バイナリ互換性ガイド

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650–960–1300

Part No: 805–5113–10
1998 年 11 月

原典: *Binary Compatibility Guide*

Part No: 805-4029-10

Revision A

目次

はじめに

SunOS™ 5.x は、AT&T の SVR4 (System V Release 4.0) をベースとしたオペレーティングシステムです。このため SunOS 5.x は、以前の SunOS 4.x リリース (4.3 BSD ベース) とバイナリ互換性を持ちません。SunOS 4.x 上で稼働しているアプリケーションのなかには、そのままでは SonOS 5.x 上で正しく実行できないものもあります。

SunOS 5.x に対し、「SunOS バイナリ互換パッケージ」を使用すると、SunOS 4.x ベースのアプリケーションを実行できるようになります。また、「OpenWindows™ バイナリ互換パッケージ」は、SunOS 4.x 上のウィンドウアプリケーションを、Solaris™ 2.x (SunOS 5.x と OpenWindows) 環境で実行可能とします。SunOS バイナリ互換パッケージと OpenWindows バイナリ互換パッケージ (ソース互換パッケージも別個必要です) を使用すると、SunOS 4.x リリースで正しく働いているアプリケーションバイナリを、修正も再コンパイルもせずに Solaris 2.x 上で実行できます。しかし、これらのパッケージは、Solaris 2.x へ移行する際の一時的な支援として使用されることを意図しており、アプリケーションの移植が不要になるわけではありません。

このマニュアルにおける用語の使い方

このマニュアルでは、「SunOS 4.x」という言葉を、以下に示す各リリースの総称として使用します。

- SunOS 4.1
- SunOS 4.1.1

- SunOS 4.1.2
- SunOS 4.1.3

対象読者

このマニュアルは、SunOS 4.x のアプリケーションを Solaris 2.x リリースで簡単に実行できるようにしたいと望んでいるアプリケーション開発者を対象としています。このマニュアルでは、バイナリ互換パッケージとは何か、それによって何ができ、また何ができないか、そしてバイナリ互換パッケージのインストール方法と使用方法について記述します。また、アプリケーションを開発するとき、または既存の SunOS 4.x アプリケーションを Solaris 2.xで容易に実行できる方法を検討するときに、どのような点を考慮する必要があるかについても説明します。このマニュアルで述べる最も重要な点は、このパッケージに関する制約事項、つまり、バイナリ互換性が維持できない項目に関する記述です。

互換性の一般的な事項に関する詳しい説明は、『*Solaris* 移行ガイド』に収めてあります。

このマニュアルの構成

このマニュアルの構成は以下のとおりです。

第 1 章では、これらのパッケージをインストールし使用する方法について説明します。

第 2 章では、SunOS バイナリ互換パッケージが、システムインタフェースレベルでアプリケーションに何を提供するかについて説明します。この章では、バイナリ互換性が利用できない状況についても説明します。

第 3 章では、ウィンドウシステム互換性について詳しく説明します。この章では、SunOS 4.x リリースで使用できる各種のウィンドウマネージャおよびツールキットに対し、バイナリ互換性がどの程度まで実現できるかについて説明します。

マニュアルの注文方法

SunDocs™ プログラムでは、米国 Sun Microsystems™, Inc. の 250 冊以上のマニュアルを扱っています。このプログラムを利用して、マニュアルのセットまたは個々のマニュアルをご注文いただけます。

マニュアルのリストと注文方法については、米国 SunExpress™, Inc. のインターネットホームページ `http://www.sun.com/sunexpress` にあるカタログセクションを参照してください。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、およびディレクトリ名を示します。または、画面上のコンピュータ出力を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`
`AaBbCc123`	ユーザが入力する文字を、画面上のコンピュータ出力とは区別して示します。	`system% su` `password:`
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。

表 P–1　表記上の規則　続く

字体または記号	意味	例
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章や節を示します。また、ボタンやメニューなど、強調する単語を囲む場合にも使用します。	第 5 章「衝突の回避」を参照してください。

コード例は次のように表示されます。

■ C シェルプロンプト

```
system% command [filename]
```

■ Bourne シェルおよび Korn シェルのプロンプト

```
system$ command [filename]
```

■ スーパーユーザのプロンプト

```
system# command [filename]
```

[ ]は省略可能な項目を示します。上記の場合、*filename* は省略してもよいことを示します。

ただし AnswerBook2 では、ユーザが入力する文字と画面上のコンピュータ出力は区別して表示されません。

一般規則

■ このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が適宜、併記されています。

第 1 章

バイナリ互換パッケージの紹介

バイナリ互換パッケージの目的と機能

SunOS バイナリ互換パッケージと OpenWindows バイナリ互換パッケージは、Solaris 2.x リリースで使用できるオプションパッケージです。これらのパッケージを用いることで、OpenWindows アプリケーションも含めた既存の SunOS 4.x アプリケーションを、修正も再コンパイルもせずに、 Solaris 2.x 上で実行できます。これらのパッケージは、 2 つのリリース間のバイナリインタフェースの差異のほとんどを透過的に処理し、SunOS 4.x アプリケーションを正しく実行できる動作環境を提供します。

