# Oracle Solaris 11 移行の開発ガイド

**Part No: E62872**
2016 年 11 月

**Part No: E62872**



**ドキュメントのアクセシビリティについて**

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

**Oracle Supportへのアクセス**

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

# 目次

# このドキュメントの使用法

- **概要** – このドキュメントでは、ほかの UNIX ベースのプラットフォーム (Linux、HP-UX、AIX など) から Oracle Solaris 11 OS へ移行するために役立つ、アプリケーション開発者向けの情報とガイドラインを提供します。Oracle Solaris 11 でアプリケーションを開発する場合に役立つツールおよびヒントについても説明しています。
- **対象読者** – ほかの UNIX ベースのプラットフォームから Oracle Solaris への移行を行う開発者。
- **必要な知識** – UNIX ベースのプラットフォーム上でのアプリケーション開発の基本的な経験。

## 製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは `http://www.oracle.com/pls/topic/lookup?ctx=E62101-01` で入手可能です。

## フィードバック

このドキュメントに関するフィードバックを `http://www.oracle.com/goto/docfeedback` からお聞かせください。

♦ ♦ ♦ 第 1 章

# 1 Oracle Solaris への移行について

この章では、Oracle Solaris 11 への移行の利点、および移行前に考慮する必要のある要因について説明します。

## Oracle Solaris への移行の利点

Oracle Solaris 11 は、一連の多数のネイティブなコマンド (Oracle 固有)、および Linux、HP-UX、AIX などのほかの UNIX ベースのプラットフォームで使用可能なものと同様のオープンソースのコマンド、ツール、ライブラリ、およびプラットフォームサービスを提供しています。

Oracle Solaris 11 の主な利点は次のとおりです。

- 移植可能でスケーラブル、相互運用可能で互換性があります。
- 異なるアーキテクチャー間の移植性をサポートしています。
- アプリケーションの移植性に関する標準に準拠しています。
- Oracle ハードウェアデバイス用に最適化されたツールをサポートしています。
- 広範なハードウェアをサポートしています。
- コードを修正せずに大規模システムに拡張できるように設計されているインフラストラクチャーがあります。
- IEEE Std 1003.1 および IEEE Std 1003.2 (一般には、それぞれ POSIX.1 および POSIX.2 と呼ばれます) をサポートしています。具体的には、Oracle Solaris 11 は POSIX.1-2004 および POSIX.1-2008 に準拠しています。詳細は、`https://www2.opengroup.org/ogsys/catalog/1003.1_2004` および `http://pubs.opengroup.org/onlinepubs/9699919799/mindex.html` を参照してください。
- ハードウェアからアプリケーションまで、24 時間週 7 日のエンドツーエンドのサポートを Oracle (単一のベンダー) から提供しています。
- Perl、Ruby、Python などのプログラミング言語をサポートしています。
- さまざまな Oracle Solaris OS リリースにおいてアプリケーション間のバイナリ互換性を保証するバイナリ保証プログラムを提供しています。詳細は、Oracle Solaris の保証プログラム (`http://www.oracle.com/technetwork/server-storage/solaris/overview/guarantee-jsp-135402.html`) を参照してください。

- オープンソースのソフトウェアに対するサポートを提供しています。

---

**注記 -** Oracle Solaris 11 は、ほかの Oracle 製品とともに使用することにより、統合されたスタック (ハードウェアからアプリケーションまで) を利用できます。統合されたスタックは、アプリケーションの開発および配備のエンドツーエンドのプロセスを提供するように最適化されています。Oracle 統合システムを使用すると、開発作業を支援するために連携して動作するように調整されたコンポーネントを使用できます。詳細は、「Engineered Systems Documentation (`http://docs.oracle.com/en/engineered-systems/`)」を参照してください。

---

# Oracle Solaris への移行前の考慮事項

Oracle Solaris OS に移行する前に、次の要因を考慮する必要があります。

- ファイルシステムフォーマットの相違点
- 32 ビットおよび 64 ビットアーキテクチャー
- プラットフォームのエンディアン
- Oracle Solaris で必要なコンパイラオプションのサポート
- スレッドモデルの相違点
- ビルドツールおよびその他のビルドの依存関係 (`gmake`、`dmake`、`make`、ANT など)
- データベース構成
- 移行のために必要となる労力
- ライセンスのコストおよびライセンスの移行
- Oracle Solaris でのオープンソースのソフトウェアのサポート

詳細は、17 ページの「Oracle Solaris にデータベースを移行する場合の考慮事項および依存関係」を参照してください。

♦ ♦ ♦ 第 2 章

# 2 開発者環境の移行

この章では、Oracle Solaris 11 への開発環境の移行について説明します。データおよびデータベースの移行についても説明します。

この章の内容は、次のとおりです。



## Oracle Solaris の開発環境

Oracle Solaris では、アプリケーション開発用のさまざまなツールを提供およびサポートしています。開発に使用可能なツールについては、『Oracle Solaris 11 でのアプリケーション開発環境の設定』を参照してください。

### Oracle Solaris でのコンパイラの使用

Oracle Solaris はオープンソースのコンパイラシステムである GNU Compiler Collection (GCC) をサポートしており、IPS パッケージとしてインストールできます。ソフトウェア開発で GCC の代わりに Oracle Solaris Studio も使用できます。Oracle Solaris Studio は、Oracle Solaris オペレーティングシステムおよび Linux オペレーティングシステムのための包括的な C、C++、および Fortran のツールスイートです。Oracle Solaris Studio の使用方法については、25 ページの「Oracle Solaris Studio を使用したアプリケーションの最適化」を参照してください。

## アプリケーション開発のための Oracle Solaris Studio の使用

Oracle Solaris Studio は、Oracle Solaris オペレーティングシステムおよび Linux オペレーティングシステムで C、C++、および Fortran のアプリケーションを作成するためのコンパイラスイートです。Oracle Solaris Studio は、開発環境をサポートする基となるオペレーティングシステムおよびハードウェアに最適化されたツールを提供します。

Oracle Solaris Studio には、次の 2 つの主要なツールのスイートが含まれています。

- コンパイラスイート – より短時間でアプリケーションを作成するのに役立ちます。このスイートには、C と C++ のコンパイラ、Fortran コンパイラ、デバッガ、および Oracle Solaris Performance Library が含まれています。
- 分析スイート – アプリケーションの可観測性が向上します。このスイートには、パフォーマンスアナライザ、コードアナライザ、およびスレッドアナライザの各ツールが含まれています。

Oracle Solaris Studio によってアプリケーションのパフォーマンスが向上し、マルチコア環境での開発が簡略化されます。Oracle Solaris Studio は、Oracle Solaris および Linux オペレーティングシステムで、本番用途に自由に利用できます。IPS パッケージと tar ファイルの両方が使用可能です。Oracle Solaris Studio 12.4 は、Oracle Technology Network (OTN) の Web サイト (`http://www.oracle.com/technetwork/server-storage/solarisstudio/downloads/index-jsp-141149.html`) からダウンロードできます。

また、Oracle Solaris Studio には、NetBeans プラットフォーム上に構築された統合開発環境 (IDE) が付属しています。IDE には、インテリジェントな言語対応のコードエディタ、コード補完、コードフォールディング、および強調表示を含むさまざまな機能があります。詳細は、第4章「Oracle Solaris でアプリケーションを開発するための便利なツールおよびヒント」を参照してください。

## アプリケーション開発のためのその他のツール

Oracle Solaris は、アプリケーション開発のための次のツールを提供しています。

- Oracle JDeveloper – Java ベースの SOA アプリケーションとユーザーインタフェースの開発を簡略化し、完全な開発ライフサイクルをサポートする統合開発環境です。詳細は、`http://www.oracle.com/java` を参照してください。
- NetBeans – オープンソースの環境であり、NetBeans IDE はエンタープライズ、Web、デスクトップ、およびモバイルの Java アプリケーションの作成をサポート

します。すべての NetBeans IDE ツールおよび機能は完全に統合されています。詳細は、`https://netbeans.org/` を参照してください。

アプリケーション開発に使用可能なその他のツールについては、『Oracle Solaris 11 でのアプリケーション開発環境の設定』の「アプリケーション開発に役立つソフトウェアのインストール」を参照してください。

# Oracle Solaris 10 アプリケーションの Oracle Solaris 11 との互換性の確認

Preflight Applications Checker ツールを使用すると、Oracle Solaris 10 上の既存のアプリケーションを Oracle Solaris 11 上で実行できるかどうかを確認できます。アプリケーションが Oracle Solaris 11 に対応しているかどうかの判断に使用することもできます。

このツールは、次の領域に関する分析を実行して、アプリケーションが Oracle Solaris 11 に対応しているかどうかを確認します。

- バイナリまたは C++ ソースで、Oracle Solaris 11 で正しく機能しない可能性のある非推奨のシステムコール、削除されたシステムコール、サポートされていないシステムコール、または不安定なシステムコールを使用しているかどうか静的分析。
- 実行中のアプリケーションで、削除、再配置、またはアップグレードされた動的ライブラリを使用しているかどうかに関する動的な DTrace ベースの分析。たとえば OpenSSL など。
- 削除または再配置されたロケールなどの構成ファイル。
- 削除または再配置されたコマンド。

Preflight Applications Checker ツールでは、指定されたアプリケーションで検討する可能性のあるすべての領域をリストした HTML 形式のレポートを生成します。詳細は、`http://www.oracle.com/technetwork/server-storage/solaris11/downloads/preflight-checker-tool-524493.html` および付録A Preflight Applications Checkerを参照してください。

Oracle Solaris 11 への移行については、『Oracle Solaris 10 から Oracle Solaris 11.3 への移行』を参照してください。

# データ移行プロセス

組織が現在のシステムと互換性のない新しいコンピュータシステムまたはデータベース管理システムを使用すると決定した場合は、データ移行が必要となります。通常、

データ移行はデータを自動的に転送する一連のカスタマイズされたプログラムまたはスクリプトによって実行されます。

データ移行には、ファイルシステム、ファイル、アプリケーション、およびデータベースのデータの移動が含まれます。格納されているデータには、エンコードされた形式、または受信側のシステムと互換性のない形式であるデータが含まれている可能性があります。Red Hat Enterprise Linux (RHEL) および Oracle Solaris は、ASCII を使用してテキストデータを格納します。

データの移行には、次のアクティビティーが伴います。

- アプリケーションデータ、スキーマ、テーブル、インデックス、および制約の移行を含む raw データの移行
- ストアドプロシージャー、データベーストリガー、SQL クエリー、関数などに関連付けられたインフラストラクチャーの移行

ファイル内に格納されているバイナリ (raw) データは、通常、SPARC プラットフォームと x86/x64 プラットフォームの間で転送できないため、データ移行時にエンディアンが問題となることがあります。

プラットフォーム間で共有されるデータを格納するアプリケーションでは、次の 2 つの方法のいずれかを使用してエンディアンの問題を処理できます。

- テキストファイルおよび文字列を使用して、アプリケーションで定義したエンディアンに中立な形式でデータを格納します
- ビッグエンディアン方式またはリトルエンディアン方式を選択し、XDR などのイネーブリングテクノロジを使用してバイトスワップを実行します

# データベースの移行

データベースをあるプラットフォームから別のプラットフォームに移動する場合は、通常、データ変換が必要になります。移行時に両方のプラットフォームが同じベンダーのデータベースをサポートしている場合は、移行が簡単になります。たとえば、Linux 上で実行されているデータベースを標準化されたファイル形式にエクスポートし、それを Oracle Solaris 上の新しいデータベースにインポートできます。移行でデータベースベンダーの変更も伴う場合は、より広範なデータ変換が必要になることがあります。

Oracle は、さまざまなデータベースから Oracle Database へのデータベースオブジェクトおよびデータの移行をサポートしています。Oracle Database は、Oracle Solaris の性能を最大限得られるように最適にチューニングされています。Oracle SQL Developer ツールを使用すると、Oracle 以外のデータベースから Oracle Database に移行できます。詳細は、17 ページの「データベース移行のための Oracle SQL Developer ツール」を参照してください。

Oracle Solaris は、次のオープンソースのデータベースをサポートしています。

- MySQL
- PostgreSQL
- SQLite
- Ingres
- Berkeley DB

Oracle Solaris 11 OS によってサポートされている専用のデータベースは、Oracle、DB2、Sybase、および Informix です。

## Oracle Solaris にデータベースを移行する場合の考慮事項および依存関係

Oracle Solaris にデータベースを移行する場合は、次の要因を考慮する必要があります。

- プラットフォームでのデータベースのサポート
- データベースバージョンの互換性
- 移行のために必要となる労力
- ライセンスのコストおよびライセンスの移行

## Oracle Database のインストール

Oracle Database をインストールする前に考慮する必要がある要因の説明については、Oracle Database のインストール前のタスク (https://docs.oracle.com/database/121/LADBI/pre_install.htm#LADBI222) を参照してください。

Oracle Solaris への Oracle Database のインストールについては、Oracle Database オンラインドキュメント 12c Release 1 (http://docs.oracle.com/database/121/nav/portal_11.htm) を参照してください。

