enabled
  Max Readers:            1000         Max Writers:             0
  Curr Readers:              0         Curr Writers:            0
  Reads:                     0         Writes:                  0
  Blocks Read:               0         Blocks Written:          0
3 services found.
```

このコマンドは，接続中のすべてのサービスについて，サービス固有のすべての情報を表示しています。 HUDSON サービスと BAFFIN サービスではパスワード

が無効で，FUNDY サービスでは有効になっていることに注意してください。

SHOW SESSIONS

サービスに接続されているクライアント・ノードに関する情報を表示します。

フォーマット

SHOW SESSIONS *[serviceName] [device-or-partitionName]]*

パラメータ

serviceName
LAN 上で使用されるサービスの名前です。サービス名は英数字とドル記号 ($)，およびワイルドカードで構成されます。長さは最大で 255 文字です。省略した場合には，サービス名のデフォルトはすべてのサービスです。

InfoServer ユーティリティでワイルドカードが使用できるところでは， OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

device-or-partitionName
デバイス名またはパーティション名は， LAN 上で使用される OpenVMS ディスク・デバイスまたはパーティションの名前です。入力するデバイス名またはパーティション名は，作成済みのものでなければなりません。

デバイス名とパーティション名について，以下に説明します。

- デバイス名

  LAN にサービスを提供するデバイスは， OpenVMS ディスク・デバイスです。InfoServer デバイス名を指定するときには， OpenVMS デバイス名を使用します。デバイス名は， SHOW SERVICES コマンドで表示される名前に完全に一致するか，またはワイルドカードを含むかのいずれかでなければならないことに注意してください。

  InfoServer ユーティリティでワイルドカードが使用できるところでは，OpenVMS で使用されるのと同じワイルドカードが使用できます。パーセント文字 (%) は正確に 1 文字に一致し，アスタリスク文字 (*) は 0 文字以上の文字列に一致します。

  ディスク指定はコロンで終わる必要があります。

- パーティション名

パーティションは，ネットワークにサービスが提供されるコンテナ・ファイルです。したがって，デフォルトのファイル・タイプが.ESS$PARTITION の OpenVMS ファイル名を持っています。デバイス名，ディレクトリ名，ファイル名を含むパーティション名の長さは，最大で 242 文字です。

今回のバージョンでは，パーティションのサポートは制限されています。パーティションに分割されたハード・ドライブをサポートする場合には， LD デバイスを使用することをお勧めします。詳細は， DCL コマンドの LD HELP を参照してください。

修飾子

/ALL
クライアントが接続していない場合でも，選択基準に該当するすべてのサービスを表示します。この修飾子を省略した場合には，クライアントが接続しているサービスだけが表示されます。

例

1.
```
$ INFOSERVER SHOW SESSIONS


HUDSON              [ODS-2]         _MOVERS$LDA1: [ 1 Connection]
1 service found.
```

2.
```
$ INFOSERVER SHOW SESSIONS/ALL


HUDSON              [ODS-2]         _MOVERS$LDA1: [ 1 Connection]
BAFFIN              [ODS-2]         _MOVERS$LDA1:
FUNDY               [ODS-2]         _MOVERS$LDA1:
3 services found.
```

最初の例では，このコマンドはクライアント接続 HUDSON を持つサービスだけを表示しています。 2 番目の例では，このコマンドは，クライアントが接続されていないサービスも含め，すべてのセッションを表示しています。

SPAWN

プロセスを生成して DCL コマンドを実行します。コマンドを指定しなかった場合には，生成したプロセスにコマンド端末が接続されます。コマンドを指定した場合には，そのコマンドが実行され，コマンドが完了すると，制御が親プロセスに戻ります。

フォーマット

SPAWN *[DCL Command]*

例

```
InfoServer> SPAWN DIRECTORY

        .
        .
        .
    (output)
        .
        .
        .

InfoServer>
```

このコマンドは，プロセスを生成して DCL コマンドの DIRECTORY を実行します。コマンドの実行が完了すると，制御が InfoServer プロセスに戻ります。

START SERVER

このコマンドは LASTport/Disk サーバを起動し，各種のサーバ特性とキャッシュ特性を設定します。

通常このコマンドは， SYS$STARTUP:ESS$LAD_STARTUP.DAT のデータを使用して， SYS$STARTUP:ESS$LAD_STARTUP.COM から実行されます。すべての変更は SYS$STARTUP:ESS$LAD_STARTUP.DAT ファイル内で行うことをお勧めします。

現在サービスが何も定義されていない場合には， START SERVER コマンドを会話型で使用して，修飾子を指定してサーバの設定を変更することもできます。

フォーマット

START SERVER

修飾子

/BUFFER_SIZE=n
InfoServer のブロック・キャッシュは固定長バッファ (**バケット**とも呼ばれる) の配列として構成されます。/BUFFER_SIZE 修飾子では各バケットのサイズを指定します (/CACHE 修飾子では，バケット数を指定します)。

このパラメータの数値は， 3 ～ 8 の範囲の整数値です。それぞれの整数値は，次のように， 512 バイトのブロックを単位としたバケット・サイズを表わします。

```
3 -   8 ブロック（デフォルト）
4 -  16 ブロック
5 -  32 ブロック
6 -  64 ブロック
7 - 128 ブロック
8 - 256 ブロック
```

32 ブロックより大きなバケット・サイズは，多くの場合，適切ではありません。 OpenVMS クライアントでは， 31 ブロックより大きな入出力要求を 31 ブロックのかたまりにセグメント化するため，デフォルトのバケット先読み動作では，ディスクに対する不要な入出力動作が発生するからです。

/CACHE = number-of-buckets (**デフォルト**= 512)
InfoServer のブロック・キャッシュは固定長バッファ (**バケット**とも呼ばれる) の配列として構成されます。/CACHE 修飾子では，キャッシュのバケット数を指定します。 (/BUFFER_SIZE 修飾子では各バケットのサイズを指定します。)

16384 より大きな数を指定すると，性能に悪影響を与える可能性があります。必要な大きさのキャッシュを確保するためには，/BUFFER_SIZE 修飾子の値を増加させることを検討してください。

/MAXIMUM_SERVICES = maxservice (**デフォルト**= 256)
サーバの最大サービス数を設定します。これが同時に定義できるサービスの最大数になります。各サービス記述子はページングされないプールを消費します。ただし，未使用のサービス・スロットは 4 バイトしか消費しません。

最大の値は 1024 です。

/WRITE_QUOTA = n (**デフォルト**= 0)
サーバで許される同時非同期書き込みの数です。デフォルトの 0 は，すべての書き込み操作が同期して行われることを意味します。

例