バイナリ互換パッケージを提供する理由

これらのパッケージは単なる移行ツールです。既存の SunOS 4.x バイナリを Solaris 2.x エンドユーザ環境において実行できるようにするためのもので、開発環境として提供されるものではありません。Solaris 2.x ユーザアプリケーションを開発するには、Solaris 2.x の環境を使用する必要があります。

バイナリ互換パッケージのインストール

SunOS バイナリ互換パッケージと OpenWindows バイナリ互換パッケージは、オプションとしてインストールできるものです。ソース互換パッケージが既にインストールされていることが必要です。SunOS バイナリ互換パッケージは `SUNWbcp` と呼ばれ、OpenWindows バイナリ互換パッケージは `SUNWowbcp` と呼ばれます。

オプションパッケージのインストールの詳細については、『*Solaris* のインストール *(*上級編*)*』を、ソース互換パッケージの詳細については、『*Source Compatibility Guide*』を参照してください。

注 - バイナリ互換パッケージを使用したい場合は、ソフトウェアのインストールをカスタマイズする際に、上記の互換パッケージを必ず両方ともインストールしてください。

バイナリ互換パッケージの使用方法

バイナリ互換パッケージの機能は、SunOS 4.x アプリケーションを Solaris 2.x 上で実行する時に、自動的かつ透過的に呼び出されます。アプリケーションを変更する必要はありません。

すべての Solaris 2.x リリースでは SunOS 4.x の動的にリンクされた実行可能ファイルを使用できます。Solaris 2.3 以降では、静的にリンクされた実行可能ファイルも実行できます。Solaris 2.5 以降では、静的リンクと動的リンクの混在した実行可能ファイルを使用できます。

SunOS 4.x と Solaris 2.x の間の多くの相違点があります。このうちのいくつかは、コマンドのバイナリやライブラリなどがインストールされているディレクトリやファイル名の違いに起因します。このような相違点があるため、PATH 環境変数の定義のしかたによって、バイナリ互換パッケージを使用した場合のアプリケーションの実行状況が異なります。次の項ではこの点について詳しく説明します。

パス名の解決処理

パス名の解決処理については、問題点が 2 つあります。第 1 は、Solaris 2.x リリースと SunOS 4.x リリースとの間でコマンドやライブラリのファイルが存在する位置 (ディレクトリ) が異なるという点です。たとえば、SunOS 4.x リリースでは `/usr/bin` に入っていたコマンドが、Solaris 2.x リリースでは `/usr/ucb` に入っている場合があります。ソース互換パッケージは、この相違点を解決するために、可能な限りシンボリックリンクをセットアップして、旧パス名を使用しているアプリケーションがファイルにアクセスできるようにします。

第 2 に、Solaris 2.x リリースの多くのコマンドと、SunOS 4.x の対応コマンドとの間に互換性がないという点があります。これに対処するために、ソース互換パッケージには、一組の SunOS 4.x 互換コマンドバイナリが組み込まれています。アプリケーションが、実行するコマンド名だけ (パス名なし) を指定した場合は、バイナリ互換パッケージのもとで PATH 環境変数が解釈され、正しいコマンドが見つけられます。したがって、アプリケーションを実行する前に、PATH 変数を正しく設定しておく必要があります。

たとえば、SunOS 4.x のデフォルトの動作を実行したい場合は、`PATH` の中で `/usr/bin` の前に `/usr/ucb` を指定します。このようにしておかないと、Solaris 2.x のデフォルトの動作が引き起こされることになります。

フルパス名を指定したコマンドは、バイナリ互換パッケージ環境下でも、そのパス名の通りに解釈されます。パスが `/usr/5bin` で始まっている場合は、そのコマンドのデフォルトである Solaris 2.x バージョンが実行されます。そうでない場合は、ソース互換パッケージバージョンが実行されます。相対パス名も通常と同じく解釈されます。

Solaris 2.5 から、5 つのコマンドが `/usr/ucb` から `/usr/bin` へ移動され、絶対パスを使用して書いたシェルスクリプトを使用できるようになりました。これらのコマンドは、`arch(1)`、`hostid(1)`、`hostname(1)`、`mach(1)` および `pagesize(1)` です。

パフォーマンスへの影響

バイナリ互換パッケージを使って SunOS 4.x バイナリを実行するには、同じバイナリを SunOS 4.x の下で実行する場合や、同じアプリケーションを Solaris 2.x に移植してから実行する場合に比べて、多くのメモリが必要です。さらに、必要なディスク記憶容量とディスクアクセスの回数も多くなります。したがって、バイナリ互換

パッケージを使用すると、アプリケーションとシステム全体のパフォーマンスは低下します。

適正動作のアプリケーション

バイナリ互換パッケージが取り扱うことができるのは、適正動作のユーザアプリケーションです。適正動作性を備えていないアプリケーションは予期しない結果をもたらす恐れがあるので、このようなアプリケーションに対してバイナリ互換パッケージを使用することはお勧めできません。適正動作のアプリケーションとは、静的または動的にリンクされ、以下のすべての要件を満たしているものをいいます。

- カーネルに対して直接トラップを行わない。
- システムファイルへの直接の書き込みを行わない。
- /dev/kmem、/dev/mem、または libkvm を使用しない。
- 公式に発表されていない SunOS インタフェースを使用しない。
- ユーザ提供のドライバに依存しない。
- SPARC アーキテクチャに関する知識に依存しない。

