ORACLE
SOLARIS

Oracleホワイト・ペーパー
2012年5月

Oracle Solaris 11 ISV用導入ガイド

ORACLE

概要

このガイドは、独立系ソフトウェア・ベンダー（ISV）によるアプリケーションのOracle Solaris 10からOracle Solaris 11へのスムーズな移行を支援する目的で作成されています。このガイドには、既存のアプリケーションをOracle Solaris 11上で実行するために必要になる可能性のある、現時点の既知の修正がすべて示されています。大半のアプリケーションは、修正せずに使用できます。ただし、旧世代のフレームワークや文書化されていないAPIに依存しているアプリケーションについては、若干の変更が必要になる場合があります。

はじめに

Oracle Solaris 11は、Oracle Solarisのメジャー・リリースであり、OpenSolarisおよびOracle Solaris 11 Expressを通じて、近年公開されたすべての技術革新とOracle Solaris 10で使用されている既知の機能が統合されています。大半のアプリケーションは、変更せずに、Oracle Solaris 11で使用できることが分かっています。さらに、Oracle Solaris 11では、次の2つの主要オプションにより、ほとんどのアプリケーションをOracle Solarisで使用できます。

- Oracle Solaris 10の大半のアプリケーションは、Oracle Solaris 11において“そのまま”で使用可能
- Oracle Solaris 11で直接サポートされないOracle Solaris 10の機能に依存しているアプリケーションは、Oracle Solaris 10仮想環境（Oracle Solaris 10ゾーン）で使用可能

この2つのアプローチにより、以前のOracle Solarisのメジャー・リリースに比べて、Oracle Solaris 11では厳密な互換性に関する問題は少なくなっています。このガイドでは、“アプリケーションを使用可能にする”ことの実際的な側面に特に重点を置いています。このガイドでは、スナップショットを示しながら、以下の理由から構成変更が必要になる可能性のある、判断しにくい既知の事例について説明しています。

- デフォルトの構成および設定の変更（例：デフォルト・シェル、ZFSルート・ファイル・システム）
- パッケージの除外（例：廃止された物理デバイス）
- 業界標準の進化（例：標準化されたロケール名）
- 推奨されなくなった機能の削除
- 文書化されていない機能やサポートされない機能の削除
- 新しいテクノロジーによって生まれた判断しにくい事例（例：ZFSルート・ファイル・システムのデフォルト）
- バグの修正およびセキュリティの向上（アクセス権および使用権の厳格化）
- Image Packaging System（IPS）と呼ばれる新しいインストール・テクノロジー

免責事項：このガイドは、Oracle Solaris 11の後続のリリースが一般に提供されるようになると更新されます。

推奨される管理方式

リリース管理者は、次のような管理方式を採用できます。

- このガイドをおもに全体的な“チェックリスト”として使用し、詳細についてはOracle Solaris製品マニュアル自体を参照する。
- Oracle Solaris 11の互換性チェック・ツールを使用して、Oracle Solaris 10のアプリケーションがOracle Solaris 11に対応していることを確認する。
- Oracle Solaris 10とOracle Solaris 11の最新のリリース・ノートで、機能の廃止に関する項を参照する（将来的に削除される予定の機能がすべて示されています）。
- Oracle Solaris 11に関連する機能の廃止を通知するWebページ[1]を参照する（Oracle Solaris 11で削除されるフレームワークおよび機能について説明する新しいサービスです）。

Oracle Solarisのバイナリ互換性保証

Oracle Solaris 11 ExpressおよびOracle Solaris 11は、旧バージョンのOracle Solarisとバイナリ互換性があります[2]。これは、既存のアプリケーション・バイナリが、公開されている標準のOracle Solaris APIおよびABI（次の項を参照）に従ってプログラミングされている限り、再コンパイルしなくてもOracle Solaris 11上で動作することを意味します。

Oracle Solarisアプリケーション・バイナリ・インタフェース

アプリケーション・バイナリ・インタフェース（ABI）[3]は、アプリケーションとオペレーティング・システムまたは他のアプリケーションとの間の実行時インタフェースを定義します。ABIは、一連の構築時のアプリケーション・プログラミング・インタフェース（API）を定義するだけでなく、基本的なデータ型（int、floatなど）のサイズ、データ構造のレイアウトとアライメント、スタックの構成、コール規約、レジスタ割当て、システム・コール・メカニズム、および実行可能ファイル形式も定義します。

Oracle Solarisは、4つのABI（32ビットおよび64ビット・バージョンのSPARCアーキテクチャとx86/x64アーキテクチャ）を定義します。各Oracle Solaris ABIは、アプリケーションがOracle Solarisオペレーティング・システムで利用できる、次のような、サポートされる一連の実行時インタフェースで構成されます。

- UNIXマニュアルのセクション3で定義される、Oracle Solarisシステム・ライブラリAPI
- UNIXマニュアルのセクション2で定義される、Oracle Solarisカーネルによって提供されるシステム・コール
- UNIXマニュアルのセクション4に記載される、さまざまなシステム・ファイルの場所と形式
- UNIXマニュアルのセクション1に記載される、Oracle Solarisユーティリティの入出力用の構文と意味

[1] http://www.oracle.com/technetwork/jp/systems/end-of-notices/eonsolaris11-392732-ja.html
[2] http://www.oracle.com/us/products/servers-storage/solaris/solaris-guarantee-program-1426902.pdf
[3] http://docs.oracle.com/cd/E38900_01/html/E38879/index.html

アプリケーション開発者にとって、Oracle Solarisシステム・ライブラリによって定義されるC言語のAPIは、Oracle Solaris ABIの最重要部分です。他の言語で作成されるアプリケーションは、これらのC言語のAPIを使用して、Oracle Solarisカーネルとインタフェースを取る必要があります。

Oracle Solarisは、旧バージョンのOracle Solaris上で開発され、構築されたOracle Solaris ABIに準拠するアプリケーションが同じアーキテクチャ上のOracle Solaris 11において変更なしに動作することを保証する、バイナリ互換性保証を提供します。この保証が適用されるには、アプリケーションが次の制限を満たしている必要があります。

- アプリケーションは、動的にリンクされていなければならない（アプリケーションおよびそのコンポーネント・ライブラリのどれもがOracle Solarisのどのシステム・ライブラリとも静的にリンクされていてはならない）
- アプリケーションは、認定されたもの、標準のもの、または安定したものとして文書化されたAPI機能のみを使用していなければならない（インタフェースの安定性についてはattributes(5)のマニュアル・ページを参照）
- アプリケーションは、認定されていない、不安定な、または発展中のAPIインタフェースを使用してはならない
- アプリケーションは、**private**インタフェースまたは文書化されていないインタフェースを使用してはならない

Oracle Solarisには、アプリケーションがOracle Solaris ABIに準拠していることをチェックするための多数のツールがあります。

Oracle Solaris 10では、appcertおよびapptraceツールを使用して、アプリケーションがABIに準拠していることを静的および動的にチェックできます。これにより、アプリケーションをOracle Solaris 11上でテストする前に、Oracle Solaris 10を実行するシステムでチェックできます。Oracle Solaris 11には、アプリケーションの互換性をチェックするための新しいツールが用意されています。Preflight Application Checkerツールは、ダウンロード[4]して入手できます。このパッケージには、Oracle Solaris 11でアプリケーションの互換性をチェックする方法を記載したドキュメントが含まれています。

Oracle Solarisゾーンのサポート

この項では、ソフトウェア開発者がアプリケーションをOracle Solaris 10からOracle Solaris 11に移行させる際に注意する必要がある、Oracle Solaris 11における変更点について説明します。

Oracle Solarisゾーン（Oracle Solaris 10ではOracle Solarisコンテナとも呼ばれます）は、Oracle Solarisシステムにおいてユーザーレベルのワークロードを分離する軽量の仮想マシンです。これらは、Oracle Solaris 10に含まれるOracle Solarisゾーン・テクノロジーに基づいています。Oracle Solarisゾーンは、ハイパーバイザーに依存せずに、抽象化されたハードウェア・リソースを提供するとともに、ゾーン自体をネイティブ・ホスト・システムから分離し、ゾーンを相互に分離する点で、他の仮想化テクノロジー（VirtualBox、Oracle VM Server for SPARCおよびx86など）によって作成される仮想マシンと著しく異なります。

[4] Oracle Solaris 11 Express ISV用導入ガイド

Oracle SolarisゾーンはOracle Solarisカーネルに組み込まれており、カーネルのサブシステムの多くがゾーンに対応しています（つまり、抽象化とゾーンを関連付け、部分的にそのような関連付けに基づいて判断します）。その結果、ゾーン内で実行されるプロセスにおいて、オーバーヘッドがほとんどまたはまったく発生しないため（高く見積もっても総実行時間の5%）、ベアメタル・パフォーマンスの実現に近づきます。さらに、オーバーヘッドがないことから、Oracle Solarisゾーンは非常にスケーラブルであり、ローエンドの一般消費者向けデスクトップ・コンピュータでも、何十ものゾーンを同時に実行できます。ゾーン化されたプロセスによる実行オーバーヘッドが極めて少ないこと、ゾーンの作成および管理が比較的簡単であること、およびOracle Solarisゾーン・テクノロジーの完成度により、Oracle Solarisゾーンは、Oracle Solaris 10およびOracle Solaris 11でサポートされるもっとも一般的な仮想化テクノロジーとなっています。

ただし、ゾーンには、いくつかのよく知られている制限事項があります。さらに、一部のOracle Solarisカーネル・サブシステムがゾーンに対応していないため、ゾーン内で完全に分離され、サポートされるリソースの種類に制限があります。最後に、通常のゾーンは、非ネイティブの主要バージョンのOracle Solarisからユーザー環境をホストできません。最後の制限は、その大部分を、“ブランド”ゾーンを使用することによって回避できます（次の項を参照）。

Oracle Solaris 11から削除されたOracle Solarisゾーン機能

非大域ゾーンで大域ゾーンとは異なるオペレーティング環境を実行することを可能にする“ブランド・ゾーン”の概念が、Oracle Solaris 10 Update 4において導入されました。

次の非大域ゾーンのブランドは、Oracle Solaris 11では廃止されました。

- Linuxアプリケーション用のOracle Solarisコンテナ（“lx”）
- Oracle Solaris 8コンテナのブランド（“solaris8”）
- Oracle Solaris 9コンテナのブランド（“solaris9”）

Oracle Solaris 11ゾーンの依存関係

Oracle Solaris 11ゾーンは、新しいImage Packaging Systemを使用して提供されます。これにより、Oracle Solaris 10で利用されていた“完全ルート・ゾーン”モデルと“疎ルート・ゾーン”モデルの間での選択は不要になりました。Oracle Solaris 11ゾーンを使用することで、管理者は、両方の長所を活用し、固有のアプリケーション要件に適合するゾーンを構成できます。たとえば、疎ルート・ゾーンの特性は、Oracle Solaris 11ゾーンにおいて、ZFSファイル・システム、新しいIPS、およびループバック・ファイル・システム・マウントの使用の組合せによって再現できます。この構成では、インストールされたパッケージと、データの“重複排除”を正確に制御することができます。これらの2つの機能によって、複数のゾーンのインストールによるシステムへの影響が大幅に軽減されます。その結果、Oracle Solaris 11ゾーンは、より完全で、より高度に構成可能なものになっています。

- ゾーン・ルートは、ZFSデータセットでなければなりません。つまり、ZFSボリュームまたはZFSファイル・システムのいずれかです。特に、UFSはサポートされなくなりました。
- このガイドの作成時点では、オンディスクIPSリポジトリは、Oracle Solaris 11に統合されていません。つまり、Oracle Solaris 11ゾーンをインストールするには、ホスト・システム用のネットワーク・アクセスが必要です。

Oracle Solaris 11ゾーンの新機能

この項では、Oracle Solaris 11ゾーンの新機能を、ISVへの影響の観点から整理して説明します。

ISVに大きな影響を与える機能

- 非大域ゾーン

  Oracle Solaris 11では、非大域ゾーンの提供方法が変更されています。大域ゾーンと同様に、非大域ゾーン内のシステム・ソフトウェア・パッケージは、Oracle Solaris 11における新しいソフトウェア・ライフ・サイクル管理フレームワークであるImage Packaging Systemによって管理されます。Oracle Solaris 10と同様に、非大域ゾーン内の特定のパッケージは、大域ゾーンと同期が取れた状態を維持する必要があります。IPSによって、この処理がシームレスに実行され、適切なパッケージの同期状態が自動的に維持されます。