```
$ InfoServer SHOW SERVER



    Node BILBO [HP rx2600  (900MHz/1.5MB)] running OpenVMS XAR8-D2Y
    LASTport/Disk Server Version 1.2

    Max Services:        64        Write Quota:          0
    Cache Buckets:     2048        Cache Bucket Size:   32 blocks
    Cache Size:    33554432 bytes
    Hits:                0         Hit Percentage:       0%
    Misses:              0

    Current Sessions:    0         Peak Sessions:        0

                       Read             Write
    Requests:             0                0
    Blocks:               0                0
    Errors:               0                0
    Aborted:              0                0
    Conflicts:            0                0

$ InfoServer START SERVER/MAXIMUM_SERVICES=128/CACHE=2048/BUFF=5/WRITE=0

    %INFOSRVR-I-STARTED, LASTport/Disk server started.

$ InfoServer SHOW SERVER

    Node BILBO [HP rx2600  (900MHz/1.5MB)] running OpenVMS XAR8-D2Y
    LASTport/Disk Server Version 1.2
```

```
Max Services:    128        Write Quota:          0
Cache Buckets:  2048        Cache Bucket Size:   32 blocks
Cache Size: 33554432 bytes
Hits:               0       Hit Percentage:       0%
Misses:             0

Current Sessions:   0       Peak Sessions:        0

                  Read            Write
Requests:            0              0
Blocks:              0              0
Errors:              0              0
Aborted:             0              0
Conflicts:           0              0
```

この例の最初のコマンドは，サーバに関して現在の情報を表示します。2 番目のコマンドは，サーバを起動し，そのサーバのサービスの最大数を増加させます。3 番目のコマンドはサーバに関して新しい情報を表示します。この表示で，サービスの最大数が増加していることが分かります。

5
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，OpenVMS Alpha との相違点と考慮事項の概要について説明します。

この章の主なトピックは，次のとおりです。

- Alpha および VAX 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 5.2 節)
- OpenVMS I64 でのイメージのリンクの特徴 (第 5.3 節)
- Linker の新しい修飾子とオプション (第 5.4 節，第 5.5 節)
- Linker の既存の修飾子とオプションの新しい使い方 (第 5.6 節)
- I64 で拡張された Linker マップ・ファイル情報 (第 5.7 節)

Linker に関する注意事項については，第 3 章に加え， V8.2 の注意事項が含まれている『HP OpenVMS V8.2 リリース・ノート[翻訳版]』を参照してください。 V8.2 の注意事項のほとんどは本リリースにも適用されます。

5.1 Linker ユーティリティの概要

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 Alpha および VAX システムの Linker とは異なります。 OpenVMS Alpha および VAX 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・プロンプトで RUN コマンドを使用して起動できます。

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，入力オプション・ファイルに指定して，実行イメージや共有イメージを作成するリンク操作に含めます。

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， Alpha や VAX 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 Alpha および OpenVMS VAX システムの Linker と類似しています。

5.2 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
- 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
- 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=address
- UNIVERSAL=symbol-name

I64 Linker では，次のキーワードの意味が異なっています。

PER_PAGE – /DEMAND_ZERO=PER_PAGE での per_page キーワードは，各セグメントの後方のゼロを圧縮するように Linker に指示します (つまり，後方のページのゼロの圧縮を要求します)。

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。

5.2.1 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，Alpha システムと VAX 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 オプションでのベース・アドレスの指定は，すべてのイメージ・タイプで許されません。

5.2.2 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

注意

関連するリリース・ノートについては，第 3.8 節を参照してください。

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値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。

注意

I64 Linker の将来のリリースでは，セクション内の初期化済みの部分が一致した場合だけ，初期化を受け付けます。ゼロで初期化されたセクションは，ゼロ以外の初期値を持つ他のセクションとは一致しなくなります。例 5–1 に示す three.c での初期化は，one.c および two.c での初期化とは互換性がなくなります。

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

同じプログラム・セクション内で 2 つのロングワードを宣言している 2 つのモジュールがオーバレイされます。1 番目のモジュールは，プログラム・セクション内の 1 番目のロングワードを，ゼロ以外の値で初期化します。同様に，2 番目のモジュールは，プログラム・セクション内の 2 番目のロングワードを，ゼロ以外の値で初期化します。

Alpha システムと VAX 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・セクションを作成できます。 Alpha および VAX 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 システムの ELF には TIR コマンドが存在しないため， Linker はセクション内の初期化を実行せず関知もしません。逆に， Linker は初期化済みのセクション全体をコンパイラから受け取ります。 Linker はセクションに対するすべてのモジュール初期化情報を読み込み，初期化に互換性があるかチェックします。 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 は以下のエラー・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。

例 5–1 は，例 5–2 のマップ・ファイルの追加情報を示します。

例 5–1 追加情報

(次ページに続く)

例 5–1 (続き) 追加情報

```
$ cre one.c
int common_data[]={0,0,47,11};
void main (void) {return;}
Ctrl/Z
$ cc /extern=common one
$ cre two.c
int common_data[]={0,0,47,11};
Ctrl/Z
$ cc /extern=common two
$ cre three.c
int common_data[]={0,0,0,0,0,0,0,0};
Ctrl/Z
$ cc /extern=common three
$ link/map one,two,three
$
```

例 5–2 には，例 5–1 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 5–2 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
```

例 5–3 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。

例 5–3 互換性のない初期化 (1)

```
$ cre four.c
int common_data[]={0,0,47,11,0,0,17,4};
Ctrl/Z
$ 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
```