/dev/kmem、/dev/mem、および libkvm

/dev/kmem および /dev/mem の内容は、各リリースに固有のものです。/dev/kmem および /dev/mem を使用するプログラムは、特定バージョンのオペレーティングシステムに結び付いています。互換性を維持することはできません。/dev/kmem、/dev/mem、または libkvm ルーチンをアプリケーションの中で使用していると、そのアプリケーションは移植不能となります。

SunOS 4.x アプリケーションのインストール

バイナリ互換パッケージを使用するには、Solaris 2.x アプリケーションが Solaris 2.x システムで利用可能な状態になっていることが必要です。SunOS 4.x アプリケーションを、Solaris 2.x システムにインストールすることは可能です。また、SunOS 4.x バイナリを含む領域をマウントすることもできます。

SunOS 4.x アプリケーションをインストールするには、実行モジュールをそれぞれのオリジナルの `a.out` 形式で、Solaris 2.x システム内の対応するディレクトリ位置にコピーします。アプリケーションは、`cp(1)`、`rcp(1)`、`tar(1)`、`cpio(1)` などを使ってコピーできます。これらのコピーコマンドについては、*Solaris 7 Reference Manual Collection* にある『*man Pages(1): User Commands*』を参照してください。

ほとんどの場合、4.x アプリケーションはオリジナルの配布媒体からインストールできます。インストールスクリプトが正しく働くためには、`PATH` 変数の中で `/usr/bin` の前に `/usr/ucb` を指定することが必要な場合があります。

SunOS 4.x のディレクトリを Solaris 2.x システムにマウントし、バイナリ互換パッケージを使って、それらのディレクトリに含まれているバイナリを実行することもできます。

ファイルシステム内で対応する位置が存在しない場合や、SunOS 4.x のファイルと Solaris 2.x のファイルの名前が衝突する場合は、ディレクトリの作成、ファイルの名前変更 (名前の衝突の回避) を、個別に行うことが必要になります。

共用オブジェクトファイル

SunOS 4.x リリースで使用していた、SunOS と OpenWindows のライブラリ以外の (サードパーティライブラリなどの) `a.out` 共用ライブラリを、Solaris 2.x にコピーするかまたはマウントする必要があります。これらのライブラリは、`/usr/4lib` ディレクトリにコピーするのが適当です。これらのライブラリは、SunOS 4.x リリースのときと同じメジャーバージョン番号とマイナーバージョン番号を維持していることが必要です。

実行時リンカは、以下のパス (ディレクトリ間をコロンで区切ったリストの形で指定される) を検索して、関連するライブラリを見つけます。

- 環境変数 LD_LIBRARY_PATH
- リンク時に `ld(1)` の `–L` *dir* オプションにより指定されたディレクトリ
- デフォルトのディレクトリ `/usr/4lib:/usr/lib:/usr/local/lib`

`/usr/4lib` ディレクトリは、SunOS 4.x にはなかったものです。これは、SunOS 4.x の `a.out` ライブラリの格納場所として提供されています。このディレクトリがデフォルトの検索パスに追加されたため、互換パッケージのユーザは SunOS 4.x のライブラリを `/usr/4lib` に置けば、実行時リンカがそれらのライブラリを必ず見つけます。

第 2 章

バイナリ互換性

この章では、SunOS バイナリ互換パッケージが、システムインタフェースとして提供するバイナリ互換性について説明します。この章で述べる機能と制約事項は、ウィンドウベースと非ウィンドウベースのアプリケーションのどちらにも適用されます。 第 3 章には、OpenWindows バイナリ互換パッケージに固有の互換性情報を収めてあります。

パッケージの働き

まず、SunOS バイナリ互換パッケージが解決する非互換性の主なものを下にリストします。その後で、それぞれについて詳しく説明します。これらの非互換性の一部はソース互換パッケージにより解決されます。ソース互換パッケージは、バイナリ互換パッケージとともにインストールしておく必要があります。

この 2 つのパッケージは次のことを行います。

- SunOS 4.x のバイナリが実行されるとき、正しいライブラリ群が適正にリンクされロードが行われるようにします。
- `a.out` オブジェクトファイル形式の SunOS 4.x 実行ファイルを、Solaris 2.x リリースで実行できるようにします (Solaris 2.x ではデフォルトのオブジェクトファイル形式は `ELF` です)。
- すべてのライブラリルーチンおよびシステムコールについて、SunOS 4.x と同様な動作が実行されるようにします。コールへのインタフェースまたはコールの動作が異なっていても、このマッピングにより SunOS 4.x のバイナリが確実に初期の動作をとるようにされます。