## データベース移行のための Oracle SQL Developer ツール

Oracle SQL Developer は、データベース開発タスクを単純化することによってデータベース移行プロセスを容易にする、グラフィカルなユーザーインタフェースツール

を提供します。SQL Developer は、Oracle Database バージョン 10g、11g、および 12c をサポートしています。データベースオブジェクトの参照、作成、変更、SQL 文の実行、SQL の編集とデバッグ、および事前定義レポートのリストへのアクセスや独自のレポートの作成を行うことができます。SQL Developer は、バージョン 10g 以降の Oracle Database に接続でき、Windows、Linux、および Macintosh プラットフォーム上で実行できます。詳細は、サードパーティー製データベースから Oracle Solaris への移行 (http://docs.oracle.com/cd/E25259_01/appdev.31/e24285/migration.htm#RPTUG40000) および SQL Developer のドキュメント (https://docs.oracle.com/cd/E12151_01/index.htm) を参照してください。

♦ ♦ ♦ 第 3 章

# 3 RHEL から Oracle Solaris への移行

この章では、Red Hat Enterprise Linux (RHEL) から Oracle Solaris に移行する方法について説明します。RHEL と Oracle Solaris で使用できるテクノロジの違いについても説明します。

## RHEL と Oracle Solaris 11 のテクノロジのマッピング

次の表は、RHEL と Oracle Solaris のテクノロジの主な相違点を示しています。

**表 1** RHEL と Oracle Solaris 11 のテクノロジのマッピング

| タスク | Red Hat Enterprise Linux | Oracle Solaris 11 |
|---|---|---|
| パッケージング | RPM - パッケージのインストール、アップデート、アンインストール、およびクエリーに使用できるパッケージマネージャー。<br><br>コマンド: `rpm`、`rpmdb`、および `rpmsign` | Image Packaging System (IPS) - コマンド行 `pkg` およびグラフィカルなパッケージマネージャーを使用して、インストール、アップデート、アンインストール、クエリーなどのパッケージ操作を実行できます。<br><br>コマンド: `pkg`, `pkgsend`, `pkgrecv`, `pkgsign`, `pkgdiff`, `pkgfmt`, `pkgmogrify`、および `pkgrepo` |
| サービス | `systemd` – `systemd` システムとサービスマネージャーによって、システムサービスの起動と停止を制御します。 | サービス管理フレームワーク (SMF) – システムサービスおよびアプリケーションサービスを管理するフレームワーク。SMF は、ハードウェア障害またはソフトウェア障害が発生している間も、不可欠なシステムサービスおよびアプリケーションサービスが継続して実行できるように確保して、システムの可用性を高めます。<br><br>コマンド: `svcadm`、`svccfg`、および `svcprop` |
| ネットワークの仮想化 | RHEL は KVM でネットワーク仮 | ネットワーク仮想化は、データリンクレベルで管理されます。物理 NIC と同様に機能する VNIC を作成できます。物理 NIC デバイスへのネットワークトラ |

| タスク | Red Hat Enterprise Linux | Oracle Solaris 11 |
|---|---|---|
| | 想化をサポートしています。<br><br>コマンド: `ip`、`brctl`、および `tunctl` | フィックをルーティングするために、仮想スイッチが自動的に作成されます。<br><br>コマンド: `dladm`, `flowadm`, `dlstat`, `flowstat`, `evsadm`、および `evsstat` |
| 仮想化 | KVM – Red Hat Enterprise Linux の完全な仮想化ソリューション。<br><br>コマンド: `virsh`, `virt-clone`, `virt-convert`, `virt-image`, `virt-install`, `virt-viewer`, `virt-what`、および `virt-xml-validate` | Oracle Solaris ゾーン – 高度なアプリケーションの分離およびリソース管理を実現する、オーバーヘッドの少ないネイティブな OS 仮想化を提供します。<br><br>コマンド: `zoneadm`、`zonecfg`、`zonestat`、`zonename`、および `zone2pvhck`<br><br>Oracle Solaris は、Oracle VM Server for SPARC プラットフォームおよび Oracle Solaris カーネルゾーンもサポートしています。 |
| ファイルシステム | Ext4、XFS、LVM、btrfs、および UFS | Oracle Solaris ZFS – Oracle Solaris 11 のデフォルトのファイルシステム。最大ファイルサイズは 16E バイトで、最大ボリュームサイズは 16E バイトです。Oracle Solaris ZFS には、スナップショットとクローニング、複製解除、暗号化、圧縮、シャドウ移行、RAID などの統合されたデータサービスがあります。<br><br>コマンド: `zfs` および `zpool` |
| 暗号化 | LUKS (Linux Unified Key Setup) – RHEL はファイルシステムの暗号化のために LUKS をサポートしています。LUKS はシステムが停止している場合にのみ、暗号化されているパーティション内のデータを保護します。<br><br>コマンド: `cryptsetup` | ZFS – データセットの作成時に完全な暗号化をサポートしています。<br><br>コマンド: `zfs`<br><br>Oracle Solaris 暗号化フレームワーク - 暗号化要求を処理するアルゴリズムと PKCS #11 ライブラリの共通の格納場所を提供します。<br><br>コマンド: `cryptoadm` および `pktool` |
| モニタリング | SystemTap – RHEL の動的な計測を実行します。 | DTrace – 管理者および開発者が、使用中の本番システム上でユーザープログラムとオペレーティングシステム自体の動作を検査するために使用できる包括的な動的トレース機能。<br><br>コマンド: `dtrace` |

| タスク | Red Hat Enterprise Linux | Oracle Solaris 11 |
|---|---|---|
| | コマンド: `stap`, `staprun`, `stap-report`, `stapsh`, `stap-merge`、および `stap-prep` | すべての Oracle Premier Support 契約に含まれている Oracle Enterprise Manager Ops Center 12c は、Oracle Solaris システムおよび Linux システムの両方を含め、広範なモニタリングをさらに大きな規模で提供しています。詳細は、`http://www.oracle.com/technetwork/oem/ops-center/index.html` を参照してください。 |

# Oracle Solaris および RHEL のスレッドモデル

RHEL では、2 つの 1:1 スレッド実装が GNU C ライブラリ (`glibc`) によって提供されています。各スレッドは、カーネルスケジューリングエンティティーにマップされます。

2 つのスレッド実装は次のとおりです。

- LinuxThreads – 元の Pthread 実装。`glibc 2.4` 以降、この実装はサポートされなくなりました。
- ネイティブ POSIX スレッドライブラリ (NPTL) – 最新の Pthread 実装です。

Oracle Solaris は、MxN 実装よりも 1: 1 スレッドモデルを優先して使用します。既存のアプリケーションの基となるスレッド実装を単純化すると、再コンパイルを必要とせずにパフォーマンスと安定性が向上します。

Oracle Solaris は Pthread スレッドおよび Oracle Solaris スレッドの両方をサポートしています。新しいアプリケーション開発と移植作業には Pthread モデルを使用できます。Pthreads API を介して、100 個を超える標準の POSIX 関数を使用できます。詳細は、`pthreads(5)` のマニュアルページおよび『Multithreaded Programming Guide』を参照してください。

パフォーマンスを最大限に高めるため、多数のタスクが同時に実行されるようにアプリケーションを変更できます。Oracle Solaris では、複数スレッド、共有メモリー、非同期 I/O などの並列処理のサポートを提供しています。マルチスレッドアプリケーションプログラムは、並列性を向上させることを目的としており、ターゲットシステム上に構成されている CPU の数に関係なく記述できます。

スレッドモデルについては、Red Hat Enterprise Linux から Oracle Solaris への移植ガイド を参照してください。

# RHEL から Oracle Solaris へのサービスの移行

Oracle Solaris サービス管理機能 (SMF) ではレガシーサービスとの互換性を実現できるため、RHEL から SMF にサービスを移行できます。レガシーサービスには通常、RHEL 環境で使用される `/etc/rc*.d`、`/etc/init.d`、および `/etc/inittab` スクリプトが含まれます。レガシーサービスは以前と同様に引き続き動作させることができ、SMF でそれらのサービスを監視できます。ただし、自己修復機能、サービス再起動などの SMF のすべての利点を活用するには、スクリプトを SMF マニフェストに変換する必要があります。詳細は、22 ページの「Oracle Solaris のサービスマニフェスト」を参照してください。

Oracle Solaris 11 の SMF では、単一のフレームワークを使用して次のタスクを実行できます。

- システム全体のサービスの監視および管理
- 一貫性のある構成方法の提供
- 失敗したサービスの適切な依存関係の順序での自動的な再起動
- 失敗したサービスの識別
- root 以外のユーザーへの管理者タスクのセキュアな委任
- レガシーサービスとの互換性の維持
- バックアップ、復元などのシステム管理ジョブの自動構成

SMF については、『Oracle Solaris 11.3 でのシステムサービスの管理』を参照してください

システムの可用性を高めるには、Oracle Solaris の障害管理アーキテクチャー (FMA) を使用すると、システム問題の検出に役立ちます。詳細は、『Oracle Solaris 11.3 での障害、欠陥、およびアラートの管理』 の 「障害管理の概要」を参照してください。

## Oracle Solaris のサービスマニフェスト

サービス、サービスプロパティー、プロパティーグループ、およびインスタンスに関する情報は、SMF リポジトリに転送されるサービスマニフェストファイル `/lib/svc/manifest` に格納されます。SMF は、サービスを管理するとき、およびサービス障害の根本原因を特定するときにマニフェスト情報を使用します。サービスマニフェストでは、失敗したサービスが自動的に再起動される条件についても記述します。サービスまたはアプリケーションごとに個別のサービスマニフェストが必要です。Oracle Solaris は、デフォルトでいくつかのサービスマニフェストを提供しています。必要に応じて、これらのマニフェストをカスタマイズしたり、ほかのサービスのために独自

に記述したりすることもできます。詳細は、『Oracle Solaris 11.3 でのシステムサービスの管理』を参照してください。

## アプリケーションを SMF および FMA に移行するためのベストプラクティス

次のベストプラクティスを使用すると、SMF および FMA フレームワークへのアプリケーションの移行が容易になります。

- アプリケーションの健全性を分析してアプリケーションを再起動するカスタムスクリプトを排除します。SMF は、アプリケーションを起動、停止、および再起動するためのメソッドをカプセル化および標準化するための単純な方法を提供します。
- アプリケーションを移植およびテストのプロセスの早い段階で SMF 対応にします。障害状態を識別して、障害ツリーを作成します。検出されたエラーメッセージを確認し、FMA イベントになるかどうかを判別します。
- `.rc` およびカスタムスクリプトを SMF プロファイルに変換します。起動、停止、およびステータスチェックのメソッドのインスタンスを探します。

# Oracle Solaris でのパッケージング

Oracle Solaris 11 は、パッケージ管理に Image Packaging System (IPS) を使用しています。IPS は、パッチおよびアップデートの管理プロセスを簡略化して、オペレーティングシステムの保守の問題のリスクを軽減します。ソフトウェアのライフサイクル全体に渡って、ソフトウェアのインストール、アップデート、システムのアップグレード、およびソフトウェアパッケージの削除に対処する包括的な配布フレームワークです。IPS は、配布メカニズムとして、ネットワークからアクセス可能またはローカルで使用可能なソフトウェアリポジトリに依存しています。詳細は、『Oracle Solaris 11.3 パッケージリポジトリのコピーと作成』を参照してください。

各 IPS パッケージは、パッケージに関する情報を提供する障害管理リソース識別子 (FMRI) で表されます。たとえば、FMRI `pkg://solaris/system/library@0.5.11,5.11-0.151.0.1:20101105T004750Z` は、次の一連の情報で構成されています。

- スキーム – `pkg`
- パブリッシャー – `Oracle Solaris`
- カテゴリ – `system`
- パッケージ名 – `library`
- 4 つのコンポーネントで構成されているバージョン文字列。

  - コンポーネントバージョン - `5.11`
  - ビルドバージョン - `5.11`
  - ブランチバージョン - `151.0.1`
- ISO-8601 基本形式のタイムスタンプ: `20101105T004750Z`

パッケージングおよび配布については、『Oracle Solaris 11.3 でのImage Packaging System を使用したソフトウェアのパッケージ化と配布』を参照してください。

♦ ♦ ♦ 第 4 章

# Oracle Solaris でアプリケーションを開発するための便利なツールおよびヒント

この章では、アプリケーションを微調整するために Oracle Solaris で使用できるさまざまなツールおよびヒントについて説明します。これらのツールによってサポートされている機能を活用して、アプリケーションを開発し、Oracle Solaris 上で最適に実行できます。

## Oracle Solaris Studio を使用したアプリケーションの最適化

Oracle Solaris Studio で提供されているツールによって、Oracle の SPARC T シリーズと M シリーズのプロセッサ、Intel Xeon プロセッサ、AMD Opteron プロセッサなどの最新の CPU アーキテクチャーの処理能力を活用できます。これらのツールを使用すると、そのプラットフォームで並列および同時ソフトウェアアプリケーションをより簡単に作成できます。SPARC M シリーズシステムおよび T シリーズシステムについては、`http://www.oracle.com/technetwork/documentation/sparc-mseries-servers-252709.html` および `http://www.oracle.com/technetwork/documentation/sparc-tseries-servers-252697.html` を参照してください。