例 5–1 と例 5–3 は，外部オブジェクト共通ブロックモデルでコンパイルされています。 OpenVMS では，デフォルトの外部モデルは，緩やかな参照/定義 (ref/def) 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，警告メッセージ (重複定義メッセージ) が出力されます。前記の例と同じソース・ファイルを使用した例を例 5–4 に示します。

例 5–4 OpenVMS での初期化の例

```
$ 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
```

例 5–5 に，互換性のない初期化の別の例を示します。この例では， Linker は INVOVRINI エラーと MULDEF 警告の両方のメッセージを出力します。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。

例 5–5 互換性のない初期化 (2)

(次ページに続く)

例 5–5 (続き) 互換性のない初期化 (2)

```
$ 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
```

5.2.3 緩やかな参照/定義シンボルのリンク時の動作の相違点

緩やかな参照/定義 (ref/def) シンボル (I64 では「ELF 共通シンボル」とも呼ばれる) 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 ref/def 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に対して選択的にコンパイルおよび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 ref/def 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。

5.2.4 /TRACEBACK，/DEBUG，/DSF を使用し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，下記の表に示すフラグがイメージ・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， I64 で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。
- /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。/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 を指定した場合は，IMGSTA ビットだけがイメージに設定されます。

次の表は，デバッグ修飾子を使用するリンク操作で，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	イメージ内のグローバル・シンボル	DSF ファイル内のグローバル・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

5.3 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 5.3.1 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 5.3.2 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 5.3.3 項)
- 新しい/BASE_ADDRESS 修飾子の使用 (第 5.4.1 項)
- 新しい/SEGMENT_ATTRIBUTE 修飾子の使用 (第 5.4.2 項)
- 新しい/FP_MODE 修飾子の使用 (第 5.4.3 項)
- 新しい/EXPORT_SYMBOL_VECTOR 修飾子と/PUBLISH_GLOBAL_SYMBOLS 修飾子の使用 (第 5.4.4 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 5.4.5 項)
- PSECT_ATTRIBUTE オプションの新しいアラインメント (第 5.5.1 項)

5.3.1 リンケージ・メッセージの意味

一部の HP コンパイラは，呼び出し間での汎用レジスタの使い方に一貫性があることを確認できるように，呼び出しリンケージ情報を生成します。この情報により，Linker は，呼び出し元ルーチンのリンケージと呼び出し先ルーチンのリンケージに一貫性があることを確認できます。リンケージ情報は， Intel Itanium の汎用レジスタ 1 ～ 31 についてだけ生成されます。

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 はこ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

次のサンプル・メッセージでは， Intel Itanium のレジスタ 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。

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

前記の表に従えば， 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) を実行している場合にこのような問題が発生します。

Intel Itanium 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)，または呼び出し元のリンケージが不正であることを示す警告メッセージ (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。

Linkage 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。

このような衝突を避けるためには，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，リンクが問題なく行われるようにします。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。共有イメージに問題が残っている場合は，その共有イメージとリンクされる実行イメージに，SUCCESS 以外の完了ステータスが与えられます。

5.3.2 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。

5.3.3 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 (COMDAT とも呼ばれる) 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。

背景

オブジェクト・モジュール内の ELF グループは，プロシージャと変数の定義を含む，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++ コンパイラは単にグループとして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) を持ちます。イメージの場合は， Linker が，各グループに 1 つのコントリビューションを選択します。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，同一と考えられます。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。

UNIX スタイルの弱いシンボル，定義，および参照

グループ・シンボルは定義や参照として使用することができます。ただし， UNIX スタイルの弱い定義は， VMS スタイルの弱い定義とは異なり，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 の弱い参照と似ていて，未解決のままの場合にエラーや警告のメッセージは表示されません。 ELF では， Alpha および VAX のオブジェクト言語の弱い参照と定義は， VMS スタイルの弱いシンボルと呼ばれます。

Linker は，各グループの最初に出現したものを，共有イメージ・シンボルで置き換えられない限り，グループの通常シンボルおよび UNIX スタイルの弱いシンボルとともに，グループの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，すべて選択された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 スタイル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 スタイルの弱いシンボルも参照になります。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ため，これは問題にはなりません。このメカニズムを理解すると， Linker マップをより深く理解できます。グループ内に定義された UNIX スタイルの弱いシンボルは，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。

Linker マップの見方

マップの相互参照リストでは， UNIX スタイルの弱い定義と参照には，その前に UNIX スタイルの弱いシンボルであることを示すタグ (UNIX スタイルの weak タグ) が表示されます。通常は， UNIX スタイルの 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 スタイルの 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 スタイルの 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。

共有イメージのリンクについての考慮

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 スタイル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 はエラー・メッセージを出力し，イメージ・ファイルは書き出しません。

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で使用され，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た場合，ユーザ・エラーが発生します。このため， Linker はイメージを生成しません。

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，以前のリリースと同様に使うことができます。

5.4 OpenVMS I64 用 Linker の新しい修飾子

注意

関連するリリース・ノートについて，第 3.8 節を参照してください。

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，これらの修飾子および関連するキーワードについて説明します。

5.4.1 新しい/BASE_ADDRESS 修飾子

/BASE_ADDRESS 修飾子は， OpenVMS イメージ・アクティベータによって起動されないイメージ (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ます。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。 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]オプションのベース・アドレス指定子の代わりとなるものではありません。第 5.2.1 項を参照してください。

5.4.2 新しい/SEGMENT_ATTRIBUTE 修飾子

/SEGMENT_ATTRIBUTE 修飾子は，セグメントの属性の一部を設定します。この修飾子の構文は，次のとおりです。

```
/SEGMENT_ATTRIBUTE=(segment_attribute [, ...])
```

OpenVMS I64 Linker は， DYNAMIC=address_region， SHORT_DATA=WRITE， CODE=address_region，および SYMBOL_VECTOR=[NO]SHORT を，セグメント属性として受け付けます。アドレス領域は，キーワード P0 と P2 で指定できます。