- 静的リンクと動的リンクの混在したSunOS 4.x の実行可能ファイルが、Solaris 2.x リリースで正常に実行されるようにします。
- SunOS 4.x と同様なシグナル処理を提供し、SunOS 4.x と Solaris 2.x のシグナル間のマッピングを正しく行います。
- サポートされている SunOS 4.x の `ioctl()` のセットを、対応する Solaris 2.x の `ioctl()` に正しくマッピングし、`ioctl()` のパラメータもマッピングします。
- コマンドおよびユーティリティがそれぞれ予期された位置にあることを確認し、必要があれば、パス名のマッピングを透過的に行います。
- ファイル形式が異なる場合、またはSunOS 4.x のファイルに対応するファイルがSolaris 2.x リリースにない場合には、可能な限りシステムファイルの SunOS 4.x 互換バージョンを提供します。
- システムファイルの名前またはパスが異なる場合に、適正なリンクがセットアップされるようにします。

このパッケージではできないこと

この項では、バイナリ互換パッケージを使っても非互換性が残ってしまうような状況について説明します。非互換として規定されている機能を使用するアプリケーションは、処理が単に失敗するだけの場合もありますが、SunOS 4.x の下で実行したときとは異なる結果をもたらすか、または、SunOS 5.x で予定されているのとは矛盾する結果をもたらすこともあります。

`getXXbyYY()` 関数

SunOS 4.x には、ユーザ、ホスト、グループなどに関する情報を検索する一群の関数があります。これらは、情報検索の順序によって 2 つのセットに分類されます。1 セット目の関数とは、リファレンスマニュアルの次の項に説明があるものです。

`gethostent(3N)`、`getnetent(3N)`、`getnetgrent(3N)`,`getprotoent(3N)`、`getpublickey(3N)`、`getrpcent(3N)`、`getsecretkey(3N)`、`getservent(3N)`、`yp_get_default_domain(3N)`

これらの関数は、まず該当の NIS マップを検索します。そして、NIS が実行中でない場合に限り、`/etc` 内の対応するファイルを検索します。2 セット目の関数は、リファレンスマニュアルの `getpwent(3V)` と `getgrent(3C)` の項に説明がありま

す。この 2 つの関数は、まず /etc の中のファイルを検索します。その結果データが見つからない場合は、NIS が実行状態にあれば、NIS に対して照会します。静的にリンクされているアプリケーションの場合は、これらの関数は、SunOS 5.x でも、SunOS 4.x で実行した場合とまったく同じ働きをします。

SunOS 5.0 以降のリリースでは、上に列挙した情報検索関数はすべて、ネームサービススイッチを介してそれぞれの機能を実行するように変更されました。実行時に、構成ファイル /etc/nsswitch.conf を参照して情報ソースとルックアップ順序が決定されます。

SunOS 4.x で静的にリンクされているアプリケーションを、バイナリ互換パッケージを使って SunOS 5.x で実行した場合、これら情報検索関数は SunOS 4.x のときとまったく同じ働きをします。このため、その動作がネイティブ SunOS 5.x アプリケーションと異なる可能性があります。SunOS 5.x を実行するマシンのネームサービススイッチが、この項の冒頭で述べたような動作をするものとして構成されていない場合は、静的にリンクされた 4.x アプリケーションの中の情報検索関数は、ネイティブ 5.x アプリケーションの場合とは異なる動作をすることがあります。一方、SunOS 5.x で実行するものとして動的にリンクされた 4.x アプリケーションにおいては、情報検索関数は、ネイティブ 5.x アプリケーションの場合と同じ動作をします。

オブジェクトファイルと **core** ファイルの形式

SunOS 4.x リリースは、a.out オブジェクトファイル形式を使用します。Solaris 2.x リリースは、新しい拡張可能リンク形式 (ELF) を使用します。Solaris 2.x プログラミングツール (cc、ld、ar など) は ELF ファイルを生成します。また、デフォルトの exec() ファイル形式は ELF です。

Solaris 2.x の exec() システムコールは、実行ファイルの形式 (a.out または ELF) を識別し、それぞれに応じた正しいロード手順を使用します。

デフォルトの core ファイル形式は、SunOS 4.x では a.out であり、Solaris 2.x リリースでは ELF です。Solaris 2.x で バイナリ互換パッケージを使って a.out ファイルを実行した場合、core ファイルは a.out 形式で生成されます。互換性メカニズムが起動されるのは、a.out バイナリ形式のファイルをロードし実行した場合だけです。

a.out 実行ファイル形式、core 形式を想定して動作する SunOS 4.x プログラムは、他の形式のファイルに対しては働きません。たとえば、a.out 形式のファイルを対象とした SunOS 4.x の nm(1) コマンドなどは、ELF 形式に対しては働きません。

Solaris 2.x リリースのプログラミングツールおよびユーティリティは、a.out 形式のバイナリおよび core ファイルに対しては、意図したとおりの働きをしません。

システムコール

SunOS 4.x と Solaris 2.x のシステムコールの間には、システムコール番号の違いやコールインタフェースや動作の違いなど、大きな相違点がいくつかあります。バイナリ互換パッケージは、マッピングを行なったり、意図したインタフェースまたは動作を提供する代替ルーチンを使用するなどして、2 つのバージョン間のシステムコールの相違点のほとんどを吸収しています。

システムコールのマッピングの結果として、バイナリ互換パッケージの下で実行される truss(1) プログラムの出力が、予測とは異なるものになることがあります。たとえば、システムコールの名前が違ったり、予期したときにシステムコールが発生しなかったりすることがあります。また、コールに渡される引数や戻り値が違うこともあります。たとえば、パス名の解決処理を必要とする場合に (3ページの「パス名の解決処理」を参照)、あるファイルに対する open() コールの結果、実際には別のファイルがオープンされることがあります。

