Oracle® Solaris 11.1 での一般的な問題のトラブルシューティング

Part No: E36496–02
2012 年 10 月

ORACLE

Copyright © 1998, 2012, Oracle and/or its affiliates. All rights reserved.

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクル社までご連絡ください。

このソフトウェアまたは関連ドキュメントを、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供する場合は、次の通知が適用されます。

U.S. GOVERNMENT END USERS:

Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

このソフトウェアもしくはハードウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアもしくはハードウェアは、危険が伴うアプリケーション（人的傷害を発生させる可能性があるアプリケーションを含む）への用途を目的として開発されていません。このソフトウェアもしくはハードウェアを危険が伴うアプリケーションで使用する際、安全に使用するために、適切な安全装置、バックアップ、冗長性（redundancy）、その他の対策を講じることは使用者の責任となります。このソフトウェアもしくはハードウェアを危険が伴うアプリケーションで使用したことに起因して損害が発生しても、オラクル社およびその関連会社は一切の責任を負いかねます。

OracleおよびJavaはOracle Corporationおよびその関連企業の登録商標です。その他の名称は、それぞれの所有者の商標または登録商標です。

Intel、Intel Xeonは、Intel Corporationの商標または登録商標です。すべてのSPARCの商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD、Opteron、AMDロゴ、AMD Opteronロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIXは、The Open Groupの登録商標です。

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

121210@25097

目次

はじめに

『Oracle Solaris 11.1 での一般的な問題のトラブルシューティング』は、Oracle Solaris システム管理に関する重要な情報を提供するドキュメントセットの一部です。このガイドには、SPARC および x86 の両方のシステムに関する情報が含まれています。

本書は、読者が次のタスクを終了済みであることを前提としています。

- Oracle Solaris ソフトウェアのインストールが完了していること
- 使用する予定のすべてのネットワークソフトウェアを設定済み

システム管理者にとって重要と思われる Oracle Solaris の新機能については、各章の初めにある新機能に関するセクションを参照してください。

注 – この Oracle Solaris のリリースでは、SPARC および x86 系列のプロセッサアーキテクチャーを使用するシステムをサポートしています。サポートされるシステムは、Oracle Solaris OS: Hardware Compatibility Lists に記載されています。このドキュメントでは、プラットフォームにより実装が異なる場合は、それを特記します。

サポートされるシステムについては、Oracle Solaris OS: Hardware Compatibility Lists を参照してください。

対象読者

このマニュアルは、Oracle Solaris 11 リリースを実行している 1 つまたは複数のシステムの管理を行うユーザーを対象にしています。本書を使用するには、UNIX のシステム管理について 1 - 2 年の経験が必要です。UNIX システム管理のトレーニングコースに参加することも役に立ちます。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info を参照してください。聴覚に障害をお持ちの場合は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

表記上の規則

次の表では、このドキュメントで使用される表記上の規則について説明します。

表 P–1　表記上の規則

字体	説明	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.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。 `machine_name% you have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`machine_name%` **`su`** `Password:`
aabbcc123	プレースホルダ: 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。
AaBbCc123	書名、新しい単語、および強調する単語を示します。	『ユーザーズガイド』の第 6 章を参照してください。 キャッシュは、ローカルに格納されるコピーです。 ファイルを保存しないでください。 注: いくつかの強調された項目は、オンラインでは太字で表示されます。

コマンド例のシェルプロンプト

Oracle Solaris OS に含まれるシェルで使用する、UNIX のデフォルトのシステムプロンプトとスーパーユーザープロンプトを次に示します。コマンド例に示されるデフォルトのシステムプロンプトは、Oracle Solaris のリリースによって異なります。

表 P–2　シェルプロンプト

シェル	プロンプト
Bash シェル、Korn シェル、および Bourne シェル	`$`
Bash シェル、Korn シェル、および Bourne シェルのスーパーユーザー	`#`
C シェル	`machine_name%`
C シェルのスーパーユーザー	`machine_name#`

◆ ◆ ◆ 第 1 章

1 システムクラッシュ情報の管理 (タスク)

この章では、Oracle Solaris OS でシステムクラッシュ情報を管理する方法を説明します。

この章の内容は次のとおりです。

システムクラッシュ情報の管理に関する新機能

このセクションでは、Oracle Solaris でシステムリソースを管理するための新機能、または機能の変更について説明します。

savecore の動作の変更点

savecore コマンドでファイルが作成される際、最初はファイルに .partial 接尾辞が付加されるようになりました。ファイルが完全に書き込まれたあと、名前が変更され、接尾辞は削除されます。たとえば、savecore コマンドがまだビジー状態であるなど、潜在的な問題によってファイル名の変更や接尾辞の削除が妨げられることがあります。また、ブート直後のシステムクラッシュによって savecore コマンドが中断された場合も問題になります。

コマンドがビジー状態の場合は、ps コマンドを使用して実行中の savecore プロセスのプロセス ID (PID) を検索し、プロセスが完了するまで待ちます。プロセスが中断された場合は、残ったファイルを手動で削除してから、savecore コマンドを -d オプションで実行して再作成します。

詳細は、savecore(1M) のマニュアルページを参照してください。

システムクラッシュ (概要)

システムクラッシュ情報を処理する場合には、次の点に注意してください。

- システムクラッシュ情報にアクセスして管理するには、root 役割になる必要があります。『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。
- システム上でシステムクラッシュダンプを保存するオプションを無効にしないでください。システムクラッシュファイルにより、システムクラッシュの原因を判断する非常に有効な方法が提供されます。
- 重要なシステムクラッシュ情報は、カスタマサービス担当者に送信するまでは削除しないでください。

ハードウェアの障害、入出力の問題、ソフトウェアエラーなどが原因でシステムがクラッシュすることがあります。システムがクラッシュすると、システムはエラーメッセージをコンソールに表示し、物理メモリーのコピーをダンプデバイスに書き込みます。その後、システムは自動的にリブートします。システムがリブートすると、savecore コマンドが実行され、ダンプデバイスのデータを取り出して、保存されたクラッシュダンプを savecore ディレクトリに書き込みます。保存されたクラッシュダンプファイルは、問題を診断する上で貴重な情報となります。

クラッシュダンプ情報は圧縮した形式で vmdump.*n* ファイルに書き込まれます。この *n* は、クラッシュダンプ識別用の整数です。その後、同じシステムまたは別のシステムで savecore コマンドを呼び出して、圧縮されているクラッシュダンプを、unix.*n* および vmcore.*n* という名前の 1 組のファイルに展開できます。リブート時にクラッシュダンプが保存されるディレクトリも、dumpadm コマンドを使用して構成できます。

スワップ領域とダンプ領域には専用の ZFS ボリュームが使用されます。インストール後に、スワップデバイスやダンプデバイスのサイズの調整が必要になったり、場合によってはスワップボリュームやダンプボリュームの再作成が必要になることがあります。手順については、『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』の「ZFS スワップデバイスおよびダンプデバイスを管理する」を参照してください。

システムクラッシュダンプファイル

システムクラッシュのあとで自動的に実行される savecore コマンドは、ダンプデバイスからクラッシュダンプ情報を取り出し、unix.*x* と vmcore.*x* という 1 対のファイルを作成します。*x* はダンプの通し番号です。これらのファイルは 2 つで、保存されたシステムクラッシュダンプの情報を表します。

注 – クラッシュダンプファイルはコアファイルと混同されることがあります。コアファイルは、アプリケーションが異常終了したときに書き込まれるユーザーアプリケーションのイメージです。

クラッシュダンプファイルは、あらかじめ決められたディレクトリに保存されます。これはデフォルトでは /var/crash/ です。以前の リリースでは、システムを手動で有効にして物理メモリーのイメージをクラッシュダンプファイルに保存しない限り、システムがリブートされた時にクラッシュダンプファイルが上書きされていました。このリリースでは、クラッシュダンプファイルの保存がデフォルトで有効です。

システムクラッシュ情報は dumpadm コマンドで管理します。詳細は、11 ページの「dumpadm コマンドを使用したシステムクラッシュダンプ情報の管理」を参照してください。