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。 OpenVMS イメージ・アクティベータで起動されなかったイメージについては， DYNAMIC=P0 を指定すると， Linker はこの動的セグメントを P0 空間に配置します。この修飾子は，主にシステム開発者が使用します。

SHORT_DATA=WRITE キーワードは，読み取り専用のショート・データ・セグメントと読み書き用のショート・データ・セグメントを 1 つのセグメントに結合することで，最大 65,535 バイト (/BPAGE のデフォルト値) 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。

CODE=P2 キーワードを指定すると， I64 Linker は，コード・セグメントの P2 空間への割り当てを可能とします。イメージ・アクティベータがイメージを起動したときに，コード・セグメントが P2 空間に配置されます。このキーワードを使用すると，すべてのコード・アドレスが 64 ビット幅になることに注意してください。たとえば，例外ハンドラでは， 64 ビット版のシグナルとメカニズム配列だけを使用し，64 ビットの PC を取り扱うことができなければなりません。

デフォルトでは，共有イメージの場合， Linker はシンボル・ベクタを，読み取り専用のショート・データ・セグメントに格納します。 SYMBOL_VECTOR=NOSHORT を指定すると， Linker はシンボル・ベクタを，デフォルト・クラスタの読み取り専用データ・セグメントに収集します。共有イメージにこのようなセグメントがなければ，作成されます。これにより，シンボル・ベクタ・エントリのショート・データが解放されます。

5.4.3 新しい/FP_MODE 修飾子

OpenVMS I64 Linker は， main への転送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。メインへの転送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 修飾子は，初期浮動小数点モードを設定します。/FP_MODE 修飾子は，メインへの転送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，『HP OpenVMS Calling Standard』を参照してください。

5.4.4 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。たとえば，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い場合，どのシンボルをエクスポートすればよいかわからない可能性があります。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と，エクスポートするソース・コード・シンボルが対応付けられないといった場合があります。

注意

すべてのグローバル・シンボルをエクスポートすると，期待しない結果となることがあります。

UNIX システムでは，共有オブジェクトを作成するときにはすべてのシンボルをエクスポートするのが一般的です。しかし，OpenVMS システムでは，リンクのメカニズムやイメージの起動のメカニズムが異なっています。このため，OpenVMS の共有イメージの作成に UNIX のメカニズムを採用すると，正しく動作しないことがあります。すべてのグローバル・シンボルをエクスポートして生成された複数の共有イメージに対してアプリケーションをリンクしたときに，衝突が発生することがあります。これらの共有イメージには，同じ名前のユーザ生成シンボルやコンパイラ生成シンボルが含まれていることがあります。モジュールによるこれらのシンボルへの参照を Linker が解決しようとしたときに，複数の共有イメージにシンボル定義があると，リンク操作が失敗します。シンボルのエクスポートを，選択した少数のモジュール (または 1 つのモジュール) に限定しても，このような衝突が発生することがあります。

オープン・ソース・ソフトウェアの移植を手助けするために/EXPORT_SYMBOL_VECTOR 修飾子と/PUBLISH_GLOBAL_SYMBOLS 修飾子が作成されましたが，この衝突により，従来の方法でシンボル・ベクタを作成する際に，新たな問題が発生することがあります。この理由により，これらの修飾子は，I64 Linker の将来のリリースで廃止される予定です。

新しい/PUBLISH_GLOBAL_SYMBOLS 修飾子は，オブジェクト・ファイルやライブラリ・オブジェクト・モジュールにマークを付け，そのオブジェクトのすべてのグローバル・シンボルを Linker が生成するシンボル・ベクタ・オプション・ファイルに書き出すことを指定します。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 修飾子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，今後のリンク操作で用いる前に，GSMATCH 情報を付加して完成させておかなければなりません。

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 修飾子は，コマンド行からの使用も，オプション・ファイルでの指定も可能です。 Linker は，/EXPORT_SYMBOL_VECTOR 修飾子が指定されているときに，/PUBLISH_GLOBAL_SYMBOLS 修飾子がコマンド行またはオプションファイルで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 修飾子は，/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 I64 Linker に対して指示します。

/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 修飾子を指定すると，入力の SYMBOL_VECTOR=option 句は受け付けられません。

/PUBLISH_GLOBAL_SYMBOLS 修飾子を少なくとも 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，生成されるオプション・ファイルの名前として Liker が使用するファイル指定子を指定します。ファイル・タイプを指定しないと， Linker はタイプ.OPT を使用します。

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。

GSMATCH テンプレート・オプションを書き込んだ後で，生成されたオプション・ファイルを，共有イメージ・ファイルを作成する他のリンク操作で使用してください。

/PUBLISH_GLOBAL_SYMBOLS

/PUBLISH_GLOBAL_SYMBOLS 修飾子は，オブジェクト・ファイルや，ライブラリ・オブジェクト・モジュール，オブジェクト・モジュール・ライブラリ (つまり，ライブラリから抽出されたすべてのモジュール) にマーク付けし， Linker が生成するオプション・ファイル内のシンボル・ベクタ句によってグローバル・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，*HP 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 (最も考えられるケースは緩やかな ref/def 外部モデル) となっている場合，一時的な定義を行っているモジュールのいずれかに/PUBLISH_GLOBAL_SYMBOLS 修飾子が付加されていれば，そのシンボルは公開されます。 UNIX スタイルの弱いシンボルの定義が複数のモジュールにある場合も，同様です。ただしどちらの場合も，強いシンボル定義のあるモジュールが検出されると，一時的な定義は無効になります。シンボルがエクスポートされるかどうかは，定義を無効にしたモジュールに/PUBLISH_GLOBAL_SYMBOLS 修飾子があるかどうかによって決まります。

例

```
$ link/SHARE public/PUBLISH,implementation/EXPORT=public
```

この例では， 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
```

この例では，IMPLEMENTATION.OBJ 内のすべてのモジュール中のグローバル・シンボルのうち，以前に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
```

この例では，共有イメージに組み込まれる，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/PUBLISH/INCLUDE=public/EXPORT
```

この例では，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,
public/PUBLISH/SELECTIVE/EXPORT=public
```