アプリケーションは、libc に収められているラッパルーチンを介してシステムコールを呼び出す必要があります。トラップメカニズムを介してシステムコールを直接呼び出すことは、絶対にしないでください。これを行うと、データとシステムコール番号を正しくマップすることができません。

非互換インタフェース

システムコールに関する非互換性の一部は、このパッケージでは対処できません。その理由としては、コールがすでに廃止されていたり、ユーザアプリケーションで使用すべきものではなかったり、他の独立パッケージに組み込まれたりしていることが考えられます。以下、この種のコールをリストし、それぞれについて説明します。

acct

Solaris 2.x バージョンの acct() には、アプリケーションバイナリインタフェース (ABI) および System V インタフェース定義 (SVID) との互換性がありますが、SunOS 4.x にはその互換性はありません。相違点は、アカウントのオフへの切替えに関する点と、アカウントレコードの形式です。Solaris 2.x リリースでは、アカウントがすでにオンになっているときは、特定のパス名を指定して acct() をコール

すると、アカウントはオフにされます。同じ条件下での SunOS 4.x リリースの場合は、コールしても別のアカウントファイルに切り替わるだけで、アカウントはオフにはなりません。

audit、auditon、auditsvc、getauid、setauid、setaudit、setuseraudit

バイナリ互換パッケージでも、また Solaris 2.x 環境でも、監査はサポートされていません。監査は、Solaris 2.x リリースでは独立したパッケージで取り扱います。そのパッケージが提供する監査機能には、SunOS 4.x とのバイナリ互換性がありません。

getdirentries

このコールは廃止されました。Solaris 2.x リリースまたはバイナリ互換パッケージではサポートされません。

flock

Solaris 2.x では、`fcntl(2)` ロックに加えて、`flock` の 1 バージョンがインプリメントされています。このバージョンと SunOS 4.x のバージョンは、完全なバイナリ互換性を備えているわけではなく、以下のような相違点があります。

- `flock` では、排他的ロックを要求する前にファイルが書き込み用にオープンされていることが必要です。
- 共用ロックを取得するには、ファイルに対する読み取り許可が必要です。
- プロセスが `flock()` を使ってすでにロックされているセグメントをロックすると、古いロックタイプが除去され、新しいロックタイプが効力を持ちます。
- ロックは、`fork()` 関数の子プロセスには継承されません。
- SunOS 4.x の下で `flock()` メカニズムにより取得したロックは、それを設定したシステムの中でしか認識されませんでした。この制約はなくなりました。

SunOS 4.x の `flock()` 動作は、デフォルトの Solaris 2.x リリースでもこのパッケージでも利用できません。

ioctl

古い Version 7 および 4BSD ターミナルドライバによりサポートされていた `ioctl()` に加えて、`filio`、`sockio`、`streamio`、`termio`、`termios`、`mtio`、および `dkio` に関連したすべての `ioctl` がサポートされます。その他には、Solaris 2.x プラットフォームの標準デバイスに関連した `ioctl()` のみが提供されています。2 つのバージョン間での `ioctl()` 番号の相違 (サポートされている `ioctl()` の場合) は、透過的に処理されます。`ioctl()` のパラメータは、必要に応じてマップされます。

以下に示す SunOS 4.x の `ioctl()` には、Solaris 2.x との互換性がありません。

`DKIOCGCONF`	この `ioctl()` は、Solaris 2.x では使用できませんが、バイナリ互換パッケージはこれをサポートしています。この `ioctl()` は、`DKIOCINFO` で置き換えられました。`DKIOCGCONF` には、SunOS 4.x の `DKIOCGCONF` 構造体と `DKIOCINFO` 構造体とを組み合わせた情報が含まれています。
`DKIOCGLOG`	この `ioctl()` は Solaris 2.x ではサポートされていません。バイナリ互換パッケージでは、これは `EINVAL` を戻します。
`DKIOCWCHK`	SunOS 4.x では、この `ioctl()` は、フロッピーデバイスの書き込みチェックを切り替えます。バイナリ互換パッケージでは、この `ioctl()` はフロッピーデバイスの書き込みチェックを切り替えませんが、操作成功のステータスを返します。
`DKIOCSCMD`	この `ioctl()` が使用できるのは、`xd(7)`、`xy(7)`、`ipi(7)` の各ドライブの場合だけです。この `ioctl()` は、SCSI デバイスの場合は失敗します。この種のデバイスの場合は、USCSI `ioctl()` を使用してください。
`_O_TIOCCONS`	この `ioctl()` は廃止され、Solaris 2.x リリース、バイナリ互換パッケージではサポートされません。