クラッシュダンプの保存

制御構造体、アクティブなテーブル、動作中またはクラッシュしたシステムカーネルのメモリーのイメージなど、カーネルの動作状況についての情報を調べるには、mdb ユーティリティーを使用します。mdb ユーティリティーを完全に使いこなすには、カーネルについての詳細な知識が必要ですが、このマニュアルでは説明を省きます。このユーティリティーの使用法については、mdb(1) のマニュアルページを参照してください。

dumpadm コマンドを使用したシステムクラッシュダンプ情報の管理

Oracle Solaris OS でシステムクラッシュダンプ情報を管理するには、dumpadm コマンドを使用します。

- オペレーティングシステムのクラッシュダンプを構成することもできます。dumpadm 構成パラメータでは、ダンプ内容、ダンプデバイス、クラッシュダンプファイルが保存されるディレクトリなどを指定します。
- ダンプデータは、圧縮した形式でダンプデバイスに格納されます。カーネルのクラッシュダンプイメージは 4G バイトを超える場合があります。データを圧縮することにより、ダンプが速くなり、ダンプデバイスのディスク領域も少なくてすみます。
- スワップ領域ではなく、専用のダンプデバイスがダンプ構成の一部にあると、クラッシュダンプファイルの保存はバックグラウンドで行われます。つまり、システムをブートする際、savecore コマンドが完了するのを待たなくても、次の段階

に進むことができます。大容量のメモリーを搭載したシステムでは、savecore コマンドが完了する前にシステムが使用可能になります。潜在的な問題については、9 ページの「savecore の動作の変更点」を参照してください。

- savecore コマンドで生成されるシステムクラッシュダンプファイルは、デフォルトで保存されます。
- savecore -L コマンドを使用すると、動作中の Oracle Solaris OS でクラッシュダンプを取得できます。たとえば、パフォーマンスに問題が発生しているときやサービスが停止しているときなどにメモリーのスナップショットをとって、実行中のシステムの問題をトラブルシュートするのに使用します。システムが実行中で、一部のコマンドがまだ使用できる場合は、savecore -L コマンドを使用してシステムのスナップショットをダンプデバイスに保存し、クラッシュダンプファイルをただちに savecore ディレクトリに書き込むことができます。システムが実行中であるため、専用のダンプデバイスを構成してある場合のみ、savecore - L コマンドを使用できます。

ダンプ構成パラメータは、dumpadm コマンドで管理します。次の表に、dumpadm の構成パラメータを示します。

ダンプパラメータ	説明
ダンプデバイス	システムがクラッシュしたときにダンプデータを一時的に保存するデバイス。ダンプデバイスがスワップ領域でない場合は、savecore がバックグラウンドで実行されるため、ブートプロセスの速度が上がる
savecore ディレクトリ	システムのクラッシュダンプファイルを保存するディレクトリ
ダンプ内容	ダンプするメモリーデータの種類
最小空き容量	クラッシュダンプファイルを保存した後で savecore ディレクトリに必要な最小空き容量。空き容量を構成しないと、デフォルトで 1M バイトになる

詳細は、dumpadm(1M) のマニュアルページを参照してください。

dumpadm コマンドの動作

dumpadm コマンドは、システム起動時に svc:/system/dumpadm:default サービスによって呼び出されて、クラッシュダンプパラメータの構成を行います。

dumpadmコマンドは、/dev/dump インタフェースを通してダンプデバイスとダンプ内容を初期化します。

ダンプ構成が完了すると、savecore スクリプトは、クラッシュダンプファイルのディレクトリの場所を探します。次に、savecore を呼び出して、クラッシュダンプがあるかどうかを調べたり、クラッシュダンプディレクトリにある minfree ファイルの内容を確認したりします。

システムクラッシュダンプ情報の管理

このセクションでは、システムクラッシュダンプ情報を管理するためのタスクについて説明します。

システムクラッシュダンプ情報の管理 (タスクマップ)

タスク	説明	参照先
1. 現在のクラッシュダンプ構成を表示する	dumpadm コマンドを使用して、現在のクラッシュダンプ構成を表示する	14 ページの「現在のクラッシュダンプ構成を表示する方法」
2. クラッシュダンプ構成を変更する	dumpadm コマンドを使用して、ダンプするデータの種類、システムが専用のダンプデバイスを使用するかどうか、クラッシュダンプファイルを保存するディレクトリ、およびクラッシュダンプファイルが書き込まれた後に残っていなければならない容量を指定する	14 ページの「クラッシュダンプ構成を変更する方法」
3. クラッシュダンプファイルを調べる	mdb コマンドを使用して、クラッシュダンプファイルを表示する	16 ページの「クラッシュダンプ情報を検査する方法」
4.(オプション) クラッシュダンプディレクトリが一杯になった場合に復元する	システムがクラッシュした際、savecore ディレクトリに十分な空き容量がなくても、一部の重要なシステムクラッシュダンプ情報を保存したい場合	18 ページの「クラッシュダンプディレクトリが一杯になった場合に復元する方法 (オプション)」
5.(オプション) クラッシュダンプファイルの保存を有効または無効にする	dumpadm コマンドを使用して、クラッシュダンプファイルの保存を有効または無効にする。デフォルトでは、クラッシュダンプファイルは保存される	18 ページの「クラッシュダンプの保存を無効または有効にする方法」

▼ 現在のクラッシュダンプ構成を表示する方法

1 **root** 役割になります。

『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2 現在のクラッシュダンプ構成を表示します。

```
# dumpadm
 Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on
```

上記の出力例の意味は次のとおりです。

- ダンプの内容は、カーネルメモリーページである
- カーネルメモリーは、専用のダンプデバイス `/dev/zvol/dsk/rpool/dump` にダンプされます。
- システムクラッシュダンプファイルは `/var/crash/` ディレクトリに保存される
- システムクラッシュダンプファイルの保存は有効に設定されている
- クラッシュダンプを圧縮した形式で保存する

▼ クラッシュダンプ構成を変更する方法

1 **root** 役割になります。

『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2 現在のクラッシュダンプ構成を確認します。

```
# dumpadm
 Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on
```

この出力は、Oracle Solaris 11 リリースを実行するシステムのデフォルトダンプ構成を表しています。

3 クラッシュダンプ構成を変更します。

```
#  /usr/sbin/dumpadm  [-nuy] [-c content-type] [-d dump-device] [-m mink | minm | min%]
[-s savecore-dir] [-r root-dir] [-z on | off]
```

`-c` *content*	ダンプするデータの種類を指定する。すべてのカーネルメモリーをダンプするには `kernel` を、すべてのメモリーをダンプするには `all` を、カーネルメモリーとクラッシュ時に実行中だったスレッドを持つプロセスのメモリーページとをダンプするには `curproc` を使用する。デフォルトはカーネルメモリー
`-d` *dump-device*	システムがクラッシュしたときに、ダンプデータを一時的に保存するデバイスを指定する。デフォルトのダンプデバイスはプライマリダンプデバイスです。
`-m` *nnnk* | *nnnm* | *nnn%*	現在の `savecore` ディレクトリに `minfree` ファイルを作成することにより、クラッシュダンプファイルを保存する最小限の空き容量を指定する。このパラメータは K バイト (`nnnk`)、M バイト (`nnnm`)、またはファイルシステムサイズのパーセント (`nnn%`) で指定できる。`savecore` コマンドは、クラッシュダンプファイルを書き込む前にこのファイルを調べる。クラッシュダンプファイルを書き込むと空き容量が `minfree` の値より少なくなる場合、ダンプファイルは書き込まれず、エラーメッセージが記録される。このような問題を解決するには、18 ページの「クラッシュダンプディレクトリが一杯になった場合に復元する方法 (オプション)」を参照してください。
`-n`	システムがリブートするときに、`savecore` を実行しないように指定する。このダンプ構成は推奨できない。システムクラッシュ情報がスワップデバイスに書き込まれているときに、`savecore` が有効でないと、クラッシュダンプ情報はシステムがスワップを開始すると上書きされる
`-s`	クラッシュダンプファイルを保存する別のディレクトリを指定する。Oracle Solaris 11 では、デフォルトのディレクトリは `/var/crash` です。
`-u`	`/etc/dumpadm.conf` ファイルの内容に基づいてカーネルダンプ構成を強制的に更新します。
`-y`	リブート時に自動的に `savecore` コマンドを実行するようにダンプ構成を変更します。このダンプ設定では、このコマンドの自動実行がデフォルトです。
`-z on` | `off`	リブート時の `savecore` コマンドの動作を制御するために、ダンプ構成を変更します。`on` 設定では、圧縮した形式でのコアファイルの保存が有効になります。`off` 設定では、クラッシュダンプファイルを自動的に圧縮解除します。クラッシュダンプファイルはサイズが非常に大きくなる場合があり、圧縮した形式で保存すれば必要なシステム