移行するアプリケーションが標準に厳格に遵守している必要がない場合は、最適化フラグを使用してよりパフォーマンスの高いバイナリを作成し、さらにパフォーマンスを向上させることができます。たとえば、Oracle Solaris Studio の `-fast` マクロは、特定のターゲットハードウェアに最適化されたバイナリを生成するためのもっとも簡単な方法です。

**注記 -** `-fast` マクロでは `-fns`オプションも暗黙的に指定されます。

このセクションでは、Oracle Solaris Studio を使用したアプリケーションの最適化について説明します。

## Oracle Solaris Studio ツール

高パフォーマンスのバイナリを生成し、検出が困難なコードの問題を分離するには、次の Oracle Solaris Studio ツールを使用できます。

- パフォーマンスアナライザ - アプリケーションパフォーマンスの分析、パフォーマンスのホットスポットの識別、および改善を要するプログラム部分の判別を行います。
- スレッドアナライザ - マルチスレッドプログラムのコードの問題を検出します。たとえば、データの競合とデッドロック状態を検出できます。
- Discover - 実行時のプログラムメモリーの割り当ておよび使用に関連するプログラミングエラーを検出します。次のタイプのプログラミングエラーを検出できます。
  - 非割り当てメモリーからの読み取り、および非割り当てメモリーへの書き込み
  - 不正なメモリーブロックの解放
  - 空きメモリーの使用
  - 割り当て済み配列範囲外のメモリーへのアクセス
  - メモリーリーク
  - 非初期化メモリーへのアクセス
- Uncover - ユーザーアプリケーションのコードカバレージを測定します。このツールのカバレージ情報は、関数、文、基本ブロック、または命令のレベルで報告されます。デフォルトでは、高パフォーマンスの `libumem` ライブラリが Oracle Solaris 11 にバンドルされています。`libumem` ライブラリは `dbx` デバッグツールと連係して、検出が困難なメモリーリークやバッファーのオーバーランを見つけます。

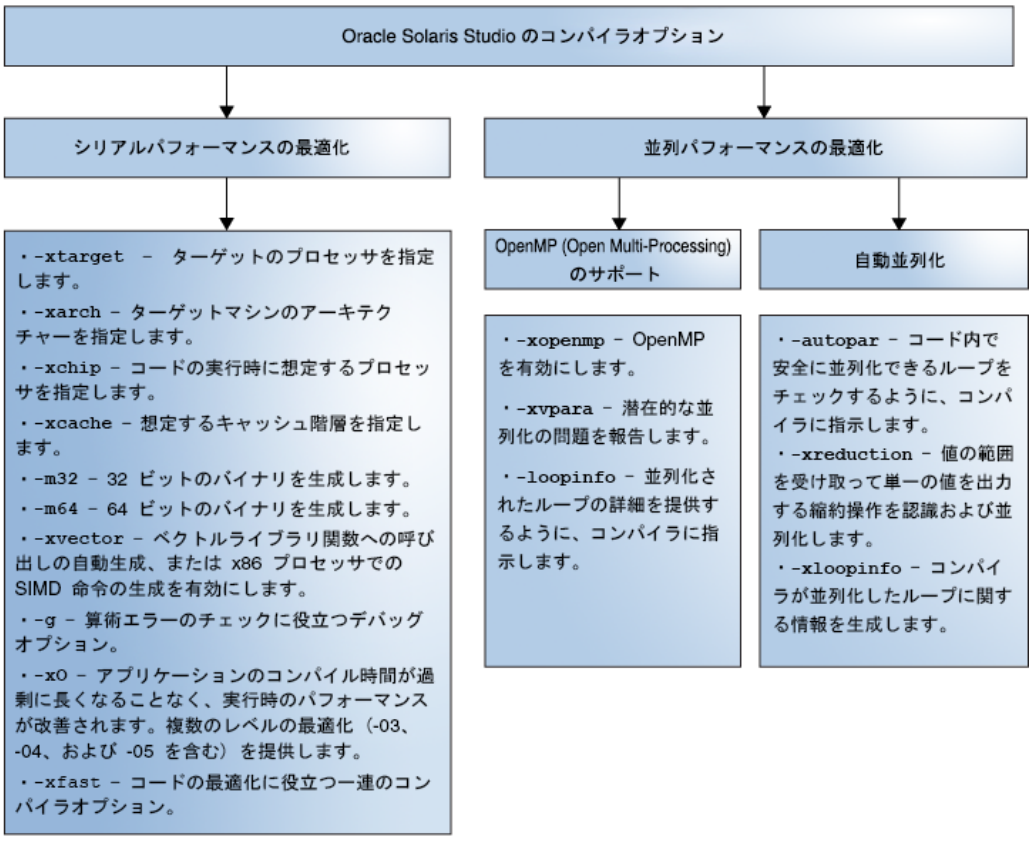
詳細は、Oracle Solaris Studio 12.4 の情報ライブラリ を参照してください。

# アプリケーションのシリアルおよび並列パフォーマンスの最適化

アプリケーションのパフォーマンスは、ハードウェア、オペレーティングシステム、およびシリアル環境と並列環境のコンテキストで表示されます。シリアル環境で実行されているアプリケーションの場合、アプリケーションがハードウェアリソースを効率的に使用するための適切なコンパイラフラグを使用することによって、パフォーマンスを最適化できます。並列環境で実行されているアプリケーションの場合、コンパイラオプションを使用して、複数コアおよび複数スレッドの実行を活用し、アプリケーションのパフォーマンスを最適化できます。自動並列化コンパイラフラグおよび OpenMP コンパイラフラグを使用すると、並列バージョンのアプリケーションをビルドおよび実行できます。

次の図は、アプリケーションのシリアルパフォーマンスおよび並列パフォーマンスを最適化するために使用する Oracle Solaris Studio のコンパイラオプションを示しています。

**図 1** Oracle Solaris Studio のコンパイラオプション




# コンパイラの最適化オプションの選択

最適化フラグは、次の 3 つの重要な特性に影響を及ぼします。

- コンパイル時間
- コンパイルされたアプリケーションの実行時間

- 最終的なバイナリで可能なデバッグアクティビティーの量

一般的に、最適化のレベルが高くなると、アプリケーションの実行が速くなり、コンパイルにかかる時間が長くなりますが、使用できるデバッグ情報は少なくなります。最適化レベルの影響はアプリケーションによって異なります。

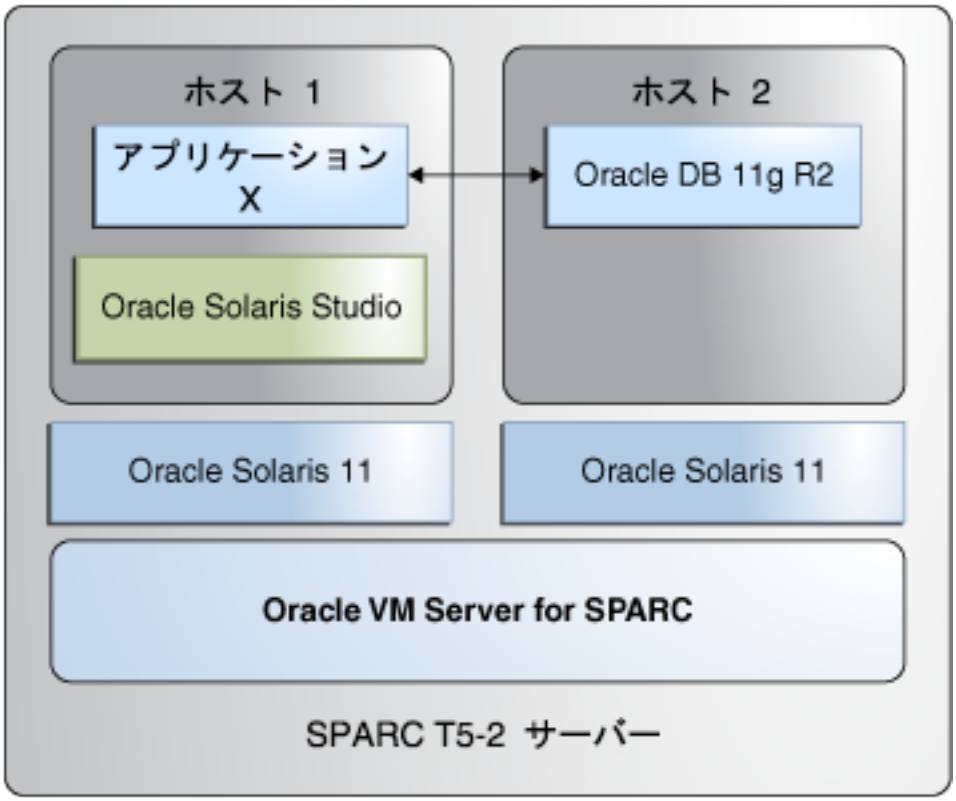
# 使用例: アプリケーションのパフォーマンスの最適化

この使用例は、Oracle Solaris Studio を使用して、アプリケーションのパフォーマンスを最適化するシナリオを示しています。この使用例では、次のアクティビティーが行われます。

- パフォーマンスアナライザツールを使用して、アプリケーションのリソース利用率を理解します
- コンパイラオプションを使用して、アプリケーションのシリアルパフォーマンスおよび並列パフォーマンスを最適化します
- フィードバックプロファイリングを使用し、アプリケーションの実行時の動作に基づいて、バイナリを最適化します

この使用例では、次の図に示すアーキテクチャーを想定しています。

**図 2** パフォーマンスの最適化の使用例のアーキテクチャー




この例のアーキテクチャーは、次のセットアップで構成されています。

- SPARC T5-2 サーバー (32 個のコアおよび 256G バイトの RAM)。
- このサーバーは、Oracle VM Server for SPARC 3.1 プラットフォームを使用してパーティション化されています。16 個のコアおよび 128G バイトの RAM を持つ 1 つの独立したパーティションが、アプリケーションの配備に使用されています。16 個のコアと 128G バイトの RAM を持つプライマリドメインが、Oracle Database 11gR2 をホストするために使用されています。
- Oracle Solaris 11 が、アプリケーションサーバーおよびデータベースサーバーの両方に構成されています。
- Oracle Solaris Studio 12.4 がアプリケーションサーバーにインストールされています。
- アプリケーション X がアプリケーションサーバーで実行されています。

# パフォーマンスアナライザの使用

Oracle Solaris Studio パフォーマンスアナライザを使用すると、アプリケーションの動作を観察し、改善を要するアプリケーション部分を判別できます。データを収集するために `collect` コマンドを使用し、パフォーマンスアナライザを起動するために `analyzer` コマンドを使用します。

次の主要な領域を分析できます。

- 大きいサイズの可変メモリー割り当て
- 実行時間の統計
- さまざまな関数についてアプリケーションで費やした時間
- キャッシュミスなどの問題を検出するためのハードウェアカウンタ

# アプリケーションを最適化するためのコンパイラオプションの使用

Oracle Solaris Studio のコンパイラで提供されているコンパイラオプションを使用して、アプリケーションを最適化できます。この使用例で使用されているコンパイラオプションを次の表に示します。

**表 2** コンパイラオプション

| コンパイラオプション | 機能 |
|---|---|
| `-DSOLARIS` | マクロシンボル Solaris をプリプロセッサに割り当てます。 |
| `-fast` | 実行時のパフォーマンスを最大化します。このオプションでは、次の個別のコンパイルが最適化されます。 |
| | `-fns` - SPARC プラットフォームで非標準浮動小数点モードを有効にします。 |
| | `-fsimple=2` - 浮動小数点の最適化の設定を選択します。 |
| | `-xarch` - ターゲットの命令セットアーキテクチャーを指定します。 |
| | `-xbuiltin=%all` - 標準ライブラリの呼び出しをさらに最適化します。 |
| | `-xcache` - オプティマイザに使用されるキャッシュプロパティーを定義します。 |
| | `-xchip` - オプティマイザに使用されるターゲットプロセッサを指定します。 |
| | `-xlibmil` - 最適化のため、選択した `libm` ライブラリルーチンをインライン化します。 |
| | `-xlibmopt` - 最適化された数学ルーチンライブラリを使用します。 |
| | `-xmemalign` - データの境界整列についてコンパイラが行う想定を制御します。 |
| | `-xO5` - 最高レベルの最適化が行われます。 |
| `-m64` | コンパイルされるバイナリオブジェクトに 64 ビットメモリーモデルを指定します。 |
| `-xtarget=T5` | 命令セットと最適化の対象とするプラットフォームを指定します。 |

| コンパイラオプション | 機能 |
|---|---|
| -xautopar | 複数プロセッサのための自動並列化を有効にします。 |
| -xdepend | 内部繰り返しデータの依存関係があるかどうかループを分析してループを再構成します (ループ交換、ループ融合、スカラー置換、および「使用されていないアレイ」の割り当ての削除を含む)。 |
| -xprefetch=auto | prefetch 命令の自動生成を有効にします。 |
| -xcode=pic32 | 位置独立コード (ラージモデル) を生成します。pic13 ほど高速でない可能性がありますが、フルレンジ対応です。-KPIC と同等です。32 ビットアーキテクチャーでは最大 2**30 個の固有の外部シンボルを、64 ビットでは 2**29 個の固有の外部シンボルをそれぞれ参照できます。 |
| -xipo=2 | 内部手続きの別名解析と、メモリーの割り当ておよび配置の最適化を実行し、キャッシュ性能を向上します。 |
| -xlinkopt=2 | オブジェクトファイル内のあらゆる最適化のほかに、結果として出力される実行可能ファイルや動的ライブラリのリンク時最適化も行うようにコンパイラに指示します。リンク時のデッドコードの除去とアドレス演算の簡素化を含む、追加のデータフロー解析を実行します。 |