この例では，PUBLIC.OBJ が処理された後，PUBLIC.OBJ 内のすべてのモジュールについて，参照された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 PUBLIC.OPT ファイルにエクスポートされます。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
Ctrl/Z
```

この例では，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。

5.4.5 /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 [=(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。
SECTION_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行うように指示します。
ALL	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。

GROUP_SECTIONS キーワードは，リンク操作に使用したすべてのグループをマップ・ファイルに含めるように， Linker に指示します。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。/FULL 修飾子のデフォルトは/FULL=SECTION_DETAILS です。 I64，Alpha，および VAX システムでのフル Linker マップには，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。

5.5 OpenVMS I64 の Linker オプションの拡張

OpenVMS I64 システムでのリンクをサポートするために，既存の Linker オプションに対して以下の拡張が追加されました。

5.5.1 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)。

5.6 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 5.6.1 項)
- イメージ名を指定する場合の規約 (第 5.6.2 項)

- アラインメントを指定する PSECT_ATTRIBUTE オプションの使用 (第 5.6.3 項)

5.6.1 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する必要がある場合は，デフォルトの動作 (すべての名前が大文字) を使用しないで， Linker の CASE_SENSITIVE オプションを使用して，大文字と小文字が混在した名前を指定できるようにします。たとえば，ライブラリ・インクルード文があり，ライブラリ内のモジュール名に大文字と小文字が混在している場合，次のように CASE_SENSITIVE オプションを指定して，名前に大文字と小文字が混在できるようにします。

```
CASE_SENSITIVE=YES
```

CASE_SENSITIVE オプションに YES を設定すると，オプション句の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。これらの文字は変更されずに，ファイル名，モジュール名，シンボル名，キーワードとして処理されます。 Linker のデフォルト動作 (オプション行全体を大文字に変換する動作) に戻すには， NO キーワード付きの CASE_SENSITIVE オプションを指定します。例を次に示します。

```
CASE_SENSITIVE=NO
```

NO キーワードを小文字 (no) で指定すると， Linker が認識しないため，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オプション・ファイルを扱う際に， 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。

Alpha および VAX での動作を維持するために， 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。

5.6.2 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。

5.6.3 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 オプション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
```

Alpha システムと VAX 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 は必ず，最低でもコンパイラが指定した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。

5.6.4 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$FIND_FILE が完了するのに時間がかかることがあり， Linker がハングしているように見えることがあります。リンクされているファイルの数と，ファイル指定で論理名を使用しているかどうかによりますが， LIB$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 を押して Linker プロセスを終了させることはできません。

どのファイルが存在しないかを調べるには，次の手順に従います。

1. LINK コマンドに SYS$INPUT:/OPTION を指定し，[Return]を押します。
   (Linker は，ユーザがリンク操作に対するオプション句を端末から入力するのを待ちます。)
2. オプション句を入力し，次の情報を含めます。
   - 1 行目として，次の行を指定します。

     ```
     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。

3. Ctrl/Z を押します。

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，SYS$INPUT:/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 (したがって，デフォルトで YES) 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 これら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_RESD$:[SYS.][COM]DOES_NOT_EXIST.OBJ; as input
-RMS-E-FNF, file not found
$
```

5.7 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，以降のサンプル・マップに示すように，次の情報が表示されます。

- 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 5–1 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                          V1.0              Lkg     Dnrm         360  28-OCT-2004 13:32  HP C V7.1-005
                 SYS$SYSROOT:[SYSMGR]GETJPI.OBJ;1
DECC$SHR                        V8.2-00           Lkg                    0  21-OCT-2004 11:23  Linker T02-17
                 SYS$COMMON:[SYSLIB]DECC$SHR.EXE;1
SYS$PUBLIC_VECTORS              X-3           Sel Lkg                    0  21-OCT-2004 11:23  Linker T02-17
                 SYS$COMMON:[SYSLIB]SYS$PUBLIC_VECTORS.EXE;1


                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
DECC$SHR                        LESS/EQUAL         1           1
SYS$PUBLIC_VECTORS              EQUAL           9114  3906113603
```

VM-1173A-AI

図 5–2 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      ! 7 !
                    +-----------------------+
```

VM-1174A-AI

図 5–3 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 5–4 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
        GETJPI          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
        GETJPI          00040000 00040000 00000000 (       0.) OCTA  4

$LITERAL$               00040000 00040017 00000018 (      24.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00040000 00040017 00000018 (      24.) OCTA  4

$READONLY$              00040020 0004002F 00000010 (      16.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00040020 0004002F 00000010 (      16.) OCTA  4

$LINKER_UNWIND$         00050000 00050017 00000018 (      24.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00050000 00050017 00000018 (      24.) QUAD  3

$LINKER_UNWINFO$        00050018 0005002F 00000018 (      24.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          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 5–5 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$TFRADR            00060050-R       WK-GETJPI
  FILLEN                00020000-R       GETJPI                   GETJPI
  FILLM                 00020010-R       GETJPI                   GETJPI
  GETJPI (U)            00000000         <Linker Option>
  INTERNAL_GETJPI       00060098-R       GETJPI                   GETJPI
  IOSB                  00020020-R       GETJPI                   GETJPI
  ITMLST                00010000-R       GETJPI
  STATUS                00020030-R       GETJPI                   GETJPI
  SYS$GETJPIW           0000009A-X       SYS$PUBLIC_VECTORS       GETJPI
```

VM-1177A-AI

図 5–6 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 !
                                  +---------------------+

          Symbols...
          ----------
Value     GETJPI (U)
-----     X-SYS$GETJPIW
00000000  X-DECC$TXPRINTF
0000009A  R-ITMLST
00000496  R-FILLEN
00010000  R-FILLM
00020000  R-IOSB
00020010  R-STATUS
00020020  R-ELF$TFRADR
00020030  R-INTERNAL_GETJPI
00060050
00060098

                                  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 5–7 イメージの概要

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34          Linker I02-17

                                  +--------------------+
                                  ! Image Synopsis !
                                  +--------------------+