領域が小さくなるため、デフォルトは on です。

例 1–1 クラッシュダンプ構成を変更する

次の例は、すべてのメモリーを専用のダンプデバイス /dev/zvol/dsk/rpool/dump にダンプします。また、クラッシュダンプファイルを保存したあとに残っていなければならない最小空き容量は、ファイルシステム容量の 10% です。

```
# dumpadm
      Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on

# dumpadm -c all -d /dev/zvol/dsk/rpool/dump -m 10%
       Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5697105KB)
  Savecore enabled: yes
   Save compressed: on
```

▼ クラッシュダンプ情報を検査する方法

1. **root** 役割になります。

   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2. クラッシュダンプ情報が保存されているディレクトリに移動します。例:

   ```
   # cd /var/crash
   ```

   クラッシュダンプの場所が不明な場合は、dumpadm コマンドを使用して、システムがカーネルクラッシュダンプファイルを格納するように構成されている場所を特定します。例:

   ```
   # /usr/sbin/dumpadm
         Dump content: kernel pages
          Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
   Savecore directory: /var/crash
     Savecore enabled: yes
      Save compressed: on
   ```

3. クラッシュダンプを検査するには、モジュラーデバッガユーティリティー **(mdb)** を使用します。

   ```
   # /usr/bin/mdb [-k] crashdump-file
   ```

`-k` オペレーティングシステムのクラッシュダンプファイルの場合のカーネルデバッグモードを指定します。

crashdump-file オペレーティングシステムのクラッシュダンプファイルを指定します。

例:

```
# /usr/bin/mdb -K vmcore.0
```

または、コマンドを次のように指定することもできます。

```
# /usr/bin/mdb -k 0
```

4 システムクラッシュのステータスを次のように表示します。

```
> ::status
   .
   .
   .
> ::system
   .
   .
   .
```

カーネルクラッシュダンプを検査するときに `::system dcmd` コマンドを使用するには、コアファイルはカーネルクラッシュダンプである必要があり、かつ、`mdb` ユーティリティーの起動時に `-k` オプションを指定しておく必要があります。

5 **mdb** ユーティリティーを終了します。

```
> $quit
```

例 1–2 クラッシュダンプ情報を検査する

次の例は、`mdb` ユーティリティーの出力例を示します。このシステムのシステム情報と `/etc/system` ファイルに設定されている調整可能パラメータが含まれています。

```
# cd /var/crash
# /usr/bin/mdb -k unix.0
Loading modules: [ unix krtld genunix ip nfs ipc ptm ]
> ::status
debugging crash dump /dev/mem (64-bit) from ozlo
operating system: 5.10 Generic sun4v
> ::system
set ufs_ninode=0x9c40 [0t40000]
set ncsize=0x4e20 [0t20000]
set pt_cnt=0x400 [0t1024]
> $q
```

▼ クラッシュダンプディレクトリが一杯になった場合に復元する方法 (オプション)

ここでは、システムがクラッシュしたが、十分な空き容量が savecore ディレクトリに残っておらず、それでも、一部の重要なシステムクラッシュダンプ情報を保存したい場合を考えます。

1. システムのリブート後、**root** 役割としてログインします。
2. すでにサービスプロバイダに送ってある既存のクラッシュダンプファイルを削除して、**savecore** ディレクトリ (通常は **/var/crash/**) を整理します。
   - 別の方法として、**savecore** コマンドを手作業で実行し、十分なディスク容量がある代替ディレクトリを指定することができます。

     # **savecore** [*directory*]

▼ クラッシュダンプの保存を無効または有効にする方法

1. **root** 役割になります。

   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。
2. システム上のクラッシュダンプの保存を有効または無効にします。

   # dumpadm -n | -y

例 1–3 クラッシュダンプの保存を無効にする

次の例は、システムでのクラッシュダンプの保存を無効にします。

```
# Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5697105KB)
  Savecore enabled: no
   Save compressed: on
```

例 1–4 クラッシュダンプの保存を有効にする

次の例は、システムでのクラッシュダンプの保存を有効にします。

```
# dumpadm -y
      Dump content: all pages
```

```
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5697105KB)
  Savecore enabled: yes
   Save compressed: on
```

◆ ◆ ◆ 第 2 章

2 コアファイルの管理 (タスク)

この章では、coreadm コマンドを使って、コアファイルを管理する方法について説明します。

この章の内容は次のとおりです。

コアファイルの管理

コアファイルは、プロセスまたはアプリケーションが異常終了した場合に生成されます。コアファイルは coreadm コマンドで管理します。たとえば、coreadm コマンドを使用して、プロセスコアファイルをすべて同じシステムディレクトリに置くようにシステムを構成できます。プロセスやデーモンが異常終了した場合に、特定のディレクトリにあるコアファイルを調べればよいため、問題の追跡が容易になります。

構成可能なコアファイルのパス

次の 2 つの構成可能なコアファイルのパスは、個別に有効または無効にすることができます。

- プロセス別コアファイルのパスにはデフォルトで core が使用されます。このパスはデフォルトで有効になっています。プロセス別コアファイルのパスが有効になっていると、プロセスが異常終了したときにコアファイルが生成されます。プロセス別のパスは、親プロセスから新しいプロセスに継承されます。

  プロセス別コアファイルは生成されるとプロセスの所有者によって所有され、所有者には読み取り/書き込み権が与えられます。所有者だけがこのファイルを表示できます。

- グローバルコアファイルのパスにはデフォルトで core が使用されます。このパスはデフォルトで無効になっています。このパスが有効になっていると、プロセス別コアファイルのパスと同じ内容のコアファイルがグローバルコアファイルのパスに追加で作成されます。

  グローバルコアファイルは生成されると root によって所有され、root だけに読み取り/書き込み権が与えられます。アクセス権のないユーザーはこのファイルを表示できません。

プロセスが異常終了すると、コアファイルがデフォルトで現在のディレクトリに作成されます。グローバルコアファイルのパスが有効になっていると、プロセスが終了するたびにコアファイルが 2 つ、1 つは現在の作業ディレクトリに、1 つはグローバルコアファイルのディレクトリにそれぞれ作成されます。

デフォルトでは、setuid プロセスは、グローバルの設定やプロセス別のパスを使ってコアファイルを生成することはありません。

拡張されたコアファイル名

グローバルコアファイルディレクトリが有効な場合、次の表に示す変数を使ってコアファイルを相互に区別できます。

変数名	変数の定義
%d	実行ファイルのディレクトリ名。最大文字数は MAXPATHLEN
%f	実行ファイルの名前。最大文字数は MAXCOMLEN
%g	実効グループ ID
%m	マシン名 (uname -m)
%n	システムノード名 (uname -n)
%p	プロセス ID
%t	time(2) の 10 進数
%u	実効ユーザー ID
%z	プロセスが実行されているゾーン名 (zonename)
%%	リテラル %

たとえば、グローバルコアファイルパスが次のように設定されている場合、

```
/var/core/core.%f.%p
```

PID 12345 の sendmail プロセスが異常終了すると、次の コアファイルが作成されます。

```
/var/core/core.sendmail.12345
```

コアファイル名パターンの設定

コアファイル名パターンは、グローバル、ゾーン別、またはプロセス別に設定できます。さらに、システムリブート後も有効なプロセス別デフォルトを設定できます。

たとえば、次の coreadm コマンドでは、デフォルトのプロセス別コアファイルパターンを設定します。この設定は、デフォルトのコアファイルパターンを明示的に上書きしていないプロセスに対して適用されます。この設定はシステムリブート後も有効です。たとえば、次の coreadm コマンドで、init プロセスによって開始されるすべてのプロセスのグローバルコアファイルパターンを設定します。このパターンはシステムリブート後も有効です。

```
# coreadm -i /var/core/core.%f.%p
```