`_O_TIOCGSIZE`	この `ioctl()` は廃止され、Solaris 2.x リリース、バイナリ互換パッケージではサポートされません。
`_O_TIOCSSIZE`	この `ioctl()` は廃止され、Solaris 2.x リリース、バイナリ互換パッケージではサポートされません。
`TIOCMODG`	この `ioctl()` は廃止され、Solaris 2.x リリース、バイナリ互換パッケージではサポートされません。
`TIOCMODS`	この `ioctl()` は廃止され、Solaris 2.x リリース、バイナリ互換パッケージではサポートされません。

kill

この Solaris 2.x システムコールは、SunOS 4.x の場合とは異なる働きをします。最初の引数として –1 が与えられた場合は、呼び出したプロセス自身も終了されます。SunOS 4.x では、この動作はありませんでした。

pipe

SunOS 4.x での `pipe()` システムコールは、1 つのディスクリプタは読み取り用に、別のディスクリプタは書き込み用にと、それぞれ用途を分けてオープンします。Solaris 2.x では、このシステムコールは、ディスクリプタを読み取りと書き込みの両用としてオープンします。この相違点の影響を受けるバイナリは、(仮にあったとしても) ほとんどないので、互換バージョンは提供されていません。

ptrace

SunOS 4.x の `ptrace()` システムコールに渡すオプションの *addr2* 引数は、サポートされていません。また、Solaris 2.x リリースでは、次のリクエストはサポートされません。

`PTRACE_ATTACH`	/* 10, attach to an existing process */
`PTRACE_DETACH`	/* 11, detach from a process */

```
PTRACE_GETREGS          /* 12, get all registers */
PTRACE_SETREGS          /* 13, set all registers */
PTRACE_GETFPREGS        /* 14, get all floating point regs */
PTRACE_SETFPREGS        /* 15, set all floating point regs */
PTRACE_READDATA         /* 16, read data segment */
PTRACE_WRITEDATA        /* 17, write data segment */
PTRACE_READTEXT         /* 18, read text segment */
PTRACE_WRITETEXT        /* 19, write text segment */
PTRACE_GETFPAREGS       /* 20, get all fpa regs */
PTRACE_SETFPAREGS       /* 21, set all fpa regs */
PTRACE_GETWINDOW        /* 22, get register window n */
PTRACE_SETWINDOW        /* 23, set register window n */
PTRACE_22               /* 22, filler */
PTRACE_23               /* 23, filler */
PTRACE_SYSCALL          /* 24, trap next sys call */
PTRACE_DUMPCORE         /* 25, dump process core */
PTRACE_SETWRBKPT        /* 26, set write breakpoint */
PTRACE_SETACBKPT        /* 27, set access breakpoint */
PTRACE_CLRDR7           /* 28, clear debug register 7 */
PTRACE_26               /* 26, filler */
PTRACE_27               /* 27, filler */
PTRACE_28               /* 28, filler */
PTRACE_GETUCODE         /* 29, get u.u_code */
```

swapon

Solaris 2.x リリースとバイナリ互換パッケージには、このシステムコールはありません。ユーザアプリケーションでこのシステムコールは必要とされないはずです。

vadvise

デフォルトの Solaris 2.x リリースとバイナリ互換パッケージには、このシステムコールはありません。

ライブラリルーチン

バイナリ互換パッケージは、SunOS 4.x のライブラリと互換性を持つ一組のライブラリを提供します。Solaris 2.x リリースにおいて、ライブラリルーチンのインタフェースや動作が変更されていても、バイナリ互換パッケージを使用していれば、ユーザアプリケーションがその影響を受けることはありません。

xtab

バイナリ互換パッケージを使用している場合は、`/etc/xtab` ファイルを取り扱うライブラリルーチンは正しく働きません。

setlocale

`setlocale()` が戻す文字列内のロケールの順序には、SunOS 4.x と Solaris 2.x の間で相違があります。この文字列は、一般に後続の `setlocale()` に対するコールで使用されるものであり、この順序は問題ではありません。特定のロケール順序を必要とするようなアプリケーションは、バイナリ互換性を備えていない場合があります。

エラー

データのマッピングはすべてバイナリ互換ライブラリの中で行われます。誤ったアドレスが引数として渡された場合に、必ずしも、予定されている `EFAULT` エラーが戻されるとは限りません。データマッピングライブラリルーチンは誤ったアドレスを捕捉しようとしますが、これは常に可能ではないため、誤ったアドレスのアクセスの結果として、バスエラー/セグメンテーション違反が生じることがあります。

シグナル

バイナリ互換パッケージは、SunOS 4.x と互換性のあるシグナル処理メカニズムを備えています。また、シグナル番号の相違も解決します。

SIGLOST シグナルはサポートされていません。

システムファイル

Solaris 2.x リリースでは、多くのシステムファイルの名前が変更されたり、移動されたりしています。このリリースでは存在しなくなったものもあり、形式が変わったものもあります。バイナリ互換パッケージは、シンボリックリンクを作成するか新規ファイルをインストールし、さらにデータをマップすることによって、可能な限りこの問題に対処します。

注 - 移植可能プログラムでは、システム依存ファイルの使用は避けてください。このようなファイルへのアクセスを必要とするプログラムは、提供されている標準インタフェースルーチンを使ってそのアクセスを行う必要があります。たとえば、この種のプログラムでは、/etc/mtab ファイルを直接オープンし読み取るのではなく、getmnt() ファミリのルーチンを使って mtab ファイルの内容にアクセスするようにします。

アカウントファイル

Solaris 2.x のアカウントファイルは、SunOS 4.x のアカウントファイルと形式が異なります。アカウントファイルにアクセスするアプリケーションには、バイナリ互換性はありません。