# フィードバックプロファイリングの使用

Oracle Solaris Studio のフィードバックプロファイリングを使用すると、アプリケーションのバイナリを最適化できます。フィードバックプロファイリングでは、アプリケーションを 2 回 (プロファイルデータ収集のために 1 回、そのプロファイルを使用した最適なコード生成のために 1 回) ビルドします。この使用例では、次のフィードバックプロファイリングの手順を使用しています。

1. 環境変数 SUN_PROFDATA_DIR を設定して、生成されるプロファイルディレクトリの場所を指定します。

   ```
   # setenv SUN_PROFDATA_DIR /tmp/consolidate
   ```

   プロファイルディレクトリは、/tmp/consolidate フォルダに格納されます。
2. 環境変数 SUN_PROFDATA を設定して、製品のプロファイルデータを単一のファイルに記録します。

   ```
   # setenv SUN_PROFDATA singlefeedbin.profile
   ```

   プロファイルデータは、singlefeedbin.profile ファイルに格納されます。
3. -xprofile オプションを使用してバイナリをコンパイルします。

   ```
   -xprofile={collect,tcov}:xxx.profile
   ```

   -collect オプションはプログラムに関するデータ収集に役立ち、-tcov オプションはプログラムの実行後にコードカバレージを提供します。
4. リアルタイム環境を模した典型的なワークロードを使用してアプリケーションを実行します。

プログラムのこのような実行中の動作に関する情報は、ステップ 2 で作成した `.profile` ディレクトリに保存されます。

5. `profile` データを使用して、バイナリを再コンパイルします。

```
-xprofile=use:/path/to/xxx.profile
```

`-use` オプションは、プロファイルデータに基づいて次の最適化を実行します。

- コードのレイアウト
- レジスタの割り当て
- ループの変換
- 分岐の最適化
- ブロックの整理
- switch case コードの生成
- グローバル命令のスケジューリング
- 遅延スロットのスケジューリング
- 分岐の予測

# マルチスレッドアプリケーションの問題への対処

このセクションでは、Oracle Solaris のマルチスレッドアプリケーション問題に対処する方法について説明します。

## スレッドアナライザの使用

Oracle Solaris Studio のスレッドアナライザは、アプリケーション最適化のための高度なツールであり、マルチスレッドアプリケーションの妥当性を保証します。具体的には、スレッドアナライザは、マルチスレッドアプリケーションで発生することがある特殊な状況を検出、分析、およびデバッグできます。詳細は、『Oracle Solaris Studio 12.4: スレッドアナライザユーザーズガイド』を参照してください。

## 競合状態およびデッドロックの検出

競合状態およびデッドロックの問題を検出するためにソースコードを計測するには、特殊なフラグを指定してソースコードをコンパイルし、`collect -r` コマンドの制御下で実行してから、スレッドアナライザにプログラムをロードします。

1. 最初に `-xinstrument=datarace` コンパイラフラグを指定してアプリケーションをコンパイルします。行番号およびコールスタック情報が正しく返されるようにするには、`-g` フラグを設定し、最適化レベルを指定しないでください。

2. `collect -r` コマンドを発行したあと、生成されたアプリケーションコードを実行し、主要な実行時の情報を収集します。`collect -r all` オプションを使用してプログラムを実行し、処理の実行中にデータ競合およびデッドロックの検出の実験を作成します。

```
% collect -r race app params
% collect -r deadlock app params
```

3. 実験の結果をスレッドアナライザにロードして、データ競合およびデッドロック状態を識別します。

# DTrace を使用したアプリケーションのデバッグ

DTrace は、本稼働システムのカーネルおよびアプリケーションの問題をリアルタイムでトラブルシューティングする包括的な動的トレースフレームワークです。ソフトウェアの複数のレイヤー間でのパフォーマンスの問題の追跡、および予期しない動作の原因の究明に役立ちます。DTrace は、システムのどの部分でもトレースできる強力な動的トレースツールです。

DTrace の主な機能は次のとおりです。

- 任意のデータを記録するために、システムを動的に変更できます
- どのアプリケーションでも再コンパイルおよび再起動することなく使用できます
- システムがアクティブ化されているときに使用するシステムコールが少なくて済みます
- カーネルレベルおよびユーザーレベルのプログラムの両方をトレースをできます
- トレースが有効にされているときは最小限のオーバーヘッドで機能し、トレースが実行されていないときはオーバーヘッドは発生しません
- システムコールごとにプログラムの実行が停止されないため、クラッシュを心配せず本番システムで使用できます
- 本番システムで使用して、パフォーマンスのボトルネックを検出できます
- リソース使用率とアプリケーションパフォーマンスの最大化、およびリソース要件の正確な数量化が可能になります

DTrace の詳細は、『Oracle Solaris 11.3 DTrace (Dynamic Tracing) Guide』を参照してください。

# dbx を使用したアプリケーションのデバッグ

Oracle Solaris Studio の `dbx` デバッガコンポーネントは、コマンド行、IDE または独立したグラフィカルインタフェース (`dbxtool`) でコード内のバグを検出するために役立

ちます。dbx を使用して、停止したプログラムの状態を調べることもできます。dbx により、パフォーマンスやメモリー使用率のデータ収集、メモリーアクセスのモニタリング、メモリーリークの検出などのプログラムの動的な実行を完全に制御できます。dbx を使用すると、スレッド間を移動、スレッドを中断、およびスタックやロックを表示することもできます。

dbx は、C、C++、Java、または Fortran で記述されたアプリケーションのデバッグに使用できます。ただし、Java コードをデバッグする場合は特定の制限事項があります。

# Oracle Solaris のファイルシステム

Oracle Solaris 11 でサポートされているファイルシステムは次のとおりです。

- ディスクベースのファイルシステム: HSFS、PCFS、UDFS、UFS、および ZFS
- ネットワークベースのファイルシステム: NFS および SMB
- 仮想ファイルシステム: CTFS、FIFOFS、MNTFS、NAMEFS、OBJFS、SHAREFS、SPECFS、および SWAPFS
- 一時ファイルシステム (TMPFS)
- ループバックファイルシステム (LOFS)
- プロセスファイルシステム (PROCFS)

UFS は旧バージョンのファイルシステムですが、ブート可能なルートファイルシステムとしてはサポートされません。ZFS はデフォルトのルートファイルシステムです。Oracle Solaris の ZFS ファイルシステムについては、Oracle Solaris 11 ZFS テクノロジおよび『Oracle Solaris 10 から Oracle Solaris 11.3 への移行』を参照してください。

# Oracle Solaris のセキュリティーおよび特権

Oracle Solaris は、ユーザーがファイルにアクセスする方法を制御し、システムデータベースおよびシステムリソースを保護する、ネットワーク全体のセキュリティーシステムを提供しています。ネットワーク処理、暗号化機能、ユーザー権利を管理するための Trusted Extensions などの複数のセキュリティーテクノロジが組み合わされています。

Oracle Solaris の主なセキュリティー関連機能のいくつかを次に示します。

準拠チェックおよびレポート
: セキュリティーコンプライアンスのテストを管理するには、compliance コマンドを使用します。セキュリティーベンチマークおよびセキュリティーポリシーとも呼ばれるセキュリ

| | |
|---|---|
| | ティー標準を使用して、Oracle Solaris システムのコンプライアンスを評価およびレポートできます。詳細は、『Oracle Solaris 11.3 セキュリティーコンプライアンスガイド』を参照してください。 |
| 検証済みブート | 悪意を持って、または誤って変更されたクリティカルなブートおよびカーネルコンポーネントを取り込むリスクを軽減するマルウェア対策および整合性機能。この機能は、ファームウェア、ブートシステム、およびカーネルおよびカーネルモジュールの暗号化署名をチェックします。検証済みブートは、SPARC T5、SPARC M5、および SPARC M6 プラットフォームに適用されます。詳細は、『Oracle Solaris 11.3 でのシステムおよび接続されたデバイスのセキュリティー保護』 の 「ベリファイドブートの使用」を参照してください。 |
| IKEv2 のサポート | ピアシステム間の自動セキュリティーアソシエーション (SA) と鍵管理を提供します。 |
| RBAC の時間ベースおよび場所ベースのアクセス | ユーザー属性を場所で修飾できます。`usermod` コマンドおよび `rolemod` コマンドの新しい修飾子オプションを使用すると、ユーザー属性を適用するシステムまたはネットグループを示すことができます。`useradd` コマンドの新しい `access_times` キーワードを使用すると、PAM サービスにアクセスするための新しい時間ベースのポリシーを指定できます。詳細は、`usermod(1M)`、`rolemod(1M)`、および `useradd(1M)` のマニュアルページを参照してください。 |

# Oracle Solaris Trusted Extensions

Oracle Solaris Trusted Extensions は、機密レベルに従ってデータおよびアプリケーションにラベル付けできる一連の高度なセキュリティー機能です。RBAC、強制アクセス制御のラベル付け、監査、およびデバイス割り当てを含むアクセス制御モデルが装備されています。これは、データセキュリティーポリシーをデータ所有者から分離できる、Oracle Solaris のセキュアなラベルテクノロジのオプションのレイヤーです。

Trusted Extensions は、ラベルにアクセスして処理できるアプリケーションを開発するための API を提供しています。Trusted Extensions API については、『Trusted Extensions ユーザーズガイド』を参照してください。

# Oracle Solaris の認証サービス

認証は、ユーザーまたはサービスが定義済みの条件と一致するかどうかを検証するためのメカニズムです。

Oracle Solaris の認証サービスの主な機能は次のとおりです。

- Secure RPC - Diffie-Hellman 認証メカニズムによってリモート手続きを保護します。
- プラグイン可能認証モジュール - PAM モジュールを Oracle Solaris OS にインストールすることによって、新しい認証方式を追加、および認証ポリシーを変更できます。
- Simple Authentication and Security Layer - ネットワークプロトコルに認証サービスおよびセキュリティーサービスを提供します。
- Secure Shell - 暗号化ネットワークプロトコルを使用して、セキュアなデータ通信およびリモートのコマンド行からのログインを提供します。

### プラグイン可能認証モジュール

プラグイン可能認証モジュール (PAM) は、システム管理者がシステムサービスを変更せずに新しい認証サービスを追加できる一連のプラグイン可能オブジェクトです。Oracle Solaris のユーザー認証、アカウント、セッション、およびパスワード管理の機能変更に使用できます。ログイン、ssh、およびその他のシステムエントリサービスは、PAM フレームワークを使用して、すべてのログインセッションがセキュリティー保護されていることを保証します。構成ファイルを柔軟に変更できることは、ユーザーにとって特に役立つ機能です。

PAM モジュールについては、『Oracle Solaris 11 セキュリティーサービス開発ガイド』 の 第 3 章,「PAM アプリケーションおよび PAM サービスの記述」を参照してください。

PAM の RHEL 実装と Oracle Solaris 実装の相違点については、Red Hat Enterprise Linux から Oracle Solaris への移植ガイドのプラグイン可能認証モジュールに関するセクションを参照してください。

## Oracle Solaris 暗号化フレームワーク

Oracle Solaris 暗号化フレームワークは、カーネルレベルおよびユーザーレベルのコンシューマに一連の暗号化サービスを提供します。Oracle Solaris は、PAM、GSS-API、SASL、PKCS#11 などの業界標準のインタフェースに基づくネットワークセキュリティーを提供しています。暗号化フレームワークは、Oracle Solaris の暗号化サービスの主力になるものです。このフレームワークは、暗号化サービスのコンシューマおよびプロバイダに対応する標準の PKCS #11 インタフェースを提供します。