次の coreadm コマンドでは、任意のプロセスに対しプロセス別コアファイル名パターンを設定します。

```
# coreadm -p /var/core/core.%f.%p $$
```

$$ 記号には、現在実行中のシェルのプロセス ID を指定します。プロセス別コアファイル名パターンは、すべての子プロセスに継承されます。

グローバルまたはプロセス別のコアファイル名パターンを設定したら、これを coreadm -e コマンドで有効にする必要があります。詳細については次の手順を参照してください。

このコマンドをユーザーの初期設定ファイル (たとえば .profile) に入れておけば、ユーザーのログインセッションで実行するすべてのプロセスに対しコアファイル名パターンを設定できます。

setuid プログラムがコアファイルを作成できるようにする

coreadm コマンドを使って setuid プログラムを有効または無効にすれば、次のパス設定を行うことによって、すべてのシステムプロセスに対して、または各プロセスに対してコアファイルを作成できます。

- グローバル setuid オプションが有効になっていると、グローバルコアファイルパスに従って、システムのすべての setuid プログラムがコアファイルを作成します。
- プロセス別 setuid オプションが有効になっていると、プロセス別コアファイルパスに従って、特定の setuid プロセスがコアファイルを作成します。

デフォルトでは、両方のフラグが無効になっています。セキュリティー上の理由により、グローバルコアファイルパスは、/ で始まるフルパス名であることが必要です。root がプロセス別コアファイルを無効にすると、個別のユーザーがコアファイルを得ることはできなくなります。

setuid コアファイルは root によって所有され、root だけに読み取り/書き込み権が与えられます。通常ユーザーは、たとえ setuid コアファイルを生成したプロセスを所有していても、それらのファイルにアクセスできません。

詳細は、coreadm(1M) のマニュアルページを参照してください。

コアファイルの管理 (タスクマップ)

タスク	説明	参照先
1. 現在のコアダンプ構成を表示する	coreadm コマンドを使用して、現在のコアダンプ構成を変更する	25 ページの「現在のコアダンプ構成の表示」
2. コアダンプ構成を変更する	次のいずれかの手順を実行して、コアダンプ構成を変更する ■ コアファイル名パターンを設定する ■ プロセス別コアファイルのパスを有効にする ■ グローバルのコアファイルのパスを有効にする	25 ページの「コアファイル名パターンを設定する方法」 25 ページの「プロセス別コアファイルパスを有効にする方法」 26 ページの「グローバルのコアファイルパスを有効にする方法」
3. コアダンプファイルを調べる	proc ツールを使用して、コアダンプファイルを表示する	26 ページの「コアファイルの調査」

現在のコアダンプ構成の表示

現在のコアダンプ構成を表示するには、オプションを指定しないで coreadm コマンドを実行します。

```
$ coreadm
              global core file pattern:
     global core file content: default
       init core file pattern: core
       init core file content: default
            global core dumps: disabled
      per-process core dumps: enabled
     global setid core dumps: disabled
per-process setid core dumps: disabled
     global core dump logging: disabled
```

▼ コアファイル名パターンを設定する方法

- プロセス別コアファイルを設定するのか、グローバルコアファイルを設定するのかを決めて、次のどちらかの手順に従います。

  a. プロセス別コアファイル名パターンを設定します。

  ```
  $ coreadm -p $HOME/corefiles/%f.%p $$
  ```

  b. **root** 役割になります。

  c. グローバルコアファイル名パターンを設定します。

  ```
  # coreadm -g /var/corefiles/%f.%p
  ```

▼ プロセス別コアファイルパスを有効にする方法

1. **root** 役割になります。

   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2. プロセス別コアファイルのパスを有効にする

   ```
   # coreadm -e process
   ```

3. 現在のプロセスのコアファイルパスを表示して構成を確認します。

   ```
   # coreadm $$
   1180:   /home/kryten/corefiles/%f.%p
   ```

▼ グローバルのコアファイルパスを有効にする方法

1. **root** 役割になります。

   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2. グローバルのコアファイルのパスを有効にする

   ```
   # coreadm -e global -g /var/core/core.%f.%p
   ```

3. 現在のプロセスのコアファイルパスを表示して構成を確認します。

   ```
   # coreadm
          global core file pattern: /var/core/core.%f.%p
        global core file content: default
          init core file pattern: core
          init core file content: default
               global core dumps: enabled
          per-process core dumps: enabled
         global setid core dumps: disabled
    per-process setid core dumps: disabled
        global core dump logging: disabled
   ```

コアファイルのトラブルシューティング

エラーメッセージ

```
NOTICE: 'set allow_setid_core = 1' in /etc/system is obsolete
NOTICE: Use the coreadm command instead of 'allow_setid_core'
```

エラーの発生原因

`setuid` コアファイルを許容する古いパラメータが `/etc/system` ファイルにあります。

解決方法

`/etc/system` ファイルから `allow_setid_core=1` を削除します。次に `coreadm` コマンドを使って、 グローバル `setuid` コアファイルパスを有効にします。

コアファイルの調査

`proc` ツールを使用して、プロセスのコアファイルやライブプロセスを調べることができます。`proc` ツールは、`/proc` ファイルシステムの機能を操作するユーティリティーです。

コアファイルを処理できるツールは /usr/proc/bin ディレクトリにある `pstack`、`pmap`、`pldd`、`pflags`、`pcred` です。これらのツールを使用するには、プロセス ID を指定するように、コアファイルの名前をコマンド行に指定します。

proc ツールを使用してコアファイルを調べる方法については、proc(1) のマニュアルページを参照してください。

例 2–1 proc ツールを使用してコアファイルを調べる

```
$ ./a.out
Segmentation Fault(coredump)
$ /usr/proc/bin/pstack ./core
core './core' of 19305: ./a.out
 000108c4 main     (1, ffbef5cc, ffbef5d4, 20800, 0, 0) + 1c
 00010880 _start   (0, 0, 0, 0, 0, 0) + b8
```

◆ ◆ ◆ 第 3 章

3 システムおよびソフトウェアのトラブルシューティング (タスク)

この章では、ソフトウェアのトラブルシューティングの概要について説明します。これには、システムクラッシュのトラブルシューティング、クラッシュダンプ情報の管理、およびシステムメッセージの表示と管理が含まれます。

この章の内容は次のとおりです:

システムクラッシュの問題のトラブルシューティング

Oracle Solaris が動作しているシステムがクラッシュした場合は、クラッシュダンプファイルを含む、可能なかぎりの情報を購入先に提供してください。

システムがクラッシュした場合の対処方法

次に、システムクラッシュが発生した場合に留意すべきもっとも重要な点について説明します。

1. システムのコンソールメッセージを書き取ります。
   - システムがクラッシュした場合は、システムを再稼動させるのを最優先に考えがちです。しかし、システムをリブートする前に、コンソール画面のメッセージを確認してください。 これらのメッセージは、クラッシュした原因を解明するのに役立ちます。システムが自動的にリブートして、コンソールメッセージが画面から消えた場合でも、システムエラーログファイル `/var/adm/messages` を表示すれば、これらのメッセージをチェックできます。システムエラーログファイルを表示する方法については、33 ページの「システムメッセージを表示する方法」を参照してください。

- クラッシュが頻繁に発生し、その原因を特定できない場合は、システムコンソールまたは `/var/adm/messages` ファイルからできるだけ多くの情報を収集し、購入先に問い合わせます。購入先に問い合わせるために収集しておくトラブルシューティング情報の完全なリストについては、29 ページの「システムクラッシュの問題のトラブルシューティング」を参照してください。

2. また、システムクラッシュダンプがシステムのクラッシュ後に生成されたかどうかを確認してください。デフォルトでは、システムクラッシュダンプが保存されます。クラッシュダンプについては、第 1 章「システムクラッシュ情報の管理 (タスク)」を参照してください。
3. システムクラッシュ後にブートが失敗する場合は、『Oracle Solaris 11.1 システムのブートおよびシャットダウン』の「復旧目的のシステムのシャットダウンおよびブート」を参照してください。