  非大域ゾーンの以下の特性については、注意が必要です。

  – IPSを使用してソフトウェア・パッケージを管理します。

  – 必要なすべてのソフトウェアが、ゾーンのプライベート・ファイル・システムにインストールされます。

  – IPSにより、Oracle Solarisゾーンの更新が非常に簡単であり、ほとんどの場合、大域ゾーンのパッケージ更新時に自動的に実行されます。大域ゾーンの場合と同様に、システムの再起動が必要な更新の場合は、更新時に、非大域ゾーンごとに新しいゾーンBEが作成されます。これらのゾーンBEは、大域ゾーン内の親BEの子BEです。

  – ゾーンの作成時には、最小限のシステム・ソフトウェアがゾーンにインストールされます。ゾーンに必要なすべての追加パッケージは、IPSコマンドによってゾーンが最初に起動された後に追加する必要があります。

- 組込みのネットワーク仮想化およびリソース管理

  Oracle Solaris 11では、以前に“Crossbow”と呼ばれていた、新しいネットワーク・スタック・アーキテクチャが導入されています。

  この新しいアーキテクチャにより、ゾーンと緊密に統合される仮想NICが追加され、非常に柔軟なネットワーク仮想化が実現されます。さらに、この新しいアーキテクチャによって、帯域幅とフローの制御によるリソースの管理が可能になります。この新しいアーキテクチャによって、真のOS仮想化、ユーティリティ・コンピューティング、およびサーバー統合の実現に必要なリソース制御、パフォーマンス、およびネットワーク使用率が提供されます。

- 結果として、以下のサービスをゾーン内で実行することが可能になっています。
- DHCPクライアント
- DHCPサーバー
- ルーティング・デーモン
- IPsec
- IPフィルタ
- IPマルチパス（IPMP）
- nddコマンド
- 設定および変更機能を持つifconfig（dladmおよびipadmの使用を推奨）
- Oracle Solaris 11上のOracle Solaris 10ゾーン

  “Oracle Solaris 11上のOracle Solaris 10ゾーンの作成およびインストール”の項を参照してください。

ISVにある程度の影響を与える機能

- 物理から仮想へ（P2V）の移行

  P2Vは、Oracle Solaris 10の既存の物理システム・イメージをOracle Solaris 11環境に移行できる機能です。インストールするシステム・イメージは、ホストOSリリースよりも新しいものであってはなりません。

- 仮想から仮想へ（V2V）の機能

  V2Vは、あるマシンにインストールされている非大域ゾーンを、zoneadm attach/detachサブコマンドを使用して、別のマシンに移行できる機能です。

- 接続時更新

  Oracle Solarisゾーンが新しいシステムに接続されるときに、そのゾーンがソース・システムにおいて依存しているものよりも新しい依存パッケージ/パッチ（より高いリビジョン番号）だけが新しいシステムに含まれている場合、新しいシステム上の高いリビジョンのパッケージに適合するようにゾーンが更新される機能です。言い換えると、接続されるゾーンが、新しいシステムのパッケージ・レベルにアップグレードされます。この更新は、次のように、zoneadm attachコマンドに-uまたは-Uフラグを付けることによって実行されます。

  ```
  # zoneadm -z myzone attach –u
  ```

- ゾーンの移動およびクローニング
  - 移動 - zonepathプロパティを変更して、ゾーンの場所を変更できます。
  - クローニング - ZFSクローニングおよびスナップショット機能を利用して、ゾーンを迅速にコピーおよび提供できます。
- 排他的IPスタック

  ゾーンが固有のIPスタックを持つためのオプションを提供します。これにより、ゾーンのIPルーティング、ARP、IPsec、IPフィルタ、およびその他の構成が、他のゾーンの構成と完全に分離されます。

顧客が個別の（V）LANを個別のゾーンに接続する場合には、ネットワーク・セキュリティの問題を発生させずに分離できます。ゾーンは、固有のIPアドレスを変更できます。以前は、排他的IPスタックに専用物理ポートが必要でしたが、Oracle Solaris 11では、VNICの実装により、この要件が排除されました。

- ゾーンの特権

  ゾーンで実行されるすべてのプロセスに制限が適用される特権セットを指定できる機能です。この特権は双方向で機能し、ゾーン内でDTraceを実行することを許可したり、ゾーンがraw ICMPパケットを送信することを禁止したりすることができます。

- ゾーンの委任管理

  ゾーンの管理がロール・ベースのアクセス管理（RBAC）と統合されます。この新しい機能により、管理者は、個別のユーザーまたはロールがゾーン内で管理タスクを実行できるようにゾーンを構成できます。

ISVにほとんどまたはまったく影響を与えない機能

- ゾーンの名前の変更
- ゾーンのデフォルト・ルート

  共有IPスタックを使用するゾーンのインタフェースのデフォルト・ルートを、新しいデフォルト・ルーター・プロパティのdefrouterを使用して、オプションで指定できます。

- ゾーン・リソース管理の機能拡張

  ゾーンは構成やインストールが非常に簡単ですが、多くのユーザーが、ゾーンの構成にともなう適切なリソース管理の設定に困難を感じています。Oracle Solaris 11では、ゾーン（リソース管理をともなうゾーン）の構成手順を単純で緊密に統合されたものにするために、いくつかの機能拡張が実施されています。

- ゾーン起動のための新しい引数

  ゾーンの起動時に標準の起動引数を使用できます。

- InfiniBandのサポート
- ゾーン内でのプロセッサ・セットの有効化

  プロセスがプロセッサ・セットにプロセスをバインドすることを可能にする新しい特権として`PRIV_SYS_RES_BIND`が作成されました。この特権をゾーンに割り当てることができます（デフォルトでは割り当てられません）。この新しい特権は`PRIV_SYS_RES_CONFIG`のサブセットであるため、`PRIV_SYS_RES_CONFIG`を持っているだけで、プロセスはプロセッサ・セットにプロセスをバインドすることができます。

- ゾーンの並行パッチ適用

  同じシステム上の複数のゾーンへの並行パッチ適用を可能にすることにより、パッチ適用ツールのパフォーマンスを向上させます。この機能は、Oracle Solaris 11上のOracle Solaris 10ゾーンにのみ適用されます。

Oracle Solaris 11ブランド・ゾーン（BrandZ）上のOracle Solaris 10ゾーンは、一連の代替の実行時動作を含むゾーンを作成するためのフレームワークを提供します。ブランドは、さまざまなオペレーティング環境を指す場合があります。たとえば、今回のような場合、非大域ゾーンは、Oracle Solaris 10オペレーティング・シ

ステムをエミュレートします。

ブランド・ゾーンは、これを、非ネイティブ・オペレーティング・システムのシステム・コール（syscall）をエミュレートすることによって実現します。syscallは、ユーザー環境とカーネルの間の単一のインタフェースを構成します。このため、ブランド・ゾーンがsyscallを、それらが特定のOS（Oracle Solaris 10など）のシステム・コールと同じ副作用を持つようにエミュレートする場合、ゾーン内で実行されるプロセスは、ターゲットOS上で実行されているかのように動作します。ブランド・ゾーンは、ゾーンのターゲットOSをエミュレートすることを可能にするサポート・ライブラリ、サポート・フック、および補助データファイルの集合体です。ブランドには、エミュレートするsyscallを持つオペレーティング・システムにちなんだ名前が付けられます。たとえば、Oracle Solaris 10上に存在するOracle Solaris 8ブランドとOracle Solaris 9ブランドは、それぞれ、Oracle Solaris 10システムがOracle Solaris 8ユーザー環境とOracle Solaris 9ユーザー環境をホストすることを可能にします。ネイティブのOracle Solarisユーザー環境をホストするゾーン（つまり、syscallをエミュレートしないゾーン）は、非大域ゾーンです。

以前のOracle Solaris 10ブランドについて

Oracle Solaris 10非ネイティブ・ゾーン（Oracle Solaris 10（5）のマニュアル・ページを参照）は、Oracle Solaris 10オペレーティング・システムを実行するSPARCおよびx86マシン上のOracle Solaris 10アプリケーションに対する完全な実行時環境です。このブランドは、Oracle Solaris 11でサポートされるプラットフォームとして定義されているすべてのsun4v、sun4u、およびx86アーキテクチャ・マシンでサポートされます。このブランドは、32ビットと64ビットのOracle Solaris 10アプリケーションの実行をサポートします。“ブランド”という用語は、今後、Oracle Solaris 11上のOracle Solaris 10ゾーンを指すためには使用されません。

SVR4（System V Release 4）パッケージ・メタデータは、ゾーン内部で利用でき、パッケージおよびパッチ・コマンドが正常に機能します。ゾーンが完全ルート・ゾーンであるため、パッケージまたはパッチのカーネル・コンポーネントが使用されなくても、すべてのパッケージまたはパッチ操作が成功します。

Oracle Solaris 10ネイティブ・ゾーンで使用されるSVR4パッケージについては、第25章“ゾーンがインストールされているSolarisシステムでのパッケージとパッチについて（概要）”および第26章“ゾーンがインストールされているSolarisシステムでのパッケージとパッチの追加および削除（手順）”[5]を参照してください。これは、Oracle Solaris 10バージョンのガイドです。Oracle Solaris 11上のOracle Solaris 10ゾーンは、完全ルート非大域ゾーン・モデルをサポートします。必要なすべてのOracle Solaris 10ソフトウェアとすべての追加パッケージは、ゾーンのプライベート・ファイル・システムにインストールされます。

Oracle Solaris 11上のOracle Solaris 10ゾーンの作成およびインストール

Oracle Solaris 11ゾーン上のOracle Solaris 10非ネイティブ・ゾーンの作成およびインストール方法について詳しくは、『System Administration Guide: Virtualization Using the Oracle Solaris 11 operating system』を参照してください。ここでは、プロセスについて簡単に説明するとともに、コマンドの例を示します。ブランドには、Oracle Solaris 10システム・イメージを非大域ゾーンにインストールするために必要なツールが含まれています。Oracle Solaris 10ゾーンは、直接Oracle Solaris 10メディアからOracle Solaris 11にインストールできません。既存のOracle Solarisシステムをターゲット・システム上の非大域ゾーンに移行するには、物理から仮想への移行（P2V）機能を使用します。ブランドは、Oracle Solaris 10ネイティブ・ゾーンをOracle Solaris 10 08/2010で導入されたOracle Solaris 10非大域ゾーンに移行するために使用されるツールもサポートします。使用中のシステムのOracle Solaris 11でOracle Solaris 10ゾーンを使用するには、system/zones/brand/s10パッケージを

[5]『Oracle Solarisのシステム管理（Oracle Solarisコンテナ：資源管理とOracle Solarisゾーン』（部品番号：821-1460）

ターゲット（Oracle Solaris 11）システムにインストールする必要があります。

以下の項で、Oracle Solaris 11上にOracle Solaris 10ゾーンを作成する次の3つの使用例について説明します。

- 非大域ゾーンが構成されていない既存のOracle Solaris 10システムまたはフラッシュ・アーカイブに基づくOracle Solaris 11上のOracle Solaris 10ゾーンの作成（P2V）
- 単一のOracle Solaris 10ネイティブ・ゾーンに基づくOracle Solaris 11上のOracle Solaris 10ゾーンの作成（V2V）
- Oracle Solaris 10ネイティブ・ゾーンを持つシステムに基づくOracle Solaris 11上のOracle Solaris 10ゾーンの作成（P2VおよびV2V）

既存のOracle Solaris 10システムまたはOracle Solaris 10フラッシュ・アーカイブ・イメージに基づくOracle Solaris 11上のOracle Solaris 10ゾーンの作成およびインストール

- ソース・システムにアクセスして、次の情報を収集します。これらの情報は、Oracle Solaris 11ターゲット・システム上でOracle Solaris 10ゾーンを作成する際に必要になります。
  - ホストID（hostid(1)）
  - RPCドメイン名（domainname(1M)）
  - ルート・パスワード
  - システムで利用されているネットワーク
  - マウントされているファイル・システム
  - 既存のシステムで使用されているローカル・ディスク・ストレージの総容量
  - /etc/systemの内容
- ソース・システムのOracle Solarisフラッシュ・アーカイブ・イメージを作成します。
- Oracle Solarisフラッシュ・アーカイブ・イメージをターゲット・システムに転送します。
- ターゲット・システムで、zonecfg(1M)コマンドによってOracle Solaris 10ゾーンを使用してゾーンを作成し、ソース・システムのフラッシュ・アーカイブ・イメージをインストール・イメージとして使用します。

```
target# zonecfg -z my-zone
my-zone: No such zone configured
Use ’create’ to begin configuring a new zone.
zonecfg:my-zone> create -t SUNWsolaris10
zonecfg:my-zone> set zonepath=/zones/myzone
…
```