                                  00010000 0006FFFF 00060000 (393216. bytes, 768. pages)
                                  00000000 00000000 00000000
                                  80000000 80010000 00010000 (65536. bytes, 128. pages)

                                        0. pages
                                        1.                1. (    1. block)
                                        2.                8. (    7. blocks)
                                  GETJPI V1.0
                                        5.
                                        3.
                                        9.
                                     3338.
                                       17.
                                        7.
                                     GETJPI
                              00000000 00060050
                              00000000 00030000
                              00000000 09800000 (IEEE DENORM_RESULTS)
                                 3332.
                              SHAREABLE, Global Section Match=EQUAL, Ident, Major=9120, Minor=2068313277
                              Image does not use RFP model
                              FULL WITH CROSS REFERENCE in file SYS$SYSROOT:[SYSMGR]GETJPI.MAP;1
                              441. blocks
```

VM-1179A-AI

図 5–8 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。モジュールの既存の属性や新しい属性を表示する 4 つの欄で構成される， Attributes というタイトルの新しい欄が追加されました。

4 つの欄の 1 番目には，モジュールのシンボルの検索が選択型 (selective) であったかどうかが示されます。シンボルの検索が選択型の場合には， Sel という略語が表示されます。シンボルの検索が選択型でなかった場合には，この欄は空白です。

2 番目の欄には，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。モジュールに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。モジュールが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，モジュールの浮動小数点モードが表示されます。この欄では，いくつかの略語が使われます。使用される略語の説明は， Object and Image Synopsis セクションの終わりにある Key for Attributes の説明に表示されます。次に示す例では，この欄に表示される可能性があるすべての略語が， Key for Attributes の説明にリストされています。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，前出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
NOFLOAT_CASE                                          Lkg RFP          504
                  DISK1:[JOE]NOFLOAT.OBJ;1
DNORM_CASE                                            Lkg     Dnrm     504
                  DISK1:[JOE]DENORM_W.OBJ;1
FAST_CASE                                             Lkg     Fast     504
                  DISK1:[JOE]FAST_W.OBJ;1
NEPCT_CASE                                            Lkg     Inex     504
                  DISK1:[JOE]INEXACT_W.OBJ;1
SPCL_CASE                                             Lkg     Spcl     504
                  DISK1:[JOE]SPECIAL_W.OBJ;1
UNDER_CASE                                            Lkg     Undr     504
                  DISK1:[JOE]UNDERFLOW_W.OBJ;1
DG_FL_CASE                                            Lkg     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，リンク操作で組み込まれ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っている可能性があるため，表示されます (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，メジャー・バージョン番号およびマイナー・バージョン番号とともに， Global Section Match (GSMATCH) の適合条件が表示されます (この情報がある場合)。

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_ONLY でリンクされており，イメージに非標準の signature 情報が含まれている場合には，それを含む (SHORT で示される) ショート・データ・セグメントの直後に独立したセグメントが表示されることがあります。

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 ref/def 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 ref/def 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 が使われますが， 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 回だけ表示されます。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。 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ます。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， I64 では廃止されました。

  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 スタイルの弱いシンボルが， OpenVMS I64 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 スタイル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 スタイルの弱いシンボルは，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 は通常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にどの制限値を増加する必要があるかを示します。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
```

6
製品ディレクトリ

この章では， OpenVMS Version 8.2-1 メディア・キットに関して以下の情報を説明します。

- インストール情報の参照先
- OpenVMS I64 Operating Environment DVD のディレクトリ構造
- OpenVMS Freeware CD と関連情報の参照先
- OpenVMS Open Source Tools CD について

  この CD には，他の OpenVMS キットのいずれにも含まれていない各種のオープン・ソース・プロジェクトの OpenVMS 版が収録されています。
- 製品のライセンス方式
- ドキュメント

6.1 OpenVMS I64 オペレーティング環境

ここでは， OpenVMS I64 OE DVD で OpenVMS I64 オペレーティング環境が格納されている場所について説明し，その DVD で提供されるすべての製品の名前と場所をリストします。

OpenVMS I64 オペレーティング環境のインストールについては，『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』を参照してください。また， OpenVMS I64 オペレーティング環境をインストールする前に，本書と『HP OpenVMS V8.2 リリース・ノート[翻訳版]』で説明している注意事項に目を通しておいてください。

6.1.1 OpenVMS I64 Operating Environment DVD のディレクトリ

最上位のディレクトリ，ドキュメント・ディレクトリ， SPD (OpenVMS Software Product Description) の OpenVMS ドキュメント・ファイル名を表 6–1 に示します。

表 6–1 OpenVMS I64 Operating Environment DVD のディレクトリ構造

ディレクトリ	ファイル名/内容
[.000000]	I64 8.2-1 OS
[.AVAILMAN_I640241]	Availablity Manager Version 2.4–1
[.VMSI18N_I640821]	C/C++ I18N
[.CDSA_I64021]	CDSA Version 2.1
[.CSWS_JAVA_I64021]	CSWS_JAVA (別名 Tomcat) Version 2.1
[.CSWS_PERL_I640111]	CSWS_PERL Version 1.1–1
[.CSWS_PHP_I640121]	CSWS_PHP Version 1.2.1
[.DCE_I64032]	DCE Version 3.2
[.DCPS_I64024A]	DCPS Version 2.4A
[.DECNET_PHASE_IV_I640821]	DECnet Phase IV Version 8.2-1
[.DECNET_PLUS_I64_V0821]	DECnet-Plus Version 8.2-1 FTAM，OSAK，VT を含む
[.DWMOTIF_I64015]	DECwindows Motif Version 1.5
[.ENTERPRISE_DIR_I64054]	Enterprise Directory Version 5.4ECO1
[.JAVA_I640142]	Java™ Version 1.4.2–1
[.KERBEROS_I64021]	Kerberos Version 2.1
[.MGMTAGENTS_I64032]	Management Agents Version 3.2
[.NETBEANS_I64036]	NetBeans Version 3.6
[.PERL_I640561]	Perl Version 5.6.1
[.SWB_I64014]	Secure Web Browser Version 1.4
[.SWS_I640131]	Secure Web Server Version 1.3–1
[.SOAP_020]	SOAP Toolkit Version 2.0
[.SSL_I64012]	SSL Version 1.2
[.TCPIP_I64055]	TCP/IP Services for OpenVMS Version 5.5
[.TDC_I64021]	TDC (The Performance Data Collector) Version 2.1-XX
[.UDDI_I64010]	UDDI4J Version 1.0
[.XML_I64020]	XML–J Version 2.0
[.XML_I64020]	XML–C Version 2.0
[.I640821.DOCUMENTATION]	.PS 形式と.TXT 形式の製品ドキュメント OVMS_V821_INSTALL.[PS,TXT] 『OpenVMS Upgrade and Installation Manual』 OVMS_V821_NEW_FEATURES_REL_ NOTES.[PS,TXT] 『OpenVMS New Features and Release Notes』 OVMS_V821_SPD.[PS,TXT] 『OpenVMS SPD』 CLUSTER_SPD.[PS,TXT] 『OpenVMS Cluster Software SPD』 DECram_SPD.[PS,TXT] 『DECram Software SPD』 VOLUME_SHADOWING_SPD.[PS,TXT] 『Volume Shadowing for OpenVMS SPD』