トラブルシューティングデータの収集

システムの問題を特定するために、次の質問に答えてください。31 ページの「システムクラッシュをトラブルシュートするためのチェックリスト」を使用し、クラッシュしたシステムのトラブルシューティングデータを収集します。

表 3–1　システムクラッシュに関するデータの収集

質問	説明
問題を再現できるか	この質問は、再現可能なテストケースは実際のハードウェア問題をデバッグするために重要であることが多いために重要である。購入先では、特殊な計測機構を使用してカーネルを構築して問題を再現し、バグを引き起こし、診断、および修正できる
Sun 以外のドライバを使用しているか	ドライバは、カーネルと同じアドレス空間で、カーネルと同じ特権で動作する。したがって、ドライバにバグがあると、システムクラッシュの原因となることがある
クラッシュの直前にシステムは何を実行していたか	システムが通常でないこと (新しい負荷テストの実行など) を行なったり、通常よりも高いロードがシステムにかかったりした場合、クラッシュの原因となることがある
クラッシュ直前に、異常なコンソールメッセージが表示されたか	システムがクラッシュする前には、なんらかの兆候を示していることがある。この情報は重要
`/etc/system` ファイルに調整パラメータを追加したか	調整パラメータは、システムクラッシュの原因となることがある。たとえば、共有メモリーセグメントを増やした結果、システムが限度以上の多くのメモリーを割り当てようとした

表 3–1 システムクラッシュに関するデータの収集 (続き)

質問	説明
問題は最近発生するようになったか	そうであれば、問題の原因は、システムの変更 (たとえば、新しいドライバ、新しいソフトウェア、作業負荷の変化、CPU のアップグレード、メモリーのアップグレードなど) にある可能性がある

システムクラッシュをトラブルシュートするためのチェックリスト

クラッシュしたシステムの問題を解決するためのデータを収集するときは、次のチェックリストを使用します。

項目	ユーザーのデータ
システムクラッシュダンプがあるか	
オペレーティングシステムのリリースと適切なソフトウェアアプリケーションのリリースレベルを確認する	
システムのハードウェアを確認する	
SPARC システムの `prtdiag` 出力を含める他のシステムの Explorer 出力を含める	
パッチはインストールされているか。そうであれば、`showrev -p` 出力を含める	
問題を再現できるか	
Sun 以外のドライバをシステムで使用しているか	
クラッシュ直前のシステムの動作は	
クラッシュ直前に、異常なコンソールメッセージが表示されたか	
`/etc/system` ファイルにパラメータを追加したか	
問題は最近発生するようになったか	

システムメッセージの管理

以降のセクションでは、Oracle Solaris のシステムメッセージ機能について説明します。

システムメッセージの表示

システムのメッセージはコンソールデバイスに表示されます。ほとんどのシステムメッセージは次の形式で表示されます。

[ID *msgid facility. priority*]

例:

```
[ID 672855 kern.notice] syncing file systems...
```

カーネルから出されるメッセージには、カーネルモジュール名が次のように表示されます。例:

```
Oct 1 14:07:24 mars ufs: [ID 845546 kern.notice] alloc: /: file system full
```

システムがクラッシュすると、システムのコンソールに次のようなメッセージが表示されることがあります。

`panic:` *error message*

まれに、パニックメッセージではなく次のメッセージが表示されることがあります。

```
Watchdog reset !
```

エラー記録デーモン `syslogd` は、さまざまなシステムの警告やエラーをメッセージファイルに自動的に記録します。デフォルトでは、これらのシステムメッセージの多くは、システムコンソールに表示されて、`/var/adm` ディレクトリに格納されます。システムメッセージ記録を設定することによって、これらのメッセージを格納する場所を指示できます。詳しくは、35 ページの「システムのメッセージ記録のカスタマイズ」を参照してください。これらのメッセージは、失敗の予兆のあるデバイスなど、システム障害をユーザーに警告できます。

`/var/adm` ディレクトリには、いくつかのメッセージファイルが含まれています。もっとも新しいメッセージは、`/var/adm/messages` (および `messages.*`) にあり、もっとも古いメッセージは、`messages.3` にあります。一定の期間 (通常は 10 日) ごとに、新しい `messages` ファイルが作成されます。`messages.0` のファイル名は `messages.1` に、`messages.1` は `messages.2` に、`messages.2` は `messages.3` にそれぞれ変更されます。その時点の `/var/adm/messages.3` は削除されます。

/var/adm ディレクトリは、メッセージやクラッシュダンプなどのデータを含む大きなファイルを格納するため、多くのディスク容量を消費します。/var/adm ディレクトリが大きくならないようにするために、そして将来のクラッシュダンプが保存できるようにするために、不要なファイルを定期的に削除しなければなりません。crontab ファイルを使用すれば、このタスクは自動化できます。このタスクの自動化については、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「クラッシュダンプファイルを削除する方法」および『Oracle Solaris 11.1 でのシステム情報、プロセス、およびパフォーマンスの管理』の第 4 章「システムタスクのスケジュール設定 (タスク)」を参照してください。

▼ システムメッセージを表示する方法

- システムクラッシュまたはリブートによって生成された最近のメッセージを表示するには、**dmesg** コマンドを使用します。

```
$ dmesg
```

あるいは、more コマンドを使用して、メッセージを 1 画面ごとに表示します。

```
$ more /var/adm/messages
```

例 3–1 システムメッセージの表示

次の例は、Oracle Solaris 10 システムでの dmesg コマンドからの出力を示します。

```
$ dmesg
Mon Sep 13 14:33:04 MDT 2010
Sep 13 11:06:16 sr1-ubrm-41 svc.startd[7]: [ID 122153 daemon.warning] ...
Sep 13 11:12:55 sr1-ubrm-41 last message repeated 398 times
Sep 13 11:12:56 sr1-ubrm-41 svc.startd[7]: [ID 122153 daemon.warning] ...
Sep 13 11:15:16 sr1-ubrm-41 last message repeated 139 times
Sep 13 11:15:16 sr1-ubrm-41 xscreensaver[25520]:  ,,,
Sep 13 11:15:16 sr1-ubrm-41 xscreensaver[25520]: ...
Sep 13 11:15:17 sr1-ubrm-41 svc.startd[7]: [ID 122153 daemon.warning]...
.
.
.
```

参照 詳細は、dmesg(1M) のマニュアルページを参照してください。

システムログローテーション

システムログファイルは、root の crontab ファイルのエントリから logadm コマンドによって実行されます。/usr/lib/newsyslog スクリプトは使用されません。

このシステムログローテーションは、/etc/logadm.conf ファイルに定義されます。このファイルには、syslogd などのプロセスのログローテーションエントリが含まれています。たとえば、/etc/logadm.conf ファイルにある 1 つのエントリ

は、/var/log/syslog ファイルが空でなければローテーションが毎週実行されることを示しています。つまり、最新の syslog ファイルが syslog.0 になり、その次に新しい syslog ファイルが syslog.1 になります。最新からさかのぼって 8 つまでの syslog ログファイルが保存されます。

また、/etc/logadm.conf ファイルには、最後のログローテーション実行時のタイムスタンプも含まれます。

logadm コマンドを使用して、必要に応じてシステムログをカスタマイズしたり、/etc/logadm.conf ファイルにログを追加したりすることができます。

たとえば、Apache アクセスとエラーログのローテーションを実行するには、次のコマンドを使用します。

```
# logadm -w /var/apache/logs/access_log -s 100m
# logadm -w /var/apache/logs/error_log -s 10m
```

この例では、Apache の access_log ファイルのローテーションは、そのサイズが 100M バイトに達したときに実行され、そのファイル名に .0、 .1 などのように接尾辞が付けられます。また、古い access_log ファイルのコピーが 10 個保存されます。また、error_log のローテーションは、そのサイズが 10M バイトに達したときに実行され、access_log ファイルと同様に、接尾辞が付けられ、コピーが保存されます。

前述の Apache ログローテーションの例における /etc/logadm.conf エントリの例は、次のようになります。

```
# cat /etc/logadm.conf
.
.
.
/var/apache/logs/error_log -s 10m
/var/apache/logs/access_log -s 100m
```

詳細は、logadm(1M) のマニュアルページを参照してください。

スーパーユーザーでログインするか、同等の役割 (ログ管理の権限を持つ) でアクセスすることによって、logadm コマンドを使用できます。RBAC を使用すると、logadm コマンドへのアクセス権を与えることによって、root 以外のユーザーにログファイルを管理する権限を与えることができます。