このフレームワークは 2 つの部分で構成されています。

- ユーザーレベルのアプリケーションのためのユーザー暗号化フレームワーク

- カーネルレベルモジュールのためのカーネル暗号化フレームワーク

Oracle Solaris 暗号化フレームワークの主な要素は次のとおりです。

- `libpkcs11.so` ライブラリ - このフレームワークは、RSA Security Inc. PKCS#11 Cryptographic Token Interface (Cryptoki) 経由でのアクセスを提供します。アプリケーションは、`libpkcs11.so` ライブラリにリンクしている必要があります。
- `pkcs11_softtoken.so` 共有オブジェクト - Oracle によって提供されているユーザーレベルの暗号化メカニズムが含まれています。
- `pkcs11_kernel.so` 共有オブジェクト - カーネルレベルの暗号化メカニズムへのアクセスに使用します。これは、カーネルのサービスプロバイダインタフェースにプラグインされた暗号化サービスに PKCS #11 のユーザーインタフェースを提供します。
- プラグイン可能インタフェース - プラグイン可能インタフェースは、Oracle およびサードパーティーの開発者が提供する、PKCS #11 暗号化サービスのためのサービスプロバイダインタフェース (SPI) です。プロバイダは、ハードウェアまたはソフトウェアを介して使用可能な暗号化サービスを使用して実装されるユーザーレベルのライブラリです。
- スケジューラおよびロードバランサ - 暗号化要求の効率的な負荷分散およびディスパッチが可能になります。
- カーネルプログラマインタフェース - カーネルレベルのコンシューマに暗号化サービスへのアクセスを提供します。
- サービスプロバイダインタフェース - ハードウェアおよびソフトウェアに実装されているカーネルレベルの暗号化サービスのプロバイダによって使用されます。
- ハードウェアおよびソフトウェアの暗号化プロバイダ - ソフトウェアアルゴリズム、ハードウェアアクセラレータボード、またはオンチップの暗号化機能を使用するカーネルレベルの暗号化サービス。
- カーネル暗号化フレームワークデーモン - 暗号化操作のためのシステムリソースを管理するプライベートなデーモン。このデーモンも暗号化プロバイダを検証します。
- モジュール検証ライブラリ - 暗号化フレームワークがインポートするすべてのバイナリの整合性と信頼性を検証するプライベートライブラリ。
- `elfsign` - サードパーティーの暗号化サービスプロバイダが Oracle に証明書を要求するために提供されているユーティリティー。
- `cryptoadm` - セキュリティーポリシーに応じた暗号化メカニズムの有効化および無効化など、暗号化サービスを管理するためのユーザーレベルのコマンド。

Oracle Solaris 暗号化フレームワークにプラグインできるアプリケーションのタイプは次の 4 つです。

- ユーザーレベルのコンシューマ
- ユーザーレベルのプロバイダ
- カーネルレベルのコンシューマ

- カーネルレベルのプロバイダ

Oracle Solaris 鍵管理フレームワークは、公開鍵インフラストラクチャー (PKI) オブジェクトを管理するツールおよびプログラミングインタフェースを提供しています。

UltraSPARC T1、UltraSPARC T2、および UltraSPARC T2+ のオンチップ暗号化アクセラレーションなどの Oracle Solaris サーバーのオンボードの暗号化によって、コプロセッサカードや電力消費の大きいアドオンのコンポーネントを追加せずに済みます。RHEL と Oracle Solaris で異なる関数呼び出しについては、Red Hat Enterprise Linux から Oracle Solaris への移植ガイドを参照してください。

Oracle Solaris 暗号化フレームワークの詳細は、『Oracle Solaris 11 セキュリティーサービス開発ガイド』 の 第 9 章,「ユーザーレベルの暗号化アプリケーションの記述」を参照してください。

# Oracle Solaris の高可用性

高可用性 (HA) により、システム、アプリケーション、またはデータベースが使用可能で、継続してアクセスでき、サービスが失われることなく、必要な期間中実行されることが保証されます。

Oracle Solaris は、アプリケーションとデータベースの可用性を提供しています。アプリケーションの高可用性は Oracle Solaris Cluster によって提供されており、負荷分散、障害検出、および障害回復によってアプリケーションやシステムの高可用性が維持されます。Oracle Solaris Cluster は、広範な高可用性フレームワークを提供します。Oracle Solaris は、従来の環境および仮想環境内において、フルレンジで単一システムおよびマルチシステムの高可用性を提供しています。

Oracle Solaris Cluster ソフトウェアの高可用性フレームワークは、ノード障害を検出し、アクションを再開できるネットワーク内の別のノードにリソースを移行します。

Oracle Solaris Cluster は、ゾーンを追跡、維持、およびモニターすることもできます。障害が発生したときにゾーンを保護するフェイルオーバーゾーン機能も提供しています。フェイルオーバーゾーンは、ゾーンの HA エージェントによってモニターされます。HA Oracle Agent ソフトウェアは、Oracle Solaris Cluster ノード上の Oracle Database のアクティビティーを制御します。このエージェントは、ノードを起動、停止および検証することによって障害検査を実行します。HA エージェントがアクティブなセッションのないデータベース上に構成されている場合は、HA Oracle Agent がテストトランザクションを開始します。HA Oracle Agent からのリターンエラーコードは、その場所の特別なアクションファイルに対して検証されます。

Oracle アプリケーションサーバーは、プロセスの異常終了の検出と自動再起動、クラスタ化、サーバーの負荷分散、ロールとパッチ、バックアップと回復などのテクノロジを使用してローカルおよびエンタープライズの高可用性を提供しています。Oracle

Clusterware ソフトウェアは、アプリケーションを管理するためのインフラストラクチャーを提供します。

Oracle Solaris Cluster には、データサービスの作成を自動化する、アプリケーションの高可用性のためのエージェントビルダーツールが含まれています。開発者は、作成されるアプリケーションおよびデータサービスに関する情報 (スケーラブルなエージェントまたはフェイルオーバーエージェントにするかどうか、サービスがネットワーク対応であるかどうか、アプリケーションを起動および停止するコマンドなど) をエージェントビルダーに指定します。エージェントビルダーは、ソースコードと実行可能コード (C または Korn シェル)、カスタマイズされたリソースタイプ登録 (RTR) ファイル、および配布用の Oracle Solaris パッケージを含むデータサービスを生成します。

アプリケーションサーバーの高可用性については、`https://docs.oracle.com/cd/E12839_01/core.1111/e10106.pdf` を参照してください。

Oracle Database は、計画停止時間および計画外の停止時間を最小限にすることによって、高可用性を提供します。Oracle Database では、Oracle Real Application Clusters (RAC) および Oracle Clusterware ソフトウェアを使用して高可用性を実現しています。これらは連係して、複数のサーバーが単一サーバーとして動作するようにバインドします。Oracle RAC および Oracle Clusterware ソフトウェアのいくつかの利点を次に示します。

- コンピュータおよびインスタンスの障害を許容し、障害からすばやく回復する機能
- Oracle Clusterware のアップグレード、パッチセット、および暫定的なパッチをロール方式で適用する機能
- Oracle Universal Connection Pool (UCP)、高速接続フェイルオーバー、および FAN イベントを使用している Java アプリケーションの接続障害の迅速な自動検出、および終了した接続の削除
- データベース機能とクラスタ機能を統合した包括的な管理
- 障害が発生した Oracle プロセスの自動的な再起動
- Oracle Virtual IP (VIP) の自動的な管理、およびノード障害時におけるクラスタ内の別のノードへのフェイルオーバー

Oracle Database の高可用性については、`https://www.oracle.com/database/high-availability/index.html`を参照してください。

Oracle Solaris Cluster については、Oracle Solaris Cluster の製品ドキュメント を参照してください。

# Oracle Solaris でのネットワーク仮想化

ネットワーク仮想化とは、ハードウェアのネットワークリソースとソフトウェアのネットワークリソースを 1 つの管理単位に結合するプロセスのことです。この 1 つ

の管理単位を、仮想ネットワークと呼びます。Oracle Solaris では、仮想ネットワークインタフェースカード (VNIC) と etherstub が仮想ネットワークの基本的なコンポーネントです。物理データリンク上に VNIC を作成できます。構成された VNIC は、物理 NIC のように動作します。etherstub は、Oracle Solaris ネットワークスタックのデータリンク層 (L2) に構成される擬似 Ethernet NIC です。

Oracle Solaris 11.2 リリース以降では、1 つ以上の計算ノードにまたがる仮想スイッチを作成および管理できる Oracle Solaris のエラスティック仮想スイッチ (EVS) 機能を使用できます。ネットワーク仮想化および EVS については、『Oracle Solaris 11.3 での仮想ネットワークとネットワークリソースの管理』を参照してください。

# Oracle Solaris Remote Lab

Oracle Solaris Remote Lab (OSRL) は、クラウドベースのセルフサービスラボであり、Oracle パートナはリモートからアクセスできる環境にアクセスし、Oracle Solaris 11 上のアプリケーションを検証および認証できます。

Oracle のパートナは、SPARC プロセッサおよび x86 プロセッサの組み合わせで最大 5 個の仮想マシン (VM) を割り当てることができます。各パートナの VM は、すべてのネットワークトラフィックを分離するプライベートネットワークで、NFS マウントされたファイルシステムを共有します。

詳細は、Oracle Solaris Remote Lab のしくみの記事を参照してください。

♦ ♦ ♦ 付 録 A

# Preflight Applications Checker

Oracle Solaris Preflight Applications Checker "Preflight Checker" を使用すると、Oracle Solaris 10 アプリケーションを Oracle Solaris 11 に移行するときの潜在的な問題を特定できます。このツールは、Oracle Solaris 10 でソースコード、静的バイナリ、および実行時分析モジュールを組み合わせて実行して、既存のアプリケーションをテストし、アプリケーションを Oracle Solaris 11 で正常に実行するために必要なコード変更を提案します。

**注記 -** Oracle Preflight Applications Checker は、開発システムまたはテストシステムでのみ使用し、本番システムでは使用しないでください。

## Preflight Checker について

Oracle Solaris Preflight Applications Checker は、潜在的な問題を見つけ、Oracle Solaris 11 システムにアプリケーションを実装する適切な方法を提案します。Preflight Applications Checker には、カーネルモジュールの潜在的な問題点を見つける際に役立つ Oracle Solaris Preflight Kernel Checker も含まれています。これには、ソースコードアナライザモジュールおよびバイナリアナライザモジュールが含まれています。

これらのツールの主な目的は、アプリケーションが対応しているかどうかを開発者が移行前にテストできるようにすることです。

詳細は、Oracle Solaris 保証プログラムを参照してください。

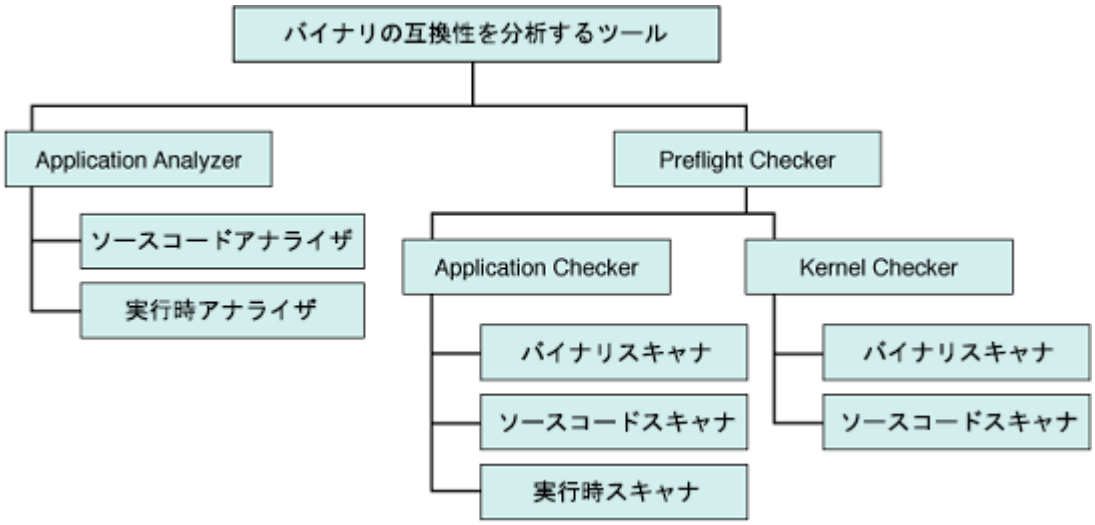
## Preflight Checker のモジュール

Preflight Applications Checker には、次の 3 つのメインモジュールがあります。

- バイナリアナライザ

- ソースコードアナライザ
- 実行時アナライザ

次の図は、バイナリの互換性を解析するために使用できるモジュールの概略を示します。




## バイナリアナライザモジュール

バイナリアナライザモジュール (ELF アナライザとも呼ばれます) は、アプリケーションのバイナリからすべてのバインディングおよび関連付けられているシンボル情報を抽出します。アプリケーションの実行のために `LD_LIBRARY_PATH` 環境変数を設定する必要がある場合、またはシステムのデフォルトの実行時リンク環境が `crle` または `LD_CONFIG` によって変更されている場合は、`LD_LIBRARY_PATH` に値を指定できます。

Preflight Checker は、`LD_LIBRARY_PATH` 値を使用して、実行時にアプリケーションで使用されるライブラリを判別します。この情報により、シンボルバインディングから Oracle Solaris システム共有ライブラリへの対応付けが容易になります。バイナリ分析の実行中に、`LD_LIBRARY_PATH` 変数を設定する必要のないアプリケーションの場合、この値を空白のままにします。`lib` パス `/usr/lib/64` (`/usr/lib` または `/lib/64`) は、`crle` と同様にバイナリアナライザモジュールで自動的に使用されます。