/etc/exports、/etc/xtab

Solaris 2.x リリースでは、システムファイルのエクスポートの取り扱い方が SunOS 4.x の場合と大幅に異なります。これらのファイルについてのバイナリ互換性はありません。しかし、ユーザアプリケーションがこれらのファイルにアクセスする必要はないので、これは問題ではありません。

/etc/fstab、/etc/mtab

これらのファイルの名前と形式が変わりました。バイナリ互換パッケージは、これらのファイルに対する読み取り専用アクセスをアプリケーションに与えるために必要なマッピングを行います。これらのファイルに対する書き込みアクセスを必要とするアプリケーションには、バイナリ互換性はありません。アプリケーションは、これらのファイルへの書き込みは行わないものと想定されます。

注 - これらのファイルへのシンボリックリンクはサポートされなくなりました。つまり、アプリケーションが `/etc/fstab` または `/etc/mtab` へのリンクを作成し、シンボリックリンクをオープンしようとすると、その `open()` コールは失敗します。

/etc/gettytab

このファイルに直接対応するファイルは Solaris 2.x リリースにはありません。バイナリ互換パッケージの一部としても提供されていません。このファイルを使用するアプリケーションには、バイナリ互換性はありません。

/etc/passwd

Solaris 2.x リリースでは、実際のパスワードはシャドウファイル内に保管されます。`passwd` ファイルにアクセスしてパスワードを取得するアプリケーションにバイナリ互換性はありません。すべてのアプリケーションは、`getpw()` インタフェースルーチンを介してこのファイルにアクセスする必要があります。

/etc/printcap

SunOS 4.x の `printcap` は、SunOS 5.x では `terminfo` ディレクトリツリーで置き換えられました。バイナリ互換パッケージは、アプリケーションが実行されているホスト上の対応するデータへの読み取り専用アクセスを、アプリケーションに与えます。`printcap` への書き込みアクセスはサポートされていません。

/etc/ttys

SunOS 4.x ではこのファイルは廃止されました。`utmp` ファイルの形式およびアクセスメカニズムが著しく異なるため、`/etc/ttys` と `utmp` ファイルの間の関係に依存するプログラムは、Solaris 2.x リリースでは働きません。動的にリンクされるアプ

リケーションには、ttyslot を使用してください (静的にリンクされるアプリケーションには無効です)。

/etc/ttytab

SunOS 4.x では、このファイルは /etc/ttys に取って代わりました。SunOS 5.x では廃止されました。互換性は提供されていません。

/etc/utmp、/var/adm/wtmp

これらのファイルには、who、write、login などのコマンドに関連するユーザ情報とアカウント情報が入っています。Solaris 2.x では、これらのファイルはそれぞれ一対のファイルで置き換えられました。たとえば、/etc/utmp は /var/adm/utmp と /var/adm/utmpx で、/var/adm/wtmp は /var/adm/wtmp と /var/adm/wtmpx で置き換えられました。これらのファイルのどちらかをオープンしようとしたアプリケーションは、それに対応する 2 つのファイルを連結したものを受け取ることになります。

注 - これらのファイルへのシンボリックリンクはサポートされなくなりました。つまり、アプリケーションが /etc/utmp または /var/adm/wtmp へのリンクを作成し、シンボリックリンクをオープンしようとすると、その open() コールは失敗します。

ローカライズされたアプリケーション

ローカライズされた 4.x のアプリケーションは、バイナリ互換パッケージが静的か動的かに関わらず、US 版の SunOS 5.x では実行されません。特定の 4.x アプリケーションがローカライズされた Solaris 2.5 で実行できるかどうかを判別するには、該当のローカライズ関係マニュアルを参照してください。

注意事項と副作用

ファイルディスクリプタの制限

1 つのプロセスがオープンできるファイルディスクリプタの数に関するデフォルトの制限は 64 です。この制限数は、`csh(1)` の `limit` コマンド、`sh(1)` および `ksh(1)` の `ulimit` コマンド、そして `setrlimit(2)` システムコールを使って大きくすることができます。ファイルディスクリプタの制限数を 256 より大きくした場合は、静的または動的のどちらのバイナリ互換パッケージの下で実行するプロセスも失敗します。これは、実際にオープンされているファイルが 256 より少ない場合でも失敗することがあります。4.x のほとんどのアプリケーションは、ファイルディスクリプタを 256 個までしか扱えません。

デバイス番号

SunOS 4.x では、システムがサポートする最大のメジャーデバイス番号は 255 でした。SunOS 5.x では、メジャーデバイス番号に関する最大制限がはるかに大きくなっています。静的または動的バイナリ互換パッケージの下で実行される 4.x アプリケーションは、有効なデバイス番号であっても 255 より大きい値は認識できません。

動的リンクのデフォルト

静的にリンクされている実行可能ファイルの動的リンクを可能にするために、パッケージではデフォルトで動的リンクを使用可能にしています。この設定により、すべて静的にリンクされた SunOS 4.x 実行可能ファイルの性能が低下します。この機能を無効にするには、スーパーユーザになり、`/etc/system` ファイルの「`set:`」部分に次の行を追加してください。

```
set enable_mixed_bcp=0
```