- zoneadm installコマンドを使用してゾーンをインストールします。

```
zoneadm -z my-zone install -a <path_to_the_Oracle Solaris 10 image flar> -u
```

単一のOracle Solaris 10ネイティブ・ゾーンに基づくOracle Solaris 11上のOracle Solaris 10ゾーンの作成

- Oracle Solaris 10システム上の既存のゾーンをOracle Solaris 11上のOracle Solaris 10ゾーンに移行するには、仮想から仮想への移行（V2V）プロセスを使用します。V2Vプロセスについては、以下の手順で大まかに説明します。Oracle Solaris 10非大域ゾーンをOracle Solaris 11上のOracle Solaris 10ゾーンに移行するためのV2Vプロセスは、P2Vプロセスと同じアーカイブ形式をサポートします。このプロセスでは、システム間でゾーンを移行するための既存のインタフェースであるzoneadm attachサブコマンドを使用します。
- ソース・システム上のゾーン構成情報を保存します（zoneadm -z <zone-name> info）。
- ソース・ゾーンを停止します。
- ゾーンのcpioアーカイブを作成します。
- アーカイブをターゲットOracle Solaris 11システムに転送します。
- ターゲットOracle Solaris 11システムで、zonecfgコマンドによってOracle Solaris 10ゾーンを使用して、ゾーンを再作成します。
- ソース・システムで作成したアーカイブからゾーンに接続します。ターゲット・システム上の/zonesディレクトリに転送されたアーカイブを使用して、次のコマンドを実行してください。

```
target# zoneadm -z my-zone attach -a <path_to_the zone_archive>
```

ゾーンのインストールが正常に完了すると、ゾーンを起動できます。

Oracle Solaris 10ネイティブ・ゾーンを持つシステムに基づくOracle Solaris 11上のOracle Solaris 10ゾーンの作成（P2VおよびV2V）

Oracle Solarisゾーンでアプリケーションを実行できなくなる可能性のある互換性の問題

以下の項では、非大域ゾーンにアプリケーションをデプロイして実行する際に発生する可能性のある互換性の問題について説明します。これらは、特に明記しない限り、すべてのOracle Solarisゾーン・テクノロジー（Oracle Solaris 10およびOracle Solaris 11非大域ゾーン、Oracle Solaris 11のOracle Solaris 11上のOracle Solaris 10ゾーンなど）に共通の問題です。これらの制限について詳しくは、『Oracle Solaris Zones: Resource Management and Oracle Solaris Zones Developer's Guide』の第7章を参照してください。

各非大域ゾーンには、その周囲にセキュリティ境界があります。このセキュリティ境界は、次のものだけで維持されます。

- 削減されたプロセス特権
- リソースとサービスの仮想化
- ネットワークを介したゾーン間の通信

削減されたプロセス特権

Oracle Solaris 10 OSにおいて、プロセス特権管理が導入されました。これにより、Oracle Solarisプロセス・モデルが特権セットによって拡張されるとともに、これらの特権を使用して、システム・リソースおよびカーネル・サービスのプロセス・アクセスが制御されるようになりました。プロセスは、次のいずれかの状態になります。

- 特権を認識する（PA）：有効なUIDを完全に無視します。
- 特権を認識しない（NPA）：従来のプロセスとほとんど同じように動作します。

プロセスがPAの場合、カーネルは、システム操作に関してプロセス特権だけをチェックします。プロセスがNPAの場合、カーネルは、UIDとプロセス特権の両方をチェックします。プロセスは、setpflags(2)を使用してNPAになることを試みることができます。

各特権セットには、0以上の特権が含まれます。各プロセスは、4つの特権セットを持ちます。特権セットの1つである実効（E）特権セットによって、プロセスが特定の特権を使用できるかどうかが決定されます。

4つの特権セットは、次のとおりです。

- E（実効セット）：現在有効な特権セット
- P（許可セット）：プロセスのための最大特権セット
- I（継承可能セット）：exec(2)で継承される特権セット
- L（制限セット）：プロセスおよびその子孫が取得できる特権の上限

Oracle Solaris 11では50の特権が定義されており、privileges(5)によってこれらの特権とその定義のリストが表示されます。ppriv(1)を使用すると、プロセスの特権セットを調べたり、変更したりすることができます。

非大域ゾーンで動作するすべてのプロセスは、削減された特権を持ちます。つまり、非大域ゾーン内のすべてのプロセスは、プロセスの作成時に割り当てられる特権セットによって制限されます。非大域ゾーンにおけるプロセスの削減された特権のために、特定のシステム・コールでエラーが返されることがあります。ほとんどの場合、特権を持たないプロセスについてはEPERMが返されます。cpc_cpuまたはnet_rawaccessをチェックする一部のシステム・コールでは、EACCESSが返されます。

非大域ゾーン内の特権を持つアプリケーションだけが、削減されたプロセス特権の影響を受けることに注意してください。特権を持たない従来のアプリケーションには、5つの基本特権だけが付与されます。これらの5つの基本特権は、非大域ゾーンに対して利用できます。

ライブラリ

次のリストのライブラリによって提供されるAPIは、ゾーンではサポートされません。共有オブジェクトはゾーンの/usr/libディレクトリ内に存在するため、これらのライブラリへの参照がコードに含まれている場合、リンク時エラーは発生しません。

- libcfgadm(3LIB) – 構成管理ライブラリ
- libdevinfo(3LIB) – デバイス情報ライブラリ
- libpool(3LIB) - プール構成操作ライブラリ
- libkvm(3LIB) - カーネル仮想メモリ・アクセス・ライブラリ
- libtnfctl(3LIB) - TNFプローブ制御ライブラリ
- libsysevent(3LIB) - システム・イベント・インタフェース・ライブラリ

ネットワーク

非大域ゾーンが排他的IPスタック・プロパティを使用せずに構成されている場合は、ネットワークに関する以下の制限がゾーンに適用されます。

- IPQoSとIPsecの構成（ipqosconf(1M)およびipsecconf(1M)）は、大域ゾーンでのみ実行できます。その構成にゾーンのIPアドレスを指定することにより、ゾーン固有の構成を作成できます。
- トランスポート層より下位の層へのrawアクセス（たとえば、リンク層へのIP、ARP、およびDLPI）は、非大域ゾーンでは許可されません。したがって、DLPIを使用してリンク層（NICデバイス・ドライバ）と直接通信するとエラーになります。snoop(1M)は、DLPIを使用してインタフェース・ドライバに直接アクセスするので、非大域ゾーンでは動作しません。
- 次のネットワーク機能は、大域ゾーンの管理者だけが構成できるシステム全体の機能として残ります。
  - ルーティング
  - IPマルチパス（IPMP）
  - モバイルIP
  - DHCPクライアント
  - Network Cache and Accelerator（NCA）
  - /etc/systemおよびndd(1M)を使用したネットワーク・チューニング
  - IPフィルタ

デバイス

- 一般的に、デバイスはシステム内の共有リソースです。したがって、ゾーンでデバイスを使用可能にするには、ゾーン分離の原則を守るために何らかの制限が必要です。/devディレクトリは、以下の制約のために厳しく制限されます。
- システム・データを表示するデバイスは、大域ゾーンでのみ使用可能です。このようなデバイスの例として、kmem(7D)、ksyms(7D)、kmdb(7D)、trapstat(1M)、lockstat(7D)などがあります。
- /dev名前空間は、/devices内の物理パスへのシンボリック・リンク（論理パス）から構成されます。/devices名前空間は大域ゾーン内でのみ使用可能で、ドライバによって作成された接続済みデバイス・インスタンスの現在の状態を表します。論理パスの/devだけは、非大域ゾーンで見ることができます。
- 大域ゾーンの管理者は、zonecfg(1M)を使用して特定のゾーンに存在するデバイスを指定します。非大域ゾーン内の/devエントリの数は、大域ゾーン内の/devエントリの数よりも大幅に少なくなります。大域ゾーンの管理者は、zonecfgのadd deviceサブコマンドを使用して、ゾーン内に追加のデバイスを含めることができます。
- ゾーンの管理者は、デバイスのアクセス権を変更することはできますが、新しいエントリを作成することはできません。
- デバイスの数は、システム全体のプロパティです。特定のデバイス番号にマッピングする特殊ファイルを作成するシステム・コール（mknod(2)など）は、エラーを返します。
- ハードウェアを構成したり、/devエントリを変更したりするユーティリティは、ゾーンでは機能しません。これらのユーティリティは、次のようなものです。
  - add_drv(1M)/rem_drv(1M)
  - modload(1M)/modunload(1M)
  - autopush(1M)
  - cfgadm(1M)
  - devfsadm(1M)、drvconfig(1M)、disks(1M)、tapes(1M)、ports(1M)、およびdevlinks(1M)

Oracle Solaris 10ゾーンとOracle Solaris 11上のOracle Solaris 10ゾーンの相違

ネットワーク

以下のリストで、Oracle Solaris 11上のOracle Solaris 10ゾーンではサポートされないか異なる点のあるOracle Solaris 10のネットワーク機能を示します。

- モバイルIPは、サポートされません。この機能は、Oracle Solaris 11システムでは使用できません。
- atun STREAMSモジュールを使用した自動トンネルは、サポートされません。Oracle Solaris 11上のOracle Solaris 10ゾーンでは、tcp、udp、またはicmpソケットが開いている場合、autopush構成が無視されます。デフォルトでは、これらのソケットは、STREAMSデバイスではないモジュールにマップされます。autopushを使用するには、soconfig(1M)およびsock2path(4)ユーティリティを使用して、これらのソケットを明示的にSTREAMSベースのデバイスにマップします。
  Oracle Solaris 11上のOracle Solaris 10ゾーンでは、パッチをインストールしないと、/dev/netのリンクが

データ・リンク・プロバイダ・インタフェース（DLPI）ライブラリでサポートされません（libdlpi(3LIB)のマニュアル・ページを参照）。パッチが適用されたlibdlpiまたはlibpcapバージョン1.0.0以降のライブラリを使用しないアプリケーションは、/dev/netのリンクにアクセスできません。次のチューニング可能なnddパラメータは、Oracle Solaris 11上のOracle Solaris 10ゾーンではサポートされません。

- ip_squeue_fanout
- ip_soft_rings_cnt
- ip_ire_pathmtu_interval
- tcp_mdt_max_pbufs

- Oracle Solaris 10ゾーンのIPネットワーク・マルチパス（IPMP）は、Oracle Solaris 11オペレーティング・システムに基づいているため、Oracle Solaris 10オペレーティング・システムのコマンド出力と比較すると、ifconfig(1M)コマンドの出力に違いがあります。ただし、ifconfigコマンドおよびIPMPの文書化された機能は、変更されていません。そのため、文書化されているインタフェースを使用するOracle Solaris 10アプリケーションは、変更なしで、Oracle Solaris 11 Express上のOracle Solaris 10ゾーンで引き続き動作します。次の例は、データ・アドレス198.162.1.3のIPMPグループipmp0と、テスト・アドレス198.162.1.1および198.162.1.2のベースとなるインタフェースe1000g1およびe1000g2に関する、Oracle Solaris 11上のOracle Solaris 10ゾーンでのifconfigコマンドの出力をそれぞれ示しています。

```
% ifconfig -a
e1000g1:flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,
DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 8 inet 198.162.1.1
netmask ffffff00 broadcast 198.162.1.255 ether
0:11:22:45:40:a0
e1000g2:flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,
DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 9 inet 198.162.1.2
netmask ffffff00 broadcast 198.162.1.255
ether 0:11:22:45:40:a1
ipmp0: flags=8011000803<UP,BROADCAST,MULTICAST,IPv4,
FAILED,IPMP> mtu 68 index 10 inet 198.162.1.3
netmask ffffff00 broadcast 198.162.1.255
groupname ipmp0
```

- Oracle Solaris 10システムで生成される表示とは異なり、Oracle Solaris 11上のOracle Solaris 10ゾーンでのifconfigコマンドでは、IPアドレスへのベースとなるインタフェースのバインディングは表示されません。この情報は、arpコマンドをanオプションとともに使用することにより取得できます。
- インタフェースがIPv6に接続され、アドレスの自動構成に成功すると、そのインタフェースに固有のグローバル・アドレスが与えられます。Oracle Solaris 10では、IPMPグループの各物理インタフェースは固有のグローバル・アドレスを保有し、IPMPグループはインタフェースと同じ数のグローバル・アドレスを保有します。Oracle Solaris 11上のOracle Solaris 10ゾーンでは、固有のグローバル・アドレスを保有するのはIPMPインタフェースのみです。ベースとなるインタフェースは、固有のグローバル・アドレスを保有しません。
- Oracle Solaris 10システムで生成される表示とは異なり、Oracle Solaris 11上のOracle Solaris 10ゾーンでの

ifconfigコマンドでは、IPアドレスへのベースとなるインタフェースのバインディングは表示されません。この情報は、arpコマンドをanオプションとともに使用することにより取得できます。