6.2 OpenVMS Freeware CD のディレクトリ

OpenVMS Version 8.2-1 メディア・キットには，3 枚の OpenVMS Freeware Version 7.0 CD が含まれています。 Freeware CD にはサポート・サービスが提供されない各種のソフトウェア・ツールやユーティリティ，および一般に普及しているオープン・ソース・パッケージの OpenVMS 版が収録されています。これらのパッケージはテクニカル，エンジニアリング，教育などの各種の分野で役に立ちます。

個々のフリーウェア・パッケージに関する情報は，該当する FREEWARE_README.TXT ファイルを参照してください。このファイルにアクセスするには，該当する CD を CD ドライブに挿入し，マウントした Freeware ボリュームに応じて以下のコマンドを実行します。下に示した MOUNT コマンドでは， ddcu: 指定は，ご使用中の OpenVMS システムにある特定の CD デバイスまたは DVD デバイスのデバイス名を表わしています。

```
$ MOUNT/OVERRIDE=IDENTIFICATION ddcu:
$ TYPE ddcu:[FREEWARE]FREEWARE_README.TXT
```

該当する CD がマウントできたら，標準の DCL コマンド (たとえば， DIRECTORY コマンド) を使って， CD の内容とディレクトリ構造を表示することができます。あるいは，対象とするボリュームに応じて次のコマンドのいずれかを実行して，Freeware メニュー・システムを起動することもできます。

```
$ @DISK$FREEWARE70_1:[FREEWARE]FREEWARE_MENU
$ @DISK$FREEWARE70_2:[FREEWARE]FREEWARE_MENU
$ @DISK$FREEWARE70_3:[FREEWARE]FREEWARE_MENU
```

パッケージ固有のライセンス情報および関連情報は，個々のパッケージで調べてください。

これらのパッケージと旧版の Freeware ディストリビューションのパッケージのコピー，今後の Freeware ディストリビューションの提供予定情報，今回の Freeware ディストリビューションのアップデート情報は，次の Web サイトから入手できます。

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

6.3 Open Source Tools CD

Open Source Tools CD は OpenVMS の技術者によって OpenVMS に移植されたオープン・ソース・ツールを集めたものです。これらのオープン・ソース・ツールは GNU Lesser General Public License の条項にもとづいてフリー・ソフトウェアとして提供されます。ユーザは Free Software Foundation Version 2.1 of the License によって公開されている GNU Lesser General Public License の条項にもとづいて，再配布や変更を行うことができます。

弊社では，このライブラリが役に立つものと期待して配布しています。ただし，弊社ではこれらのツールの使用した結果に関する保証はしていません。特殊目的に対する市場性，適合性などについても，一切の保証をいたしかねます。詳細は， Open Source Tools CD の GNV キット・ディレクトリにある GNU Lesser General Public License を参照してください。

Open Source Tools CD には，以下のコンポーネントが収録されています。

- GNV— オープン・ソース， GNU ベースの OpenVMS 用 UNIX 環境です。 UNIX のアプリケーション開発者，システム・マネージャ，ユーザに， UNIX スタイルの環境を提供します。これを使うと，ソフトウェアの開発や UNIX ソフトウェアの OpenVMS へのポーティングが容易になります (GNU は UNIX に似たオペレーティング・システムで，フリー・ソフトウェアです)。 GNV は， UNIX に似たシェル (コマンド行インタプリタ) 環境と，通常 UNIX システムが備えている C ランタイム・ライブラリ (CRTL) の実用的な補助ライブラリを提供します。 GNV で使用されているシェルは bash です (Bourne-Again SHell， GNU が開発， POSIX.2 仕様に準拠)。 Open Source Tools CD には， OpenVMS Alpha 用と OpenVMS I64 用の 2 種類の GNV キットが収録されています。
- IAS (Intel Itanium Assembler/Deassembler)—Intel が提供しているオープン・ソース Itanium アセンブラの OpenVMS I64 版です。これを使用すると，ローレベルの Itanium アセンブラ・コードが記述できます。このアセンブラにはいくつかの機能が追加され， OpenVMS I64 での実用性が向上しています。新機能に関する補足情報とアセンブラの使用方法はキットに付属しています。
- STUNNEL—OpenVMS システムから別のコンピュータへの SSL (Secure Sockets Layer) 接続内で，任意の TCP 接続を暗号化するためのプログラムです。 Stunnel を使用すると， SSL 対応でないアプリケーション (たとえば， Telnet， IMAP， LDAP) もセキュアにすることができます。このとき，オリジナルのアプリケーションを変更することなく，暗号化機能を組み込むことができます。イメージとソースの両方が用意されています。