Preflight Checker には、Oracle Solaris 10 から Oracle Solaris 11 に移行するときの潜在的な問題のデータを含む、組み込みのデータベースがあります。このデータベースは、アプリケーションで使用されているシステムシンボルごとにアクセスされます。データベース検索の出力に基づき、このツールは該当する潜在的な問題、および問題を解決するために考えられる回避方法や推奨事項を報告します。

## ソースコードアナライザモジュール

ソースコードアナライザモジュールは、Oracle Solaris で削除、名前変更、変更されたか、または非推奨となったライブラリ関数が使用されていないかどうかをチェックします。また、このモジュールは C および C++ のソースコードの問題をチェックし、シェルスクリプトの潜在的な問題を検出します。

---

**注記 -** デフォルトのシステムシェル `/bin/sh` および Korn シェル `/bin/ksh` は、Oracle Solaris 11 では `ksh93` の実装によって置き換えられています。

---

シェルスキャナモジュールによって潜在的な問題の特定が容易になり、警告メッセージを報告できます。さらにこのモジュールは、Oracle Solaris 11 で使用不可能なファイルの使用についても報告します。

シェルスキャナモジュールはスキャン時に、ターゲットアプリケーションのディレクトリの内側で使用可能なシェルスクリプト内で検索することによってシェル変数を解決しようとします。適切な結果を得るには、ターゲットスキャンディレクトリの外側に環境変数を設定し、アプリケーション環境を設定するためのスクリプトのコンマ区切りのリストを `init_script` フィールドに設定します。次に例を示します。

```
init_script = /export/home/user/.profile,/export/home/user/.bashrc,/export/home/user/
setenv.sh
```

---

**注記 -** シェルスクリプトの静的分析中に実行時環境の情報を使用できない場合、シェルスキャナモジュールから間違った検出結果が生成されることがあります。

---

## 実行時アナライザモジュール

実行時アナライザモジュールは、既存の Oracle Solaris 10 テスト環境で実行可能であり、潜在的な問題を報告します。Preflight Checker のこのモジュールでは、DTrace を使用してプロセスの詳細を調べます。このモジュールは、Oracle Solaris 10 で使用可能なほかのシステムツールを呼び出すこともあります。

次のいずれかのスコープで実行するように実行時アナライザモジュールを構成できます。

- システム全体
- 特定のゾーンの内側で実行されているすべてのプロセス
- 特定の UID を持つすべてのプロセス
- 特定の実行可能ファイル名に対するすべての実行中のプロセス
- 特定の PID

- ユーザーが指定したカスタムコマンド

**注記 -** 実行時アナライザモジュールは、ユーザーが開始したプロセスのみを分析し、システムプロセスを無視します。

フィルタで除去されたプロセスのリストは、編集可能な構成ファイル *tool-base-dir*`/conf/SystemProcs.conf` に保持されます。

実行時アナライザモジュールは、次のような場合に既知の問題を報告できます。

- 削除、名前変更、再配置されたライブラリ、スクリプト、コマンド、およびデバイスまたはその他のシステム提供ファイルの呼び出し。
- 非推奨の非公開型関数や廃止された関数、API、シンボルの使用または呼び出し。
- 削除、名前変更、または移動されたライブラリの `dlopen()`。
- 非公開インタフェースの使用または呼び出し。
- 古い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ーまたはパッケージの使用 (たとえば、削除されたファイルへのハードコードされたパス)。
- 最近の Oracle Solaris OS リリースでアップグレードまたは置換されたファイルまたはライブラリの使用。

## Preflight Checker を使用する場合の考慮事項

このアプリケーションは Oracle Solaris 依存関係が記録されていない共有オブジェクトにアクセスしているため、`dlopen()` 関数または `mmap()` 関数を使用している場合は、静的バイナリアナライザおよびソースコードアナライザが共有オブジェクトの場所にアクセスできません。ただし、実行時アナライザは、`dlopen()` 関数を使用してロードされたライブラリ、および `mmap()` を使用してメモリーにロードされたファイルにアクセスできます。詳細は、`dlopen(3C)` および `mmap(9E)` のマニュアルページを参照してください。

バイナリアナライザおよびソースコードアナライザが検査できないその他のオブジェクトは次のとおりです。

- 読み取り/実行アクセス権が設定されていない実行可能ファイル
- ELF 以外の実行可能ファイル

Preflight Checker を使用するときは、次のガイドラインに留意してください。

- Preflight Checker の実行時は、アプリケーション環境の詳細を正しく指定する必要があります。環境が異なると、潜在的なバイナリの非互換性や Oracle Solaris ライブラリ内のインタフェースへの未解決参照が間違って検出される可能性があります。たとえば、`LD_LIBRARY_PATH` が正しく設定されていない場合、バインドされていないシンボルが Preflight Checker から追加で報告されます。

- Preflight Checker から正確に報告されるように、Oracle Solaris Studio 12.4 の `-z defs` リンクオプションを使用して、コンパイル時に動的ライブラリの依存関係を記録します。このオプションにより、オブジェクトが自己完結型となり、さらに詳しく定義されます。リンクの終了時に未定義のシンボルが残っている場合は、致命的なエラーが発生します。このオプションは、実行可能オブジェクトを作成する場合のデフォルトの設定です。
- 実行時アナライザモジュールは、DTrace を使用してプロセスの詳細を調べるので、Oracle Solaris 10 上で使用可能なその他のシステムツールを呼び出すことがあります。システム全体のパフォーマンスや応答性に対する実行時アナライザのオーバーヘッドを制限するには、スコープをできるかぎり狭いレベルに制限します。たとえば、ターゲットとなるアプリケーションの特定の PID がわかっている場合は、システム全体の実行時解析は実行しないでください。

# Preflight Checker レポートの理解

スキャンの対応する各段階で、Oracle Solaris Preflight Applications Checker から出力されるレポートには、ターゲットとなった分析済みファイル (バイナリ分析およびソースコード分析の場合) やプロセス (Runtime 分析の場合) が一覧表示されます。

レポートの先頭に、検査されたバイナリの数および見つかった潜在的なバイナリ安定性の問題の数を示すサマリーが表示されます。サマリーレポートは *tool-install-dir*`/reports` からも参照できます。デフォルトでは、レポートは HTML 形式で作成されます。

レポートには、潜在的な問題を解決するために実行するアクションの推奨事項も含まれています。スキャンの実行後に Preflight Checker によって報告される一般的な問題には、次のものがあります。

- プライベートシンボルの使用 - このシンボルが現在のリリースで変更されていない場合、アプリケーションはまだ実行可能です。ただし、このシンボルが今後も使用されるという保証はなく、問題が発生する可能性があります。このようなプライベートなシンボルの使用は中止してください。
- 静的リンク - アプリケーションが `libc.a` を使用した静的リンクでビルドされている場合、アプリケーションが参照するカーネルインタフェースは `libc.a` アーカイブから抽出され、アプリケーションの一部になります。このアプリケーションは、使用したカーネルインタフェースと同期しているカーネルでのみ実行できます。アプリケーションは共有バージョンのシステム提供のライブラリとリンクする必要があるため、カーネルインタフェースが変更されると、アプリケーションが動作しなくなることがあります。
- 降格されたシンボルまたは非推奨のシンボル - 非推奨のシンボルの使用は中止してください。
- バインディングが見つかりません - バイナリオブジェクトが自身が依存するすべてのライブラリを見つけられるように、`LD_LIBRARY_PATH` などの環境変数をチェッ

クして正しく設定されていることを確認します。バインドされていないシンボルが生じる可能性のある問題への対処方法については、対応する API のマニュアルページを参照してください。

- 廃止されたライブラリの使用 - 実施可能になりしだい、これらのシンボルの使用を中止します。

## Preflight Checker で使用されるデフォルト

Preflight Checker のインストールフェーズ、または解析の開始時にカスタム情報を入力しない場合、このツールは次のデフォルト値を使用します。

- LD_LIBRARY_PATH = ""
- レポート = *tool-install-dir*/`reports`
- レポートの形式 = `html`
- PATH
  - SPARC の場合、PATH = `"/sbin:/bin:/usr/bin:/usr/bin/sparcv9:/bin/sparcv9"`
  - x64 の場合、PATH = `"/sbin:/bin:/usr/bin:/usr/bin/amd64:/bin/amd64"`
- 定義されているマクロ = ""
- システムのインクルードディレクトリ = `"/usr/include"`

# Preflight Checker のインストール

Preflight Checker は、Oracle Solaris 10 以降のシステムでインストールおよび実行できます。

## ▼ Preflight Checker のインストール方法

1. **Preflight Checker 用のディレクトリを作成します。**

   ```
   # mkdir /export/home/preflightcheck/app1
   ```

2. **Oracle Solaris Preflight Applications Checker のダウンロードページから、Oracle Solaris Preflight Checker のインストール可能 zip ファイルをダウンロードします。**

   - **SPARC システムの場合は、`PreflightCheckerTool-v11-2-0-SPARC.zip` をダウンロードします。**

- **X86 システムの場合は、`PreflightCheckerTool-v11-2-0-X86.zip` をダウンロードします。**

3. **ダウンロードしたファイルを新しいディレクトリに解凍します。**
   次に例を示します。

```
# unzip PreflightCheckerTool-v11-2-0-SPARC.zip
```

4. **セットアップスクリプトを実行して、アプリケーションをインストールし、最終的なレポートを保存する場所を指定します。**

```
# cd /export/home/preflightcheck/app1/scripts
# chmod +x setup.sh
# ./setup.sh
```

   インストール時に最終的なレポートの保存場所を指定しない場合、最終的なレポートはデフォルトディレクトリ `/export/home/preflightcheck/app1/reports` に保存されます。

   スクリプトによってツールが展開され、インストールされます。

5. **アプリケーションと同時に `Docs` サブディレクトリにダウンロードされたドキュメントを参照してください。**

# Preflight Checker の使用

バイナリアナライザ、ソースコードアナライザ、または実行時アナライザモジュールは、Oracle Solaris 10 システムで実行されているチェック対象のバイナリやソースコード、またはプロセスを含むディレクトリを示すことによって実行します。

Preflight Checker では、Oracle Solaris 10 上で実行中のアプリケーションプロセス、アプリケーションバイナリ、およびアプリケーションのソースコードを分析して、Oracle Solaris 11 上でアプリケーションを実行した場合の既知の潜在的な問題を報告できます。

Preflight Checker をインストールおよび実行するための詳細な手順は、ツールのインストールディレクトリの `docs` サブディレクトリにある `README` ファイルに記載されています。

## コマンド行での Preflight Checker アプリケーションの実行

Preflight Checker のインストール後は、次のコマンドを使用して起動できます。

```
% preflightchecker.sh [options] file-or-directory
```

Preflight Checker (`preflightchecker.sh`) の実行に使用できるオプションをコマンド行で一覧表示するには、`-h` オプションを使用します。

```
% preflightchecker.sh –h
```

静的バイナリの分析中、Preflight Checker はソース内でオブジェクトファイルを特定し、Oracle Solaris 11 では機能しない可能性がある非推奨の API、サポートされていない API、および不安定な API が使用されていないかどうかをチェックします。

# GUI からの Preflight Checker モジュールの実行

このセクションでは、グラフィカルユーザーインタフェース (GUI) から Preflight Checker で使用できるさまざまなモジュールを使用する方法について説明します。