ゾーンのルート・ファイル・システム

Oracle Solaris 10では、ゾーンを作成する際、ルート・ファイル・システムを作成するために疎ルートと完全ルートの2つのモデルから選択できます。完全ルート・モデル（zonecfg(1M)のcreate -bサブコマンドによって指定）では、すべての必要なOracle Solarisソフトウェア・パッケージと選択したオプションのOracle Solarisソフトウェア・パッケージをゾーンのプライベート・ファイル・システムにインストールすることにより、最大限のカスタマイズ性が実現されます。このモデルの長所には、ゾーンの管理者がファイル・システムのレイアウトをカスタマイズ（たとえば、/usr/localを作成）できることや、任意のバンドルされていないパッケージまたはサード・パーティ製パッケージを追加できることなどがあります。このモデルの短所には、仮想メモリ・システムによる実行可能ファイルや共有ライブラリのテキスト・セグメントの共有が失われ、そのように構成された非大域ゾーンごとのディスクの占有サイズがはるかに大きくなり（約2GBの容量が余分に必要）、ゾーンのインストールやパッチのインストール（パッチ・ファイルをゾーンのルート・ファイル・システムにコピーしなければならないため）にかかる時間も長くなることなどがあります。疎ルート・モデル（デフォルト）では、ルート・パッケージ（pkginfo(4)のSUNW_PKGTYPEパラメータがrootに設定されているパッケージ）のサブセットだけがインストールされ、その他のファイルへのアクセスには読取り専用のループバック・ファイル・システムが使用されることにより、オブジェクトの共有が最適化されます。このモデルでは、デフォルトで、/lib、/platform、/sbin、および/usrディレクトリが読取り専用のループバック・ファイル・システムとしてマウントされます。このモデルの長所は、実行可能ファイルと共有ライブラリを効率的に共有できるためにパフォーマンスが高いこと、ゾーン自身のディスク占有サイズが大幅に小さいこと、およびパッチ適用にかかる時間が短いことです。疎ルート・モデルでは、必要なゾーン自体のファイル・システム空間は、約100MBだけです。疎ルート・モデルと完全ルート・モデルに加えて、Oracle Solaris 10では、管理者は、さまざまなファイル・システム（UFS、ZFS、およびNFS）上にゾーン・ルートを作成することを選択できます。Oracle Solaris 11上のOracle Solaris 10ゾーン機能は、完全ルート・ゾーン・モデルをサポートします。必要なすべてのOracle Solaris 10ソフトウェアとすべての追加パッケージは、ゾーンのプライベート・ファイル・システムにインストールされます。非大域ゾーンは、固有のZFSデータセット上に存在する必要があります。ZFSだけがサポートされます。ZFSデータセットは、ゾーンのインストールまたは接続時に自動的に作成されます。ZFSデータセットを作成できない場合、ゾーンはインストールされません。

その他の制限

- mdbユーティリティは、Oracle Solaris 11上のOracle Solaris 10ゾーン内で実行されるプロセスを調べるために大域ゾーンで使用する場合、完全には機能しません。一部のシンボルが解決されません。このユーティリティは、Oracle Solaris 11上のOracle Solaris 10ゾーン内から使用できます。
- スナップショットを受け取るためのzfs recvコマンドは、Oracle Solaris 11上のOracle Solaris 10ゾーン内では使用できません。
- /dev/soundデバイスは、Oracle Solaris 11上のOracle Solaris 10ゾーン内に構成できません。

コマンドラインの変更点

この項では、Oracle Solaris 10からOracle Solaris 11に移行した場合のコマンドラインの変更点とエンドユーザーの使用環境の変更点について説明します。多くのISVアプリケーションは、シェル・スクリプトから起動され、またPATHおよびその他の環境変数の設定を必要とします。アプリケーションが起動される、コマンドで利用可能な環境に関する前提は、これまでのOracle Solarisリリースの旧来の使用方法に基づいています。Oracle Solaris 11のコマンドライン環境には大幅な変更が加えられており、この項では、これらの変更点について説明するとともに、アプリケーションをOracle Solaris 10からOracle Solaris 11に移行する場合にこれらの変更が関係するかどうか、どのようなときに関係するかを示します。また、この項では、アプリケーションやマニュアルをOracle Solaris 11用に更新する場合に変更を最小限に抑えるために使用できるベスト・プラクティスについても詳しく説明します。

Oracle Solaris 11は、ユーザー、開発者、およびシステム管理者用のOracle SolarisおよびGNUのユーザーレベル・コマンド機能が混在する最新のシステム環境を備えています。この組合せは、既存の顧客およびアプリケーションをサポートするとともに、GNU環境を使い慣れているOracle Linuxユーザーやその他ユーザーおよび開発者に使いやすい環境を提供するために必要です。

パス	説明
/etc	従来のシンボリック・リンクは削除されました。
/usr/bin	コマンドライン・ユーティリティのデフォルト・パスです。
/usr/old	削除されました。
/usr/sunos/bin	旧来の（Sun OS 4）コマンドライン・ユーティリティのパスです。
/usr/gnu/bin	GNUコマンドライン・ユーティリティのパスです。
/usr/xpg4/bin	複数のバージョンのコマンドライン・ユーティリティ（UNIX98準拠のために必要）のパスです。
/usr/xpg7/bin	（提案）複数のバージョンのコマンドライン・ユーティリティ（UNIX08準拠のために必要）のパスです。
/sbin	/usr/binへのシンボリック・リンクです。
/usr/sbin	システム管理ユーティリティです。
/usr/ucb	BSD互換性ユーティリティ（オプション）です。
/usr/5bin	推奨されないディレクトリです。/usr/binへのシンボリック・リンクです。
/usr/ccs	推奨されないディレクトリです。/usrへのシンボリック・リンクです。C準拠コマンドおよびシステム・ユーティリティです。
usr/local/bin	バンドルされないGNUユーティリティの共通インストール・ディレクトリです。
/etc	従来のシンボリック・リンクは削除されました。

パス	説明
/usr/old	段階的に廃止されているユーティリティです（現在は空です）。
/usr/has/bin	GNUの同等のユーティリティによって置き換えられた/usr/binの旧来のユーティリティです。
/usr/sfw/bin	SFW統合によるGNUユーティリティです。
/opt/sfw/bin	Oracle Solaris 10における、SFW統合によるGNUユーティリティ用のオプションのディレクトリです。Oracle Solaris 11では使用されていません。

デフォルト・パス

以下の項では、すべての変数をインストール時のデフォルト設定にしているOracle Solaris 11の新規ユーザー向けに、Oracle Solaris 10システムの同様の状況からの変更点について説明します。

Oracle Solaris 11ユーザーのデフォルトのPATHは、そのユーザーが最初に作成された方法によって異なる場合があります。現時点で発生する可能性の高い状況は、次の3つです。

- ユーザーがuseraddまたはSMCを使用して作成されており、標準のPATH=/usr/binが使用されている。
- ユーザーがインストール時に作成されており、拡張されたデフォルトのPATH=/usr/gnu/bin:/usr/binを使用している。
- ユーザーがインストール時に作成され、ルート・ロールが割り当てられており、PATH=/usr/gnu/bin:/usr/bin:/usr/sbin:/sbinを使用している。

既存のユーザーのPATH変数は、Oracle Solaris 10の場合と同様に、そのユーザーのホーム・ディレクトリにある.profileおよび.*rcファイルを使用してカスタマイズすることができます。新規ユーザーのデフォルトのPATHも変更できます。

Oracle Solaris 11でインストール時に作成されるユーザー（ルート・ロールの割当てのあるなしに関わらず）については、現時点では、そのPATHの先頭にパス`/usr/gnu/bin`が付加されます。これにより、通常は/usr/binまたは/usr/sbinに保存される一般的なユーティリティ（tar、chroot、grepなど）のGNUバージョンがユーザーに提供されます。これらのユーティリティのOracle Solarisバージョンも引き続き使用できますが、フル・パス（/usr/bin/tar、/usr/sbin/chrootなど）を使用して実行する必要があります。このため、ユーザーがこれらのツールにOracle Solarisバージョンの動作を期待している場合には問題がある可能性があります。

`/usr/gnu/bin`には、Oracle Solarisに類似のものがないいくつかのコマンド（toe、ncurses5-config）も含まれています。

より一般的ツールの一部（またはユーザーがGNUバージョンとOracle Solarisバージョンの両方にアクセスする必要があるツール）については、GNUバージョンが/usr/binにある場合がありますが、先頭に‘g’が付いています。たとえば、gtar、gmake、およびggrepは、GNUバージョンのtar、make、およびgrepです。これらには、/usr/gnu/bin内に、適切な名前のシンボリック・リンクがあります。

ユーザーがこれらのユーティリティのOracle Solarisバージョンをデフォルトで使用したい場合は、そのユーザーのPATHから/usr/gnu/binを削除するだけで、そのように変更できます。この場合、先頭にgの付いたコマンドによって、一部のGNUバージョンのユーティリティにアクセスすることもできます。

シェルおよびシェル環境の変更点

次の表に、Oracle Solaris 11でユーザーが使用できるシェルとそのバージョン番号およびパスを示します。

シェル	バージョン	パス	注
Bourneシェル		/usr/has/bin/sh /usr/xpg4/bin/sh	/usr/bin/shは、/usr/bin/bashにリンクされています。
Kornシェル	93	/usr/bin/ksh /usr/bin/ksh93 /bin/sh /usr/bin/sh	kshとksh93は、ハードリンクされています。useraddによって追加されるユーザーのデフォルトです。
Bourne Againシェル	4.0.28	/usr/bin/bash /usr/gnu/bin/sh /bin/bash	インストール時に追加されるユーザーのデフォルトです。
Zsh	'4.3.10'	/usr/bin/zsh /usr/sfw/bin/zsh	
Tcsh	6.17.00（Astron）	/usr/bin/tcsh	
Csh	?	/usr/bin/csh	

さらに、‘プロファイル’シェル（pfksh、pfsh、およびpfcsh）とともに、通常‘制限された’バージョンのkshおよびbash（rksh、rbash）を使用できます。

bourneシェルがデフォルトのPATHで使用できなくなっていることと、/usr/bin/shを実行すると新しいbashシェルが開くことに注意してください。また、Kornシェルはバージョン93に更新されており、古いバージョンは使用できません。

インストール時に作成されるユーザーのデフォルト・シェルは/usr/bin/bashであり、useraddを使用して作成されるユーザーのデフォルト・シェルはそのユーザーの/usr/bin/shです。

注：/bin/shは、ユーザーが期待する/bin/bashではなく、ksh93の1つのバージョンにリンクされます。

/usr/ucbのディレクトリの変更点

CDによるライブ・インストールの場合、パス/usr/ucbは、デフォルトでは存在しません。このパスは、リポジトリから‘compatibility/ucb’パッケージをインストールすることにより、多くの標準/usr/ucbツールとともに追加できます。ただし、従来のプロッタのサポートが削除されていることと、これにより以下のファイルが削除されていることに注意してください（PSARC/2009/540）。

- /usr/ucb/aedplot
- /usr/ucb/atoplot
- /usr/ucb/bgplot
- /usr/ucb/crtplot
- /usr/ucb/dumbplot
- /usr/ucb/gigiplot
- /usr/ucb/hp7221plot
- /usr/ucb/hpplot
- /usr/ucb/implot
- /usr/ucb/plot
- /usr/ucb/vplot
- /usr/ucb/t300
- /usr/ucb/t300s
- /usr/ucb/t4013
- /usr/ucb/t450
- /usr/ucb/tek

また、/usr/ucb/ucblinksコマンドによるSunOS 4互換デバイス名の生成のサポートが廃止されていることにも注意してください（PSARC/2009/346）。

/usr/ucbの最大の変更点は、コンパイラおよびリンカーのコマンド・ラッパーとCヘッダーが削除されたことです。Oracle Solaris 11以降は、次のラッパー・スクリプトを使用できません。

- /usr/ucb/ld
- /usr/ucb/lint
- /usr/ucb/cc

また、/usr/ucbincludeディレクトリとその内容が完全に削除されました。/usr/usrlibは、下位互換性のために、保持されています（compatibility/ucbパッケージの一部として）。

これらの変更を除き、Oracle Solaris 10の/usr/ucbで使用できるすべてのコマンドは、compatibility/ucbパッケージにより、Oracle Solaris 11でも引き続き使用できます。