たとえば、次のエントリを /etc/user_attr ファイルに追加すれば、logadm コマンドを使用する権限がユーザー andy に与えられます。

```
andy::::profiles=Log Management
```

システムのメッセージ記録のカスタマイズ

/etc/syslog.conf ファイルを変更すると、さまざまなシステムプロセスが生成するさらに多くのエラーメッセージを記録できます。デフォルトでは、/etc/syslog.conf は、多くのシステムプロセスのメッセージが /var/adm/messages ファイルに格納されるように指示します。クラッシュとブートのメッセージも、同様にこのファイルに格納されます。/var/adm メッセージを表示する方法については、33 ページの「システムメッセージを表示する方法」を参照してください。

/etc/syslog.conf ファイルは、タブで区切られた 2 つの列から構成されています。

facility.level ... action

facility.level 機能またはメッセージや状態のシステムでの出所。コンマで区切られた機能のリスト。機能の値については、表 3–2 を参照してください。*level* は、記録する状態の重要度や優先順位を示します。優先レベルについては、表 3–3 を参照してください。

同じ機能の 2 つのエントリは、それぞれの優先順位が異なる場合、同じ行に入力しないでください。syslog ファイルに優先順位を入力すると、この優先順位以上のすべてのメッセージが記録され、最後のメッセージが優先されます。指定の機能とレベルに対し、syslogd はそのレベル以上のすべてのメッセージを記録します。

action 動作フィールドは、メッセージが転送される場所を示します。

次の例は、デフォルトの /etc/syslog.conf ファイルのサンプルを示します。

```
user.err                                        /dev/sysmsg
user.err                                        /var/adm/messages
user.alert                                      ‘root, operator’
user.emerg                                      *
```

この例は、次のユーザーメッセージが自動的に記録されることを意味します。

- ユーザーエラーはコンソールに出力され、/var/adm/messages ファイルにも記録されます。
- 早急な対応が必要なユーザーメッセージ (alert) は、root ユーザーと operator ユーザーに送信されます。
- ユーザー緊急メッセージは、各ユーザーに送信されます。

注 – エントリを個別の行に入力すると、/etc/syslog.conf ファイルでログの対象が複数回指定された場合に、メッセージのログ順が変わることがあります。単独行のエントリに複数のセレクタを指定できます。その際、セレクタはセミコロンで区切ります。

一般的なエラー状態の送信元を次の表に示します。一般的な優先順位を、重要度順に表 3–3 に示します。

表 3–2 syslog.conf メッセージの送信元の機能

送信元	説明
kern	カーネル
auth	認証
デーモン	すべてのデーモン
mail	メールシステム
lp	スプールシステム
user	ユーザープロセス

注 – /etc/syslog.conf ファイルで有効化できる syslog 機能の数に制限はありません。

表 3–3 syslog.conf メッセージの優先レベル

優先順位	説明
emerg	システムの緊急事態
alert	すぐに修正が必要なエラー
crit	致命的なエラー
err	その他のエラー
info	情報メッセージ
debug	デバッグ用の出力
none	この設定は出力を記録しない

▼ システムのメッセージ記録をカスタマイズする方法

1 **root** 役割になるか、**solaris.admin.edit/etc/syslog.conf** 承認が割り当てられている役割になります。

『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2 **pfedit** コマンドを使用して **/etc/syslog.conf** ファイルを編集します。**syslog.conf(4)** で説明している構文に従って、メッセージの送信元、優先順位、およびメッセージの格納場所を追加または変更します。

```
$ pfedit /etc/syslog.conf
```

3 変更を保存します。

例 3–2 システムのメッセージ記録のカスタマイズ

次の /etc/syslog.conf の user.emerg 機能の例は、ユーザー緊急メッセージを root ユーザーおよび個別のユーザーに送信します。

```
user.emerg                                        'root, *'
```

リモートコンソールメッセージングを有効にする

次の新しいリモートコンソール機能を使うと、リモートシステムの問題をトラブルシュートしやすくなります。

- consadm コマンドでは、補助 (またはリモート) コンソールとしてシリアルデバイスを選択できます。consadm コマンドを使用すると、システム管理者は 1 つまたは複数のシリアルポートを構成して、出力先が変更されたコンソールメッセージを表示したり、システムの実行レベルが変わったときに sulogin セッションをサポートしたりできます。この機能を使用して、モデム付きのシリアルポートにダイヤルインしてコンソールメッセージを監視し、init 状態の変更を表示できます(詳細については、sulogin(1M) と、以下の詳しい手順を参照)。

  補助コンソールとして構成されたポートからシステムにログインすることもできますが、このポートは主に、デフォルトコンソールに表示される情報を表示する出力デバイスです。ブートスクリプトやその他のアプリケーションがデフォルトコンソールに対して読み書きを行う場合、書き込み出力はすべての補助コンソールに出力されますが、入力はデフォルトコンソールからだけ読み込まれます対話型ログインセッションでの consadm コマンドの使用方法については、39 ページの「対話型ログインセッションで consadm コマンドを使用するためのガイドライン」を参照してください。
- コンソール出力は、新しい仮想デバイス /dev/sysmsg に書き込まれる、カーネルメッセージと syslog メッセージからなります。さらに、rc スクリプト起動メッセージが /dev/msglog に書き込まれます。以前のリリースでは、これらのメッセージはすべて /dev/console に書き込まれていました。

  スクリプトメッセージを補助コンソールに表示したい場合は、コンソール出力を /dev/console に出力しているスクリプトで出力先を /dev/msglog に変更する必要があります。メッセージ出力先を補助デバイスに変更したい場合は、/dev/console を参照しているプログラムで syslog() または strlog() を使用するように明示的に変更してください。

- consadm コマンドは、デーモンを実行して補助コンソールデバイスを監視します。補助コンソールに指定された表示デバイスがハングアップしたりキャリア信号がなくなって切り離されると、そのデバイスは補助コンソールデバイスのリストから削除され、アクティブでなくなります。1 つまたは複数の補助コンソールを有効にしても、メッセージがデフォルトコンソールに表示されなくなるわけではありません。メッセージは引き続き /dev/console に表示されます。

実行レベルの変更中に補助コンソールメッセージングを使用する

実行レベルの変更中に補助コンソールメッセージングを使う場合は、次の点に注意してください。

- システムのブート時に実行する rc スクリプトにユーザーの入力がある場合は、補助コンソールから入力を行うことはできません。入力はデフォルトコンソールから行う必要があります。
- 実行レベルの変更中に、スーパーユーザーパスワード入力を要求するために sulogin プログラムが init によって呼び出されます。このプログラムは、デフォルトのコンソールデバイスだけでなく各補助デバイスにもスーパーユーザーパスワードの入力要求を送信するように変更されています。
- システムがシングルユーザーモードで動作し、1 つまたは複数の補助コンソールが consadm コマンドによって有効になっていると、最初のデバイスでコンソールログインセッションが実行され、正確なスーパーユーザーパスワードを要求する sulogin プロンプトが表示されます。コンソールデバイスから正しいパスワードを受け取ると、sulogin は他のすべてのコンソールデバイスからの入力を受信できないようにします。
- コンソールの 1 つがシングルユーザー特権を取得すると、デフォルトコンソールとその他の補助コンソールにメッセージが出力されます。このメッセージは、どのデバイスから正しいスーパーユーザーパスワードが入力され、コンソールになったかを示します。シングルユーザーシェルが動作する補助コンソールのキャリア信号が失われると、次のどちらかのアクションが起ることがあります。
  - 補助コンソールが実行レベル 1 のシステムを表している場合は、システムはデフォルトの実行レベルに移行します。
  - 補助コンソールが実行レベル S のシステムを表している場合は、シェルから init s または shutdown コマンドが入力されたデバイスに「ENTER RUN LEVEL (0-6, s or S): 」というメッセージが表示されます。このデバイスのキャリア信号も失われている場合は、キャリア信号を復活して正確な実行レベルを入力する必要があります。init コマンドや shutdown コマンドを実行しても、実行レベルプロンプトが再表示されることはありません。
- シリアルポートを使用してシステムにログインしている場合には、init または shutdownコマンドを使用して別の実行レベルに移行すると、このデバイスが補助コンソールかどうかに関係なくログインセッションは失われます。この状況は、補助コンソール機能がない リリースと同じです。