第 3 章

ウィンドウシステム互換性

Solaris 2.x リリースにおいて、デフォルトのウィンドウシステムは、OpenWindows バージョン 3.3 です。SunOS 4.x のアプリケーションの多くは、OpenWindows の旧バージョンを基礎としています。この章では、ウィンドウシステムに関連して使用できるバイナリ互換性について説明します。ウィンドウベースのアプリケーションにも、第 2 章で述べたバイナリ互換性に関する制約が適用されます。

OpenWindows のバイナリ互換性

OpenWindows 環境でのバイナリ互換性は、使用するプロトコルとツールキットによって異なります。以下の項では、バイナリ互換性のある OpenWindows アプリケーションに関する標準ガイドラインを示し、X11 プロトコルおよび各種のツールキットに関して使用できるバイナリ互換性について説明します。

SunOS 4.x リリースでは、OpenWindows バージョン 2.0 (V2) および OpenWindows バージョン 3.0 (V3) が使用できました。Solaris 2.x には、OpenWindows バージョン 3.3 (V3.3) がデフォルトのウィンドウシステムとして組み込まれています。一般に、OpenWindows V3 で実行されていた OpenWindows V2 アプリケーションは、OpenWindows V3.3 でも実行できます。以下の項に示す制約および問題点は、OpenWindows V3.3 で実行される OpenWindows V2 アプリケーションすべてに適用されます。

OpenWindows アプリケーションのリンク

アプリケーションがライブラリをリンクする場合は、すべてのライブラリを動的にリンクする必要があります。

たとえば、OpenWindows V2 ライブラリに対して、`libxview` が動的にリンクされ、`libolgx` が静的にリンクされている場合、このアプリケーションは V3 上で実行できません。ユーザには、次のような `ld.so` エラーメッセージが表示されます。

```
ld.so: call to undefined procedure _olgx_xxx from 0xf77906ec
```

`libc` を除くすべてのライブラリが OpenWindows V2 に対して動的にリンクされているとすれば、次のようなエラーメッセージ (`libc` が動的にリンクされてないため) が出ます。

```
ld.so: call to undefined procedure _strdup from 0xf778ea30
```

あるアプリケーションの作成時に、OpenWindows V2 XView ソースを使用して修正し、そしてライブラリを動的にリンクしたとすれば、そのアプリケーションは V3 XView を用いて実行することはできません。修正した XView ソースを削除する必要があります。

X11

OpenWindows V3 では、X11 プロトコルと XLib (クライアントライブラリ) は、OpenWindows V2 との 100% の下位互換性を備えていました。どちらのリリースも、MIT の X11R4 プロトコルをサポートしています。しかし、OpenWindows V3.3 リリースには大量のバグ修正が含まれているため、OpenWindows V2 の場合よりプロトコルへの依存性が高くなっています。アプリケーションが、プロトコルとの互換性のない機能や属性に依存している場合は、そのアプリケーションは働かなくなります。

サポートされていないプロトコル・方式

NeWS

NeWS プロトコルは OpenWindows V3.3 ではサポートされていません。

SunView

SunView プロトコルは OpenWindows V3.3 ではサポートされていません。

Pixrects

Pixrects は OpenWindows V3.3 ではサポートされていません。

ツールキット

XView

XView バイナリ互換性はサポートされています。XView バイナリ互換性とは、OpenWindows V2 XView に対して動的にリンクされた任意の OpenWindows V2 アプリケーションを、 V3 XView ライブラリに接続し実行可能とすることです。ただし、XView バイナリ互換性には次の 2 つの例外があります。

- ドラッグ&ドロップ
- XView PostScript (XVPS)

XView 2.0 には、ドラッグ&ドロッププロトコルの変更からクライアントを保護するための一時 API が含まれていました。このインタフェースを使って「ドロップを受け取る」アプリケーションは、OpenWindows V3 のライブラリに対しても適正に実行されます。しかし、アプリケーションがプロトコルの変更を直接対処している場合は、ソースレベルでの変更が必要になります。

XVPS を使用する XView アプリケーションは、Solaris 2.x ではサポートされていません。

OLIT

OpenWindows V2 ライブラリによって書かれた OLIT アプリケーションは、OpenWindows V3.3 を使用した場合でも問題なく実行できますが、ドラッグ&ドロップサポートだけは例外です。OpenWindows V2 アプリケーションが、OpenWindows V3.3 に含まれるドラッグ&ドロップの機能を使用しようとしても正しく動作しません。つまりドロップが失敗します。カット&ペーストは、まったく問題なく働きます。ドラッグ&ドロッププロトコルに変更があったため、アプ

リケーションが引き続きドラッグ&ドロップをサポートするには、ソースレベルの変更が必要です。

TNT と Lite

TNT または Lite ツールキットのいずれかのバージョンを使って書かれたアプリケーションは、Solaris 2.x ではサポートされません。

DeskSet

OpenWindows V2 および OpenWindows V3 で使用できる DeskSet アプリケーションは、OpenWindows V3.3 でも使用できます。OpenWindows V2 または V3 の Deskset アプリケーションを Solaris 2.x で実行することは不要であり、サポートもされていません。

索引