Oracle Solaris 11のパッケージ化メカニズム

この項では、Oracle Solaris 11のIPSの概要について、特にISVに関連する点を中心に説明します。また、ISVがIPSによって迅速化するために役立つ他の有益な資料へのリンクを示すとともに、IPSの使用を開始しようとしているISVへの助言も記載します。パッケージの作成については、別の資料でより詳しく説明する予定です。

以下の点にも重点を置いて説明します。

- 保持される、既存のSVR4パッケージとの互換性
- Oracle Solaris 11のSVR4で削除されたもの
- SVR4パッケージからIPSにアプリケーションを移行させるISVおよびIHVに役立つソリューションおよびベスト・プラクティスの提供

最後に、この項では、IPSパッケージを顧客が利用できるようにするための“ノウハウ”についても説明します。これには、実行可能なインストーラを作成するために使用できるリポジトリおよびオンディスク形式の設定も含まれます。

IPSとは

Image Packaging System（IPS）は、ソフトウェア・コンテンツの既知の構成間での、効率的で、可視的で、制御可能な移行を実現できるように設計された、優れた移植性を持つ、ネットワーク中心のパッケージ化および配布システムです。最初にOpenSolarisの一部として開発されたIPSは、Oracle Solaris 11においてSVR4パッケージに代えて使用されていますが、既存のパッケージとのある程度の互換性を維持するために、SVR4の一部の側面が残されています。

IPSは、Oracle Solaris 11より古いバージョンのOracle Solarisで使用されていた従来のパッケージ化システムの根本的な変革であり、一見すると少し圧倒されるような印象を受けるかもしれません。この新しいシステムにより、システム管理者のパッケージ化およびパッチ適用の作業が簡素化されるため、すべてのISVができるだけ早くこのシステムに移行することを強く推奨します。しかし、オラクルでは、既存のもので開始、運用し、この変更には余裕を持って対処したいという希望があることも認識しています。そのような希望に応じるために、Oracle Solaris 11では、SVR4パッケージのサポートを継続するとともに、これまでSVR4パッケージのインストール、管理、およびメンテナンスに使用されていたツールも引き続き提供します。いくつかの制限も存在しますが、これらについては「従来のパッケージ化」の項で説明します。

IPSについて詳しくは、オンライン・マニュアルを参照してください。[6]

[6] http://docs.oracle.com/cd/E37932_01/html/E36679/toc.html

IPS開発の重要な動機の1つは、SVR4のパッケージ化とパッチ適用の間の断絶でした。一連の任意のパッチを任意のパッケージの集合体に適用できるようにするという発想から、予想もしなかった、まったく新しいソフトウェア構成が生まれ、推測されるサポートをすべて実現できるようになりました。その結果、Oracle Solaris 11は、パッケージ化とパッチ適用に関する次のような基本方針に基づいて設計されました。

- パッケージ化を、個別のファイルベース・パッケージという考え方から、ネットワークまたはディスクベースのパッケージ“リポジトリ”という概念に移行させる。
- パッケージ操作は、より直感的で高精度のものになるとともに、ほとんどのシステムがネットワーク・アクセスを持つことを前提とする（ただし必須ではない）。
- パッチは存在しなくなり、ソフトウェアの変更はそのソフトウェアの新バージョンのパッケージに組み込まれ、システム管理者が適切と判断する場合にインストールやアップグレードに使用できる。

繰り返しになりますが、

Oracle Solaris 11 Express以降ではパッチ適用という概念は存在しません。

IPSパッケージ

IPSでは、すべてのパッケージ化データがパッケージに含まれており、パッケージが最小の構成単位です。パッチは存在せず、パッケージのバージョンと依存関係だけが存在します。インストール時スクリプトは、サポートされなくなりました。代わりに、IPSでは、ソフトウェアの自己アセンブリの概念が使用されます。これにより、環境に関する考慮によってパッケージの最終構成におけるいくつかの側面が決定されます。

このようにSVR4パッケージからIPSに移行するには、すべてのスクリプトを排除する必要があります。必要な場合には、同様のインストール時機能を実現する代替方法を使用できます。基本的な手法については、Constantin Gonzalezのブログの記事[7]を参照してください。

繰り返しになりますが、

各パッケージ（バージョン）は、一意の障害管理リソース識別子（FMRI）によって表されます。

パッケージは、アクションのリストによって構成されるマニフェストによって定義されます。IPSは、ファイル、ディレクトリ、依存関係、およびリンクのアクションを含む、限定された一連のアクションを定義します。

パッケージ間の依存関係は、マニフェストで宣言され、IPSによって管理されます。依存関係は、次のいずれかになります。

[7] http://constantin.glez.de/blog/2010/08/how-add-pre-post-scripts-ips-packages

依存関係名	説明
require	パッケージが、指定されたバージョン以上のバージョンで存在している必要があります。
optional	パッケージがインストールされている場合は、指定されたバージョン以上のバージョンである必要があります。
exclude	パッケージがインストールされている場合は、指定されたバージョンよりも古いバージョンである必要があります。
incorporate	パッケージがインストールされている場合は、指定された精度内の指定されたバージョンに制約されます。

パッケージは、ファイル・システム内のイメージと呼ばれる場所にインストールされます。フル・イメージでは、すべてのパッケージの依存関係がイメージ内で満たされます。一方、リンクされたイメージには、フル・イメージにインストールされるパッケージによって満たされる依存関係を持つパッケージが含まれています。リンクされたイメージの例は、デフォルトでOracleゾーンにインストールされる一連のパッケージです。ライブ・イメージは、起動したオペレーティング・システムを含むフル・イメージです。

パッケージのインストール

IPSを使用すると、パッケージが、パッケージ・リポジトリからイメージにインストールされます。パッケージ・リポジトリは、URIによって指定される、パッケージ・パブリッシャに関連付けられた場所です。パブリッシャは、複数のパッケージ・リポジトリを統合できます（ただし、一般的には、パブリッシャからリポジトリへの1対1のマッピングが存在します）。イメージは、複数のパブリッシャをパッケージのソースとして使用するように構成される場合もあります。

リポジトリURIでは、プロトコルとして、fileまたはhttpを指定できます。これにより、ネットワークベースのリポジトリまたはファイル・システムベースのリポジトリからのパッケージのインストールが可能になります。

前述したように、パッケージは、FMRIによって完全に識別されます。完全なFMRIは、パブリッシャ名、パッケージ名、および4つの部分から成るバージョン文字列によって構成されます。パッケージを指定する際、パブリッシャ名とバージョン文字列（一部またはすべて）が省略される場合があります。1つのイメージに同時に複数のバージョンのパッケージをインストールすることはできません。

IPSの中核には、pkg(5)と呼ばれるものが存在します。pkgはコマンドであり、(5)はpkgコマンドの仕様が記載されているマニュアル・ページのセクション（この場合は、セクション5の「標準、環境、マクロ」）を示します（‘man -s 5 pkg’によってこのマニュアル・ページが表示されます）。pkg(5)およびその他の少数のコマンドによって、パッケージの作成からリポジトリの構築にいたるIPSのすべての側面が管理されます。

pkgコマンドに加えて、システム管理者がpkgベースのほとんどの機能を実行できるGUIツールのパッケージ・マネージャも提供されています。

従来のパッケージ化

Oracle Solaris 11では、多数のIPSパッケージが、1つまたは複数の‘legacy’アクションを持ちます。legacyアクションには、そのパッケージに関連付けられた従来の（SVR4）パッケージ名（サポートされるアーキテクチャごと）が含まれます。たとえば、‘network/ssh’パッケージは、このパッケージに関連付けられたSUNWsshr、SUNWsshu、SUNWsshdu、およびSUNWsshdrという名前の従来のパッケージを持ちます。これらは、Oracle Solaris 10上の同等のパッケージの名前です。

Oracle Solaris 11には、‘package/SVR4’という名前のIPSパッケージが含まれています。このパッケージには、Oracle Solaris 10にある複数のバージョンのSVR4ツール（pkgadd、pkgrm、pkgchkなど）が含まれています。SVR4パッケージ用に、これらのツールは、Oracle Solaris 10の場合と同じ機能を引き続き提供します。legacyアクションを持つIPSパッケージの場合、これらのツールによって、それらのパッケージを認識できますが、管理することはできません。

‘package/SVR4’パッケージ・ツールを使用すると、従来のパッケージを、Oracle Solaris 10の場合と同じ方法でインストール、管理、およびメンテナンスすることができます。

前述したように、Oracle Solaris 11には、パッチ適用の概念がなく、Oracle Solaris 10でアプリケーションをデプロイするために使用されていたパッチ管理コマンドを利用できません。

パッチが適用されず、パッチそのものも存在しないため、showrevコマンドはありません。showrevは、Oracle Solaris 10において、インストールされているパッチのリストを得るためのコマンドであり、システムまたは特定のパッケージのパッチ・レベルを確認するためにスクリプト内で正規表現とともに頻繁に使用されていました。Oracle Solaris 11では、

```
$ pkg list [pkgname]
```

コマンドを使用して、インストールされている1つまたは複数のパッケージのバージョン番号を確認します。

SVR4パッケージの提供を継続する場合は、どのような更新を行う場合でも、それらのパッケージの更新を通して提供する必要があります。

IPSで使用するためのSVR4からのパッケージの変換

`pkg(5)`には、`pkgsend`コマンドが含まれています。このコマンドは、新しいパッケージまたはパッケージの新しいバージョンをリポジトリに公開するために使用されます。`pkgsend`はさまざまな方法で使用できますが、ここでは、このコマンドを使用して既存のSVR4パッケージからマニフェストを生成し、そのマニフェストを（必要なファイルとともに）IPSリポジトリに公開する方法について説明します。

既存のSVR4パッケージからのマニフェストの生成は簡単ですが、`pkgtrans`で作成する場合のようにパッケージ・データ・ストリーム上ではなく、SVR4パッケージ・ディレクトリ上でしか実行できません。マニフェストを生成するには、次のコマンドを実行します。

```
$ pkgsend generate MYpkg > Mypkg.mfst
```

ここで、`MYpkg`は公開するSVR4パッケージを表すディレクトリのパスであり、`Mypkg.mfst`は作成するマニフェスト・ファイルの名前です。

IPSパッケージの配布

前述のように、IPSパッケージは、ユーザーによって、パッケージ・パブリッシャがメンテナンスするパッケージ・リポジトリからインストールされます。リポジトリは、ネットワークベースの場合とファイル・システムベースの場合があります。ネットワークベースのリポジトリには、通常、インターネット経由でアクセスするか、企業のイントラネット経由または限定された数の他のシステムだけがアクセス可能な特定のサーバー経由（実験室またはデータセンター環境などの場合）でアクセスします。

インターネット経由でアクセス可能なネットワークベースのリポジトリの例は、次のとおりです。

- http://pkg.oracle.com
- http://pkg.sunfreeware.com:9000

Oracle Solaris 11が企業全体で広く使用されている場合、ネットワークベースのリポジトリを企業のイントラネット上に配置できる可能性があります。データセンターや実験室内では、外部ネットワーク・アクセスが制限されている場合があり、そのようなときは、ネットワークベースのリポジトリを実験室またはデータセンターの範囲内で利用可能にすることができます。

ネットワークベースのリポジトリへのアクセスの要件は、そのリポジトリで公開されるパッケージに大きく依存します。前述した使用例は、Oracle Solaris 11ディストリビューションを構成するすべてのパッケージが含まれるOracle Solaris 11リリース・リポジトリの内容にアクセスするための一般的なソリューションです。

ネットワークベースのリポジトリの代替方法は、ファイル・システムベースのリポジトリです。ファイル・システムベースのリポジトリは、IPSパッケージのローカル・インストールに使用されます。このリポジトリからパッケージをインストールするユーザーは、同じ`pkg(5)`コマンド・セットを使用しますが、代替のコマンドライン・オプションを利用します（または、`packagemanager`の構成にいくつかの変更を加えます）。

POSIX準拠ヘッダー・ファイル

現在、29の宣言が、POSIX:2008またはIEEE Std 1003.1-2008仕様に適合しています。インタフェースは同じですが、関数プロトタイプは変更されています。dirent.h、iconv.h、ndbm.h、stdlib.h、unistd.h、またはxti.hで宣言される29の関数を使用するコードをコンパイルしない場合は、最新のPOSIX仕様および変更されたヘッダーに適合するようにコードを変更することを推奨します。コードが大量であるために適合させることができない場合は、コンパイル行で、-D__USE_LEGACY_PROTOTYPES__を指定することにより、古い型式の宣言を使用してコンパイルできます。