## ▼ GUI からバイナリスキャナを使用する方法

1. **Preflight Checker の GUI を起動します。**

   ```
   # tool-install-dir/bin/preflightcheckerGUI.sh
   ```

   アプリケーション名やバージョンなどのアプリケーション情報を指定します。

   「詳細」ボタンをクリックすると、データベースバージョンなどのアプリケーションの詳細も追加できます。「次へ」ボタンをクリックして、次のウィンドウに移動します。

2. **「Application Checker」を選択します。**

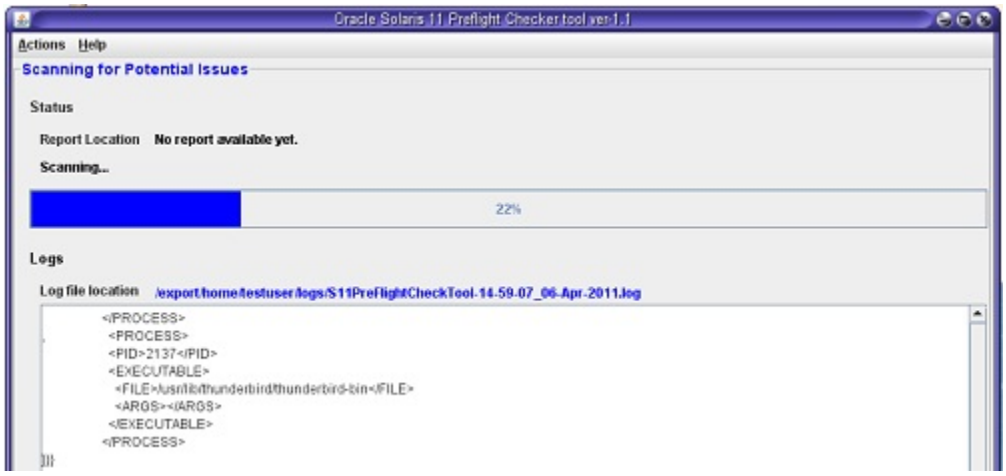
3. **「バイナリスキャナ」オプションを選択します。**

4. **「サマリー」画面で、アプリケーション名、バージョン、パス、`LD_LIBRARY_PATH` などのアプリケーションの詳細を指定します。**

5. **「ディレクトリを追加」ボタンをクリックして、スキャンするアプリケーションの場所を追加します。**

6. **「ディレクトリを指定」ボタンをクリックして、スキャンレポートを保存するフォルダを選択します。**

7. **「分析を開始」ボタンをクリックしてバイナリスキャンを開始します。**




バイナリスキャンが完了すると、スキャナによってレポートが生成されます。

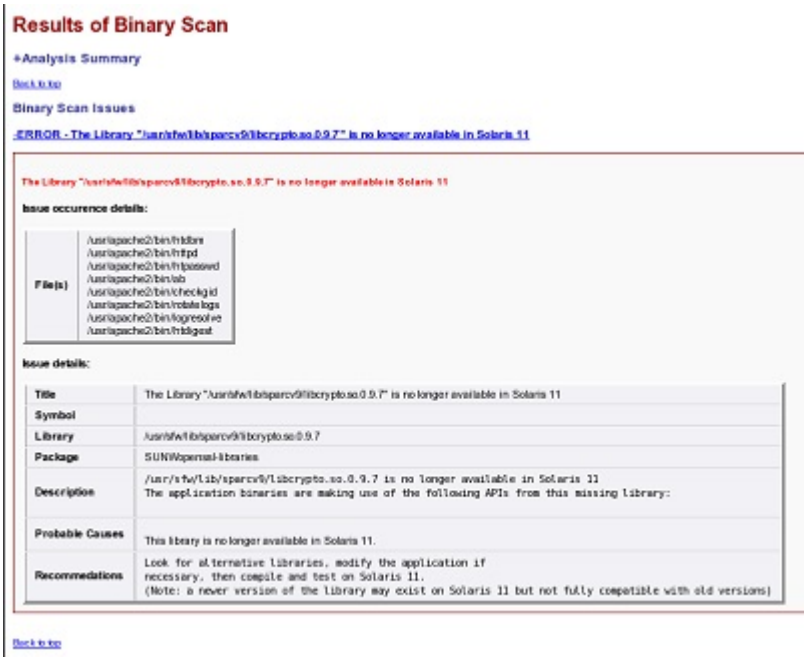
8. **(オプション) テストを再度実行するには、S11preflightcheck の再実行ボタンをクリックします。**

   変更していない場合は、現在の値が使用されます。

9. **レポートの場所をクリックしてレポートを開きます。**


10. **目次の分析サマリーのリンクをクリックして、スキャンの分析を表示します。**


11. **エラーまたは警告の詳細を表示するには、そのリンクをクリックします。**
    次の例は、問題の説明の例を示しています。



## ▼ GUI からソースコードスキャナを使用する方法

1. **Preflight Checker の GUI を起動します。**

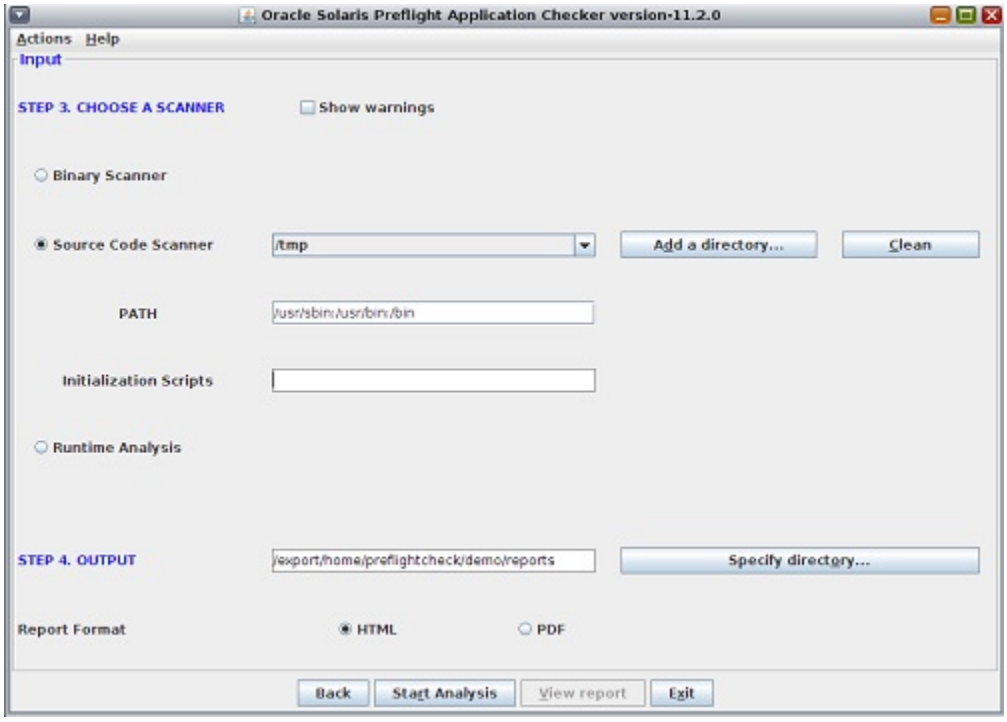
   # *tool-install-dir*`/bin/preflightcheckerGUI.sh`

2. **「Application Checker」を選択します。**

3. **「ソースコードスキャナ」オプションを選択します。**

4. **「ディレクトリを追加」ボタンをクリックして、スキャンするアプリケーションの場所を追加します。**

   「PATH」フィールドは、システムの実行可能ファイルおよびユーザーが作成した実行可能ファイルを探す場所を示しています。図に示すように、`/usr/sbin`、`/usr/bin`、および `/bin` がすでに入力されています。PATH 値にさらに場所を追加するには、コロン (`:`) で区切って指定します。

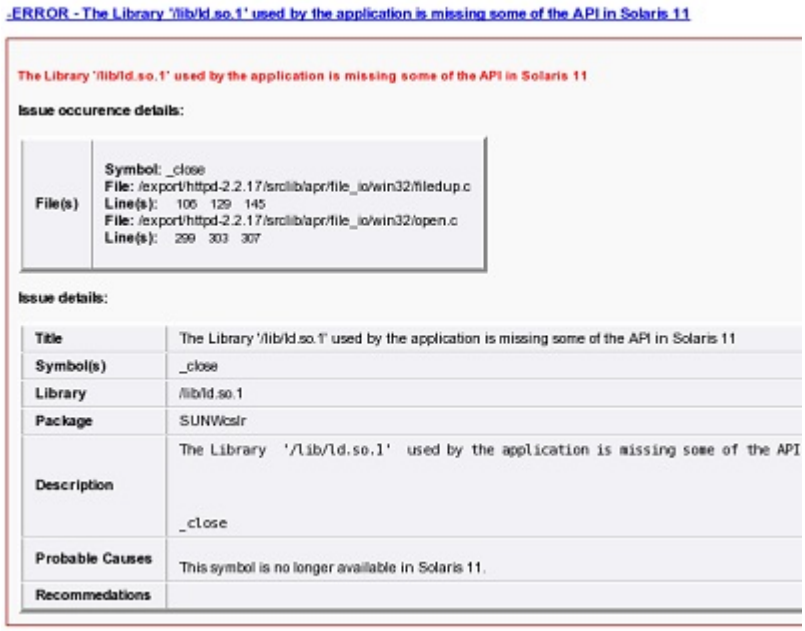



5. **「ディレクトリを指定」ボタンをクリックして、スキャンレポートを保存するフォルダを選択します。**

6. **「分析を開始」ボタンをクリックしてスキャンを開始します。**

   ソースコードスキャンが完了すると、スキャナによってレポートが生成されます。

7. **(オプション) テストを再度実行するには、S11preflightcheck の再実行ボタンをクリックします。**
   変更していない場合は、現在の値が使用されます。
8. **レポートの場所をクリックしてレポートを開きます。**
9. **目次の分析サマリーのリンクをクリックして、スキャンの分析を表示します。**
10. **エラーまたは警告の詳細を表示するには、そのリンクをクリックします。**
    次の例は、問題の説明の例を示しています。




## GUI からの実行時アナライザの使用

このセクションでは、実行時アナライザを使用するための前提条件、および GUI から実行時アナライザを起動する方法について説明します。

### ▼ 実行時アナライザのパラメータを設定する方法

**始める前に** 次の Oracle Solaris の権限を持っている必要があります。

- `basic`

- dtrace_user
- dtrace_proc
- dtrace_kernel

**注意 -** proc_owner 権限はユーザーに付与しないでください。この権限をユーザーに追加すると、そのユーザーはシステム上のほかのすべてのユーザーが所有するプロセスから情報を収集できるようになります。

1. **root として、パラメータを設定するスクリプトを実行します。**

   # **/export/home/preflightcheck/app1/scripts/setPriv.sh** *username*

2. **/etc/system ファイルに次の行を追加して、DTrace 値を設定します。**

   ```
   set dtrace:dtrace_dof_maxsize=0x800000
   ```

3. **/kernel/drv/fasttrap.conf ファイルに次の行を追加して、fasttrap 値を設定します。**

   ```
   fasttrap-max-probes=2500000;
   fasttrap-hash-size=65535;
   ```

4. **カーネルパラメータを設定したら、システムをリブートします。**

## ▼ GUI から実行時アナライザを実行する方法

1. **Preflight Checker の GUI を起動します。**

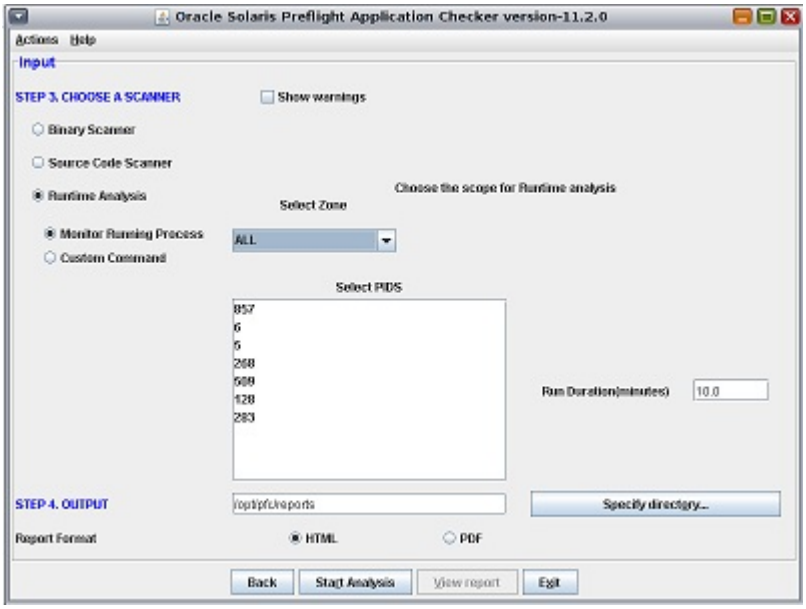
   # *tool-install-dir***/bin/preflightcheckerGUI.sh**