Open Source Tools CD には，以下のソフトウェアも収録されています。

- SSL (Secure Sockets Layer) のソース —OpenVMS Version 8.2-1 に，オプションのレイヤード・プロダクトとして含まれています。 CD には， OpenVMS 版の SSL Version 1.2 のソースが収められています。
- CD-Record のソース —OpenVMS Version 8.2-1 の一部として組み込まれています。 CD レコード・イメージを作成するために使用するフル・ソース・キットが提供されています。
- GnuPG (GNU Privacy Guard)— セキュアな通信とデータ・ストレージのための GNU ツールです。データを暗号化したり，デジタル署名を作成するために使用できます。 GnuPG には高度な鍵管理機能が含まれています。 GnuPG は PGP の完全な代替品で無償です。特許が認められた IDEA アルゴリズムを使用していないため，制限なしで使用することができます。 GnuPG は RFC 2440 (OpenPGP) に準拠したアプリケーションです。

- CDSA (Common Data Security Architecture) のソース — OpenVMS Version 8.2-1 の一部として組み込まれています。 CD には CDSA for OpenVMS Version 2.1 を作成するために使用されたフル・ソース・キットが収められています。
- Kerberos のソース —OpenVMS Version 8.2-1 の一部として組み込まれています。 CD には Kerberos for OpenVMS Version 2.1 を作成するために使用されたフル・キットが収められています。
- GTK+—GUI を作成するための，オープン・ソースのフリー・ソフトウェア・ライブラリです。
- libIDL—IDL コンパイラ用のライブラリで， CORBA Interface Definition Language (IDL) ファイルのツリーを作成するために使用する，オープン・ソースのフリー・ソフトウェア・ライブラリです。
- [.000TOOLS]ディレクトリにあるフリーウェア・ツールの tar と zip。

6.3.1 Open Source Tools CD のディレクトリ

OpenVMS Open Source Tools Version 3.0 CD の内容とディレクトリを表 6–2 に示します。

表 6–2 OpenVMS Open Source Tools Version 3.0 CD

製品	ディレクトリ
CD-Record のソース	[.CDRECORD_SOURCE]
CDSA のソース	[.CDSA_SOURCE]
解凍 (unzip) その他のツール	[.000TOOLS]
GnuPG	[.GNUPG]
GNV for I64 Version 1.6–4	[.GNV_I64]
GTK+	[.GTK]
IAS (Intel Assembler Source)	[.IAS]
Kerberos のソース	[.KERBEROS_SOURCE]
libIDL	[.LIBIDL]
SSL のソース	[.SSL_SOURCE]
Stunnel	[.STUNNEL]

6.4 製品のライセンス方式

OpenVMS I64 メディアに収められているソフトウェアは，弊社に帰属します。ソフトウェアの使用は，各製品のソフトウェア・ライセンスを弊社から取得している場合に限り認められます。

LMF (License Management Facility) の PAK (Product Authorization Key) によって，ソフトウェア製品の使用が認められます。この CD から関連ソフトウェアをインストールする前に， PAK を登録してロードする必要があります。 PAK を取得するには，弊社のサポート担当，または弊社の各支店/営業所にお問い合わせください。

OpenVMS I64 のライセンス方式は， OpenVMS オペレーティング・システムといくつかのレイヤード・プロダクトが，オペレーティング環境として提供されるという点で， OpenVMS Alpha のライセンス方式とは異なります。オペレーティング環境には， Foundation Operating Environment (FOE)， Enterprise Operating Environment (EOE)，それに近日中に提供が開始される Mission Critical Operating Environment (MCOE) があります。 EOE は FOE の内容を完全に含み，他にいくつかのレイヤード・プロダクトが追加されています。 MCOE は EOE の内容を完全に含み，他に OpenVMS Cluster と Reliable Transaction Router (RTR) が追加される予定です。

FOE のライセンスと追加レイヤード・プロダクトのライセンスを購入することができます。あるいは必要なレイヤード・プロダクトに応じて， EOE または MCOE を選択することもできます。オペレーティング環境とその内容についての詳細は，次の Web サイトにある『HP Operating Environments for OpenVMS Industry Standard 64 Version 8.2-1 for Integrity Servers SPD』を参照してください。

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

6.5 OpenVMS Version 8.2-1 のドキュメント

OpenVMS Version 8.2-1 のドキュメントは， OpenVMS Version 8.2 で提供されたマニュアルに，それを補足する 2 冊の新規マニュアルを追加して構成されています。新規マニュアルには Version 8.2-1 をインストールして使用するために必要な新しい情報が説明されています。

OpenVMS Version 8.2-1 メディア・キットには，以下のリリース・ドキュメントが含まれています。

- ハードコピー・ドキュメント
  - 『HP OpenVMS Version 8.2-1 for Integrity Servers Upgrade and Installation Manual』
  - 『HP OpenVMS V8.2-1 新機能およびリリース・ノート[翻訳版]』
  - HP OpenVMS Version 8.2-1 Integrity Server のカバー・レター
  - 『HP OpenVMS License Management Utility Manual』
  - 『HP OpenVMS V8.2 リリース・ノート[翻訳版]』

- オンライン・ドキュメント
  - Online Document Library for HP OpenVMS I64 and Microsoft Windows Platforms CD

ドキュメント・セット内のすべてのドキュメントが各リリースに合わせて改訂されるわけではないことに注意してください。通常，新しいリリースにはいくつかの新規ドキュメントと改訂されたドキュメントが含まれ，残りは旧リリースのままのドキュメントが含まれます。たとえば， OpenVMS Version 8.2 セットには，いくつかの新規ドキュメントと改訂されたドキュメントが含まれ，そして残りのドキュメントは旧リリースのままになっています。各セット内のドキュメントが，利用可能な最新のバージョンです。

また，すべてのドキュメントは， HP OpenVMS Systems Documentation Web サイトから入手できます。

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

索引

O

P

R

S

T

V

ア

イ

オ

カ

キ

ク

サ

シ

セ

ソ

テ

ト

ネ

フ

ヘ

マ

メ

ラ

リ

レ

HP OpenVMS V8.2-1 **新機能およびリリース・ノート【翻訳版】**

2005 年 10 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)

BA322-90035