一部の削除されたパッケージおよびフレームワーク

この項では、Oracle Solaris 11から削除されたソフトウェアについて説明します。これには、アプリケーション、ライブラリ、ミドルウェア、ユーティリティ、およびドライバが含まれます。オラクルでは、Oracle Technology Network（OTN）において、新しい“Oracle Solaris 11に関連する機能の廃止を通知する”Webページ[8]を維持しています。このWebページを定期的に参照してください。

このWebページは、随時更新されます。この項では、既知の削除に関するリストの一部について説明します。また、適切な場合には、代替方法を提示します。

[8] http://www.oracle.com/technetwork/jp/systems/end-of-notices/eonsolaris11-392732-ja.html

アプリケーションおよびミドルウェア

- evolution-jescsがLightning 0.3に置き換えられました。
- StarOfficeが削除されました。OpenOfficeは、下位互換性を実現するために個別にインストールできます。
- Apache 1.3がApache httpd 2.2に置き換えられました。
- MySQL 4.0、5.0がMySQL 5.1にアップグレードされました。
- PostgreSQLが削除されました。代替製品は、オラクルのRDBMS、MySQL、またはコミュニティ版PostgreSQLです。
- Berkeley DB 4.2が4.7にアップグレードされました。
- メディア管理システム（MMS）が廃止されました。
- NIS+がLDAPネーム・サービスに置き換えられました。

ライブラリ

- audit_user(4)とgetausernam(3bsm)の両方がuser_attr(4)およびprof_attr(4)の“audit_flags”キーワードに置き換えられました。
- getacinfo(3bsm)およびaudit_control(4)がSMFの使用によって不要になりました。
- 形式とメニュー言語のインタプリタ（FMLI）が廃止されました。
- グラフ/スプライン機能がGNUのplotutilsで対応できるようになりました。
- iSCSIターゲット・デーモンがCOMSTARに置き換えられました。
- SEAがNet-SNMP 5.4.1に置き換えられました。
- SERがlibsipに置き換えられました。
- XsunがSunRay用のXorgまたはXnewtに置き換えられました。
- Oracle Solaris x86上のXsunがXorg Xサーバーに置き換えられました。
- 古くなった@euroロケールが廃止されました。
- 短縮ロケールも削除されました。
- mptドライバの古くなったioctlインタフェースがPSARC/2006/544に置き換えられました。
- OCF/SCFスマートカード・フレームワークが削除されました。
- pcmciadおよびpemがdevfsadmに置き換えられました。
- X Image Extension（XIE）ライブラリが削除されました。画像処理は、さまざまなライブラリおよびユーティリティで広く実行されます。

ユーティリティ

- Linux、Oracle Solaris 8、およびOracle Solaris 9ブランド・ゾーンがサポートされなくなりました。
- /usr/ucb開発ツールがSun Studioツールと置き換えられました。
- audit_control/startup、bsmrecord/conv/unconvがauditrecordおよびSMFと置き換えられました。
- bdftosnfおよびshowsnfコマンドがbdftopcfおよびshowfontと置き換えられました。
- CD Font AdministratorがGNOMEのフォント制御パネルに置き換えられました。
- mixerctlがaudioctlに置き換えられました。
- グラフ表示機能がgnuplotによってサポートされるようになりました。
- sag(1)がkSarおよびsar2rddcによって実行されるようになりました。
- xmhがThunderbirdおよびEvolutionの使用によって不要になりました。
- xorgcfgおよびxorgconfigが削除されました。Xorgの自動構成が標準になりました。
- XprintがCUPSおよびGNOMEの印刷システムに置き換えられました。

ドライバ

以下のドライバが削除されるのは、単に、ハードウェアが古くなったためか、ハードウェアの関連性がなくなったためです。

- CreatorおよびCreator 3D（ffb）SPARCグラフィックス
- Elite3D（afb）SPARCグラフィックス
- Expert3DおよびXVR-500（ifb）SPARCグラフィックス
- m64 SPARCグラフィックス
- SPWR NICドライバ
- XVR-600およびXVR-1200（jfb）SPARCグラフィックス
- GXシリーズのSBUSグラフィックス・カード用のCG6ドライバ
- PCMCIAメモリ・カードのサポート
- SBPRO（Sound Blaster 16ドライバ）
- SPARC用のSDカードのサポート
- 2006よりも古いSPARCワークステーションのサポート
- XVR-4000、XVR-1000、PGX32グラフィックス・カード

国際化

削除された@euroロケール（該当地域：ES、DK、AT、DE、GR、IE、FI、FR、IT、NL、PT）

アプリケーションが、ES、DK、AT、DE、GR、IE、FI、FR、IT、NL、PTの国コードに関して@euroロケールへの切り替えを試みると、Oracle Solaris 11ではエラーになります。

ユーロは、スペイン、デンマーク、オーストリア、ドイツ、アイルランド、フィンランド、フランス、イタリア、オランダ、およびポルトガルにおいて、1999年1月1日に法定通貨となりました。ユーロは、2002年7月までに国立銀行券と置き換わりました。

この期間は、ユーロの国別書式ロケールと古い通貨の国別書式ロケールを持つ必要がありました。この問題は、通常のロケール（たとえば、fr_FR.ISO8859-15）に加えて、ロケール名の最後に“@euro”という修飾子の付いた“ユーロ”ロケールを提供することによって解決されました。

@euroロケールは、二重通貨期間が終了した2002年7月に不要になりました。前述した国の@euroロケールは、Oracle Solaris 11に含まれていません。代わりに使用する必要のあるロケールは、最後に“@euro”修飾子の付いていないロケールです。@euroの付いていないロケールの機能は、Oracle Solaris 10以降の@euroロケールと同じです。ただし、Oracle Solaris 9は、@euroの付いていないロケールによって古い通貨の国別書式を適用していました。次の表に、Oracle Solaris 9、10、および11の間での機能の相違を示します。

Oracle Solarisのバージョン	@euroロケール	通常のロケール
Oracle Solaris 9	ユーロの国別書式	古い通貨の国別書式
Oracle Solaris 10	ユーロの国別書式	ユーロの国別書式
Oracle Solaris 11	利用不可（@euroのない同等物を使用）	ユーロの国別書式

@euroロケールについて詳しくは、Oracleホワイト・ペーパー『Euro Currency Support in the Oracle Solaris Operating Environment』を参照してください。

短縮ロケールの廃止

Oracle Solaris 10には、Oracle Solarisで提供され、サポートされていた113のいわゆる短縮ロケール名（“ar”、“de”、“fr_CH”など）が含まれていました。これらの短縮ロケールには、意図する地域とコード・セットのいずれかまたは両方を識別する際に固有の曖昧さがあります。

最新のすべてのPOSIX様式オペレーティング・システム（Oracle Solaris、AIX、HP-UXなど）とさまざまなLinuxディストリビューションは、次の規則に従ったロケール名を提供およびサポートしています。

language_territory.codeset[@modifier]
ここで、“language”はISO 639で規定されている2文字の国コード、“territory”はISO 3166で規定されている2文字の大文字の地域コード、“code set”は業界で広く受け入れられているいずれかのコード・セット名、オプションの“@modifier”はロケール固有の追加属性（照合順序など）を示す任意の文字列です。

Oracle Solaris 11では、これらの短縮ロケールが提供されなくなりました。アプリケーションは、新しく導入

された“ロケール・エイリアシング”[9]と呼ばれるSolarisテクノロジーにより、引き続き短縮ロケールを使用できます。ISO準拠ロケールへの簡潔なマッピングについては、Solaris 11のlocale_alias(5)[10]のマニュアル・ページを参照してください。

コーディングに関する既知の問題によるクラッシュのトラブルシューティング

Oracle Solaris 10とOracle Solaris 11には継続的なバイナリ互換性がありますが、開発者は、Oracle Solaris 11において、特定のアプリケーション・バイナリによる、以前には見られなかった不可解なエラーの発生を経験する可能性があります。この項では、不安定なプライベートAPIに変更を加えたために、元から不適切だったコーディングによってOracle Solaris 11で新しいエラーが発生したように見えるいくつかの事例について詳しく説明します。

事例1：不整列のmutexに基づくセグメンテーション違反

- 環境：
  - Oracle Solaris 10システム、パッチ・バージョン137111-01～137111-08（SPARCのみ）
  - Oracle Solaris 11（SPARCのみ）
- 症状：
  - アプリケーション・バイナリで突然エラーが発生し、“Memory fault”というエラーだけが表示される。
- 根本原因：
  - mutex_tおよびpthread_mutex_t型のオブジェクトは、8バイト境界で整列されるアドレスで始まる必要があります。このように開発されていないアプリケーションでは、SPARC用のOracle Solaris 11においてエラーが発生する可能性があります。
- 正式な解決方法：
  - mutexが8バイト境界で整列されるようにソース・コードを変更し、バイナリを再構築します。
- 回避方法：
  - セグメンテーション違反は、環境変数の_THREAD_LOCKS_MISALIGNEDを1に設定することによって回避できます（ただし、ある程度パフォーマンスが低下します）。

[9] http://docs.oracle.com/cd/E26924_01/html/E27144/glmen.html#scrolltoc

[10] http://docs.oracle.com/cd/E26924_01/html/E29116/locale-alias-5.html#REFMAN5locale-alias-5

事例2：静的にリンクされたlibcライブラリによるセグメンテーション違反

- 環境：
  - Oracle Solaris 10（発生する可能性がある）
  - Oracle Solaris 11（高い確率で発生する）
- 症状：
  - アプリケーション・バイナリでセグメンテーション違反のためにエラーが発生する。
- 根本原因：
  - アプリケーションのlibc.aおよびlibthread.aへの静的リンクは、Oracle Solaris 10以降は禁止されています。Oracle Solaris 9以前で作成され、この静的リンクを含むアプリケーション・バイナリは、エラーとなり、Oracle Solaris 10以降での動作が保証されません。Oracle Solarisのlddコマンドを使用すると、この状況を識別できます。
- 正式な解決方法：
  - ソース・コードを変更し、これらのライブラリに静的にリンクされていない環境を構築します。追加情報：Rod Evanのブログの記事[11]で、技術的な背景が分かりやすく説明されています。

Oracle Solaris 11への移行を管理するためのツール

Oracle Solaris Studio

Oracle Solaris Studioは、ISVがOracle Solaris 10および11用のアプリケーションを作成することを可能にする、従来のC、C++、およびFortran開発者向けの包括的なツール・スイートです。最新バージョンの製品（現時点ではStudio 12.3）を使用することを推奨します。最新バージョンの製品には、最新の最適化と、最新のプロセッサのサポートが組み込まれており、旧バージョンよりも機能が強化されています。

Oracle Solaris StudioのOTNページ[12]は、製品ダウンロード、ホワイト・ペーパー、技術情報記事、スクリーン・キャスト、Webキャスト、フォーラムなどのランディング・ページです。

概要

Oracle Solaris Studioは、基盤となるオペレーティング・システムおよびハードウェア向けに最適化されたツールを提供することで、スケーラブルで信頼できる高パフォーマンス・アプリケーションの構築を支援します。Oracle Solaris Studioには、コンパイラ・スイートと分析スイートという2つの主要なツール・スイートが含まれています。ツール・スイート内のツールは連携して動作するように設計されており、単独アプリケーション、マルチスレッド・アプリケーション、分散アプリケーションの開発向けに最適化された開発環境を提供します。

[11] http://blogs.oracle.com/rie/entry/static_linking_where_did_it

[12] http://www.oracle.com/technetwork/jp/server-storage/solarisstudio/overview/index.html

コンパイラ・スイート

このスイートに含まれるコンポーネントには、CおよびC++コンパイラ、Fortranコンパイラ、デバッガ、パフォーマンス・ライブラリがあります。

CおよびC++コンパイラは、高度なコード生成テクノロジーにより、最新のSPARCおよびx86ベースOracle Sunサーバー向けに最高レベルのパフォーマンスのアプリケーションを作成する開発者を支援します。このコンパイラは、堅牢でパフォーマンスに優れたパラレル・コードを構築するための信頼できる基盤を提供します。Oracle Solaris Studioソフトウェアは最新の言語標準をサポートしているだけでなく、GNU C/C++に対する互換性機能も備えており、以前のリリースとのソースおよびオブジェクト・レベルからの互換性も有しています。このソフトウェアを使用することにより、古いバージョンのコンパイラでコンパイルされたオブジェクトと新しいバージョンのコンパイラでコンパイルされたオブジェクトをシームレスにリンクさせることができます。