2. **「Application Checker」を選択します。**

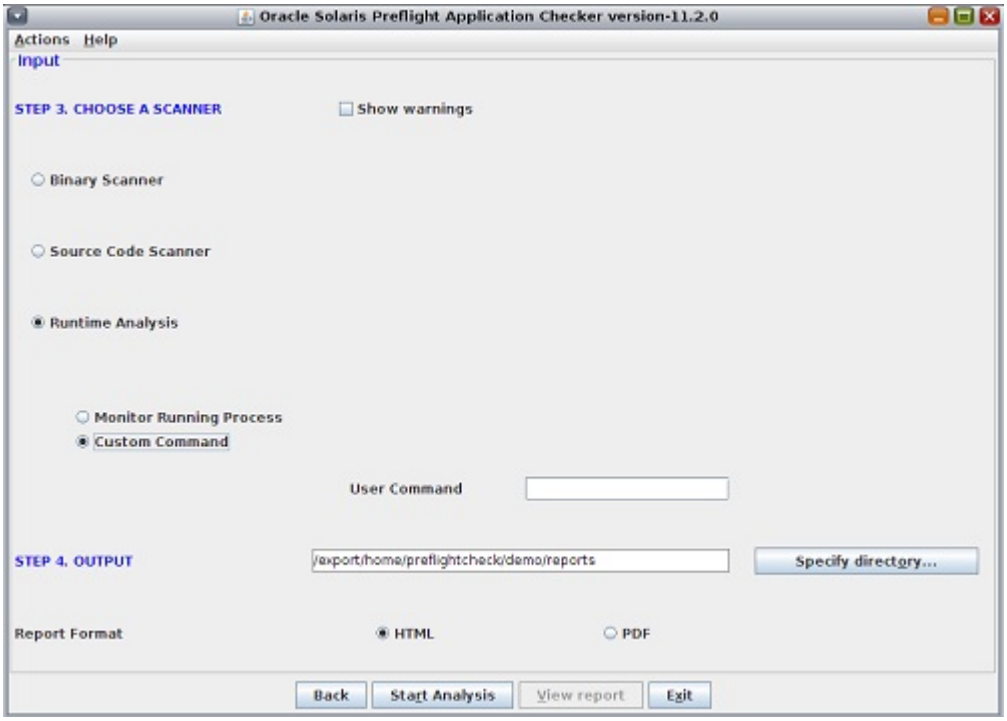
3. **「実行時分析」オプションを選択します。**

4. **分析のスコープを設定します。**

   - **特定の実行中プロセスをモニターするには、「実行中プロセスをモニター」オプションを選択して、リストからターゲットの PID を選択します。**




- ユーザーコマンドを解析するには、「カスタムコマンド」オプションを選択し、プロセスをモニターするためアプリケーションを起動する完全なコマンドを入力します。




5. **(オプション) Preflight Checker でアプリケーションをスキャンする時間を設定するには、「実行時間」フィールドを使用します。**

   Preflight Checker が指定された時間に実行され、スキャンに基づいてレポートが生成されます。

6. **「ディレクトリを指定」ボタンをクリックして、スキャンレポートを保存するフォルダを選択します。**

7. **「分析を開始」ボタンをクリックしてスキャンを開始します。**

8. **(オプション) テストを再度実行するには、S11preflightcheck の再実行ボタンをクリックします。**

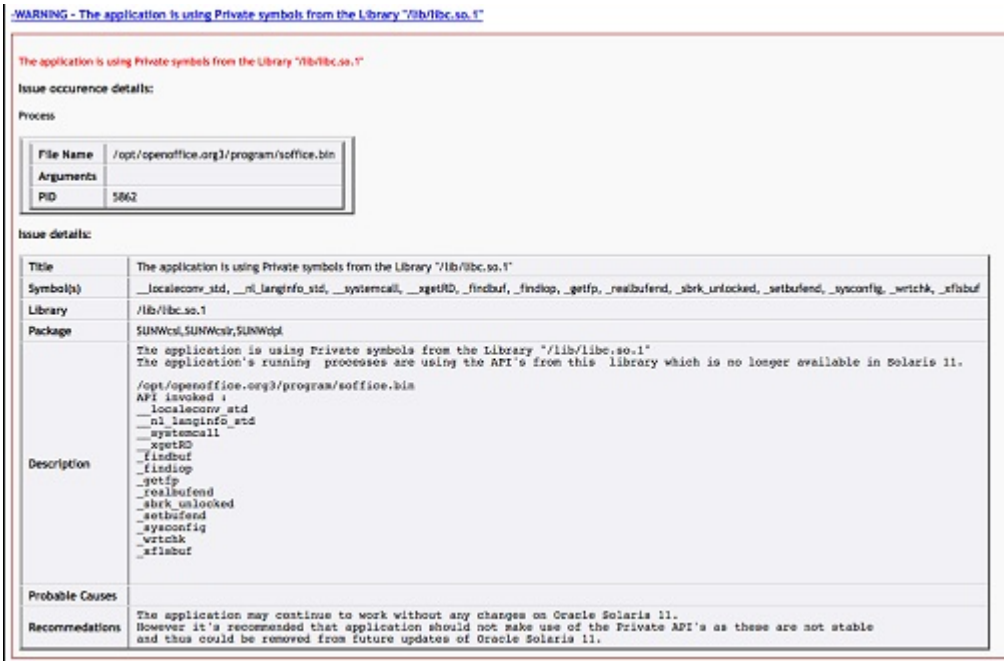
   変更していない場合は、現在の値が使用されます。

9. **レポートの場所をクリックしてレポートを開きます。**

10. **目次の分析サマリーのリンクをクリックして、スキャンの分析を表示します。**

11. **エラーまたは警告の詳細を表示するには、そのリンクをクリックします。**

    次の例は、問題の説明の例を示しています。




# Kernel Checker

Kernel Checker は、カーネルモジュールまたはデバイスドライバのバイナリとそのソースコードを分析し、Oracle Solaris 11 での既知の潜在的なコンプライアンスの問題を報告します。

バイナリアナライザまたはソースコードアナライザを実行する場合は、Oracle Solaris 10 システム上のチェック対象のバイナリおよびソースコードが含まれているディレクトリを示します。

## コマンド行での Preflight Kernel Checker の実行

Kernel Checker を起動するには、次のコマンドを発行します。

```
% ./preflightchecker.sh -kernelchecker [options] file-or-directory
```

Preflight Kernel Checker (`kernelchecker.sh`) の実行で使用可能なオプションを一覧表示するには、`-h` オプションを使用します。

```
% ./preflightchecker.sh -kernelchecker –h
```

Kernel Checker は、静的バイナリ分析中にディレクトリを再帰的に検索してオブジェクトファイルを検出し、Oracle Solaris 11 では機能しない可能性がある非推奨の API、サポートされていない API、および不安定な API が使用されていないかどうかをチェックします。

### ▼ GUI から Kernel Checker を実行する方法

1. **Kernel Checker の GUI を起動します。**

   ```
   # tool-install-dir/bin/preflightcheckerGUI.sh -kernelchecker
   ```

2. **「Kernel Checker」オプションを選択します。**

3. **「サマリー」画面で、アプリケーション名、バージョン、パス、マクロの定義などのアプリケーションの詳細を指定します。**

4. **カーネルモジュールまたはデバイスドライバのどちらをチェックするかを選択します。**

   - **カーネルモジュールが Oracle Solaris 11 上で問題なく実行されるかどうかをチェックするには、「互換性の確認」オプションを選択します。**
   - **デバイスドライバのコンプライアンスを確認するには、「DDI/DKI へのコンプライアンスの確認」オプションを選択します。**

5. **「ディレクトリを指定」ボタンをクリックして、スキャンレポートを保存するフォルダを選択します。**

6. **「分析を開始」ボタンをクリックしてスキャンを開始します。**

   カーネルスキャンが完了すると、スキャナによってレポートが生成されます。

7. **(オプション) テストを再度実行するには、S11preflightcheck の再実行ボタンをクリックします。**

   変更していない場合は、現在の値が使用されます。

8. **レポートを開くには、「レポートを表示」ボタンをクリックします。**

## Application Analyzer

Application Analyzer は、最適でないコーディングや実装、および特定の Oracle Solaris 機能の使用についてアプリケーションをチェックし、Oracle Solaris 11 プラットフォームを最大限に活用できるようにします。

また、Application Analyzer は、アプリケーションをスキャンして、コードを変更するとパフォーマンスがただちに改善されるかどうかをチェックします。このツールは、アプリケーションのプロセスおよびソースコードを分析し、推奨事項レポートを生成します。想定される推奨事項には、 gcc コンパイラから Oracle Solaris Studio への移行、正しいコンパイラフラグを使用したアプリケーションの最適化、Oracle Solaris Studio の高パフォーマンスライブラリの使用、およびオンチップのハードウェア暗号化を使用したサーバーのパフォーマンスの改善が含まれます。

Application Analyzer は、次のカテゴリの問題についてアプリケーションを分析します。

- 暗号化 - Application Analyzer の暗号化モジュールは、実行中のアプリケーションプロセスで暗号化機能が使用されているかどうかを検出します。Java アプリケーションの場合、アナライザは Java JCE API を使用しているかどうかをチェックします。ネイティブアプリケーションの場合、アナライザは `crypto` ライブラリの既知のリストの使用を検索します。どちらの場合も、アナライザから SPARC システムで使用可能なハードウェア暗号化機能を活用する推奨事項が提供されます。
- Makefile - Makefile ソースコードスキャナは、アプリケーションのソースコードをスキャンし、推奨レポートを HTML で生成します。

  このレポートには次の情報が表示されます。

  - アプリケーションが現在 gcc コンパイラを使用してコンパイルしている場合、Oracle Solaris Studio コンパイラを使用してコンパイルを完了するため変更する必要のあるフラグの詳細がレポートに示されます。
  - アプリケーションで古いバージョンの Oracle Solaris Studio コンパイラ (forte 6. x、Oracle Solaris Studio 10 など) を使用している場合は、Oracle Solaris 11 でのパフォーマンスを改善するため、最新の Oracle Solaris Studio コンパイラで使用する新しいフラグがレポートに示されます。このツールは、古いコンパイラの非推奨のフラグまたはサポートされていないフラグが使用されているかどうかも報告します。
  - メイクファイルとソースコードに基づく、Oracle Solaris 11 でアプリケーションのパフォーマンスを改善するために適したコンパイラの最適化フラグに関する推奨事項。
- 高パフォーマンスライブラリ - このモジュールは、実行中のアプリケーションプロセスをスキャンし、特定の Oracle Solaris 11 ライブラリの使用が適切かどうかに関する推奨事項レポートを生成します。たとえば、Sun Performance Library は、線形代数問題や非線形問題を数値的に解くための最適化された、かつ高速な数学サブルーチンを集めたものです。このツールは、アプリケーションプロセスを実行してそのようなパブリックドメインサブルーチンが使用されているかどうかを検出し、対応する Sun Performance Library を提案します。

## コマンド行での Application Analyzer の実行

Application Analyzer を起動するには、次のコマンドを発行します。

```
% appAnalyser.sh [options] file-or-directory
```

Preflight Application Analyzer (`appanalyser.sh`) の実行に使用可能なオプションを一覧表示するには、-h オプションを使用します。

```
% appAnalyser.sh -h
```

## ▼ GUI から Application Analyzer を実行する方法

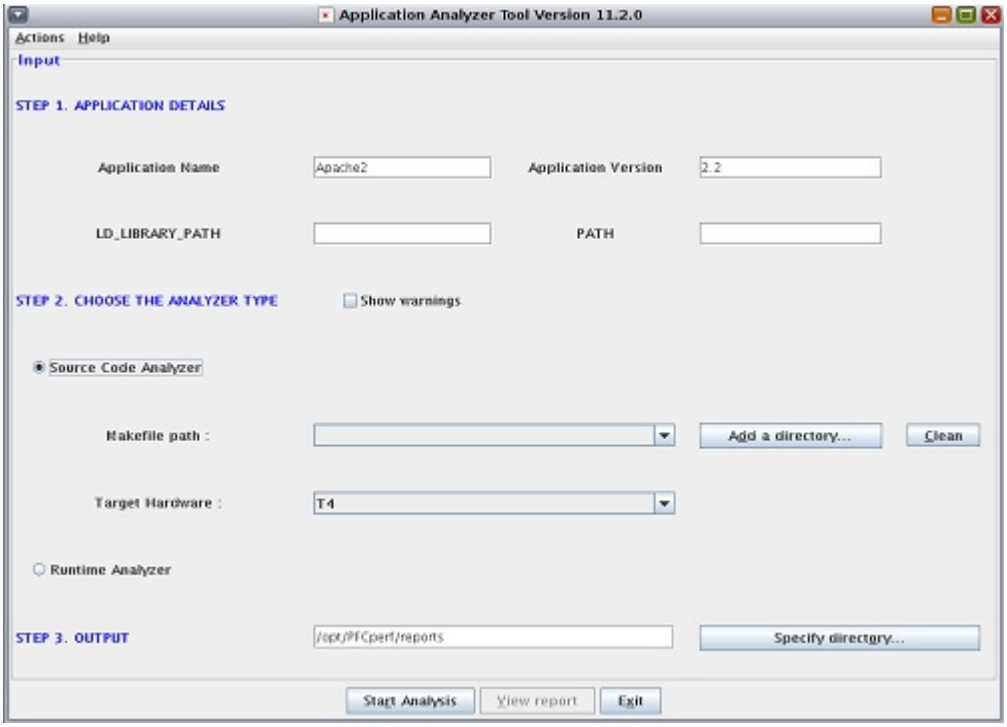
1. **Application Analyzer の GUI を起動します。**

   $ *tool-install-dir*`/bin/appAnalyzerGUI.sh`

2. **アプリケーション名、バージョン、パスなどのアプリケーションの詳細を指定します。**

3. **アナライザのタイプを選択します。**

   - **削除、名前変更、または変更されたライブラリ関数が使用されていないかどうかをチェックするには、「ソースコードアナライザ」オプションを選択します。**
   - **ゾーンや指定した実行可能ファイル名の特定のプロセスなどのスコープ内のソースコードを分析するには、「実行時アナライザ」オプションを選択します。**



4. **「ディレクトリを指定」ボタンをクリックして、スキャンレポートを保存するフォルダを選択します。**

5. **「分析を開始」ボタンをクリックしてスキャンを開始します。**

   アプリケーションスキャンが完了すると、スキャナによってレポートが生成されます。

6. **(オプション) テストを再度実行するには、S11preflightcheck の再実行ボタンをクリックします。**

   変更していない場合は、現在の値が使用されます。

7. **レポートを開くには、「レポートを表示」ボタンをクリックします。**

## ヘルプおよびサポート

Oracle Solaris Preflight Applications Checker のアップデートおよび最新バージョンについては、Oracle Solaris Preflight Applications Checker 11.3 の製品ページを参照してください。

独立系ソフトウェアベンダー (ISV) は、`isvsupport_ww@oracle.com` に連絡することによって、このツールに関する支援またはサポートを受けることができます。

# 索引

## I



## O



## P



## R