- consadm コマンドを使って補助コンソールにするデバイスを選択すると、システムをリブートするか補助コンソールの選択を解除するまで、そのデバイスは補助コンソールとして有効です。ただし、consadm コマンドには、システムリブート後も同じデバイスを補助コンソールとして使用するオプションがあります(以下の詳しい手順を参照)。

対話型ログインセッションで consadm コマンドを使用するためのガイドライン

シリアルポートに接続された端末からシステムにログインしてから、 consadm コマンドを使ってこの端末にコンソールメッセージを表示して、対話型ログイン セッションを行う場合、次の点に注意してください。

- この端末で対話型ログインセッションを行う場合、補助コンソールがアクティブだと、コンソールメッセージは /dev/sysmsg デバイスまたは /dev/msglog デバイスに送られます。
- この端末からコマンドを発行すると、入力はデフォルトコンソール (/dev/console) ではなく対話型セッションに送られます。
- init コマンドを実行して実行レベルを変更すると、リモートコンソールソフトウェアは対話型セッションを終了し、sulogin プログラムを実行します。この時点では、入力はこの端末からだけ可能で、入力はコンソールデバイスから行われたかのように扱われます。そのため、38 ページの「実行レベルの変更中に補助コンソールメッセージングを使用する」 の説明のとおりに、sulogin プログラムにパスワードを入力できます。

  次に、(補助) 端末から正しいパスワードを入力すると、補助コンソールは、対話型 sulogin セッションを実行し、デフォルトコンソールおよび競合する補助コンソールを使えなくします。つまり、その端末は実質的にシステムコンソールとして機能します。
- この端末から実行レベル 3 または別の実行レベルに変更できます。実行レベルを変更すると、すべてのコンソールデバイスで sulogin が再び実行されます。終了したり、システムが実行レベル 3 で起動されるように指定すると、どの補助コンソールからも入力を行えなくなります。すべての補助コンソールはコンソールメッセージを表示するだけのデバイスに戻ります。

  システムが起動する際には、デフォルトのコンソールデバイスから rc スクリプトに情報を入力する必要があります。システムが再び起動すると login プログラムがシリアルポートで実行されるため、別の対話型セッションを開始できます。そのデバイスを補助コンソールに指定していれば、コンソールメッセージはその端末に引き続き出力されます。ただし、端末からの入力はすべて対話型セッションに送られます。

▼ 補助 (リモート) コンソールを有効にする方法

consadm デーモンは、consadm コマンドで補助コンソールを追加するまでポートの監視を開始しません。セキュリティー機能として、コンソールメッセージは、キャリア信号が失われるまでか、補助コンソールデバイスの選択が解除されるまでの間だけ出力変更されます。そのため、consadm コマンドを使うには、そのポートでキャリア信号が確立されている必要があります。

補助コンソールの有効化については、consadm(1m) のマニュアルページを参照してください。

1. システムにログインし、**root** 役割になります。

   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2. 補助コンソールを有効にします。

   ```
   # consadm -a devicename
   ```

3. 現在の接続が補助コンソールであることを確認します。

   ```
   # consadm
   ```

例 3–3 補助 (リモート) コンソールを有効にする

```
# consadm -a /dev/term/a
# consadm
 /dev/term/a
```

▼ 補助コンソールのリストを表示する方法

1. システムにログインし、**root** 役割になります。

   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。

2. 次のどちらかの手順に従います。

   a. 補助コンソールのリストを表示します。

      ```
      # consadm
      /dev/term/a
      ```

   b. 持続的補助コンソールのリストを表示します。

      ```
      # consadm -p
      /dev/term/b
      ```

▼ システムリブート後も補助 (リモート) コンソールを有効にする方法

1. システムにログインし、**root** 役割になります。
   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。
2. 複数のシステムリブート後も補助コンソールを有効にします。
   # consadm -a -p *devicename*
   このデバイスが持続的な補助コンソールのリストに追加されます。
3. デバイスが持続的な補助コンソールのリストに追加されているか確認します。
   # consadm

例 **3–4** システムリブート後も補助 (リモート) コンソールを有効にする

```
# consadm -a -p /dev/term/a
# consadm
/dev/term/a
```

▼ 補助 (リモート) コンソールを無効にする方法

1. システムにログインし、**root** 役割になります。
   『Oracle Solaris 11.1 の管理: セキュリティーサービス』の「割り当てられている管理権限を使用する方法」を参照してください。
2. 次のどちらかの手順に従います。
   a. 補助コンソールを無効にします。
      # consadm -d *devicename*
      または
   b. 補助コンソールを無効にし、持続的な補助コンソールのリストから削除します。
      # consadm -p -d *devicename*
3. 補助コンソールが無効になっていることを確認します。
   # consadm

例 **3–5** 補助 (リモート) コンソールを無効にする

```
# consadm -d /dev/term/a
# consadm
```

ファイルアクセスでの問題のトラブルシューティング

以前は使用できていたプログラム、ファイル、またはディレクトリにアクセスできないため、システム管理者に問い合わせる場合があります。

このようなときは、次の 3 点を調べてください。

- ユーザーの検索パスが変更されているか、または検索パス中のディレクトリが適切な順序であるか
- ファイルまたはディレクトリに適切なアクセス権や所有権があるか
- ネットワーク経由でアクセスするシステムの構成が変更されているか

この章では、これらの 3 点を確認する方法を簡単に説明して、可能な解決策を提案します。

検索パスに関連する問題を解決する (コマンドが見つかりません)

「コマンドが見つかりません」のメッセージは、以下のどれかを意味します。

- コマンドがそのシステムに存在しない
- コマンドのディレクトリが検索パスに存在しない

検索パスの問題を解決するには、コマンドが格納されているディレクトリのパス名を知る必要があります。

間違ったバージョンのコマンドが見つかってしまうのは、同じ名前のコマンドを持つディレクトリが検索パスにある場合です。この場合、正しいディレクトリが検索パスの後ろの方にあるか、まったく存在しない可能性があります。

現在の検索パスを表示するには、`echo $PATH` コマンドを使用します。

間違ったバージョンのコマンドを実行しているかどうかを調べるには、`type` コマンドを使用します。例:

```
$ type acroread
acroread is /usr/bin/acroread
```

▼ 検索パスの問題を診断して解決する方法

1. 現在の検索パスを表示して、コマンドが入っているディレクトリがユーザーのパス内に存在しない (あるいはスペルが間違っている) ことを確認します。

   ```
   $ echo $PATH
   ```

2. 次の点を確認します。

- 検索パスは正しいか
- 検索パスは、別バージョンのコマンドが見つかったほかの検索パスの前にリストされているか
- 検索パスのいずれかにコマンドが存在するか

パスを修正する必要がある場合は、手順 3 に進みます。修正する必要がない場合は、手順 4 に進みます。

3 次の表に示すように、適切なファイルでパスを追加します。

シェル	ファイル	構文	注意事項
bash および ksh93	$HOME/.profile	$ PATH=*$HOME/bin:/sbin:/usr/local/bin ...* $ export PATH	パス名はコロンで区切る

4 次のように、新しいパスを有効にします。

シェル	パスの場所	パスを有効にするコマンド
bash および ksh93	.profile	. $HOME/.profile
	.login	hostname$ source $HOME/.login

5 新しいパスを確認します。

$ **which** *command*

例 3–6 検索パスの問題を診断して解決する

この例は、type コマンドを使用して、mytool の実行可能ファイルが検索パス中のどのディレクトリにも存在しないことを示しています。

```
$ mytool
 -bash: mytool: command not found
 $ type mytool
 -bash: type: mytool: not found
 $ echo $PATH
 /usr/bin:
 $ vi $HOME/.profile
 (Add appropriate command directory to the search path)
 $ . $HOME/.profile
 $ mytool
```