マルチコア・システムのハードウェア同時実行性を活用するため、コンパイラの自動並列化機能によって、並列アプリケーションの作成が簡素化されています。コンパイラは自動並列化機能を使用することで、シングル・スレッド・コードから安全かつ有益に並列化できる可能性を特定し、これらのセグメントをマルチスレッド・コードに自動変換します。さらに、このコンパイラは、タスクベースの並列処理をサポートするOpenMP 3.0仕様に対応しています。

Oracle Solaris Studioのコンパイラには、アプリケーション・パフォーマンスを向上するための最適化オプションが多数含まれています。マイクロアーキテクチャに固有の命令やプロファイル・フィードバックからプログラム全体の最適化にいたるまで、すべてを生成するため、このコンパイラはさまざまな個別オプションと使いやすいメタオプションの両方を提供し、積極的なアプリケーション・パフォーマンスの最適化を行っています。

CおよびC++コンパイラに加えて、Oracle Solaris Studioには、Fortranコンパイラも含まれています。このコンパイラは、Fortran77、Fortran90、およびFortran95の標準に対する互換性オプションを提供することで、技術市場における既存のコード・ベースをサポートします。

dbxデバッガは、アプリケーションの信頼性を確保するために役立ちます。dbxデバッガは、ソース・レベルのインタラクティブな事後およびリアルタイムのデバッグ・ツールであり、コマンドライン、IDE、およびスタンドアロンのグラフィカル・デバッガ（dbxtool）から使用できます。

技術計算市場のISV向けに、Oracle Solaris Studioは、計算集中型アプリケーションのパフォーマンスを最大限に高めるために役立つ高度な数値問題ソルバー・ライブラリを含むパフォーマンス・ライブラリも提供します。

適切なコンパイラ・フラグの選択

コンパイラは、コンパイラに提供されるフラグに応じて、さまざまな最適化を実行できます。マニュアルには、これらのフラグに関する情報が完全に記載されています。このガイドでは、コンパイラ・フラグの使用に関するもっとも重要な点についてのみ説明します。

最初に確認する点は、最適化されたコードを生成するために最適化フラグが必要であることです。このため、どのアプリケーションを構築する場合も、少なくとも-0の最適化レベルを使用することを推奨します。

最適化されていないビルドが使用される可能性のある唯一の状況は、アプリケーションがデバッグのためだけに使用される場合です。最適化されていないビルドにより完全なデバッグ機能を持つビルドが生み出されますが、最適化を使用すると、利用できるデバッグ情報の量が減る場合があります。デバッグ情報を生成するためのフラグは-gであり、このフラグは常に（最適化されたビルドであっても）含めることを推奨します。このフラグは、パフォーマンスにほとんど影響を与えません。

また、コンパイラは、積極的な最適化フラグである-fastフラグもサポートします。これにより、アプリケーションのパフォーマンスを向上させる多数の最適化が可能になります。ただし、このフラグでのパフォーマンスと-0でのパフォーマンスと比較することを推奨します。-fastの方が、パフォーマンスが優れている場合は、パフォーマンスの向上に役立つ-fastによって実現される最適化を決定し、これらを明示的に使用してください。ここでは、理解しているコンパイラ・フラグであり、かつパフォーマンスを向上させるものだけを使用することを推奨します。

64ビットまたは32ビットのコード

重要な意志決定の1つは、作成するアプリケーションを32ビットと64ビットのどちらにするかということです。この変更はアプリケーションの内部構造に影響を与えるため、結果的に、一部のアプリケーションが64ビット・コンパイルに対応しないことに注意が必要です。Cアプリケーションの場合は、lintを使用して、64ビットの移植性の問題をチェックすることができます。

64ビット・アプリケーションに移行する主要な動機は、アプリケーションがアドレス指定できるメモリを増やすということです。32ビット・アプリケーションでは4GBに制限されますが、このメモリ容量は、今日発生する規模の問題のいくつかを解決するには不十分です。

x86プロセッサ上では、一般に、64ビット・アプリケーションに移行することにより、パフォーマンスが向上します。このメリットは、命令セットにおける改善と、関数コールの方法における改善によって生まれます。ただし、一般に、64ビット・アプリケーションは、32ビットの同等物よりも多くのメモリを消費するため、すべてのアプリケーションでパフォーマンスが向上するわけではありません。アプリケーション内のデータ構造がポインタ変数（またはlong型の変数）によって支配される場合、メモリ増加のオーバーヘッドが、命令セットの改善によるパフォーマンスの向上を上回る可能性があります。

32ビットのSPARCアプリケーションは、すでに、x86を64ビット移行することによって得られる最適化のほとんどを備えています。このため、SPARC上の64ビット・アプリケーションは、通常、追加のメモリ・フットプリントのためにパフォーマンスが若干低下します。

高度なコンパイラの最適化

コンパイル時間が長くなるものの、多くの場合に実用的なパフォーマンスの向上が得られる、2つのコンパイラの最適化があります。これらはクロスファイル最適化とプロファイル・フィードバックであり、この2つのオプションは多くの方法で補完します。

クロスファイル最適化では、ファイルをコンパイルするときとアプリケーションをリンクするときの両方で-xipoフラグを使用する必要があります。これにより、コンパイラは、リンク時に2回目の最適化パスを実行して、パフォーマンスが向上する可能性のある追加の場所を探します。このため、リンク時間が長くなるデメリットがありますが、すべてのオブジェクト・ファイルを調べて、パフォーマンス向上の可能性を探すことができるメリットがあり、ソース・コードが多数のファイルに分散している場合には特に有益です。

プロファイル・フィードバックは、ブランチによって支配されるアプリケーションを特に対象としています。ブランチ支配のコードは、通常、他の方法で最適化することが困難であるため、この方法は重要です。この方法は、-xprofile=collect:<file>と-xprofile=use:<file>のフラグの組合せによって制御されます。最初のフラグによって、インスツルメント処理された実行可能ファイルが生成されます。インスツルメント処理された実行可能ファイルは、異なる代表的なワークロードで複数回実行でき、アプリケーションの実行時動作に関するデータが収集されます。このデータは、その後、-xprofile=use:<file>フラグによる2回目のコンパイラ・パスで使用されます。2回目のパスでは、コンパイラが、インスツルメント処理された実行時に収集されたデータを使用して最適化を決定できます。これにより、コンパイラは、一般にすべてのワークロードについてパフォーマンスが向上するコード・レイアウトや関数インライン化に関するより優れた決定を行うことができます。

C++標準ライブラリの使用

C++コンパイラは、最初から、3つのバージョンのC++標準ライブラリをサポートしています。同じアプリケーション内に異なるバージョンのSTLを混在させると、未定義の動作が発生します。このため、アプリケーションとそのすべてのライブラリを単一のバージョンで標準化することが重要です。

サポートされる3つのバージョンは、デフォルトSTL、stlport4、およびApache STLです。デフォルトSTLは、C++標準より前から存在し、多数の機能が欠落していますが、すべてのシステム上に存在します。また、代替バージョンよりもパフォーマンスの点で劣る場合があります。

コンパイラは、stlport4が代替バージョンとして含まれています。このライブラリは、-library=stlport4フラグを使用して選択できます。このライブラリは、一般にデフォルトSTLよりも高速で動作し、より完全な機能セットを備えています。また、このライブラリは、コンパイラに付属しており、アプリケーションとともに配布する必要があります。

Apache STLは、-library=stdcxxコンパイラ・フラグで利用できます。このライブラリもデフォルトSTLよりも高速で動作する傾向があり、より完全な機能セットをサポートします。Apacheライブラリは、Solarisの一部として配布されます。

アプリケーションのリンク

アプリケーションは、コンパイラを起動してリンクする必要があります。リンクを実行する際は、決してリンカーを直接呼び出さないでください。これは、アプリケーションが正常に動作するために必要なサポート・ライブラリをコンパイラが認識しているためであり、コンパイラは、ライブラリのリンクの正しい順序も認識しています。

純粋な言語のアプリケーションのリンクは、適切なコンパイラを使用して実行する必要があります。たとえば、純粋なCのアプリケーションをリンクするには、ccを使用する必要があります。言語が混在するアプリケーションについては、コードにCとFortranが混在する場合はFortranコンパイラを使用し、C++のコードがFortranまたはCと混在する場合はC++コンパイラを使用します。

実行時ライブラリに依存するアプリケーションは、-Lフラグを使用してそれらのライブラリのリンク時に存在する場所を指定し、-Rフラグを使用してライブラリが実行時に存在する場所を指定する必要があります。$ORIGINトークンを-Rフラグとともに使用して、アプリケーションに関連する実行時ライブラリのパスを示すことができます。アプリケーションを正常に機能させるために、LD_LIBRARY_PATHの設定には決して依存させないでください。

移植可能なアプリケーションの作成

移植可能なアプリケーションを作成するには、サポートを予定しているもっとも古いビルドのSolaris上で、そのプラットフォームでサポートされるもっとも新しいツールを使用して構築する必要があります。多くの場合、これは、Solaris 10上でOracle Solaris Studio 12.3を使用して構築することを意味します。Solarisのバイナリ互換性保証は、古いバージョンのSolaris上で動作するアプリケーションが、より新しいバージョンのSolaris上でも動作することを保証しています。このため、Solaris 10上で構築されるアプリケーションは、Solaris 11で引き続き動作します。

Oracle Solaris Studioにより、アプリケーションでサポートされるプロセッサの範囲を選択できます。デフォルトでは、コンパイラ・リリースでサポートされるすべてのプロセッサ上で動作する“汎用”バイナリが生成されます。

ただし、アプリケーションが特定のプロセッサ上または特定の機能セットをサポートするプロセッサ上でしか動作しないいくつかの例があります。アプリケーションは、一連のプロセッサの機能を利用することによってパフォーマンスが向上する場合があります。この機能を利用する場合の唯一のデメリットは、それらの機能を備えないプロセッサ上でアプリケーションが動作しないことです。

x86プロセッサでは、場合により、SIMD命令を使用することによってパフォーマンスが向上します。x86上の64ビット・アプリケーションはSSE2拡張命令セットの存在を前提とすることができますが、32ビット・アプリケーションはそれを前提とすることができません。32ビット・アプリケーションの場合は、コンパイラが-xarch=sse2コマンドライン・フラグを使用してSSE2命令を生成できることにより、メリットが得られる場合があります。32ビットと64ビットの両方のアプリケーションでこれらの命令の完全なメリットを得るには、これらの命令を生成可能なコード内のループをコンパイラに積極的に識別させる-xvector=simdフラグを追加する必要があります。

最近のSPARCプロセッサは、融合積和演算命令をサポートします。これは、単一のステップで浮動小数点の乗算と加算を実行する命令で、システムの浮動小数点能力が潜在的に倍増します。これらの命令を生成するには、-xarch=sparcfmaf -fma=fusedコンパイラ・フラグを使用します。すべての浮動小数点最適化と同様に、これらのフラグも結果が異なる原因になる場合があります。乗算と加算を個別に実行する場合、結果がレジスタに書き込まれる際に2回の丸め操作が実行されます。乗算と加算を組み合わせた操作の場合、中間値ではなく最終値だけが丸められます。このため、結果の最下位の数字が異なる可能性があります。

パラレル・アプリケーションの作成

Studioコンパイラは、マルチスレッド・アプリケーションを開発するための複数の方法をサポートします。このツールは、マルチスレッド・コードの分析、最適化、およびデバッグもサポートします。

既存のシリアル・コードからマルチスレッド・アプリケーションを作成するもっとも簡単な方法は、自動パラレル化の使用です。これにより、コンパイラは、タスクを複数のスレッドに分散できるコード内のループの検出を要求されます。この分析は-xautoparおよび-xreductionフラグによって制御され、コンパイラがどのループをパラレル化したかについては、-xloopinfoフラグを使用して調べることができます。

OpenMPは、パラレル化のためのより柔軟なアプローチです。この場合、アプリケーションをパラレル化する方法をコンパイラに示す命令を手動でコードに挿入する必要があります。OpenMPを使用する場合、スレッドを管理したりコードを生成したりする厄介な作業をコンパイラが実行するため、パラレル・アプリケーションの作成が非常に簡単になります。コンパイラがOpenMPディレクティブを認識するかどうかは-xopenmpコンパイラ・フラグによって制御されます。-xvparaフラグにより、コンパイラが検出したパラレル化に関する問題を示すメッセージが発行されます。

パラレル・コードを作成するための第3のアプローチは、POSIXスレッドの使用です。このモードは、ツールによって完全にサポートされており、-mt以外の特殊なコンパイラ・フラグは不要です。-mtフラグにより、マルチスレッド・コードが作成されていることが示され、ヘッダー・ファイルで定義されるAPIが変更されてスレッドが安全であることが反映されます。

分析スイート

分析スイートには、アプリケーションに対する可観測性を向上するための高度なツール・スイートが含まれています。具体的には、パフォーマンス・アナライザ、コード・アナライザ、スレッド・アナライザが含まれています。

パフォーマンス・アナライザ

パフォーマンス・アナライザは、パフォーマンスに影響を与えている機能、コード・セグメント、ソース行を特定するだけでなく、パフォーマンスを最適化するためのチューニングに必要なツールも提供することにより、アプリケーション・パフォーマンスのボトルネックを明確にします。コンパイラによってさまざまな情報が示される注釈付きのコンパイラ記録リストから、最適化ステータスや実行時のスレッド・パフォーマンスまで、ユーザーはパフォーマンスのホットスポットをGUIによって視覚化できます。パフォーマンス・アナライザ・ツールを使用すると、シングルスレッド・アプリケーションだけでなくマルチスレッド・アプリケーションのプロファイリングも実行できます。

パフォーマンス・アナライザは、特別なビルドのアプリケーションを必要としません。ただし、アプリケーションがデバッグ情報をともなって構築されている場合は、より詳細な結果を生成できます。アプリケーションのプロファイリングは、2ステップのプロセスです。プロファイルは、collectツール（たとえば、collect./a.out）の下でアプリケーションを実行することにより収集されます。これにより、デフォルトでは、test.1.erという名前の実験リポジトリが作成されます。この実験は、アナライザGUIまたはコマンドライン・ツールのer_printを使用して調べることができます。パフォーマンス・アナライザは、collectの-P <pid>オプションを使用して、実行中のプロセスからデータを収集でき、collectの-c onオプションを使用して、正確な命令回数データを生成できます。

コード・アナライザ

コード・アナライザは、動的分析、静的分析、コード・カバレッジ分析を利用して、メモリ・リークやメモリ・アクセス違反を含むアプリケーションの脆弱性を検出することで、アプリケーションの安定性を確保します。コード・アナライザを使用することで、アプリケーションの信頼性を向上させるために役立つ包括的なメモリ・エラー検出が可能になります。コード・アナライザはアプリケーションのコンパイル時には静的分析を調べ、アプリケーションの実行時には動的分析を調べて、エラーが発生した可能性のある箇所についてのフィードバックを提供します。さらに、コード・カバレッジ・データを調べて、テスト・スイートの対象になっていない機能の情報を提供するとともに、これらの機能を対象にすることでどのようなメリットが得られるかについてアドバイスを提供します。

静的エラー検出は、コンパイル時に-xanalyze=codeコンパイラ・フラグを使用することによって有効になります。これにより、アプリケーションで検出された問題に関するレポートが生成されます。このレポートは、コード・アナライザGUIを使用して表示できます。静的エラー・チェックは、ソース・コードの詳細な分析によってコンパイラが検出できるエラーのみを表します。多くのエラーは、コードの純粋に静的な分析では検出できないため、コード・アナライザには、実行時分析のサポートも含まれています。

コード・アナライザは、コード・カバレッジ分析をサポートします。コード・カバレッジは、テスト・ワークロードによってアプリケーションのすべての機能が対象になっているかどうかを確認するために役立ちます。テストの対象となっているコードの部分を調べることよりもテストの対象となっていないコードの部分の削減に重点を置くことの方が重要であるため、このツールには“uncover”（見つけ出す）という名前が付いています。ただし、コード・カバレッジはテスト・スイートの弱点を示すだけで、潜在的な実行時の問題を検出するわけではありません。

discoverツールでは、アプリケーションの実行時の動作が評価され、一般的なメモリ・アクセス・エラーが検出されます。メモリ・アクセス・エラー（配列の範囲外への書込み、初期化されていない値の読取りなど）は、もっとも一般的な種類のプログラミングおよびセキュリティ・エラーを表します。静的分析ツールでは、それらを検出するための詳細な分析を実行できません。discoverの重要なメリットの1つは、アプリケーションの特別なビルドを必要としないことです。アプリケーションがコンパイルされていれば、実行可能ファイルでdiscoverコマンドを実行するだけで、発生した実行時メモリ・アクセス・エラーをレポートするアプリケーションのバージョンが生成されます。アプリケーションの特別なビルドを必要としないことの非常に重要な点は、特別なビルドを実行しなければならない場合にはアプリケーションで発生するエラーの詳細が得られなくなり、異なる最適化によって構築されたバージョンのアプリケーションからのエラーを得ることです。

スレッド・アナライザ

スレッド・アナライザを使用すると、検出の難しいスレッド・エラーを、エラーの発生前に特定できます。このツールは、潜在的な競合状態とデッドロック状態を実行時に検出し、それらをアプリケーション内のソース行にマッピングした上で、ユーザーがコマンドライン・オプションまたはグラフィカル・ユーザー・インタフェース（GUI）オプションを使用して結果を表示できるようにします。マルチスレッド・プログラミング・エラーをデバッグすることは困難な作業ですが、スレッド・アナライザは、確定的な動作と非確定的な動作の両方を特定することによって、このプロセスを簡素化します。

スレッド・エラーの詳細を収集する仕組みは、collect -r allの下でインスツルメント処理された実行可能ファイルを実行して、結果をスレッド・アナライザで調べるというものです。インスツルメント処理されたバージョンの実行可能ファイルは、-xinstrument=datarace コンパイラ・フラグを使用して構築するか、discover -idatarace./a.outコマンドを使用して既存のバイナリをインスツルメント処理することによって得られます。

Oracle Solaris Studio IDE

Oracle Solaris Studioは、コンパイラ・スイートと分析スイートに加えて、統合開発環境（IDE）を提供しています。このIDEは、CおよびC++の開発者向けに特化されており、開発生産性の向上に役立ちます。

Oracle Solaris StudioのIDEはNetBeansプラットフォームをベースに構築されており、インテリジェントな言語認識コード・エディタや、コードの補完、コードの折りたたみ、構文ハイライトなど、開発者生産性を高める高度な機能を豊富に提供しています。

Oracle Solaris Studio IDEは、Oracle Solaris以外のクライアントからOracle Solarisアプリケーションを作成することを可能にするリモート開発もサポートしています。また、動的なクラス・ブラウザとメイク・ファイルの作成ウィザードを提供しており、高度な設定を行うことができます。

Oracle Solaris 11 Preflight Application Checker[13]

“Oracle Solaris 11 Preflight Application Checker”の目的は、Oracle Solaris 10からOracle Solaris 11 Expressおよび Oracle Solaris 11へのアプリケーションの移行を容易にすることです。“Oracle Solaris 11 Preflight Application Checker”（S11CompatCheckTool）は、すでにOracle Solaris 10で使用可能なアプリケーションを分析して、Oracle Solaris 11の既知の互換性に関する問題をレポートします。

S11CompatCheckToolを使用することにより、開発者は、より迅速に、より少ない労力で、アプリケーションをOracle Solaris 11上でサポートできるようになります。

概要

現在Oracle Solaris 10上で動作しているアプリケーションの大部分は、変更しなくてもOracle Solaris 11上で引き続き動作すると予想されますが、ごく一部には、Oracle Solaris 11上でエラーを発生させるいくつかの非互換性を持つアプリケーションが存在する可能性があります。Oracle Solaris 11 Preflight Application Checkerは、既存の非互換性を評価し、アプリケーションを新しいプラットフォームでサポートするために必要な変更を示すことによって、開発者を支援します。Oracle Solaris 11の互換性チェック・ツールを、Oracle Solaris 10ベースの開発またはテスト・デプロイ・システム上で使用することにより、開発者は、Oracle Solaris 11におけるアプリケーションの互換性に関する問題を把握できます。これは、すでにOracle Solaris 10で使用可能なアプリケーション上で、S11CompatCheckToolの“Source Code analysis”、“Static Binary analysis”、および“Runtime analysis”モジュールを組み合わせて実行することによって実現されます。

バイナリ/ELFアナライザ

Oracle Solarisのすべてのインタフェースは、バージョンが異なっていても上位互換性が確保されており、パブリックかプライベートかを示すラベルが付けられています。パブリック・インタフェースはOracle Solaris上で動作するアプリケーションまたはミドルウェアが使用するためのインタフェースであり、プライベート・インタフェースは他のOracle Solarisインタフェースのみが使用するためのインタフェースです。

[13] URL：http://www.oracle.com/technetwork/jp/server-storage/solarisstudio/overview/index.html

バイナリの非互換性に関する問題のある状況は、そのほとんどがプライベート・シンボルの使用によるものですが、不安定な使用方法によってその他の問題が発生する可能性もあります。これには、コンパイル時のOracle Solaris共有オブジェクトのアプリケーションへの静的リンクなどが含まれます（Oracle Solarisインタフェースが時間の経過とともに変更されている可能性があるため）。Oracle Solarisインタフェースが進化すると、静的にリンクされた共有オブジェクトとその他のOracle Solarisライブラリの間での対話も、バイナリ非互換性のためにリスクを持つようになります。このため、互換性チェックの段階で、静的リンクがオブジェクトのプロファイルに示されていると、互換性チェック・ツールはこれが危険であるとレポートします。さらに、特定のシンボルを調べることによって、部分的な静的リンクをチェックするために経験則が使用されます。

バイナリ・スキャンの最初の段階で、アプリケーションによって使用されるすべてのバインディングと関連シンボルが抽出されます。S11CompatCheckToolは、ユーザー提供のアプリケーション環境変数のLD_LIBRARY_PATHを調べて、アプリケーションが実行時に使用するライブラリを判定します。次に、それらのライブラリを解析して、調べているオブジェクトのパス名と、Oracle Solaris共有ライブラリへのシンボル・バインディングを特定します。プロファイリング段階で使用されるもう1つのユーティリティが、elfdumpコマンドです。このコマンドは、オブジェクトが依存している動的リンクの情報をダンプします。

ソース・コード・アナライザ

静的ソース・コード分析モジュールによって、C/C++ソース・コード、Kシェル、およびその他のシェル・スクリプトに含まれている互換性に関する問題をチェックすることができます。このモジュールは、おもに、削除された、名前が変更された、修正された、または推奨されなくなったOracle Solarisライブラリ関数およびシェル・コマンドの使用をチェックします。

実行時アナライザ

実行時分析モジュールを既存のOracle Solaris 10テスト環境で実行して、潜在的な互換性に関する問題のレポートを得ることができます。このS11CompatCheckToolのモジュールは、Oracle Solaris DTraceを使用してプロセスの詳細を調べ、Oracle Solaris 10上で使用可能な他のシステム・ツールを起動することがあります（必要な場合）。

S11CompatCheckToolは、システム全体にわたるチェックから特定のプロセスのみのチェックにいたるまで、さまざまな範囲で実行するために構成できます。

実行時アナライザ・ツールは、次のような状況における既知の問題をレポートできます。

- 推奨されなくなった（EOLになった）コマンドの呼出し
- 名前や場所が変更されたコマンドの呼出し
- 推奨されなくなったオプションを持つコマンドの呼出し
- “推奨されなくなった、または名前や場所が変更された”ファイルやデバイスを開く、読み出す、書き込む
- 旧世代のバージョンおよび削除されたバージョンのユーティリティやパッケージの使用（postgreSQLを使用するためにハードコードされたパスなど）
- 削除された、または交換されたデバイス・ドライバ・インタフェースの使用
- 推奨されなくなった、またはEOLになった関数、API、およびシンボルの使用または呼出し

– プライベート・インタフェースの使用または呼出し
– 推奨されなくなった、または名前が変更されたライブラリの動的読込み

Oracle Solaris 11 Preflight Application Checkerの取得

非バンドル・バージョンのS11CompatCheckToolは、X64およびSPARCアーキテクチャ用に無償でダウンロードできます。[14]

結論

Oracle Solaris 10またはそれ以前のOracle Solarisオペレーティング・システム用に正しく開発されたアプリケーションは、Oracle Solaris 11とバイナリ互換性があります。ただし、コンポーネント・バージョンの変更、フレームワークの除外、およびコマンドラインの変更に関連したオペレーティング・システムの進化により、Oracle Solaris 11上で正常に動作させるためにアプリケーションに変更を加えなければならない場合もあります。このガイドでは、Oracle Solaris 11で動作させるためにアプリケーションの変更が必要になる場合の可能性の高い原因を示しました。

[14] ダウンロード・サイトのURL：http://www.oracle.com/technetwork/jp/server-storage/solarisstudio/downloads/index.html

Oracle Solaris 11 ISV用導入ガイド
2012年5月

著者：Joseph Balenzano、Bruce Chapman、Darryl Gove、Abhishek Gupta、Malcolm Kavalsky、William Knoche、Eric Reid、Stefan Schneider、Caryl Takvorian、
Andrew WaltonOracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口：
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together