コマンドを見つけることができなかった場合は、マニュアルページでそのディレクトリパスを調べます。

ファイルとグループの所有権の変更

ファイルとディレクトリの所有権は、誰かがスーパーユーザーとしてファイルを編集したために、変更されることが頻繁にあります。新しいユーザーのホームディレクトリを作成するときには、必ず、そのユーザーをホームディレクトリ内のドット (.) ファイルの所有者にしてください。ユーザーをドット (.) ファイルの所有者にしなかった場合、そのユーザーは自分のホームディレクトリにファイルを作成できません。

アクセスに関する問題は、グループの所有権が変更されたとき、またはユーザーの属するグループが `/etc/group` データベースから削除されたときにも発生します。

アクセスに問題のあるファイルのアクセス権または所有権を変更する方法については、『Oracle Solaris 11.1 の管理: セキュリティーサービス』の第 7 章「ファイルアクセスの制御 (タスク)」を参照してください。

ファイルアクセスの問題を解決する

以前はアクセスできていたファイルまたはディレクトリにアクセスできない場合は、そのファイルまたはディレクトリのアクセス権または所有権が変更されていることがあります。

ネットワークアクセスで発生する問題の把握

リモートコピーコマンド `rcp` を使用してネットワーク上でファイルをコピーするときに問題が発生した場合、リモートシステム上のディレクトリやファイルは、アクセス権の設定によりアクセスが制限されている可能性があります。他に考えられる問題の原因は、リモートシステムとローカルシステムがアクセスを許可するように構成されていないことです。

ネットワークアクセスに伴う問題、および AutoFS を通じたシステムへのアクセスでの問題については、『Oracle Solaris 11.1 でのネットワークファイルシステムの管理』の「NFS のトラブルシューティングの方法」を参照してください。

◆ ◆ ◆ 第 4 章

4 その他各種のシステムおよびソフトウェアのトラブルシューティング (タスク)

この章では、ときどき発生するが比較的修正しやすい、さまざまなシステムおよびソフトウェアの問題について説明します。トラブルシューティングのプロセスには、通常、特定のソフトウェアアプリケーションや内容に関連しない問題 (リブートの失敗やファイルシステムが一杯になるなど) の解決方法が含まれます。

この章の内容は次のとおりです:

リブートが失敗した場合の対処

システムがリブートに失敗した場合、またはシステムがリブートしたが再度クラッシュした場合は、システムのブートを妨害しているソフトウェアまたはハードウェアの障害があると考えられます。

システムがブートしない原因	問題の解決方法
システムが `/platform/‘uname -m‘/kernel/sparcv9/unix` を見つけられない。	SPARC システムの PROM 内の `boot-device` 設定を変更します。デフォルトのブートデバイスの変更については、『Oracle Solaris 11.1 システムのブートおよびシャットダウン』の「ブート属性の表示と設定」を参照してください。

システムがブートしない原因	問題の解決方法
Oracle Solaris ブートアーカイブが壊れた。または、SMF ブートアーカイブサービスが失敗した。`svcs -x` コマンドを実行すると、エラーメッセージが表示される。	プライマリブート環境のバックアップとなるセカンダリブート環境を作成します。プライマリブート環境をブートできなくなった場合は、バックアップのブート環境をブートします。あるいは、Live CD または USB メディアからブートすることもできます。
`/etc/passwd` ファイル内に無効なエントリが存在する。	無効な `passwd` ファイルからの回復については、『Oracle Solaris 11.1 システムのブートおよびシャットダウン』の「メディアからブートして、不明な root パスワードを解決する方法」を参照してください。
x86 ブートローダー (GRUB) が破損している。または、GRUB メニューが失われたか、壊れた。	破損した x86 ブートローダーまたは失われたか壊れた GRUB メニューからの回復については、『Oracle Solaris 11.1 システムのブートおよびシャットダウン』の「メディアからブートして、システムのブートを妨げている GRUB 構成の問題を解決する方法」を参照してください。
ディスクなどのデバイスに、ハードウェアの問題がある	ハードウェアの接続を確認します。 ■ 装置が接続されていることを確認します。 ■ すべてのスイッチが適切に設定されていることを確認します。 ■ すべてのコネクタおよびケーブル (Ethernet ケーブルも含む) を検査します。 ■ これらの手順がすべて失敗した場合は、システムの電源を切り、10 秒 - 20 秒ほど待って、もう一度電源を投入します。

上記のリストで問題が解決できない場合は、ご購入先にお問い合わせください。

root パスワードを忘れた場合や、システムのブートを妨害する問題が発生した場合の対処

`root` パスワードを忘れた場合や、システムのブートを妨害する別の問題が発生した場合は、次を実行します。

- システムを停止します。
- 『Oracle Solaris 11.1 システムのブートおよびシャットダウン』の「メディアからブートして、不明な root パスワードを解決する方法」の指示に従います。

- root パスワードが問題である場合は、/etc/shadow ファイルから root パスワードを削除します。
- システムをリブートします。
- ログインして root パスワードを設定します。

システムのハングが発生した場合の対処

ソフトウェアプロセスに問題がある場合、システムは完全にクラッシュせずに凍結、つまりハングすることがあります。ハングしたシステムから回復するには、次の手順に従ってください。

1. システムがウィンドウ環境を実行していたかどうかを調べて、次の推奨事項に従ってください。これらのリストで問題が解決できなかった場合は、手順 2 に進みます。
   - コマンドを入力しているウィンドウの中に、ポインタがあることを確認します。
   - 間違って Control-s キー (画面を凍結する) を押した場合は、Control-q キーを押します。Control-s キーはウィンドウだけを凍結し、画面全体は凍結しません。ウィンドウが凍結している場合は、他のウィンドウを試します。
   - 可能であれば、ネットワーク上の他のシステムからリモートでログインします。pgrep コマンドを使用して、ハングしているプロセスを見つけます。ウィンドウシステムがハングしている場合は、そのプロセスを特定して強制終了します。
2. Control-\ キーを押すことによって、強制的にプログラムの実行を終了し、(場合によっては) コアファイルを書き込みます。
3. Control-c キーを押すことによって、実行されている可能性があるプログラムを中断します。
4. リモートからログインして、システムをハングさせているプロセスを特定して強制終了します。
5. リモートからログインして、root 役割になってからシステムをリブートします。
6. システムがまだ応答しない場合は、強制的にクラッシュダンプしてリブートします。強制的なクラッシュダンプとブートの実行については、『Oracle Solaris 11.1 システムのブートおよびシャットダウン』の「クラッシュダンプを強制してシステムをリブートする」を参照してください。
7. システムがまだ応答しない場合は、電源を切ってから数分待ち、もう一度電源を入れます。
8. システムがまったく応答しない場合は、ご購入先にお問い合わせください。

ファイルシステムが一杯になった場合の対処

ルート (/) ファイルシステムや他のファイルシステムが一杯になると、次のようなメッセージがコンソールウィンドウに表示されます。

```
.... file system full
```

ファイルシステムが一杯になる原因はいくつかあります。次のセクションでは、一杯になったファイルシステムを回復する方法をいくつか説明します。

大規模ファイルまたはディレクトリを作成したために、ファイルシステムが一杯になる

エラーの原因	問題の解決方法
ファイルかディレクトリを間違った場所にコピーした。これは、アプリケーションがクラッシュして、大きなコアファイルをファイルシステムに書き込んだときにも発生する。	ログインして `root` 役割になり、特定のファイルシステムで `ls -tl` コマンドを使用して、新しく作成された大きなファイルを特定して削除します。

システムのメモリーが不足したために、**TMPFS** ファイルシステムが一杯になる

エラーの原因	問題の解決方法
これは、`tmpfs` に許可されているよりも多く書き込もうとした、または現在のプロセスがメモリーを多く使用している場合に発生する	`tmpfs` に関連するエラーメッセージから回復する方法については、`tmpfs(7FS)` のマニュアルページを参照してください。

コピーまたは復元後にファイルの ACL が消失した場合の対処

エラーの原因	問題の解決方法
ACL を持つファイルまたはディレクトリを /tmp ディレクトリにコピーしたり復元したりすると、ACL 属性が消失する。/tmp ディレクトリは、通常、一時ファイルシステムとしてマウントされ、ACL などの UFS ファイルシステム属性はサポートしない	代わりに、/var/tmp ディレクトリにファイルをコピーまたは復元する

索引